Leitfaden

Amazon Athena

Amazon Athena: Leitfaden

Table of Contents

Was ist Amazon Athena?

Amazon Athena ist ein interaktiver Abfrageservice, der die direkte Analyse von Daten in Amazon Simple Storage Service (Amazon S3) mit Standard-SQL erleichtert. Mit einigen Aktionen in der AWS Management Console können Sie Athena auf die in Amazon S3 gespeicherten Daten verweisen und Ad-hoc-Abfragen mithilfe von Standard-SQL ausführen, deren Ergebnisse Sie innerhalb von Sekunden erhalten.

Weitere Informationen finden Sie unter Erste Schritte.

Amazon Athena vereinfacht auch die interaktive Ausführung von Datenanalysen mit Apache Spark, ohne Ressourcen planen, konfigurieren oder verwalten zu müssen. Wenn Sie Apache-Spark-Anwendungen auf Athena ausführen, übermitteln Sie Spark-Code zur Verarbeitung und erhalten die Ergebnisse direkt. Verwenden Sie die vereinfachte Notebook-Erfahrung in der Amazon-Athena-Konsole, um Apache Spark-Anwendungen mit Python oder Athena-Notebooks-APIs zu entwickeln.

Weitere Informationen finden Sie unter Erste Schritte mit Apache Spark auf Amazon Athena.

Athena SQL und Apache Spark auf Amazon Athena sind Serverless, sodass keine Infrastruktur eingerichtet oder verwaltet werden muss und Sie nur für die von Ihnen ausgeführten Abfragen bezahlen. Athena skaliert automatisch und führt Abfragen parallel aus, sodass schnell Ergebnisse vorliegen, und zwar auch bei umfangreichen Datasets und komplexen Abfragen.

Themen

Wann sollte ich Athena verwenden?

Abfrageservices wie Amazon Athena, Data Warehouses wie Amazon Redshift und hochentwickelte Datenverarbeitungs-Frameworks wie Amazon EMR erfüllen alle unterschiedliche Anforderungen und Anwendungsfälle. Die folgende Anleitung kann Ihnen helfen, einen oder mehrere Services basierend auf Ihren Anforderungen auszuwählen.

Amazon Athena

Mit Athena können Sie in Amazon S3 gespeicherte unstrukturierte, semistrukturierte und strukturierte Daten analysieren. Beispiele hierfür sind CSV und JSON oder spaltenbasierte Datenformate wie Apache Parquet und Apache ORC. Mit Athena lassen sich Ad-hoc-Abfragen über ANSI SQL ausführen; dabei müssen die Daten weder aggregiert noch in Athena geladen werden.

Für die einfache Datenvisualisierung lässt sich Athena mit Amazon QuickSight integrieren. Sie können mit Athena Berichte generieren oder Daten mit Business-Intelligence-Tools oder SQL-Clients analysieren, die eine Verbindung über einen JDBC- oder ODBC-Treiber herstellen. Weitere Informationen finden Sie unter Was ist Amazon QuickSight? im Amazon-QuickSight-Benutzerhandbuch und Herstellen einer Verbindung mit Amazon Athena mit ODBC- und JDBC-Treibern.

Athena kann mit dem AWS Glue Data Catalog integriert werden, der einen dauerhaften Metadatenspeicher für Ihre Daten in Amazon S3 bietet. Auf diese Weise können Sie in Athena Tabellen erstellen und Daten abfragen, die auf einem zentralen Metadatenspeicher für Ihr Amazon-Web-Services-Konto basieren und mit den Datenermittlungs- und ETL-Features von AWS Glue nutzbar sind. Weitere Informationen finden Sie unter Integration mit AWS Glue und Was ist AWS Glue im AWS Glue-Entwicklerhandbuch.

Mit Amazon Athena können Sie ganz einfach interaktive Abfragen zu Daten direkt in Amazon S3 ausführen, ohne Daten formatieren oder Infrastruktur verwalten zu müssen. Zum Beispiel ist Athena nützlich, wenn Sie eine schnelle Abfrage für Webprotokolle ausführen möchten, um ein Leistungsproblem auf Ihrer Website zu beheben. Mit Athena können Sie schnell loslegen: Sie definieren einfach eine Tabelle für Ihre Daten und fragen mit Standard-SQL ab.

Sie sollten Amazon Athena verwenden, wenn Sie interaktive Ad-hoc-SQL-Abfragen für Daten auf Amazon S3 ausführen möchten, ohne eine Infrastruktur oder ein Cluster verwalten zu müssen. Amazon Athena bietet die einfachste Möglichkeit, Ad-hoc-Abfragen für Daten in Amazon S3 auszuführen, ohne dass Server eingerichtet oder verwaltet werden müssen.

Eine Liste der AWS-Services, die Athena nutzt bzw. mit denen es integriert werden kann, finden Sie unter the section called “AWS-Service Integrationen mit Athena”.

Amazon EMR

Mit Amazon EMR können Sie ganz einfach und günstig hochverteilte Verarbeitungs-Frameworks wie Hadoop, Spark und Presto im Vergleich zu On-Premises-Bereitstellungen ausführen. Amazon EMR ist flexibel – Sie können benutzerdefinierte Anwendungen und Code ausführen und spezifische

Datenverarbeitungs-, Speicher-, Speicherplatz- und Anwendungsparameter definieren, um Ihre Analyseanforderungen zu optimieren.

Neben der Ausführung von SQL-Abfragen kann Amazon EMR eine Vielzahl von Datenverarbeitungsaufgaben zur Aufskalierung für Anwendungen wie Machine Learning, Graph-Analytik, Datentransformation, Streaming-Daten und praktisch alles, was Sie programmieren können, ausführen. Sie sollten Amazon EMR verwenden, wenn Sie benutzerdefinierten Code verwenden, um extrem große Datensätze mit den neuesten Big-Data-Verarbeitungs-Frameworks wie Spark, Hadoop, Presto oder Hbase zu verarbeiten und zu analysieren. Amazon EMR gibt Ihnen die volle Kontrolle über die Konfiguration Ihrer Cluster und die darauf installierte Software.

Sie können Amazon Athena zum Abfragen von Daten verwenden, die Sie mit Amazon EMR verarbeiten. Amazon Athena unterstützt viele der gleichen Datenformate wie Amazon EMR. Athenas Datenkatalog ist kompatibel mit Hive-Metastore. Wenn Sie EMR verwenden und bereits über einen Hive-Metastore verfügen, können Sie Ihre DDL-Anweisungen bei Amazon Athena ausführen und Ihre Daten sofort abfragen, ohne Ihre Amazon-EMR-Aufträge zu beeinträchtigen.

Amazon Redshift

Ein Data Warehouse wie Amazon Redshift ist die beste Wahl, wenn Sie Daten aus vielen verschiedenen Quellen – wie Lagersystemen, Finanzsystemen und Einzelhandelsverkaufssystemen – in einem gemeinsamen Format zusammenfassen und für lange Zeiträume speichern müssen. Wenn Sie aus historischen Daten umfangreiche Geschäftsberichte erstellen möchten, ist ein Data Warehouse wie Amazon Redshift die beste Wahl. Die Abfrage-Engine in Amazon Redshift wurde optimiert, um bei der Ausführung komplexer Abfragen, die eine große Anzahl sehr großer Datenbanktabellen verbinden, besonders gut zu funktionieren. Wenn Sie Abfragen für hochstrukturierte Daten mit vielen Joins in vielen sehr großen Tabellen ausführen müssen, entscheiden Sie sich für Amazon Redshift.

Weitere Informationen darüber, wann Sie Athena verwenden sollten, finden Sie unter den folgenden Ressourcen:

- Entscheidungsleitfaden für Analyseservices in AWS im im Ressourcen-Center für die ersten Schritte
- Wann Athena im Vergleich zu anderen Big-Data-Services verwendet werden sollte, finden Sie in Häufig gestellte Fragen zu Amazon Athena
- Amazon Athena – Übersicht
- Amazon-Athena-Features

- Amazon Athena – Häufig gestellte Fragen
- Blog-Posts von Amazon Athena

AWS-Service Integrationen mit Athena

Sie können Athena verwenden, um Daten aus den in diesem Abschnitt AWS-Services aufgelisteten Daten abzufragen. Die Liste der unterstützten Regionen für jeden Service finden Sie unter Regionen und Endpunkte im Allgemeine Amazon Web Services-Referenz.

AWS-Services integriert mit 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
- Amazon S3 Inventory
- AWS Step Functions
- AWS Systems Manager -Bestand
- Amazon Virtual Private Cloud

Weitere Informationen zu jeder Integration finden Sie in folgenden Abschnitten.

AWS CloudFormation

Kapazitätsreservierung

Referenzthema: AWS: :Athena:: CapacityReservation im Benutzerhandbuch AWS CloudFormation

Gibt eine Kapazitätsreservierung mit dem angegebenen Namen und der Anzahl der angeforderten Datenverarbeitungseinheiten an. Weitere Informationen finden Sie Kapazität

zur Abfrageverarbeitung verwalten im Amazon Athena Athena-Benutzerhandbuch und CreateCapacityReservationin der Amazon Athena Athena-API-Referenz.

Datenkatalog

Referenzthema: AWS: :Athena:: DataCatalog im Benutzerhandbuch AWS CloudFormation

Geben Sie einen Athena-Datenkatalog an, einschließlich eines Namens, einer Beschreibung, eines Typs, Parametrierung und Tags. Weitere Informationen finden Sie Grundlegendes zu Tabellen, Datenbanken und zum Datenkatalog im Amazon Athena Athena-Benutzerhandbuch und CreateDataCatalogin der Amazon Athena Athena-API-Referenz.

Benannte Abfrage

Referenzthema: AWS: :Athena:: NamedQuery im Benutzerhandbuch AWS CloudFormation

Geben Sie benannte Abfragen mit Athena an AWS CloudFormation und führen Sie sie in Athena aus. Benannte Abfragen ermöglichen es Ihnen, einen Abfragenamen einer Abfrage zuzuordnen und ihn dann als gespeicherte Abfrage von der Athena-Konsole aus auszuführen. Weitere Informationen finden Sie Verwenden von gespeicherten Abfragen im Amazon Athena Athena-Benutzerhandbuch und CreateNamedQueryin der Amazon Athena Athena-API-Referenz.

Vorbereitete Anweisung

Referenzthema: AWS: :Athena:: PreparedStatement im Benutzerhandbuch AWS CloudFormation

Gibt eine vorbereitete Anweisung zur Verwendung mit SQL-Abfragen in Athena an. Eine vorbereitete Anweisung enthält Parameterplatzhalter, deren Werte zur Ausführungszeit angegeben werden. Weitere Informationen finden Sie Verwenden von parametrisierten Abfragen im Amazon Athena Athena-Benutzerhandbuch und CreatePreparedStatementin der Amazon Athena Athena-API-Referenz.

Arbeitsgruppe

Referenzthema: AWS: :Athena:: WorkGroup im Benutzerhandbuch AWS CloudFormation

Geben Sie Athena-Arbeitsgruppen mit an. AWS CloudFormation Verwenden Sie Athena-Arbeitsgruppen, um Abfragen für Sie oder Ihre Gruppe von anderen Abfragen im selben Konto zu isolieren. Weitere Informationen finden Sie Verwendung von Arbeitsgruppen zur Kontrolle des Abfragenzugriffs und der Kosten im Amazon Athena Athena-Benutzerhandbuch und CreateWorkGroupin der Amazon Athena Athena-API-Referenz.

Amazon CloudFront

Referenzthema: CloudFront Amazon-Logs abfragen

Verwenden Sie Athena, um CloudFront Amazon-Logs abzufragen. Weitere Informationen zur Verwendung CloudFront finden Sie im Amazon CloudFront Developer Guide.

AWS CloudTrail

Referenzthema: Abfragen von AWS CloudTrail-Protokollen

Die Verwendung von Athena mit CloudTrail Protokollen ist eine leistungsstarke Methode, um Ihre Analyse der AWS Serviceaktivitäten zu verbessern. Beispielsweise können Sie mithilfe von Abfragen Trends ermitteln und Vorgänge nach Attributen (z. B. Quell-IP-Adresse oder Benutzer) trennen. Sie können Tabellen für die Abfrage von Protokollen direkt von der CloudTrail Konsole aus erstellen und diese Tabellen verwenden, um Abfragen in Athena auszuführen. Weitere Informationen finden Sie unter Verwenden der CloudTrail Konsole zum Erstellen einer Athena-Tabelle für CloudTrail Protokolle .

Amazon DataZone

Referenzthema: Verwenden von Amazon DataZone in Athena

Verwenden Sie Amazon, DataZone um Daten in großem Umfang über Unternehmensgrenzen hinweg zu teilen, zu suchen und zu entdecken. DataZonevereinfacht Ihre Erfahrung mit AWS Analysediensten wie Athena, AWS Glue, und AWS Lake Formation. Wenn Sie über große Datenmengen in verschiedenen Datenquellen verfügen, können Sie Amazon verwenden, DataZone um Gruppen von Personen, Daten und Tools zu erstellen, die auf geschäftlichen Anwendungsfällen basieren.

In Athena können Sie den Abfrage-Editor verwenden, um auf DataZone Umgebungen zuzugreifen und diese abzufragen. Weitere Informationen finden Sie unter Verwenden von Amazon DataZone in Athena.

Elastic Load Balancing

Referenzthema: Abfragen von Application-Load-Balancer-Protokollen

Durch das Abfragen von Application Load Balancer-Protokollen können Sie die Datenverkehrsquelle, die Latenz und die Bytes anzeigen, die zwischen Elastic Load Balancing-Instances und Backend-Anwendungen übermittelt werden. Weitere Informationen finden Sie unter Abfragen von Application-Load-Balancer-Protokollen.

Referenzthema: Abfragen von Classic Load Balancer-Protokollen

Fragen Sie Classic-Load-Balancer-Protokollen ab, um Datenverkehrsmuster von und zu Elastic-Load-Balancing-Instances und Backend-Anwendungen analysieren und nachvollziehen. Dabei werden Ihnen die Datenverkehrsquelle, die Latenz und die übertragenen Bytes angezeigt. Weitere Informationen finden Sie unter Erstellen der Tabelle für ELB-Protokolle.

Amazon EMR Studio

Referenzthema: Verwenden des Amazon-Athena-SQL-Editors in EMR Studio

Sie können Athena in einem EMR Studio verwenden, um interaktive Abfragen zu entwickeln und auszuführen. Auf diese Weise können Sie EMR Studio für SQL-Analytik in Athena über dieselbe Amazon-EMR-Schnittstelle verwenden, die Sie für Ihre Spark-, Scala- und anderen Workloads verwenden. Mit der Athena-Integration in EMR Studio können Sie die folgenden Aufgaben ausführen:

- Athena-SQL-Abfragen ausführen
- Abfrageergebnisse anzeigen
- Abfrageverlauf anzeigen
- Gespeicherte Abfragen anzeigen
- Parametrisierten Abfragen durchführen
- Datenbanken, Tabellen und Ansichten für einen Datenkatalog anzeigen

Die folgenden Athena-Features sind in Amazon EMR Studio nicht verfügbar:

- Admin-Features wie das Erstellen oder Aktualisieren von Athena-Arbeitsgruppen, Datenquellen oder Kapazitätsreservierungen
- Athena-for-Spark- oder Spark-Notebooks
- DataZone Integration
- Step Functions

Die EMR Studio-Integration mit Athena ist überall verfügbar, AWS-Regionen wo EMR Studio und Athena verfügbar sind. Weitere Informationen zur Verwendung von Athena in EMR Studio finden Sie unter Verwenden des Amazon-Athena-SQL-Editors in EMR Studio im Amazon-EMR-Verwaltungshandbuch.

AWS Glue Data Catalog

Referenzthema: Integration in AWS Glue

Athena lässt sich in das integrieren AWS Glue Data Catalog, das einen persistenten Metadatenspeicher für Ihre Daten in Amazon S3 bietet. Auf diese Weise können Sie Tabellen

erstellen und Daten in Athena auf der Grundlage eines zentralen Metadatenspeichers abfragen, der in Ihrem gesamten Amazon Web Services Services-Konto verfügbar ist und in die ETL- und Datenerkennungsfunktionen von AWS Glue integriert ist. Weitere Informationen finden Sie unter Integration in AWS Glue Was ist AWS Glue? im Entwicklerhandbuch für AWS Glue .

AWS Identity and Access Management (IAM)

Referenzthema: Aktionen für Amazon Athena

Sie können Athena-API-Aktionen in IAM-Berechtigungsrichtlinien verwenden. Weitere Informationen finden Sie unter Aktionen für Amazon Athena und Identity and Access Management in Athena.

Amazon QuickSight

Referenzthema: Herstellen einer Verbindung mit Amazon Athena mit ODBC- und JDBC-Treibern

Athena lässt sich QuickSight für eine einfache Datenvisualisierung in Amazon integrieren. Sie können mit Athena Berichte generieren oder Daten mit Business-Intelligence-Tools oder SQL-Clients analysieren, die eine Verbindung über einen JDBC- oder ODBC-Treiber herstellen. Weitere Informationen zu Amazon QuickSight finden Sie unter Was ist Amazon QuickSight im QuickSight Amazon-Benutzerhandbuch. Informationen zur Verwendung von JDBC- und ODBC-Treibern mit Athena finden Sie unter Herstellen einer Verbindung mit Amazon Athena mit ODBC- und JDBC-Treibern.

Amazon S3 Inventory

Referenzthema: Bestandsabfrage mit Athena im Benutzerhandbuch von Amazon Simple Storage Service

Sie können Amazon Athena zum Abfragen von Amazon S3 Inventory mit Standard-SQL verwenden. Verwenden Sie Amazon S3 Inventory für die Prüfung und Meldung des Replikations- und Verschlüsselungsstatus Ihrer Objekte für Unternehmens-, Compliance- und regulatorische Anforderungen. Weitere Informationen finden Sie unter Amazon S3 Inventory im Benutzerhandbuch zu Amazon Simple Storage Service.

AWS Step Functions

Referenzthema: Athena mit Step Functions aufrufen im AWS Step Functions -Entwicklerhandbuch

Ruf Athena mit AWS Step Functions an. AWS Step Functions kann die Auswahl AWS-Services direkt in der Sprache der Amazon-Staaten steuern. Sie können Step Functions mit Athena verwenden, um die Abfrageausführung zu starten und zu stoppen, Abfrageergebnisse abzurufen,

Ad-hoc- oder geplante Datenabfragen auszuführen und Ergebnisse aus Data Lakes in Amazon S3 abzurufen. Die Step-Functions-Rolle muss über Berechtigungen zur Verwendung von Athena verfügen. Weitere Informationen finden Sie im AWS Step Functions -Entwicklerhandbuch.

Video: Orchestrieren Sie Amazon Athena Athena-Abfragen mit AWS Step Functions

Das folgende Video zeigt, wie Sie Amazon Athena verwenden und AWS Step Functions eine regelmäßig geplante Athena-Abfrage ausführen und einen entsprechenden Bericht generieren.

Orchestrieren Sie Amazon Athena Athena-Abfragen mit AWS Step Functions

Ein Beispiel, das Step Functions und Amazon EventBridge zur Orchestrierung AWS Glue DataBrew von Athena und Amazon verwendet QuickSight, finden Sie unter Orchestrieren eines AWS Glue DataBrew Jobs und Amazon Athena Athena-Abfrage mit AWS Step Functions im AWS Big Data-Blog.

AWS Systems Manager -Bestand

Referenzthema: Abfragen von Bestandsdaten aus mehreren Regionen und Konten im Benutzerhandbuch für AWS Systems Manager

AWS Systems Manager Inventory ist in Amazon Athena integriert, sodass Sie Inventardaten von mehreren AWS-Regionen AND-Konten abfragen können. Weitere Informationen finden Sie im AWS Systems Manager -Benutzerhandbuch.

Amazon Virtual Private Cloud

Referenzthema: Abfragen von Amazon-VPC-Flow-Protokollen

Amazon-Virtual-Private-Cloud-Flow-Protokolle erfassen Informationen zum IP-Datenverkehr zu und von Netzwerkschnittstellen in einer VPC. Fragen Sie die Protokolle in Athena zur Untersuchung von Netzwerkdatenverkehrsmustern und zur Identifizierung von Bedrohungen und Risiken in Ihrem Amazon-VPC-Netzwerk ab. Weitere Informationen zur Amazon VPC finden Sie im Amazon-VPC-Benutzerhandbuch.

Einrichtung

Wenn Sie bereits bei Amazon Web Services registriert sind, können Sie Amazon Athena sofort verwenden. Wenn Sie sich nicht für AWS angemeldet haben oder Hilfe beim Einstieg benötigen, führen Sie die folgenden Aufgaben aus.

So melden Sie sich für ein AWS-Konto an

Wenn Sie kein AWS-Konto haben, führen Sie die folgenden Schritte zum Erstellen durch.

Anmeldung für ein AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/signup.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für ein AWS-Konto anmelden, wird ein Root-Benutzer des AWS-Kontos erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

AWS sendet Ihnen eine Bestätigungs-E-Mail, sobald die Anmeldung abgeschlossen ist. Sie können jederzeit Ihre aktuelle Kontoaktivität anzeigen und Ihr Konto verwalten. Rufen Sie dazu https://aws.amazon.com/ auf und klicken Sie auf Mein Konto.

Einen Administratorbenutzer erstellen

Wenn Sie sich für AWS-Konto registriert haben, sichern Sie Root-Benutzer des AWS-Kontos, aktivieren Sie AWS IAM Identity Center erstellen Sie einen Administratorbenutzer, damit Sie nicht den Root-Benutzer für alltägliche Aufgaben verwenden.

Schützen Ihres Root-Benutzer des AWS-Kontos

1. Melden Sie sich bei AWS Management Console als Kontobesitzer an, indem Sie Stammbenutzer auswählen und Ihre AWS-Konto-E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-AnmeldungBenutzerhandbuch zu .
2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

Anweisungen dazu finden Sie unter Aktivieren eines virtuellen MFA-Geräts für den Root-Benutzer Ihres AWS-Konto (Konsole) im IAM-Benutzerhandbuch.

Erstellen eines Administratorbenutzers

1. IAM Identity Center aktivieren.

   Eine genaue Anleitung finden Sie unter Aktivierung von AWS IAM Identity Center im AWS IAM Identity Center-Benutzerhandbuch.
2. Gewähren Sie im IAM Identity Center einem Administratorbenutzer Administratorzugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie unter Benutzerzugriff mit der Standardeinstellung konfigurieren IAM-Identity-Center-Verzeichnis im AWS IAM Identity CenterBenutzerhandbuch.

Als Administratorbenutzer anmelden

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

  Hilfe bei der Anmeldung mit einem IAM-Identity-Center-Benutzer finden Sie unter Anmelden beim AWS-Zugangsportal im AWS-Anmeldung Benutzerhandbuch zu.

Erteilen programmgesteuerten Zugriffs

Benutzer benötigen programmgesteuerten Zugriff, wenn sie außerhalb der AWS Management Console mit AWS interagieren möchten. Die Vorgehensweise, um programmgesteuerten Zugriff zu gewähren, hängt davon ab, welcher Benutzertyp auf zugreift AWS.

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zur AWS CLI finden Sie unter Konfigurieren der AWS CLI für die Verwendung von AWS IAM Identity Center im AWS Command Line Interface-Benutzerhandbuch. • Informationen zu AWS-SDKs, Tools und AWS-APIs finden Sie unter IAM-Identity-Center-Authentifizierung im Referenzhandbuch zu AWS-SDKs und Tools.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Folgen Sie den Anweisungen unter Verwenden temporärer Anmeldeinformationen mit AWS-Ressourcen im IAM-Benutzerhandbuch.
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zur AWS CLI finden Sie unter Authentifizierung mit IAM-Benutzer-Anmeldeinformationen im AWS Command

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
		Line Interface-Benutzer handbuch. • Informationen zu AWS-SDKs und Tools finden Sie unter Authentifizierung mit langfristigen Anmeldeinformationen im Referenzhandbuch zu AWS-SDKs und Tools. • Informationen zu AWS-APIs finden Sie unter Verwalten von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch.

Anfügen von verwalteten Richtlinien für Athena

Von Athena verwaltete Richtlinien gewähren Berechtigungen zur Verwendung der Athena-Funktionen. Sie können diese verwalteten Richtlinien an eine oder mehrere IAM-Rollen anfügen, die Benutzer annehmen können, um Athena zu verwenden.

Eine IAM-Rolle ist eine IAM-Identität, die Sie in Ihrem Konto mit bestimmten Berechtigungen erstellen können. Eine IAM-Rolle ist einem IAM-Benutzer ähnlich, weil es sich um eine AWS-Identität mit Berechtigungsrichtlinien handelt, die festlegen, welche Aktionen die Identität in AWS ausführen kann und welche nicht. Eine Rolle ist jedoch nicht einer einzigen Person zugeordnet, sondern kann von allen Personen angenommen werden, die diese Rolle benötigen. Einer Rolle sind außerdem keine standardmäßigen, langfristigen Anmeldeinformationen (Passwörter oder Zugriffsschlüssel) zugeordnet. Wenn Sie eine Rolle übernehmen, erhalten Sie stattdessen temporäre Anmeldeinformationen für Ihre Rollensitzung.

Weitere Informationen über Rollen finden Sie unter IAM-Rollen und Erstellen von IAM-Rollen im IAM-Benutzerhandbuch.

Um eine Rolle zu erstellen, die Zugriff auf Athena gewährt, fügen Sie der Rolle von Athena verwaltete Richtlinien hinzu. Für Athena gibt es zwei verwaltete Richtlinien: `AmazonAthenaFullAccess` und `AWSQuicksightAthenaAccess`. Diese Richtlinien erteilen Athena die Berechtigungen, für Sie Abfragen in Amazon S3 auszuführen und die Abfrageergebnisse in einem eigenen Bucket zu speichern. Den Inhalt dieser Richtlinien für Athena finden Sie unter AWS verwaltete Richtlinien für Amazon Athena.

Schritte zum Anfügen der von Athena verwalteten Richtlinien an eine Rolle finden Sie unter Hinzufügen von IAM-Identitätsberechtigungen (Konsole) im IAM-Benutzerhandbuch. Fügen Sie der von Ihnen erstellen Rolle die verwalteten `AmazonAthenaFullAccess`- und `AWSQuicksightAthenaAccess`-Richtlinien hinzu.

> **Note**
>
> Möglicherweise sind für den Zugriff auf den zugrunde liegenden Datensatz in Amazon S3 zusätzliche Berechtigungen erforderlich. Wenn Sie nicht der Kontoeigentümer sind oder anderweitig eingeschränkten Zugriff auf einen Bucket haben, wenden Sie sich an den Bucket-Eigentümer. Dieser kann Ihnen den Zugriff mithilfe einer ressourcenbasierten Bucket-Richtlinie gewähren. Alternativ wenden Sie sich an Ihren Kontoadministrator, um den Zugriff mithilfe einer rollenbasierten Richtlinie zu gewähren. Weitere Informationen finden Sie unter Zugriff auf Amazon S3. Falls der Datensatz oder die Athena-Abfrageergebnisse verschlüsselt sind, benötigen Sie möglicherweise weitere Berechtigungen. Weitere Informationen finden Sie unter Verschlüsselung im Ruhezustand.

Zugriff auf Athena

Sie können über die AWS Management Console, eine JDBC- oder ODBC-Verbindung, die Athena-API, die Athena-CLI, das AWS SDK oder auf Athena zugreifen AWS Tools for Windows PowerShell.

- Informationen zu den ersten Schritten bei der Verwendung von Athena SQL mit der Konsole finden Sie unter Erste Schritte.
- Informationen zum Erstellen von Jupyter-kompatiblen Notebooks und Apache-Spark-Anwendungen, die Python verwenden, finden Sie unter Verwenden von Apache Spark in Amazon Athena.

- Informationen zur Verwendung von JDBC- oder ODBC-Treibern finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC und Herstellen einer Verbindung mit Amazon Athena mit ODBC.
- Informationen zur Verwendung der Athena-API finden Sie in der Amazon-Athena-API-Referenz.
- Zur Verwendung der CLI installieren Sie die AWS CLI und geben dann in die Befehlszeile `aws athena help` ein, um die verfügbaren Befehle anzuzeigen. Weitere Informationen zu den verfügbaren Befehlen finden Sie in der Amazon-Athena-Befehlszeilenreferenz.
- Informationen zur Verwendung der AWS SDK for Java 2.xfinden Sie im Athena-Abschnitt der APIAWS SDK for Java 2.x -Referenz , in den Athena-Java-V2-Beispielen auf GitHub.com und im AWS SDK for Java 2.x -Entwicklerhandbuch.
- Informationen zur Verwendung der AWS SDK for .NETfinden Sie im -`Amazon.Athena`Namespace in der APIAWS SDK for .NET -Referenz zu , in den .NET-Athena-Beispielen auf GitHub.com und im AWS SDK for .NET -Entwicklerhandbuch.
- Informationen zur Verwendung von AWS Tools for Windows PowerShellfinden Sie in der – AWS Tools for PowerShell Amazon Athena-Cmdlet-Referenz, auf der AWS Tools for PowerShell Portalseite und im AWS Tools for Windows PowerShell -Benutzerhandbuch.
- Weitere Informationen zu Athena-Service-Endpunkten, mit denen Sie programmgesteuert eine Verbindung herstellen können, finden Sie unter Amazon-Athena-Endpunkte und Kontingente im Allgemeine Amazon Web Services-Referenz.

Nutzung von Athena SQL

Sie können Athena SQL verwenden, um Ihre Daten direkt in Amazon S3 abzufragen, indem Sie den AWS Glue Data Catalog, einen externen Hive-Metastore oder Verbundabfragen verwenden, wobei Sie eine Vielzahl von vorgefertigten Konnektoren zu anderen Datenquellen nutzen können.

Sie können auch:

- Verbindung zu Business Intelligence Tools und anderen Anwendungen mithilfe der JDBC- und ODBC-Treiber von Athena herstellen.
- Abfragen von AWS-Serviceprotokollen durchführen.
- Abfragen von Apache-Iceberg-Tabellen durchführen, einschließlich Zeitreiseabfragen, und Apache-Hudi-Datensätzen.
- Abfragen von Geodaten durchführen.
- Abfrage mithilfe von Machine-Learning-Inferenz von Amazon SageMaker durchführen.
- Abfragen mit Ihren eigenen benutzerdefinierten Funktionen durchführen.
- Die Abfrageverarbeitung von stark partitionierten Tabellen beschleunigen und die Partitionsverwaltung mithilfe der Partitionsprojektion automatisieren.

Themen

- SQL-Referenz für Athena
- Athena-Fehlerbehebung
- Codebeispiele

Grundlegendes zu Tabellen, Datenbanken und zum Datenkatalog

In Athena fungieren Kataloge, Tabellen und Datenbanken als Container für Metadatendefinitionen, die das Schema für zugrunde liegende Quelldaten definieren.

Athena verwendet die folgenden Begriffe, um sich auf Hierarchien von Datenobjekten zu beziehen:

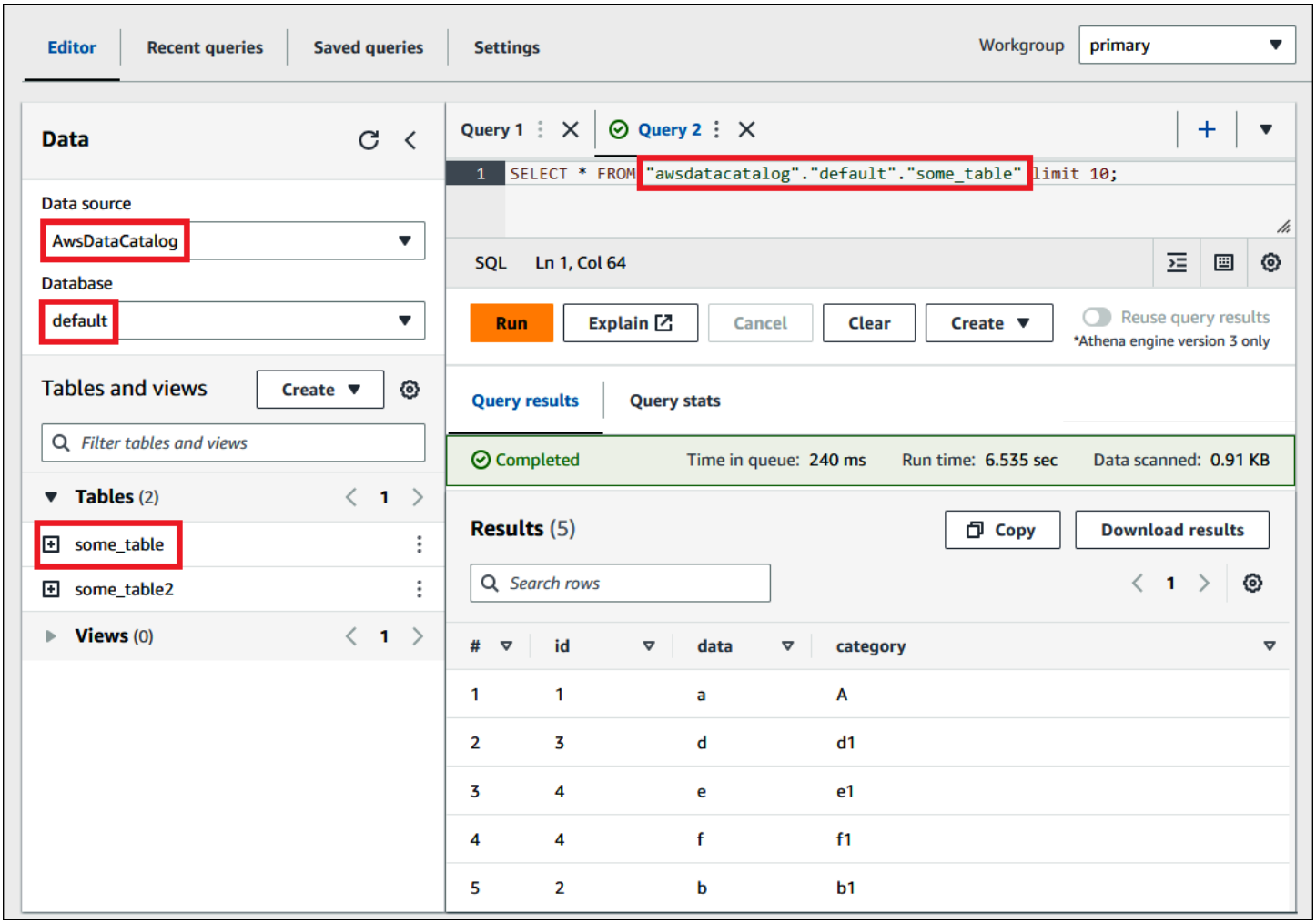
- Datenquelle – eine Gruppe von Datenbanken
- Datenbank – eine Gruppe von Tabellen
- Tabelle – Daten, die als Gruppe von Zeilen oder Spalten organisiert sind

Manchmal werden diese Objekte auch mit alternativen, aber gleichwertigen Namen bezeichnet, z. B. den folgenden:

- Eine Datenquelle wird manchmal auch als Katalog bezeichnet.
- Eine Datenbank wird manchmal auch als Schema bezeichnet.

> **Note**
>
> Diese Terminologie kann in den verbundenen Datenquellen, die Sie mit Athena verwenden, variieren. Weitere Informationen finden Sie unter Qualifizierer für Athena und verbundene Tabellennamen.

Die folgende Beispielabfrage in der Athena-Konsole verwendet die `awsdatacatalog`-Datenquelle, die `default`-Datenbank und die `some_table`-Tabelle.



Für jeden Datensatz muss in Athena eine Tabelle vorhanden sein. Aus den Metadaten in der Tabelle entnimmt Athena den Speicherort der Daten in Amazon S3 und die Struktur der Daten, z. B. Spaltennamen, Datentypen und Tabellenname. Bei Datenbanken handelt es sich um eine logische Gruppe von Tabellen; sie enthalten daher auch nur Metadaten und Schemainformationen für ein Dataset.

Für jedes Dataset, für das Sie eine Abfrage ausführen möchten, muss in Athena eine zugrunde liegende Tabelle vorhanden sein, die zum Abruf und zur Rückgabe der Abfrageergebnisse verwendet wird. Daher muss die Tabelle vor einer Datenabfrage in Athena registriert werden. Die Registrierung erfolgt bei der automatischen oder manuellen Tabellenerstellung.

Sie können mithilfe eines AWS Glue-Crawlers automatisch eine Tabelle erstellen. Weitere Informationen über AWS Glue und Crawler finden Sie unter Integration mit AWS Glue. Wird eine Tabelle in AWS Glue erstellt, erfolgt die Registrierung im eigenen AWS Glue-Datenkatalog. Athena

nutzt den AWS Glue-Datenkatalog zum Speichern und Abrufen der enthaltenen Metadaten, die bei der Abfrageausführung für die Analyse des zugrunde liegenden Datensatzes verwendet werden.

Unabhängig davon, wie die Tabellen erstellt werden, wird der Datensatz im Rahmen der Tabellenerstellung in Athena registriert. Diese Registrierung erfolgt im AWS Glue Data Catalog und ermöglicht Athena die Ausführung von Abfragen für die Daten. Im Athena-Abfrage-Editor wird auf diesen Katalog (oder diese Datenquelle) mit der Bezeichnung `AwsDataCatalog` verwiesen.

Nachdem Sie eine Tabelle erstellt haben, können Sie sie mit SQL-SELECT-Anweisungen abfragen, einschließlich des Abrufens bestimmter Dateispeicherorte für Ihre Quelldaten. Ihre Abfrageergebnisse werden in Amazon S3 an dem von Ihnen angegebenen Abfrageergebnisspeicherort gespeichert.

Der AWS Glue-Datenkatalog ist über Ihr Amazon-Web-Services-Konto zugänglich. Auch andere AWS-Services können den AWS Glue-Datenkatalog nutzen, sodass Sie alle in der Organisation erstellten Datenbanken und Tabellen in Athena sehen können – und umgekehrt.

- So erstellen Sie eine Tabelle manuell
  - Rufen Sie in der Athena-Konsole den Assistenten „Tabelle erstellen“ auf.
  - Schreiben Sie in der Athena-Konsole Hive-DDL-Anweisungen in den Abfrage-Editor.
  - Führen Sie mithilfe der Athena-API oder der CLI eine SQL-Abfragezeichenfolge mit DDL-Anweisungen aus.
  - Verwenden Sie den JDBC- oder ODBC-Treiber von Athena.

Wenn Sie Tabellen und Datenbanken manuell erstellen, verwendet Athena HiveQL-Data-Definition-Language (DDL)-Anweisungen wie `CREATE TABLE`, `CREATE DATABASE` und `DROP TABLE` im Hintergrund, um Tabellen und Datenbanken im AWS Glue Data Catalog zu erstellen.

Für den Einstieg können Sie ein Tutorial in der Athena-Konsole verwenden oder eine schrittweise Anleitung in der Athena-Dokumentation durcharbeiten.

- Um das Tutorial in der Athena-Konsole zu verwenden, wählen Sie das Informationssymbol oben rechts in der Konsole und dann die Registerkarte Tutorial.
- Eine Schritt-für-Schritt-Anleitung zum Erstellen einer Tabelle und zum Schreiben von Abfragen im Athena Query Editor finden Sie unter Erste Schritte.

Erste Schritte

In diesem Tutorial erfahren Sie, wie Sie Amazon Athena zum Abfragen von Daten verwenden. Sie werden eine Tabelle basierend auf Beispieldaten aus Amazon Simple Storage Service erstellen, die Tabelle abfragen und die Ergebnisse der Abfrage prüfen.

Für das Tutorial werden Live-Ressourcen verwendet, daher fallen für ausgeführte Abfragen Kosten an. Die Beispieldaten an dem in diesem Tutorial verwendeten Speicherort werden Ihnen nicht in Rechnung gestellt. Wenn Sie jedoch eigene Datendateien in Amazon S3 hochladen, fallen Gebühren an.

Voraussetzungen

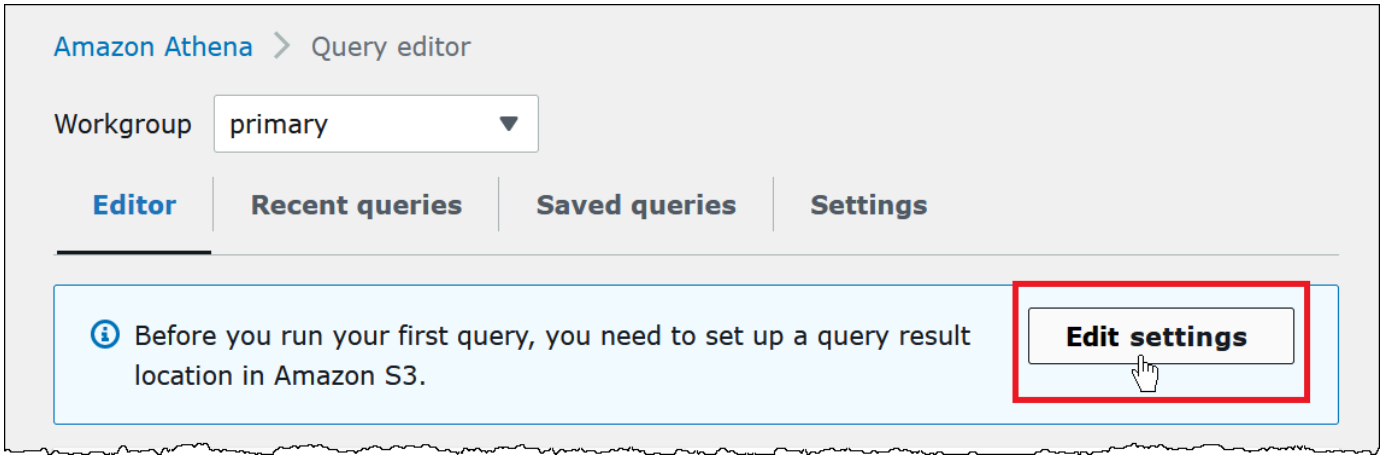
- Falls Sie das noch nicht getan haben, registrieren Sie sich für ein AWS-Konto.
- Erstellen Sie unter Verwendung derselben AWS-Region (z. B. USA West (Oregon)) und des Kontos, das Sie für Athena verwenden, einen Bucket in Amazon S3, um Ihre Athena-Abfrageergebnisse zu speichern. Sie konfigurieren diesen Bucket als Speicherort für die Abfrageausgabe.

Schritt 1: Erstellen einer Datenbank

Zunächst müssen Sie eine Datenbank in Athena erstellen.

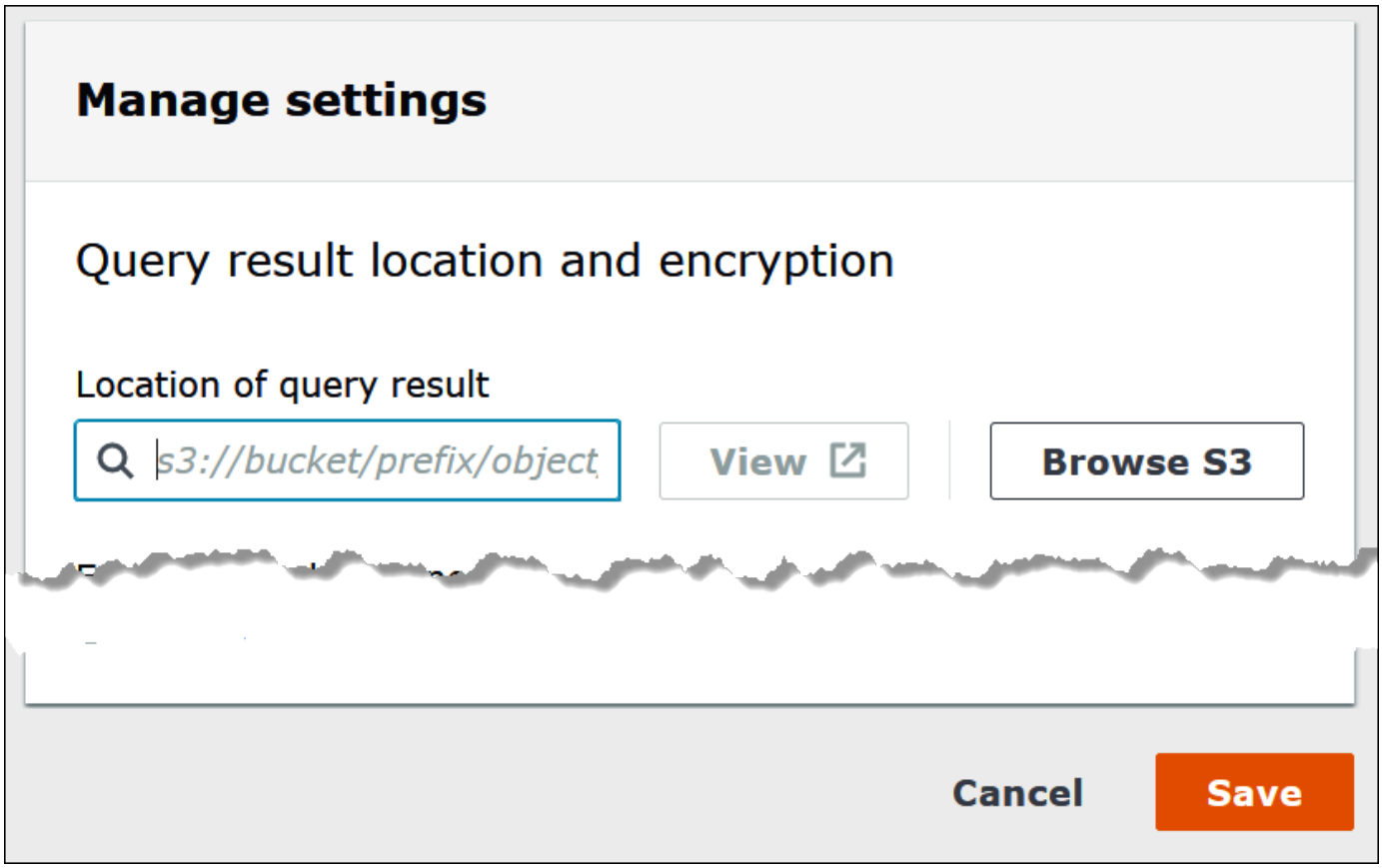
So erstellen Sie eine Athena-Datenbank

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn Sie die Athena-Konsole in Ihrer aktuellen AWS-Region zum ersten Mal besuchen, wählen Sie Entdecken des Abfrage-Editors, um den Abfrage-Editor zu öffnen. Andernfalls wird Athena im Abfrage-Editor geöffnet.
3. Wählen Sie Edit Settings (Einstellungen bearbeiten), um einen Speicherort für Abfrageergebnisse in Amazon S3 einzurichten.

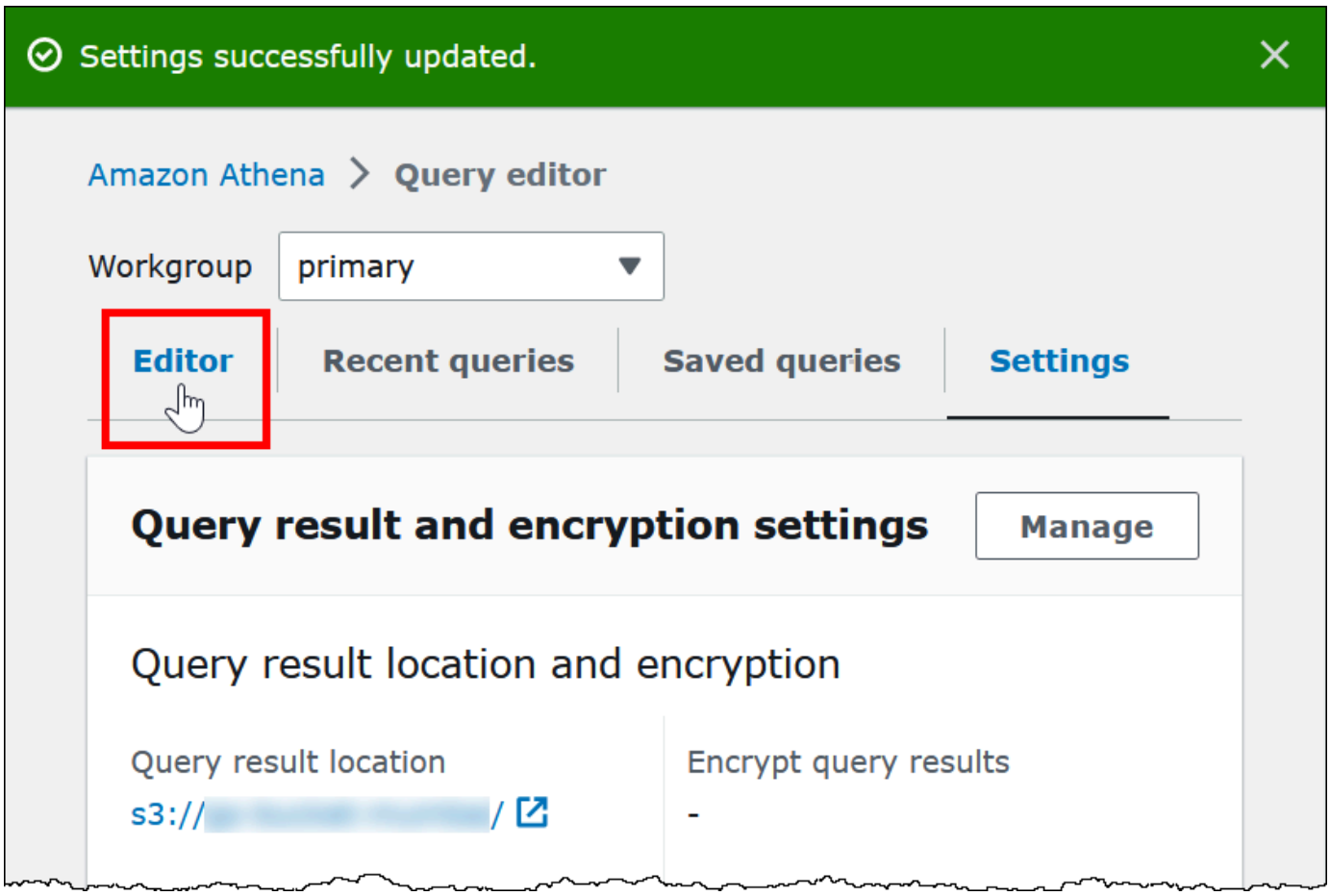


4. Führen Sie für Einstellungen verwalten einen der folgenden Schritte aus:

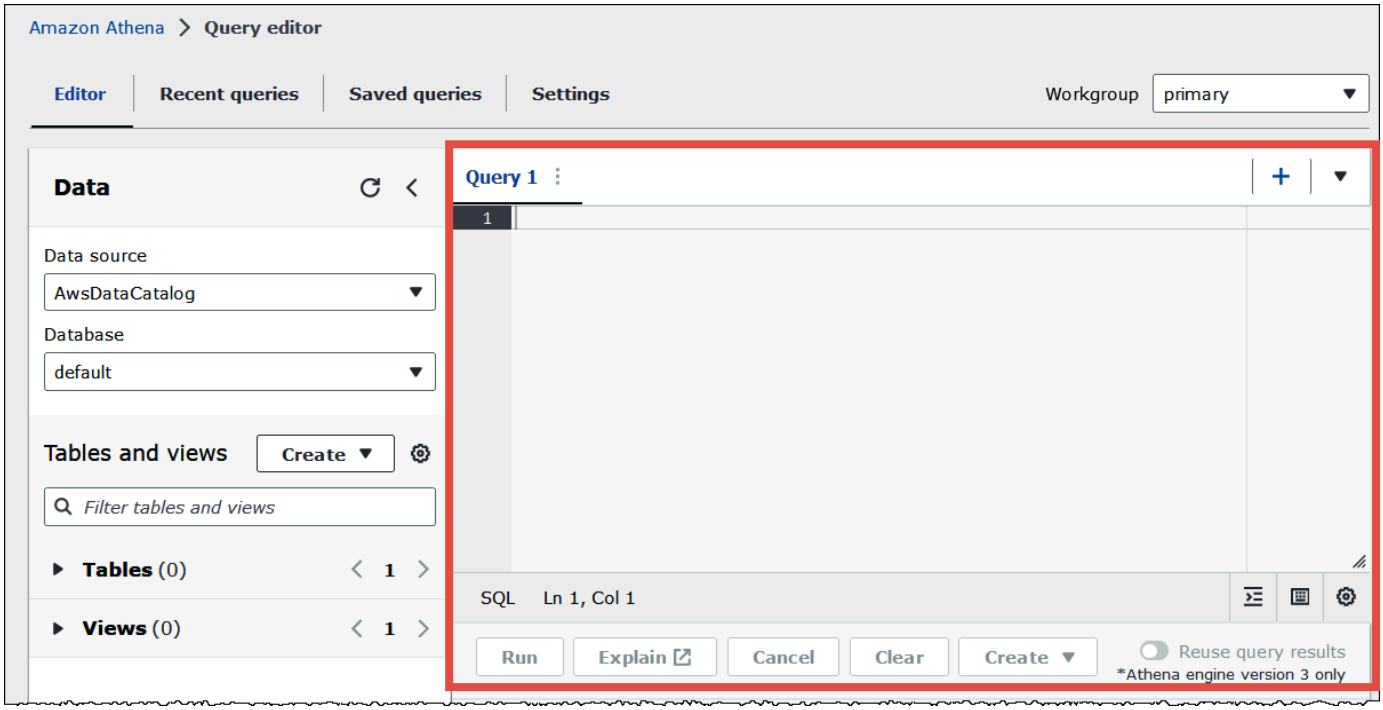
   - Geben Sie im Feld Speicherort der Abfrageergebnisse den Pfad zu dem Bucket ein, den Sie in Amazon S3 für Ihre Abfrageergebnisse erstellt haben. Stellen Sie dem Pfad einen Präfix mit `s3://` aus.
   - Wählen Sie das Symbol S3 durchsuchen, wählen Sie den Amazon-S3-Bucket aus, den Sie in Ihrer aktuellen Region erstellt haben und wählen Sie dann Auswählen.



5. Wählen Sie Save (Speichern) aus.
6. Wählen Sie Editor, um zum Abfrage-Editor zu wechseln.



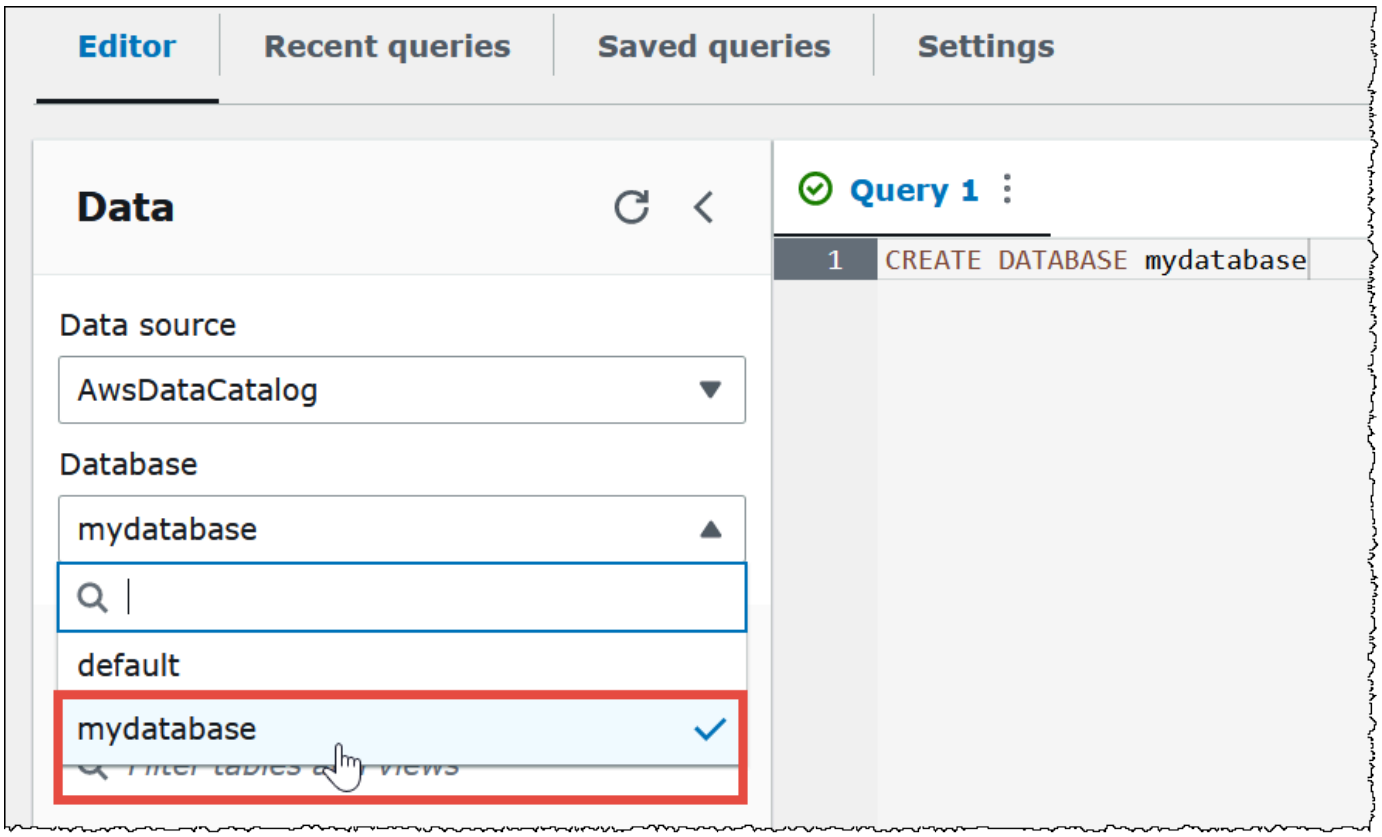
7. Auf der rechten Seite des Navigationsbereichs können Sie mit dem Athena-Abfrage-Editor Abfragen und Anweisungen eingeben und ausführen.



8. Um eine Datenbank mit dem Namen „`mydatabase`“ zu erstellen, geben Sie die folgende CREATE DATABASE-Anweisung ein.

```
CREATE DATABASE mydatabase
```

9. Wählen Sie Run (Ausführen) oder drücken Sie **`Ctrl+ENTER`**.
10. Wählen Sie aus der Database (Datenbank)-Liste auf der linken Seite `mydatabase` aus, um sie zu Ihrer aktuellen Datenbank zu machen.



Schritt 2: Erstellen einer Tabelle

Nachdem Sie nun eine Datenbank haben, können Sie eine Athena-Tabelle dafür erstellen. Die von Ihnen erstellte Tabelle basiert auf Amazon- CloudFront Protokolldaten am Speicherort `s3://athena-examples-`*`myregion`*`/cloudfront/plaintext/`, wobei *`myregion`* Ihr aktueller istAWS-Region.

Die Beispielprotokolldaten liegen im Format von tabulatorgetrennten Werten (TSV) vor, was bedeutet, dass ein Tabulatorzeichen als Trennzeichen verwendet wird, um die Felder zu trennen. Die Daten sollten wie das folgende Beispiel aussehen. Zur besseren Lesbarkeit wurden die Tabulatoren im Auszug in Leerzeichen umgewandelt und das letzte Feld gekürzt.

```
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[...]
```

Damit Athena diese Daten lesen kann, können Sie eine einfache `CREATE EXTERNAL TABLE` Anweisung wie die folgende erstellen. Die Anweisung, mit der die Tabelle erstellt wird, definiert Spalten, die den Daten zugeordnet werden, legt fest, wie die Daten getrennt werden, und gibt den Amazon-S3-Speicherort an, an dem die Beispieldaten enthalten sind. Beachten Sie, dass die `LOCATION`-Klausel einen Amazon S3-Ordnerspeicherort und keine bestimmte Datei angibt, da Athena erwartet, alle Dateien in einem Ordner zu scannen.

Verwenden Sie dieses Beispiel noch nicht, da es eine wichtige Einschränkung hat, die in Kürze erläutert wird.

```
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/';
```

Im Beispiel wird eine Tabelle mit dem Namen `cloudfront_logs` erstellt und für jedes Feld ein Name und ein Datentyp angegeben. Diese Felder werden zu den Spalten in der Tabelle. Da ein reserviertes Wort `date` ist, wird es mit Backtick-Zeichen (`) maskiert. `ROW FORMAT DELIMITED` bedeutet, dass Athena eine Standardbibliothek namens verwendet, LazySimpleSerDe um die eigentliche Arbeit des Parsens der Daten zu erledigen. Das Beispiel gibt auch an, dass die Felder durch Tabulatoren getrennt sind (`FIELDS TERMINATED BY '\t'`) und dass jeder Datensatz in der Datei mit einem Zeilenumbruchzeichen (`LINES TERMINATED BY '\n`) endet. Schließlich gibt die `LOCATION`-Klausel den Pfad in Amazon S3 an, in dem sich die tatsächlich zu lesenden Daten befinden.

Wenn Sie über Ihre eigenen tabulator- oder kommagetrennten Daten verfügen, können Sie eine `CREATE TABLE` Anweisung wie das soeben dargestellte Beispiel verwenden, sofern Ihre

Felder keine verschachtelten Informationen enthalten. Wenn Sie jedoch eine Spalte wie haben, `ClientInfo` die verschachtelte Informationen enthält, die ein anderes Trennzeichen verwenden, ist ein anderer Ansatz erforderlich.

Extrahieren von Daten aus dem ClientInfo Feld

Im Folgenden finden Sie ein vollständiges Beispiel für das letzte Feld `ClientInfo`:

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

Wie Sie sehen können, ist dieses Feld mehrwertig. Da die gerade dargestellte `CREATE TABLE` Beispielanweisung Registerkarten als Feldtrennzeichen angibt, können die einzelnen Komponenten innerhalb des `ClientInfo` Felds nicht in separate Spalten aufgeteilt werden. Daher ist eine neue `CREATE TABLE` Anweisung erforderlich.

Um Spalten aus den Werten innerhalb des `ClientInfo` Felds zu erstellen, können Sie einen regulären Ausdruck (Regex) verwenden, der Regex-Gruppen enthält. Die von Ihnen angegebenen Regex-Gruppen werden zu separaten Tabellenspalten. Um eine Regex in Ihrer `CREATE TABLE`-Anweisung zu verwenden, verwenden Sie eine Syntax wie die folgende. Diese Syntax weist Athena an, die Regex SerDe-Bibliothek und den von Ihnen angegebenen regulären Ausdruck zu verwenden.

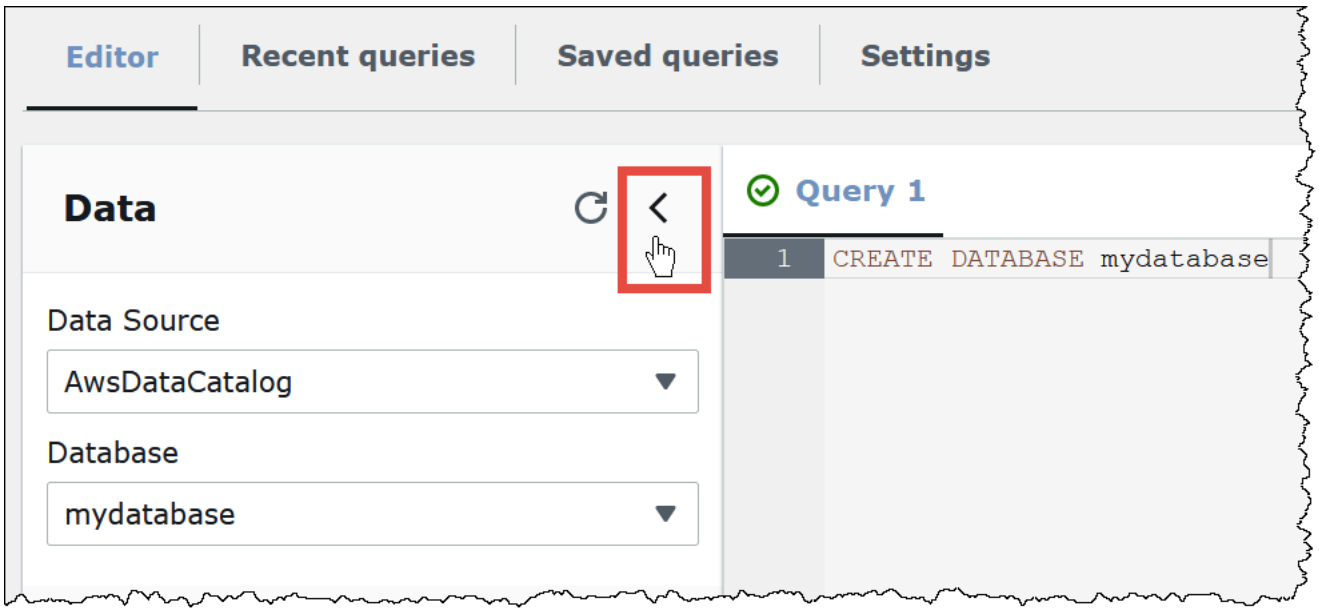
```
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
  WITH SERDEPROPERTIES ("input.regex" = "regular_expression")
```

Reguläre Ausdrücke können nützlich sein, um Tabellen aus komplexen CSV- oder TSV-Daten zu erstellen, können aber schwierig zu schreiben und zu warten sein. Glücklicherweise gibt es andere Bibliotheken, die Sie für Formate wie JSON, Parquet und ORC verwenden können. Weitere Informationen finden Sie unter Unterstützte SerDes- und Daten-Formate.

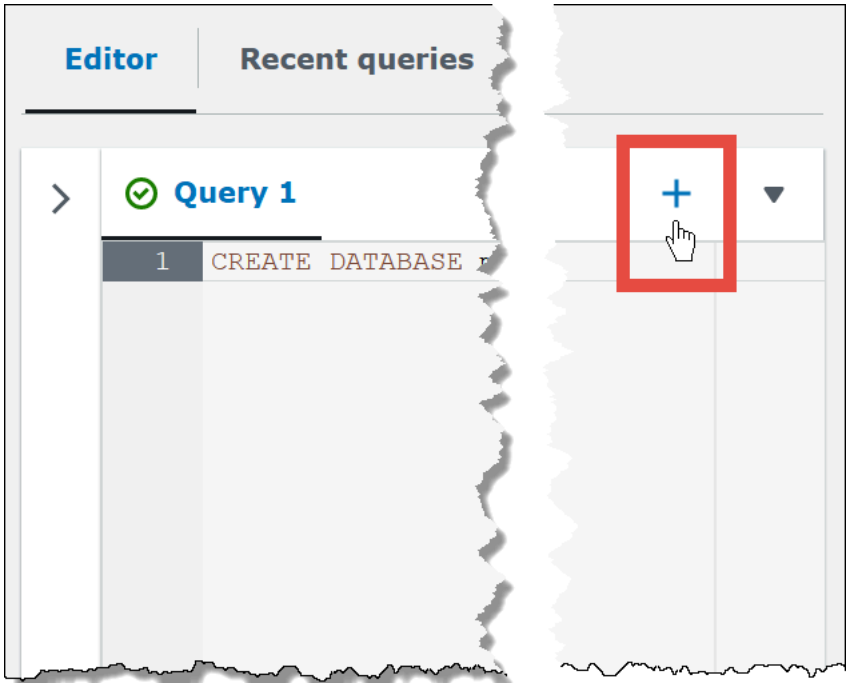
Jetzt können Sie die Tabelle im Athena-Abfrage-Editor erstellen. Die `CREATE TABLE`-Anweisung und Regex werden für Sie bereitgestellt.

So erstellen Sie eine Tabelle in Athena

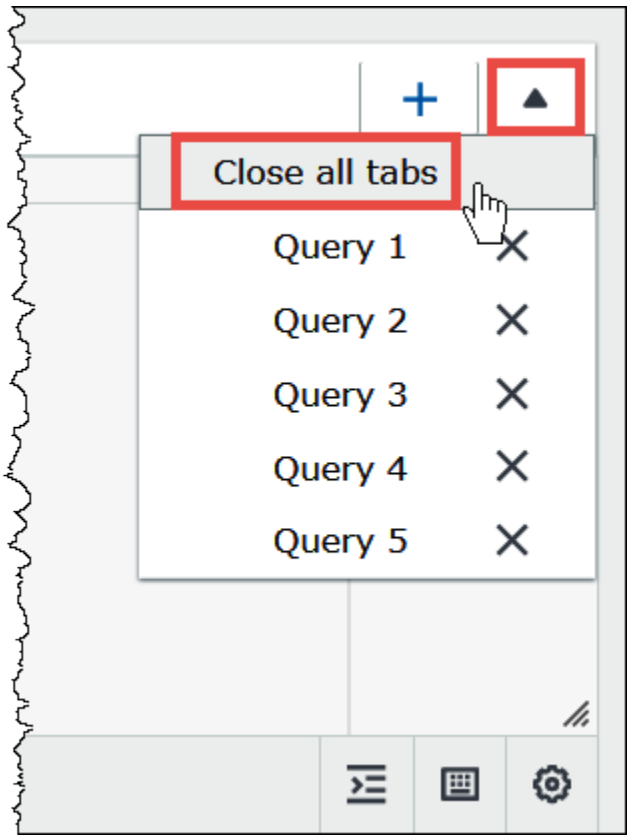
1. Stellen Sie im Navigationsbereich für Database (Datenbank) sicher, dass `mydatabase` ausgewählt ist.
2. Um mehr Platz im Abfrageeditor zu schaffen, können Sie das Pfeilsymbol auswählen, um den Navigationsbereich zu reduzieren.



3. Um eine Registerkarte für eine neue Abfrage zu erstellen, wählen Sie im Abfrageeditor das Pluszeichen (+). Sie können bis zu zehn Abfrageregisterkarten gleichzeitig öffnen.



4. Um eine oder mehrere Abfrage-Registerkarten zu schließen, wählen Sie den Pfeil neben dem Pluszeichen aus. Um alle Registerkarten gleichzeitig zu schließen, wählen Sie den Pfeil und wählen Sie dann Close all tabs (Schließen aller Tabs) aus.



5. Geben Sie im Abfragebereich die folgende `CREATE EXTERNAL TABLE`-Anweisung ein. Die Regex bricht die Informationen zu Betriebssystem, Browser und Browserversion aus dem `ClientInfo`-Feld in den Protokolldaten aus.

```
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/';
```

6. Ersetzen Sie in der LOCATION-Anweisung *myregion* durch die AWS-Region, die Sie zurzeit verwenden (z. B. `us-west-1`).
7. Wählen Sie Ausführen aus.

   Die Tabelle „`cloudfront_logs`“ wird erstellt und unter der Liste Tables (Tabellen) für die `mydatabase`-Datenbank angezeigt.

Schritt 3: Abfragen von Daten

Nachdem Sie nun die Tabelle `cloudfront_logs` in Athena basierend auf den Daten in Amazon S3 erstellt haben, können Sie SQL-Abfragen für die Tabelle ausführen und die Ergebnisse in Athena aufrufen. Weitere Informationen zur Verwendung von SQL in Athena finden Sie unter SQL-Referenz für Athena.

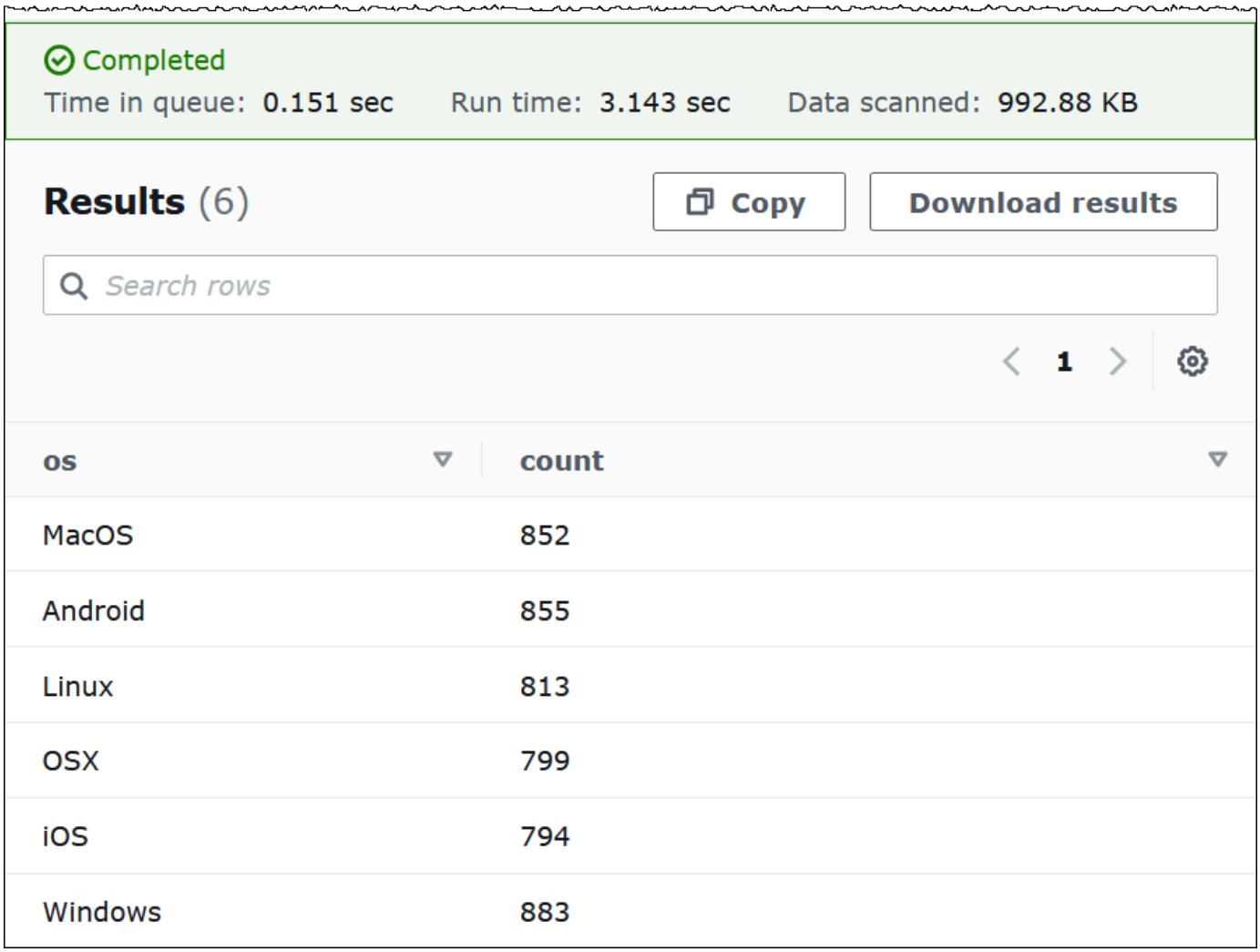
Eine Abfrage ausführen

1. Wählen Sie das Pluszeichen (+), um eine neue Abfrageregisterkarte zu öffnen, und geben Sie die folgende SQL-Anweisung in den Abfragebereich ein.

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

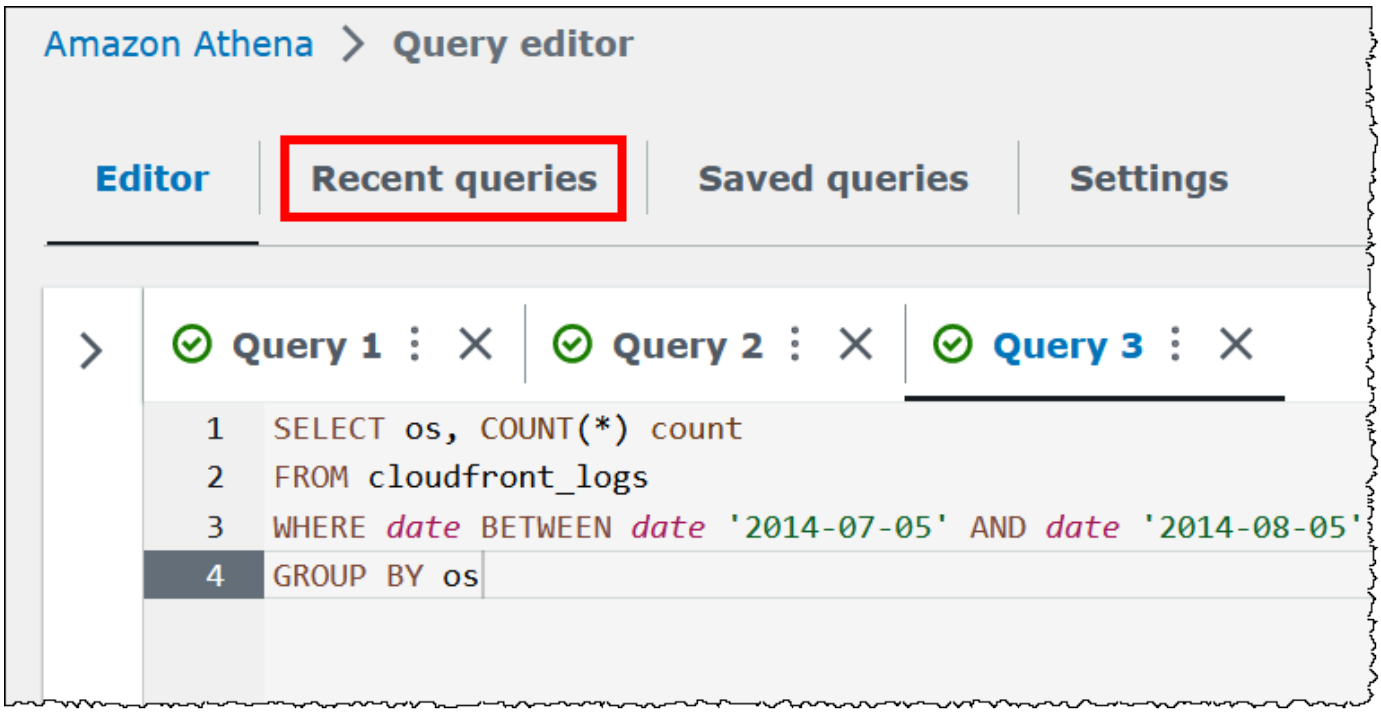
2. Wählen Sie Ausführen aus.

   Die Ergebnisse sehen wie folgt aus:

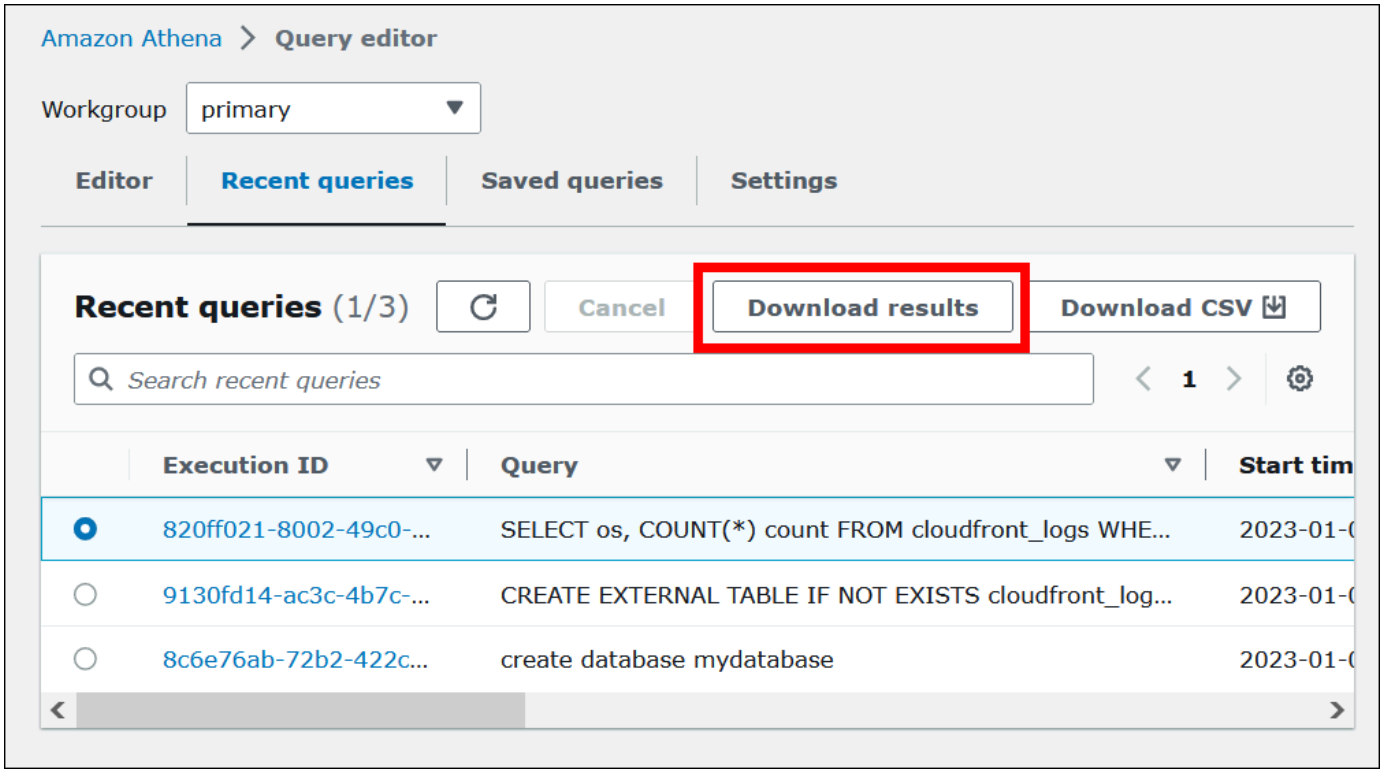


3. Um die Ergebnisse der Abfrage in einer `.csv`-Datei zu speichern, wählen Sie Ergebnisse herunterladen.

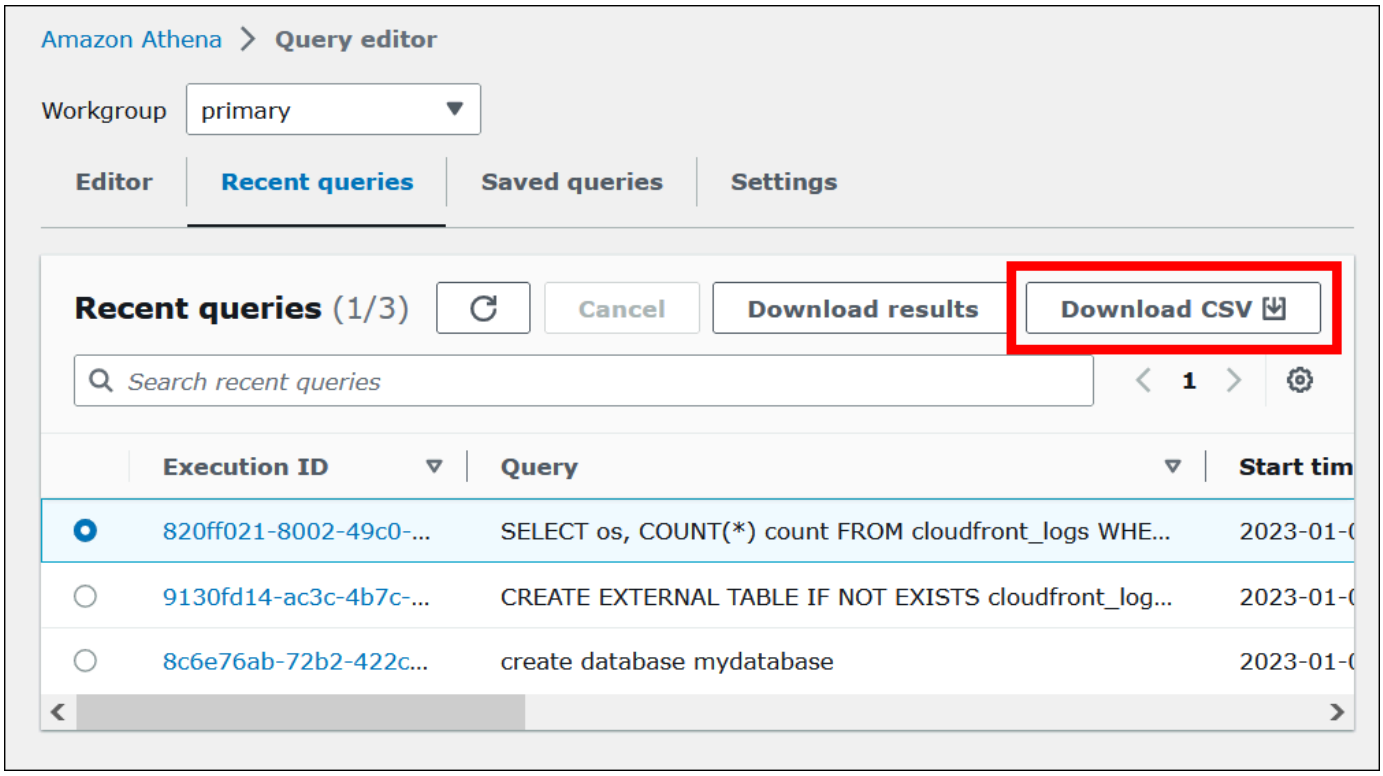
4. Um vorherige Abfragen anzuzeigen oder auszuführen, wählen Sie die Registerkarte Kürzliche Abfragen.



5. Um die Ergebnisse einer vorherigen Abfrage von der Registerkarte Kürzliche Abfragen herunterzuladen, wählen Sie die Abfrage aus, und wählen Sie dann Ergebnisse herunterladen. Abfragen werden 45 Tage lang aufbewahrt.



6. Um eine oder mehrere neuere SQL-Abfragezeichenfolgen in eine CSV-Datei herunterzuladen, wählen Sie Download CSV (CSV herunterladen).



Weitere Informationen finden Sie unter Arbeiten mit Abfrageergebnissen, letzten Abfragen und Ausgabedateien.

Speichern Ihrer Abfragen

Sie können die von Ihnen erstellten oder bearbeiteten Abfragen im Abfrage-Editor unter einem Namen speichern. Athena speichert diese Abfragen auf der Registerkarte Saved queries (Gespeicherte Abfragen). Sie können die Registerkarte Saved queries (Gespeicherte Abfragen) zum Abrufen, Ausführen, Umbenennen oder Löschen Ihrer gespeicherten Abfragen verwenden. Weitere Informationen finden Sie unter Verwenden von gespeicherten Abfragen.

Tastenkombinationen und Type-Ahead-Vorschläge

Der Athena-Abfrageeditor bietet zahlreiche Tastenkombinationen für Aktionen wie das Ausführen einer Abfrage, das Formatieren einer Abfrage, Zeilenoperationen sowie Suchen und Ersetzen. Weitere Informationen und eine vollständige Liste der Tastenkombinationen finden Sie unter Verbessern der Produktivität durch die Verwendung von Tastenkombinationen im Amazon-Athena-Abfrageeditor im AWS-Big-Data-Blog.

Der Athena-Abfrage-Editor unterstützt Typeahead-Code-Vorschläge für eine schnellere Abfrageerstellung. Um Sie beim Schreiben von SQL-Abfragen mit verbesserter Genauigkeit und höherer Effizienz zu unterstützen, bietet er die folgenden Features:

- Während der Eingabe werden in Echtzeit Vorschläge für Schlüsselwörter, lokale Variablen, Codefragmente und Katalogelemente angezeigt.
- Wenn Sie einen Datenbank- oder Tabellennamen gefolgt von einem Punkt eingeben, zeigt der Editor komfortabel eine Liste von Tabellen oder Spalten an, aus denen Sie wählen können.
- Wenn Sie den Mauszeiger über einen Codefragmentvorschlag bewegen, wird in einer Zusammenfassung ein kurzer Überblick über die Syntax und Verwendung des Codefragments angezeigt.
- Um die Lesbarkeit des Codes zu verbessern, wurden auch die Keywords und ihre Hervorhebungsregeln aktualisiert, sodass sie der neuesten Syntax von Trino und Hive entsprechen.

Dieses Feature ist standardmäßig aktiviert. Um das Feature zu aktivieren oder zu deaktivieren, verwenden Sie die Code-Editor-Einstellungen (Zahnradsymbol) unten rechts im Abfrage-Editor-Fenster.

Herstellen einer Verbindung mit anderen Datenquellen

In diesem Tutorial wurde eine Datenquelle in Amazon S3 im CSV-Format verwendet. Weitere Informationen zur Verwendung von Athena mit der AWS Glue finden Sie unter Verwenden von AWS Glue zum Herstellen einer Verbindung mit Datenquellen in Amazon S3. Sie können eine Verbindung zwischen Athena und einer Vielzahl von Datenquellen herstellen, indem Sie, ODBC- und JDBC-Treiber, externe Hive-Metastores und Athena-Datenquellen-Connectors verwenden. Weitere Informationen finden Sie unter Herstellen von Verbindungen mit Datenquellen.

Herstellen von Verbindungen mit Datenquellen

Sie können mittels Amazon Athena Daten abfragen, die in einem Datensatz an verschiedenen Speicherorten und in verschiedenen Formaten gespeichert sind. Dieser Datensatz kann im CSV-, JSON-, Avro-, Parquet- oder anderen Formaten vorliegen.

Die Tabellen und Datenbanken, die Sie in Athena zum Ausführen von Abfragen verwenden, basieren auf Metadaten. Metadaten sind Daten zu den Daten im Datensatz. Die Form, in der diese Metadaten den Datensatz beschreiben, wird Schema genannt. Beispielsweise stellen ein Tabellenname,

die Namen der Spalten der Tabelle und die Datentypen der einzelnen Spalten als Metadaten gespeicherte Schemas dar, die den zugrunde liegende Datensatz beschreiben. In Athena wird das System zum Organisieren von Metadaten als Datenkatalog oder Metastore bezeichnet. Die Kombination aus Datensatz und dem Datenkatalog, der diesen Datensatz beschreibt, wird als Datenquelle bezeichnet.

Die Art der Beziehung zwischen den Metadaten und dem zugrunde liegenden Datensatz ist vom Typ der Datenquelle abhängig, mit der Sie arbeiten. Relationale Datenquellen wie MySQL, PostgreSQL und SQL Server integrieren Metadaten eng mit dem Datensatz. Sehr häufig werden in diesen Systemen die Metadaten geschrieben, wenn die Daten geschrieben werden. Andere Datenquellen, beispielsweise mit Hive erstellte Datenquellen, ermöglichen Ihnen das Definieren von Metadaten, während der Datensatz gelesen wird. Der Datensatz kann in verschiedenen Formaten vorliegen, z. B. CSV, JSON, Parquet oder Avro.

Athena unterstützt den AWS Glue Data Catalog nativ. Der AWS Glue Data Catalog ist ein Datenkatalog, der auf anderen Datensätzen und Datenquellen wie Amazon S3, Amazon Redshift und Amazon DynamoDB aufbaut. Sie können Athena über Connectors mit anderen Datenquellen verbinden.

Themen

Integration in AWS Glue

AWS Glue ist ein vollständig verwalteter AWS-Service zum Extrahieren, Transformieren und Laden (ETL). Eine seiner wichtigsten Fähigkeiten besteht darin, Daten zu analysieren und zu kategorisieren. Sie können AWS Glue-Crawler automatisch die Datenbank und das Tabellenschema aus Ihren Daten in Amazon S3 abzuleiten und die zugehörigen Metadaten im AWS Glue Data Catalog speichern.

Athena verwendet das AWS Glue Data Catalog, um Tabellenmetadaten für die Amazon-S3-Daten in Ihrem Amazon-Web-Services-Konto zu speichern und abzurufen. Mithilfe der Tabellenmetadaten

kann die Athena-Abfrage-Engine wissen, wie die Daten, die Sie abfragen möchten, gefunden, gelesen und verarbeitet werden.

Zum Erstellen eines Datenbank- und Tabellenschemas im AWS Glue Data Catalog können Sie ein AWS Glue-Crawler aus Athena auf einer Datenquelle aus, oder Sie können Data-Definition-Language (DDL)-Abfragen direkt im Athena-Abfrage-Editor ausführen. Anschließend können Sie mithilfe des erstellten Datenbank- und Tabellenschemas DML-Abfragen (Data Manipulation) in Athena verwenden, um die Daten abzufragen.

Sie können eine AWS Glue Data Catalog von einem anderen Konto als Ihrem eigenen anmelden. Nachdem Sie die erforderlichen IAM-Berechtigungen für AWS Glue konfiguriert haben, können Sie Athena verwenden, um kontoübergreifende Abfragen auszuführen. Weitere Informationen finden Sie unter Kontoübergreifender Zugriff auf AWS Glue -Datenkataloge.

Weitere Informationen zu AWS Glue Data Catalog finden Sie unter Data Catalog und Crawler AWS Glue im AWS Glue-Entwicklerhandbuch.

Für AWS Glue gelten separate Gebühren. Weitere Informationen finden Sie unter AWS Glue Preise.

Themen

- Verwenden von AWS Glue zum Herstellen einer Verbindung mit Datenquellen in Amazon S3
- Registrieren eines AWS Glue Data Catalog von einem anderen Konto
- Bewährte Methoden bei der Verwendung von Athena mit AWS Glue
- AWS CLI verwenden, um eine AWS Glue-Datenbank und ihre Tabellen neu zu erstellen

Verwenden von AWS Glue zum Herstellen einer Verbindung mit Datenquellen in Amazon S3

Athena kann eine Verbindung mit den in Amazon S3 gespeicherten Daten herstellen, um mit AWS Glue Data Catalog Metadaten wie Tabellen- und Spaltennamen zu speichern. Nachdem die Verbindung hergestellt wurde, werden Ihre Datenbanken, Tabellen und Ansichten im Athena-Abfrage-Editor angezeigt.

Um Schemainformationen zur Verwendung durch AWS Glue zu definieren, können Sie einen AWS Glue-Crawler erstellen, um die Informationen automatisch abzurufen, oder Sie können eine Tabelle manuell hinzufügen und die Schemainformationen eingeben.

Erstellen eines AWS Glue-Crawlers

Sie können einen Crawler erstellen, indem Sie in der Athena-Konsole beginnen und dann die AWS Glue-Konsole in integrierter Weise verwenden. Wenn Sie den Crawler erstellen, geben Sie einen Datenspeicherort in Amazon S3 an, der gecrawlt werden soll.

So erstellen Sie einen Crawler in AWS Glue ausgehend von der Athena-Konsole

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wählen Sie im Abfrage-Editor neben Tables and views (Tabellen und Ansichten) Create (Erstellen) und danach AWS Glue-Crawler aus.
3. Führen Sie auf der AWS Glue-Konsolenseite Add crawler (Crawler hinzufügen) die Schritte zum Erstellen eines Crawlers aus. Weitere Informationen finden Sie unter Benutzen von AWS Glue-Crawler in diesem Leitfaden und Befüllen des AWS Glue Data Catalog im AWS Glue-Entwicklerhandbuch.

> **Note**
>
> Athena erkennt keine Ausschlussmuster, die Sie für einen AWS Glue-Crawler angeben. Wenn Sie beispielsweise über einen Amazon-S3-Bucket verfügen, der sowohl `.csv`- als auch `.json`-Dateien enthält und Sie die `.json`-Dateien vom Crawler ausschließen, fragt Athena beide Dateigruppen ab. Um dies zu vermeiden, platzieren Sie die Dateien, die Sie ausschließen möchten, an einem anderen Speicherort.

Hinzufügen einer Tabelle mit einem Formular

Das folgende Verfahren zeigt Ihnen, wie Sie die Athena-Konsole verwenden, um eine Tabelle mithilfe des Formulars Create Table From S3 bucket data (Erstellen einer Tabelle aus S3-Bucket-Daten) hinzuzufügen.

So fügen Sie eine Tabelle hinzu und geben Schemainformationen mithilfe eines Formulars ein

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wählen Sie im Abfrage-Editor neben Tables and views (Tabellen und Ansichten) Create (Erstellen) und danach S3 bucket data (S3-Bucket-Daten) aus.
3. Geben im Formular Create Table From S3 bucket data (Tabelle aus S3-Bucket-Daten erstellen) für Table name (Tabellenname) einen Namen für die Tabelle ein.

4. Wählen Sie für Database configuration (Datenbankkonfiguration) eine vorhandene Datenbank aus oder erstellen Sie eine neue.
5. Geben Sie unter Location of Input Data Set (Speicherort des Eingabedatensatzes) den Pfad in Amazon S3 dem Ordner an, der den zu verarbeitende Datensatz enthält. Fügen Sie keinen Dateinamen in den Pfad ein. Athena scannt alle Dateien in dem von Ihnen angegebenen Ordner. Wenn Ihre Daten bereits partitioniert sind (z. B.

   s3://*DOC-EXAMPLE-BUCKET*/logs/year=2004/month=12/day=11/), geben Sie nur den Basispfad ein (z. B. s3://*DOC-EXAMPLE-BUCKET*/logs/).
6. Wählen Sie für Data Format (Datenformat) eine der folgenden Optionen:

   - Wählen Sie als Table type (Tabellentyp) Apache Hive, Apache Iceberg oder Delta Lake aus. Athena verwendet den Tabellentyp Apache Hive als Standard. Informationen zum Abfragen von Apache-Iceberg-Tabellen in Athena finden Sie unter Apache-Iceberg-Tabellen verwenden. Informationen zur Verwendung von Delta-Lake-Tabellen in Athena finden Sie unter Delta-Lake-Tabellen von Linux Foundation abfragen.
   - Wählen Sie für File format (Dateiformat) das Datei- oder Protokollformat aus, in dem Ihre Daten vorliegen.
     - Geben Sie für die Option Textdatei mit benutzerdefinierten Trennzeichen einen Field terminator (Feldtrennzeichen) an (d. h. ein Spaltentrennzeichen). Optional können Sie ein Collection terminator (Sammlungsendzeichen) angeben, das das Ende eines Array-Typs markiert, oder ein Collection terminator (Sammlungsendzeichen), das das Ende eines Zuordnungsdatentyps markiert.
   - SerDe library (SerDe-Bibliothek) – Eine SerDe-Bibliothek (Serialisierer-Deserialisierer) analysiert ein bestimmtes Datenformat, sodass Athena eine Tabelle dafür erstellen kann. Für die meisten Formate wird eine Standard-SerDe-Bibliothek für Sie ausgewählt. Wählen Sie für die folgenden Formate eine Bibliothek entsprechend Ihren Anforderungen aus:
     - Apache Web Logs (Apache-Web-Protokolle) – Wählen Sie entweder die RegexSerDe- oder die GrokSerDe-Bibliothek. Geben Sie für RegexSerDe einen regulären Ausdruck im Feld Regex definition (Regex-Definition) an. Stellen Sie für GrokSerDe eine Reihe benannter regulärer Ausdrücke für die `input.format`-SerDe-Eigenschaft bereit. Benannte reguläre Ausdrücke sind einfacher zu lesen und zu verwalten als reguläre Ausdrücke. Weitere Informationen finden Sie unter Abfragen von Apache-Protokollen in Amazon S3 abfragen.
     - CSV – Wählen Sie LazySimpleSerDe aus, wenn Ihre durch Kommas getrennten Daten keine in doppelte Anführungszeichen eingeschlossenen Werte enthalten oder das `java.sql.Timestamp`-Format verwenden. Wählen Sie OpenCSVSerDe aus, wenn Ihre

Daten Anführungszeichen enthalten oder das numerische UNIX-Format für `TIMESTAMP` verwenden (z. B. `1564610311`). Weitere Informationen finden Sie unter LazySimpleSerDe für CSV- und TSV-Dateien sowie für benutzerdefinierte, durch Trennzeichen getrennte Dateien und OpenCSVSerDe für CSV-Verarbeitung.

- JSON – Wählen Sie entweder die OpenX- oder die Hive-JSON-SerDe-Bibliothek. Beide Formate erwarten, dass sich jedes JSON-Dokument in einer einzelnen Textzeile befindet und dass Felder nicht durch Zeilenumbruchzeichen getrennt werden. Der OpenX SerDe bietet einige zusätzliche Eigenschaften. Weitere Informationen zu diesen Eigenschaften finden Sie unter OpenX JSON SerDe. Weitere Informationen zum Hive SerDe finden Sie unter Hive JSON SerDe.

Weitere Informationen zur Verwendung von SerDe-Bibliotheken in Athena finden Sie unter Unterstützte SerDes- und Daten-Formate.

7. Für SerDe properties (SerDe-Eigenschaften) können Sie Eigenschaften und Werte entsprechend der von Ihnen verwendeten SerDe-Bibliothek und Ihren Anforderungen hinzufügen, bearbeiten oder entfernen.

   - Um eine SerDe-Eigenschaft hinzuzufügen, wählen Sie Add SerDe property (SerDe-Eigenschaft hinzufügen).
   - Geben Sie im Feld Name den Namen der Eigenschaft ein.
   - Geben Sie im Feld Value (Wert) einen Wert für die Eigenschaft ein.
   - Um eine SerDe-Eigenschaft zu entfernen, wählen Sie Remove (Entfernen).

8. Wählen oder bearbeiten Sie für Table properties (Tabelleneigenschaften) die Tabelleneigenschaften entsprechend Ihren Anforderungen.

   - Wählen Sie für Write compression (Schreibkomprimierung) eine Komprimierungsoption aus. Die Verfügbarkeit der Option Schreibkomprimierung und der verfügbaren Komprimierungsoptionen hängt vom Datenformat ab. Weitere Informationen finden Sie unter Athena-Komprimierungs-Support.
   - Wählen Sie für Encryption (Verschlüsselung) die Option Encrypted data set (Verschlüsselter Datensatz) aus, wenn die zugrunde liegenden Daten in Amazon S3 verschlüsselt sind. Diese Option setzt die `has_encrypted_data`-Tabelleneigenschaft in der `CREATE TABLE`-Anweisung auf wahr.

9. Geben Sie für Column details (Spaltendetails) die Namen und Datentypen der Spalten ein, die Sie der Tabelle hinzufügen möchten.

- Um mehrere Spalten einzeln hinzuzufügen, wählen Sie Add a column (Spalte hinzufügen).
- Um schnell weitere Spalten hinzuzufügen, wählen Sie Bulk add columns (Massenhinzufügung von Spalten). Geben Sie in das Textfeld eine durch Kommas getrennte Liste von Spalten im Format *column_name data_type*, *column_name data_type*[, ...] ein, und wählen Sie dann Add (Hinzufügen).

10. (Optional) Fügen Sie für Partition details (Details zur Partition) einen oder mehrere Spaltennamen und Datentypen hinzu. Die Partitionierung hält verwandte Daten basierend auf Spaltenwerten zusammen und kann dazu beitragen, die Menge der pro Abfrage gescannten Daten zu reduzieren. Weitere Informationen zur Partitionierung finden Sie unter Daten in Athena partitionieren.
11. (Optional) Für das Bucketing können Sie eine oder mehrere Spalten angeben, die Zeilen enthalten, die Sie gruppieren möchten, und diese Zeilen dann in mehrere Buckets einfügen. Auf diese Weise können Sie nur den Bucket abfragen, den Sie lesen möchten, wenn der Bucket-Spaltenwert angegeben ist.

    - Wählen Sie für Buckets eine oder mehrere Spalten aus, die eine große Anzahl eindeutiger Werte enthalten (z. B. einen Primärschlüssel) und die häufig zum Filtern der Daten in Ihren Abfragen verwendet werden.
    - Geben Sie im Feld Number of buckets (Anzahl der Buckets) eine Zahl ein, die zulässt, dass Dateien die optimale Größe haben. Weitere Informationen finden Sie unter Top 10 Tipps zur Leistungsoptimierung für Amazon Athena im AWS-Big-Data-Blog.
    - Um Ihre Bucket-Spalten anzugeben, verwendet die `CREATE TABLE`-Anweisung die folgende Syntax:

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

    > **Note**
    >
    > Die Bucketing-Option ist für die Iceberg-Tabellentypen nicht verfügbar.

12. Das Preview table query (Vorschau der Tabellenabfrage)-Feld zeigt die `CREATE TABLE`-Anweisung, die durch die Informationen generiert wird, die Sie in das Formular eingegeben haben. Die Vorschauanweisung kann nicht direkt bearbeitet werden. Um die Anweisung zu ändern, ändern Sie die Formularfelder über der Vorschau oder erstellen Sie die Anweisung direkt im Abfrageeditor, anstatt das Formular zu verwenden.

13. Wählen Sie Create table (Tabelle erstellen) aus, um die generierte Anweisung im Abfrage-Editor auszuführen und eine Tabelle zu erstellen.

Registrieren eines AWS Glue Data Catalog von einem anderen Konto

Sie können das kontoübergreifende AWS Glue-Katalogfeature von Athena verwenden, um einen AWS Glue-Katalog von einem anderen Konto als Ihrem eigenen zu registrieren. Nachdem Sie die erforderlichen IAM-Berechtigungen für AWS Glue konfiguriert und den Katalog als Athena-`DataCatalog`-Ressource registriert haben, können Sie Athena verwenden, um kontoübergreifende Abfragen auszuführen. Weitere Informationen zur Konfiguration der erforderlichen Berechtigungen finden Sie unter Kontoübergreifender Zugriff auf AWS Glue - Datenkataloge.

Das folgende Verfahren zeigt Ihnen, wie Sie mithilfe der Athena-Konsole ein AWS Glue Data Catalog in einem anderen Amazon-Web-Services-Konto als Ihrem eigenen als Datenquelle konfigurieren.

Anmelden eines AWS Glue Data Catalog von einem anderen Konto

1. Führen Sie die Schritte in Kontoübergreifender Zugriff auf AWS Glue -Datenkataloge aus, um sicherzustellen, dass Sie über Berechtigungen zum Abfragen des Datenkatalogs im anderen Konto verfügen.
2. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
3. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.

4. Wählen Sie Data sources (Datenquellen) aus.
5. Wählen Sie oben rechts Create data source (Datenquelle erstellen) aus.
6. Wählen Sie auf der Seite Choose data source (Datenquelle auswählen) für Data sources (Datenquellen) die Option S3 – AWS Glue Data Catalog und dann Next (Weiter) aus.
7. Wählen Sie auf der Seite Enter data source details (Details zur Datenquelle eingeben) im Abschnitt AWS Glue Data Catalog für Choose anAWS Glue Data Catalog (auswählen) die Option AWS Glue Data Catalog in another account (in einem anderen Konto) aus.
8. Geben Sie für Data source details (Datenquellen-Details) die folgenden Informationen ein:

   - Data source name (Datenquellenname) – Geben Sie den Namen ein, den Sie in Ihren SQL-Abfragen verwenden möchten, um auf den Datenkatalog im anderen Konto zu verweisen.
   - Beschreibung – (Optional) Geben Sie eine Beschreibung des Datenkatalogs im anderen Konto ein.
   - Katalog-ID – Geben Sie die 12-stellige Amazon-Web-Services-Konto-ID des Kontos ein, zu dem der Datenkatalog gehört. Die Amazon-Web-Services-Konto-ID ist die Katalog-ID.
9. (Optional) Geben Sie für Tags Schlüssel-Wert-Paare ein, die Sie mit der Datenquelle verknüpfen möchten. Weitere Informationen zu Tags erhalten Sie unter Markieren von Athena-Ressourcen.
10. Wählen Sie Next (Weiter).
11. Überprüfen Sie auf der Seite Review and create (Überprüfen und erstellen) die von Ihnen bereitgestellten Informationen, und wählen Sie dann Create data source (Datenquelle erstellen) aus. Die Seite Data source details (Datenquellen-Details) listet die Datenbanken und Tags für den von Ihnen registrierten Datenkatalog auf.
12. Wählen Sie Data sources (Datenquellen) aus. Der von Ihnen registrierte Datenkatalog wird in der Spalte Data source name (Datenquellen-Name) aufgeführt.
13. Um Informationen zum Datenkatalog anzuzeigen oder zu bearbeiten, wählen Sie den Katalog und dann Actions (Aktionen), Edit (Bearbeiten) aus.
14. Um den neuen Datenkatalog zu löschen, wählen Sie den Katalog und dann Actions (Aktionen), Delete (Löschen) aus.

Weitere Informationen finden Sie unter Abfrage von kontoübergreifenden AWS Glue Data Catalog mit Amazon Athena im AWS-Big-Data-Blog.

Bewährte Methoden bei der Verwendung von Athena mit AWS Glue

Wenn Sie Athena mit der verwenden AWS Glue Data Catalog, können Sie verwenden, AWS Glue um Datenbanken und Tabellen (Schema) zu erstellen, die in Athena abgefragt werden sollen, oder Sie können Athena verwenden, um Schemata zu erstellen und sie dann in AWS Glue und verwandten Services zu verwenden. In diesem Thema werden Überlegungen und bewährte Methoden für beide Methoden vorgestellt.

Im Hintergrund verwendet Athena Trino, um DML-Anweisungen zu verarbeiten, und Hive, um die DDL-Anweisungen zu verarbeiten, die das Schema erstellen und ändern. Bei diesen Technologien müssen einige Konventionen befolgt werden, damit Athena und gut AWS Glue zusammenarbeiten.

In diesem Thema

Datenbank-, Tabellen- und Spaltennamen

Wenn Sie ein Schema in erstellen, AWS Glue um es in Athena abzufragen, sollten Sie Folgendes berücksichtigen:

- Datenbanknamen dürfen nicht länger als 255 Zeichen sein.
- Tabellennamen dürfen nicht länger als 255 Zeichen sein.
- Spaltennamen dürfen nicht länger als 255 Zeichen sein.
- Für Datenbank-, Tabellen- und Spaltennamen dürfen nur Kleinbuchstaben, Zahlen und Unterstriche verwendet werden.

Weitere Informationen finden Sie unter Databases (Datenbanken) und Tables (Tabellen) im AWS Glue -Entwicklerhandbuch.

> **Note**
>
> Wenn Sie eine -AWS::Glue::Database AWS CloudFormation Vorlage verwenden, um eine -AWS Glue Datenbank zu erstellen und keinen Datenbanknamen anzugeben, generiert AWS Glue automatisch einen Datenbanknamen im Format *resource_name-random_string*, der nicht mit Athena kompatibel ist.

Sie können den AWS Glue Catalog Manager verwenden, um Spalten umzubenennen, aber keine Tabellennamen oder Datenbanknamen. Um diese Einschränkung zu umgehen, müssen Sie eine Definition der alten Datenbank verwenden, um eine Datenbank mit dem neuen Namen zu erstellen. Anschließend verwenden Sie Definitionen der Tabellen aus der alten Datenbank, um die Tabellen in der neuen Datenbank neu zu erstellen. Dazu können Sie die AWS CLI oder das AWS Glue SDK verwenden. Informationen zu den erforderlichen Schritten finden Sie unter AWS CLI verwenden, um eine AWS Glue-Datenbank und ihre Tabellen neu zu erstellen.

Verwenden von AWS Glue Crawlern

AWS Glue -Crawler helfen dabei, das Schema für Datensätze zu ermitteln und sie als Tabellen im AWS Glue Data Catalog zu registrieren. Die Crawler gehen Ihre Daten durch und bestimmen das Schema. Darüber hinaus können Crawler Partitionen erkennen und registrieren. Weitere Informationen finden Sie unter Definieren von Crawlern im AWS Glue -Entwicklerhandbuch. Tabellen aus Daten, die erfolgreich gecrawlt wurden, können von Athena abgefragt werden.

> **Note**
>
> Athena erkennt keine Ausschlussmuster, die Sie für einen AWS Glue -Crawler angeben. Wenn Sie beispielsweise über einen Amazon-S3-Bucket verfügen, der sowohl `.csv`- als auch `.json`-Dateien enthält und Sie die `.json`-Dateien vom Crawler ausschließen, fragt Athena beide Dateigruppen ab. Um dies zu vermeiden, platzieren Sie die Dateien, die Sie ausschließen möchten, an einem anderen Speicherort.

Planen eines Crawlers zur Synchronisierung des AWS Glue Data Catalog mit Amazon S3

AWS Glue -Crawler können so eingerichtet werden, dass sie nach einem Zeitplan oder nach Bedarf ausgeführt werden. Weitere Informationen finden Sie unter Zeitpläne für Aufträge und Crawler im AWS Glue -Entwicklerhandbuch.

Wenn Sie Daten haben, die zu einem festen Zeitpunkt für eine partitionierte Tabelle eingehen, können Sie einen AWS Glue -Crawler einrichten, der nach Zeitplan ausgeführt wird, um Tabellenpartitionen zu erkennen und zu aktualisieren. So müssen Sie keinen potenziell langen und aufwändigen `MSCK REPAIR`-Befehl oder manuell einen `ALTER TABLE ADD PARTITION`-Befehl ausführen. Weitere Informationen finden Sie unter Tabellenpartitionen im AWS Glue -Entwicklerhandbuch.

Verwenden mehrerer Datenquellen mit Crawlern

Wenn ein AWS Glue -Crawler Amazon S3 scannt und mehrere Verzeichnisse erkennt, verwendet er eine Heuristik, um festzustellen, wo sich das Stammverzeichnis für eine Tabelle in der Verzeichnisstruktur befindet und welche Verzeichnisse Partitionen für die Tabelle sind. In einigen Fällen, wenn in zwei oder mehr Verzeichnissen ein ähnliches Schema erkannt wird, kann es vorkommen, dass der Crawler diese als Partitionen statt als eigenständige Tabellen behandelt. Eine Möglichkeit sicherzustellen, dass der Crawler eigenständige Tabellen erkennt, besteht darin, das Stammverzeichnis jeder Tabelle als Datenspeicher für den Crawler hinzuzufügen.

Nachfolgend finden Sie ein Beispiel für Partitionen in Amazon S3:

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

Wenn das Schema für `table1` und ähnlich `table2` ist und eine einzelne Datenquelle `s3://bucket01/folder1/` in auf festgelegt ist AWS Glue, kann der Crawler eine einzelne Tabelle mit zwei Partitionsspalten erstellen: eine Partitionsspalte, die `table1` und enthält`table2`, und eine zweite Partitionsspalte, die `partition1` über enthält`partition5`.

Damit der AWS Glue Crawler zwei separate Tabellen erstellt, legen Sie den Crawler auf zwei Datenquellen fest, `s3://bucket01/folder1/table1/` und `s3://bucket01/folder1/table2`, wie im folgenden Verfahren gezeigt.

So fügen Sie einen S3-Datenspeicher zu einem vorhandenen Crawler in hinzu AWS Glue

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die - AWS Glue Konsole unter https://console.aws.amazon.com/glue/.
2. Wählen Sie im Navigationsbereich Crawlers (Crawler) aus.
3. Wählen Sie den Link zu Ihrem Crawler und wählen Sie dann Edit (Bearbeiten).
4. Für Schritt 2: Auswählen von Datenquellen und Klassifizierern wählen Sie Edit (Bearbeiten).
5. Wählen Sie für Data sources (Datenquellen) Add a data source (Datenquelle hinzufügen) aus.
6. Wählen Sie im Dialogfeld Add a data source (Datenquelle hinzufügen) für den S3 path (S3-Pfad) Browse (Durchsuchen).
7. Wählen Sie das Bucket aus, das Sie verwenden möchten, wählen Sie anschließend Choose (Auswählen).

   Die hinzugefügte Datenquelle wird in der Data sources-Liste (Datenquellenliste) erscheinen.
8. Wählen Sie Weiter aus.
9. Erstellen Sie auf der Seite Configure security settings (Sicherheitseinstellungen konfigurieren) eine IAM-Rolle für den Crawler und wählen Sie dann Next (Weiter).
10. Stellen Sie sicher, dass der S3-Pfad mit einem Schrägstrich endet, und wählen Sie dann Add an S3 data source (Hinzufügen einer S3-Datenquelle).
11. Wählen Sie auf der Seite Set output and scheduling (Ausgabe und Terminplanung festlegen) für die Output configuration (Ausgabe-Konfiguration) die Zieldatenbank.
12. Wählen Sie Weiter aus.
13. Überprüfen Sie auf der Seite Review and update (überprüfen und aktualisieren) die von Ihnen getroffenen Entscheidungen. Um einen Schritt zu bearbeiten, wählen Sie Edit (Bearbeiten).
14. Wählen Sie Aktualisieren.

Synchronisieren von Partitionsschemata zur Vermeidung von "HIVE_PARTITION_SCHEMA_MISMATCH"

Für jede Tabelle im AWS Glue Data Catalog, die Partitionsspalten enthält, wird das Schema auf Tabellenebene und für jede einzelne Partition innerhalb der Tabelle gespeichert. Das Schema für Partitionen wird von einem AWS Glue -Crawler basierend auf der Stichprobe der Daten gefüllt, die innerhalb der Partition gelesen werden. Weitere Informationen finden Sie unter Verwenden mehrerer Datenquellen mit Crawlern.

Wenn Athena eine Abfrage ausführt, werden das Schema der Tabelle und das Schema der für die Abfrage erforderlichen Partitionen geprüft. Dabei werden die Spaltendatentypen der Reihe nach verglichen und es wird sichergestellt, dass sie für überlappende Spalten übereinstimmen. So wird verhindert, dass es zu unerwartetem Hinzufügen oder Entfernen von Spalten in der Mitte der Tabelle kommt. Wenn Athena erkennt, dass das Schema einer Partition vom Schema der Tabelle abweicht, kann Athena die Abfrage nicht verarbeiten und bricht mit dem Fehler `HIVE_PARTITION_SCHEMA_MISMATCH` ab.

Es gibt mehrere Möglichkeiten, dieses Problem zu beheben. Wenn die Daten versehentlich hinzugefügt wurden, können Sie die Datendateien, die zu der Schemaabweichung geführt haben, entfernen, die Partition verwerfen und die Daten erneut durchsuchen. Sie können auch die einzelne Partition verwerfen und dann mit Athena `MSCK REPAIR` ausführen, um die Partition mit dem Schema der Tabelle neu zu erstellen. Diese zweite Option funktioniert jedoch nur, wenn Sie sicher sind, dass das angewendete Schema die Daten weiterhin korrekt ausliest.

Aktualisieren von Tabellenmetadaten

Nach einem Crawl weist der AWS Glue Crawler automatisch bestimmte Tabellenmetadaten zu, um sie mit anderen externen Technologien wie Apache Hive, Presto und Spark kompatibel zu machen. Es kann vorkommen, dass der Crawler dabei Metadateneigenschaften falsch zuweist. Korrigieren Sie die Eigenschaften in manuell, AWS Glue bevor Sie die Tabelle mit Athena abfragen. Weitere Informationen finden Sie unter Anzeigen und Bearbeiten von Tabellendetails im AWS Glue -Entwicklerhandbuch.

AWS Glue weist Metadaten möglicherweise falsch zu, wenn eine CSV-Datei Anführungszeichen um jedes Datenfeld enthält, sodass die `serializationLib` Eigenschaft falsch ist. Weitere Informationen finden Sie unter CSV-Daten in Anführungszeichen.

Arbeiten mit CSV-Dateien

CSV-Dateien enthalten gelegentlich Anführungszeichen um die Datenwerte der einzelnen Spalten. Außerdem können sie Überschriften enthalten, die nicht Bestandteil der zu analysierenden Daten sind. Wenn Sie verwenden AWS Glue , um ein Schema aus diesen Dateien zu erstellen, folgen Sie den Anweisungen in diesem Abschnitt.

CSV-Daten in Anführungszeichen

Sie könnten eine CSV-Datei verwenden, die Datenfelder in doppelte Anführungszeichen wie im folgenden Beispiel enthält:

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

Um eine Abfrage in Athena für eine Tabelle auszuführen, die aus einer CSV-Datei mit Werten in Anführungszeichen erstellt wurde, müssen Sie die Tabelleneigenschaften in ändern, AWS Glue um OpenCSVSerDe zu verwenden. Weitere Informationen über OpenCSV SerDe finden Sie unter OpenCSVSerDe für CSV-Verarbeitung.

So bearbeiten Sie Tabelleneigenschaften in der AWS Glue Konsole

1. Wählen Sie im Navigationsbereich der AWS Glue Konsole Tabellen aus.
2. Wählen Sie den Link für die Tabelle, die Sie bearbeiten möchten. Wählen Sie dann Action (Aktion), Edit table (Tabelle bearbeiten).
3. Auf der Seite Edit table (Tabelle bearbeiten) nehmen Sie die folgenden Änderungen vor:
   - Geben Sie für Serialization lib (Serialisierungsbibliothek) `org.apache.hadoop.hive.serde2.OpenCSVSerde` ein.
   - Geben Sie für Serde-Parameter die folgenden Werte für die Schlüssel `escapeChar`, `quoteChar` und `separatorChar` ein:
     - Geben Sie für `escapeChar` einen umgekehrten Schrägstrich (****) ein.
     - Geben Sie für `quoteChar` ein doppeltes Anführungszeichen (**"**) ein.
     - Geben Sie für `separatorChar` ein Komma (**,**) ein.
4. Wählen Sie Speichern.

Weitere Informationen finden Sie unter Anzeigen und Bearbeiten von Tabellendetails im AWS Glue - Entwicklerhandbuch.

Programmgesteuertes Aktualisieren von AWS Glue Tabelleneigenschaften

Sie können die AWS Glue UpdateTable -API-Operation oder den CLI-Befehl update-table verwenden, um den `SerDeInfo` Block in der Tabellendefinition zu ändern, wie im folgenden Beispiel-JSON.

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

CSV-Dateien mit Überschriften

Wenn Sie eine Tabelle in Athena mit einer `CREATE TABLE`-Anweisung definieren, können Sie die Tabelleneigenschaft `skip.header.line.count` verwenden, um Header in Ihren CSV-Daten zu ignorieren, wie im folgenden Beispiel.

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

Alternativ können Sie die CSV-Überschriften vorher entfernen, damit diese Informationen nicht in den Abfrageergebnissen von Athena enthalten sind. Eine Möglichkeit, dies zu erreichen, besteht darin, AWS Glue Aufträge zu verwenden, die Extract, Transform, Load (ETL)-Arbeiten ausführen. Sie können Skripte in AWS Glue mit einer Sprache schreiben, die eine Erweiterung des PySpark Python-Dialekts ist. Weitere Informationen finden Sie unter Autorisieren von Aufträgen in AWS Glue im AWS Glue Entwicklerhandbuch für .

Das folgende Beispiel zeigt eine -Funktion in einem AWS Glue -Skript`from_options`, das einen dynamischen Frame mit schreibt und die `writeHeader` Formatoption auf „false“ setzt, wodurch die Header-Informationen entfernt werden:

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

AWS Glue Indizierung und Filterung von Partitionen

Wenn Athena partitionierte Tabellen abfragt, ruft es die verfügbaren Tabellenpartitionen ab und filtert sie nach der für Ihre Abfrage relevanten Teilmenge. Wenn neue Daten und Partitionen hinzugefügt werden, ist mehr Zeit für die Verarbeitung der Partitionen erforderlich, und die Abfragelaufzeit kann sich erhöhen. Wenn Sie eine Tabelle mit einer großen Anzahl von Partitionen haben, die im Laufe der Zeit wächst, sollten Sie die AWS Glue -Partitionsindizierung und -filterung verwenden. Die Partitionsindizierung ermöglicht Athena, die Partitionsverarbeitung zu optimieren und die Abfrageleistung für stark partitionierte Tabellen zu verbessern. Das Einrichten der Partitionsfilterung in den Eigenschaften einer Tabelle ist ein zweistufiger Prozess:

1. Erstellen eines Partitionsindex in AWS Glue.
2. Aktivieren der Partitionsfilterung für die Tabelle.

Erstellen eines Partitionsindex

Schritte zum Erstellen eines Partitionsindex in AWS Gluefinden Sie unter Arbeiten mit Partitionsindizes im - AWS Glue Entwicklerhandbuch. Informationen zu den Einschränkungen für Partitionsindizes in AWS Gluefinden Sie im Abschnitt Informationen zu Partitionsindizes auf dieser Seite.

Aktivieren der Partitionsfilterung

Um die Partitionsfilterung für die Tabelle zu aktivieren, müssen Sie eine neue Tabelleneigenschaft in AWS Gluefestlegen. Schritte zum Festlegen von Tabelleneigenschaften in AWS Gluefinden Sie auf der Seite Einrichten der Partitionsprojektion. Wenn Sie die Tabellendetails in bearbeiten AWS Glue, fügen Sie dem Abschnitt Tabelleneigenschaften das folgende Schlüssel-Wert-Paar hinzu:

- Fügen Sie für Key (Schlüssel) `partition_filtering.enabled` hinzu
- Fügen Sie für Wert `true` hinzu

Sie können die Partitionsfilterung für diese Tabelle jederzeit deaktivieren, indem Sie den Wert `partition_filtering.enabled` auf `false` setzen.

Nachdem Sie die obigen Schritte ausgeführt haben, können Sie zur Athena-Konsole zurückkehren, um die Daten abzufragen.

Weitere Informationen zur Verwendung der Partitionsindizierung und -filterung finden Sie unter Verbessern der Abfrageleistung von Amazon Athena mithilfe von AWS Glue Data Catalog Partitionsindizes im AWS -Big-Data-Blog.

Arbeiten mit Geodaten

AWS Glue unterstützt nicht nativ Well-known Text (WKT), Well-Known Binary (WKB) oder andere PostGIS-Datentypen. Der AWS Glue Classifier analysiert Geodaten und klassifiziert sie mithilfe unterstützter Datentypen für das Format, z. B. `varchar` für CSV. Wie bei anderen AWS Glue Tabellen müssen Sie möglicherweise die Eigenschaften von Tabellen aktualisieren, die aus Geodaten erstellt wurden, damit Athena diese Datentypen unverändert analysieren kann. Weitere Informationen finden Sie unter Verwenden von AWS Glue Crawlern und Arbeiten mit CSV-Dateien. Athena kann einige Geodatentypen in AWS Glue Tabellen möglicherweise nicht analysieren. Weitere Informationen zum Arbeiten mit Geodaten in Athena finden Sie unter Abfragen von koordinatenbasierten Daten.

Verwenden von AWS Glue Aufträgen für ETL mit Athena

AWS Glue -Aufträge führen ETL-Operationen aus. Ein - AWS Glue Auftrag führt ein Skript aus, das Daten aus Quellen extrahiert, die Daten transformiert und in Ziele lädt. Weitere Informationen finden Sie unter Erstellen von Aufträgen in AWS Glue im AWS Glue Entwicklerhandbuch für .

Erstellen von Tabellen mit Athena für AWS Glue ETL-Aufträge

Innerhalb von Athena erstellte Tabellen benötigen eine Tabelleneigenschaft namens `classification`, über die das Format der Daten identifiziert wird. Auf diese Weise kann die Tabellen für ETL-Aufträge AWS Glue verwenden. Die Klassifizierungswerte können `avro`, `csv`, `json`, `orc`, `parquet` oder `xml` sein. Es folgt ein Beispiel für eine `CREATE TABLE`-Anweisung in Athena:

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

Wenn die Tabelleneigenschaft beim Erstellen der Tabelle nicht hinzugefügt wurde, können Sie sie mithilfe der AWS Glue Konsole hinzufügen.

So fügen Sie die Eigenschaft der Klassifikationstabelle mithilfe der AWS Glue Konsole hinzu

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die - AWS Glue Konsole unter https://console.aws.amazon.com/glue/.
2. Wählen Sie im Navigationsbereich der Konsole Tables (Tabellen) aus.
3. Wählen Sie den Link für die Tabelle, die Sie bearbeiten möchten. Wählen Sie dann Action (Aktion), Edit table (Tabelle bearbeiten).
4. Scrollen Sie nach unten zum Abschnitt Table properties (Tabelleneigenschaften).
5. Wählen Sie Hinzufügen aus.
6. Geben Sie für Key (Schlüssel) **`classification`** ein.
7. Für Value (Wert), geben Sie einen Datentyp ein (z. B. **`json`**).
8. Wählen Sie Speichern.

   Im Abschnitt Table details (Angaben zur Tabelle) erscheint der von Ihnen eingegebene Datentyp im Feld Classification (Klassifizierung) der Tabelle.

Weitere Informationen finden Sie unter Working with Tables (Arbeiten mit Tabellen) im AWS Glue - Entwicklerhandbuch.

Verwenden von ETL-Aufträgen zur Optimierung der Abfrageleistung

AWS Glue -Aufträge können Ihnen helfen, Daten in ein Format umzuwandeln, das die Abfrageleistung in Athena optimiert. Datenformate wirken sich erheblich auf die Abfrageleistung und Abfragekosten in Athena aus.

Wir empfehlen, Parquet- und ORC-Datenformate zu verwenden. AWS Glue unterstützt das Schreiben in beide Datenformate, was es Ihnen einfacher und schneller macht, Daten in ein optimales Format für Athena umzuwandeln. Weitere Informationen zu diesen Formaten und anderen Möglichkeiten zur Verbesserung der Leistung finden Sie unter Top 10 Tipps zur Leistungsoptimierung für Amazon Athena.

Konvertieren der Datentypen SMALLINT und TINYINT in INT bei der Konvertierung in ORC

Um die Wahrscheinlichkeit zu verringern, dass Athena die von einem AWS Glue ETL-Auftrag erzeugten `TINYINT` Datentypen `SMALLINT` und nicht lesen kann, konvertieren Sie `SMALLINT` und `TINYINT` in , `INT` wenn Sie den Assistenten verwenden oder ein Skript für einen ETL-Auftrag schreiben.

Automatisieren von AWS Glue Aufträgen für ETL

Sie können AWS Glue ETL-Aufträge so konfigurieren, dass sie automatisch basierend auf Auslösern ausgeführt werden. Diese Funktion ist ideal, wenn Daten von außerhalb in einem suboptimalen Format für Abfragen in Athena in einen Amazon S3-Bucket übertragen AWS werden. Weitere Informationen finden Sie unter Auslösen von AWS Glue -Aufträgen im AWS Glue - Entwicklerhandbuch.

AWS CLI verwenden, um eine AWS Glue-Datenbank und ihre Tabellen neu zu erstellen

Das direkte Umbenennen einer AWS Glue-Datenbank ist nicht möglich, aber Sie können ihre Definition kopieren, die Definition ändern und die Definition verwenden, um die Datenbank mit einem anderen Namen neu zu erstellen. Ebenso können Sie die Definitionen der Tabellen in der alten Datenbank kopieren, die Definitionen ändern und die geänderten Definitionen verwenden, um die Tabellen in der neuen Datenbank neu zu erstellen.

> **Note**
>
> Die vorgestellte Methode kopiert keine Tabellenpartitionierung.

Bei dem folgenden Verfahren für Windows wird davon ausgegangen, dass Ihr AWS CLI für die JSON-Ausgabe konfiguriert ist. Um das Standardausgabeformat in der AWS CLI zu ändern, führen Sie den Befehl `aws configure` aus.

Wie Sie eine AWS Glue-Datenbank mit der AWS CLI kopieren

1. Führen Sie in der Eingabeaufforderung den folgenden AWS CLI-Befehl aus, um die Definition der AWS Glue-Datenbank abzurufen, die Sie kopieren möchten.

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

   Weitere Informationen über den Befehl `get-database` finden Sie unter Datenbank holen.
2. Speichern Sie die JSON-Ausgabe in einer Datei mit dem Namen der neuen Datenbank (z. B. *new_database_name*`.json`) auf Ihrem Desktop.
3. Öffnen Sie die Datei *new_database_name*`.json` in einem Text-Editor.
4. Ändern Sie in der JSON-Datei den `Name`-Eintrag in den neuen Datenbanknamen.

5. Entfernen Sie das Feld `CatalogId`.
6. Speichern Sie die Datei.
7. Führen Sie an einer Eingabeaufforderung den folgenden AWS CLI-Befehl aus, um die Datenbank mit dem neuen Namen unter Verwendung der geänderten Datenbankdefinitionsdatei zu erstellen.

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

   Weitere Informationen über den Befehl `create-database` finden Sie unter Datenbank erstellen. Weitere Informationen zum Laden von AWS CLI-Parametern aus einer Datei finden Sie unter Laden von AWS CLI-Parametern aus einer Datei im AWS Command Line Interface-Benutzerhandbuch.
8. Um zu überprüfen, ob die neue Datenbank in AWS Glue erstellt wurde, führen Sie den folgenden Befehl aus:

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

Jetzt sind Sie bereit, die Definition für eine Tabelle abzurufen, die Sie in die neue Datenbank kopieren möchten, die Definition zu ändern und die geänderte Definition zu verwenden, um die Tabelle in der neuen Datenbank neu zu erstellen. Durch dieses Verfahren wird der Tabellenname nicht geändert.

Kopieren einer AWS Glue-Tabelle mit der AWS CLI

1. Führen Sie an einer Eingabeaufforderung den folgenden AWS CLI-Befehl aus.

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

   Weitere Informationen über den Befehl `get-table` finden Sie unter Tabelle holen.
2. Speichern Sie die JSON-Ausgabe in einer Datei mit dem Namen der neuen Datenbank (z. B. *table_name*.json) auf Ihrem Desktop.
3. Öffnen Sie die Datei in einem Text-Editor.
4. Entfernen Sie in der JSON-Datei den äußeren `{"Table":` -Eintrag und die entsprechende schließende Klammer `}` am Ende der Datei.
5. Entfernen Sie in der JSON-Datei die folgenden Einträge und ihre Werte:

- `DatabaseName` – Dieser Eintrag ist nicht erforderlich, da der `create-table`-CLI-Befehl den `--database-name`-Parameter verwendet.
- `CreateTime`
- `UpdateTime`
- `CreatedBy`
- `IsRegisteredWithLakeFormation`
- `CatalogId`
- `VersionId`

6. Speichern Sie die Tabellendefinitionsdatei.
7. Führen Sie bei einer Eingabeaufforderung den folgenden AWS CLI-Befehl aus, um die Tabelle in der neuen Datenbank neu zu erstellen:

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

   Weitere Informationen über den Befehl `create-table` finden Sie unter Datenbank erstellen.

   Die Tabelle erscheint jetzt in der neuen Datenbank in AWS Glue und kann von Athena abgefragt werden.
8. Wiederholen Sie die Schritte, um jede weitere Tabelle in die neue Datenbank in AWS Glue zu kopieren.

Verwenden von Athena-Daten-Connector für externen Hive-Metastore

Sie können den Amazon-Athena-Daten-Connector für den externen Hive-Metastore verwenden, um Datensätze in Amazon S3 abzufragen in denen ein Apache-Hive-Metastore verwendet wird. Es ist keine Migration von Metadaten zu AWS Glue Data Catalog erforderlich. In der Athena-Verwaltungskonsole konfigurieren Sie eine Lambda-Funktion für die Kommunikation mit dem Hive-Metastore in Ihrer privaten VPC und verbinden sie dann mit dem Metastore. Die Verbindung von Lambda zu Ihrem Hive-Metastore wird durch einen privaten Amazon-VPC-Kanal gesichert und nutzt nicht das öffentliche Internet. Sie können Ihren eigenen Lambda-Funktionscode angeben oder die Standardimplementierung des Athena-Daten-Connectors für externen Hive-Metastore verwenden.

Themen

Übersicht über die Funktionen

Mit dem Athena-Daten-Connector für den externen Hive-Metastore können Sie die folgenden Aufgaben ausführen:

- Verwenden Sie die Athena-Konsole, um benutzerdefinierte Kataloge zu registrieren und Abfragen damit auszuführen.
- Definieren Sie Lambda-Funktionen für verschiedene externe Hive-Metastores und verbinden Sie sie in Athena-Abfragen.
- Verwenden Sie die AWS Glue Data Catalog und Ihre externen Hive-Metastores in derselben Athena-Abfrage.
- Geben Sie einen Katalog im Kontext der Abfrageausführung als aktuellen Standardkatalog an. Dadurch entfällt die Notwendigkeit, Datenbanknamen in Ihren Abfragen Katalognamen voranzustellen. Anstatt die Syntax `catalog.database.table` zu verwenden, können Sie `database.table` verwenden.
- Verwenden Sie eine Vielzahl von Tools, um Abfragen auszuführen, die auf externe Hive-Metastores verweisen. Sie können die Athena-Konsole, die AWS CLI, das AWS-SDK, Athena-APIs und aktualisierte Athena-JDBC- und ODBC-Treiber verwenden. Die aktualisierten Treiber unterstützen benutzerdefinierte Kataloge.

API-Unterstützung

Der Athena-Daten-Connector für den externen Hive-Metastore bietet Unterstützung für Katalogregistrierungs-API-Operationen und Metadaten-API-Operationen.

- Katalogregistrierung – Registrieren Sie benutzerdefinierte Kataloge für externe Hive-Metastores und Verbunddatenquellen.
- Metadaten – Verwenden Sie Metadaten-APIs, um Datenbank- und Tabelleninformationen für AWS Glue und alle Kataloge bereitzustellen, bei denen Sie sich mit Athena registrieren.
- Athena-JAVA-SDK-Client – Verwenden Sie Katalogregistrierungs-APIs, Metadaten-APIs und Unterstützung für Kataloge in der `StartQueryExecution`-Operation im aktualisierten Athena-Java-SDK-Client.

Referenz-Implementierung

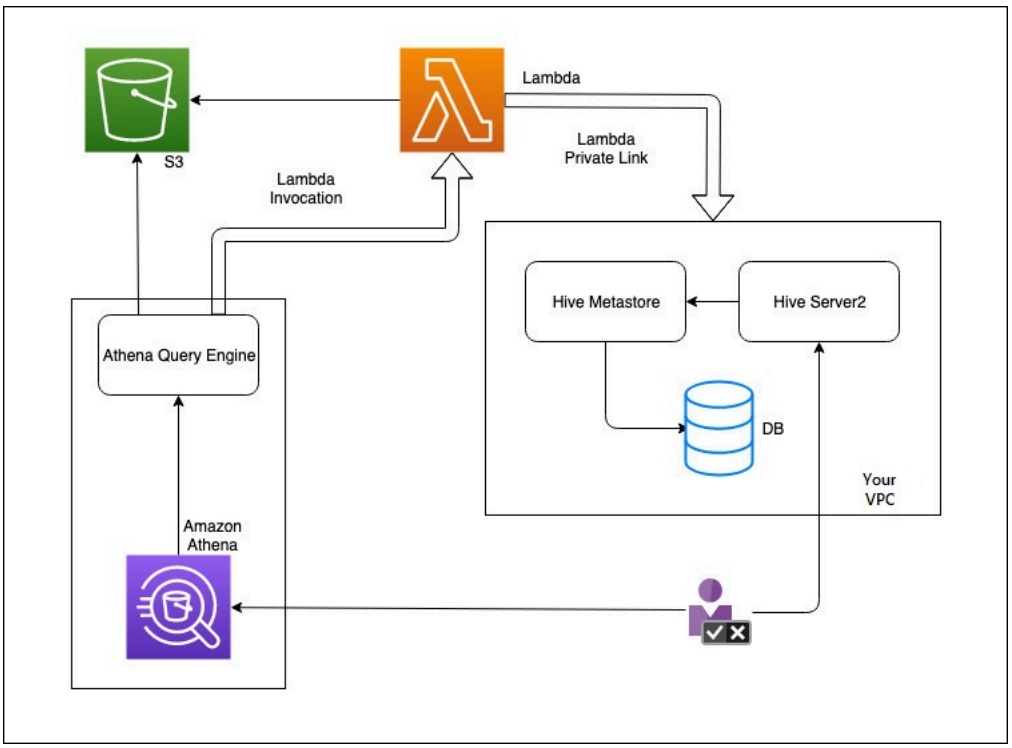
Athena bietet eine Referenzimplementierung für die Lambda-Funktion, die sich mit externen Hive-Metastores verbindet. Die Referenzimplementierung wird auf GitHub als Open-Source-Projekt unter Athena Hive Metastore bereitgestellt.

Die Referenzimplementierung ist in Form der folgenden beiden AWS SAM-Anwendungen in der AWS Serverless Application Repository (SAR) verfügbar. Sie können eine dieser Anwendungen im SAR verwenden, um eigene Lambda-Funktionen zu erstellen.

- **`AthenaHiveMetastoreFunction`** – Uber-Lambda-Funktion `.jar`-Datei. Ein „uber“ JAR (auch bekannt als Fat JAR oder JAR mit Abhängigkeiten) ist eine `.jar`-Datei, die sowohl ein Java-Programm als auch seine Abhängigkeiten in einer einzigen Datei enthält.
- **`AthenaHiveMetastoreFunctionWithLayer`** – Lambda-Ebene und dünne Lambda-Funktions-`.jar`-Datei.

Workflow

Das folgende Diagramm zeigt, wie Athena mit Ihrem externen Hive-Metastore interagiert.



In diesem Workflow befindet sich Ihr mit der Datenbank verbundener Hive-Metastore in Ihrer VPC. Sie verwenden Hive Server2, um Ihren Hive-Metastore mithilfe der Hive-CLI zu verwalten.

Der Workflow für die Verwendung externer Hive-Metastores von Athena beinhaltet die folgenden Schritte.

1. Sie erstellen eine Lambda-Funktion, die eine Athena-Verbindung mit dem Hive-Metastore innerhalb Ihrer VPC herstellt.
2. Sie registrieren einen eindeutigen Katalognamen für Ihren Hive-Metastore und einen entsprechenden Funktionsnamen in Ihrem Konto.
3. Wenn Sie eine Athena-DML- oder DDL-Abfrage ausführen, die den Katalognamen verwendet, ruft die Athena-Abfrage-Engine den Lambda-Funktionsnamen auf, den Sie dem Katalognamen zugeordnet haben.
4. Mit AWS PrivateLink kommuniziert die Lambda-Funktion mit dem externen Hive-Metastore in Ihrer VPC und empfängt Antworten auf Metadatenanforderungen. Athena verwendet die Metadaten Ihres externen Hive-Metastores genauso wie die Metadaten des Standard-AWS Glue Data Catalog.

Überlegungen und Einschränkungen

Berücksichtigen Sie bei der Verwendung des Athena-Daten-Connectors für den externen Hive-Metastore die folgenden Punkte:

- Sie können CTAS verwenden, um eine Tabelle auf einem externen Hive-Metastore zu erstellen.
- Sie können INSERT INTO verwenden, um Daten in einen externen Hive-Metastore einzufügen.
- Die DDL-Unterstützung für externe Hive-Metastores ist auf die folgenden Anweisungen beschränkt.
  - ALTER DATABASE SET DBPROPERTIES
  - TABELLE ÄNDERN SPALTEN HINZUFÜGEN
  - 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
- Die maximale Anzahl registrierter Kataloge beträgt 1.000.
- Die Kerberos-Authentifizierung wird für Hive-Metastores nicht unterstützt.
- Um den JDBC-Treiber mit einem externen Hive-Metastore oder Verbundabfragen zu verwenden, schließen Sie `MetadataRetrievalMethod=ProxyAPI` in Ihre JDBC-Verbindungszeichenfolge ein.

ein. Informationen zum JDBC-Treiber finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC.

- Die ausgeblendeten Hive-Spalten `$path`, `$bucket`, `$file_size`, `$file_modified_time`, `$partition`, `$row_id` können nicht für eine differenzierte Filterung der Zugriffskontrolle verwendet werden.
- In Hive ausgeblendete Systemtabellen wie *`example_table`*`$partitions` oder *`example_table`*`$properties` werden von der detaillierten Zugriffskontrolle nicht unterstützt.

Berechtigungen

Vorab erstellte und benutzerdefinierte Daten-Connectors benötigen möglicherweise Zugriff auf die folgenden Ressourcen, um ordnungsgemäß zu funktionieren. Überprüfen Sie die Informationen für den von Ihnen verwendeten Connector, um sicherzustellen, dass die VPC korrekt konfiguriert ist. Informationen zu den erforderlichen IAM-Berechtigungen zum Ausführen von Abfragen und zum Erstellen eines Datenquellen-Connectors in Athena finden Sie unter Zugriff auf einen Athena-Daten-Connector für externen Hive-Metastore zulassen und Gewährung von Lambda-Funktionszugriff auf externe Hive-Metastores.

- Amazon S3 – Zusätzlich zum Schreiben von Abfrageergebnissen zum Athena-Abfrageergebnisspeicherort in Amazon S3 schreiben Daten-Connectors auch zu einem Spill-Bucket in Amazon S3. Konnektivität und Berechtigungen für diesen Amazon-S3-Standort sind erforderlich. Weitere Informationen finden Sie unter Spill-Speicherort in Amazon S3 an späterer Stelle in diesem Thema.
- Athena – Zugriff ist erforderlich, um den Abfragestatus zu überprüfen und das Overscanning zu verhindern.
- AWS Glue – Zugriff ist erforderlich, wenn Ihr Connector AWS Glue für zusätzliche oder primäre Metadaten verwendet.
- AWS Key Management Service
- Richtlinien – Hive-Metastore, Athena Query Federation und UDFs erfordern zusätzlich zu AWS Verwaltete Richtlinie: AmazonAthenaFullAccess weitere Richtlinien. Weitere Informationen finden Sie unter Identity and Access Management in Athena.

Spill-Speicherort in Amazon S3

Aufgrund des Grenzwerts für Lambda-Funktionsantwortgrößen werden Antworten, die größer als der Schwellenwert sind, an einen Amazon-S3-Speicherort übertragen, den Sie beim Erstellen der Lambda-Funktion angeben. Athena liest diese Antworten direkt von Amazon S3.

> **Note**
>
> Athena entfernt die Antwortdateien nicht von Amazon S3. Es wird empfohlen, eine Aufbewahrungsrichtlinie einzurichten, um Antwortdateien automatisch zu löschen.

Verbinden von Athena mit einem Apache Hive Metastore

Um eine Verbindung von Athena zu einem Apache-Hive-Metastore herzustellen, müssen Sie eine Lambda-Funktion erstellen und konfigurieren. Für eine grundlegende Implementierung können Sie alle erforderlichen Schritte zum Start von der Athena-Verwaltungskonsole aus ausführen.

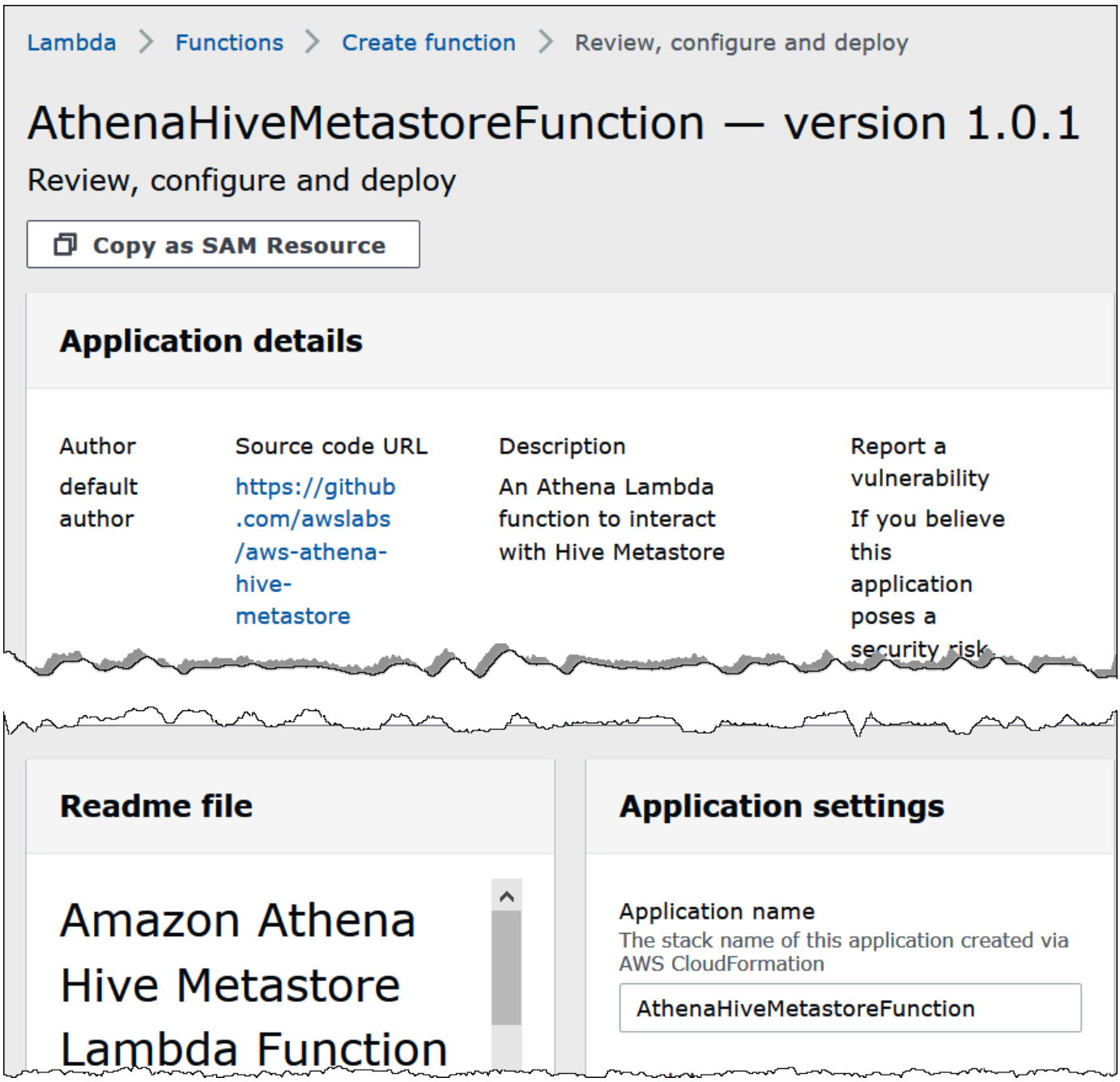
> **Note**
>
> Das folgende Verfahren erfordert, dass Sie über die Berechtigung verfügen, eine benutzerdefinierte IAM-Rolle für die Lambda-Funktion zu erstellen. Wenn Sie nicht über die Berechtigung zum Erstellen einer benutzerdefinierten Rolle verfügen, können Sie mithilfe der Athena-Referenzimplementierung separat eine Lambda-Funktion erstellen und dann mithilfe der AWS Lambda-Konsole eine vorhandene IAM-Rolle für die Funktion auswählen. Weitere Informationen finden Sie unter Verbinden von Athena mit einem Hive-Metastore mithilfe einer vorhandenen IAM-Ausführungsrolle.

So stellen Sie eine Verbindung von Athena zu einem Hive-Metastore her:

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.

3. Wählen Sie Data sources (Datenquellen) aus.
4. Wählen Sie oben rechts in der Konsole Datenquelle erstellen aus.
5. Wählen Sie auf der Seite Eine Datenquelle wählen für Datenquelle S3 - Apache Hive Metastore.
6. Wählen Sie Next (Weiter).
7. Geben Sie im Abschnitt Details zur Datenquelle für Datenquellenname den Namen ein, den Sie in Ihren SQL-Anweisungen verwenden möchten, wenn Sie die Datenquelle von Athena abfragen. Der Name kann bis zu 127 Zeichen lang sein und muss innerhalb Ihres Kontos eindeutig sein. Er kann nicht mehr geändert werden, nachdem Sie ihn erstellt haben. Gültige Zeichen sind a-z, A-Z, 0-9, _ (Unterstrich), @ (At-Zeichen) und - (Bindestrich). Die Namen `awsdatacatalog`, `hive`, `jmx` und `system` sind von Athena reserviert und können nicht für Datenquellennamen verwendet werden.
8. Wählen Sie für Lambda-Funktion Eine Lambda-Funktion erstellen und anschließend Eine neue Lambda-Funktion in AWS Lambda erstellen

   Die Seite AthenaHiveMetastoreFunction wird in der AWS Lambda-Konsole geöffnet. Die Seite enthält detaillierte Informationen zum Connector.

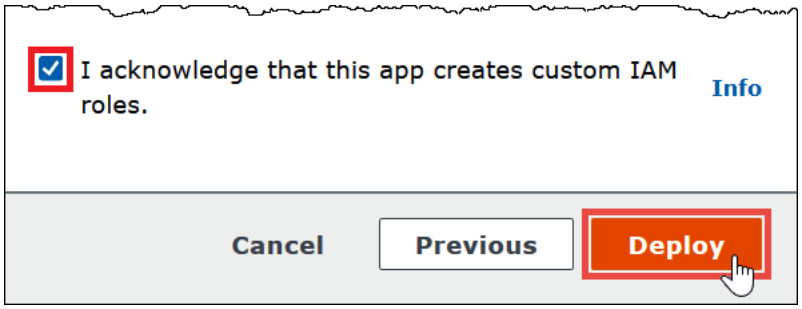


9. Geben Sie unter Anwendungseinstellungen die Parameter für Ihre Lambda-Funktion ein.

   - LambdaFuncName – Geben Sie einen Namen für die Funktion an. Zum Beispiel myHiveMetastore.
   - SpillLocation – Geben Sie in diesem Konto einen Amazon S3-Speicherort an, um Spillover-Metadaten zu speichern, wenn die Antwortgröße der Lambda-Funktion 4 MB überschreitet.
   - HMSUris – Geben Sie den URI Ihres Hive-Metastore-Hosts ein, der das Thrift-Protokoll an Port 9083 verwendet. Verwenden der Syntax `thrift://<host_name>:9083`.

- LambdaMemory – Geben Sie einen Wert zwischen 128 MB und 3008 MB an. Der Lambda-Funktion werden CPU-Zyklen proportional zur von Ihnen konfigurierten Speichermenge zugewiesen. Der Standardwert ist 1024.
- LambdaTimeout – Geben Sie die maximal zulässige Laufzeit des Lambda-Aufrufs in Sekunden von 1 bis 900 an (900 Sekunden sind 15 Minuten). Der Standardwert ist 300 Sekunden (5 Minuten).
- VpcSecurityGroupIds – Geben Sie eine durch Komma getrennte Liste der VPC-Sicherheitsgruppen-IDs für den Hive-Metastore ein.
- VPCSubnetIds – Geben Sie eine durch Komma getrennte Liste der VPC-Subnetz-IDs für den Hive-Metastore ein.

10. Wählen Sie Ich bestätige, dass diese Anwendung benutzerdefinierte IAM-Rollen erstellt und dann Bereitstellen.



Wenn die Bereitstellung abgeschlossen ist, wird Ihre Funktion in Ihrer Liste der Lambda-Anwendungen angezeigt. Nachdem die Hive-Metastore-Funktion für Ihr Konto bereitgestellt wurde, können Sie Athena für die Verwendung konfigurieren.

11. Kehren Sie zur Seite Datenquellendetails eingeben der Athena-Konsole zurück.
12. Wählen Sie im Abschnitt Lambada-Funktion das Symbol Aktualisieren neben dem Lambda-Funktionssuchfeld aus. Wenn Sie die Liste der verfügbaren Funktionen aktualisieren, wird Ihre neu erstellte Funktion in der Liste angezeigt.
13. Wählen Sie den Namen der Funktion aus, die Sie gerade in der Lambda-Konsole erstellt haben. Der ARN der Lambda-Funktion wird angezeigt.
14. (Optional) Fügen Sie für Tags Schlüssel-Wert-Paare hinzu, die mit dieser Datenquelle verknüpft werden sollen. Weitere Informationen zu Tags erhalten Sie unter Markieren von Athena-Ressourcen.
15. Wählen Sie Next (Weiter).
16. Auf der Seite Überprüfen und erstellen prüfen Sie die Datenquellendetails und wählen Sie dann Datenquelle erstellen aus.

17. Der Abschnitt Datenquellendetails auf der Seite für Ihre Datenquelle zeigt Informationen über Ihren neuen Connector an.

    Sie können nun den Data source name (Datenquellennamen) verwenden, den Sie angegeben haben, um auf den Hive-Metastore in Ihren SQL-Abfragen in Athena zu verweisen. Verwenden Sie in Ihren SQL-Abfragen die folgende Beispielsyntax und ersetzen Sie `hms-catalog-1` durch den zuvor angegebenen Katalognamen.

    ```
    SELECT * FROM hms-catalog-1.CustomerData.customers
    ```

18. Informationen zum Anzeigen, Bearbeiten oder Löschen der von Ihnen erstellten Datenquellen finden Sie unter Verwalten von Datenquellen.

Verwendung des AWS Serverless Application Repository für die Bereitstellung eines Hive-Datenquellen-Connectors

Um einen Athena-Datenquellen-Connector für Hive bereitzustellen, können Sie dieAWS Serverless Application Repository verwenden anstatt mit der Athena-Konsole zu beginnen. Verwenden Sie AWS Serverless Application Repository, um den Connector zu suchen, den Sie verwenden möchten, die Parameter anzugeben, die der Connector benötigt, und den Connector dann für Ihr Konto bereitzustellen. Nachdem Sie den Connector bereitgestellt haben, verwenden Sie die Athena-Konsole, um die Datenquelle für Athena verfügbar zu machen.

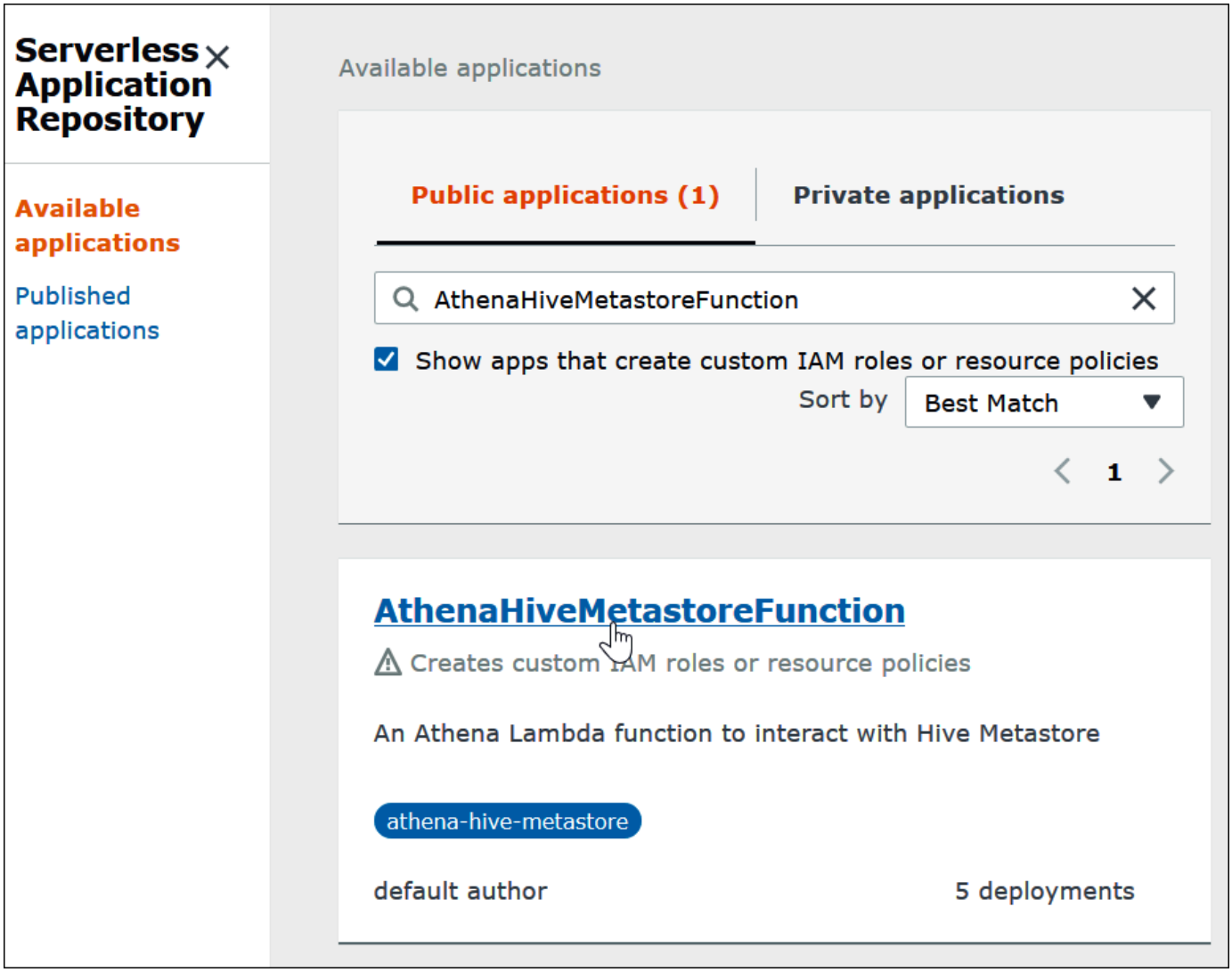
So verwenden Sie den AWS Serverless Application Repository zur Bereitstellung eines Datenquellen-Connectors für Hive in Ihrem Konto

1. Melden Sie sich an der AWS Management Console an und öffnen Sie Serverless App Repository (Serverless-App-Repository).
2. Wählen Sie im Navigationsbereich Available applications (Verfügbare Anwendungen) aus.
3. Wählen Sie die Option Apps anzeigen, die benutzerdefinierte IAM-Rollen oder Ressourcenrichtlinien erstellen.
4. Geben Sie in das Suchfeld ei **`Hive`**. Zu den angezeigten Connectors gehören die folgenden zwei:

    - AthenaHiveMetastoreFunction – Uber-Lambda-Funktions-`.jar`-Datei.
    - AthenaHiveMetastoreFunctionWithLayer – Lambda-Ebene und `.jar`-Thin-Lambda-Funktionsdatei.

Die beiden Anwendungen besitzen dieselbe Funktionalität und unterscheiden sich nur in der Implementierung. Sie können beide für die Erstellung einer Lambda-Funktion verwenden, die Athena mit Ihrem Hive-Metastore verbindet.

5. Wählen Sie den Namen des Connectors aus, den Sie verwenden möchten. In diesem Tutorial wird AthenaHiveMetastoreFunction verwendet.



6. Geben Sie unter Application settings (Anwendungseinstellungen) die Parameter für Ihre Lambda-Funktion ein.

   - LambdaFuncName – Geben Sie einen Namen für die Funktion an. Zum Beispiel myHiveMetastore.
   - SpillLocation – Geben Sie in diesem Konto einen Amazon S3-Speicherort an, um Spillover-Metadaten zu speichern, wenn die Antwortgröße der Lambda-Funktion 4 MB überschreitet.

- HMSUris – Geben Sie den URI Ihres Hive-Metastore-Hosts ein, der das Thrift-Protokoll an Port 9083 verwendet. Verwenden der Syntax `thrift://<host_name>:9083`.
- LambdaMemory – Geben Sie einen Wert zwischen 128 MB und 3008 MB an. Der Lambda-Funktion werden CPU-Zyklen proportional zur von Ihnen konfigurierten Speichermenge zugewiesen. Der Standardwert ist 1024.
- LambdaTimeout – Geben Sie die maximal zulässige Laufzeit des Lambda-Aufrufs in Sekunden von 1 bis 900 an (900 Sekunden sind 15 Minuten). Der Standardwert ist 300 Sekunden (5 Minuten).
- VpcSecurityGroupIds – Geben Sie eine durch Komma getrennte Liste der VPC-Sicherheitsgruppen-IDs für den Hive-Metastore ein.
- VPCSubnetIds – Geben Sie eine durch Komma getrennte Liste der VPC-Subnetz-IDs für den Hive-Metastore ein.

7. Wählen Sie unten rechts auf der Seite Anwendungsdetails die Option Ich bestätige, dass diese App benutzerdefinierte IAM-Rollen erstellt und wählen Sie dann Bereitstellen aus.

An diesem Punkt können Sie Athena so konfigurieren, dass Ihre Lambda-Funktion zur Verbindung mit Ihrem Hive-Metastore verwendet wird. Informationen zu den erforderlichen Schritten finden Sie unter Konfigurieren Sie Athena für die Verwendung eines bereitgestellten Hive-Metastore-Connectors.

Verbinden von Athena mit einem Hive-Metastore mithilfe einer vorhandenen IAM-Ausführungsrolle

Um Ihren externen Hive-Metastore mit Athena über eine Lambda-Funktion zu verbinden, die eine vorhandene IAM-Rolle verwendet, können Sie die Referenzimplementierung des Athena-Connectors für externen Hive-Metastore von Athena verwenden.

Die drei wichtigsten Schritte sind wie folgt:

1. Klonen und Entwickeln – Klonen Sie die Athena Referenzimplementierung und erstellen Sie die JAR-Datei, die den Lambda-Funktionscode enthält.
2. AWS Lambda-Konsole – Erstellen Sie in der AWS Lambda-Konsole eine Lambda-Funktion, weisen Sie ihr eine vorhandene IAM-Ausführungsrolle zu und laden Sie den von Ihnen generierten Funktionscode hoch.

3. Amazon-Athena-Konsole – Erstellen Sie in der Amazon-Athena-Konsole einen Datenquellennamen, mit dem Sie in Ihren Athena-Abfragen auf Ihren externen Hive-Metastore verweisen können.

Wenn Sie bereits über Berechtigungen zum Erstellen einer benutzerdefinierten IAM-Rolle verfügen, können Sie einen einfacheren Workflow verwenden, der die Athena-Konsole und das AWS Serverless Application Repository verwendet, um eine Lambda-Funktion zu erstellen und zu konfigurieren. Weitere Informationen finden Sie unter Verbinden von Athena mit einem Apache Hive Metastore.

Voraussetzungen

- Git muss auf Ihrem System installiert sein.
- Sie müssen Apache Maven installiert haben.
- Sie haben eine IAM-Ausführungsrolle, die Sie der Lambda-Funktion zuweisen können. Weitere Informationen finden Sie unter Gewährung von Lambda-Funktionszugriff auf externe Hive-Metastores.

Klonen und entwickeln Sie die Lambda-Funktion

Der Funktionscode für die Athena-Referenzimplementierung ist ein Maven-Projekt, das sich auf GitHub unter awslabs/aws-athena-hive-metastore befindet. Ausführliche Informationen zum Projekt finden Sie in der entsprechenden README-Datei auf GitHub oder im Referenz-Implementierung-Thema in dieser Dokumentation.

So Klonen und erstellen Sie den Lambda-Funktionscode

1. Geben Sie den folgenden Befehl ein, um die Athena Referenzimplementierung zu klonen:

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

2. Führen Sie den folgenden Befehl aus, um die `.jar`-Datei für die Lambda-Funktion zu entwickeln:

```
mvn clean install
```

   Nachdem das Projekt erfolgreich erstellt wurde, wird die folgende `.jar`-Datei im Zielordner Ihres Projekts erstellt:

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

Im nächsten Abschnitt verwenden Sie die AWS Lambda-Konsole, um diese Datei in Ihr Amazon-Web-Services-Konto hochzuladen.

Erstellen und konfigurieren Sie die Lambda-Funktion in der AWS Lambda-Konsole

In diesem Abschnitt verwenden Sie die AWS Lambda-Konsole, um eine Funktion zu erstellen, die eine vorhandene IAM-Ausführungsrolle verwendet. Nachdem Sie eine VPC für die Funktion konfiguriert haben, laden Sie den Funktionscode hoch und konfigurieren die Umgebungsvariablen für die Funktion.

So erstellen Sie die Lambda-Funktion:

In diesem Schritt erstellen Sie in der AWS Lambda-Konsole eine Funktion, die eine vorhandene IAM-Rolle verwendet.

So erstellen Sie eine Lambda-Funktion, die eine vorhandene IAM-Rolle verwendet

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die AWS Lambda-Konsole an https://console.aws.amazon.com/lambda.
2. Wählen Sie im Navigationsbereich Functions aus.
3. Wählen Sie Create function (Funktion erstellen).
4. Wählen Sie Author from scratch aus.
5. Geben Sie für Funktionsname den Namen Ihrer Lambda-Funktion ein (z. B. **`EHMSBasedLambda`**).
6. Wählen Sie für Runtime die Option Java 8.
7. Erweitern Sie unter Berechtigungen die Option Standardausführungsrolle ändern.
8. Wählen Sie für Execution role (Ausführungsrolle) die Option Use an existing role (Vorhandene Rolle verwenden) aus.
9. Wählen Sie für Vorhandene Rolle die IAM-Ausführungsrolle aus, die Ihre Lambda-Funktion für Athena verwendet (in diesem Beispiel wird eine Rolle namens `AthenaLambdaExecutionRole` verwendet).
10. Erweiterten Sie Advanced settings (Erweiterte Einstellungen).
11. Wählen Sie Netzwerk aktivieren aus.

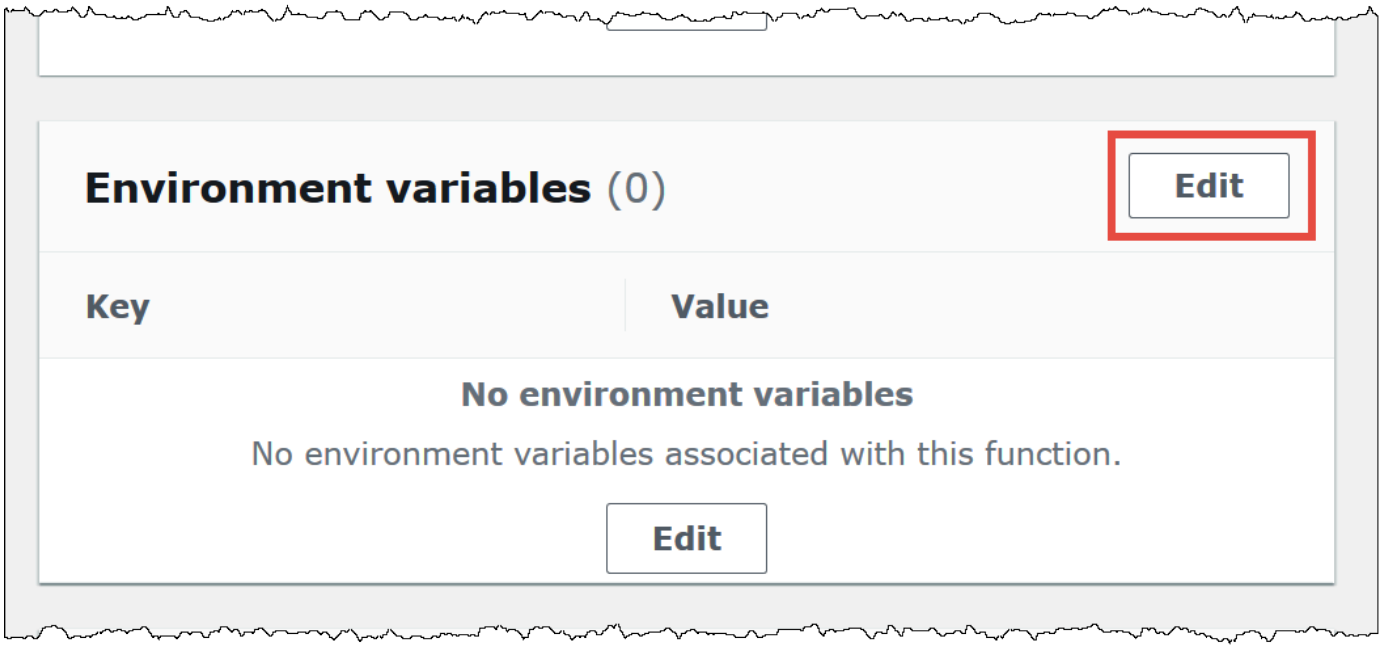
12. Wählen Sie für VPC die VPC aus, auf die Ihre Funktion Zugriff hat.
13. Wählen Sie für Subnetze die VPC-Subnetze aus, die Lambda verwenden soll.
14. Wählen Sie für Sicherheitsgruppen die VPC-Sicherheitsgruppen aus, die Lambda verwenden soll.
15. Wählen Sie Create function (Funktion erstellen). Die AWS Lambda-Konsole und öffnet die Konfigurationsseite für Ihre Funktion und beginnt mit der Erstellung Ihrer Funktion.

Laden Sie den Code hoch und konfigurieren Sie die Lambda-Funktion

Wenn die Konsole Sie darüber informiert, dass Ihre Funktion erfolgreich erstellt wurde, können Sie den Funktionscode hochladen und seine Umgebungsvariablen konfigurieren.

So laden Sie Ihren Lambda-Funktionscode hoch und konfigurieren die Umgebungsvariablen

1. Stellen Sie in der Lambda-Konsole sicher, dass Sie sich auf der Seite der von Ihnen angegebenen Funktion auf der Registerkarte Code befinden.
2. Wählen Sie für Code source (Quellcode) Upload from (Hochladen von) und anschließend .zip or .jar file (.zip- oder .jar-Datei) aus.
3. Laden Sie die zuvor erstellte `hms-lambda-func-1.0-SNAPSHOT-withdep.jar`-Datei hoch.
4. Wählen Sie auf der Seite der Lambda-Funktion den Tab Konfiguration.
5. Wählen Sie im Bereich auf der linken Seite Umgebungsvariablen aus.
6. Wählen Sie im Abschnitt Environment variables (Umgebungsvariablen) Edit (Bearbeiten) aus.

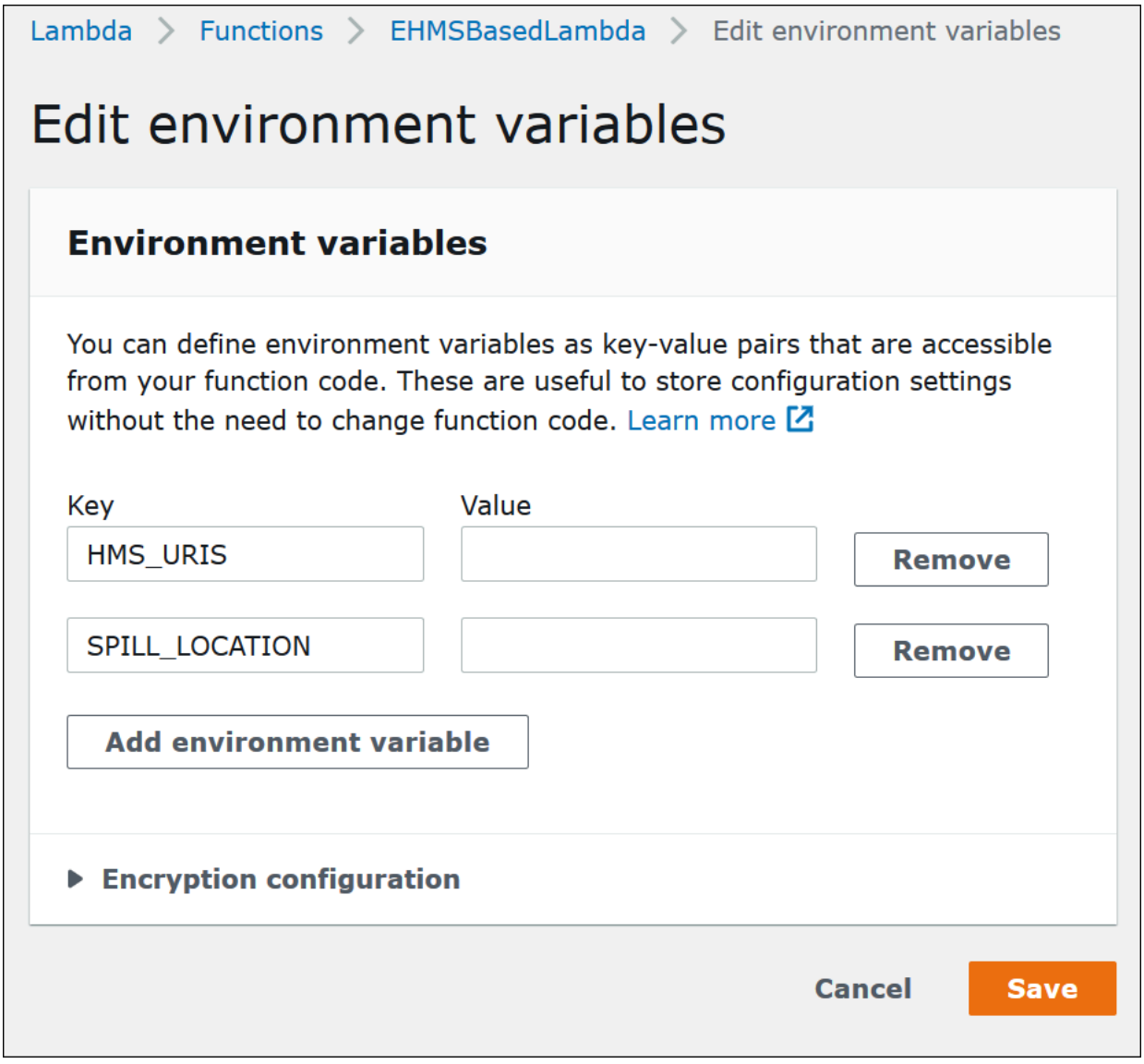


7. Verwenden Sie auf der Seite Edit environment variables (Umgebungsvariablen bearbeiten) die Option Add environment variable (Umgebungsvariable hinzufügen), um die folgenden Umgebungsvariablenschlüssel und -werte hinzuzufügen:

   - HMS_URIS – Verwenden Sie die folgende Syntax, um den URI Ihres Hive-Metastore-Hosts einzugeben, der das Thrift-Protokoll an Port 9083 verwendet.

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

   - SPILL_LOCATION – Geben Sie in Ihrem Amazon-Web-Services-Konto einen Amazon-S3-Speicherort an, um Spillover-Metadaten zu speichern, wenn die Antwortgröße der Lambda-Funktion 4 MB überschreitet.



8. Wählen Sie Save (Speichern) aus.

Zu diesem Zeitpunkt können Sie Athena so konfigurieren, dass Ihre Lambda-Funktion zum Herstellen einer Verbindung mit Ihrem Hive-Metastore verwendet wird. Informationen zu den erforderlichen Schritten finden Sie unter Konfigurieren Sie Athena für die Verwendung eines bereitgestellten Hive-Metastore-Connectors.

Konfigurieren Sie Athena für die Verwendung eines bereitgestellten Hive-Metastore-Connectors

Nachdem Sie einen Lambda-Datenquellen-Connector wie `AthenaHiveMetastoreFunction` in Ihrem Konto bereitgestellt haben, können Sie Athena für die Verwendung konfigurieren. Dazu erstellen Sie einen Datenquellennamen, der auf Ihren externen Hive-Metastore verweist, der in Ihren Athena-Abfragen verwendet werden soll.

So verbinden Sie Athena mit Ihrem Hive-Metastore über eine vorhandene Lambda-Funktion

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.

3. Wählen Sie Data sources (Datenquellen) aus.
4. Wählen Sie auf der Seite Datenquellen die Option Datenquellen erstellen aus.
5. Wählen Sie auf der Seite Eine Datenquelle wählen für Datenquelle S3 - Apache Hive Metastore.
6. Wählen Sie Next (Weiter).
7. Geben Sie im Abschnitt Data source details (Details zur Datenquelle) für Data source name (Datenquellenname) den Namen ein, den Sie in Ihren SQL-Anweisungen verwenden möchten,

wenn Sie die Datenquelle von Athena abfragen (z. B `MyHiveMetastore`). Der Name kann bis zu 127 Zeichen lang sein und muss innerhalb Ihres Kontos eindeutig sein. Er kann nicht mehr geändert werden, nachdem Sie ihn erstellt haben. Gültige Zeichen sind a-z, A-Z, 0-9, _ (Unterstrich), @ (At-Zeichen) und - (Bindestrich). Die Namen `awsdatacatalog`, `hive`, `jmx` und `system` sind von Athena reserviert und können nicht für Datenquellennamen verwendet werden.

8. Im Abschnitt Verbindungsdetails verwenden Sie das Feld Auswählen oder Eingeben einer Lambda-Funktion, um den Namenn der Funktion, die Sie soeben erstellt haben, auszuwählen. Der ARN der Lambda-Funktion wird angezeigt.
9. (Optional) Fügen Sie für Tags Schlüssel-Wert-Paare hinzu, die mit dieser Datenquelle verknüpft werden sollen. Weitere Informationen zu Tags erhalten Sie unter Markieren von Athena-Ressourcen.
10. Wählen Sie Next (Weiter).
11. Auf der Seite Überprüfen und erstellen prüfen Sie die Datenquellendetails und wählen Sie dann Datenquelle erstellen aus.
12. Der Abschnitt Datenquellendetails auf der Seite für Ihre Datenquelle zeigt Informationen über Ihren neuen Connector an.

    Sie können nun den Data source name (Datenquellennamen) verwenden, den Sie angegeben haben, um auf den Hive-Metastore in Ihren SQL-Abfragen in Athena zu verweisen.

    Verwenden Sie in Ihren SQL-Abfragen die folgende Beispielsyntax und ersetzen Sie `ehms-catalog` durch den zuvor angegebenen Datenquellennamen.

    ```
    SELECT * FROM ehms-catalog.CustomerData.customers
    ```

13. Informationen zum Anzeigen, Bearbeiten oder Löschen der von Ihnen erstellten Datenquellen finden Sie unter Verwalten von Datenquellen.

Verwenden eines Standard-Namens für die Datenquelle in externen Hive-Metastore-Abfragen

Wenn Sie DML- und DDL-Abfragen für externe Hive-Metastores ausführen, können Sie die Abfragesyntax vereinfachen, indem Sie den Katalognamen weglassen, wenn dieser Name im Abfrage-Editor ausgewählt ist. Für diese Funktionalität gelten bestimmte Einschränkungen.

DML-Anweisungen.

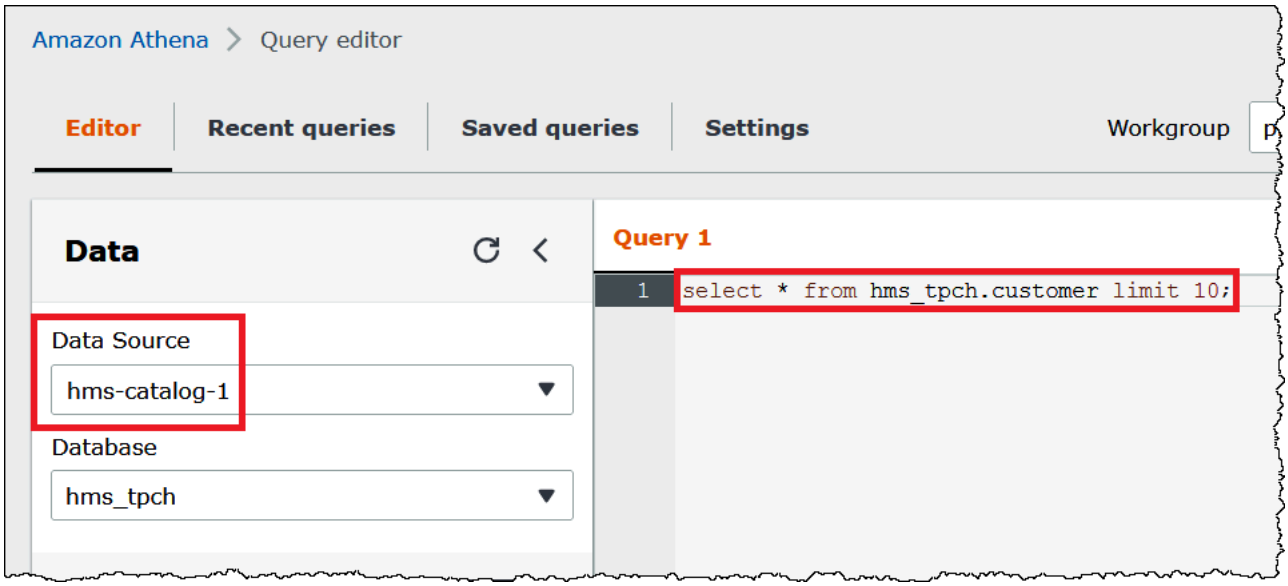
So führen Sie Abfragen mit registrierten Katalogen aus:

1. Sie können den Datenquellennamen mithilfe der Syntax `[[data_source_name].database_name].table_name` vor die Datenbank einfügen, wie im folgenden Beispiel gezeigt.

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

2. Wenn die Datenquelle, die Sie verwenden möchten, in dem Abfrage-Editor bereits ausgewählt ist, können Sie den Namen wie im folgenden Beispiel aus der Abfrage weglassen.

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



3. Wenn Sie mehrere Datenquellen in einer Abfrage verwenden, können Sie nur den Standarddatenquellennamen weglassen und müssen den vollständigen Namen für alle nicht standardmäßigen Datenquellen angeben.

   Angenommen, `AwsDataCatalog` ist im Abfrage-Editor als Standarddatenquelle ausgewählt. Die `FROM`-Anweisung im folgenden Abfrage-Auszug qualifiziert die ersten beiden Datenquellennamen vollständig, lässt jedoch den Namen für die dritte Datenquelle aus, da sie sich im AWS Glue-Datenkatalog befindet.

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

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

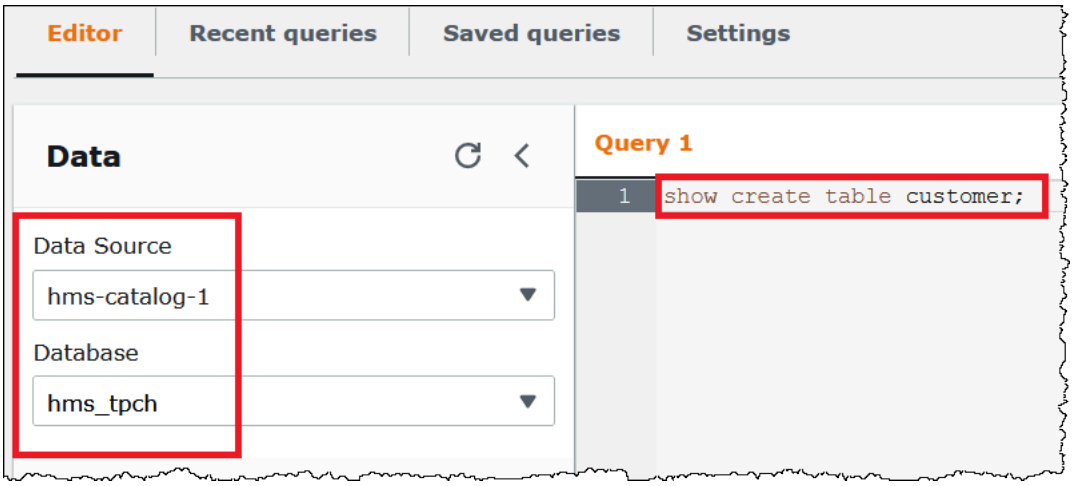
DDL-Anweisungen

Die folgenden Athena-DDL-Anweisungen unterstützen Katalognamenpräfixe. Katalognamenpräfixe in anderen DDL-Anweisungen verursachen Syntaxfehler.

```
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] | [.'$elem$'] | [.'$key$'] | [.'$value$'] )]
```

Wie bei DML-Anweisungen können Sie die Datenquelle und die Datenbankpräfixe aus der Abfrage weglassen, wenn die Datenquelle und die Datenbank im Abfrage-Editor ausgewählt werden.

Im folgenden Bild werden die `hms-catalog-1`-Datenquelle und die `hms_tpch`-Datenbank im Abfrage-Editor ausgewählt. Die `show create table customer`-Anweisung ist erfolgreich, obwohl das `hms-catalog-1`-Präfix und der `hms_tpch`-Datenbankname von der Abfrage selbst weggelassen werden.



Angeben einer Standard-Datenquelle in einer JDBC-Verbindungszeichenfolge

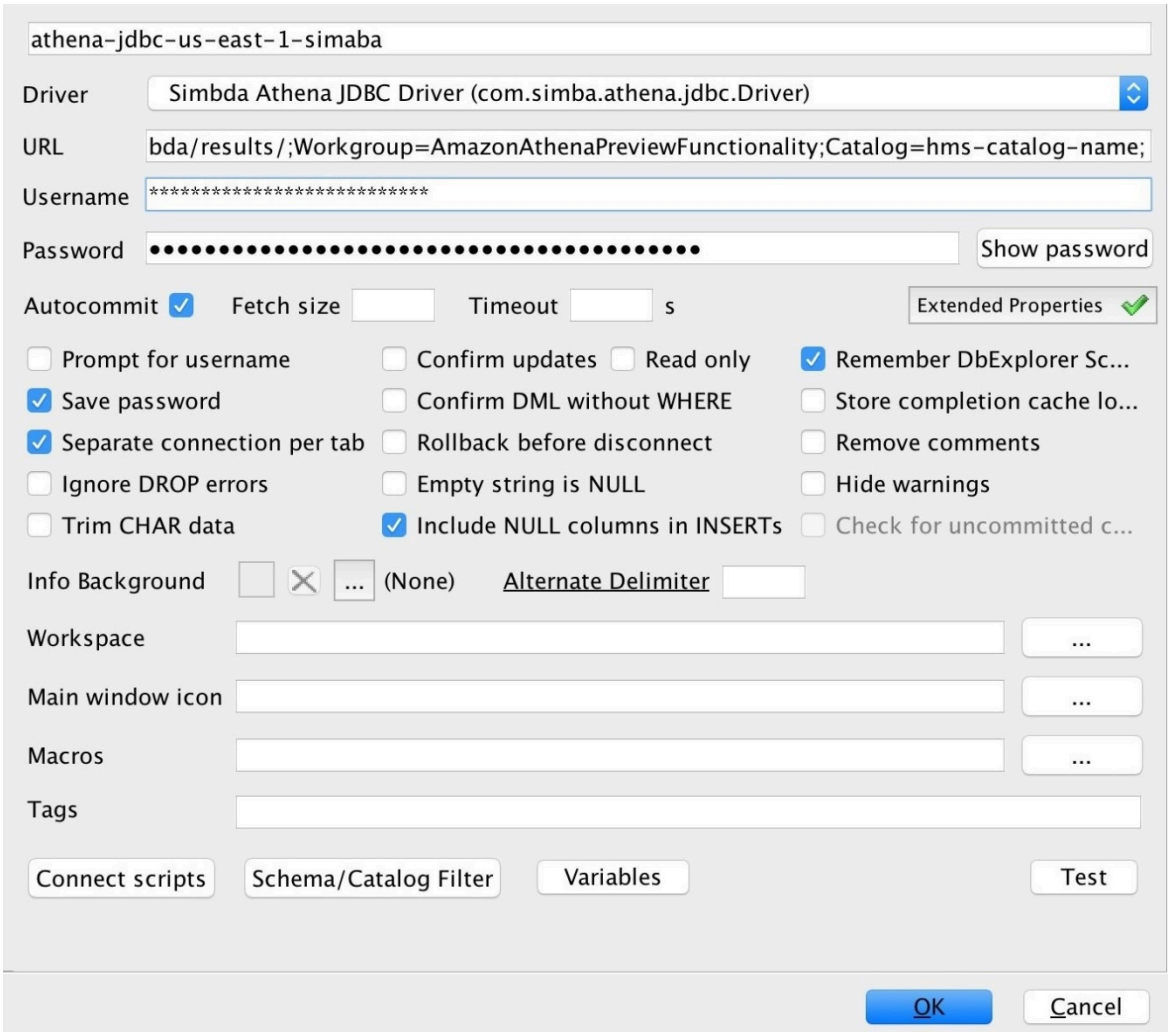
Wenn Sie den Athena-JDBC-Treiber verwenden, um Athena mit einem externen Hive-Metastore zu verbinden, können Sie den `Catalog`-Parameter verwenden, um den Standard-Namen der Datenquelle in Ihrer Verbindungszeichenfolge in einem SQL-Editor wie SQL Workbench anzugeben.

> **Note**
>
> Informationen zum Herunterladen des neuesten Athena-JDBC-Treibers finden Sie unter Verwenden von Athena mit dem JDBC-Treiber.

Die folgende Verbindungszeichenfolge gibt die Standarddatenquelle *`hms-catalog-name`* an.

```
    jdbc:awsathena://AwsRegion=us-east-1;S3OutputLocation=s3://<location>/lambda/
results/;Workgroup=AmazonAthenaPreviewFunctionality;Catalog=hms-catalog-name;
```

Das folgende Image zeigt ein Beispiel für eine JDBC-Verbindungs-URL, wie sie in SQL Workbench konfiguriert ist.



Arbeiten mit Hive-Ansichten

Sie können Athena verwenden, um vorhandene Ansichten in Ihren externen Apache-Hive-Metaspeichern abzufragen. Athena übersetzt Ihre Ansichten zur Laufzeit für Sie, ohne die ursprüngliche Ansicht zu ändern oder die Übersetzung zu speichern.

Angenommen, Sie haben eine Hive-Ansicht wie die folgende, die eine Syntax verwendet, die in Athena nicht unterstützt wird, wie `LATERAL VIEW explode()`:

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

Athena übersetzt die Hive-View-Abfragezeichenfolge in eine Anweisung wie die folgende, die Athena ausführen kann:

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

Informationen zum Verbinden eines externen Hive-Metastores mit Athena finden Sie unter Verwenden von Athena-Daten-Connector für externen Hive-Metastore.

Überlegungen und Einschränkungen

Berücksichtigen Sie bei der Abfrage von Hive-Ansichten von Athena die folgenden Punkte:

- Athena unterstützt das Erstellen von Hive-Ansichten nicht. Sie können Hive-Ansichten in Ihrem externen Hive-Metastore erstellen, die Sie dann von Athena abfragen können.
- Athena unterstützt keine benutzerdefinierten UDFs für Hive-Ansichten.
- Aufgrund eines bekannten Problems in der Athena-Konsole werden Hive-Ansichten unter der Tabellenliste anstelle der Liste der Ansichten angezeigt.
- Obwohl der Übersetzungsprozess automatisch abläuft, werden bestimmte Hive-Funktionen für Hive-Ansichten nicht unterstützt oder erfordern eine spezielle Handhabung. Weitere Informationen finden Sie im folgenden Abschnitt.

Einschränkungen bei der Unterstützung von Hive-Funktionen

In diesem Abschnitt werden die Hive-Funktionen hervorgehoben, die Athena nicht für Hive-Ansichten unterstützt oder die eine spezielle Behandlung erfordern. Da Athena hauptsächlich Funktionen von Hive 2.2.0 unterstützt, sind derzeit Funktionen, die nur in höheren Versionen (wie Hive 4.0.0) verfügbar sind, nicht verfügbar. Eine vollständige Liste der Hive-Funktionen finden Sie unter Hive Sprachhandbuch UDF.

Aggregationsfunktionen

Aggregatfunktionen, die eine besondere Handhabung erfordern

Die folgende Aggregatfunktion für Hive-Ansichten erfordert eine besondere Handhabung.

- Avg – Verwenden Sie `avg(CAST(i AS DOUBLE))` anstelle von `avg(INT i)`.

Aggregatfunktionen werden nicht unterstützt

Die folgenden Hive-Aggregatfunktionen werden in Athena für Hive-Ansichten nicht unterstützt.

```
covar_pop
histogram_numeric
ntile
percentile
percentile_approx
```

Regressionsfunktionen wie `regr_count`, `regr_r2` und `regr_sxx` werden in Athena für Hive-Ansichten nicht unterstützt.

Datumsfunktionen werden nicht unterstützt

Die folgenden Hive-Datumsfunktionen werden in Athena für Hive-Ansichten nicht unterstützt.

```
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)
```

Maskierungsfunktionen werden nicht unterstützt

Hive-Maskierungsfunktionen wie `mask()` und `mask_first_n()` werden in Athena für Hive-Ansichten nicht unterstützt.

Verschiedene Funktionen

Aggregatfunktionen, die eine besondere Handhabung erfordern

Die folgenden verschiedenen Funktionen für Hive-Ansichten erfordern eine besondere Handhabung.

- md5 .. Athena unterstützt `md5(binary)` aber nicht `md5(varchar)`.
- Explode – Athena unterstützt `explode`, wenn es in der folgenden Syntax verwendet wird:

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

- Postexplode – Athena unterstützt `posexplode`, wenn es in der folgenden Syntax verwendet wird:

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

  In der `(pos, val)`-Ausgabe behandelt Athena die `pos`-Spalte als `BIGINT`. Aus diesem Grund müssen Sie möglicherweise die `pos`-Spalte in `BIGINT` umwandeln, um eine veraltete Ansicht zu vermeiden. Das folgende Beispiel illustriert diese Technik.

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

Verschiedene Funktionen werden nicht unterstützt

Die folgenden Hive-Funktionen werden in Athena für Hive-Ansichten nicht unterstützt.

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

Operatoren

Betreiber, die eine Sonderverarbeitung benötigen

Die folgenden Operatoren für Hive-Ansichten erfordern eine besondere Handhabung.

- Mod-Operator (%) – Da der `DOUBLE`-Typ implizit in `DECIMAL(x,y)` umgewandelt wird, kann die folgende Syntax zu einer View is stale-Fehlermeldung führen:

```
a_double % 1.0 AS column
```

  Um dieses Problem zu umgehen, verwenden Sie `CAST`, wie im folgenden Beispiel.

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

- Divisionsbetreiber (/) – In Hive produziert `int` geteilt durch `int` `double`. In Athena erzeugt dieselbe Operation einen abgeschnittenen `int`.

Operatoren werden nicht unterstützt

Athena unterstützt die folgenden Operatoren für Hive-Ansichten nicht.

~A – Bitwise `NOT`

A ^ b – Bitwise `XOR`

A & b – Bitwise `AND`

A | b – Bitwise `OR`

A **<=>** b – Gibt das gleiche Ergebnis wie der Operator Gleich (=) für Nicht-Null-Operanden zurück. Gibt `TRUE` zurück, wenn beide `NULL` sind, `FALSE`, wenn einer von ihnen `NULL` ist.

Zeichenfolgenfunktionen

Zeichenfolgen-Funktionen, die eine besondere Handhabung erfordern

Die folgenden Hive-Zeichenfolgen-Funktionen für Hive-Ansichten erfordern eine besondere Handhabung.

- chr(bigint|double a) – Hive erlaubt negative Argumente; Athena nicht.
- instr(string str, string substr) – Da das Athena-Mapping für die `instr`-Funktion `BIGINT` statt `INT` zurückgibt, verwenden Sie die folgende Syntax:

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

  Ohne diesen Schritt wird die Ansicht als abgestanden angesehen.
- length(string a) – Da das Athena-Mapping für die `length`-Funktion `BIGINT` anstelle von `INT` zurückgibt, verwenden Sie die folgende Syntax, damit die Ansicht nicht als veraltet betrachtet wird:

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

Zeichenfolgen-Funktionen werden nicht unterstützt

Die folgenden Hive-Zeichenfolgen-Funktionen werden in Athena für Hive-Ansichten nicht unterstützt.

```
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)
```

XPath-Funktionen werden nicht unterstützt

Hive-XPath-funktionen wie `xpath`, `xpath_short` und `xpath_int` werden in Athena für Hive-Ansichten nicht unterstützt.

Fehlerbehebung

Wenn Sie Hive-Ansichten in Athena verwenden, treten möglicherweise die folgenden Probleme auf:

- Ansicht ***<Ansichtsname>*** ist veraltet – Diese Meldung weist normalerweise auf eine Typenabweichung zwischen der Ansicht in Hive und Athena hin. Wenn dieselbe Funktion in der Hive LanguageManual UDF- und Presto-Dokumentation zu Funktionen und Operatoren unterschiedliche Signaturen hat, versuchen Sie, den nicht übereinstimmenden Datentyp umzuwandeln.
- Funktion nicht registriert – Athena unterstützt die Funktion derzeit nicht. Einzelheiten finden Sie weiter oben in diesem Dokument.

Verwendung der AWS CLI mit Hive-Metastores

Sie können `aws athena`-CLI-Befehle verwenden, um die Hive-Metastore-Datenkataloge zu verwalten, die Sie mit Athena verwenden. Nach der Definition eines oder mehrerer Kataloge für die Verwendung mit Athena können Sie in den `aws athena`-DDL- und -DML-Befehlen auf diese Kataloge verweisen.

Verwendung der AWS CLI für die Verwaltung von Hive-Metastore-Katalogen

Registrierung eines Katalogs: Create-data-catalog

Um einen Datenkatalog zu registrieren, verwenden Sie den Befehl `create-data-catalog`. Sie verwenden den Parameter `name`, um den Namen anzugeben, den Sie für den Katalog verwenden möchten. Sie übergeben den ARN der Lambda-Funktion an die Option `metadata-function` des Arguments `parameters`. Um Tags für den neuen Katalog zu erstellen, verwenden Sie den Parameter `tags` mit einem oder mehreren, durch Leerzeichen getrennten Argumentpaaren `Key=`*`key`*`,Value=`*`value`*.

Im folgenden Beispiel wird ein Hive-Metastore-Katalog mit dem Namen `hms-catalog-1` registriert. Der Befehl wurde zur besseren Lesbarkeit formatiert.

```
$ 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
```

Anzeige von Katalogdetails: Get-data-catalog

Um die Details eines Katalogs anzuzeigen, übergeben Sie den Namen des Katalogs an den Befehl `get-data-catalog` wie im folgenden Beispiel gezeigt.

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

Das folgende Beispielergebnis verwendet das JSON-Format.

```
{
```

```
    "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"
        }
    }
}
```

Auflistung registrierter Kataloge: List-data-catalogs

Um die registrierten Kataloge aufzulisten, verwenden Sie den Befehl `list-data-catalogs` und geben optional eine Region an, wie im folgenden Beispiel gezeigt. Die aufgelisteten Kataloge enthalten stets AWS Glue.

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

Das folgende Beispielergebnis verwendet das JSON-Format.

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

Aktualisierung eines Katalogs: Update-data-catalog

Um einen Datenkatalog zu aktualisieren, verwenden Sie den Befehl `update-data-catalog` wie im folgenden Beispiel gezeigt. Der Befehl wurde zur besseren Lesbarkeit formatiert.

```
$ 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
```

Löschung eines Katalogs: Delete-data-catalog

Um einen Datenkatalog zu löschen, verwenden Sie den Befehl `delete-data-catalog` wie im folgenden Beispiel gezeigt.

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

Anzeige von Datenbankdetails: Get-database

Um die Details einer Datenbank anzuzeigen, übergeben Sie den Namen des Katalogs und der Datenbank an den Befehl `get-database` wie im folgenden Beispiel gezeigt.

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

Das folgende Beispielergebnis verwendet das JSON-Format.

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

Auflistung von Datenbanken in einem Katalog: List-databases

Um die Datenbanken in einem Katalog aufzulisten, verwenden Sie den Befehl `list-databases` und geben optional eine Region an, wie im folgenden Beispiel gezeigt.

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

Das folgende Beispielergebnis verwendet das JSON-Format.

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

Anzeige von Tabellendetails: Get-table-metadata

Um die Metadaten für eine Tabelle einschließlich Spaltennamen und Datentypen anzuzeigen, übergeben Sie den Namen des Katalogs, der Datenbank und der Tabelle an den Befehl `get-table-metadata` wie im folgenden Beispiel gezeigt.

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

Das folgende Beispielergebnis verwendet das JSON-Format.


```
{
    "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://athena-data/",
                "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"
            }
        }
}
```


Anzeige von Metadaten für alle Tabellen in einer Datenbank: List-table-metadata

Um die Metadaten für alle Tabellen in einer Datenbank anzuzeigen, übergeben Sie den Namen des Katalogs und der Datenbank an den Befehl `list-table-metadata`. Der Befehl `list-table-`

`metadata` ist dem Befehl `get-table-metadata` vergleichbar, abgesehen davon, dass kein Tabellenname angegeben wird. Um die Anzahl der Ergebnisse zu begrenzen, können Sie die Option `--max-results`verwenden wie im folgenden Beispiel gezeigt.

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

Das folgende Beispielergebnis verwendet das JSON-Format.

```
{
    "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://athena-data/",
                "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://myjasondata/",
                "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
}
```

Ausführung von DDL- und DML-Anweisungen

Wenn Sie für die Ausführung von DDL- und DML-Anweisungen die AWS CLI verwenden, können Sie den Namen des Hive-Metastore-Katalogs auf zwei Arten übergeben:

- Direkt an die Anweisungen, die ihn unterstützen.
- An den Parameter `--query-execution-context` `Catalog`.

DDL-Anweisungen

Im folgenden Beispiel wird der Katalogname direkt als Teil der DDL-Anweisung `show create table` übergeben. Der Befehl wurde zur besseren Lesbarkeit formatiert.

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

In der folgenden DDL-Beispielanweisung `show create table` wird der Parameter `Catalog` von `--query-execution-context` verwendet, um den Namen des Hive-Metastore-Katalogs `hms-catalog-1` zu übergeben. Der Befehl wurde zur besseren Lesbarkeit formatiert.

```
$ 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://mybucket/lambda/results"
```

DML-Anweisungen.

Die folgende DML-Beispielanweisung `select` übergibt den Katalognamen direkt an die Abfrage. Der Befehl wurde zur besseren Lesbarkeit formatiert.

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

In der folgenden DML-Beispielanweisung `select` wird der Parameter `Catalog` von `--query-execution-context` verwendet, um den Namen des Hive-Metastore-Katalogs `hms-catalog-1` zu übergeben. Der Befehl wurde zur besseren Lesbarkeit formatiert.

```
$ 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://mybucket/lambda/results"
```

Referenz-Implementierung

Athena stellt eine Referenzimplementierung seines Connectors für den externen Hive-Metastore auf GitHub.com unter https://github.com/awslabs/aws-athena-hive-metastore bereit.

Die Referenzimplementierung ist ein Apache-Maven-Projekt das die folgenden Module hat:

- **`hms-service-api`** – Enthält die API-Operationen zwischen der Lambda-Funktion und den Athena-Serviceclients. Diese API-Operationen sind in der `HiveMetaStoreService`-Schnittstelle definiert. Da es sich um einen Servicevertrag handelt, sollten Sie in diesem Modul nichts ändern.
- **`hms-lambda-handler`** – Eine Reihe von Standard-Lambda-Handlern, die alle Hive-Metastore-API-Aufrufe verarbeiten. Die Klasse `MetadataHandler` ist der Dispatcher für alle API-Aufrufe. Sie müssen dieses Paket nicht ändern.
- **`hms-lambda-func`** – Eine Lambda-Beispielfunktion mit den folgenden Komponenten.
  - **`HiveMetaStoreLambdaFunc`** – Eine Lambda-Beispielfunktion, die `MetadataHandler` erweitert.
  - **`ThriftHiveMetaStoreClient`** – Ein Thrift-Client, der mit dem Hive-Metastore kommuniziert. Dieser Client ist für Hive 2.3.0 geschrieben. Wenn Sie eine andere Hive-Version verwenden, müssen Sie diese Klasse möglicherweise aktualisieren, um sicherzustellen, dass die Antwortobjekte kompatibel sind.
  - **`ThriftHiveMetaStoreClientFactory`** – Steuert das Verhalten der Lambda-Funktion. Beispielsweise können Sie Ihre eigene Gruppe von Handler-Anbietern bereitstellen, indem Sie die `getHandlerProvider()`-Methode überschreiben.
  - `hms.properties` – Konfigurieren Sie die Lambda-Funktion In den meisten Fällen müssen nur die folgenden beiden Eigenschaften aktualisiert werden.
    - `hive.metastore.uris` – der URI des Hive-Metastores im Format `thrift://<host_name>:9083`.
    - `hive.metastore.response.spill.location`: Der Amazon-S3-Speicherort zum Speichern von Antwortobjekten, wenn ihre Größe einen bestimmten Schwellenwert überschreitet (z. B. 4 MB). Der Schwellenwert wird in der Eigenschaft `hive.metastore.response.spill.threshold` definiert. Das Ändern des Standardwerts wird nicht empfohlen.

> **Note**
>
> Diese beiden Eigenschaften können von den Lambda-Umgebungsvariablen `HMS_URIS` und `SPILL_LOCATION` überschrieben werden. Verwenden Sie diese Variablen, anstatt den Quellcode für die Lambda-Funktion neu zu kompilieren, wenn Sie die Funktion mit einem anderen Hive-Metastore oder Überlaufspeicherort verwenden möchten.

- **`hms-lambda-layer`** – Ein Maven-Assemblyprojekt, das `hms-service-api`, `hms-lambda-handler` und ihre Abhängigkeiten in eine `.zip`-Datei setzt. Die `.zip`-Datei wird als Lambda-Ebene für die Verwendung durch mehrere Lambda-Funktionen registriert.
- **`hms-lambda-rnp`** – Zeichnet die Antworten einer Lambda-Funktion auf und verwendet sie dann zur Wiedergabe der Antwort. Sie können dieses Modell verwenden, um Lambda-Antworten zu Testzwecken zu simulieren.

Entwickeln der Artefakte

In den meisten Anwendungsfällen ist es nicht erforderlich, die Referenzimplementierung zu ändern. Bei Bedarf können Sie jedoch den Quellcode ändern, die Artefakte selbst erstellen und zu einem Amazon-S3-Speicherort hochladen.

Bevor Sie die Artefakte erstellen, aktualisieren Sie die Eigenschaften `hive.metastore.uris` und `hive.metastore.response.spill.location` in der `hms.properties`-Datei im `hms-lambda-func`-Modul.

Um die Artefakte zu erstellen, müssen Sie Apache Maven installiert haben und den Befehl `mvn install` ausführen. Dies erzeugt die Ebene-`.zip`-Datei im Ausgabeordner mit dem Namen `target` im Modul `hms-lambda-layer` und die Lambda-Funktions-`.jar`-Datei im Modul `hms-lambd-func`.

Nutzung von Amazon-Athena-Verbundabfrage

Wenn Daten in anderen Quellen als Amazon S3 vorliegen, können Sie Athena-Verbundabfrage verwenden, um die Daten direkt in diesen Quellen abzufragen oder um Pipelines zu erstellen, die Daten aus mehreren Datenquellen extrahieren und in Amazon S3 speichern. Athena-Verbundabfrage ermöglicht Ihnen die Ausführung von SQL-Abfragen für Daten, die in relationalen, nicht relationalen, benutzerdefinierten und Objektdatenquellen gespeichert sind.

Athena verwendet zum Ausführen von Verbundabfragen Datenquellen-Connectors, die in AWS Lambda ausgeführt werden. Ein Datenquellen-Connector ist ein Codestück, das zwischen

der Zieldatenquelle und Athena übersetzen kann. Sie können sich einen Connector als eine Erweiterung der Abfrage-Engine von Athena vorstellen. Es gibt vorab entwickelte Athena-Datenquellen-Connectors für Datenquellen wie Amazon CloudWatch Logs, Amazon DynamoDB, Amazon DocumentDB, Amazon RDS, JDBC-kompatible relationale Datenquellen wie MySQL sowie PostgreSQL unter der Apache-2.0-Lizenz. Sie können auch den SDK von Athena Query Federation verwenden, um Connectors zu schreiben. Um einen Datenquellen-Connector für Ihr Konto auszuwählen, zu konfigurieren und bereitzustellen, können Sie die Athena- und Lambda-Konsole oder AWS Serverless Application Repository verwenden. Nach der Bereitstellung wird der Datenquellen-Connector einem Katalog zugeordnet, den Sie in SQL-Abfragen angeben können. Sie können SQL-Anweisungen aus mehreren Katalogen kombinieren und mehrere Datenquellen mit einer einzigen Abfrage abfragen.

Wenn für eine Datenquelle eine Abfrage ausgeführt wird, ruft Athena den entsprechenden Connector auf, um die Teile der Tabellen zu identifizieren, die gelesen werden müssen, verwaltet die Parallelität und schiebt Filterprädikate nach unten. Basierend auf dem Benutzer, der die Abfrage übermittelt, können Connectors den Zugriff auf bestimmte Datenelemente zulassen oder einschränken. Connectors verwenden Apache Arrow als Format für die von einer Abfrage angeforderten zurückgegebenen Daten. Dies ermöglicht die Implementierung von Connectors in Sprachen wie C, C ++, Java, Python und Rust. Da Connectors in Lambda verarbeitet werden, können sie für den Zugriff auf Daten aus beliebigen Cloud- oder On-Premises-Datenquellen verwendet werden, auf die Lambda zugreifen kann.

Um einen eigenen Datenquellen-Connector zu schreiben, können Sie mittels des SDKs für Athena Query Federation einen der vorab entwickelten Connectors anpassen, die Amazon Athena bereitstellt und verwaltet. Sie können eine Kopie des Quellcodes aus dem GitHub-Repository kopieren und anschließend das Connector Publish Tool verwenden, um ein eigenes AWS Serverless Application Repository-Paket zu erstellen.

> **Note**
>
> Drittanbieter-Entwickler haben möglicherweise das SDK für Athena Query Federation zum Schreiben von Datenquellen-Connectors verwendet. Bei Support- oder Lizenzierungsproblemen mit diesen Datenquellen-Connectors wenden Sie sich bitte an Ihren Connector-Anbieter. Diese Connectors werden nicht getestet oder unterstützt von AWS.

Eine Liste der Datenquellen-Connectors, die von Athena geschrieben und getestet wurden, finden Sie unter Verfügbare Datenquellenkonnektoren.

Informationen zum Schreiben eines eigenen Datenquellen-Connectors finden Sie unter Beispiel-Athena-Connector auf GitHub.

Überlegungen und Einschränkungen

- Engine-Versionen – Athena-Verbundabfrage wird nur von der Athena-Engine-Version 2 und späteren Versionen unterstützt. Weitere Informationen über Athena-Engine-Versionen finden Sie unter Athena-Engine-Versionierung.
- Ansichten – Sie können Ansichten für verbundene Datenquellen erstellen und abfragen. Verbundene Ansichten werden in AWS Glue und nicht in der zugrunde liegenden Datenquelle gespeichert. Weitere Informationen finden Sie unter Verbundansichten abfragen.
- Schreiboperationen – Schreiboperationen wie INSERT INTO werden nicht unterstützt. Ein Versuch, dies zu tun, kann zur Fehlermeldung führen: Dieser Vorgang wird derzeit für externe Kataloge nicht unterstützt.
- Preise – Informationen zu den Preisen finden unter Amazon-Athena-Preise.

  JDBC-Treiber – Um den JDBC-Treiber mit Verbundabfragen oder einem externen Hive-Metastore zu verwenden, schließen Sie `MetadataRetrievalMethod=ProxyAPI` in Ihre JDBC-Verbindungszeichenfolge ein. Informationen zum JDBC-Treiber finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC.
- Secrets Manager – So verwenden Sie das Athena-Federated-Query-Feature mit AWS Secrets Manager müssen Sie einen privaten Amazon-VPC-Endpunkt für Secrets Manager konfigurieren. Weitere Informationen finden Sie unter Erstellen eines privaten Secrets-Manager-VPC-Endpunkts im Benutzerhandbuch für AWS Secrets Manager.

Datenquellen-Connectors benötigen möglicherweise Zugriff auf die folgenden Ressourcen, um ordnungsgemäß zu funktionieren. Wenn Sie einen vorab erstellten Connector verwenden, müssen Sie die Informationen für den Connector prüfen, um sicherzustellen, dass Ihre VPC korrekt konfiguriert ist. Sie müssen außerdem sicherstellen, dass IAM-Prinzipale, die Abfragen ausführen und Connectors erstellen, die nötigen Berechtigungen für die erforderlichen Aktionen besitzen. Weitere Informationen finden Sie unter Beispiel für IAM-Berechtigungsrichtlinien zum Zulassen von Athena Federated Query.

- Amazon S3 – Zusätzlich zum Schreiben von Abfrageergebnissen zum Athena-Abfrageergebnisspeicherort in Amazon S3 schreiben Daten-Connectors auch zu einem Spill-Bucket in Amazon S3. Konnektivität und Berechtigungen für diesen Amazon-S3-Standort sind erforderlich.

- Athena – Datenquellen benötigen Konnektivität mit Athena und umgekehrt, um den Abfragestatus zu prüfen und ein übermäßiges Scannen zu verhindern.
- AWS Glue Data Catalog – Es sind Verbindungen und Berechtigungen erforderlich, wenn Ihr Connector Datenkatalog für ergänzende oder primäre Metadaten verwendet.

Videos

Sehen Sie sich die folgenden Videos an, um mehr über die Verwendung von Athena Federated Query zu erfahren.

Video: Analysieren der Ergebnisse von Federated Query in Amazon Athena in Amazon QuickSight

Das folgende Video veranschaulicht, wie Sie Ergebnisse einer Athena-Verbundabfrage in Amazon QuickSight analysieren.

Analysieren der Ergebnisse von Verbundabfragen in Amazon Athena in Amazon QuickSight

Video: Game Analytics Pipeline

Das folgende Video zeigt, wie Sie eine skalierbare Serverless-Data-Pipeline bereitstellen, um Telemetriedaten aus Spielen und Diensten mithilfe von Amazon-Athena-Verbundabfragen aufzunehmen, zu speichern und zu analysieren.

Spieleanalytik-Pipeline

Verfügbare Datenquellenkonnektoren

In diesem Abschnitt werden vorab erstellte Athena-Datenquellen-Konnektors aufgeführt, mit denen Sie zahlreiche Datenquellen außerhalb von Amazon S3 abfragen können. Um einen Konnektor in Ihren Athena-Abfragen zu verwenden, konfigurieren Sie ihn und stellen ihn in Ihrem Konto bereit.

Überlegungen und Einschränkungen

- Für einige vorgefertigte Konnektors müssen Sie eine VPC und eine Sicherheitsgruppe erstellen, bevor Sie den Konnektor verwenden können. Informationen zum Erstellen eigener Subnetze finden Sie unter Erstellen einer VPC für einen Datenquellen-Connector.
- Um die Athena Federated Query-Funktion mit verwenden zu können AWS Secrets Manager, müssen Sie einen privaten Amazon VPC-Endpunkt für Secrets Manager konfigurieren. Weitere Informationen finden Sie unter Erstellen eines privaten Secrets-Manager-VPC-Endpunkts im Benutzerhandbuch für AWS Secrets Manager .

- Bei Konnektoren, die kein Prädikat-Pushdown unterstützen, dauert die Ausführung von Abfragen, die ein Prädikat enthalten, deutlich länger. Bei kleinen Datensätzen werden nur sehr wenige Daten gescannt, und Abfragen dauern durchschnittlich etwa 2 Minuten. Bei großen Datensätzen kann es jedoch zu einer Zeitüberschreitung bei vielen Abfragen kommen.
- Einige föderierte Datenquellen verwenden Terminologie, um auf Datenobjekte zu verweisen, die sich von Athena unterscheiden. Weitere Informationen finden Sie unter Qualifizierer für Athena und verbundene Tabellennamen.
- Bei Konnektoren, die beim Auflisten von Tabellen keine Paginierung unterstützen, kann es beim Webservice zu einem Timeout kommen, wenn Ihre Datenbank viele Tabellen und Metadaten enthält. Die folgenden Konnektoren bieten Unterstützung für die Paginierung von Auflistungstabellen:
  - DocumentDB
  - DynamoDB
  - MySQL
  - OpenSearch
  - Oracle
  - PostgreSQL
  - Redshift
  - SQL Server

Zusätzliche Informationen

- Hinweise zum Bereitstellen eines Athena-Datenquellen-Konnektors finden Sie unter Datenquellen-Konnektor bereitstellen.
- Informationen zu Abfragen, die Athena-Datenquellen-Konnektors verwenden, finden Sie unter Verbundabfragen ausführen.
- Ausführliche Informationen zu den Athena-Datenquellenconnectors finden Sie unter Verfügbare Konnektoren unter GitHub.

Athena-Datenquellen-Konnektors

- Amazon Athena Azure Data Lake Storage (ADLS) Gen2-Konnektor
- Amazon Athena Azure Synapse Konnektor
- Amazon Athena Cloudera Hive Konnektor

- Amazon Athena Cloudera Impala Connector
- Amazon Athena CloudWatch Konnektor
- Amazon Athena CloudWatch Metrics Konnektor
- Amazon Athena AWS CMDB Konnektor
- Amazon-Athena-IBM-Db2-Konnektor
- Amazon Athena IBM Db2 AS/400-Anschluss (Db2 iSeries)
- Amazon-Athena-DocumentDB-Konnektor
- Amazon Athena DynamoDB Konnektor
- Amazon Athena Google-Konnektor BigQuery
- Google-Cloud-Storage-Konnektor für Amazon Athena
- Amazon Athena HBase Konnektor
- Amazon Athena Hortonworks Konnektor
- Apache-Kafka-Konnektor von Amazon Athena
- Amazon-Athena-MSK-Konnektor
- Amazon Athena MySQL Konnektor
- Amazon Athena Neptune Konnektor
- Amazon Athena OpenSearch Konnektor
- Amazon Athena Oracle Konnektor
- Amazon Athena PostgreSQL Konnektor
- Amazon Athena Redis Konnektor
- Amazon Athena Redshift Konnektor
- Amazon Athena SAP HANA Konnektor
- Amazon Athena Snowflake Konnektor
- Amazon Athena Microsoft SQL Server Konnektor
- Amazon Athena Teradata Konnektor
- Amazon Athena Timestream Konnektor
- Amazon Athena TPC Benchmark DS (TPC-DS) Konnektor
- Amazon Athena Vertica Konnektor

> **Note**
>
> Die AthenaJdbcConnector(neueste Version 2022.4.1) ist veraltet. Verwenden Sie stattdessen einen datenbankspezifischen Konnektor wie den für MySQL, Redshift oder PostgreSQL.

Amazon Athena Azure Data Lake Storage (ADLS) Gen2-Konnektor

Der Amazon-Athena-Konnektor für Azure Data Lake Storage (ADLS) Gen2 ermöglicht es Amazon-Athena, SQL-Abfragen für Daten auszuführen, die in ADLS gespeichert sind. Athena kann nicht direkt auf gespeicherte Dateien im Data Lake zugreifen.

- Workflow – Der Konnektor implementiert die JDBC-Schnittstelle, die den `com.microsoft.sqlserver.jdbc.SQLServerDriver`-Treiber verwendet. Der Konnektor leitet Abfragen an die Azure-Synapse-Engine weiter, die dann auf den Data Lake zugreift.
- Datenverarbeitung und S3 – Normalerweise fragt der Lambda-Konnektor Daten direkt ab, ohne sie an Amazon S3 zu übertragen. Wenn die von der Lambda-Funktion zurückgegebenen Daten jedoch die Lambda-Grenzwerte überschreiten, werden die Daten in den von Ihnen angegebenen Amazon-S3-Spill-Bucket geschrieben, sodass Athena den Überschuss lesen kann.
- AAD-Authentifizierung – AAD kann als Authentifizierungsmethode für den Azure Synapse-Konnektor verwendet werden. Um AAD verwenden zu können, muss die JDBC-Verbindungszeichenfolge, die der Konnektor verwendet, die URL-Parameter `authentication=ActiveDirectoryServicePrincipal`, `AADSecurePrincipalId` und `AADSecurePrincipalSecret` enthalten. Diese Parameter können entweder direkt oder von Secrets Manager übergeben werden.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Einschränkungen

- Schreiboperationen wie DDL werden nicht unterstützt.

- In einem Multiplexer-Setup werden der Überlauf-Bucket und das Präfix von allen Datenbank-Instances gemeinsam genutzt.
- Alle relevanten Lambda-Grenzwerte. Weitere Informationen finden Sie unter Lambda quotas (Lambda-Kontingente) im AWS Lambda -Entwicklerhandbuch.
- Datums- und Zeitstempeldatentypen in Filterbedingungen müssen in geeignete Datentypen umgewandelt werden.

Bedingungen

Die folgenden Begriffe beziehen sich auf den Gen2-Konnektor von Azure Data Lake Storage.

- Datenbank-Instance – Jede Instance einer Datenbank, die On-Premises, in Amazon EC2 oder auf Amazon RDS bereitgestellt wird.
- Handler – Ein Lambda-Handler, der auf Ihre Datenbank-Instance zugreift. Ein Handler kann für Metadaten oder für Datensätze verwendet werden.
- Metadaten-Handler – Ein Lambda-Handler, der Metadaten von Ihrer Datenbank-Instance abruft.
- Record Handler – Ein Lambda-Handler, der Datensätze aus Ihrer Datenbank-Instance abruft.
- Composite Handler – Ein Lambda-Handler, der sowohl Metadaten als auch Datensätze aus Ihrer Datenbank-Instance abruft.
- Eigenschaft oder Parameter – Eine Datenbankeigenschaft, die von Handlern zum Extrahieren von Datenbankinformationen verwendet wird. Sie konfigurieren diese Eigenschaften als Lambda-Umgebungsvariablen.
- Verbindungszeichenfolge – Eine Textzeichenfolge, die verwendet wird, um eine Verbindung zu einer Datenbank-Instance herzustellen.
- Katalog — Ein nicht bei Athena registrierter AWS Glue Katalog, der ein erforderliches Präfix für die `connection_string` Immobilie ist.
- Multiplex-Handler – Ein Lambda-Handler, der mehrere Datenbankverbindungen akzeptieren und verwenden kann.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den Gen2-Konnektor von Azure Data Lake Storage zu konfigurieren.

Verbindungszeichenfolge

Verwenden Sie eine JDBC-Verbindungszeichenfolge im folgenden Format, um eine Verbindung zu einer Datenbank-Instance herzustellen.

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

Verwenden eines Multiplexing-Handlers

Sie können einen Multiplexer verwenden, um mit einer einzigen Lambda-Funktion eine Verbindung zu mehreren Datenbank-Instances herzustellen. Anfragen werden anhand des Katalognamens weitergeleitet. Verwenden Sie die folgenden Klassen in Lambda.

Handler	Klasse
Composite Handler	`DataLakeGen2MuxCompositeHandler`
Metadaten-Handler	`DataLakeGen2MuxMetadataHandler`
Record Handler	`DataLakeGen2MuxRecordHandler`

Multiplex-Handler-Parameter

Parameter	Beschreibung
$*catalog*_connection_string	Erforderlich Eine Verbindungszeichenfolge einer Datenbank-Instance. Stellen Sie der Umgebungsvariablen den Namen des in Athena verwendeten Katalogs voran. Wenn zum Beispiel der bei Athena registrierte Katalog `mydatalakegentwocatalog` ist, dann lautet der Name der Umgebungsvariablen `mydatalakegentwocatalog_connection_string` .
`default`	Erforderlich Die standardmäßige Verbindungszeichenfolge. Diese Zeichenfolge wird verwendet, wenn der Katalog `lambda:${` *AWS_LAMBDA_FUNCTION_NAME* } ist.

Die folgenden Beispieleigenschaften beziehen sich auf eine DataLakeGen 2-MUX-Lambda-Funktion, die zwei Datenbankinstanzen unterstützt: `datalakegentwo1` (Standard) und. `datalakegentwo2`

Eigenschaft	Wert
`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 }`

Bereitstellen von Anmeldeinformationen

Um einen Benutzernamen und ein Kennwort für Ihre Datenbank in Ihrer JDBC-Verbindungszeichenfolge anzugeben, können Sie Eigenschaften von Verbindungszeichenfolgen oder AWS Secrets Manager verwenden.

- Verbindungszeichenfolge – Ein Benutzername und ein Kennwort können als Eigenschaften in der JDBC-Verbindungszeichenfolge angegeben werden.

  > **Important**
  >
  > Als bewährte Sicherheitsmethode sollten Sie keine fest kodierten Anmeldeinformationen in Ihren Umgebungsvariablen oder Verbindungszeichenfolgen verwenden. Informationen zum Verschieben Ihrer hartcodierten Geheimnisse nach finden Sie im AWS Secrets Manager Benutzerhandbuch unter Verschieben von hartcodierten AWS Secrets Manager Geheimnissen nach.AWS Secrets Manager

- AWS Secrets Manager— Um die Athena Federated Query-Funktion verwenden zu können AWS Secrets Manager, muss die mit Ihrer Lambda-Funktion verbundene VPC über Internetzugang oder einen VPC-Endpunkt verfügen, um eine Verbindung zu Secrets Manager herzustellen.

Sie können den Namen eines Geheimnisses in AWS Secrets Manager Ihre JDBC-Verbindungszeichenfolge eingeben. Der Konnektor ersetzt den geheimen Namen durch `username`- und `password`-Werte von Secrets Manager.

Für Amazon RDS-Datenbank-Instances ist diese Unterstützung eng integriert. Wenn Sie Amazon RDS verwenden, empfehlen wir dringend, eine Rotation der Anmeldeinformationen zu verwenden AWS Secrets Manager . Wenn Ihre Datenbank Amazon RDS nicht verwendet, speichern Sie die Anmeldeinformationen als JSON im folgenden Format:

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

Beispiel einer Verbindungszeichenfolge mit einem geheimen Namen

Die folgende Zeichenfolge hat den geheimen Namen `${secret1_name}`.

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

Der Konnektor verwendet den geheimen Namen, um Secrets abzurufen und den Benutzernamen und das Kennwort bereitzustellen, wie im folgenden Beispiel gezeigt.

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

Verwenden eines einzelnen Verbindungs-Handlers

Sie können die folgenden Einzelverbindungsmetadaten und Datensatzhandler verwenden, um eine Verbindung zu einer einzelnen Gen2-Instance von Azure Data Lake Storage herzustellen.

Handler-Typ	Klasse
Composite Handler	`DataLakeGen2CompositeHandler`
Metadaten-Handler	`DataLakeGen2MetadataHandler`
Record Handler	`DataLakeGen2RecordHandler`

Parameter für Einzelverbindungs-Handler

Parameter	Beschreibung
`default`	Erforderlich Die standardmäßige Verbindungszeichenfolge.

Die Einzelverbindungs-Handler unterstützen eine Datenbank-Instance und müssen einen `default`-Verbindungszeichenfolgenparameter bereitstellen. Alle anderen Verbindungszeichenfolgen werden ignoriert.

Die folgende Beispieleigenschaft gilt für eine einzelne Gen2-Instance von Azure Data Lake Storage, die von einer Lambda-Funktion unterstützt wird.

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

Überlauf-Parameter

Das Lambda-SDK kann Daten an Amazon S3 übertragen. Alle Datenbank-Instances, auf die mit derselben Lambda-Funktion zugegriffen wird, werden an denselben Speicherort verschoben.

Parameter	Beschreibung
`spill_bucket`	Erforderlich Überlauf-Bucket-Name.
`spill_prefix`	Erforderlich Schlüssel-Prefix für den Überlauf-Bucket.
`spill_put_request_headers`	(Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Weitere mögliche Header finden Sie PutObject in der Amazon Simple Storage Service API-Referenz.

Datentypunterstützung

Die folgende Tabelle enthält die entsprechenden Datentypen für ADLS Gen2 und Arrow.

ADLS Gen2	Arrow
Bit	TINYINT
tinyint	SMALLINT
smallint	SMALLINT
int	INT
bigint	BIGINT
Dezimalwert	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	Arrow
char[n]	VARCHAR
varchar[n/max]	VARCHAR

Partitionen und Splits

Azure Data Lake Storage Gen2 verwendet Hadoop-kompatiblen Gen2-Blob-Speicher zum Speichern von Datendateien. Die Daten aus diesen Dateien werden von der Azure-Synapse-Engine abgefragt. Die Azure-Synapse-Engine behandelt in Dateisystemen gespeicherte Gen2-Daten als externe Tabellen. Die Partitionen werden basierend auf dem Datentyp implementiert. Wenn die Daten bereits innerhalb des Gen 2-Speichersystems partitioniert und verteilt wurden, ruft der Konnektor die Daten als Single Split ab.

Leistung

Der Gen2-Konnektor von Azure Data Lake Storage zeigt eine langsamere Abfrageleistung, wenn mehrere Abfragen gleichzeitig ausgeführt werden, und unterliegt einer Drosselung.

Der Athena-Gen2-Konnektor von Azure Data Lake Storage führt einen Prädikat-Pushdown durch, um die von der Abfrage durchsuchten Daten zu reduzieren. Einfache Prädikate und komplexe Ausdrücke werden an den Konnektor übertragen, um die Menge der gescannten Daten zu reduzieren und die Laufzeit der Abfrageausführung zu verkürzen.

Prädikate

Ein Prädikat ist ein Ausdruck in der `WHERE`-Klausel einer SQL-Abfrage, der einen booleschen Wert ergibt und Zeilen auf der Grundlage mehrerer Bedingungen filtert. Der Athena-Gen2-Konnektor von Azure Data Lake Storage kann diese Ausdrücke kombinieren und sie direkt an Azure Data Lake Storage Gen2 weiterleiten, um die Funktionalität zu erweitern und die Menge der gescannten Daten zu reduzieren.

Die folgenden Athena-Gen2-Konnektor-Operatoren von Azure Data Lake Storage unterstützen den Prädikat-Pushdown:

- Boolean: UND, ODER, NICHT
- Gleichheit: GLEICH, NICHT-GLEICH, WENIGER_ALS, WENIGER_ODER_GLEICH, GRÖSSER_ALS, GRÖSSER_ODER_GLEICH, NULL_WENN, IST_NULL

- Arithmetik: ADDIEREN, SUBTRAHIEREN, MULTIPLIZIEREN, DIVIDIEREN, MODULIEREN, NEGIEREN
- Andere: WIE_MUSTER, IN

Beispiel für einen kombinierten Pushdown

Kombinieren Sie für erweiterte Abfragefunktionen die Pushdown-Typen wie im folgenden Beispiel:

```
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%');
```

Lizenzinformationen

Durch die Verwendung dieses Connectors erkennen Sie die Einbindung von Komponenten von Drittanbietern an. Eine Liste dieser Komponenten finden Sie in der Datei pom.xml für diesen Connector und stimmen den Bedingungen der jeweiligen Drittanbieterlizenzen zu, die in der Datei LICENSE.txt auf GitHub .com enthalten sind.

Weitere Informationen finden Sie auch unter

Die neuesten Informationen zur JDBC-Treiberversion finden Sie in der Datei pom.xml für den Azure Data Lake Storage Gen2-Connector auf .com. GitHub

Weitere Informationen zu diesem Connector finden Sie auf der entsprechenden Website unter .com. GitHub

Amazon Athena Azure Synapse Konnektor

Der Amazon-Athena-Konnektor für Azure-Synapse-Analytik ermöglicht Amazon Athena, SQL-Abfragen für Ihre Azure-Synapse gespeicherte Daten mit JDBC auszuführen.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Einschränkungen

- Schreiboperationen wie DDL werden nicht unterstützt.
- In einem Multiplexer-Setup werden der Überlauf-Bucket und das Präfix von allen Datenbank-Instances gemeinsam genutzt.
- Alle relevanten Lambda-Grenzwerte. Weitere Informationen finden Sie unter Lambda quotas (Lambda-Kontingente) im AWS Lambda -Entwicklerhandbuch.
- Unter Filterbedingungen müssen Sie die `Date`- und `Timestamp`-Datentypen in den entsprechenden Datentyp umwandeln.
- So suchen Sie nach negativen Werten des Typs `Real` und `Float`, verwenden Sie den <=- oder >=-Operator.
- Die Datentypen `binary`, `varbinary`, `image` und `rowversion` werden nicht unterstützt.

Bedingungen

Die folgenden Begriffe beziehen sich auf den Synapse-Konnektor.

- Datenbank-Instance – Jede Instance einer Datenbank, die On-Premises, in Amazon EC2 oder auf Amazon RDS bereitgestellt wird.
- Handler – Ein Lambda-Handler, der auf Ihre Datenbank-Instance zugreift. Ein Handler kann für Metadaten oder für Datensätze verwendet werden.
- Metadaten-Handler – Ein Lambda-Handler, der Metadaten von Ihrer Datenbank-Instance abruft.
- Record Handler – Ein Lambda-Handler, der Datensätze aus Ihrer Datenbank-Instance abruft.
- Composite Handler – Ein Lambda-Handler, der sowohl Metadaten als auch Datensätze aus Ihrer Datenbank-Instance abruft.
- Eigenschaft oder Parameter – Eine Datenbankeigenschaft, die von Handlern zum Extrahieren von Datenbankinformationen verwendet wird. Sie konfigurieren diese Eigenschaften als Lambda-Umgebungsvariablen.
- Verbindungszeichenfolge – Eine Textzeichenfolge, die verwendet wird, um eine Verbindung zu einer Datenbank-Instance herzustellen.
- Katalog — Ein nicht bei Athena registrierter AWS Glue Katalog, der ein erforderliches Präfix für die `connection_string` Immobilie ist.
- Multiplex-Handler – Ein Lambda-Handler, der mehrere Datenbankverbindungen akzeptieren und verwenden kann.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den Synapse-Konnektor zu konfigurieren.

Verbindungszeichenfolge

Verwenden Sie eine JDBC-Verbindungszeichenfolge im folgenden Format, um eine Verbindung zu einer Datenbank-Instance herzustellen.

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

Verwenden eines Multiplexing-Handlers

Sie können einen Multiplexer verwenden, um mit einer einzigen Lambda-Funktion eine Verbindung zu mehreren Datenbank-Instances herzustellen. Anfragen werden anhand des Katalognamens weitergeleitet. Verwenden Sie die folgenden Klassen in Lambda.

Handler	Klasse
Composite Handler	`SynapseMuxCompositeHandler`
Metadaten-Handler	`SynapseMuxMetadataHandler`
Record Handler	`SynapseMuxRecordHandler`

Multiplex-Handler-Parameter

Parameter	Beschreibung
$*catalog*_connection_string	Erforderlich Eine Verbindungszeichenfolge einer Datenbank-Instance. Stellen Sie der Umgebungsvariablen den Namen des in Athena verwendeten Katalogs voran. Wenn zum Beispiel der bei Athena registrierte Katalog `mysynapsecatalog` ist, dann lautet der Name der Umgebungsvariablen `mysynapsecatalog_connection_string` .

Parameter	Beschreibung
`default`	Erforderlich Die standardmäßige Verbindungszeichenfolge. Diese Zeichenfolge wird verwendet, wenn der Katalog `lambda:${` *AWS_LAMBDA_FUNCTION_NAME* `}` ist.

Die folgenden Beispieleigenschaften gelten für eine Synapse MUX Lambda-Funktion, die zwei Datenbank-Instances unterstützt: `synapse1` (die Standardeinstellung) und `synapse2`.

Eigenschaft	Wert
`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*`}`

Bereitstellen von Anmeldeinformationen

Um einen Benutzernamen und ein Kennwort für Ihre Datenbank in Ihrer JDBC-Verbindungszeichenfolge anzugeben, können Sie Eigenschaften von Verbindungszeichenfolgen oder AWS Secrets Manager verwenden.

- Verbindungszeichenfolge – Ein Benutzername und ein Kennwort können als Eigenschaften in der JDBC-Verbindungszeichenfolge angegeben werden.

> **Important**
>
> Als bewährte Sicherheitsmethode sollten Sie keine fest kodierten Anmeldeinformationen in Ihren Umgebungsvariablen oder Verbindungszeichenfolgen verwenden. Informationen

zum Verschieben von hartcodierten Geheimnissen nach finden Sie im AWS Secrets Manager Benutzerhandbuch unter Verschieben von hartcodierten AWS Secrets Manager Geheimnissen nach.AWS Secrets Manager

- AWS Secrets Manager— Um die Athena Federated Query-Funktion verwenden zu können AWS Secrets Manager, muss die mit Ihrer Lambda-Funktion verbundene VPC über Internetzugang oder einen VPC-Endpunkt verfügen, um eine Verbindung zu Secrets Manager herzustellen.

  Sie können den Namen eines Geheimnisses in AWS Secrets Manager Ihre JDBC-Verbindungszeichenfolge eingeben. Der Konnektor ersetzt den geheimen Namen durch `username`- und `password`-Werte von Secrets Manager.

  Für Amazon RDS-Datenbank-Instances ist diese Unterstützung eng integriert. Wenn Sie Amazon RDS verwenden, empfehlen wir dringend, eine Rotation der Anmeldeinformationen zu verwenden AWS Secrets Manager . Wenn Ihre Datenbank Amazon RDS nicht verwendet, speichern Sie die Anmeldeinformationen als JSON im folgenden Format:

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

Beispiel einer Verbindungszeichenfolge mit einem Secret-Namen

Die folgende Zeichenfolge hat den geheimen Namen ${secret_name}.

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

Der Konnektor verwendet den geheimen Namen, um Secrets abzurufen und den Benutzernamen und das Kennwort bereitzustellen, wie im folgenden Beispiel gezeigt.

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

Verwenden eines einzelnen Verbindungs-Handlers

Sie können die folgenden Einzelverbindungsmetadaten und Record Handler verwenden, um eine Verbindung zu einer einzelnen Synapse-Instance herzustellen.

Handler-Typ	Klasse
Composite Handler	`SynapseCompositeHandler`
Metadaten-Handler	`SynapseMetadataHandler`
Record Handler	`SynapseRecordHandler`

Parameter für Einzelverbindungs-Handler

Parameter	Beschreibung
`default`	Erforderlich Die standardmäßige Verbindungszeichenfolge.

Die Einzelverbindungs-Handler unterstützen eine Datenbank-Instance und müssen einen `default`-Verbindungszeichenfolgenparameter bereitstellen. Alle anderen Verbindungszeichenfolgen werden ignoriert.

Die folgende Beispieleigenschaft gilt für eine einzelne Synapse-Instance, die von einer Lambda-Funktion unterstützt wird.

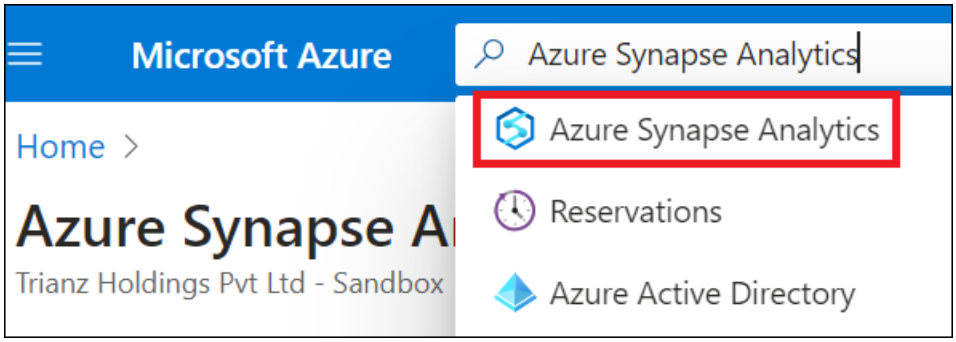
Eigenschaft	Wert
`defa`	`synapse://jdbc:sqlserver://hostname:port;databaseName=` *`<database_name>`* `;${`*`secret_name`*` }`

Active-Directory-Authentifizierung konfigurieren

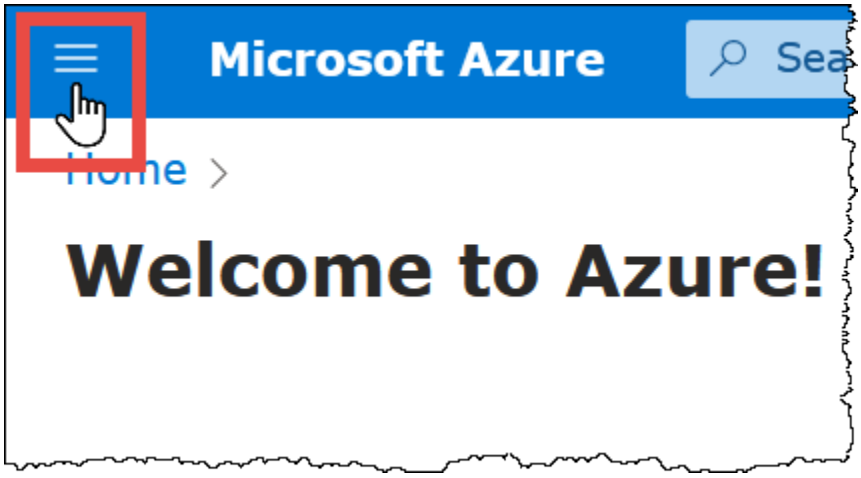
Der Konnektor von Amazon Athena Azure Synapse unterstützt die Microsoft-Active-Directory-Authentifizierung. Bevor Sie beginnen, müssen Sie im Microsoft Azure-Portal einen Benutzer mit Administratorrechten konfigurieren und diesen dann AWS Secrets Manager zum Erstellen eines Geheimnisses verwenden.

So legen Sie den administrativen Active-Directory-Benutzer fest

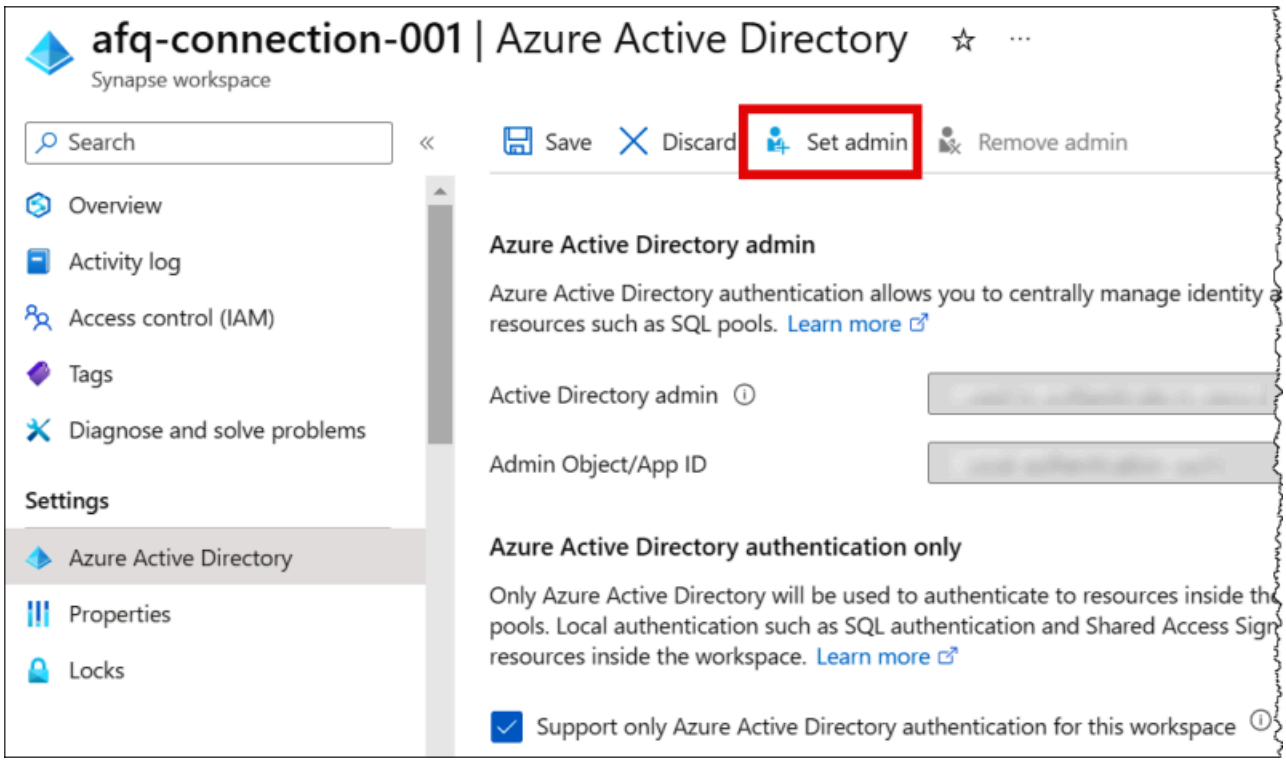
1. Melden Sie sich mit einem Konto mit Administratorrechten beim Microsoft-Azure-Portal unter https://portal.azure.com/ an.
2. Geben Sie in das Suchfeld Azure Synapse Analytics ein und wählen Sie dann Azure Synapse Analytics aus.



3. Öffnen Sie das Menü auf der linken Seite.



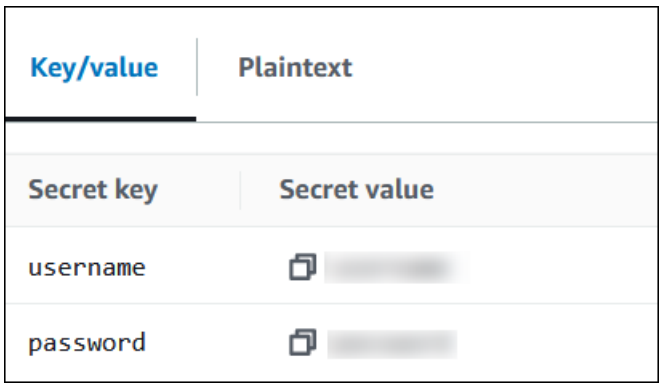
4. Wählen Sie im Navigationsbereich Azure Active Directory.
5. Stellen Sie auf der Registerkarte Administrator festlegen den Active-Directory-Administrator auf einen neuen oder vorhandenen Benutzer ein.



6. Speichern Sie den Administratorbenutzernamen und das Kennwort unter. AWS Secrets Manager Informationen zum Erstellen eines Geheimnisses im Secrets Manager finden Sie unter Erstellen eines AWS Secrets Manager -Geheimnisses.

So zeigen Sie Ihr Geheimnis in Secrets Manager an

1. Öffnen Sie die Secrets-Manager-Konsole unterhttps://console.aws.amazon.com/secretsmanager/.
2. Wählen Sie im Navigationsbereich Secrets (Geheimnisse).
3. Wählen Sie auf der Seite Secrets (Geheimnisse) den Link zu Ihrem Geheimnis aus.
4. Wählen Sie auf der Detailseite für Ihr Geheimnis die Option Retrieve secret value (Geheimniswert abrufen).



Verbindungszeichenfolge ändern

Um die Active-Directory-Authentifizierung für den Konnektor zu aktivieren, ändern Sie die Verbindungszeichenfolge mit der folgenden Syntax:

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

Verwenden ActiveDirectoryServicePrincipal

Der Konnektor von Amazon Athena Azure Synapse unterstützt ebenfalls `ActiveDirectoryServicePrincipal`. Um es zu aktivieren, ändern Sie die Zeichenfolge wie folgt.

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

Geben Sie für `secret_name` die Anwendungs- oder Client-ID als Benutzernamen und das Geheimnis einer Dienstprinzipalidentität im Passwort an.

Überlauf-Parameter

Das Lambda-SDK kann Daten an Amazon S3 übertragen. Alle Datenbank-Instances, auf die mit derselben Lambda-Funktion zugegriffen wird, werden an denselben Speicherort verschoben.

Parameter	Beschreibung
`spill_bucket`	Erforderlich Überlauf-Bucket-Name.

Parameter	Beschreibung
`spill_prefix`	Erforderlich Schlüssel-Prefix für den Überlauf-Bucket.
`spill_put_request_headers`	(Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Weitere mögliche Header finden Sie PutObject in der Amazon Simple Storage Service API-Referenz.

Datentypunterstützung

Die folgende Tabelle zeigt die entsprechenden Datentypen für Synapse und Apache Arrow.

Synapse	Arrow
Bit	TINYINT
tinyint	SMALLINT
smallint	SMALLINT
int	INT
bigint	BIGINT
Dezimalwert	DECIMAL
numeric	FLOAT8
smallmoney	FLOAT8
money	DECIMAL
float[24]	FLOAT4
float[53]	FLOAT8
real	FLOAT4

Synapse	Arrow
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

Partitionen und Splits

Eine Partition wird durch eine einzelne Partitionsspalte vom Typ `varchar` dargestellt. Synapse unterstützt die Bereichspartitionierung, sodass die Partitionierung durch Extrahieren der Partitionsspalte und des Partitionsbereichs aus den Synapse-Metadatentabellen implementiert wird. Diese Bereichswerte werden verwendet, um die Splits zu erstellen.

Leistung

Die Auswahl einer Teilmenge von Spalten verlangsamt die Abfragelaufzeit erheblich. Der Konnektor zeigt aufgrund der Gleichzeitigkeit eine erhebliche Drosselung.

Der Athena-Synapse-Konnektor führt einen Prädikat-Pushdown durch, um die von der Abfrage durchsuchten Daten zu reduzieren. Einfache Prädikate und komplexe Ausdrücke werden an den Konnektor übertragen, um die Menge der gescannten Daten zu reduzieren und die Laufzeit der Abfrageausführung zu verkürzen.

Prädikate

Ein Prädikat ist ein Ausdruck in der `WHERE`-Klausel einer SQL-Abfrage, der einen booleschen Wert ergibt und Zeilen auf der Grundlage mehrerer Bedingungen filtert. Der Athena-Synapse-Konnektor kann diese Ausdrücke kombinieren und sie direkt an Synapse weiterleiten, um die Funktionalität zu verbessern und die Menge der gescannten Daten zu reduzieren.

Die folgenden Athena-Synapse-Konnektor-Operatoren unterstützen Prädikat-Pushdown:

- Boolean: UND, ODER, NICHT
- Gleichheit: GLEICH, NICHT-GLEICH, WENIGER_ALS, WENIGER_ODER_GLEICH, GRÖSSER_ALS, GRÖSSER_ODER_GLEICH, NULL_WENN, IST_NULL
- Arithmetik: ADDIEREN, SUBTRAHIEREN, MULTIPLIZIEREN, DIVIDIEREN, MODULIEREN, NEGIEREN
- Andere: WIE_MUSTER, IN

Beispiel für einen kombinierten Pushdown

Kombinieren Sie für erweiterte Abfragefunktionen die Pushdown-Typen wie im folgenden Beispiel:

```
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%');
```

Lizenzinformationen

Durch die Verwendung dieses Connectors erkennen Sie die Einbindung von Komponenten von Drittanbietern an. Eine Liste dieser Komponenten finden Sie in der Datei pom.xml für diesen Connector und stimmen den Bedingungen der jeweiligen Drittanbieterlizenzen zu, die in der Datei LICENSE.txt auf GitHub .com enthalten sind.

Weitere Informationen finden Sie auch unter

- Einen Artikel, der zeigt, wie Amazon QuickSight und Amazon Athena Federated Query verwendet werden können, um Dashboards und Visualisierungen für Daten zu erstellen, die in Microsoft Azure Synapse-Datenbanken gespeichert sind, finden Sie unter Durchführen von Multi-Cloud-Analysen

mit Amazon QuickSight, Amazon Athena Federated Query und Microsoft Azure Synapse im Big Data-Blog.AWS

- Die neuesten Informationen zur JDBC-Treiberversion finden Sie in der Datei pom.xml für den Synapse-Connector auf .com. GitHub
- Weitere Informationen zu diesem Connector finden Sie auf der entsprechenden Website unter .com. GitHub

Amazon Athena Cloudera Hive Konnektor

Der Amazon Athena-Konnektor für Cloudera Hive ermöglicht es Athena, SQL-Abfragen auf der Hadoop-Verteilung von Cloudera Hive auszuführen. Der Konnektor wandelt Ihre Athena-SQL-Abfragen in ihre äquivalente HiveQL-Syntax um.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.
- Richten Sie eine VPC und eine Sicherheitsgruppe ein, bevor Sie diesen Konnektor verwenden. Weitere Informationen finden Sie unter Erstellen einer VPC für einen Datenquellen-Connector.

Einschränkungen

- Schreiboperationen wie DDL werden nicht unterstützt.
- In einem Multiplexer-Setup werden der Überlauf-Bucket und das Präfix von allen Datenbank-Instances gemeinsam genutzt.
- Alle relevanten Lambda-Grenzwerte. Weitere Informationen finden Sie unter Lambda quotas (Lambda-Kontingente) im AWS Lambda-Entwicklerhandbuch.

Bedingungen

Die folgenden Begriffe beziehen sich auf den Cloudera-Hive-Konnektor.

- Datenbank-Instance – Jede Instance einer Datenbank, die On-Premises, in Amazon EC2 oder auf Amazon RDS bereitgestellt wird.

- Handler – Ein Lambda-Handler, der auf Ihre Datenbank-Instance zugreift. Ein Handler kann für Metadaten oder für Datensätze verwendet werden.
- Metadaten-Handler – Ein Lambda-Handler, der Metadaten von Ihrer Datenbank-Instance abruft.
- Record Handler – Ein Lambda-Handler, der Datensätze aus Ihrer Datenbank-Instance abruft.
- Composite Handler – Ein Lambda-Handler, der sowohl Metadaten als auch Datensätze aus Ihrer Datenbank-Instance abruft.
- Eigenschaft oder Parameter – Eine Datenbankeigenschaft, die von Handlern zum Extrahieren von Datenbankinformationen verwendet wird. Sie konfigurieren diese Eigenschaften als Lambda-Umgebungsvariablen.
- Verbindungszeichenfolge – Eine Textzeichenfolge, die verwendet wird, um eine Verbindung zu einer Datenbank-Instance herzustellen.
- Katalog – Ein Nicht-AWS Glue-Katalog, der bei Athena registriert ist und ein erforderliches Präfix für die `connection_string`-Eigenschaft darstellt.
- Multiplex-Handler – Ein Lambda-Handler, der mehrere Datenbankverbindungen akzeptieren und verwenden kann.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den Cloudera-Hive-Konnektor zu konfigurieren.

Verbindungszeichenfolge

Verwenden Sie eine JDBC-Verbindungszeichenfolge im folgenden Format, um eine Verbindung zu einer Datenbank-Instance herzustellen.

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

Verwenden eines Multiplexing-Handlers

Sie können einen Multiplexer verwenden, um mit einer einzigen Lambda-Funktion eine Verbindung zu mehreren Datenbank-Instances herzustellen. Anfragen werden anhand des Katalognamens weitergeleitet. Verwenden Sie die folgenden Klassen in Lambda.

Handler	Klasse
Composite Handler	`HiveMuxCompositeHandler`

Handler	Klasse
Metadaten-Handler	`HiveMuxMetadataHandler`
Record Handler	`HiveMuxRecordHandler`

Multiplex-Handler-Parameter

Parameter	Beschreibung
`$`*`catalog`*`_connection_string`	Erforderlich. Eine Verbindungszeichenfolge einer Datenbank-Instance. Stellen Sie der Umgebungsvariablen den Namen des in Athena verwendeten Katalogs voran. Wenn zum Beispiel der bei Athena registrierte Katalog `myhivecatalog` ist, dann lautet der Name der Umgebungsvariablen `myhivecatalog_connection_string` .
`default`	Erforderlich. Die standardmäßige Verbindungszeichenfolge. Diese Zeichenfolge wird verwendet, wenn der Katalog `lambda:${`*`AWS_LAMBDA_FUNCTION_NAME`*`}` ist.

Die folgenden Beispieleigenschaften gelten für eine Hive MUX Lambda-Funktion, die zwei Datenbankinstanzen unterstützt: `hive1` (die Standardeinstellung) und `hive2`.

Property (Eigenschaft)	Wert
`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`

Bereitstellen von Anmeldeinformationen

Um einen Benutzernamen und ein Kennwort für Ihre Datenbank in Ihrer JDBC-Verbindungszeichenfolge anzugeben, können Sie Eigenschaften von Verbindungszeichenfolgen oder AWS Secrets Manager verwenden.

- Verbindungszeichenfolge – Ein Benutzername und ein Kennwort können als Eigenschaften in der JDBC-Verbindungszeichenfolge angegeben werden.

  > **Important**
  >
  > Als bewährte Sicherheitsmethode sollten Sie keine fest kodierten Anmeldeinformationen in Ihren Umgebungsvariablen oder Verbindungszeichenfolgen verwenden. Informationen zum Verschieben von hartkodierten Geheimnissen nach AWS Secrets Manager finden Sie unter Verschieben von hartkodierten Geheimnissen nach AWS Secrets Manager im AWS Secrets Manager-Benutzerhandbuch.

- AWS Secrets Manager – Um das Athena-Federated-Query-Feature mit AWS Secrets Manager zu verwenden, sollte die mit Ihrer Lambda-Funktion verbundene VPC über einen Internetzugang oder einen VPC-Endpunkt verfügen, um eine Verbindung zu Secrets Manager herzustellen.

  Sie können den Namen eines Secrets in AWS Secrets Manager in Ihrer JDBC-Verbindungszeichenfolge eingeben. Der Konnektor ersetzt den geheimen Namen durch `username`- und `password`-Werte von Secrets Manager.

  Für Amazon RDS-Datenbank-Instances ist diese Unterstützung eng integriert. Wenn Sie Amazon RDS verwenden, empfehlen wir dringend die Verwendung von AWS Secrets Manager und Wechsel der Anmeldeinformationen. Wenn Ihre Datenbank Amazon RDS nicht verwendet, speichern Sie die Anmeldeinformationen als JSON im folgenden Format:

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

Beispiel einer Verbindungszeichenfolge mit einem geheimen Namen

Die folgende Zeichenfolge hat den geheimen Namen `${Test/RDS/hive1}`.

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

Der Konnektor verwendet den geheimen Namen, um Secrets abzurufen und den Benutzernamen und das Kennwort bereitzustellen, wie im folgenden Beispiel gezeigt.

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

Derzeit erkennt der Cloudera-Hive-Konnektor die JDBC-Eigenschaften `UID` und `PWD`.

Verwenden eines einzelnen Verbindungs-Handlers

Sie können die folgenden Einzelverbindungsmetadaten und Record Handler verwenden, um eine Verbindung zu einer einzelnen Cloudera Hive-Instance herzustellen.

Handler-Typ	Klasse
Composite Handler	`HiveCompositeHandler`
Metadaten-Handler	`HiveMetadataHandler`
Record Handler	`HiveRecordHandler`

Parameter für Einzelverbindungs-Handler

Parameter	Beschreibung
`default`	Erforderlich. Die standardmäßige Verbindungszeichenfolge.

Die Einzelverbindungs-Handler unterstützen eine Datenbank-Instance und müssen einen `default`-Verbindungszeichenfolgenparameter bereitstellen. Alle anderen Verbindungszeichenfolgen werden ignoriert.

Die folgende Beispieleigenschaft gilt für eine einzelne Cloudera Hive-Instance, die von einer Lambda-Funktion unterstützt wird.

Property (Eigenschaft)	Wert
default	`hive://jdbc:hive2://hive1:10000/default?secret=${Test/RDS/hive1}`

Überlauf-Parameter

Das Lambda-SDK kann Daten an Amazon S3 übertragen. Alle Datenbank-Instances, auf die mit derselben Lambda-Funktion zugegriffen wird, werden an denselben Speicherort verschoben.

Parameter	Beschreibung
`spill_bucket`	Erforderlich. Überlauf-Bucket-Name.
`spill_prefix`	Erforderlich. Schlüssel-Prefix für den Überlauf-Bucket.
`spill_put_request_headers`	(Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Andere mögliche Header finden Sie unter PutObject in der API-Referenz zu Amazon Simple Storage Service.

Datentypunterstützung

Die folgende Tabelle enthält die entsprechenden Datentypen für JDBC, Cloudera Hive und Arrow.

JDBC	Cloudera Hive	Arrow
Boolesch	Boolesch	Bit
Ganzzahl	TINYINT	Tiny
Short	SMALLINT	Smallint

JDBC	Cloudera Hive	Arrow
Ganzzahl	INT	Int
Long	BIGINT	Bigint
float	float4	Float4
Double	float8	Float8
Datum	date	Dateday
Zeitstempel	timestamp	DateMilli
Zeichenfolge	VARCHAR	Varchar
Bytes	bytes	Varbinary
BigDecimal	Dezimal	Dezimal
ARRAY	N.z. (siehe Hinweis)	Auflisten

> **Note**
>
> Derzeit unterstützt Cloudera Hive nicht die Aggregattypen `ARRAY`, `MAP`, `STRUCT` oder `UNIONTYPE`. Spalten mit Aggregattypen werden als `VARCHAR`-Spalten in SQL behandelt.

Partitionen und Splits

Partitionen werden verwendet, um zu bestimmen, wie Splits für den Konnektor generiert werden. Athena konstruiert eine synthetische Säule vom Typ `varchar`, die das Partitionierungsschema für die Tabelle darstellt, das dem Konnektor beim Generieren von Splits hilft. Der Konnektor ändert nicht die eigentliche Tabellendefinition.

Leistung

Cloudera Hive unterstützt statische Partitionen. Der Athena-Cloudera-Hive-Konnektor kann Daten von diesen Partitionen parallel abrufen. Wenn Sie sehr große Datenmengen mit einheitlicher

Partitionsverteilung abfragen möchten, wird eine statische Partitionierung dringend empfohlen. Der Cloudera Hive-Konnektor ist aufgrund der Gleichzeitigkeit widerstandsfähig gegenüber Drosselung.

Der Athena-Cloudera-Hive-Konnektor führt einen Prädikat-Pushdown durch, um die von der Abfrage gescannten Daten zu verringern. `LIMIT`-Klauseln, einfache Prädikate und komplexe Ausdrücke werden an den Konnektor weitergegeben, um die Menge der gescannten Daten und die Laufzeit der Abfrage zu verringern.

LIMIT-Klauseln

Eine `LIMIT N`-Anweisung reduziert die von der Abfrage durchsuchten Daten. Mit `LIMIT N`-Pushdown gibt der Konnektor nur `N` Zeilen an Athena zurück.

Prädikate

Ein Prädikat ist ein Ausdruck in der `WHERE`-Klausel einer SQL-Abfrage, der einen booleschen Wert ergibt und Zeilen auf der Grundlage mehrerer Bedingungen filtert. Der Athena-Cloudera-Hive-Konnektor kann diese Ausdrücke kombinieren und sie direkt an Cloudera Hive weiterleiten, um die Funktionalität zu verbessern und die Menge der gescannten Daten zu reduzieren.

Die folgenden Operatoren des Athena-Cloudera-Hive-Konnektors unterstützen Prädikat-Pushdown:

- Boolean: UND, ODER, NICHT
- Gleichheit: GLEICH, NICHT GLEICH, WENIGER_ALS, WENIGER_ODER_GLEICH, GRÖSSER_ALS, GRÖSSER_ODER_GLEICH, IST_NULL
- Arithmetik: ADDIEREN, SUBTRAHIEREN, MULTIPLIZIEREN, DIVIDIEREN, MODULIEREN, NEGIEREN
- Andere: WIE_MUSTER, IN

Beispiel für einen kombinierten Pushdown

Kombinieren Sie für erweiterte Abfragefunktionen die Pushdown-Typen wie im folgenden Beispiel:

```
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;
```

Lizenzinformationen

Durch die Verwendung dieses Konnektors erkennen Sie die Aufnahme von Komponenten von Drittanbietern an. Eine Liste dieser Komponenten finden Sie in der pom.xml-Datei für diesen Konnektor. Zudem stimmen Sie den Bedingungen der jeweiligen Drittanbieterlizenzen zu, die in der LICENSE.txt-Datei auf GitHub.com aufgeführt werden.

Weitere Informationen finden Sie auch unter

Aktuelle Informationen zur JDBC-Treiberversion finden Sie in der pom.xml-Datei für den Cloudera-Hive-Konnektor auf GitHub.com.

Weitere Informationen zu diesem Konnektor finden Sie unter der entsprechenden Seite auf GitHub.com.

Amazon Athena Cloudera Impala Connector

Der Amazon Athena-Cloudera-Impala-Konnektor ermöglicht es Athena, SQL-Abfragen auf der Cloudera-Impala-Verteilung auszuführen. Der Konnektor wandelt Ihre Athena-SQL-Abfragen in die entsprechende Impala-Syntax um.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repositorybereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.
- Richten Sie eine VPC und eine Sicherheitsgruppe ein, bevor Sie diesen Konnektor verwenden. Weitere Informationen finden Sie unter Erstellen einer VPC für einen Datenquellen-Connector.

Einschränkungen

- Schreiboperationen wie DDL werden nicht unterstützt.
- In einem Multiplexer-Setup werden der Überlauf-Bucket und das Präfix von allen Datenbank-Instances gemeinsam genutzt.
- Alle relevanten Lambda-Grenzwerte. Weitere Informationen finden Sie unter Lambda quotas (Lambda-Kontingente) im AWS Lambda -Entwicklerhandbuch.

Bedingungen

Die folgenden Begriffe beziehen sich auf den Cloudera-Impala-Konnektor.

- Datenbank-Instance – Jede Instance einer Datenbank, die On-Premises, in Amazon EC2 oder auf Amazon RDS bereitgestellt wird.
- Handler – Ein Lambda-Handler, der auf Ihre Datenbank-Instance zugreift. Ein Handler kann für Metadaten oder für Datensätze verwendet werden.
- Metadaten-Handler – Ein Lambda-Handler, der Metadaten von Ihrer Datenbank-Instance abruft.
- Record Handler – Ein Lambda-Handler, der Datensätze aus Ihrer Datenbank-Instance abruft.
- Composite Handler – Ein Lambda-Handler, der sowohl Metadaten als auch Datensätze aus Ihrer Datenbank-Instance abruft.
- Eigenschaft oder Parameter – Eine Datenbankeigenschaft, die von Handlern zum Extrahieren von Datenbankinformationen verwendet wird. Sie konfigurieren diese Eigenschaften als Lambda-Umgebungsvariablen.
- Verbindungszeichenfolge – Eine Textzeichenfolge, die verwendet wird, um eine Verbindung zu einer Datenbank-Instance herzustellen.
- Katalog – Ein nichtAWS Glue katalogisiertes, bei Athena registriertes Präfix, das ein erforderliches Präfix für die -`connection_string`Eigenschaft ist.
- Multiplex-Handler – Ein Lambda-Handler, der mehrere Datenbankverbindungen akzeptieren und verwenden kann.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den Cloudera-Impala-Konnektor zu konfigurieren.

Verbindungszeichenfolge

Verwenden Sie eine JDBC-Verbindungszeichenfolge im folgenden Format, um eine Verbindung zu einem Impala-Cluster herzustellen.

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

Verwenden eines Multiplexing-Handlers

Sie können einen Multiplexer verwenden, um mit einer einzigen Lambda-Funktion eine Verbindung zu mehreren Datenbank-Instances herzustellen. Anfragen werden anhand des Katalognamens weitergeleitet. Verwenden Sie die folgenden Klassen in Lambda.

Handler	Klasse
Composite Handler	`ImpalaMuxCompositeHandler`
Metadaten-Handler	`ImpalaMuxMetadataHandler`
Record Handler	`ImpalaMuxRecordHandler`

Multiplex-Handler-Parameter

Parameter	Beschreibung
`$`*`catalog`*`_connection_string`	Erforderlich Eine Impala-Cluster-Verbindungszeichenfolge für einen Athena-Katalog. Stellen Sie der Umgebungsvariablen den Namen des in Athena verwendeten Katalogs voran. Wenn zum Beispiel der bei Athena registrierte Katalog `myimpalacatalog` ist, dann lautet der Name der Umgebungsvariablen `myimpalacatalog_connection_string` .
`default`	Erforderlich Die standardmäßige Verbindungszeichenfolge. Diese Zeichenfolge wird verwendet, wenn der Katalog `lambda:${`*`AWS_LAMBDA_FUNCTION_NAME`*`}` ist.

Die folgenden Beispieleigenschaften gelten für eine Impala MUX Lambda-Funktion, die zwei Datenbankinstanzen unterstützt: `impala1` (die Standardeinstellung) und `impala2`.

Eigenschaft	Wert
`default`	`impala://jdbc:impala://some.impala.host.name:21050/?${Test/impala1}`

Eigenschaft	Wert
`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`

Bereitstellen von Anmeldeinformationen

Um einen Benutzernamen und ein Kennwort für Ihre Datenbank in Ihrer JDBC-Verbindungszeichenfolge anzugeben, können Sie Eigenschaften von Verbindungszeichenfolgen oder AWS Secrets Managerverwenden.

- Verbindungszeichenfolge – Ein Benutzername und ein Kennwort können als Eigenschaften in der JDBC-Verbindungszeichenfolge angegeben werden.

  > **Important**
  >
  > Als bewährte Sicherheitsmethode sollten Sie keine fest kodierten Anmeldeinformationen in Ihren Umgebungsvariablen oder Verbindungszeichenfolgen verwenden. Informationen zum Verschieben Ihrer fest codierten Secrets in finden Sie AWS Secrets Managerunter Verschieben von fest codierten Secrets in AWS Secrets Manager im AWS Secrets Manager -Benutzerhandbuch.

- AWS Secrets Manager – Um die Athena-Federated-Query-Funktion mit zu verwenden AWS Secrets Manager, sollte die mit Ihrer Lambda-Funktion verbundene VPC über einen Internetzugang oder einen VPC-Endpunkt verfügen, um eine Verbindung zu Secrets Manager herzustellen.

  Sie können den Namen eines Secrets in AWS Secrets Manager in Ihrer JDBC-Verbindungszeichenfolge ablegen. Der Konnektor ersetzt den geheimen Namen durch `username`- und `password`-Werte von Secrets Manager.

  Für Amazon RDS-Datenbank-Instances ist diese Unterstützung eng integriert. Wenn Sie Amazon RDS verwenden, empfehlen wir dringend die Verwendung von AWS Secrets Manager und der Rotation von Anmeldeinformationen. Wenn Ihre Datenbank Amazon RDS nicht verwendet, speichern Sie die Anmeldeinformationen als JSON im folgenden Format:

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

Beispiel einer Verbindungszeichenfolge mit einem geheimen Namen

Die folgende Zeichenfolge hat den geheimen Namen `${Test/impala1host}`.

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

Der Konnektor verwendet den geheimen Namen, um Secrets abzurufen und den Benutzernamen und das Kennwort bereitzustellen, wie im folgenden Beispiel gezeigt.

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

Derzeit erkennt Cloudera Impala die `UID`- und `PWD`-JDBC-Eigenschaften.

Verwenden eines einzelnen Verbindungs-Handlers

Sie können die folgenden Einzelverbindungsmetadaten und Record Handler verwenden, um eine Verbindung zu einer einzelnen Cloudera Impala-Instance herzustellen.

Handler-Typ	Klasse
Composite Handler	`ImpalaCompositeHandler`
Metadaten-Handler	`ImpalaMetadataHandler`
Record Handler	`ImpalaRecordHandler`

Parameter für Einzelverbindungs-Handler

Parameter	Beschreibung
`default`	Erforderlich Die standardmäßige Verbindungszeichenfolge.

Die Einzelverbindungs-Handler unterstützen eine Datenbank-Instance und müssen einen `default`-Verbindungszeichenfolgenparameter bereitstellen. Alle anderen Verbindungszeichenfolgen werden ignoriert.

Die folgende Beispieleigenschaft gilt für eine einzelne Cloudera Impala-Instance, die von einer Lambda-Funktion unterstützt wird.

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

Überlauf-Parameter

Das Lambda-SDK kann Daten an Amazon S3 übertragen. Alle Datenbank-Instances, auf die mit derselben Lambda-Funktion zugegriffen wird, werden an denselben Speicherort verschoben.

Parameter	Beschreibung
`spill_bucket`	Erforderlich Überlauf-Bucket-Name.
`spill_prefix`	Erforderlich Schlüssel-Prefix für den Überlauf-Bucket.
`spill_put_request_headers`	(Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Weitere mögliche Header finden Sie unter PutObject in der API-Referenz zu Amazon Simple Storage Service.

Datentypunterstützung

Die folgende Tabelle zeigt die entsprechenden Datentypen für JDBC, Cloudera Impala und Arrow.

JDBC	Cloudera Impala	Arrow
Boolesch	Boolesch	Bit
Ganzzahl	TINYINT	Tiny
Short	SMALLINT	Smallint
Ganzzahl	INT	Int
Long	BIGINT	Bigint
float	float4	Float4
Double	float8	Float8
Datum	date	DateDay
Zeitstempel	Zeitstempel	DateMilli
String	VARCHAR	Varchar
Bytes	bytes	Varbinary
BigDecimal	Dezimal	Dezimal
ARRAY	N.z. (siehe Hinweis)	Auflisten

> **Note**
>
> Derzeit unterstützt Cloudera Impala nicht die Aggregattypen `ARRAY`, `MAP`, `STRUCT` oder`UNIONTYPE`. Spalten mit Aggregattypen werden als `VARCHAR`-Spalten in SQL behandelt.

Partitionen und Splits

Partitionen werden verwendet, um zu bestimmen, wie Splits für den Konnektor generiert werden. Athena konstruiert eine synthetische Säule vom Typ `varchar`, die das Partitionierungsschema für die Tabelle darstellt, das dem Konnektor beim Generieren von Splits hilft. Der Konnektor ändert nicht die eigentliche Tabellendefinition.

Leistung

Cloudera Impala unterstützt statische Partitionen. Der Athena-Cloudera-Impala-Konnektor kann Daten von diesen Partitionen parallel abrufen. Wenn Sie sehr große Datenmengen mit einheitlicher Partitionsverteilung abfragen möchten, wird eine statische Partitionierung dringend empfohlen. Der Cloudera-Impala-Konnektor ist aufgrund der Gleichzeitigkeit widerstandsfähig gegenüber Drosselung.

Der Athena-Cloudera-Impala-Konnektor führt einen Prädikat-Pushdown durch, um die von der Abfrage gescannten Daten zu verringern. `LIMIT`-Klauseln, einfache Prädikate und komplexe Ausdrücke werden an den Konnektor weitergegeben, um die Menge der gescannten Daten und die Laufzeit der Abfrage zu verringern.

LIMIT-Klauseln

Eine `LIMIT N`-Anweisung reduziert die von der Abfrage durchsuchten Daten. Mit `LIMIT N`-Pushdown gibt der Konnektor nur `N` Zeilen an Athena zurück.

Prädikate

Ein Prädikat ist ein Ausdruck in der `WHERE`-Klausel einer SQL-Abfrage, der einen booleschen Wert ergibt und Zeilen auf der Grundlage mehrerer Bedingungen filtert. Der Athena-Cloudera-Impala-Konnektor kann diese Ausdrücke kombinieren und sie direkt an Cloudera Impala weiterleiten, um die Funktionalität zu verbessern und die Menge der gescannten Daten zu reduzieren.

Die folgenden Operatoren des Athena-Cloudera-Impala-Konnektors unterstützen Prädikat-Pushdown:

- Boolean: UND, ODER, NICHT
- Gleichheit: GLEICH, NICHT-GLEICH, WENIGER_ALS, WENIGER_ODER-GLEICH, GRÖSSER_ALS, GRÖSSER_ODER-GLEICH, IST_UNTERSCHIEDEN VON, NULL_WENN, IST_NULL
- Arithmetik: ADDIEREN, SUBTRAHIEREN, MULTIPLIZIEREN, DIVIDIEREN, MODULIEREN, NEGIEREN
- Andere: WIE_MUSTER, IN

Beispiel für einen kombinierten Pushdown

Kombinieren Sie für erweiterte Abfragefunktionen die Pushdown-Typen wie im folgenden Beispiel:

```
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;
```

Lizenzinformationen

Durch die Verwendung dieses Konnektors bestätigen Sie die Aufnahme von Komponenten von Drittanbietern, von denen eine Liste in der pom.xml-Datei für diesen Konnektor zu finden ist, und stimmen den Bedingungen in den jeweiligen Drittanbieterlizenzen zu, die in der LICENSE.txt-Datei auf GitHub.com bereitgestellt werden.

Weitere Informationen finden Sie auch unter

Die neuesten Informationen zur JDBC-Treiberversion finden Sie in der pom.xml-Datei für den Cloudera-Impala-Konnektor auf GitHub.com.

Weitere Informationen zu diesem Konnektor finden Sie auf der entsprechenden Website unter GitHub.com.

Amazon Athena CloudWatch Konnektor

Der Amazon Athena CloudWatch-Konnektor ermöglicht Amazon Athena, mit CloudWatch zu kommunizieren, sodass Sie Ihre Protokolldaten mit SQL abfragen können.

Der Konnektor ordnet Ihre LogGroups als Schemas und jeden LogStream als Tabelle zu. Der Konnektor ordnet auch eine spezielle `all_log_streams`-Ansicht zu, die alle LogStreams in der LogGroup enthält. Diese Ansicht ermöglicht es Ihnen, alle Protokolle in einer LogGroup gleichzeitig abzufragen, anstatt jeden LogStream einzeln zu durchsuchen.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den CloudWatch-Konnektor zu konfigurieren.

- spill_bucket – Gibt den Amazon S3-Bucket für Daten an, die die Lambda-Funktionsgrenzen überschreiten.
- spill_prefix – (Optional) Ist standardmäßig ein Unterordner im angegebenen `spill_bucket` genannt `athena-federation-spill`. Wir empfehlen Ihnen, einen Amazon-S3-Speicher-Lebenszyklus an dieser Stelle zu konfigurieren, um die Überlaufe zu löschen, die älter als eine festgelegte Anzahl von Tagen oder Stunden sind.
- spill_put_request_headers – (Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Andere mögliche Header finden Sie unter PutObject in der API-Referenz zu Amazon Simple Storage Service.
- kms_key_id – (Optional) Standardmäßig werden alle Daten, die an Amazon S3 gesendet werden, mit dem AES-GCM-authentifizierten Verschlüsselungsmodus und einem zufällig generierten Schlüssel verschlüsselt. Damit Ihre Lambda-Funktion stärkere Verschlüsselungsschlüssel verwendet, die von KMS generiert werden, wie`a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, können Sie eine ID einer Verschlüsselung angeben.
- disable_spill_encryption – (Optional) Bei Einstellung auf `True`, wird die Spill-Verschlüsselung deaktiviert. Die Standardeinstellung ist `False`, sodass Daten, die an S3 übertrragen werden, mit AES-GCM verschlüsselt werden - entweder mit einem zufällig generierten Schlüssel oder mit KMS zum Generieren von Schlüsseln. Das Deaktivieren der Überlauf-Verschlüsselung kann die Leistung verbessern, insbesondere wenn Ihr Überlauf-Standort eine serverseitige Verschlüsselung verwendet.

Der Konnektor unterstützt auch AIMD-Staukontrolle für den Umgang mit Drosselungsereignissen von CloudWatch über das Konstrukt Amazon Athena Query Federation SDK `ThrottlingInvoker`. Sie können das Standarddrosselungsverhalten optimieren, indem Sie eine der folgenden optionalen Umgebungsvariablen festlegen:

- throttle_initial_delay_ms – Die erste Aufrufverzögerung, die nach dem ersten Stauereignis angewendet wurde. Der Standardwert beträgt 10 Millisekunden.
- throttle_max_delay_ms – Die maximale Verzögerung zwischen Aufrufen. Sie können TPS ableiten, indem Sie es in 1000 ms teilen. Der Standardwert beträgt 1 000 Millisekunden.
- throttle_decrease_factor – Der Faktor, um den Athena die Aufrufrate reduziert. Der Standardwert ist 0,5.
- throttle_increase_ms – Die Geschwindigkeit, mit der Athena die Aufrufverzögerung verringert. Der Standardwert beträgt 10 Millisekunden.

Datenbanken und Tabellen

Der Athena-CloudWatch-Konnektor ordnet Ihre LogGroups als Schemas (d.h. Datenbanken) und jeden LogStream als Tabelle zu. Der Konnektor ordnet auch eine spezielle `all_log_streams`-Ansicht zu, die alle LogStreams in der LogGroup enthält. Diese Ansicht ermöglicht es Ihnen, alle Protokolle in einer LogGroup gleichzeitig abzufragen, anstatt jeden LogStream einzeln zu durchsuchen.

Jede vom Athena-CloudWatch-Konnektor zugeordnete Tabelle hat das folgende Schema. Dieses Schema stimmt mit den von CloudWatch Logs bereitgestellten Feldern überein.

- log_stream – Ein `VARCHAR`, das den Namen des LogStream enthält, aus dem die Zeile stammt.
- Zeit – Ein `INT64`, das die Epochenzeit enthält, zu der die Protokollzeile generiert wurde.
- Botschaft – Ein `VARCHAR`, das die Protokollnachricht enthält.

Beispiele

Das folgende Beispiel zeigt, wie Sie eine `SELECT`-Abfrage in einem angegebenen LogStream durchführen.

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

Das folgende Beispiel zeigt, wie Sie eine `all_log_streams`-Ansicht, um eine Abfrage für alle LogStreams in einer angegebenen LogGroup durchzuführen.

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

Erforderliche Berechtigungen

Ausführliche Informationen zu den für diesen Konnektor erforderlichen IAM-Richtlinien finden Sie im `Policies`-Abschnitt der athena-cloudwatch.yaml-Datei. In der folgenden Liste sind die erforderlichen Berechtigungen zusammengefasst.

- Amazon-S3-Schreibzugriff – Der Konnektor benötigt Schreibzugriff auf einen Speicherort in Amazon S3, um Ergebnisse aus großen Abfragen zu übertragen.

- Athena GetQueryExecution – Der Konnektor verwendet diese Berechtigung, um ein Fast-Fail durchzuführen, wenn die vorgeschaltete Athena-Abfrage beendet wurde.
- CloudWatch Logs Lese-/Schreibrechte – Der Konnektor verwendet diese Berechtigung, um Ihre Protokolldaten zu lesen und seine Diagnoseprotokolle zu schreiben.

Leistung

Der Athena-CloudWatch-Konnektor versucht, Abfragen für CloudWatch zu optimieren, indem er Scans der für Ihre Abfrage erforderlichen Protokoll-Streams parallelisiert. Für bestimmte Zeitraumfilter wird das Prädikat-Pushdown sowohl innerhalb der Lambda-Funktion als auch in CloudWatch Logs ausgeführt.

Verwenden Sie für Ihre Protokoll-Gruppennamen und Protokoll-Stream-Namen nur Kleinbuchstaben, um eine optimale Leistung zu erzielen. Bei der Verwendung gemischter Groß- und Kleinschreibung führt der Konnektor eine Suche ohne Berücksichtigung der Groß- und Kleinschreibung durch, die rechenintensiver ist.

Lizenzinformationen

Das Amazon Athena CloudWatch Konnektor-Projekt ist lizenziert unter der Apache-2.0-Lizenz.

Weitere Informationen finden Sie auch unter

Weitere Informationen zu diesem Konnektor finden Sie unter der entsprechenden Seite auf GitHub.com.

Amazon Athena CloudWatch Metrics Konnektor

Der CloudWatch-Metrik-Konnektor von Amazon Athena ermöglicht Amazon Athena, CloudWatch-Metrik-Daten mit SQL abzufragen.

Informationen zum Veröffentlichen von Abfragemetriken in CloudWatch von Athena selbst finden Sie unter Steuern von Kosten und Überwachen von Abfragen mit CloudWatch Metriken und Ereignissen.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den CloudWatch-Metrics-Konnektor zu konfigurieren.

- spill_bucket – Gibt den Amazon S3-Bucket für Daten an, die die Lambda-Funktionsgrenzen überschreiten.
- spill_prefix – (Optional) Ist standardmäßig ein Unterordner im angegebenen `spill_bucket` genannt `athena-federation-spill`. Wir empfehlen Ihnen, einen Amazon-S3-Speicher-Lebenszyklus an dieser Stelle zu konfigurieren, um die Überlaufe zu löschen, die älter als eine festgelegte Anzahl von Tagen oder Stunden sind.
- spill_put_request_headers – (Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Andere mögliche Header finden Sie unter PutObject in der API-Referenz zu Amazon Simple Storage Service.
- kms_key_id – (Optional) Standardmäßig werden alle Daten, die an Amazon S3 gesendet werden, mit dem AES-GCM-authentifizierten Verschlüsselungsmodus und einem zufällig generierten Schlüssel verschlüsselt. Damit Ihre Lambda-Funktion stärkere Verschlüsselungsschlüssel verwendet, die von KMS generiert werden, wie`a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, können Sie eine ID einer Verschlüsselung angeben.
- disable_spill_encryption – (Optional) Bei Einstellung auf `True`, wird die Spill-Verschlüsselung deaktiviert. Die Standardeinstellung ist `False`, sodass Daten, die an S3 übertrragen werden, mit AES-GCM verschlüsselt werden - entweder mit einem zufällig generierten Schlüssel oder mit KMS zum Generieren von Schlüsseln. Das Deaktivieren der Überlauf-Verschlüsselung kann die Leistung verbessern, insbesondere wenn Ihr Überlauf-Standort eine serverseitige Verschlüsselung verwendet.

Der Konnektor unterstützt auch AIMD-Staukontrolle für den Umgang mit Drosselungsereignissen von CloudWatch über das Konstrukt Amazon Athena Query Federation SDK `ThrottlingInvoker`. Sie können das Standarddrosselungsverhalten optimieren, indem Sie eine der folgenden optionalen Umgebungsvariablen festlegen:

- throttle_initial_delay_ms – Die erste Aufrufverzögerung, die nach dem ersten Stauereignis angewendet wurde. Der Standardwert beträgt 10 Millisekunden.
- throttle_max_delay_ms – Die maximale Verzögerung zwischen Aufrufen. Sie können TPS ableiten, indem Sie es in 1000 ms teilen. Der Standardwert beträgt 1 000 Millisekunden.

- throttle_decrease_factor – Der Faktor, um den Athena die Aufrufrate reduziert. Der Standardwert ist 0,5.
- throttle_increase_ms – Die Geschwindigkeit, mit der Athena die Aufrufverzögerung verringert. Der Standardwert beträgt 10 Millisekunden.

Datenbanken und Tabellen

Der Athena CloudWatch-Metrics-Konnektor ordnet Ihre Namesbereiche, Dimensionen, Metriken und Metrikwerte in zwei Tabellen in einem einzigen Schema zu, das `default` genannt wird.

Die Metriktabelle

Die`metrics`-Tabelle enthält die verfügbaren Metriken, die durch eine Kombination aus Namensbereichen, Set und Name eindeutig definiert sind. Die `metrics`-Tabelle enthält die folgenden Spalten.

- Namensbereich – Ein `VARCHAR`, der den Namensbereich enthält.
- metric_name – Ein `VARCHAR`, der den Metriknamen enthält.
- dimensions – Ein `LIST` von `STRUCT`-Objekten bestehend aus `dim_name (VARCHAR)` und `dim_value (VARCHAR)`.
- statistic – Ein `LIST` von `VARCH`-Statistiken (zum Beispiel `p90`, `AVERAGE`,...) verfügbar für die Metrik.

Die Tabelle metric_samples

Die `metric_samples`-Tabelle enthält die verfügbaren metrischen Stichproben für jede Metrik in der `metrics`-Tabelle. Die `metric_samples`-Tabelle enthält die folgenden Spalten.

- Namensraum – Ein `VARCHAR`, der den Namensbereich enthält.
- metric_name – Ein `VARCHAR`, der den Metriknamen enthält.
- dimensions – Ein `LIST` von `STRUCT`-Objekten bestehend aus `dim_name (VARCHAR)` und `dim_value (VARCHAR)`.
- dim_name – Ein `VARCHAR`-Komfortfeld, das Sie verwenden können, um einfach nach einem einzelnen Dimensionsnamen zu filtern.
- dim_value – Ein `VARCHAR`-Komfortfeld, das Sie verwenden können, um einfach nach einem einzelnen Dimensionswert zu filtern.

- period – Ein `INT`-Feld, das die „Periode“ der Metrik in Sekunden darstellt (z. B. eine 60-Sekunden-Metrik).
- timestamp – Ein `BIGINT`-Feld, das die Zeit in Sekunden darstellt, für die die metrische Stichprobe gilt.
- value – Ein `FLOAT8`-Feld, das den Wert der Stichprobe enthält.
- Statistik – Ein `VARCHAR`, der den Statistiktyp der Stichprobe enthält (z. B. `AVERAGE` oder `p90`).

Erforderliche Berechtigungen

Ausführliche Informationen zu den für diesen Konnektor erforderlichen IAM-Richtlinien finden Sie im `Policies`-Abschnitt der athena-cloudwatch-metrics.yaml-Datei. In der folgenden Liste sind die erforderlichen Berechtigungen zusammengefasst.

- Amazon-S3-Schreibzugriff – Der Konnektor benötigt Schreibzugriff auf einen Speicherort in Amazon S3, um Ergebnisse aus großen Abfragen zu übertragen.
- Athena GetQueryExecution – Der Konnektor verwendet diese Berechtigung, um ein Fast-Fail durchzuführen, wenn die vorgeschaltete Athena-Abfrage beendet wurde.
- CloudWatch Metrics ReadOnly – Der Konnektor verwendet diese Berechtigung, um Ihre Metrikdaten abzufragen.
- CloudWatch Logs Schreibrechte – Der Konnektor verwendet diesen Zugriff, um seine Diagnoseprotokolle zu schreiben.

Leistung

Der Athena CloudWatch-Metrics-Konnektor versucht, Abfragen anhand von CloudWatch-Metriken zu optimieren, indem Scans der für Ihre Abfrage erforderlichen Protokollstreams parallelisiert werden. Für bestimmte Zeiträume, Metrik-, Namensbereich- und Dimensionsfilter wird der Prädikat-Pushdown sowohl innerhalb der Lambda-Funktion als auch in CloudWatch Logs ausgeführt.

Lizenzinformationen

Das Amazon Athena CloudWatch Metrics Konnektor-Projekt ist lizenziert unter der Apache-2.0-Lizenz.

Weitere Informationen finden Sie auch unter

Weitere Informationen zu diesem Konnektor finden Sie unter der entsprechenden Seite auf GitHub.com.

Amazon Athena AWS CMDB Konnektor

Der AWS-CMDB-Konnektor von Amazon Athena ermöglicht es Amazon Athena, mit verschiedenen AWS-Services zu kommunizieren, sodass Sie diese mit SQL abfragen können.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den AWS-CMDB-Konnektor zu konfigurieren.

- spill_bucket – Gibt den Amazon S3-Bucket für Daten an, die die Lambda-Funktionsgrenzen überschreiten.
- spill_prefix – (Optional) Ist standardmäßig ein Unterordner im angegebenen `spill_bucket` genannt `athena-federation-spill`. Wir empfehlen Ihnen, einen Amazon-S3-Speicher-Lebenszyklus an dieser Stelle zu konfigurieren, um die Überlaufe zu löschen, die älter als eine festgelegte Anzahl von Tagen oder Stunden sind.
- spill_put_request_headers – (Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Andere mögliche Header finden Sie unter PutObject in der API-Referenz zu Amazon Simple Storage Service.
- kms_key_id – (Optional) Standardmäßig werden alle Daten, die an Amazon S3 gesendet werden, mit dem AES-GCM-authentifizierten Verschlüsselungsmodus und einem zufällig generierten Schlüssel verschlüsselt. Damit Ihre Lambda-Funktion stärkere Verschlüsselungsschlüssel verwendet, die von KMS generiert werden, wie`a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, können Sie eine ID einer Verschlüsselung angeben.
- disable_spill_encryption – (Optional) Bei Einstellung auf `True`, wird die Spill-Verschlüsselung deaktiviert. Die Standardeinstellung ist `False`, sodass Daten, die an S3 übertrragen werden, mit AES-GCM verschlüsselt werden - entweder mit einem zufällig generierten Schlüssel oder mit KMS zum Generieren von Schlüsseln. Das Deaktivieren der Überlauf-Verschlüsselung kann die

Leistung verbessern, insbesondere wenn Ihr Überlauf-Standort eine serverseitige Verschlüsselung verwendet.

- default_ec2_image_owner – (Optional) Wenn festgelegt, steuert es den standardmäßigen Amazon EC2-Image-Eigentümer, der Amazon Machine Images (AMI) filtert. Wenn Sie diesen Wert nicht festlegen und Ihre Abfrage für die EC2-Image-Tabelle keinen Filter für Besitzer enthält, enthalten Ihre Ergebnisse alle öffentlichen Images.

Datenbanken und Tabellen

Die Athena-AWS-CMDB-Konnektor stellt die folgenden Datenbanken und Tabellen für die Abfrage Ihres AWS-Bestands der Ressourcen bereit. Um weitere Informationen zu den in den einzelnen Tabellen verfügbaren Spalten zu erhalten, führen Sie eine `DESCRIBE` *`database`*`.`*`table`*-Anweisung mithilfe der Athena-Konsole oder -API aus.

- ec2 – Diese Datenbank enthält Amazon EC2-bezogene Ressourcen, einschließlich der Folgenden.

- ebs_volumes – Enthält Details Ihrer Amazon EBS-Volumes.
- ec2_instances – Enthält Details zu Ihren EC2-Instances.
- ec2_images – Enthält Details zu Ihren EC2-Instance-Images.
- routing_tables – Enthält Details zu Ihren VPC-Routing-Tabellen.
- security_groups – Enthält Details zu Ihren Sicherheitsgruppen.
- subnets – Enthält Details zu Ihren VPC-Subnetzen.
- vpcs – Enthält Details zu Ihren VPCs.

- emr – Diese Datenbank enthält Amazon EMR-bezogene Ressourcen, einschließlich der Folgenden.

- emr_cluster – Enthält Details zu Ihren EMR-Clustern.

- rds – Diese Datenbank enthält Amazon RDS-bezogene Ressourcen, einschließlich der Folgenden.

- rds_instances – Enthält Details zu Ihren RDS-Instances.

- S3 – Diese Datenbank enthält RDS-bezogene Ressourcen, einschließlich der Folgenden.

- buckets – Enthält Details Ihrer Amazon-S3-Buckets.
- objects – Enthält Details zu Ihren Amazon-S3-Objekten, ausgenommen deren Inhalt.

Erforderliche Berechtigungen

Ausführliche Informationen zu den für diesen Konnektor erforderlichen IAM-Richtlinien finden Sie im `Policies`-Abschnitt der athena-aws-cmdb.yaml-Datei. In der folgenden Liste sind die erforderlichen Berechtigungen zusammengefasst.

- Amazon-S3-Schreibzugriff – Der Konnektor benötigt Schreibzugriff auf einen Speicherort in Amazon S3, um Ergebnisse aus großen Abfragen zu übertragen.
- Athena GetQueryExecution – Der Konnektor verwendet diese Berechtigung, um ein Fast-Fail durchzuführen, wenn die vorgeschaltete Athena-Abfrage beendet wurde.
- S3 List – Der Konnektor verwendet diese Berechtigung, um Ihre Amazon-S3-Buckets und -Objekte aufzulisten.
- EC2 Describe – Der Konnektor verwendet diese Berechtigung, um Ressourcen wie Ihre Amazon EC2-Instances, Sicherheitsgruppen, VPCs und Amazon EBS-Volumes zu beschreiben.
- EMR Describe / List – Der Konnektor verwendet diese Berechtigung, um Ihre EMR-Cluster zu beschreiben.
- RDS Describe – Der Konnektor verwendet diese Berechtigung, um Ihre RDS-Instanzen zu beschreiben.

Leistung

Aktuell unterstützt der AWS-CMDB-Konnektor von Athena keine parallel Scans. Der Prädikat-Pushdown wird innerhalb der Lambda-Funktion ausgeführt. Wenn möglich, werden Teilprädikate an die abgefragten Dienste übertragen. Beispielsweise ruft eine Abfrage nach den Details einer bestimmten Amazon EC2-Instance die EC2-API mit der spezifischen Instance-ID auf, um einen gezielten Beschreibungsvorgang auszuführen.

Lizenzinformationen

Die AWS-CMDB-Konnektor-Projekt von Amazon Athena ist lizenziert unter der Apache-2.0-Lizenz.

Weitere Informationen finden Sie auch unter

Weitere Informationen zu diesem Konnektor finden Sie unter der entsprechenden Seite auf GitHub.com.

Amazon-Athena-IBM-Db2-Konnektor

Der Amazon-Athena-Konnektor für Db2 ermöglicht es Amazon Athena, SQL-Abfragen auf Ihren IBM-Db2-Datenbanken mit JDBC auszuführen.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.
- Richten Sie eine VPC und eine Sicherheitsgruppe ein, bevor Sie diesen Konnektor verwenden. Weitere Informationen finden Sie unter Erstellen einer VPC für einen Datenquellen-Connector.

Einschränkungen

- Schreiboperationen wie DDL werden nicht unterstützt.
- In einem Multiplexer-Setup werden der Überlauf-Bucket und das Präfix von allen Datenbank-Instances gemeinsam genutzt.
- Alle relevanten Lambda-Grenzwerte. Weitere Informationen finden Sie unter Lambda quotas (Lambda-Kontingente) im AWS Lambda-Entwicklerhandbuch.
- Datums- und Zeitstempeldatentypen in Filterbedingungen müssen in geeignete Datentypen umgewandelt werden.

Bedingungen

Die folgenden Begriffe beziehen sich auf den Db2-Konnektor.

- Datenbank-Instance – Jede Instance einer Datenbank, die On-Premises, in Amazon EC2 oder auf Amazon RDS bereitgestellt wird.
- Handler – Ein Lambda-Handler, der auf Ihre Datenbank-Instance zugreift. Ein Handler kann für Metadaten oder für Datensätze verwendet werden.
- Metadaten-Handler – Ein Lambda-Handler, der Metadaten von Ihrer Datenbank-Instance abruft.
- Record Handler – Ein Lambda-Handler, der Datensätze aus Ihrer Datenbank-Instance abruft.
- Composite Handler – Ein Lambda-Handler, der sowohl Metadaten als auch Datensätze aus Ihrer Datenbank-Instance abruft.

- Eigenschaft oder Parameter – Eine Datenbankeigenschaft, die von Handlern zum Extrahieren von Datenbankinformationen verwendet wird. Sie konfigurieren diese Eigenschaften als Lambda-Umgebungsvariablen.
- Verbindungszeichenfolge – Eine Textzeichenfolge, die verwendet wird, um eine Verbindung zu einer Datenbank-Instance herzustellen.
- Katalog – Ein Nicht-AWS Glue-Katalog, der bei Athena registriert ist und ein erforderliches Präfix für die `connection_string`-Eigenschaft darstellt.
- Multiplex-Handler – Ein Lambda-Handler, der mehrere Datenbankverbindungen akzeptieren und verwenden kann.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den Db2-Konnektor zu konfigurieren.

Verbindungszeichenfolge

Verwenden Sie eine JDBC-Verbindungszeichenfolge im folgenden Format, um eine Verbindung zu einer Datenbank-Instance herzustellen.

```
dbtwo://${jdbc_connection_string}
```

Verwenden eines Multiplexing-Handlers

Sie können einen Multiplexer verwenden, um mit einer einzigen Lambda-Funktion eine Verbindung zu mehreren Datenbank-Instances herzustellen. Anfragen werden anhand des Katalognamens weitergeleitet. Verwenden Sie die folgenden Klassen in Lambda.

Handler	Klasse
Composite Handler	`Db2MuxCompositeHandler`
Metadaten-Handler	`Db2MuxMetadataHandler`
Record Handler	`Db2MuxRecordHandler`

Multiplex-Handler-Parameter

Parameter	Beschreibung
$*catalog*_connection_string	Erforderlich. Eine Verbindungszeichenfolge einer Datenbank-Instance. Stellen Sie der Umgebungsvariablen den Namen des in Athena verwendeten Katalogs voran. Wenn zum Beispiel der bei Athena registrierte Katalog `mydbtwocatalog` ist, dann lautet der Name der Umgebungsvariablen `mydbtwocatalog_connection_string` .
`default`	Erforderlich. Die standardmäßige Verbindungszeichenfolge. Diese Zeichenfolge wird verwendet, wenn der Katalog `lambda:${` *AWS_LAMBDA_FUNCTION_NAME* `}` ist.

Die folgenden Beispieleigenschaften gelten für eine Db2-MUX-Lambda-Funktion, die zwei Datenbank-Instances unterstützt: `dbtwo1` (die Standardeinstellung) und `dbtwo2`.

Property (Eigenschaft)	Wert
`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*`}`

Bereitstellen von Anmeldeinformationen

Um einen Benutzernamen und ein Kennwort für Ihre Datenbank in Ihrer JDBC-Verbindungszeichenfolge anzugeben, können Sie Eigenschaften von Verbindungszeichenfolgen oder AWS Secrets Manager verwenden.

- Verbindungszeichenfolge – Ein Benutzername und ein Kennwort können als Eigenschaften in der JDBC-Verbindungszeichenfolge angegeben werden.

> **Important**
>
> Als bewährte Sicherheitsmethode sollten Sie keine fest kodierten Anmeldeinformationen in Ihren Umgebungsvariablen oder Verbindungszeichenfolgen verwenden. Informationen zum Verschieben von hartkodierten Geheimnissen nach AWS Secrets Manager finden Sie unter Verschieben von hartkodierten Geheimnissen nach AWS Secrets Manager im AWS Secrets Manager-Benutzerhandbuch.

- AWS Secrets Manager – Um das Athena-Federated-Query-Feature mit AWS Secrets Manager zu verwenden, sollte die mit Ihrer Lambda-Funktion verbundene VPC über einen Internetzugang oder einen VPC-Endpunkt verfügen, um eine Verbindung zu Secrets Manager herzustellen.

  Sie können den Namen eines Secrets in AWS Secrets Manager in Ihrer JDBC-Verbindungszeichenfolge eingeben. Der Konnektor ersetzt den geheimen Namen durch `username`- und `password`-Werte von Secrets Manager.

  Für Amazon RDS-Datenbank-Instances ist diese Unterstützung eng integriert. Wenn Sie Amazon RDS verwenden, empfehlen wir dringend die Verwendung von AWS Secrets Manager und Wechsel der Anmeldeinformationen. Wenn Ihre Datenbank Amazon RDS nicht verwendet, speichern Sie die Anmeldeinformationen als JSON im folgenden Format:

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

Beispiel einer Verbindungszeichenfolge mit einem geheimen Namen

Die folgende Zeichenfolge hat den geheimen Namen `${secret_name}`.

```
dbtwo://jdbc:db2://hostname:port/database_name:${secret_name}
```

Der Konnektor verwendet den geheimen Namen, um Secrets abzurufen und den Benutzernamen und das Kennwort bereitzustellen, wie im folgenden Beispiel gezeigt.

```
dbtwo://jdbc:db2://hostname:port/database_name:user=user_name;password=password;
```

Verwenden eines einzelnen Verbindungs-Handlers

Sie können die folgenden Einzelverbindungs-Metadaten und Datensatz-Handler verwenden, um eine Verbindung zu einer einzelnen Db2-Instance herzustellen.

Handler-Typ	Klasse
Composite Handler	`Db2CompositeHandler`
Metadaten-Handler	`Db2MetadataHandler`
Record Handler	`Db2RecordHandler`

Parameter für Einzelverbindungs-Handler

Parameter	Beschreibung
`default`	Erforderlich. Die standardmäßige Verbindungszeichenfolge.

Die Einzelverbindungs-Handler unterstützen eine Datenbank-Instance und müssen einen `default`-Verbindungszeichenfolgenparameter bereitstellen. Alle anderen Verbindungszeichenfolgen werden ignoriert.

Die folgende Beispieleigenschaft gilt für eine einzelne Db2-Instance, die von einer Lambda-Funktion unterstützt wird.

Property (Eigenschaft)	Wert
`default`	dbtwo://jdbc:db2://hostname:port/*`database_name`* :${secret_name}

Überlauf-Parameter

Das Lambda-SDK kann Daten an Amazon S3 übertragen. Alle Datenbank-Instances, auf die mit derselben Lambda-Funktion zugegriffen wird, werden an denselben Speicherort verschoben.

Parameter	Beschreibung
`spill_bucket`	Erforderlich. Überlauf-Bucket-Name.
`spill_prefix`	Erforderlich. Schlüssel-Prefix für den Überlauf-Bucket.
`spill_put_request_headers`	(Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Andere mögliche Header finden Sie unter PutObject in der API-Referenz zu Amazon Simple Storage Service.

Datentypunterstützung

Die folgende Tabelle zeigt die entsprechenden Datentypen für JDBC und Arrow.

Db2	Arrow
CHAR	VARCHAR
VARCHAR	VARCHAR
DATUM	DATEDAY
TIME	VARCHAR
TIMESTAMP	DATEMILLI
DATETIME	DATEMILLI
BOOLEAN	BOOL
SMALLINT	SMALLINT
INTEGER	INT
BIGINT	BIGINT

Db2	Arrow
DECIMAL	DECIMAL
REAL	FLOAT8
DOUBLE	FLOAT8
DECFLOAT	FLOAT8

Partitionen und Splits

Eine Partition wird durch eine oder mehrere Partitionsspalte(n) vom Typ `varchar` dargestellt. Der Db2-Konnektor erstellt Partitionen mithilfe der folgenden Organisationsschemas.

- Nach Hash verteilen
- Nach Bereich partitionieren
- Nach Dimensionen organisieren

Der Konnektor ruft Partitionsdetails wie die Anzahl der Partitionen und den Spaltennamen aus einer oder mehreren Db2-Metadatentabellen ab. Splits werden basierend auf der Anzahl der identifizierten Partitionen erstellt.

Leistung

Der Athena-Db2-Konnektor führt einen Prädikat-Pushdown durch, um die Anzahl der von der Abfrage gescannten Daten zu reduzieren. `LIMIT`-Klauseln, einfache Prädikate und komplexe Ausdrücke werden an den Konnektor übertragen, um die Menge der gescannten Daten zu reduzieren und die Laufzeit der Abfrage zu verkürzen.

LIMIT-Klauseln

Eine `LIMIT N`-Anweisung reduziert die von der Abfrage durchsuchten Daten. Mit `LIMIT N`-Pushdown gibt der Konnektor nur `N` Zeilen an Athena zurück.

Prädikate

Ein Prädikat ist ein Ausdruck in der `WHERE`-Klausel einer SQL-Abfrage, der einen booleschen Wert ergibt und Zeilen auf der Grundlage mehrerer Bedingungen filtert. Der Athena-Db2-Konnektor kann

diese Ausdrücke kombinieren und sie direkt an Db2 weiterleiten, um die Funktionalität zu erweitern und die Menge der gescannten Daten zu reduzieren.

Die folgenden Athena-Db2-Konnektor-Operatoren unterstützen Prädikat-Pushdown:

- Boolean: UND, ODER, NICHT
- Gleichheit:GLEICH, NICHT_GLEICH, WENIGER_ALS, WENIGER_ODER_GLEICH, GRÖSSER_ALS, GRÖSSER_ODER_GLEICH, IST_UNTERSCHIEDLICH_VON, IST_NULL
- Arithmetik: ADDIEREN, SUBTRAHIEREN, MULTIPLIZIEREN, DIVIDIEREN, MODULIEREN, NEGIEREN
- Andere: WIE_MUSTER, IN

Beispiel für einen kombinierten Pushdown

Kombinieren Sie für erweiterte Abfragefunktionen die Pushdown-Typen wie im folgenden Beispiel:

```
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;
```

Lizenzinformationen

Durch die Verwendung dieses Konnektors erkennen Sie die Aufnahme von Komponenten von Drittanbietern an. Eine Liste dieser Komponenten finden Sie in der pom.xml-Datei für diesen Konnektor. Zudem stimmen Sie den Bedingungen der jeweiligen Drittanbieterlizenzen zu, die in der LICENSE.txt-Datei auf GitHub.com aufgeführt werden.

Weitere Informationen finden Sie auch unter

Die neuesten Informationen zur JDBC-Treiberversion finden Sie in der pom.xml-Datei für den Db2-Konnektor auf GitHub.com.

Weitere Informationen zu diesem Konnektor finden Sie unter der entsprechenden Seite auf GitHub.com.

Amazon Athena IBM Db2 AS/400-Anschluss (Db2 iSeries)

Der Amazon Athena-Connector für Db2 AS/400 ermöglicht es Amazon Athena, mithilfe von JDBC SQL-Abfragen auf Ihren IBM Db2 AS/400-Datenbanken (Db2 iSeries) auszuführen.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.
- Richten Sie eine VPC und eine Sicherheitsgruppe ein, bevor Sie diesen Konnektor verwenden. Weitere Informationen finden Sie unter Erstellen einer VPC für einen Datenquellen-Connector.

Einschränkungen

- Schreiboperationen wie DDL werden nicht unterstützt.
- In einem Multiplexer-Setup werden der Überlauf-Bucket und das Präfix von allen Datenbank-Instances gemeinsam genutzt.
- Alle relevanten Lambda-Grenzwerte. Weitere Informationen finden Sie unter Lambda quotas (Lambda-Kontingente) im AWS Lambda -Entwicklerhandbuch.
- Datums- und Zeitstempeldatentypen in Filterbedingungen müssen in geeignete Datentypen umgewandelt werden.

Bedingungen

Die folgenden Begriffe beziehen sich auf den Db2 AS/400-Konnektor.

- Datenbank-Instance – Jede Instance einer Datenbank, die On-Premises, in Amazon EC2 oder auf Amazon RDS bereitgestellt wird.
- Handler – Ein Lambda-Handler, der auf Ihre Datenbank-Instance zugreift. Ein Handler kann für Metadaten oder für Datensätze verwendet werden.
- Metadaten-Handler – Ein Lambda-Handler, der Metadaten von Ihrer Datenbank-Instance abruft.
- Record Handler – Ein Lambda-Handler, der Datensätze aus Ihrer Datenbank-Instance abruft.
- Composite Handler – Ein Lambda-Handler, der sowohl Metadaten als auch Datensätze aus Ihrer Datenbank-Instance abruft.

- Eigenschaft oder Parameter – Eine Datenbankeigenschaft, die von Handlern zum Extrahieren von Datenbankinformationen verwendet wird. Sie konfigurieren diese Eigenschaften als Lambda-Umgebungsvariablen.
- Verbindungszeichenfolge – Eine Textzeichenfolge, die verwendet wird, um eine Verbindung zu einer Datenbank-Instance herzustellen.
- Katalog — Ein nicht bei Athena registrierter AWS Glue Katalog, der ein erforderliches Präfix für die `connection_string` Immobilie ist.
- Multiplex-Handler – Ein Lambda-Handler, der mehrere Datenbankverbindungen akzeptieren und verwenden kann.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den Db2 AS/400-Connector zu konfigurieren.

Verbindungszeichenfolge

Verwenden Sie eine JDBC-Verbindungszeichenfolge im folgenden Format, um eine Verbindung zu einer Datenbank-Instance herzustellen.

```
db2as400://${jdbc_connection_string}
```

Verwenden eines Multiplexing-Handlers

Sie können einen Multiplexer verwenden, um mit einer einzigen Lambda-Funktion eine Verbindung zu mehreren Datenbank-Instances herzustellen. Anfragen werden anhand des Katalognamens weitergeleitet. Verwenden Sie die folgenden Klassen in Lambda.

Handler	Klasse
Composite Handler	`Db2MuxCompositeHandler`
Metadaten-Handler	`Db2MuxMetadataHandler`
Record Handler	`Db2MuxRecordHandler`

Multiplex-Handler-Parameter

Parameter	Beschreibung
$*catalog*_connection_string	Erforderlich Eine Verbindungszeichenfolge einer Datenbank-Instance. Stellen Sie der Umgebungsvariablen den Namen des in Athena verwendeten Katalogs voran. Wenn zum Beispiel der bei Athena registrierte Katalog `mydb2as400catalog` ist, dann lautet der Name der Umgebungsvariablen `mydb2as400catalog_connection_string` .
`default`	Erforderlich Die standardmäßige Verbindungszeichenfolge. Diese Zeichenfolge wird verwendet, wenn der Katalog `lambda:${` *AWS_LAMBDA_FUNCTION_NAME* } ist.

Die folgenden Beispieleigenschaften gelten für eine Db2-MUX-Lambda-Funktion, die zwei Datenbank-Instances unterstützt: `db2as4001` (die Standardeinstellung) und `db2as4002`.

Eigenschaft	Wert
`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* `}`
`db2as400_catalog3_connection_string`	`db2as400://jdbc:as400://` *<ip_address>* `;user=`*<username>* `;password=` *<password>* `;`*<properties>* `;`

Bereitstellen von Anmeldeinformationen

Um einen Benutzernamen und ein Kennwort für Ihre Datenbank in Ihrer JDBC-Verbindungszeichenfolge anzugeben, können Sie Eigenschaften von Verbindungszeichenfolgen oder AWS Secrets Manager verwenden.

- Verbindungszeichenfolge – Ein Benutzername und ein Kennwort können als Eigenschaften in der JDBC-Verbindungszeichenfolge angegeben werden.

  > **Important**
  >
  > Als bewährte Sicherheitsmethode sollten Sie keine fest kodierten Anmeldeinformationen in Ihren Umgebungsvariablen oder Verbindungszeichenfolgen verwenden. Informationen zum Verschieben Ihrer hartcodierten Geheimnisse nach AWS Secrets Manager finden Sie unter Verschieben von hartcodierten Geheimnissen nach AWS Secrets Manager im Benutzerhandbuch.AWS Secrets Manager

- AWS Secrets Manager— Um die Athena Federated Query-Funktion verwenden zu können AWS Secrets Manager, muss die mit Ihrer Lambda-Funktion verbundene VPC über Internetzugang oder einen VPC-Endpunkt verfügen, um eine Verbindung zu Secrets Manager herzustellen.

  Sie können den Namen eines Geheimnisses in AWS Secrets Manager Ihre JDBC-Verbindungszeichenfolge eingeben. Der Konnektor ersetzt den geheimen Namen durch `username`- und `password`-Werte von Secrets Manager.

  Für Amazon RDS-Datenbank-Instances ist diese Unterstützung eng integriert. Wenn Sie Amazon RDS verwenden, empfehlen wir dringend, eine Rotation der Anmeldeinformationen zu verwenden AWS Secrets Manager . Wenn Ihre Datenbank Amazon RDS nicht verwendet, speichern Sie die Anmeldeinformationen als JSON im folgenden Format:

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

Beispiel einer Verbindungszeichenfolge mit einem geheimen Namen

Die folgende Zeichenfolge hat den geheimen Namen `${secret_name}`.

```
db2as400://jdbc:as400://<ip_address>;<properties>;:${<secret_name>};
```

Der Konnektor verwendet den geheimen Namen, um Secrets abzurufen und den Benutzernamen und das Kennwort bereitzustellen, wie im folgenden Beispiel gezeigt.

```
db2as400://jdbc:as400://<ip_address>;user=<username>;password=<password>;<properties>;
```

Verwenden eines einzelnen Verbindungs-Handlers

Sie können die folgenden Metadaten und Datensatz-Handler für einzelne Verbindungen verwenden, um eine Verbindung zu einer einzelnen DB2-AS/400-Instance herzustellen.

Handler-Typ	Klasse
Composite Handler	`Db2CompositeHandler`
Metadaten-Handler	`Db2MetadataHandler`
Record Handler	`Db2RecordHandler`

Parameter für Einzelverbindungs-Handler

Parameter	Beschreibung
`default`	Erforderlich Die standardmäßige Verbindungszeichenfolge.

Die Einzelverbindungs-Handler unterstützen eine Datenbank-Instance und müssen einen `default`-Verbindungszeichenfolgenparameter bereitstellen. Alle anderen Verbindungszeichenfolgen werden ignoriert.

Die folgende Beispieleigenschaft bezieht sich auf eine einzelne DB2-AS/400-Instanz, die von einer Lambda-Funktion unterstützt wird.

Eigenschaft	Wert
`default`	`db2as400://jdbc:as400://` *`<ip_address>`* `;`*`<properties>`* `::` `${`*`<secret_name>`* `};`

Überlauf-Parameter

Das Lambda-SDK kann Daten an Amazon S3 übertragen. Alle Datenbank-Instances, auf die mit derselben Lambda-Funktion zugegriffen wird, werden an denselben Speicherort verschoben.

Parameter	Beschreibung
`spill_bucket`	Erforderlich Überlauf-Bucket-Name.
`spill_prefix`	Erforderlich Schlüssel-Prefix für den Überlauf-Bucket.
`spill_put_request_headers`	(Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Weitere mögliche Header finden Sie PutObject in der Amazon Simple Storage Service API-Referenz.

Datentypunterstützung

Die folgende Tabelle zeigt die entsprechenden Datentypen für JDBC und Apache Arrow.

Db2 AS/400	Arrow
CHAR	VARCHAR
VARCHAR	VARCHAR
DATUM	DATEDAY
TIME	VARCHAR
TIMESTAMP (ZEITSTEMPEL)	DATEMILLI
DATETIME	DATEMILLI

Db2 AS/400	Arrow
BOOLEAN	BOOL
SMALLINT	SMALLINT
INTEGER	INT
BIGINT	BIGINT
DECIMAL	DECIMAL
REAL	FLOAT8
DOUBLE	FLOAT8
DECFLOAT	FLOAT8

Partitionen und Splits

Eine Partition wird durch eine oder mehrere Partitionsspalte(n) vom Typ `varchar` dargestellt. Der Db2 AS/400-Konnektor erstellt Partitionen mithilfe der folgenden Organisationsschemata.

- Nach Hash verteilen
- Nach Bereich partitionieren
- Nach Dimensionen organisieren

Der Connector ruft Partitionsdetails wie die Anzahl der Partitionen und den Spaltennamen aus einer oder mehreren DB2-AS/400-Metadatentabellen ab. Splits werden basierend auf der Anzahl der identifizierten Partitionen erstellt.

Bewährte Methoden

Verwenden Sie das Prädikat Pushdown, um eine Abfrage von Athena aus durchzuführen, wie in den folgenden Beispielen.

```
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'
```

Lizenzinformationen

Durch die Verwendung dieses Connectors erkennen Sie die Einbindung von Komponenten von Drittanbietern an. Eine Liste dieser Komponenten finden Sie in der Datei pom.xml für diesen Connector und stimmen den Bedingungen der jeweiligen Drittanbieterlizenzen zu, die in der Datei LICENSE.txt auf .com enthalten sind. GitHub

Weitere Informationen finden Sie auch unter

Die neuesten Informationen zur JDBC-Treiberversion finden Sie in der Datei pom.xml für den Db2 AS/400-Connector auf .com. GitHub

Weitere Informationen zu diesem Connector finden Sie auf der entsprechenden Website unter .com. GitHub

Amazon-Athena-DocumentDB-Konnektor

Der DocumentDB-Konnektor von Amazon Athena ermöglicht Amazon Athena die Kommunikation mit Ihren DocumentDB-Instances, sodass Sie Ihre DocumentDB-Daten mit SQL abfragen können. Der Konnektor funktioniert auch mit jedem Endpunkt, der mit MongoDB kompatibel ist.

Im Gegensatz zu herkömmlichen relationalen Datenspeichern haben Amazon-DocumentDB-Sammlungen kein festgelegtes Schema. DocumentDB besitzt keinen Metadatenspeicher. Jeder Eintrag in einer DocumentDB-Sammlung kann unterschiedliche Felder und Datentypen haben.

Der DocumentDB-Konnektor unterstützt zwei Mechanismen zum Generieren von Tabellenschemainformationen: grundlegende Schemainferenz und AWS Glue Data Catalog Metadaten.

Die Schemainferenz ist die Standardeinstellung. Mit dieser Option werden eine kleine Anzahl von Dokumenten in Ihrer Sammlung gescannt, eine Vereinigung aller Felder gebildet und Felder mit nicht überlappenden Datentypen erzwungen. Diese Option eignet sich gut für Sammlungen, die größtenteils einheitliche Einträge haben.

Für Sammlungen mit einer größeren Vielfalt an Datentypen unterstützt der Konnektor das Abrufen von Metadaten aus dem AWS Glue Data Catalog. Wenn der Konnektor eine AWS Glue Datenbank und eine Tabelle erkennt, die Ihren DocumentDB-Datenbank- und Sammlungsnamen entsprechen, bezieht er seine Schemainformationen aus der entsprechenden AWS Glue Tabelle. Wenn Sie Ihre

AWS Glue Tabelle erstellen, empfehlen wir, dass Sie sie zu einer Obermenge aller Felder machen, auf die Sie möglicherweise von Ihrer DocumentDB-Sammlung aus zugreifen möchten.

Wenn Sie Lake Formation in Ihrem Konto aktiviert haben, AWS Serverless Application Repository muss die IAM-Rolle für Ihren Athena Federated Lambda Connector, den Sie in der bereitgestellt haben, Lesezugriff in Lake Formation haben. AWS Glue Data Catalog

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den DocumentDB-Konnektor zu konfigurieren.

- spill_bucket – Gibt den Amazon S3-Bucket für Daten an, die die Lambda-Funktionsgrenzen überschreiten.
- spill_prefix – (Optional) Ist standardmäßig ein Unterordner im angegebenen `spill_bucket` genannt `athena-federation-spill`. Wir empfehlen Ihnen, einen Amazon-S3-Speicher-Lebenszyklus an dieser Stelle zu konfigurieren, um die Überlaufe zu löschen, die älter als eine festgelegte Anzahl von Tagen oder Stunden sind.
- spill_put_request_headers – (Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Weitere mögliche Header finden Sie PutObjectin der Amazon Simple Storage Service API-Referenz.
- kms_key_id – (Optional) Standardmäßig werden alle Daten, die an Amazon S3 gesendet werden, mit dem AES-GCM-authentifizierten Verschlüsselungsmodus und einem zufällig generierten Schlüssel verschlüsselt. Damit Ihre Lambda-Funktion stärkere Verschlüsselungsschlüssel verwendet, die von KMS generiert werden, wie`a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, können Sie eine ID einer Verschlüsselung angeben.
- disable_spill_encryption – (Optional) Bei Einstellung auf `True`, wird die Spill-Verschlüsselung deaktiviert. Die Standardeinstellung ist `False`, sodass Daten, die an S3 übertrragen werden, mit AES-GCM verschlüsselt werden - entweder mit einem zufällig generierten Schlüssel oder mit

KMS zum Generieren von Schlüsseln. Das Deaktivieren der Überlauf-Verschlüsselung kann die Leistung verbessern, insbesondere wenn Ihr Überlauf-Standort eine serverseitige Verschlüsselung verwendet.

- disable_glue — (Optional) Falls vorhanden und auf true gesetzt, versucht der Connector nicht, zusätzliche Metadaten von abzurufen. AWS Glue
- glue_catalog – (Optional) Verwenden Sie diese Option, um einen kontoübergreifenden AWS Glue -Katalog anzugeben. Standardmäßig versucht der Connector, Metadaten von seinem eigenen Konto abzurufen. AWS Glue
- default_docdb – Gibt, falls vorhanden, eine DocumentDB-Verbindungszeichenfolge an, die verwendet wird, wenn keine katalogspezifische Umgebungsvariable vorhanden ist.
- disable_projection_and_casing – (Optional) Deaktiviert Projektion und Groß-/Kleinschreibung. Verwenden Sie diese Option, wenn Sie Amazon-DocumentDB-Tabellen abfragen möchten, die die Groß- und Kleinschreibung von Spaltennamen verwenden. Der `disable_projection_and_casing`-Parameter verwendt die folgenden Werte, um das Verhalten der Groß-/Kleinschreibung und Spaltenzuordnung festzulegen:
  - falsch – Dies ist die Standardeinstellung. Die Projektion ist aktiviert, und der Konnektor erwartet, dass alle Spaltennamen in Kleinbuchstaben geschrieben sind.
  - wahr – Deaktiviert Projektion und Groß- und Kleinschreibung. Beachten Sie bei der Verwendung des `disable_projection_and_casing` Parameters die folgenden Punkte:
    - Die Verwendung des Parameters kann zu höherer Bandbreitennutzung führen. Wenn sich Ihre Lambda-Funktion nicht in derselben AWS-Region wie Ihre Datenquelle befindet, entstehen Ihnen darüber hinaus aufgrund der höheren Bandbreitennutzung höhere standardmäßige AWS -Übertragungskosten zwischen den Regionen. Weitere Informationen zu den Kosten für die Übertragung zwischen Regionen finden Sie im AWS Partnernetzwerk-Blog unter Gebühren für AWS Datenübertragung für Server- und serverlose Architekturen.
    - Da eine größere Anzahl von Bytes übertragen wird und die größere Anzahl von Bytes eine höhere Deserialisierungszeit erfordert, kann sich die Gesamtlatenz erhöhen.
- enable_case_insensitive_match — (Optional) Wenn, führt Suchen ohne Berücksichtigung der Groß- und Kleinschreibung in Schema- und `true` Tabellennamen in Amazon DocumentDB durch. Der Standardwert ist `false`. Verwenden Sie diese Option, wenn Ihre Abfrage Schema- oder Tabellennamen in Großbuchstaben enthält.

Angeben von Verbindungszeichenfolgen

Sie können eine oder mehrere Eigenschaften angeben, die die DocumentDB-Verbindungsdetails für die DocumentDB-Instances definieren, die Sie mit dem Konnektor verwenden. Legen Sie dazu eine Lambda-Umgebungsvariable fest, die dem Katalognamen entspricht, den Sie in Athena verwenden möchten. Angenommen, Sie möchten die folgenden Abfragen verwenden, um zwei verschiedene DocumentDB-Instances von Athena abzufragen:

```
SELECT * FROM "docdb_instance_1".database.table
```

```
SELECT * FROM "docdb_instance_2".database.table
```

Bevor Sie diese beiden SQL-Anweisungen verwenden können, müssen Sie Ihrer Lambda-Funktion zwei Umgebungsvariablen hinzufügen: `docdb_instance_1` und `docdb_instance_2`. Der Wert für jede sollte eine Document-D-Verbindungszeichenfolge mit folgendem Format sein:

```
mongodb://:@:/?ssl=true&ssl_ca_certs=rds-combined-ca-bundle.pem&replicaSet=rs0
```

Verwendung von Secrets

Sie können den Wert optional AWS Secrets Manager für einen Teil oder den gesamten Wert für Ihre Verbindungszeichenfolgendetails verwenden. Um das Athena-Federated-Query-Feature mit Secrets Manager zu verwenden, sollte die mit Ihrer Lamba-Funktion verbundene VPC über einen Internetzugang oder einen VPC-Endpunkt verfügen, um eine Verbindung zu Secrets Manager herzustellen.

Wenn Sie die Syntax `${my_secret}` verwenden, um den Namen eines Secrets aus dem Secrets Manager in Ihre Verbindungszeichenfolge einzufügen, ersetzt der Konnektor `${my_secret}` genau durch den Klartextwert aus Secrets Manager. Secrets sollten als Klartext-Secrets mit dem Wert *`<username>`*`:`*`<password>`* gespeichert werden. Secrets, die als `{username:`*`<username>`*`,password:`*`<password>`*`}` gespeichert wurden, werden nicht ordnungsgemäß an die Verbindungszeichenfolge übergeben.

Secrets können auch vollständig für die gesamte Verbindungszeichenfolge verwendet werden, und der Benutzername und das Passwort können innerhalb des Secrets definiert werden.

Angenommen, Sie setzen die Lambda-Umgebungsvariable für `docdb_instance_1` auf den folgenden Wert:

```
mongodb://${docdb_instance_1_creds}@myhostname.com:123/?ssl=true&ssl_ca_certs=rds-
combined-ca-bundle.pem&replicaSet=rs0
```

Das Athena Query Federation SDK versucht automatisch, ein Secret mit dem Namen `docdb_instance_1_creds` vom Secrets Manager abzurufen und setzt diesen Wert anstelle von `${docdb_instance_1_creds}`. Ein beliebiger Teil der Verbindungszeichenfolge, der in der `${ }`-Zeichenkombination enthalten ist, wird als Secret aus Secrets Manager interpretiert. Wenn Sie einen geheimen Namen angeben, den der Konnektor in Secrets Manager nicht finden kann, ersetzt der Konnektor den Text nicht.

Einrichten von Datenbanken und Tabellen in AWS Glue

Da die integrierte Schemainferenzfunktion des Connectors nur eine begrenzte Anzahl von Dokumenten scannt und nur eine Teilmenge von Datentypen unterstützt, sollten Sie ihn stattdessen AWS Glue für Metadaten verwenden.

Um eine AWS Glue Tabelle für die Verwendung mit Amazon DocumentDB zu aktivieren, benötigen Sie eine AWS Glue Datenbank und eine Tabelle für die DocumentDB-Datenbank und -Sammlung, für die Sie zusätzliche Metadaten bereitstellen möchten.

Um eine AWS Glue Tabelle für zusätzliche Metadaten zu verwenden

1. Wenn Sie die Tabelle und die Datenbank in der AWS Glue Konsole bearbeiten, fügen Sie die folgende Tabelleneigenschaft hinzu.

   - docdb-metadata-flag— Diese Eigenschaft zeigt dem DocumentDB-Konnektor an, dass der Konnektor die Tabelle für zusätzliche Metadaten verwenden kann. Sie können einen beliebigen Wert für `docdb-metadata-flag` angeben, solange die `docdb-metadata-flag`-Eigenschaft in der Liste der Tabelleneigenschaften vorhanden ist.
2. (Optional) Fügen Sie die Tabelleneigenschaft sourceTable hinzu. Diese Eigenschaft definiert den Namen der Quelltabelle in Amazon DocumentDB. Verwenden Sie diese Eigenschaft, wenn Regeln zur AWS Glue Tabellennennung Sie daran hindern, eine AWS Glue Tabelle mit demselben Namen wie Ihre Amazon DocumentDB-Tabelle zu erstellen. Beispielsweise sind Großbuchstaben in AWS Glue -Tabellennamen nicht zulässig, aber sie sind in Amazon-DocumentDB-Tabellennamen zulässig.
3. (Optional) Fügen Sie die Tabelleneigenschaft columnMapping hinzu. Diese Eigenschaft definiert die Zuordnungen von Spaltennamen. Verwenden Sie diese Eigenschaft, wenn die Regeln zur Benennung von AWS Glue Spalten Sie daran hindern, eine AWS Glue Tabelle zu erstellen, die

dieselben Spaltennamen wie die in Ihrer Amazon DocumentDB-Tabelle hat. Dies kann nützlich sein, da Großbuchstaben in Amazon-DocumentDB-Spaltennamen zulässig sind, aber nicht in AWS Glue -Spaltennamen.

Es wird erwartet, dass es sich bei dem `columnMapping`-Eigenschaftswert um eine Reihe von Zuordnungen im Format `col1=Col1,col2=Col2` handelt.

> **Note**
>
> Die gilt Spaltenzuordnung nur für Spaltennamen der obersten Ebene und nicht für verschachtelte Felder.

Nachdem Sie die AWS Glue `columnMapping` Tabelleneigenschaft hinzugefügt haben, können Sie die `disable_projection_and_casing` Lambda-Umgebungsvariable entfernen.

4. Stellen Sie sicher, dass Sie die in diesem Dokument aufgeführten Datentypen verwenden, die für AWS Glue geeignet sind.

Datentypunterstützung

In diesem Abschnitt werden die Datentypen aufgeführt, die der DocumentDB-Konnektor für Schemainferenz verwendet, sowie die Datentypen, wenn AWS Glue Metadaten verwendet werden.

Datentypen für Schemainferenz

Das Schemainferenzfeature des DocumentDB-Konnektors versucht, Werte als zu einem der folgenden Datentypen gehörend abzuleiten. Die Tabelle zeigt die entsprechenden Datentypen für Amazon DocumentDB, Java und Apache Arrow.

Apache Arrow	Java oder DocDB
VARCHAR	String
INT	Ganzzahl
BIGINT	Long
BIT	Boolesch

Apache Arrow	Java oder DocDB
FLOAT4	Gleitkommazahl
FLOAT8	Double
ZEITSTEMPEL/SEK	Datum
VARCHAR	ObjectId
LIST	Auflisten
STRUCT	Dokument

AWS Glue Datentypen

Wenn Sie zusätzliche Metadaten verwenden AWS Glue , können Sie die folgenden Datentypen konfigurieren. Die Tabelle zeigt die entsprechenden Datentypen für AWS Glue und Apache Arrow.

AWS Glue	Apache Arrow
int	INT
bigint	BIGINT
double	FLOAT8
float	FLOAT4
boolesch	BIT
Binary	VARBINARY
Zeichenfolge	VARCHAR
Liste	LIST
Struct	STRUCT

Erforderliche Berechtigungen

Ausführliche Informationen zu den für diesen Konnektor erforderlichen IAM-Richtlinien finden Sie im `Policies`-Abschnitt der athena-docdb.yaml-Datei. In der folgenden Liste sind die erforderlichen Berechtigungen zusammengefasst.

- Amazon-S3-Schreibzugriff – Der Konnektor benötigt Schreibzugriff auf einen Speicherort in Amazon S3, um Ergebnisse aus großen Abfragen zu übertragen.
- Athena GetQueryExecution — Der Konnektor verwendet diese Berechtigung, um einen Fast-Fail auszuführen, wenn die Upstream-Athena-Abfrage beendet wurde.
- AWS Glue Data Catalog— Der DocumentDB-Konnektor benötigt nur Lesezugriff auf die AWS Glue Data Catalog , um Schemainformationen abzurufen.
- CloudWatch Logs — Der Connector benötigt Zugriff auf CloudWatch Logs, um Logs zu speichern.
- AWS Secrets Manager Lesezugriff — Wenn Sie DocumentDB-Endpunktdetails in Secrets Manager speichern möchten, müssen Sie dem Connector Zugriff auf diese Secrets gewähren.
- Zugriff auf die VPC – Der Konnektor erfordert die Fähigkeit, Schnittstellen an Ihre VPC anzuhängen und zu trennen, damit diese eine Verbindung zu dieser herstellen und mit Ihren DocumentDB-Instances kommunizieren kann.

Leistung

Der Amazon-DocumentDB-Konnektor von Athena unterstützt derzeit keine parallelen Scans, sondern versucht, Prädikate als Teil seiner DocumentDB-Abfragen nach unten zu verschieben. Prädikate für Indizes in Ihrer DocumentDB-Sammlung führen zu deutlich weniger gescannten Daten.

Die Lambda-Funktion führt Projektions-Pushdown durch, um die von der Abfrage gescannten Daten zu reduzieren. Die Auswahl einer Teilmenge von Spalten führt jedoch manchmal zu einer längeren Laufzeit der Abfrageausführung. `LIMIT`-Klauseln reduzieren die Menge der gescannten Daten, aber wenn Sie kein Prädikat angeben, sollten Sie davon ausgehen, dass `SELECT`-Abfragen mit einer `LIMIT`-Klausel mindestens 16 MB an Daten scannen.

Weitere Informationen finden Sie auch unter

- Einen Artikel über die Verwendung von Amazon Athena Federated Query, um eine MongoDB-Datenbank mit Amazon zu verbinden, um Dashboards und Visualisierungen QuickSight zu erstellen, finden Sie unter Visualisieren von MongoDB-Daten von Amazon QuickSight mithilfe von Amazon Athena Federated Query im Big Data-Blog.AWS

- Weitere Informationen zu diesem Connector finden Sie auf der entsprechenden Website auf .com. GitHub

Amazon Athena DynamoDB Konnektor

Der Amazon-Athena-DynamoDB-Konnektor ermöglicht Amazon Athena die Kommunikation mit DynamoDB, sodass Sie Ihre Tabellen mit SQL abfragen können. Schreiboperationen wie INSERT INTO werden nicht unterstützt.

Wenn Sie Lake Formation in Ihrem Konto aktiviert haben, muss die IAM-Rolle für Ihren Athena-Verbund-Lambda-Konnektor, den Sie im AWS Serverless Application Repository bereitstellen, in Lake Formation über Lesezugriff auf das AWS Glue Data Catalog verfügen.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den DynamoDB-Konnektor zu konfigurieren.

- spill_bucket – Gibt den Amazon S3-Bucket für Daten an, die die Lambda-Funktionsgrenzen überschreiten.
- spill_prefix – (Optional) Ist standardmäßig ein Unterordner im angegebenen `spill_bucket` genannt `athena-federation-spill`. Wir empfehlen Ihnen, einen Amazon-S3-Speicher-Lebenszyklus an dieser Stelle zu konfigurieren, um die Überlaufe zu löschen, die älter als eine festgelegte Anzahl von Tagen oder Stunden sind.
- spill_put_request_headers – (Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Andere mögliche Header finden Sie unter PutObject in der API-Referenz zu Amazon Simple Storage Service.
- kms_key_id – (Optional) Standardmäßig werden alle Daten, die an Amazon S3 gesendet werden, mit dem AES-GCM-authentifizierten Verschlüsselungsmodus und einem zufällig generierten

Schlüssel verschlüsselt. Damit Ihre Lambda-Funktion stärkere Verschlüsselungsschlüssel verwendet, die von KMS generiert werden, wie`a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, können Sie eine ID einer Verschlüsselung angeben.

- disable_spill_encryption – (Optional) Bei Einstellung auf `True`, wird die Spill-Verschlüsselung deaktiviert. Die Standardeinstellung ist `False`, sodass Daten, die an S3 übertrragen werden, mit AES-GCM verschlüsselt werden - entweder mit einem zufällig generierten Schlüssel oder mit KMS zum Generieren von Schlüsseln. Das Deaktivieren der Überlauf-Verschlüsselung kann die Leistung verbessern, insbesondere wenn Ihr Überlauf-Standort eine serverseitige Verschlüsselung verwendet.
- disable_glue – (Optional) Falls vorhanden und auf „true“ (wahr) gesetzt, versucht der Konnektor nicht, zusätzliche Metadaten aus AWS Glue abzurufen.
- glue_catalog – (Optional) Verwenden Sie diese Option, um einen kontoübergreifenden AWS Glue-Katalog anzugeben. Standardmäßig versucht der Konnektor, Metadaten von seinem eigenen AWS Glue-Konto abzurufen.
- disable_projection_and_casing – (Optional) Deaktiviert Projektion und Groß-/Kleinschreibung. Verwenden Sie diese Option, wenn Sie DynamoDB-Tabellen abfragen möchten, deren Spaltennamen Groß- und Kleinschreibung enthalten, und Sie keine `columnMapping`-Eigenschaft für Ihre AWS Glue-Tabelle festlegen wollen.

  Der `disable_projection_and_casing`-Parameter verwendt die folgenden Werte, um das Verhalten der Groß-/Kleinschreibung und Spaltenzuordnung festzulegen:

  - auto – Deaktiviert Projektion und Groß-/Kleinschreibung, wenn ein zuvor nicht unterstützter Typ erkannt wird und die Spaltennamenzuordnung für die Tabelle nicht festgelegt ist. Dies ist die Standardeinstellung.
  - always – Deaktiviert Projektion und Groß-/Kleinschreibung bedingungslos. Dies ist nützlich, wenn Ihre DynamoDB-Spaltennamen Groß- und Kleinschreibung enthalten, aber keine Spaltennamenzuordnung angeben möchten.

  Beachten Sie bei der Verwendung des `disable_projection_and_casing` Parameters die folgenden Punkte:

  - Die Verwendung des Parameters kann zu höherer Bandbreitennutzung führen. Wenn sich Ihre Lambda-Funktion nicht in derselben AWS-Region wie Ihre Datenquelle befindet, entstehen Ihnen darüber hinaus aufgrund der höheren Bandbreitennutzung höhere standardmäßige AWS-Übertragungskosten zwischen den Regionen. Weitere Informationen zu den regionsübergreifenden Übertragungskosten finden Sie unter AWS-Datenübertragungsgebühren für Server- und Serverless Architekturen im AWS-Partner-Network-Blog.

- Da eine größere Anzahl von Bytes übertragen wird und die größere Anzahl von Bytes eine höhere Deserialisierungszeit erfordert, kann sich die Gesamtlatenz erhöhen.

Einrichten von Datenbanken und Tabellen in AWS Glue

Da die integrierten Schemainferenzfunktionen des Konnektors begrenzt sind, sollten Sie AWS Glue für Metadaten verwenden. Dazu müssen Sie über eine Datenbank und eine Tabelle in AWS Glue verfügen. Um sie für die Verwendung mit DynamoDB zu aktivieren, müssen Sie ihre Eigenschaften bearbeiten.

So bearbeiten Sie Datenbankeigenschaften in der AWS Glue-Konsole

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die AWS Glue-Konsole unter https://console.aws.amazon.com/glue/.
2. Wählen Sie die Registerkarte Databases (Datenbanken) aus.

   Auf der Seite Databases (Datenbanken) können Sie eine vorhandene Datenbank bearbeiten oder Add Databases (Datenbank hinzufügen) auswählen, um eine zu erstellen.
3. Wählen Sie in der Liste der Datenbanken den Link für die Datenbank aus, die Sie bearbeiten möchten.
4. Wählen Sie Edit (Bearbeiten) aus.
5. Fügen Sie auf der Seite Update a database (Eine Datenbank aktualisieren) für Location (Speicherort) die Zeichenfolge **`dynamo-db-flag`** hinzu. Dieses Schlüsselwort gibt an, dass die Datenbank Tabellen enthält, die der Athena-DynamoDB-Konnektor für ergänzende Metadaten verwendet und ist für andere AWS Glue-Datenbanken als `default` erforderlich. Die `dynamo-db-flag`-Eigenschaft ist nützlich, um Datenbanken in Konten mit vielen Datenbanken herauszufiltern.

Tabelleneigenschaften in der AWS Glue-Konsole bearbeiten

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die AWS Glue-Konsole unter https://console.aws.amazon.com/glue/.
2. Wählen Sie die Registerkarte Tables (Tabellen).

   Bearbeiten Sie auf der Registerkarte Tabellen eine vorhandene Tabelle. Informationen zum manuellen Hinzufügen von Tabellen oder zur Verwendung eines Crawlers dafür finden Sie unter Arbeiten mit Tabellen in der AWS Glue-Konsole im AWS Glue-Entwicklerhandbuch.

3. Wählen Sie in der Liste der Tabellen den Link für die Tabelle aus, die Sie bearbeiten möchten.
4. Wählen Sie Actions (Aktionen) und Edit table (Tabelle bearbeiten).
5. Fügen Sie auf der Seite Edit table (Tabelle bearbeiten) im Abschnitt Table properties (Tabelleneigenschaften) nach Bedarf die folgenden Tabelleneigenschaften hinzu. Wenn Sie den AWS Glue DynamoDB-Crawler verwenden, werden diese Eigenschaften automatisch festgelegt.

   - DynamoDB – Zeichenfolge, die dem Athena-DynamoDB-Konnektor anzeigt, dass die Tabelle für ergänzende Metadaten verwendet werden kann. Geben Sie die `dynamodb`-Zeichenfolge in den Tabelleneigenschaften unter einem Feld mit der Bezeichnung classification (Einstufung) (exakte Übereinstimmung) ein.

     > **Note**
     >
     > Die Seite Tabelleneigenschaften festlegen, die Teil der Tabellenerstellung in der AWS Glue-Konsole ist, enthält einen Abschnitt Datenformat mit einem Feld Klassifizierung. Sie können hier nicht `dynamodb` eingeben oder wählen. Nachdem Sie Ihre Tabelle erstellt haben, folgen Sie stattdessen den Schritten zum Bearbeiten der Tabelle und zum Eingeben von `classification` und `dynamodb` als Schlüssel-Wert-Paar im Abschnitt Tabelleneigenschaften.

   - sourceTable – Optionale Tabelleneigenschaft, die den Namen der Quelltabelle in DynamoDB definiert. Verwenden Sie dies, wenn AWS Glue-Regeln für die Tabellenbenennung das Erstellen einer AWS Glue-Tabelle mit demselben Namen wie Ihre DynamoDB-Tabelle verhindern. Zum Beispiel sind Großbuchstaben in AWS Glue-Tabellennamen nicht erlaubt, aber sie sind in DynamoDB-Tabellennamen zulässig.
   - columnMapping – Optionale Tabelleneigenschaft, die Spaltennamenzuordnungen definiert. Verwenden Sie dies, wenn AWS Glue-Regeln für die Spaltenbenennung das Erstellen einer AWS Glue-Tabelle mit denselben Spaltennamen wie Ihre DynamoDB-Tabelle verhindern. Zum Beispiel sind Großbuchstaben in AWS Glue-Spaltennamen nicht erlaubt, sind aber in DynamoDB-Spaltennamen zulässig. Es wird erwartet, dass der Eigenschaftswert im Format col1=COL1, col2=COL2 vorliegt. Beachten Sie, dass die Spaltenzuordnung nur für Spaltennamen der obersten Ebene und nicht für verschachtelte Felder gilt.
   - defaultTimeZone – Optionale Tabelleneigenschaft, die auf `date`- oder `datetime`-Werte angewendet wird, die keine explizite Zeitzone haben. Das Festlegen dieses Werts ist eine bewährte Methode, um Diskrepanzen zwischen der Standardzeitzone der Datenquelle und der Zeitzone der Athena-Sitzung zu vermeiden.

- datetimeFormatMapping – Optionale Tabelleneigenschaft, die das `date`- oder `datetime`-Format festlegt, das beim Analysieren von Daten aus einer Spalte des AWS Glue-, `date`- oder `timestamp`-Datentyps zu verwenden ist. Wenn diese Eigenschaft nicht angegeben ist, versucht der Konnektor ein ISO-8601-Format ableiten. Wenn der Konnektor das `date`- oder `datetime`-Format nicht ermitteln oder die rohe Zeichenkette nicht analysieren kann, wird der Wert im Ergebnis ausgelassen.

  Der Wert `datetimeFormatMapping` muss das Format `col1=someformat1,col2=someformat2` haben. Im Folgenden sind einige Beispielformate aufgeführt:

  ```
  yyyyMMdd'T'HHmmss
  ddMMyyyy'T'HH:mm:ss
  ```

  Wenn Ihre Kolumne `date`- oder `datetime`-Werte ohne Zeitzone hat und Sie die Spalte in der `WHERE`-Klausel verwenden möchten, legen Sie die `datetimeFormatMapping`-Eigenschaft für die Spalte fest.

6. Wenn Sie Ihre Spalten manuell definieren, stellen Sie sicher, dass Sie die entsprechenden Datentypen verwenden. Wenn Sie einen Crawler verwendet haben, überprüfen Sie die Spalten und Typen, die der Crawler erkannt hat.

Erforderliche Berechtigungen

Ausführliche Informationen zu den für diesen Konnektor erforderlichen IAM-Richtlinien finden Sie im `Policies`-Abschnitt der athena-dynamodb.yaml-Datei. In der folgenden Liste sind die erforderlichen Berechtigungen zusammengefasst.

- Amazon-S3-Schreibzugriff – Der Konnektor benötigt Schreibzugriff auf einen Speicherort in Amazon S3, um Ergebnisse aus großen Abfragen zu übertragen.
- Athena GetQueryExecution – Der Konnektor verwendet diese Berechtigung, um ein Fast-Fail durchzuführen, wenn die vorgeschaltete Athena-Abfrage beendet wurde.
- AWS Glue Data Catalog – Der DynamoDB-Konnektor benötigt schreibgeschützten Zugriff auf AWS Glue Data Catalog, um Schemainformationen zu erhalten.
- CloudWatch Logs – Der Konnektor benötigt Zugriff auf CloudWatch Logs zum Speichern von Protokollen.

- DynamoDB-Lesezugriff – Der Konnektor verwendet die `DescribeTable`, `ListSchemas`, `ListTables`, `Query` und `Scan` API-Operationen.

Leistung

Der Athena DynamoDB-Konnektor unterstützt parallel Scans und versucht, Prädikat-Push-Down als Teil seiner DynamoDB-Abfragen durchzuführen. Ein Hash-Schlüssel-Prädikat mit verschiedenen `X`-Werten führt zu `X`-Abfragen von DynamoDB. Alle anderen Prädikatszenarien führen zu einer Anzahl von `Y` Scan-Aufrufen, wobei `Y` heuristisch basierend auf der Größe Ihrer Tabelle und dem bereitgestellten Durchsatz bestimmt wird. Die Auswahl einer Teilmenge von Spalten führt jedoch manchmal zu einer längeren Laufzeit der Abfrageausführung.

`LIMIT`-Klauseln und einfache Prädikate werden nach unten verschoben, wodurch die Menge der gescannten Daten reduziert und die Laufzeit der Abfrageausführung verkürzt wird.

LIMIT-Klauseln

Eine `LIMIT N`-Anweisung reduziert die von der Abfrage durchsuchten Daten. Mit `LIMIT N`-Pushdown gibt der Konnektor nur `N` Zeilen an Athena zurück.

Prädikate

Ein Prädikat ist ein Ausdruck in der `WHERE`-Klausel einer SQL-Abfrage, der einen booleschen Wert ergibt und Zeilen auf der Grundlage mehrerer Bedingungen filtert. Um die Funktionalität zu erweitern und die Menge der gescannten Daten zu reduzieren, kann der Athena-DynamoDB-Konnektor diese Ausdrücke kombinieren und sie direkt an DynamoDB weiterleiten.

Die folgenden Athena-DynamoDB-Konnektor-Operatoren unterstützen Prädikat-Pushdown:

- Boolean: UND
- Gleichheit: GLEICH, NICHT GLEICH, WENIGER_ALS, WENIGER_ODER_GLEICH, GRÖSSER_ALS, GRÖSSER_ODER_GLEICH, IST_NULL

Beispiel für einen kombinierten Pushdown

Kombinieren Sie für erweiterte Abfragefunktionen die Pushdown-Typen wie im folgenden Beispiel:

```
SELECT *
FROM my_table
```

```
WHERE col_a > 10 and col_b < 10
LIMIT 10
```

Einen Artikel über die Verwendung von Prädikat-Pushdown zur Verbesserung der Leistung in Verbundabfragen, einschließlich DynamoDB, finden Sie unter Verbessern von Verbundabfragen mit Prädikat-Pushdown in Amazon Athena im AWS-Big-Data-Blog.

Fehlerbehebung

Mehrere Filter in einer Sortierschlüsselspalte

Fehlermeldung: KeyConditionExpressions darf nur eine Bedingung pro Schlüssel enthalten

Ursache: Dieses Problem kann in Athena-Engine-Version 3 bei Abfragen auftreten, die sowohl einen unteren als auch einen oberen Grenzfilter für eine DynamoDB-Sortierschlüsselspalte haben. Da DynamoDB nicht mehr als eine Filterbedingung für einen Sortierschlüssel unterstützt, wird ein Fehler ausgelöst, wenn der Konnektor versucht, eine Abfrage herunterzufahren, auf die beide Bedingungen angewendet wurden.

Lösung: Aktualisieren Sie die Konnektor-Version auf Version 2023.11.1 oder höher. Anweisungen zum Aktualisieren eines Konnektors finden Sie unter Schreiben eines Datenquellen-Konnektors.

Kosten

Die Kosten für die Nutzung des Steckverbinders hängen von den zugrundeliegenden AWS-Ressourcen ab, die verwendet werden. Weil Abfragen, die Scans verwenden, eine große Anzahl von Lesekapazitätseinheiten (RCUs) verbrauchen, beachten Sie genau die Informationen für Amazon DynamoDB-Prüfung.

Weitere Informationen finden Sie auch unter

- Eine Einführung in die Verwendung des Amazon-Athena-DynamoDB-Konnektors finden Sie unter Zugreifen, Abfragen und Verbinden von Amazon-DynamoDB-Tabellen mit Athena im Leitfaden Muster von AWS Prescriptive Guidance.
- Einen Artikel zur Verwendung des Amazon-Athena-DynamoDB-Konnektors mit Amazon DynamoDB, Athena und Amazon QuickSight zur Erstellung eines einfachen Governance-Dashboards finden Sie unter Kontenübergreifende Amazon-DynamoDB-Tabellen mit Amazon Athena Federated Query abfragen im AWS-Big-Data-Blog.
- Weitere Informationen zu diesem Konnektor finden Sie unter der entsprechenden Seite auf GitHub.com.

Amazon Athena Google-Konnektor BigQuery

Der Amazon Athena-Connector für Google BigQueryermöglicht es Amazon Athena, SQL-Abfragen für Ihre BigQuery Google-Daten auszuführen.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Einschränkungen

- Lambda-Funktionen haben einen maximalen Timeout-Wert von 15 Minuten. Jeder Split führt eine Abfrage aus BigQuery und muss so lange abgeschlossen sein, bis die Ergebnisse gespeichert werden können, damit Athena sie lesen kann. Wenn bei der Lambda-Funktion eine Zeitüberschreitung auftritt, schlägt die Abfrage fehl.
- Google unterscheidet zwischen Groß- und BigQuery Kleinschreibung. Der Konnektor versucht, die Groß- und Kleinschreibung von Datensatznamen und Tabellennamen zu korrigieren, führt jedoch keine Korrektur der Groß- und Kleinschreibung für Projekt-IDs durch. Dies ist notwendig, da Athena alle Metadaten in Kleinbuchstaben schreibt. Diese Korrekturen führen zu vielen zusätzlichen Aufrufen bei Google BigQuery.
- Der Datentyp BINARY wird nicht unterstützt.
- Aufgrund der BigQuery Parallelität von Google und der Kontingentbeschränkungen kann es beim Connector zu Problemen mit der Google-Kontingentbegrenzung kommen. Um diese Probleme zu vermeiden, sollten Sie Google so viele Einschränkungen BigQuery wie möglich übertragen. Informationen zu BigQuery Kontingenten finden Sie in der BigQuery Google-Dokumentation unter Kontingente und Beschränkungen.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den BigQuery Google-Connector zu konfigurieren.

- spill_bucket – Gibt den Amazon S3-Bucket für Daten an, die die Lambda-Funktionsgrenzen überschreiten.

- spill_prefix – (Optional) Ist standardmäßig ein Unterordner im angegebenen `spill_bucket` genannt `athena-federation-spill`. Wir empfehlen Ihnen, einen Amazon-S3-Speicher-Lebenszyklus an dieser Stelle zu konfigurieren, um die Überlaufe zu löschen, die älter als eine festgelegte Anzahl von Tagen oder Stunden sind.
- spill_put_request_headers – (Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Weitere mögliche Header finden Sie PutObjectin der Amazon Simple Storage Service API-Referenz.
- kms_key_id – (Optional) Standardmäßig werden alle Daten, die an Amazon S3 gesendet werden, mit dem AES-GCM-authentifizierten Verschlüsselungsmodus und einem zufällig generierten Schlüssel verschlüsselt. Damit Ihre Lambda-Funktion stärkere Verschlüsselungsschlüssel verwendet, die von KMS generiert werden, wie`a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, können Sie eine ID einer Verschlüsselung angeben.
- disable_spill_encryption – (Optional) Bei Einstellung auf `True`, wird die Spill-Verschlüsselung deaktiviert. Die Standardeinstellung ist `False`, sodass Daten, die an S3 übertrragen werden, mit AES-GCM verschlüsselt werden - entweder mit einem zufällig generierten Schlüssel oder mit KMS zum Generieren von Schlüsseln. Das Deaktivieren der Überlauf-Verschlüsselung kann die Leistung verbessern, insbesondere wenn Ihr Überlauf-Standort eine serverseitige Verschlüsselung verwendet.
- gcp_projekt_id – Die Projekt-ID (nicht der Projektname), die die Datensätze enthält, aus denen der Konnektor lesen soll (z. B. `semiotic-primer-1234567`).
- secret_manager_gcp_creds_name — Der Name des Geheimnisses, das Ihre Anmeldeinformationen im JSON-Format enthält (z. AWS Secrets Manager B.). BigQuery `GoogleCloudPlatformCredentials`
- big_query_endpoint — (Optional) Die URL eines privaten Endpunkts. BigQuery Verwenden Sie diesen Parameter, wenn Sie BigQuery über einen privaten Endpunkt zugreifen möchten.

Splits und Ansichten

Da der BigQuery Connector die BigQuery Storage Read API zum Abfragen von Tabellen verwendet und die BigQuery Speicher-API keine Ansichten unterstützt, verwendet der Connector den BigQuery Client mit einer einzigen Aufteilung für Ansichten.

Leistung

Um Tabellen abzufragen, verwendet der BigQuery Connector die BigQuery Storage Read API, die ein RPC-basiertes Protokoll verwendet, das schnellen Zugriff auf BigQuery verwalteten Speicher ermöglicht. Weitere Informationen zur BigQuery Storage Read API finden Sie in der Google Cloud-Dokumentation unter Verwenden der BigQuery Storage Read API zum Lesen von Tabellendaten.

Die Auswahl einer Teilmenge von Spalten beschleunigt die Abfragelaufzeit erheblich und reduziert die gescannten Daten. Der Konnektor ist mit zunehmender Parallelität Abfragefehlern ausgesetzt und in der Regel ein langsamer Konnektor.

Der Athena BigQuery Google-Connector führt einen Prädikat-Pushdown durch, um die Anzahl der von der Abfrage gescannten Daten zu verringern. `LIMIT`Klauseln, `ORDER BY` Klauseln, einfache Prädikate und komplexe Ausdrücke werden an den Konnektor übertragen, um die Menge der gescannten Daten zu reduzieren und die Laufzeit der Abfrage zu verkürzen.

LIMIT-Klauseln

Eine `LIMIT N`-Anweisung reduziert die von der Abfrage durchsuchten Daten. Mit `LIMIT N`-Pushdown gibt der Konnektor nur `N` Zeilen an Athena zurück.

Top-N-Abfragen

Eine Top-`N`-Abfrage gibt eine Reihenfolge der Ergebnismenge und eine Obergrenze für die Anzahl der zurückgegebenen Zeilen an. Sie können diesen Abfragetyp verwenden, um die höchsten `N`-Höchstwerte oder die höchsten `N`-Minimalwerte für Ihre Datensätze zu ermitteln. Mit `N`-Pushdown gibt der Konnektor nur `N`-geordnete Zeilen an Athena zurück.

Prädikate

Ein Prädikat ist ein Ausdruck in der `WHERE`-Klausel einer SQL-Abfrage, der einen booleschen Wert ergibt und Zeilen auf der Grundlage mehrerer Bedingungen filtert. Der Athena BigQuery Google-Connector kann diese Ausdrücke kombinieren und sie direkt an Google weiterleiten, um BigQuery die Funktionalität zu verbessern und die Menge der gescannten Daten zu reduzieren.

Die folgenden Athena Google BigQuery Connector-Operatoren unterstützen das Prädikat Pushdown:

- Boolean: UND, ODER, NICHT
- Gleichheit: GLEICH, NICHT-GLEICH, WENIGER_ALS, WENIGER_ODER-GLEICH, GRÖSSER_ALS, GRÖSSER_ODER-GLEICH, IST_UNTERSCHIEDEN VON, NULL_WENN, IST_NULL

- Arithmetik: ADDIEREN, SUBTRAHIEREN, MULTIPLIZIEREN, DIVIDIEREN, MODULIEREN, NEGIEREN
- Andere: WIE_MUSTER, IN

Beispiel für einen kombinierten Pushdown

Kombinieren Sie für erweiterte Abfragefunktionen die Pushdown-Typen wie im folgenden Beispiel:

```
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;
```

Lizenzinformationen

Das Amazon Athena Google BigQuery Connector-Projekt ist unter der Apache-2.0-Lizenz lizenziert.

Durch die Verwendung dieses Connectors erkennen Sie die Einbeziehung von Komponenten von Drittanbietern an. Eine Liste dieser Komponenten finden Sie in der Datei pom.xml für diesen Connector, und Sie stimmen den Bedingungen der jeweiligen Drittanbieterlizenzen zu, die in der Datei LICENSE.txt auf .com enthalten sind. GitHub

Weitere Informationen finden Sie auch unter

Weitere Informationen zu diesem Connector finden Sie auf der entsprechenden Website auf GitHub .com.

Google-Cloud-Storage-Konnektor für Amazon Athena

Der Google-Cloud-Storage-Konnektor für Amazon Athena ermöglicht es Amazon Athena, Abfragen für Parquet- und CSV-Dateien auszuführen, die in einem Google Cloud Storage (GCS)-Bucket gespeichert sind. Nachdem Sie eine oder mehrere Parquet- oder CSV-Dateien in einem unpartitionierten oder partitionierten Ordner in einem GCS-Bucket gruppiert haben, können Sie diese in einer AWS Glue-Datenbanktabelle organisieren.

Wenn Sie Lake Formation in Ihrem Konto aktiviert haben, muss die IAM-Rolle für Ihren Athena-Verbund-Lambda-Konnektor, den Sie in AWS Serverless Application Repository bereitgestellt haben, in Lake Formation über Lesezugriff auf das AWS Glue Data Catalog verfügen.

Voraussetzungen

- Richten Sie eine AWS Glue-Datenbank und Tabelle ein, die Ihrem Bucket und Ihren Ordnern in Google Cloud Storage entsprechen. Die Schritte finden Sie unter Einrichten von Datenbanken und Tabellen in AWS Glue weiter unten in diesem Dokument.
- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Einschränkungen

- Schreiboperationen wie DDL werden nicht unterstützt.
- Alle relevanten Lambda-Grenzwerte. Weitere Informationen finden Sie unter Lambda quotas (Lambda-Kontingente) im AWS Lambda-Entwicklerhandbuch.
- Derzeit unterstützt der Konnektor nur den `VARCHAR`-Typ für Partitionsspalten (`string` oder `varchar` in einem AWS Glue-Tabellenschema). Andere Partitionsfeldtypen lösen Fehler aus, wenn Sie sie in Athena abfragen.

Bedingungen

Die folgenden Begriffe beziehen sich auf den GCS-Konnektor.

- Handler – Ein Lambda-Handler, der auf Ihren GCS-Bucket zugreift. Ein Handler kann für Metadaten oder für Datensätze verwendet werden.
- Metadaten-Handler – Ein Lambda-Handler, der Metadaten von Ihrem GCS-Bucket abruft.
- Datensatz-Handler – Ein Lambda-Handler, der Datensätze aus Ihrem GCS-Bucket abruft.
- Zusammengesetzter Handler – Ein Lambda-Handler, der sowohl Metadaten als auch Datensätze aus Ihrem GCS-Bucket abruft.

Unterstützte Dateitypen

Der GCS-Konnektor unterstützt die Parquet- und CSV-Dateitypen.

> **Note**
>
> Stellen Sie sicher, dass Sie CSV- und Parquet-Dateien nicht im selben GCS-Bucket oder -Pfad platzieren. Dies kann zu einem Laufzeitfehler führen, wenn versucht wird, Parquet-Dateien als CSV zu lesen oder umgekehrt.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt zum Konfigurieren des GCS-Konnektors.

- spill_bucket – Gibt den Amazon S3-Bucket für Daten an, die die Lambda-Funktionsgrenzen überschreiten.
- spill_prefix – (Optional) Ist standardmäßig ein Unterordner im angegebenen `spill_bucket` genannt `athena-federation-spill`. Wir empfehlen Ihnen, einen Amazon-S3-Speicher-Lebenszyklus an dieser Stelle zu konfigurieren, um die Überlaufe zu löschen, die älter als eine festgelegte Anzahl von Tagen oder Stunden sind.
- spill_put_request_headers – (Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Andere mögliche Header finden Sie unter PutObject in der API-Referenz zu Amazon Simple Storage Service.
- kms_key_id – (Optional) Standardmäßig werden alle Daten, die an Amazon S3 gesendet werden, mit dem AES-GCM-authentifizierten Verschlüsselungsmodus und einem zufällig generierten Schlüssel verschlüsselt. Damit Ihre Lambda-Funktion stärkere Verschlüsselungsschlüssel verwendet, die von KMS generiert werden, wie`a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, können Sie eine ID einer Verschlüsselung angeben.
- disable_spill_encryption – (Optional) Bei Einstellung auf `True`, wird die Spill-Verschlüsselung deaktiviert. Die Standardeinstellung ist `False`, sodass Daten, die an S3 übertrragen werden, mit AES-GCM verschlüsselt werden - entweder mit einem zufällig generierten Schlüssel oder mit KMS zum Generieren von Schlüsseln. Das Deaktivieren der Überlauf-Verschlüsselung kann die Leistung verbessern, insbesondere wenn Ihr Überlauf-Standort eine serverseitige Verschlüsselung verwendet.
- secret_manager_gcp_creds_name – Der Name des Secrets in AWS Secrets Manager, das Ihre GCS-Anmeldeinformationen im JSON-Format enthält (z. B. `GoogleCloudPlatformCredentials`).

Einrichten von Datenbanken und Tabellen in AWS Glue

Da die integrierte Schema-Inferenzfunktion des GCS-Konnektors begrenzt ist, empfehlen wir Ihnen, AWS Glue für Ihre Metadaten zu verwenden. Die folgenden Verfahren zeigen, wie Sie eine Datenbank und Tabelle in AWS Glue erstellen, auf die Sie von Athena aus zugreifen können.

Erstellen einer Datenbank in AWS Glue

Sie können die AWS Glue-Konsole verwenden, um eine Datenbank für die Verwendung mit dem GCS-Konnektor zu erstellen.

So erstellen Sie eine Datenbank in AWS Glue

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die AWS Glue-Konsole unter https://console.aws.amazon.com/glue/.
2. Wählen Sie im Navigationsbereich Databases (Datenbanken) aus.
3. Wählen Sie Add database (Datenbank hinzufügen).
4. Geben Sie unter Name einen Namen für die Datenbank ein, die Sie mit dem GCS-Konnektor verwenden möchten.
5. Geben Sie für Location (Speicherort) `s3://google-cloud-storage-flag.` an. Dieser Standort teilt dem GCS-Konnektor mit, dass die AWS Glue-Datenbank Tabellen für GCS-Daten enthält, die in Athena abgefragt werden sollen. Der Konnektor erkennt Datenbanken in Athena, die dieses Flag haben und ignoriert Datenbanken, die dies nicht tun.
6. Wählen Sie Datenbank erstellen aus.

Erstellen einer Tabelle in AWS Glue

Jetzt können Sie eine Tabelle für die Datenbank erstellen. Wenn Sie eine AWS Glue-Tabelle zur Verwendung mit dem GCS-Konnektor erstellen, müssen Sie zusätzliche Metadaten angeben.

So erstellen Sie eine Tabelle in der AWS Glue-Konsole

1. Wählen Sie im Navigationsbereich der AWS Glue-Konsole Tables (Tabellen) aus.
2. Wählen Sie auf der Seite Tables (Tabellen) die Option Add table (Tabelle hinzufügen) aus.
3. Geben Sie auf der Seite Set table properties (Tabelleneigenschaften festlegen) die folgenden Informationen ein.

    - Name – Ein eindeutiger Name für die Tabelle.

- Datenbank – Wählen Sie die AWS Glue-Datenbank aus, die Sie für den GCS-Konnektor erstellt haben.
- Pfad einbeziehen – Geben Sie im Abschnitt Data store (Datenspeicher) für Include path (Pfad einbeziehen) den URI-Speicherort für GCS ein, dem das Präfix `gs://` (z. B. `gs://gcs_table/data/`) vorangestellt ist. Wenn Sie über einen oder mehrere Partitionsordner verfügen, nehmen Sie diese nicht in den Pfad auf.

  > **Note**
  >
  > Wenn Sie den Pfad eingeben, der nicht zur `s3://`-Tabelle gehört, zeigt die AWS Glue-Konsole einen Fehler an. Sie können diesen Fehler ignorieren. Die Tabelle wird erfolgreich erstellt.

- Datenformat – Wählen Sie als Classification (Klassifizierung) CSV oder Parquet aus.

4. Wählen Sie Next (Weiter).
5. Auf der Seite Choose or define schema (Schema auswählen oder definieren) wird das Definieren eines Tabellenschemas dringend empfohlen, ist jedoch nicht zwingend erforderlich. Wenn Sie kein Schema definieren, versucht der GCS-Konnektor, ein Schema für Sie abzuleiten.

   Führen Sie eine der folgenden Aktionen aus:

   - Wenn der GCS-Konnektor versuchen soll, ein Schema für Sie abzuleiten, wählen Sie Next (Weiter) und dann Create (Erstellen) aus.
   - Um selbst ein Schema zu definieren, folgen Sie den Schritten im nächsten Abschnitt.

Definition eines Tabellenschemas in AWS Glue

Das Definieren eines Tabellenschemas in AWS Glue erfordert mehr Schritte, gibt Ihnen aber mehr Kontrolle über den Tabellenerstellungsprozess.

So definieren Sie ein Schema für Ihre Tabelle in AWS Glue

1. Wählen Sie auf der Seite Choose or define schema (Schema auswählen oder definieren) die Option Add (Hinzufügen) aus.
2. Verwenden Sie das Dialogfeld Add schema entry (Schemaeintrag hinzufügen), um einen Spaltennamen und einen Datentyp anzugeben.

3. Um die Spalte als Partitionsspalte zu kennzeichnen, wählen Sie die Option Set as partition key (Als Partitionsschlüssel festlegen) aus.
4. Wählen Sie Save (Speichern), um die Spalte zu speichern.
5. Wählen Sie Add (Hinzufügen), um eine weitere Spalte hinzuzufügen.
6. Wenn Sie alle Spalten hinzugefügt haben, wählen Sie Next (Weiter).
7. Überprüfen Sie auf der Seite Review and create (Überprüfen und erstellen) die Tabelle und wählen Sie dann Create (Erstellen) aus.
8. Wenn Ihr Schema Partitionsinformationen enthält, führen Sie die Schritte im nächsten Abschnitt aus, um den Eigenschaften der Tabelle in AWS Glue ein Partitionsmuster hinzuzufügen.

Hinzufügen eines Partitionsmusters zu Tabelleneigenschaften in AWS Glue

Wenn Ihre GCS-Buckets über Partitionen verfügen, müssen Sie das Partitionsmuster zu den Eigenschaften der Tabelle in AWS Glue hinzufügen.

So fügen Sie Partitionsinformationen zu Tabelleneigenschaften AWS Glue hinzu

1. Wählen Sie auf der Detailseite für die Tabelle, die Sie in AWS Glue erstellt haben, Actions (Aktionen), Edit table (Tabelle bearbeiten).
2. Scrollen Sie auf der Seite Edit table (Tabelle bearbeiten) nach unten zum Abschnitt Table properties (Tabelleneigenschaften).
3. Wählen Sie Add (Hinzufügen), um einen Partitionsschlüssel hinzuzufügen.
4. Geben Sie für Key (Schlüssel) **`partition.pattern`** ein. Dieser Schlüssel definiert das Ordnerpfadmuster.
5. Geben Sie für Value (Wert) ein Ordnerpfadmuster wie **`StateName=${statename}/ ZipCode=${zipcode}/`** ein, wobei **`statename`** und **`zipcode`** eingeschlossen von **`${}`** Partitionsspaltennamen sind. Der GCS-Konnektor unterstützt sowohl Hive- als auch Nicht-Hive-Partitionsschemas.
6. Klicken Sie auf Save , sobald Sie fertig sind.
7. Um die soeben erstellten Tabelleneigenschaften anzuzeigen, wählen Sie die Registerkarte Advanced properties (Erweiterte Eigenschaften) aus.

An dieser Stelle können Sie zur Athena-Konsole navigieren. Die Datenbank und Tabelle, die Sie in AWS Glue erstellt haben, stehen für Abfragen in Athena zur Verfügung.

Datentypunterstützung

Die folgenden Tabellen zeigen die unterstützten Datentypen für CSV und Parquet.

CSV

Nature of data (Art der Daten)	Inferred data type (Abgeleiteter Datentyp)
Daten sehen aus wie eine Zahl	BIGINT
Daten sehen aus wie eine Zeichenfolge	VARCHAR
Daten sehen aus wie Fließkommazahlen (Float, Double oder Decimal)	DOUBLE
Daten sehen aus wie ein Datum	Zeitstempel
Daten, die wahre/falsche Werte enthalten	BOOL

Parquet

PARQUET	Athena (Arrow) (Athena (Pfeil))
BINARY	VARCHAR
BOOLEAN	BOOL
DOUBLE	DOUBLE
ENUM	VARCHAR
FIXED_LEN _BYTE_ARRAY	DECIMAL
FLOAT	FLOAT (32-Bit)
INT32	1. INT32 2. DATEDAY (wenn der logische Typ der Parquet-Spalte DATE ist)
INT64	1. INT64

PARQUET	Athena (Arrow) (Athena (Pfeil))
	2. TIMESTAMP (wenn der logische Typ der Parquet-Spalte TIMESTAMP ist)
INT96	Zeitstempel
MAP	MAP
STRUCT	STRUCT
LIST	LIST

Erforderliche Berechtigungen

Ausführliche Informationen zu den für diesen Konnektor erforderlichen IAM-Richtlinien finden Sie im `Policies`-Abschnitt der athena-gcs.yaml-Datei. In der folgenden Liste sind die erforderlichen Berechtigungen zusammengefasst.

- Amazon-S3-Schreibzugriff – Der Konnektor benötigt Schreibzugriff auf einen Speicherort in Amazon S3, um Ergebnisse aus großen Abfragen zu übertragen.
- Athena GetQueryExecution – Der Konnektor verwendet diese Berechtigung, um ein Fast-Fail durchzuführen, wenn die vorgeschaltete Athena-Abfrage beendet wurde.
- AWS Glue Data Catalog – Der GCS-Konnektor benötigt schreibgeschützten Zugriff auf AWS Glue Data Catalog, um Schemainformationen abzurufen.
- CloudWatch Logs – Der Konnektor benötigt Zugriff auf CloudWatch Logs zum Speichern von Protokollen.

Leistung

Wenn das Tabellenschema Partitionsfelder enthält und die `partition.pattern`-Tabelleneigenschaft richtig konfiguriert ist, können Sie das Partitionsfeld in die `WHERE`-Klausel Ihrer Abfragen aufnehmen. Für solche Abfragen verwendet der GCS-Konnektor die Partitionsspalten, um den GCS-Ordnerpfad zu verfeinern und das Scannen nicht benötigter Dateien in GCS-Ordnern zu vermeiden.

Bei Parquet-Datensätzen führt die Auswahl einer Teilmenge von Spalten dazu, dass weniger Daten gescannt werden. Dies führt in der Regel zu einer kürzeren Ausführungszeit der Abfrage, wenn die Spaltenprojektion angewendet wird.

Bei CSV-Datensätzen wird die Spaltenprojektion nicht unterstützt und reduziert nicht die Menge der gescannten Daten.

`LIMIT`-Klauseln reduzieren die Menge der gescannten Daten, aber wenn Sie kein Prädikat angeben, sollten Sie davon ausgehen, dass `SELECT`-Abfragen mit einer `LIMIT`-Klausel mindestens 16 MB Daten scannen. Der GCS-Konnektor scannt mehr Daten bei größeren Datensätzen als bei kleineren Datensätzen, unabhängig von der angewendeten `LIMIT`-Klausel. Zum Beispiel scannt die Abfrage `SELECT * LIMIT 10000` mehr Daten bei einem größeren zugrunde liegenden Datensatz als bei einem kleineren.

Lizenzinformationen

Durch die Verwendung dieses Konnektors erkennen Sie die Aufnahme von Komponenten von Drittanbietern an. Eine Liste dieser Komponenten finden Sie in der pom.xml-Datei für diesen Konnektor. Zudem stimmen Sie den Bedingungen der jeweiligen Drittanbieterlizenzen zu, die in der LICENSE.txt-Datei auf GitHub.com aufgeführt werden.

Weitere Informationen finden Sie auch unter

Weitere Informationen zu diesem Konnektor finden Sie unter der entsprechenden Seite auf GitHub.com.

Amazon Athena HBase Konnektor

Der Amazon-Athena-HBase-Konnektor ermöglicht Amazon Athena die Kommunikation mit Ihren Apache-HBase-Instances, sodass Sie Ihre HBase-Daten mit SQL abfragen können.

Im Gegensatz zu herkömmlichen relationalen Datenspeichern haben HBase-Sammlungen kein festgelegtes Schema. HBase hat keinen Metadatenspeicher. Jeder Eintrag in eine HBase-Sammlung kann unterschiedliche Felder und Datentypen haben.

Der HBase-Konnektor unterstützt zwei Mechanismen zum Generieren von Tabellenschemainformationen: grundlegende Schemainferenz und AWS Glue Data Catalog-Metadaten.

Die Schemainferenz ist die Standardeinstellung. Mit dieser Option werden eine kleine Anzahl von Dokumenten in Ihrer Sammlung gescannt, eine Vereinigung aller Felder gebildet und Felder mit

nicht überlappenden Datentypen erwzungen. Diese Option eignet sich gut für Sammlungen, die größtenteils einheitliche Einträge haben.

Für Sammlungen mit einer größeren Vielfalt an Datentypen unterstützt der Konnektor das Abrufen von Metadaten aus dem AWS Glue Data Catalog. Wenn der Konnektor eine AWS Glue-Datenbank und -Tabelle findet, die Ihrem HBase-Namensbereich und Ihren -Sammlungsnamen entsprechen, erhält er seine Schemainformationen von den entsprechenden AWS Glue-Tabelle. Wenn Sie Ihre AWS Glue-Tabelle erstellen, wird empfohlen, dass Sie sie zu einer Übergruppe aller Felder machen, auf die Sie möglicherweise von Ihrer HBase-Sammlung aus zugreifen möchten.

Wenn Sie Lake Formation in Ihrem Konto aktiviert haben, muss die IAM-Rolle für Ihren Athena-Verbund-Lambda-Konnektor, den Sie im AWS Serverless Application Repository bereitstellen, in Lake Formation über Lesezugriff auf das AWS Glue Data Catalog verfügen.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den HBase-Konnektor zu konfigurieren.

- spill_bucket – Gibt den Amazon S3-Bucket für Daten an, die die Lambda-Funktionsgrenzen überschreiten.
- spill_prefix – (Optional) Ist standardmäßig ein Unterordner im angegebenen `spill_bucket` genannt `athena-federation-spill`. Wir empfehlen Ihnen, einen Amazon-S3-Speicher-Lebenszyklus an dieser Stelle zu konfigurieren, um die Überlaufe zu löschen, die älter als eine festgelegte Anzahl von Tagen oder Stunden sind.
- spill_put_request_headers – (Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Andere mögliche Header finden Sie unter PutObject in der API-Referenz zu Amazon Simple Storage Service.
- kms_key_id – (Optional) Standardmäßig werden alle Daten, die an Amazon S3 gesendet werden, mit dem AES-GCM-authentifizierten Verschlüsselungsmodus und einem zufällig generierten

Schlüssel verschlüsselt. Damit Ihre Lambda-Funktion stärkere Verschlüsselungsschlüssel verwendet, die von KMS generiert werden, wie`a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, können Sie eine ID einer Verschlüsselung angeben.

- disable_spill_encryption – (Optional) Bei Einstellung auf `True`, wird die Spill-Verschlüsselung deaktiviert. Die Standardeinstellung ist `False`, sodass Daten, die an S3 übertrragen werden, mit AES-GCM verschlüsselt werden - entweder mit einem zufällig generierten Schlüssel oder mit KMS zum Generieren von Schlüsseln. Das Deaktivieren der Überlauf-Verschlüsselung kann die Leistung verbessern, insbesondere wenn Ihr Überlauf-Standort eine serverseitige Verschlüsselung verwendet.
- disable_glue – (Optional) Falls vorhanden und auf „true“ (wahr) gesetzt, versucht der Konnektor nicht, zusätzliche Metadaten aus AWS Glue abzurufen.
- glue_catalog – (Optional) Verwenden Sie diese Option, um einen kontoübergreifenden AWS Glue-Katalog anzugeben. Standardmäßig versucht der Konnektor, Metadaten von seinem eigenen AWS Glue-Konto abzurufen.
- default_base – Gibt, falls vorhanden, eine HBase-Verbindungszeichenfolge an, die verwendet werden soll, wenn keine katalogspezifische Umgebungsvariable vorhanden ist.

Angeben von Verbindungszeichenfolgen

Sie können eine oder mehrere Eigenschaften angeben, die die HBase-Verbindungsdetails für die HBase-Instances definieren, die Sie mit dem Konnektor verwenden. Legen Sie dazu eine Lambda-Umgebungsvariable fest, die dem Katalognamen entspricht, den Sie in Athena verwenden möchten. Angenommen, Sie möchten die folgenden Abfragen verwenden, um zwei verschiedene HBase-Instances von Athena abzufragen:

```
SELECT * FROM "hbase_instance_1".database.table
```

```
SELECT * FROM "hbase_instance_2".database.table
```

Bevor Sie diese beiden SQL-Anweisungen verwenden können, müssen Sie Ihrer Lambda-Funktion zwei Umgebungsvariablen hinzufügen: `hbase_instance_1` und `hbase_instance_2`. Der Wert für jede Verbindung sollte eine HBase-Verbindungszeichenfolge mit folgendem Format sein:

```
master_hostname:hbase_port:zookeeper_port
```

Verwendung von Secrets

Sie können optional AWS Secrets Manager für einen Teil oder den gesamten Wert für die Details Ihrer Verbindungszeichenfolge verwenden. Um das Athena-Federated-Query-Feature mit Secrets Manager zu verwenden, sollte die mit Ihrer Lamba-Funktion verbundene VPC über einen Internetzugang oder einen VPC-Endpunkt verfügen, um eine Verbindung zu Secrets Manager herzustellen.

Wenn Sie die Syntax `${my_secret}` verwenden, um den Namen eines Secrets aus Secrets Manager in Ihre Verbindungszeichenfolge einzufügen, ersetzt der Konnektor den geheimen Namen durch Ihren Benutzernamen und die Kennwortwerte aus Secrets Manager.

Angenommen, Sie setzen die Lambda-Umgebungsvariable für `hbase_instance_1` auf den folgenden Wert:

```
${hbase_host_1}:${hbase_master_port_1}:${hbase_zookeeper_port_1}
```

Das Athena Query Federation SDK versucht automatisch, ein Secret mit dem Namen `hbase_instance_1_creds` vom Secrets Manager abzurufen und setzt diesen Wert anstelle von `${hbase_instance_1_creds}`. Ein beliebiger Teil der Verbindungszeichenfolge, der in der `${ }`-Zeichenkombination enthalten ist, wird als Secret aus Secrets Manager interpretiert. Wenn Sie einen geheimen Namen angeben, den der Konnektor in Secrets Manager nicht finden kann, ersetzt der Konnektor den Text nicht.

Einrichten von Datenbanken und Tabellen in AWS Glue

Die integrierte Schemainferenz des Konnektors unterstützt nur Werte, die in HBase als Strings serialisiert sind (z. B. `String.valueOf(int)`). Da die integrierten Schemainferenzfunktionen des Konnektors begrenzt sind, sollten Sie stattdessen AWS Glue für Metadaten verwenden. Für die Aktivierung einer AWS Glue-Tabelle für die Verwendung mit HBase benötigen Sie eine AWS Glue-Datenbank und -Tabelle mit Namen, die mit dem HBase-Namensbereich und der -Tabelle übereinstimmen, für die Sie zusätzliche Metadaten bereitstellen möchten. Die Verwendung der Namenskonventionen für HBase-Spaltenfamilien ist optional, aber nicht erforderlich.

Verwenden Sie eine AWS Glue-Tabelle für ergänzende Metadaten wie folgt

1. Wenn Sie die Tabelle und die Datenbank in der AWS Glue-Konsole bearbeiten, fügen Sie die folgenden Tabelleneigenschaften hinzu:

- hbase-metadata-flag – Diese Eigenschaft gibt dem HBase-Konnektor an, dass der Konnektor die Tabelle für ergänzende Metadaten verwenden kann. Sie können einen beliebigen Wert für `hbase-metadata-flag` angeben, solange die `hbase-metadata-flag`-Eigenschaft in der Liste der Tabelleneigenschaften vorhanden ist.
- hbase-native-storage-flag – Verwenden Sie dieses Flag, um die beiden vom Konnektor unterstützten Wertserialisierungsmodi umzuschalten. Wenn dieses Feld nicht vorhanden ist, geht der Konnektor standardmäßig davon aus, dass alle Werte in HBase als Zeichenfolge gespeichert sind. Daher wird er versuchen, Datentypen wie `INT`, `BIGINT` und `DOUBLE` von HBase als Zeichenfolgen zu analysieren. Wenn dieses Feld mit einem beliebigen Wert in der Tabelle in AWS Glue festgelegt ist, wechselt der Konnektor in den „nativen“ Speichermodus und versucht `INT`, `BIGINT`, `BIT` und `DOUBLE` als Byte mithilfe der folgenden Funktionen zu lesen:

```
ByteBuffer.wrap(value).getInt()
ByteBuffer.wrap(value).getLong()
ByteBuffer.wrap(value).get()
ByteBuffer.wrap(value).getDouble()
```

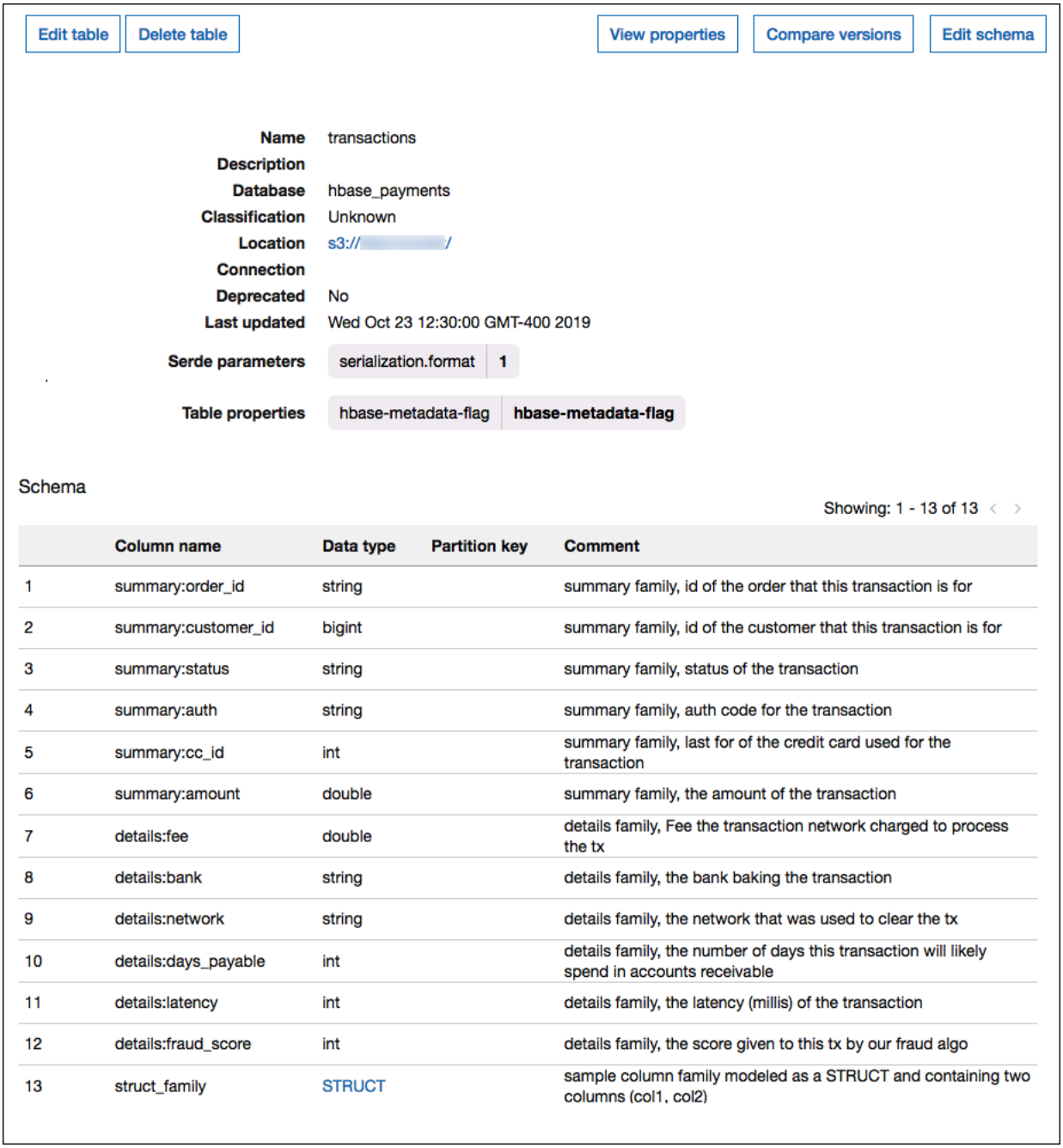
2. Achten Sie darauf, Datentypen zu verwenden, die für AWS Glue geeignet sind, wie in diesem Dokument aufgeführt.

Modellieren von Spaltenfamilien

Der Athena-HBase-Konnektor unterstützt zwei Möglichkeiten zur Modellierung von HBase-Spaltenfamilien: vollqualifizierte (abgeflachte) Benennung wie `family:column` oder die Verwendung von `STRUCT`-Objekten.

Im `STRUCT`-Model sollte der Name des `STRUCT`-Feldes mit der Spaltenfamilie übereinstimmen und die untergeordneten Elementen von `STRUCT` sollten mit den Namen der Spalten der Familie übereinstimmen. Da Prädikat-Push-Down- und Columnar-Lesevorgänge jedoch für komplexe Typen wie `STRUCT` noch nicht vollständig unterstützt werden, wird die Verwendung von `STRUCT` derzeit nicht empfohlen.

Das folgende Image zeigt eine in AWS Glue konfigurierte Tabelle, die eine Kombination der beiden Ansätze verwendet.



Datentypunterstützung

Der Konnektor ruft alle HBase-Werte als Basis-Byte-Typ ab. Dann, basierend darauf, wie Sie Ihre Tabellen im AWS Glue-Datenkatalog definiert haben, ordnet er die Werte einem der Apache Arrow-Datentypen in der folgenden Tabelle zu.

AWS Glue-Datentyp	Apache Arrow-Datentyp
int	INT
bigint	BIGINT
double	FLOAT8
float	FLOAT4
boolesch	BIT
Binary	VARBINARY
Zeichenfolge	VARCHAR

> **Note**
>
> Wenn Sie AWS Glue nicht verwenden, um Ihre Metadaten zu ergänzen, verwendet die Schema-Inferenzierung des Konnektors nur die Datentypen `BIGINT`, `FLOAT8` und `VARCHAR`.

Erforderliche Berechtigungen

Ausführliche Informationen zu den für diesen Konnektor erforderlichen IAM-Richtlinien finden Sie im `Policies`-Abschnitt der athena-hbase.yaml-Datei. In der folgenden Liste sind die erforderlichen Berechtigungen zusammengefasst.

- Amazon-S3-Schreibzugriff – Der Konnektor benötigt Schreibzugriff auf einen Speicherort in Amazon S3, um Ergebnisse aus großen Abfragen zu übertragen.
- Athena GetQueryExecution – Der Konnektor verwendet diese Berechtigung, um ein Fast-Fail durchzuführen, wenn die vorgeschaltete Athena-Abfrage beendet wurde.
- AWS Glue Data Catalog – Der HBase-Konnektor benötigt schreibgeschützten Zugriff auf AWS Glue Data Catalog, um Schemainformationen zu erhalten.
- CloudWatch Logs – Der Konnektor benötigt Zugriff auf CloudWatch Logs zum Speichern von Protokollen.

- AWS Secrets Manager-Lesezugriff – Wenn Sie HBase-Endpunktdetails in Secrets Manager speichern möchten, müssen Sie dem Konnektor Zugriff auf diese Secrets gewähren.
- Zugriff auf VPC – Der Konnektor erfordert die Fähigkeit, Schnittstellen an Ihre VPC anzuhängen und zu trennen, damit diese eine Verbindung zu dieser herstellen und mit Ihren HBase-Instances kommunizieren kann.

Leistung

Der Athena-HBase-Konnektor versucht, Abfragen für Ihre HBase-Instance zu parallelisieren, indem er jeden Regionsserver parallel liest. Der Athena-HBase-Konnektor führt einen Prädikat-Pushdown durch, um die von der Abfrage durchsuchten Daten zu reduzieren.

Die Lambda-Funktion führt auch Projektions-Pushdown durch, um die von der Abfrage gescannten Daten zu reduzieren. Die Auswahl einer Teilmenge von Spalten führt jedoch manchmal zu einer längeren Laufzeit der Abfrageausführung. `LIMIT`-Klauseln reduzieren die Menge der gescannten Daten, aber wenn Sie kein Prädikat angeben, sollten Sie davon ausgehen, dass `SELECT`-Abfragen mit einer `LIMIT`-Klausel mindestens 16 MB an Daten scannen.

HBase ist anfällig für Abfragefehler und schwankende Abfrageausführungszeiten. Möglicherweise müssen Sie Ihre Abfragen mehrmals wiederholen, damit diese erfolgreich sind. Der HBase-Konnektor ist aufgrund der Gleichzeitigkeit widerstandsfähig gegenüber Drosselung.

Lizenzinformationen

Das Amazon-Athena-HBase-Konnektor-Projekt ist lizenziert unter Apache-2.0-Lizenz.

Weitere Informationen finden Sie auch unter

Weitere Informationen zu diesem Konnektor finden Sie unter der entsprechenden Seite auf GitHub.com.

Amazon Athena Hortonworks Konnektor

Der Amazon Athena-Konnektor für Hortonworks ermöglicht es Amazon Athena, SQL-Abfragen auf der Cloudera-Hortonworks-Datenplattform auszuführen. Der Konnektor wandelt Ihre Athena-SQL-Abfragen in ihre äquivalente HiveQL-Syntax um.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor

bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Einschränkungen

- Schreiboperationen wie DDL werden nicht unterstützt.
- In einem Multiplexer-Setup werden der Überlauf-Bucket und das Präfix von allen Datenbank-Instances gemeinsam genutzt.
- Alle relevanten Lambda-Grenzwerte. Weitere Informationen finden Sie unter Lambda quotas (Lambda-Kontingente) im AWS Lambda -Entwicklerhandbuch.

Bedingungen

Die folgenden Begriffe beziehen sich auf den Hortonworks Hive-Konnektor.

- Datenbank-Instance – Jede Instance einer Datenbank, die On-Premises, in Amazon EC2 oder auf Amazon RDS bereitgestellt wird.
- Handler – Ein Lambda-Handler, der auf Ihre Datenbank-Instance zugreift. Ein Handler kann für Metadaten oder für Datensätze verwendet werden.
- Metadaten-Handler – Ein Lambda-Handler, der Metadaten von Ihrer Datenbank-Instance abruft.
- Record Handler – Ein Lambda-Handler, der Datensätze aus Ihrer Datenbank-Instance abruft.
- Composite Handler – Ein Lambda-Handler, der sowohl Metadaten als auch Datensätze aus Ihrer Datenbank-Instance abruft.
- Eigenschaft oder Parameter – Eine Datenbankeigenschaft, die von Handlern zum Extrahieren von Datenbankinformationen verwendet wird. Sie konfigurieren diese Eigenschaften als Lambda-Umgebungsvariablen.
- Verbindungszeichenfolge – Eine Textzeichenfolge, die verwendet wird, um eine Verbindung zu einer Datenbank-Instance herzustellen.
- Katalog — Ein nicht bei Athena registrierter AWS Glue Katalog, der ein erforderliches Präfix für die `connection_string` Immobilie ist.
- Multiplex-Handler – Ein Lambda-Handler, der mehrere Datenbankverbindungen akzeptieren und verwenden kann.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den Hortonworks-Hive-Konnektor zu konfigurieren.

Verbindungszeichenfolge

Verwenden Sie eine JDBC-Verbindungszeichenfolge im folgenden Format, um eine Verbindung zu einer Datenbank-Instance herzustellen.

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

Verwenden eines Multiplexing-Handlers

Sie können einen Multiplexer verwenden, um mit einer einzigen Lambda-Funktion eine Verbindung zu mehreren Datenbank-Instances herzustellen. Anfragen werden anhand des Katalognamens weitergeleitet. Verwenden Sie die folgenden Klassen in Lambda.

Handler	Klasse
Composite Handler	`HiveMuxCompositeHandler`
Metadaten-Handler	`HiveMuxMetadataHandler`
Record Handler	`HiveMuxRecordHandler`

Multiplex-Handler-Parameter

Parameter	Beschreibung
$*catalog*_connection_string	Erforderlich Eine Verbindungszeichenfolge einer Datenbank-Instance. Stellen Sie der Umgebungsvariablen den Namen des in Athena verwendeten Katalogs voran. Wenn zum Beispiel der bei Athena registrierte Katalog `myhivecatalog` ist, dann lautet der Name der Umgebungsvariablen `myhivecatalog_connection_string` .

Parameter	Beschreibung
`default`	Erforderlich Die standardmäßige Verbindungszeichenfolge. Diese Zeichenfolge wird verwendet, wenn der Katalog `lambda:${` *AWS_LAMBDA_FUNCTION_NAME* `}` ist.

Die folgenden Beispieleigenschaften gelten für eine Hive MUX Lambda-Funktion, die zwei Datenbankinstanzen unterstützt: `hive1` (die Standardeinstellung) und `hive2`.

Eigenschaft	Wert
`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`

Bereitstellen von Anmeldeinformationen

Um einen Benutzernamen und ein Kennwort für Ihre Datenbank in Ihrer JDBC-Verbindungszeichenfolge anzugeben, können Sie Eigenschaften von Verbindungszeichenfolgen oder AWS Secrets Manager verwenden.

- Verbindungszeichenfolge – Ein Benutzername und ein Kennwort können als Eigenschaften in der JDBC-Verbindungszeichenfolge angegeben werden.

> **Important**
>
> Als bewährte Sicherheitsmethode sollten Sie keine fest kodierten Anmeldeinformationen in Ihren Umgebungsvariablen oder Verbindungszeichenfolgen verwenden. Informationen zum Verschieben von hartcodierten Geheimnissen nach finden Sie im AWS Secrets Manager Benutzerhandbuch unter Verschieben von hartcodierten AWS Secrets Manager Geheimnissen nach.AWS Secrets Manager

- AWS Secrets Manager— Um die Athena Federated Query-Funktion verwenden zu können AWS Secrets Manager, muss die mit Ihrer Lambda-Funktion verbundene VPC über Internetzugang oder einen VPC-Endpunkt verfügen, um eine Verbindung zu Secrets Manager herzustellen.

  Sie können den Namen eines Geheimnisses in AWS Secrets Manager Ihre JDBC-Verbindungszeichenfolge eingeben. Der Konnektor ersetzt den geheimen Namen durch `username`- und `password`-Werte von Secrets Manager.

  Für Amazon RDS-Datenbank-Instances ist diese Unterstützung eng integriert. Wenn Sie Amazon RDS verwenden, empfehlen wir dringend, eine Rotation der Anmeldeinformationen zu verwenden AWS Secrets Manager . Wenn Ihre Datenbank Amazon RDS nicht verwendet, speichern Sie die Anmeldeinformationen als JSON im folgenden Format:

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

Beispiel einer Verbindungszeichenfolge mit einem geheimen Namen

Die folgende Zeichenfolge hat den geheimen Namen `${Test/RDS/hive1host}`.

```
hive://jdbc:hive2://hive1host:10000/default?...&${Test/RDS/hive1host}&...
```

Der Konnektor verwendet den geheimen Namen, um Secrets abzurufen und den Benutzernamen und das Kennwort bereitzustellen, wie im folgenden Beispiel gezeigt.

```
hive://jdbc:hive2://hive1host:10000/default?...&UID=sample2&PWD=sample2&...
```

Derzeit erkennt der Hortonworks-Hive-Konnektor die `UID`- und `PWD`-JDBC-Eigenschaften.

Verwenden eines einzelnen Verbindungs-Handlers

Sie können die folgenden Einzelverbindungsmetadaten und Record Handler verwenden, um eine Verbindung zu einer einzelnen Hortonworks Hive-Instance herzustellen.

Handler-Typ	Klasse
Composite Handler	`HiveCompositeHandler`
Metadaten-Handler	`HiveMetadataHandler`

Handler-Typ	Klasse
Record Handler	`HiveRecordHandler`

Parameter für Einzelverbindungs-Handler

Parameter	Beschreibung
`default`	Erforderlich Die standardmäßige Verbindungszeichenfolge.

Die Einzelverbindungs-Handler unterstützen eine Datenbank-Instance und müssen einen `default`-Verbindungszeichenfolgenparameter bereitstellen. Alle anderen Verbindungszeichenfolgen werden ignoriert.

Die folgende Beispieleigenschaft gilt für eine einzelne Hortonworks Hive-Instance, die von einer Lambda-Funktion unterstützt wird.

Eigenschaft	Wert
`default`	`hive://jdbc:hive2://hive1host:10000/default?secret=${Test/RDS/hive1host}`

Überlauf-Parameter

Das Lambda-SDK kann Daten an Amazon S3 übertragen. Alle Datenbank-Instances, auf die mit derselben Lambda-Funktion zugegriffen wird, werden an denselben Speicherort verschoben.

Parameter	Beschreibung
`spill_bucket`	Erforderlich Überlauf-Bucket-Name.
`spill_prefix`	Erforderlich Schlüssel-Prefix für den Überlauf-Bucket.
`spill_put_request_headers`	(Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject` -

Parameter	Beschreibung
	Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Weitere mögliche Header finden Sie PutObjectin der Amazon Simple Storage Service API-Referenz.

Datentypunterstützung

Die folgende Tabelle zeigt die entsprechenden Datentypen für JDBC, Hortonworks Hive und Arrow.

JDBC	Hortonworks Hive	Arrow
Boolesch	Boolesch	Bit
Ganzzahl	TINYINT	Tiny
Short	SMALLINT	Smallint
Ganzzahl	INT	Int
Long	BIGINT	Bigint
float	float4	Float4
Double	float8	Float8
Datum	date	DateDay
Zeitstempel	Zeitstempel	DateMilli
String	VARCHAR	Varchar
Bytes	bytes	Varbinary
BigDecimal	Dezimal	Dezimal
ARRAY	N.z. (siehe Hinweis)	Auflisten

> **Note**
>
> Derzeit unterstützt Hortonworks Hive nicht die Aggregattypen `ARRAY`, `MAP`, `STRUCT` oder `UNIONTYPE`. Spalten mit Aggregattypen werden als `VARCHAR`-Spalten in SQL behandelt.

Partitionen und Splits

Partitionen werden verwendet, um zu bestimmen, wie Splits für den Konnektor generiert werden. Athena konstruiert eine synthetische Säule vom Typ `varchar`, die das Partitionierungsschema für die Tabelle darstellt, das dem Konnektor beim Generieren von Splits hilft. Der Konnektor ändert nicht die eigentliche Tabellendefinition.

Leistung

Hortonworks Hive unterstützt statische Partitionen. Der Athena-Hortonworks-Hive-Konnektor kann Daten von diesen Partitionen parallel abrufen. Wenn Sie sehr große Datenmengen mit einheitlicher Partitionsverteilung abfragen möchten, wird eine statische Partitionierung dringend empfohlen. Die Auswahl einer Teilmenge von Spalten beschleunigt die Abfragelaufzeit erheblich und reduziert die gescannten Daten. Der Hortonworks-Hive-Konnektor ist aufgrund der Gleichzeitigkeit widerstandsfähig gegenüber Drosselung.

Der Athena-Hortonworks-Hive-Konnektor führt einen Prädikat-Pushdown durch, um die von der Abfrage gescannten Daten zu verringern. `LIMIT`-Klauseln, einfache Prädikate und komplexe Ausdrücke werden an den Konnektor weitergegeben, um die Menge der gescannten Daten und die Laufzeit der Abfrage zu verringern.

LIMIT-Klauseln

Eine `LIMIT N`-Anweisung reduziert die von der Abfrage durchsuchten Daten. Mit `LIMIT N`-Pushdown gibt der Konnektor nur `N` Zeilen an Athena zurück.

Prädikate

Ein Prädikat ist ein Ausdruck in der `WHERE`-Klausel einer SQL-Abfrage, der einen booleschen Wert ergibt und Zeilen auf der Grundlage mehrerer Bedingungen filtert. Der Athena-Hortonworks-Hive-Konnektor kann diese Ausdrücke kombinieren und sie direkt an Hortonworks Hive weiterleiten, um die Funktionalität zu erweitern und die Menge der gescannten Daten zu reduzieren.

Die folgenden Operator des Athena-Hortonworks-Hive-Konnektors unterstützen Prädikat-Pushdown:

- Boolean: UND, ODER, NICHT
- Gleichheit: GLEICH, NICHT GLEICH, WENIGER_ALS, WENIGER_ODER_GLEICH, GRÖSSER_ALS, GRÖSSER_ODER_GLEICH, IST_NULL
- Arithmetik: ADDIEREN, SUBTRAHIEREN, MULTIPLIZIEREN, DIVIDIEREN, MODULIEREN, NEGIEREN
- Andere: WIE_MUSTER, IN

Beispiel für einen kombinierten Pushdown

Kombinieren Sie für erweiterte Abfragefunktionen die Pushdown-Typen wie im folgenden Beispiel:

```
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;
```

Lizenzinformationen

Durch die Verwendung dieses Connectors erkennen Sie die Einbindung von Komponenten von Drittanbietern an. Eine Liste dieser Komponenten finden Sie in der Datei pom.xml für diesen Connector und stimmen den Bedingungen der jeweiligen Drittanbieterlizenzen zu, die in der Datei LICENSE.txt auf GitHub .com enthalten sind.

Weitere Informationen finden Sie auch unter

Die neuesten Informationen zur JDBC-Treiberversion finden Sie in der Datei pom.xml für den Hortonworks Hive-Connector auf .com. GitHub

Weitere Informationen zu diesem Connector finden Sie auf der entsprechenden Website unter .com. GitHub

Apache-Kafka-Konnektor von Amazon Athena

Der Amazon-Athena-Konnektor für Apache Kafka ermöglicht es Amazon Athena, SQL-Abfragen für Ihre Apache-Kafka-Themen auszuführen. Verwenden Sie diesen Konnektor, um Apache-Kafka-Themen als Tabellen und Nachrichten als Zeilen in Athena anzuzeigen.

Voraussetzungen

Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Einschränkungen

- Schreiboperationen wie DDL werden nicht unterstützt.
- Alle relevanten Lambda-Grenzwerte. Weitere Informationen finden Sie unter Lambda quotas (Lambda-Kontingente) im AWS Lambda-Entwicklerhandbuch.
- Datums- und Zeitstempeldatentypen in Filterbedingungen müssen in geeignete Datentypen umgewandelt werden.
- Datums- und Zeitstempel-Datentypen werden für den CSV-Dateityp nicht unterstützt und als Varchar-Werte behandelt.
- Die Zuordnung zu verschachtelten JSON-Feldern wird nicht unterstützt. Der Konnektor ordnet nur Felder der obersten Ebene zu.
- Der Konnektor unterstützt keine komplexen Typen. Komplexe Typen werden als Zeichenfolgen interpretiert.
- Verwenden Sie zum Extrahieren oder Arbeiten mit komplexen JSON-Werten die in Athena verfügbaren JSON-bezogenen Funktionen. Weitere Informationen finden Sie unter Extrahieren von Daten aus JSON.
- Der Konnektor unterstützt keinen Zugriff auf Kafka-Nachrichtenmetadaten.

Bedingungen

- Metadaten-Handler – Ein Lambda-Handler, der Metadaten von Ihrer Datenbank-Instance abruft.
- Record Handler – Ein Lambda-Handler, der Datensätze aus Ihrer Datenbank-Instance abruft.
- Composite Handler – Ein Lambda-Handler, der sowohl Metadaten als auch Datensätze aus Ihrer Datenbank-Instance abruft.
- Kafka-Endpunkt – Eine Textzeichenfolge, die eine Verbindung zu einer Kafka-Instance herstellt.

Cluster-Kompatibilität

Der Kafka-Konnektor kann mit den folgenden Cluster-Typen verwendet werden.

- Eigenständiges Kafka – Eine direkte Verbindung zu Kafka (authentifiziert oder unauthentifiziert).
- Confluent – Eine direkte Verbindung zu Confluent Kafka. Informationen zur Verwendung von Athena mit Confluent-Kafka-Daten finden Sie unter Visualisieren von Confluent-Daten in Amazon QuickSight mithilfe von Amazon Athena im AWS-Business-Intelligence-Blog.

Zu Confluent verbinden

Die Verbindung zu Confluent erfordert die folgenden Schritte:

1. Generieren Sie einen API-Schlüssel von Confluent.
2. Speichern Sie den Benutzernamen und das Passwort für den Confluent-API-Schlüssel in AWS Secrets Manager.
3. Geben Sie den geheimen Namen für die `secrets_manager_secret`-Umgebungsvariable im Kafka-Konnektor an.
4. Folgen Sie den Schritten im Den Kafka-Konnektor einrichten-Abschnitt dieses Dokuments.

Unterstützte Authentifizierungsmethoden

Der Konnektor unterstützt die folgenden Authentifizierungsmethoden.

- SSL
- SASL/SCRAM
- SASL/PLAIN
- SASL/PLAINTEXT
- NO_AUTH
- Selbstverwaltete Kafka- und Confluent-Plattform – SSL, SASL/SCRAM, SASL/PLAINTEXT, NO_AUTH
- Selbstverwaltetes Kafka und Confluent Cloud – SASL/PLAIN

Weitere Informationen finden Sie unter Konfigurieren der Authentifizierung für den Athena-Kafka-Konnektor.

Unterstützte Eingabedatenformate

Der Konnektor unterstützt die folgenden Eingabedatenformate.

- JSON
- CSV

Parameter

Verwenden Sie die in diesem Abschnitt erwähnten Lambda-Umgebungsvariablen, um den Athena-Kafka-Konnektor zu konfigurieren.

- auth_type – Gibt den Authentifizierungstyp des Clusters an. Der Konnektor unterstützt die folgenden Arten der Authentifizierung:
  - NO_AUTH – Stellen Sie eine direkte Verbindung zu Kafka her (z. B. zu einem Kafka-Cluster, der über eine EC2-Instance bereitgestellt wird, die keine Authentifizierung verwendet).
  - SASL_SSL_PLAIN – Diese Methode verwendet das `SASL_SSL`-Sicherheitsprotokoll und den `PLAIN`-SASL-Mechanismus. Weitere Informationen finden Sie unter SASL-Konfiguration in der Apache-Kafka-Dokumentation.
  - SASL_PLAINTEXT_PLAIN – Diese Methode verwendet das `SASL_PLAINTEXT`-Sicherheitsprotokoll und den `PLAIN`-SASL-Mechanismus. Weitere Informationen finden Sie unter SASL-Konfiguration in der Apache-Kafka-Dokumentation.
  - SASL_SSL_SCRAM_SHA512 – Mit diesem Authentifizierungstyp können Sie den Zugriff auf Ihre Apache-Kafka-Cluster steuern. Diese Methode speichert den Benutzernamen und das Passwort in AWS Secrets Manager. Das Geheimnis muss dem Kafka-Cluster zugeordnet sein. Weitere Informationen finden Sie unter Authentifizierung mit SASL/SCRAM in der Apache-Kafka-Dokumentation.
  - SASL_PLAINTEXT_SCRAM_SHA512 – Diese Methode verwendet das `SASL_PLAINTEXT`-Sicherheitsprotokoll und den `SCRAM_SHA512` `SASL`-Mechanismus. Diese Methode verwendet den in AWS Secrets Manager gespeicherten Benutzernamen und das Passwort. Weitere Informationen finden Sie unter SASL-Konfiguration in der Apache-Kafka-Dokumentation.
  - SSL – SSL-Authentifizierung verwendet Schlüsselspeicher- und Vertrauensspeicherdateien, um eine Verbindung mit dem Apache-Kafka-Cluster herzustellen. Sie müssen die Trust-Store- und Key-Store-Dateien generieren, sie in einen Amazon-S3-Bucket hochladen und die Referenz auf Amazon S3 angeben, wenn Sie den Konnektor bereitstellen. Der Schlüsselspeicher, der Vertrauensspeicher und der SSL-Schlüssel werden in AWS Secrets Manager gespeichert. Ihr Client muss den geheimen AWS-Schlüssel bereitstellen, wenn der Konnektor bereitgestellt wird. Weitere Informationen finden Sie unter Authentifizierung mit SSL in der Apache-Kafka-Dokumentation.

Weitere Informationen finden Sie unter Konfigurieren der Authentifizierung für den Athena-Kafka-Konnektor.

- certificates_s3_reference – Der Amazon-S3-Speicherort, der die Zertifikate enthält (die Schlüsselspeicher- und Vertrauensspeicherdateien).
- disable_spill_encryption – (Optional) Bei Einstellung auf `True`, wird die Spill-Verschlüsselung deaktiviert. Die Standardeinstellung ist `False`, sodass Daten, die an S3 übertrragen werden, mit AES-GCM verschlüsselt werden - entweder mit einem zufällig generierten Schlüssel oder mit KMS zum Generieren von Schlüsseln. Das Deaktivieren der Überlauf-Verschlüsselung kann die Leistung verbessern, insbesondere wenn Ihr Überlauf-Standort eine serverseitige Verschlüsselung verwendet.
- kafka_endpoint – Die Endpunktdetails, die Kafka bereitgestellt werden sollen.
- secrets_manager_secret – Der Name des AWS-Geheimnisses, in dem die Anmeldeinformationen gespeichert sind.
- Überlauf-Parameter – Lambda-Funktionen speichern vorübergehend („Überlauf“) Daten, die nicht in den Speicher von Amazon S3 passen. Alle Datenbank-Instances, auf die mit derselben Lambda-Funktion zugegriffen wird, werden an denselben Speicherort verschoben. Verwenden Sie die Parameter in der folgenden Tabelle, um den Überlauf-Standort anzugeben.

Parameter	Beschreibung
`spill_bucket`	Erforderlich. Der Name des Amazon-S3-Buckets, in den die Lambda-Funktion Daten übertragen kann.
`spill_prefix`	Erforderlich. Das Präfix innerhalb des Überlauf-Buckets, in dem die Lambda-Funktion Daten ausgeben kann.
`spill_put_request_headers`	(Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject` -Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Andere mögliche Header finden Sie unter PutObject in der API-Referenz zu Amazon Simple Storage Service.

- Subnetz-IDs – Eine oder mehrere Subnetz-IDs, die dem Subnetz entsprechen, das die Lambda-Funktion für den Zugriff auf Ihre Datenquelle verwenden kann.

- Öffentlicher Kafka-Cluster oder Standard-Confluent-Cloud-Cluster – Ordnen Sie den Konnektor einem privaten Subnetz zu, das über ein NAT-Gateway verfügt.
- Confluent-Cloud-Cluster mit privater Konnektivität – Ordnen Sie den Konnektor einem privaten Subnetz zu, das eine Route zum Confluent-Cloud-Cluster hat.
  - Für AWS-Transit-Gateway müssen sich die Subnetze in einer VPC befinden, die an dasselbe Transit-Gateway angeschlossen ist, das Confluent Cloud verwendet.
  - Für VPC-Peering müssen sich die Subnetze in einer VPC befinden, die per Peering mit Confluent-Cloud-VPC verbunden ist.
  - Für AWS PrivateLink müssen sich die Subnetze in einer VPC befinden die eine Route zu den VPC-Endpunkten hat, die eine Verbindung zu Confluent Cloud herstellen.

> **Note**
>
> Wenn Sie den Konnektor in einer VPC bereitstellen, um auf private Ressourcen zuzugreifen und auch eine Verbindung zu einem öffentlich zugänglichen Service wie Confluent herstellen möchten, müssen Sie den Konnektor einem privaten Subnetz zuordnen, das über ein NAT-Gateway verfügt. Weitere Informationen finden Sie unter NAT-Gateways im Amazon-VPC-Benutzerhandbuch.

Datentypunterstützung

Die folgende Tabelle zeigt die entsprechenden Datentypen, die für Kafka und Apache Arrow unterstützt werden.

Kafka	Arrow
CHAR	VARCHAR
VARCHAR	VARCHAR
TIMESTAMP	MILLISEKUNDE
DATUM	TAG
BOOLEAN	BOOL

Kafka	Arrow
SMALLINT	SMALLINT
INTEGER	INT
BIGINT	BIGINT
DECIMAL	FLOAT8
DOUBLE	FLOAT8

Partitionen und Splits

Kafka-Themen sind in Partitionen unterteilt. Jede Partition ist geordnet. Jede Nachricht in einer Partition hat eine inkrementelle ID, die Offset genannt wird. Jede Kafka-Partition wird für die parallele Verarbeitung in weitere Splits unterteilt. Daten sind für den in Kafka-Clustern konfigurierten Aufbewahrungszeitraum verfügbar.

Bewährte Methoden

Verwenden Sie als bewährte Methode bei der Abfrage von Athena das Prädikat-Pushdown, wie in den folgenden Beispielen.

```
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'
```

Den Kafka-Konnektor einrichten

Bevor Sie den Konnektor verwenden können, müssen Sie Ihren Apache-Kafka-Cluster einrichten, die AWS Glue-Schema-Registry verwenden, um Ihr Schema zu definieren, und die Authentifizierung für den Konnektor konfigurieren.

Beacjten Sie bei der Arbeit mit dem AWS Glue-Schema-Registry die folgenden Punkte:

- Stellen Sie sicher, dass der Text im Feld Description (Beschreibung) des AWS Glue-Schema-Registry die Zeichenfolge `{AthenaFederationKafka}` enthält. Diese Markierungszeichenfolge ist für AWS Glue-Registrys erforderlich, die Sie mit dem Kafka-Konnektor von Amazon Athena verwenden.
- Verwenden Sie für Ihre Datenbanknamen und -tabellen nur Kleinbuchstaben, um eine optimale Leistung zu erzielen. Bei der Verwendung gemischter Groß- und Kleinschreibung führt der Konnektor eine Suche ohne Berücksichtigung der Groß- und Kleinschreibung durch, die rechenintensiver ist.

So richten Sie Ihre Apache-Kafka-Umgebung und das AWS Glue-Schema-Registry ein

1. Richten Sie Ihre Apache-Kafka-Umgebung ein.
2. Laden Sie die Kafka-Themenbeschreibungsdatei (d. h. das Schema) im JSON-Format in das AWS Glue-Schema-Registry hoch. Weitere Informationen finden Sie unter Integration mit AWS Glue-Schema-Registry im AWS Glue-Entwicklerhandbuch. Beispielschemata finden Sie im folgenden Abschnitt.

Schemabeispiele für das AWS Glue-Schema-Registry

Verwenden Sie das Format der Beispiele in diesem Abschnitt, wenn Sie Ihr Schema in das AWS Glue-Schema Registry hochladen.

Beispielschema für einen JSON-Typ

Im folgenden Beispiel gibt das Schema, das in der AWS Glue-Schema-Registry erstellt werden soll, `json` als Wert für `dataFormat` an und verwendet `datatypejson` für `topicName`.

> **Note**
>
> Der Wert für `topicName` sollte die gleiche Schreibweise wie der Themenname in Kafka verwenden.

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

Beispielschema für einen CSV-Typ

Im folgenden Beispiel gibt das Schema, das in der AWS Glue-Schema-Registry erstellt werden soll, `csv` als Wert für `dataFormat` an und verwendet `datatypecsvbulk` für `topicName`. Der Wert für `topicName` sollte die gleiche Schreibweise wie der Themenname in Kafka verwenden.

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

Konfigurieren der Authentifizierung für den Athena-Kafka-Konnektor

Sie können verschiedene Methoden verwenden, um sich bei Ihrem Apache-Kafka-Cluster zu authentifizieren, darunter SSL, SASL/SCRAM, SASL/PLAIN und SASL/PLAINTEXT.

Die folgende Tabelle zeigt die Authentifizierungstypen für den Konnektor sowie das jeweilige Sicherheitsprotokoll und den SASL-Mechanismus. Weitere Informationen zur Sicherheit von Kafka finden Sie unter Sicherheit in der Apache-Kafka-Dokumentation.

auth_type	security.protocol	sasl.mechanik	Cluster-Typ-Kompatibilität
`SASL_SSL_PLAIN`	`SASL_SSL`	`PLAIN`	• Selbstverwaltetes Kafka • Confluent-Plattform • Confluent-Cloud
`SASL_PLAINTEXT_PLAIN`	`SASL_PLAINTEXT`	`PLAIN`	• Selbstverwaltetes Kafka • Confluent-Plattform
`SASL_SSL_SCRAM_SHA512`	`SASL_SSL`	`SCRAM-SHA-512`	• Selbstverwaltetes Kafka • Confluent-Plattform

auth_type	security.protocol	sasl.mechanik	Cluster-Typ-Kompatibilität
`SASL_PLAINTEXT_SCRAM_SHA512`	`SASL_PLAINTEXT`	`SCRAM-SHA-512`	• Selbstverwaltetes Kafka • Confluent-Plattform
`SSL`	`SSL`	–	• Selbstverwaltetes Kafka • Confluent-Plattform

SSL

Wenn der Cluster SSL-authentifiziert ist, müssen Sie die Vertrauensspeicher- und Schlüsselspeicherdateien generieren und sie in den Amazon-S3-Bucket hochladen. Sie müssen diese Amazon-S3-Referenz angeben, wenn Sie den Konnektor bereitstellen. Der Schlüsselspeicher, der Vertrauensspeicher und der SSL-Schlüssel werden in der AWS Secrets Manager gespeichert. Sie geben den geheimen AWS-Schlüssel an, wenn Sie den Konnektor bereitstellen.

Informationen zum Erstellen eines Geheimnisses im Secrets Manager finden Sie unter Erstellen eines AWS Secrets Manager-Geheimnisses.

Um diesen Authentifizierungstyp zu verwenden, legen Sie die Umgebungsvariablen wie in der folgenden Tabelle gezeigt fest.

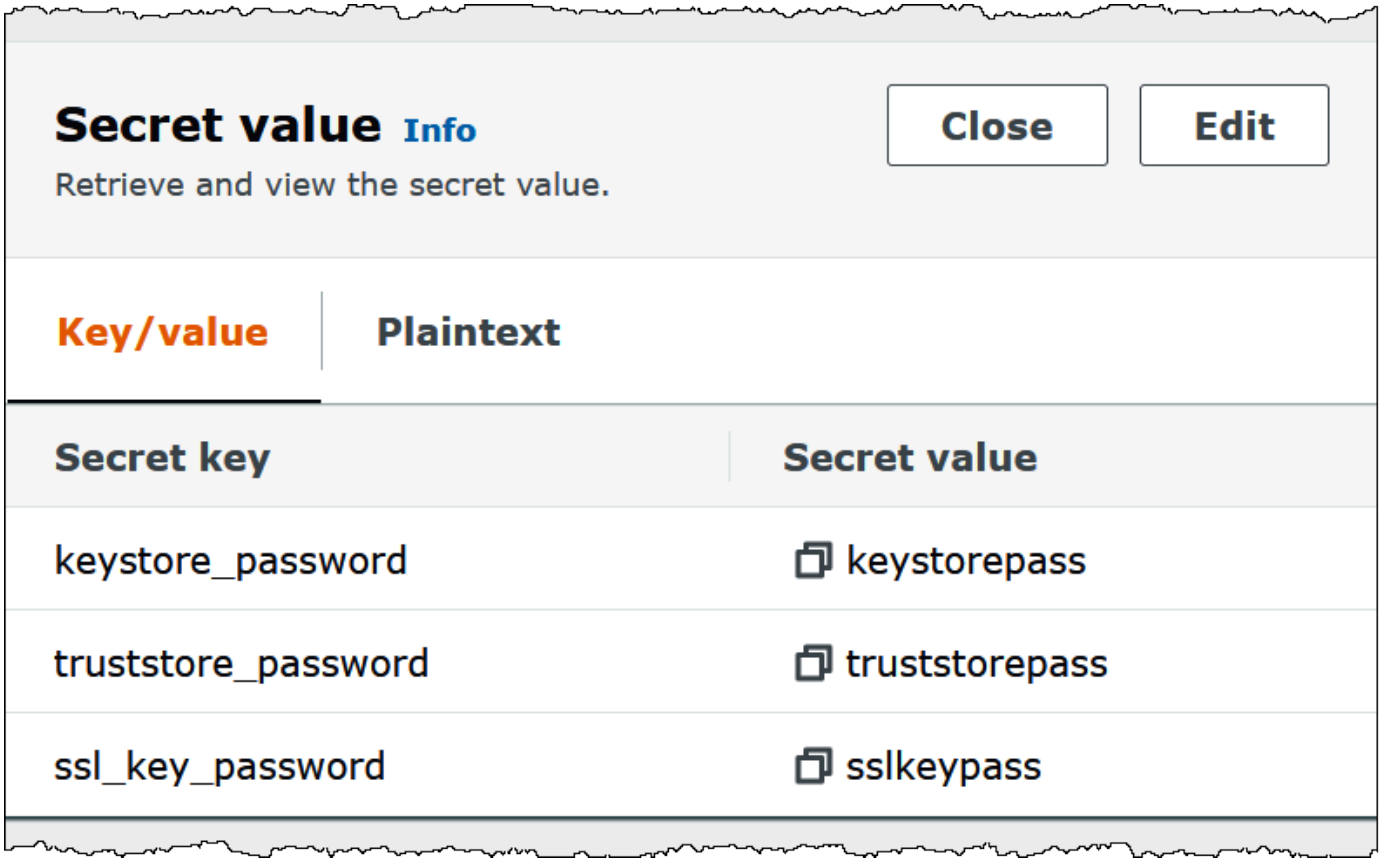
Parameter	Wert
`auth_type`	`SSL`
`certificates_s3_reference`	Der Amazon-S3-Speicherort, der die Zertifikate enthält.
`secrets_manager_secret`	Der Name Ihres geheimen AWS-Schlüssels.

Nachdem Sie ein Geheimnis in Secrets Manager erstellt haben, können Sie es in der Secrets-Manager-Konsole anzeigen.

So zeigen Sie Ihr Geheimnis in Secrets Manager an

1. Öffnen Sie die Secrets-Manager-Konsole unterhttps://console.aws.amazon.com/secretsmanager/.
2. Wählen Sie im Navigationsbereich Secrets (Geheimnisse).
3. Wählen Sie auf der Seite Secrets (Geheimnisse) den Link zu Ihrem Geheimnis aus.
4. Wählen Sie auf der Detailseite für Ihr Geheimnis die Option Retrieve secret value (Geheimniswert abrufen).

   Das folgende Image zeigt ein Beispiel für ein Geheimnis mit drei Schlüssel/Wert-Paaren: `keystore_password`, `truststore_password` und `ssl_key_password`.



Weitere Informationen zur Verwendung von SSL mit Kafka finden Sie unter Verschlüsselung und Authentifizierung mit SSL in der Apache-Kafka-Dokumentation.

SASL/SCRAM

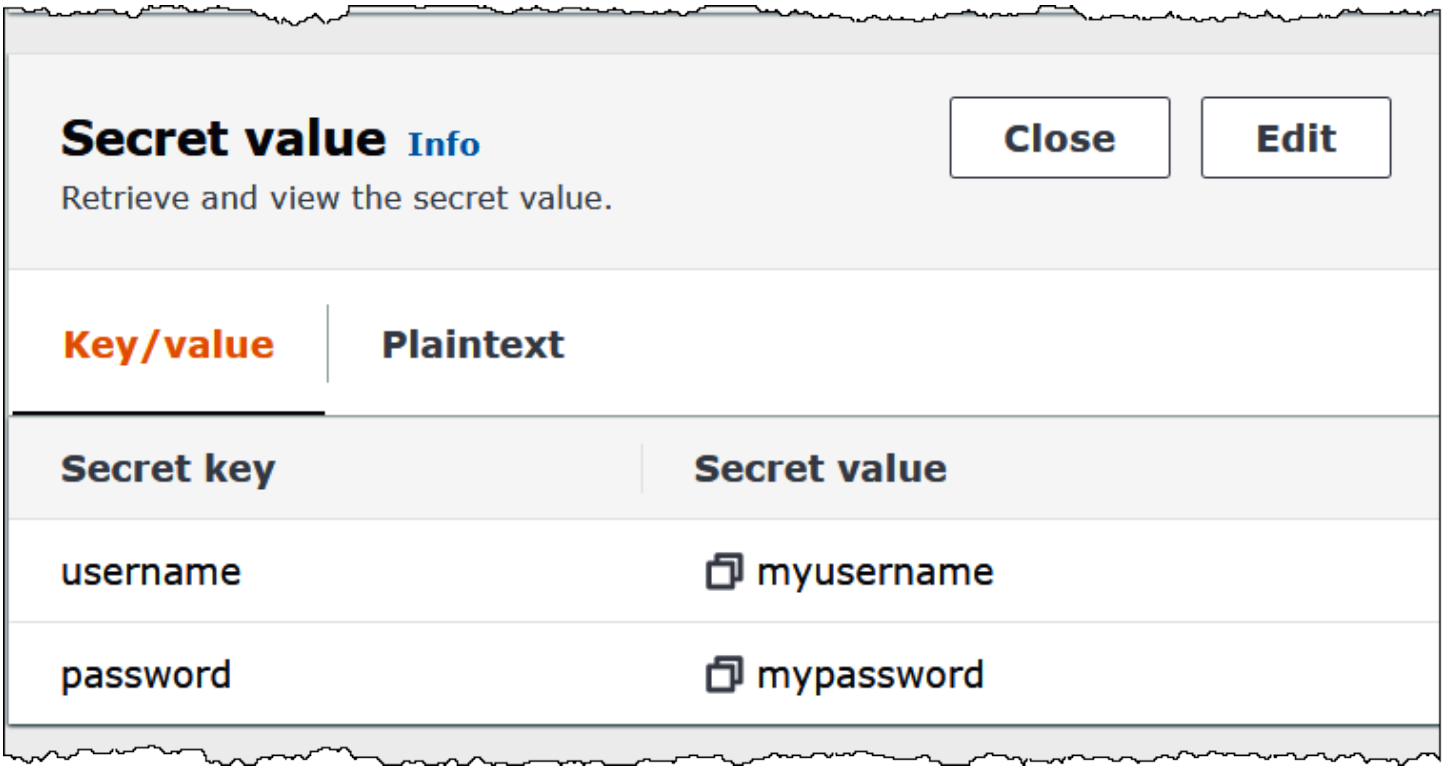
Wenn Ihr Cluster SCRAM-Authentifizierung verwendet, geben Sie bei der Bereitstellung des Konnektor den Secrets-Manager-Schlüssel an, der dem Cluster zugeordnet ist. Die AWS-

Anmeldeinformationen des Benutzers (geheimer Schlüssel und Zugriffsschlüssel) werden für die Authentifizierung mit dem Cluster verwendet.

Legen Sie die Umgebungsvariablen wie in der folgenden Tabelle angegeben fest.

Parameter	Wert
`auth_type`	`SASL_SSL_SCRAM_SHA512`
`secrets_manager_secret`	Der Name Ihres geheimen AWS-Schlüssels.

Das folgende Image zeigt ein Beispiel für ein Geheimnis in der Secrets-Manager-Konsole mit zwei Schlüssel/Wert-Paaren: eines für `username` und eines für `password`.



Weitere Informationen zur Verwendung von SASL/SCRAM mit Kafka finden Sie unter Authentifizierung mit SASL/SCRAM in der Apache-Kafka-Dokumentation.

Lizenzinformationen

Durch die Verwendung dieses Konnektors erkennen Sie die Aufnahme von Komponenten von Drittanbietern an. Eine Liste dieser Komponenten finden Sie in der pom.xml-Datei für diesen

Konnektor. Zudem stimmen Sie den Bedingungen der jeweiligen Drittanbieterlizenzen zu, die in der LICENSE.txt-Datei auf GitHub.com aufgeführt werden.

Weitere Informationen finden Sie auch unter

Weitere Informationen zu diesem Konnektor finden Sie unter der entsprechenden Seite auf GitHub.com.

Amazon-Athena-MSK-Konnektor

Der Amazon-Athena-Konnektor für Amazon MSK ermöglicht es Amazon Athena, SQL-Abfragen für Ihre Apache-Kafka-Themen auszuführen. Verwenden Sie diesen Konnektor, um Apache-Kafka-Themen als Tabellen und Nachrichten als Zeilen in Athena anzuzeigen. Weitere Informationen finden Sie unter Analysieren von Echtzeit-Streaming-Daten in Amazon MSK mit Amazon Athena im AWS-Big-Data-Blog.

Voraussetzungen

Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Einschränkungen

- Schreiboperationen wie DDL werden nicht unterstützt.
- Alle relevanten Lambda-Grenzwerte. Weitere Informationen finden Sie unter Lambda quotas (Lambda-Kontingente) im AWS Lambda-Entwicklerhandbuch.
- Datums- und Zeitstempeldatentypen in Filterbedingungen müssen in geeignete Datentypen umgewandelt werden.
- Datums- und Zeitstempel-Datentypen werden für den CSV-Dateityp nicht unterstützt und als Varchar-Werte behandelt.
- Die Zuordnung zu verschachtelten JSON-Feldern wird nicht unterstützt. Der Konnektor ordnet nur Felder der obersten Ebene zu.
- Der Konnektor unterstützt keine komplexen Typen. Komplexe Typen werden als Zeichenfolgen interpretiert.
- Verwenden Sie zum Extrahieren oder Arbeiten mit komplexen JSON-Werten die in Athena verfügbaren JSON-bezogenen Funktionen. Weitere Informationen finden Sie unter Extrahieren von Daten aus JSON.

- Der Konnektor unterstützt keinen Zugriff auf Kafka-Nachrichtenmetadaten.

Bedingungen

- Metadaten-Handler – Ein Lambda-Handler, der Metadaten von Ihrer Datenbank-Instance abruft.
- Record Handler – Ein Lambda-Handler, der Datensätze aus Ihrer Datenbank-Instance abruft.
- Composite Handler – Ein Lambda-Handler, der sowohl Metadaten als auch Datensätze aus Ihrer Datenbank-Instance abruft.
- Kafka-Endpunkt – Eine Textzeichenfolge, die eine Verbindung zu einer Kafka-Instance herstellt.

Cluster-Kompatibilität

Der MSK-Konnektor kann mit den folgenden Cluster-Typen verwendet werden.

- Von MSK bereitgestellter Cluster – Sie spezifizieren, überwachen und skalieren die Cluster-Kapazität manuell.
- Serverless MSK-Cluster – Bietet On-Demand-Kapazität, die automatisch skaliert wird, wenn die Anwendungs-I/O skaliert wird.
- Eigenständiges Kafka – Eine direkte Verbindung zu Kafka (authentifiziert oder unauthentifiziert).

Unterstützte Authentifizierungsmethoden

Der Konnektor unterstützt die folgenden Authentifizierungsmethoden.

- SASL/IAM
- SSL
- SASL/SCRAM
- SASL/PLAIN
- SASL/PLAINTEXT
- NO_AUTH

  Weitere Informationen finden Sie unter Konfigurieren der Authentifizierung für den Athena-MSK-Konnektor.

Unterstützte Eingabedatenformate

Der Konnektor unterstützt die folgenden Eingabedatenformate.

- JSON
- CSV

Parameter

Verwenden Sie die in diesem Abschnitt erwähnten Lambda-Umgebungsvariablen, um den Athena-MSK-Konnektor zu konfigurieren.

- auth_type – Gibt den Authentifizierungstyp des Clusters an. Der Konnektor unterstützt die folgenden Arten der Authentifizierung:
  - NO_AUTH – Stellen Sie eine direkte Verbindung zu Kafka ohne Authentifizierung her (z. B. zu einem Kafka-Cluster, der über eine EC2-Instance bereitgestellt wird, die keine Authentifizierung verwendet).
  - SASL_SSL_PLAIN – Diese Methode verwendet das `SASL_SSL`-Sicherheitsprotokoll und den `PLAIN`-SASL-Mechanismus.
  - SASL_PLAINTEXT_PLAIN – Diese Methode verwendet das `SASL_PLAINTEXT`-Sicherheitsprotokoll und den `PLAIN`-SASL-Mechanismus.

    > **Note**
    >
    > Die `SASL_SSL_PLAIN`- und `SASL_PLAINTEXT_PLAIN`-Authentifizierungstypen werden von Apache Kafka unterstützt, jedoch nicht von Amazon MSK.

  - SASL_SSL_AWS_MSK_IAM – Die IAM-Zugriffskontrolle für Amazon MSK ermöglicht es Ihnen, sowohl die Authentifizierung als auch die Autorisierung für Ihren MSK-Cluster zu verwalten. Die AWS-Anmeldeinformationen Ihres Benutzers (geheimer Schlüssel und Zugriffsschlüssel) werden zum Herstellen einer Verbindung mit dem Cluster verwendet. Weitere Informationen finden Sie unter IAM-Zugriffskontrolle im Entwicklerhandbuch für Amazon Managed Streaming for Apache Kafka.
  - SASL_SSL_SCRAM_SHA512 – Mit diesem Authentifizierungstyp können Sie den Zugriff auf Ihre Amazon-MSK-Cluster steuern. Diese Methode speichert den Benutzernamen und das Passwort auf AWS Secrets Manager. Das Geheimnis muss dem Amazon-MSK-Cluster zugeordnet sein. Weitere Informationen finden Sie unter Einrichten der SASL/SCRAM-Authentifizierung für einen

Amazon-MSK-Cluster im Entwicklerhandbuch für Amazon Managed Streaming for Apache Kafka.

- SSL – Die SSL-Authentifizierung verwendet Schlüsselspeicher- und Vertrauensspeicherdateien, um eine Verbindung mit dem Amazon-MSK-Cluster herzustellen. Sie müssen die Trust-Store- und Key-Store-Dateien generieren, sie in einen Amazon-S3-Bucket hochladen und die Referenz auf Amazon S3 angeben, wenn Sie den Konnektor bereitstellen. Der Schlüsselspeicher, der Vertrauensspeicher und der SSL-Schlüssel werden in AWS Secrets Manager gespeichert. Ihr Client muss den geheimen AWS-Schlüssel bereitstellen, wenn der Konnektor bereitgestellt wird. Weitere Informationen finden Sie unter Gegenseitige TLS-Authentifizierung im Entwicklerhandbuch für Amazon Managed Streaming for Apache Kafka.

  Weitere Informationen finden Sie unter Konfigurieren der Authentifizierung für den Athena-MSK-Konnektor.

- certificates_s3_reference – Der Amazon-S3-Speicherort, der die Zertifikate enthält (die Schlüsselspeicher- und Vertrauensspeicherdateien).
- disable_spill_encryption – (Optional) Bei Einstellung auf `True`, wird die Spill-Verschlüsselung deaktiviert. Die Standardeinstellung ist `False`, sodass Daten, die an S3 übertrragen werden, mit AES-GCM verschlüsselt werden - entweder mit einem zufällig generierten Schlüssel oder mit KMS zum Generieren von Schlüsseln. Das Deaktivieren der Überlauf-Verschlüsselung kann die Leistung verbessern, insbesondere wenn Ihr Überlauf-Standort eine serverseitige Verschlüsselung verwendet.
- kafka_endpoint – Die Endpunktdetails, die Kafka bereitgestellt werden sollen. Für einen Amazon MSK-Cluster stellen Sie beispielsweise eine Bootstrap-URL für den Cluster bereit.
- secrets_manager_secret – Der Name des AWS-Geheimnisses, in dem die Anmeldeinformationen gespeichert sind. Dieser Parameter ist für die IAM-Authentifizierung nicht erforderlich.
- Überlauf-Parameter – Lambda-Funktionen speichern vorübergehend („Überlauf“) Daten, die nicht in den Speicher von Amazon S3 passen. Alle Datenbank-Instances, auf die mit derselben Lambda-Funktion zugegriffen wird, werden an denselben Speicherort verschoben. Verwenden Sie die Parameter in der folgenden Tabelle, um den Überlauf-Standort anzugeben.

Parameter	Beschreibung
`spill_bucket`	Erforderlich. Der Name des Amazon-S3-Buckets, in den die Lambda-Funktion Daten übertragen kann.

Parameter	Beschreibung
`spill_prefix`	Erforderlich. Das Präfix innerhalb des Überlauf-Buckets, in dem die Lambda-Funktion Daten ausgeben kann.
`spill_put_request_headers`	(Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Andere mögliche Header finden Sie unter PutObject in der API-Referenz zu Amazon Simple Storage Service.

Datentypunterstützung

Die folgende Tabelle zeigt die entsprechenden Datentypen, die für Kafka und Apache Arrow unterstützt werden.

Kafka	Arrow
CHAR	VARCHAR
VARCHAR	VARCHAR
TIMESTAMP	MILLISEKUNDE
DATUM	TAG
BOOLEAN	BOOL
SMALLINT	SMALLINT
INTEGER	INT
BIGINT	BIGINT
DECIMAL	FLOAT8
DOUBLE	FLOAT8

Partitionen und Splits

Kafka-Themen sind in Partitionen unterteilt. Jede Partition ist geordnet. Jede Nachricht in einer Partition hat eine inkrementelle ID, die Offset genannt wird. Jede Kafka-Partition wird für die parallele Verarbeitung in weitere Splits unterteilt. Daten sind für den in Kafka-Clustern konfigurierten Aufbewahrungszeitraum verfügbar.

Bewährte Methoden

Verwenden Sie als bewährte Methode bei der Abfrage von Athena das Prädikat-Pushdown, wie in den folgenden Beispielen.

```
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'
```

Einrichten des MSK-Konnektors

Bevor Sie den Konnektor verwenden können, müssen Sie Ihren Amazon-MSK-Cluster einrichten, die AWS Glue-Schema-Registry verwenden, um Ihr Schema zu definieren, und die Authentifizierung für den Connector konfigurieren.

> **Note**
>
> Wenn Sie den Konnektor in einer VPC bereitstellen, um auf private Ressourcen zuzugreifen und auch eine Verbindung zu einem öffentlich zugänglichen Service wie Confluent herstellen möchten, müssen Sie den Konnektor einem privaten Subnetz zuordnen, das über ein NAT-Gateway verfügt. Weitere Informationen finden Sie unter NAT-Gateways im Amazon-VPC-Benutzerhandbuch.

Beacjten Sie bei der Arbeit mit dem AWS Glue-Schema-Registry die folgenden Punkte:

- Stellen Sie sicher, dass der Text im Feld Description (Beschreibung) des AWS Glue-Schema-Registry die Zeichenfolge `{AthenaFederationMSK}` enthält. Diese Markierungszeichenfolge

ist für AWS Glue-Registrys erforderlich, die Sie mit dem MSK-Konnektor von Amazon Athena verwenden.

- Verwenden Sie für Ihre Datenbanknamen und -tabellen nur Kleinbuchstaben, um eine optimale Leistung zu erzielen. Bei der Verwendung gemischter Groß- und Kleinschreibung führt der Konnektor eine Suche ohne Berücksichtigung der Groß- und Kleinschreibung durch, die rechenintensiver ist.

So richten Sie Ihre Amazon-MSK-Umgebung und das AWS Glue-Schema-Registry ein

1. Richten Sie Ihre Amazon MSK-Umgebung ein. Informationen und Schritte finden Sie unter Einrichten von Amazon MSK und Erste Schritte mit Amazon MSK im Entwicklerhandbuch für Amazon Managed Streaming for Apache Kafka.
2. Laden Sie die Kafka-Themenbeschreibungsdatei (d. h. das Schema) im JSON-Format in das AWS Glue-Schema-Registry hoch. Weitere Informationen finden Sie unter Integration mit AWS Glue-Schema-Registry im AWS Glue-Entwicklerhandbuch. Beispielschemata finden Sie im folgenden Abschnitt.

Schemabeispiele für das AWS Glue-Schema-Registry

Verwenden Sie das Format der Beispiele in diesem Abschnitt, wenn Sie Ihr Schema in das AWS Glue-Schema Registry hochladen.

Beispielschema für einen JSON-Typ

Im folgenden Beispiel gibt das Schema, das in der AWS Glue-Schema-Registry erstellt werden soll, `json` als Wert für `dataFormat` an und verwendet `datatypejson` für `topicName`.

> **Note**
>
> Der Wert für `topicName` sollte die gleiche Schreibweise wie der Themenname in Kafka verwenden.

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

Beispielschema für einen CSV-Typ

Im folgenden Beispiel gibt das Schema, das in der AWS Glue-Schema-Registry erstellt werden soll, `csv` als Wert für `dataFormat` an und verwendet `datatypecsvbulk` für `topicName`. Der Wert für `topicName` sollte die gleiche Schreibweise wie der Themenname in Kafka verwenden.

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

Konfigurieren der Authentifizierung für den Athena-MSK-Konnektor

Sie können verschiedene Methoden verwenden, um sich bei Ihrem Amazon-MSK-Cluster zu authentifizieren, darunter IAM, SSL, SCRAM und eigenständiges Kafka.

Die folgende Tabelle zeigt die Authentifizierungstypen für den Konnektor sowie das jeweilige Sicherheitsprotokoll und den SASL-Mechanismus. Weitere Informationen finden Sie unter Authentifizierung und Autorisierung für Apache-Kafka-APIs im Entwicklerhandbuch für Amazon Managed Streaming for Apache Kafka.

auth_type	security.protocol	sasl.mechanik
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	–

> **Note**
>
> Die `SASL_SSL_PLAIN`- und `SASL_PLAINTEXT_PLAIN`-Authentifizierungstypen werden von Apache Kafka unterstützt, jedoch nicht von Amazon MSK.

SASL/IAM

Wenn der Cluster die IAM-Authentifizierung verwendet, müssen Sie bei der Einrichtung des Clusters die IAM-Richtlinie für den Benutzer konfigurieren. Weitere Informationen finden Sie unter IAM-Zugriffskontrolle im Entwicklerhandbuch für Amazon Managed Streaming for Apache Kafka.

Um diesen Authentifizierungstyp zu verwenden, legen Sie die `auth_type`-Lambda-Umgebungsvariable für den Konnektor auf `SASL_SSL_AWS_MSK_IAM` fest.

SSL

Wenn der Cluster SSL-authentifiziert ist, müssen Sie die Vertrauensspeicher- und Schlüsselspeicherdateien generieren und sie in den Amazon-S3-Bucket hochladen. Sie müssen diese Amazon-S3-Referenz angeben, wenn Sie den Konnektor bereitstellen. Der Schlüsselspeicher, der Vertrauensspeicher und der SSL-Schlüssel werden in der AWS Secrets Manager gespeichert. Sie geben den geheimen AWS-Schlüssel an, wenn Sie den Konnektor bereitstellen.

Informationen zum Erstellen eines Geheimnisses im Secrets Manager finden Sie unter Erstellen eines AWS Secrets Manager-Geheimnisses.

Um diesen Authentifizierungstyp zu verwenden, legen Sie die Umgebungsvariablen wie in der folgenden Tabelle gezeigt fest.

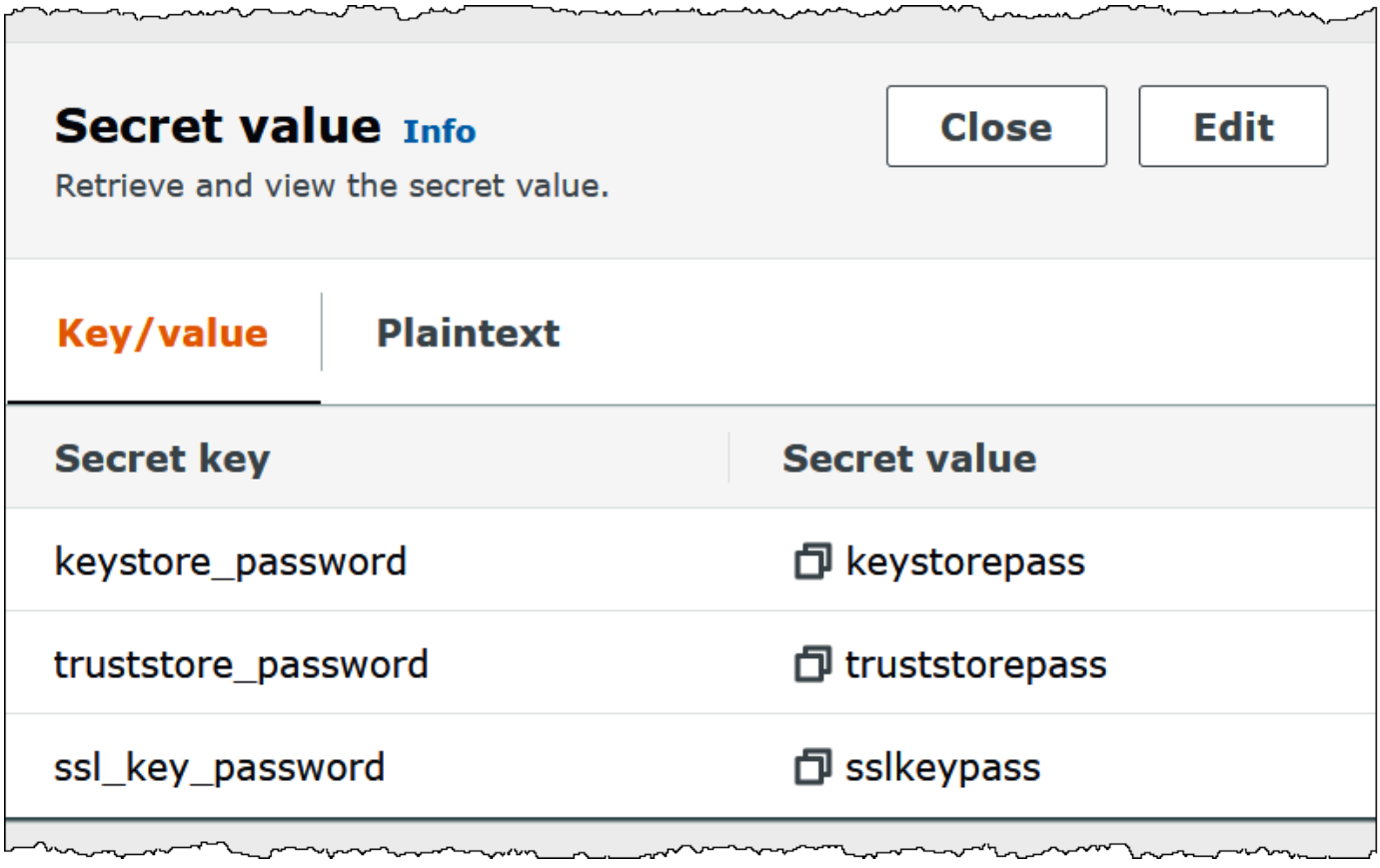
Parameter	Wert
`auth_type`	`SSL`
`certificates_s3_reference`	Der Amazon-S3-Speicherort, der die Zertifikate enthält.
`secrets_manager_secret`	Der Name Ihres geheimen AWS-Schlüssels.

Nachdem Sie ein Geheimnis in Secrets Manager erstellt haben, können Sie es in der Secrets-Manager-Konsole anzeigen.

So zeigen Sie Ihr Geheimnis in Secrets Manager an

1. Öffnen Sie die Secrets-Manager-Konsole unterhttps://console.aws.amazon.com/secretsmanager/.
2. Wählen Sie im Navigationsbereich Secrets (Geheimnisse).
3. Wählen Sie auf der Seite Secrets (Geheimnisse) den Link zu Ihrem Geheimnis aus.
4. Wählen Sie auf der Detailseite für Ihr Geheimnis die Option Retrieve secret value (Geheimniswert abrufen).

   Das folgende Image zeigt ein Beispiel für ein Geheimnis mit drei Schlüssel/Wert-Paaren: `keystore_password`, `truststore_password` und `ssl_key_password`.



SASL/SCRAM

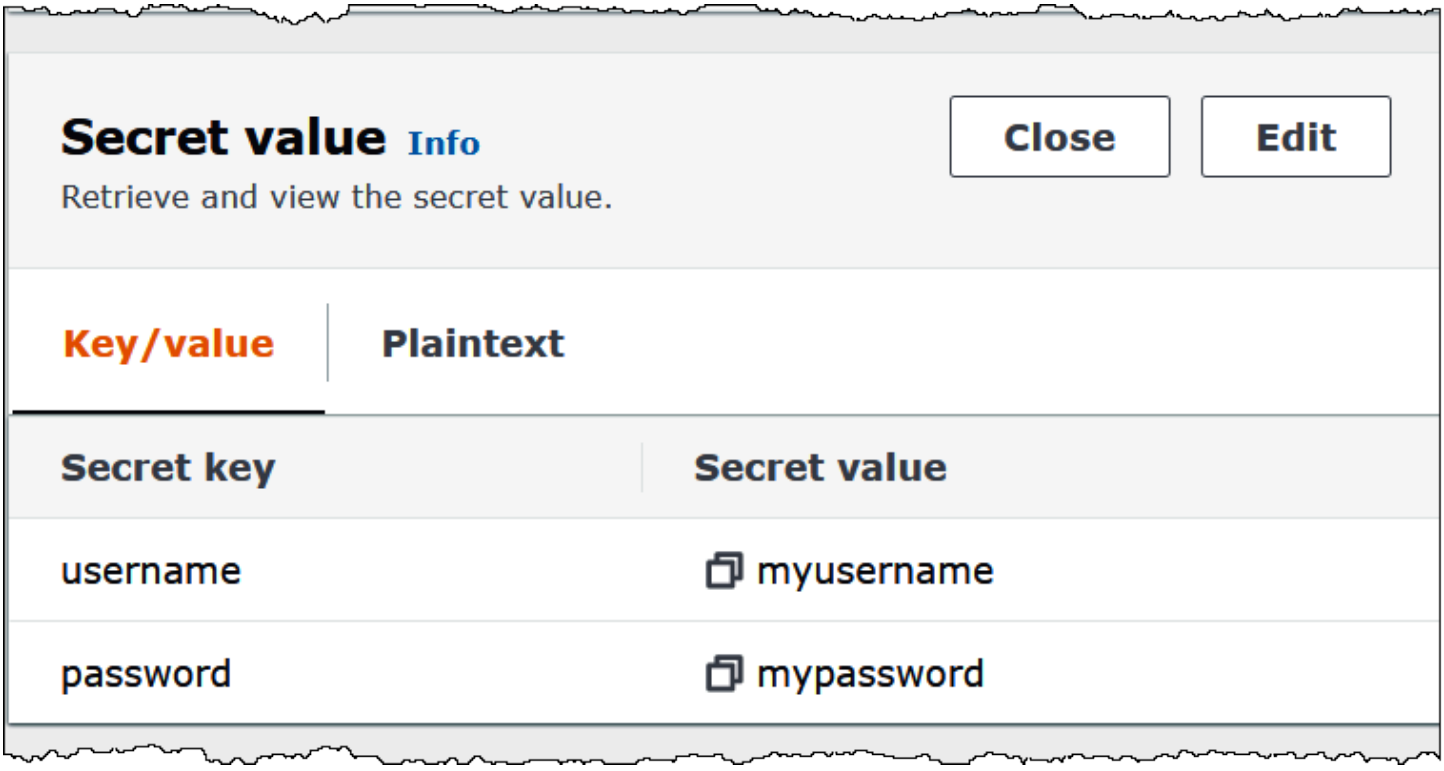
Wenn Ihr Cluster SCRAM-Authentifizierung verwendet, geben Sie bei der Bereitstellung des Konnektor den Secrets-Manager-Schlüssel an, der dem Cluster zugeordnet ist. Die AWS-

Anmeldeinformationen des Benutzers (geheimer Schlüssel und Zugriffsschlüssel) werden für die Authentifizierung mit dem Cluster verwendet.

Legen Sie die Umgebungsvariablen wie in der folgenden Tabelle angegeben fest.

Parameter	Wert
`auth_type`	`SASL_SSL_SCRAM_SHA512`
`secrets_manager_secret`	Der Name Ihres geheimen AWS-Schlüssels.

Das folgende Image zeigt ein Beispiel für ein Geheimnis in der Secrets-Manager-Konsole mit zwei Schlüssel/Wert-Paaren: eines für `username` und eines für `password`.



Lizenzinformationen

Durch die Verwendung dieses Konnektors erkennen Sie die Aufnahme von Komponenten von Drittanbietern an. Eine Liste dieser Komponenten finden Sie in der pom.xml-Datei für diesen Konnektor. Zudem stimmen Sie den Bedingungen der jeweiligen Drittanbieterlizenzen zu, die in der LICENSE.txt-Datei auf GitHub.com aufgeführt werden.

Weitere Informationen finden Sie auch unter

Weitere Informationen zu diesem Konnektor finden Sie unter der entsprechenden Seite auf GitHub.com.

Amazon Athena MySQL Konnektor

Der Amazon Athena Lambda-MySQL-Konnektor ermöglicht Amazon Athena den Zugriff auf MySQL-Datenbanken.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.
- Richten Sie eine VPC und eine Sicherheitsgruppe ein, bevor Sie diesen Konnektor verwenden. Weitere Informationen finden Sie unter Erstellen einer VPC für einen Datenquellen-Connector.

Einschränkungen

- Schreiboperationen wie DDL werden nicht unterstützt.
- In einem Multiplexer-Setup werden der Überlauf-Bucket und das Präfix von allen Datenbank-Instances gemeinsam genutzt.
- Alle relevanten Lambda-Grenzwerte. Weitere Informationen finden Sie unter Lambda quotas (Lambda-Kontingente) im AWS Lambda-Entwicklerhandbuch.
- Da Athena Abfragen in Kleinbuchstaben umwandelt, müssen MySQL-Tabellennamen in Kleinbuchstaben geschrieben werden. Zum Beispiel werden Athena-Abfragen für eine Tabelle mit dem Namen `myTable` scheitern.

Bedingungen

Die folgenden Begriffe beziehen sich auf den MySQL-Konnektor.

- Datenbank-Instance – Jede Instance einer Datenbank, die On-Premises, in Amazon EC2 oder auf Amazon RDS bereitgestellt wird.
- Handler – Ein Lambda-Handler, der auf Ihre Datenbank-Instance zugreift. Ein Handler kann für Metadaten oder für Datensätze verwendet werden.

- Metadaten-Handler – Ein Lambda-Handler, der Metadaten von Ihrer Datenbank-Instance abruft.
- Record Handler – Ein Lambda-Handler, der Datensätze aus Ihrer Datenbank-Instance abruft.
- Composite Handler – Ein Lambda-Handler, der sowohl Metadaten als auch Datensätze aus Ihrer Datenbank-Instance abruft.
- Eigenschaft oder Parameter – Eine Datenbankeigenschaft, die von Handlern zum Extrahieren von Datenbankinformationen verwendet wird. Sie konfigurieren diese Eigenschaften als Lambda-Umgebungsvariablen.
- Verbindungszeichenfolge – Eine Textzeichenfolge, die verwendet wird, um eine Verbindung zu einer Datenbank-Instance herzustellen.
- Katalog – Ein Nicht-AWS Glue-Katalog, der bei Athena registriert ist und ein erforderliches Präfix für die `connection_string`-Eigenschaft darstellt.
- Multiplex-Handler – Ein Lambda-Handler, der mehrere Datenbankverbindungen akzeptieren und verwenden kann.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den MySQL-Konnektor zu konfigurieren.

Verbindungszeichenfolge

Verwenden Sie eine JDBC-Verbindungszeichenfolge im folgenden Format, um eine Verbindung zu einer Datenbank-Instance herzustellen.

```
mysql://${jdbc_connection_string}
```

> **Note**
>
> Wenn Sie den Fehler java.sql.SqlException: Zero date value prohibited bei einer `SELECT`-Abfrage zu einer MySQL-Tabelle erhalten, fügen Sie der Verbindungszeichenfolge den folgenden Parameter hinzu:
>
> ```
> zeroDateTimeBehavior=convertToNull
> ```
>
> Weitere Informationen finden Sie unter Error „Zero date value prohibited“ beim Versuch, aus der MySQL-Tabelle auszuwählen auf GitHub.com.

Verwenden eines Multiplex-Handlers

Sie können einen Multiplexer verwenden, um mit einer einzigen Lambda-Funktion eine Verbindung zu mehreren Datenbank-Instances herzustellen. Anfragen werden anhand des Katalognamens weitergeleitet. Verwenden Sie die folgenden Klassen in Lambda.

Handler	Klasse
Composite Handler	`MySqlMuxCompositeHandler`
Metadaten-Handler	`MySqlMuxMetadataHandler`
Record Handler	`MySqlMuxRecordHandler`

Multiplex-Handler-Parameter

Parameter	Beschreibung
$*catalog*_connection_string	Erforderlich. Eine Verbindungszeichenfolge einer Datenbank-Instance. Stellen Sie der Umgebungsvariablen den Namen des in Athena verwendeten Katalogs voran. Wenn zum Beispiel der bei Athena registrierte Katalog `mymysqlcatalog` ist, dann lautet der Name der Umgebungsvariablen `mymysqlcatalog_connection_string` .
`default`	Erforderlich. Die standardmäßige Verbindungszeichenfolge. Diese Zeichenfolge wird verwendet, wenn der Katalog `lambda:${`*AWS_LAMBDA_FUNCTION_NAME*`}` ist.

Die folgenden Beispieleigenschaften gelten für eine MySql MUX Lambda-Funktion, die zwei Datenbankinstanzen unterstützt: `mysql1` (die Standardeinstellung) und `mysql2`.

Property (Eigenschaft)	Wert
`default`	`mysql://jdbc:mysql://mysql2.host:3333/default?` benutzer=Beispiel2&password=Beispiel2

Property (Eigenschaft)	Wert
`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?` benutzer=Beispiel2&password=Beispiel2

Bereitstellen von Anmeldeinformationen

Um einen Benutzernamen und ein Kennwort für Ihre Datenbank in Ihrer JDBC-Verbindungszeichenfolge anzugeben, können Sie Eigenschaften von Verbindungszeichenfolgen oder AWS Secrets Manager verwenden.

- Verbindungszeichenfolge – Ein Benutzername und ein Kennwort können als Eigenschaften in der JDBC-Verbindungszeichenfolge angegeben werden.

  > **Important**
  >
  > Als bewährte Sicherheitsmethode sollten Sie keine fest kodierten Anmeldeinformationen in Ihren Umgebungsvariablen oder Verbindungszeichenfolgen verwenden. Informationen zum Verschieben von hartkodierten Geheimnissen nach AWS Secrets Manager finden Sie unter Verschieben von hartkodierten Geheimnissen nach AWS Secrets Manager im AWS Secrets Manager-Benutzerhandbuch.

- AWS Secrets Manager – Um das Athena-Federated-Query-Feature mit AWS Secrets Manager zu verwenden, sollte die mit Ihrer Lambda-Funktion verbundene VPC über einen Internetzugang oder einen VPC-Endpunkt verfügen, um eine Verbindung zu Secrets Manager herzustellen.

  Sie können den Namen eines Secrets in AWS Secrets Manager in Ihrer JDBC-Verbindungszeichenfolge eingeben. Der Konnektor ersetzt den geheimen Namen durch `username`- und `password`-Werte von Secrets Manager.

  Für Amazon RDS-Datenbank-Instances ist diese Unterstützung eng integriert. Wenn Sie Amazon RDS verwenden, empfehlen wir dringend die Verwendung von AWS Secrets Manager und

Wechsel der Anmeldeinformationen. Wenn Ihre Datenbank Amazon RDS nicht verwendet, speichern Sie die Anmeldeinformationen als JSON im folgenden Format:

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

Beispiel einer Verbindungszeichenfolge mit einem geheimen Namen

Die folgende Zeichenfolge hat den geheimen Namen `${Test/RDS/MySql1}`.

```
mysql://jdbc:mysql://mysql1.host:3306/default?...&${Test/RDS/MySql1}&...
```

Der Konnektor verwendet den geheimen Namen, um Secrets abzurufen und den Benutzernamen und das Kennwort bereitzustellen, wie im folgenden Beispiel gezeigt.

```
mysql://jdbc:mysql://mysql1host:3306/default?...&user=sample2&password=sample2&...
```

Derzeit erkennt der MySQL-Konnektor die `user`- und `password`-JDBC-Eigenschaften.

Verwenden eines einzelnen Verbindungs-Handlers

Sie können die folgenden Einzelverbindungsmetadaten und Record Handler verwenden, um eine Verbindung zu einer einzelnen MySQL-Instance herzustellen.

Handler-Typ	Klasse
Composite Handler	`MySqlCompositeHandler`
Metadaten-Handler	`MySqlMetadataHandler`
Record Handler	`MySqlRecordHandler`

Parameter für Einzelverbindungs-Handler

Parameter	Beschreibung
`default`	Erforderlich. Die standardmäßige Verbindungszeichenfolge.

Die Einzelverbindungs-Handler unterstützen eine Datenbank-Instance und müssen einen `default`-Verbindungszeichenfolgenparameter bereitstellen. Alle anderen Verbindungszeichenfolgen werden ignoriert.

Die folgende Beispieleigenschaft gilt für eine einzelne MySQL-Instance, die von einer Lambda-Funktion unterstützt wird.

Property (Eigenschaft)	Wert
`default`	`mysql://mysql1.host:3306/default?secret=Test/RDS/` MySql1

Überlauf-Parameter

Das Lambda-SDK kann Daten an Amazon S3 übertragen. Alle Datenbank-Instances, auf die mit derselben Lambda-Funktion zugegriffen wird, werden an denselben Speicherort verschoben.

Parameter	Beschreibung
`spill_bucket`	Erforderlich. Überlauf-Bucket-Name.
`spill_prefix`	Erforderlich. Schlüssel-Prefix für den Überlauf-Bucket.
`spill_put_request_headers`	(Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Andere mögliche Header finden Sie unter PutObject in der API-Referenz zu Amazon Simple Storage Service.

Datentypunterstützung

Die folgende Tabelle zeigt die entsprechenden Datentypen für JDBC und Arrow.

JDBC	Arrow
Boolesch	Bit
Ganzzahl	Tiny
Short	Smallint
Ganzzahl	Int
Long	Bigint
float	Float4
Double	Float8
Datum	Dateday
Zeitstempel	DateMilli
Zeichenfolge	Varchar
Bytes	Varbinary
BigDecimal	Dezimal
ARRAY	Auflisten

Partitionen und Splits

Partitionen werden verwendet, um zu bestimmen, wie Splits für den Konnektor generiert werden. Athena konstruiert eine synthetische Säule vom Typ `varchar`, die das Partitionierungsschema für die Tabelle darstellt, das dem Konnektor beim Generieren von Splits hilft. Der Konnektor ändert nicht die eigentliche Tabellendefinition.

Leistung

MySQL unterstützt native Partitionen. Der Athena-MySQL-Konnektor kann Daten von diesen Partitionen parallel abrufen. Wenn Sie sehr große Datenmengen mit einheitlicher Partitionsverteilung abfragen möchten, wird eine native Partitionierung dringend empfohlen.

Der Athena-MySQL-Konnektor führt einen Prädikat-Pushdown durch, um die Anzahl der von der Abfrage gescannten Daten zu reduzieren. `LIMIT`-Klauseln, einfache Prädikate und komplexe Ausdrücke werden an den Konnektor übertragen, um die Menge der gescannten Daten zu reduzieren und die Laufzeit der Abfrage zu verkürzen.

LIMIT-Klauseln

Eine `LIMIT N`-Anweisung reduziert die von der Abfrage durchsuchten Daten. Mit `LIMIT N`-Pushdown gibt der Konnektor nur `N` Zeilen an Athena zurück.

Prädikate

Ein Prädikat ist ein Ausdruck in der `WHERE`-Klausel einer SQL-Abfrage, der einen booleschen Wert ergibt und Zeilen auf der Grundlage mehrerer Bedingungen filtert. Der Athena-MySQL-Konnektor kann diese Ausdrücke kombinieren und sie direkt an MySQL weiterleiten, um die Funktionalität zu erweitern und die Menge der gescannten Daten zu reduzieren.

Die folgenden Athena-MySQL-Konnektor-Operatoren unterstützen Prädikat-Pushdown:

- Boolean: UND, ODER, NICHT
- Gleichheit: GLEICH, NICHT-GLEICH, WENIGER_ALS, WENIGER_ODER-GLEICH, GRÖSSER_ALS, GRÖSSER_ODER-GLEICH, IST_UNTERSCHIEDEN VON, NULL_WENN, IST_NULL
- Arithmetik: ADDIEREN, SUBTRAHIEREN, MULTIPLIZIEREN, DIVIDIEREN, MODULIEREN, NEGIEREN
- Andere: WIE_MUSTER, IN

Beispiel für einen kombinierten Pushdown

Kombinieren Sie für erweiterte Abfragefunktionen die Pushdown-Typen wie im folgenden Beispiel:

```
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;
```

Einen Artikel über die Verwendung von Prädikat-Pushdown zur Verbesserung der Leistung bei Verbundabfragen, einschließlich MySQL, finden Sie unter Verbessern von Verbundabfragen mit Prädikat-Pushdown in Amazon Athena im AWS-Big-Data-Blog.

Lizenzinformationen

Durch die Verwendung dieses Konnektors erkennen Sie die Aufnahme von Komponenten von Drittanbietern an. Eine Liste dieser Komponenten finden Sie in der pom.xml-Datei für diesen Konnektor. Zudem stimmen Sie den Bedingungen der jeweiligen Drittanbieterlizenzen zu, die in der LICENSE.txt-Datei auf GitHub.com aufgeführt werden.

Weitere Informationen finden Sie auch unter

Die neuesten Informationen zur JDBC-Treiberversion finden Sie in der pom.xml-Datei für den MySQL-Konnektor auf GitHub.com.

Weitere Informationen zu diesem Konnektor finden Sie unter der entsprechenden Seite auf GitHub.com.

Amazon Athena Neptune Konnektor

Amazon Neptune ist ein schneller, zuverlässiger, vollständig verwalteter Graph-Datenbankservice, mit dem es ganz einfach ist, Anwendungen zu erstellen und auszuführen, die mit stark verbundenen Datensätzen arbeiten. Die speziell entwickelte, leistungsstarke Diagrammdatenbank-Engine von Neptune speichert Milliarden von Beziehungen optimal und fragt Diagramme mit einer Latenz von nur Millisekunden ab. Weitere Informationen finden Sie unter Neptune-Benutzerhandbuch.

Der Amazon Athena Neptune-Konnektor ermöglicht Athena die Kommunikation mit Ihrer Neptune-Graphdatenbank-Instance, sodass Ihre Neptun-Diagrammdaten über SQL-Abfragen zugänglich sind.

Wenn Sie Lake Formation in Ihrem Konto aktiviert haben, AWS Serverless Application Repository muss die IAM-Rolle für Ihren Athena-Verbund-Lambda-Konnektor, den Sie in bereitgestellt haben, Lesezugriff in Lake Formation auf die haben AWS Glue Data Catalog.

Voraussetzungen

Die Verwendung des Neptune-Konnektors erfordert die folgenden drei Schritte.

- Einen Neptun-Cluster einrichten
- Einrichten eines AWS Glue Data Catalog
- Bereitstellen des Konnektors für Ihren AWS-Konto. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit

AWS Serverless Application Repository. Weitere Informationen zur Bereitstellung des Neptune-Konnektors finden Sie unter Bereitstellen des Amazon Athena-Neptune-Konnektors auf GitHub.com.

Einschränkungen

Derzeit weist der Neptune-Konnektor die folgenden Einschränkungen auf.

- Nur das Eigenschaftsdiagrammmodell wird unterstützt.
- Das Projizieren von Spalten, einschließlich des Primärschlüssels (ID), ist nicht unterstützt.

Einen Neptun-Cluster einrichten

Wenn Sie keinen vorhandenen Amazon Neptune-Cluster- und Eigenschaftsgraph-Datensatz haben, den Sie verwenden möchten, müssen Sie einen einrichten.

Stellen Sie sicher, dass Sie über einen Internet-Gateway und einen NAT-Gateway in der VPC verfügen, die Ihren Neptune-Cluster hostet. Die privaten Subnetze, die die Lambda-Funktion des Neptune-Konnektors verwendet, sollten über diesen NAT-Gateway eine Verbindung zum Internet haben. Die Lambda-Funktion des Neptune-Konnektors verwendet das NAT-Gateway, um mit zu kommunizieren AWS Glue.

Anweisungen zum Einrichten eines neuen Neptune-Clusters und zum Laden dieses Clusters mit einem Beispieldatensatz finden Sie unter Beispiel für die Einrichtung eines Neptune-Clusters auf GitHub.com.

Einrichten eines AWS Glue Data Catalog

Im Gegensatz zu herkömmlichen relationalen Datenspeichern verwenden Neptune Graph DB-Knoten und -Edges kein festgelegtes Schema. Jeder Eintrag kann unterschiedliche Felder und Datentypen haben. Da der Neptune-Konnektor jedoch Metadaten aus dem abruft AWS Glue Data Catalog, müssen Sie eine - AWS Glue Datenbank erstellen, die Tabellen mit dem erforderlichen Schema enthält. Nachdem Sie die AWS Glue -Datenbank und -Tabellen erstellt haben, kann der Konnektor die Liste der Tabellen füllen, die für die Abfrage von Athena verfügbar sind.

Aktivieren des Spaltenabgleichs ohne Berücksichtigung der Groß-/Kleinschreibung

Um Spaltennamen aus Ihrer Neptune-Tabelle mit der richtigen Groß- und Kleinschreibung aufzulösen, auch wenn die Spaltennamen in alle Kleinbuchstaben enthalten AWS Glue, können Sie

den Neptune-Konnektor für den Abgleich ohne Berücksichtigung der Groß- und Kleinschreibung konfigurieren.

Um dieses Feature zu aktivieren, setzen Sie die Umgebungsvariable `enable_caseinsensitivematch` in der Lambda-Funktion des Neptune-Konnektors auf `true`.

Angeben des AWS Glue glabel-Tabellenparameters für Tabellennamen mit Groß- und Kleinschreibung

Da nur Tabellennamen in Kleinbuchstaben AWS Glue unterstützt, ist es wichtig, den `glabel` AWS Glue Tabellenparameter anzugeben, wenn Sie eine - AWS Glue Tabelle für Neptune erstellen und Ihr Neptune-Tabellenname Groß- und Kleinschreibung enthält.

Fügen Sie in Ihre AWS Glue Tabellendefinition den `glabel` Parameter ein und legen Sie dessen Wert auf Ihren Tabellennamen mit der ursprünglichen Groß-/Kleinschreibung fest. Dadurch wird sichergestellt, dass die richtige Groß- und Kleinschreibung erhalten bleibt, wenn mit Ihrer Neptune-Tabelle AWS Glue interagiert. Im folgenden Beispiel wird der Wert von `glabel` auf den Tabellennamen `Airport` gesetzt.

```
glabel = Airport
```

Table properties (3)

Key	Value
separatorChar	,
componenttype	vertex
glabel	Airport

Weitere Informationen zum Einrichten eines AWS Glue Data Catalog für die Arbeit mit Neptune finden Sie unter Einrichten von - AWS Glue Katalog auf GitHub.com.

Leistung

Der Athena-Neptune-Konnektor führt einen Prädikat-Pushdown durch, um die von der Abfrage durchsuchten Daten zu reduzieren. Prädikate, die den Primärschlüssel verwenden, führen jedoch zu einem Abfragefehler. `LIMIT`-Klauseln reduzieren die Menge der gescannten Daten, aber wenn Sie kein Prädikat angeben, sollten Sie davon ausgehen, dass `SELECT`-Abfragen mit einer `LIMIT`-Klausel mindestens 16 MB Daten scannen. Der Neptune-Konnektor ist aufgrund der Gleichzeitigkeit widerstandsfähig gegenüber Drosselung.

Weitere Informationen finden Sie auch unter

Weitere Informationen zu diesem Konnektor finden Sie auf der entsprechenden Website unter GitHub.com.

Amazon Athena OpenSearch Konnektor

OpenSearch Service

Der Amazon-Athena-OpenSearch-Konnektor ermöglicht Amazon Athena die Kommunikation mit Ihren OpenSearch-Instances, sodass Sie Ihre OpenSearch-Daten mithilfe von SQL abfragen können.

> **Note**
>
> Aufgrund eines bekannten Problems kann der OpenSearch-Konnektor nicht mit einer VPC verwendet werden.

Wenn Sie Lake Formation in Ihrem Konto aktiviert haben, muss die IAM-Rolle für Ihren Athena-Verbund-Lambda-Konnektor, den Sie im AWS Serverless Application Repository bereitstellen, in Lake Formation über Lesezugriff auf das AWS Glue Data Catalog verfügen.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Bedingungen

Die folgenden Begriffe beziehen sich auf den OpenSearch-Konnektor.

- Domain – Ein Name, den dieser Konnektor mit dem Endpunkt Ihrer OpenSearch-Instance verknüpft. Die Domain wird auch als Datenbankname verwendet. Für OpenSearch-Instances, die im Amazon OpenSearch Service definiert sind, ist die Domain automatisch auffindbar. Für andere Instances müssen Sie eine Zuordnung zwischen dem Domainnamen und dem Endpunkt bereitstellen.
- Index – Eine Datenbanktabelle, die in Ihrer OpenSearch-Instance definiert ist.

- Mapping – Wenn ein Index eine Datenbanktabelle ist, dann ist das Mapping sein Schema (d. h. die Definitionen seiner Felder und Attribute).

  Dieser Konnektor unterstützt sowohl das Abrufen von Metadaten aus der OpenSearch-Instance als auch vonAWS Glue Data Catalog. Wenn der Konnektor eine AWS Glue-Datenbank und -Tabelle findet, die Ihren OpenSearch-Domain- und Indexnamen entsprechen, versucht der Konnektor, sie für die Schemadefinition zu verwenden. Wir empfehlen Ihnen, Ihre AWS Glue-Tabelle so zu erstellen, dass sie eine Übergruppe aller Felder in Ihrem OpenSearch-Index ist.
- Dokument – Ein Datensatz innerhalb einer Datenbanktabelle.
- Datenstrom – Zeitbasierte Daten, die aus mehreren Hintergrundindizes bestehen. Weitere Informationen finden Sie unter Datenströme in der OpenSearch-Dokumentation und Erste Schritte mit Datenströmen im Entwicklerhandbuch für Amazon OpenSearch Service.

> **Note**
>
> Da Datenstrom-Indizes intern von Open Search erstellt und verwaltet werden, wählt der Konnektor die Schemazuordnung aus dem ersten verfügbaren Index aus. Aus diesem Grund empfehlen wir dringend, eine AWS Glue-Tabelle als zusätzliche Metadatenquelle einzurichten. Weitere Informationen finden Sie unter Einrichten von Datenbanken und Tabellen in AWS Glue.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den OpenSearch-Konnektor zu konfigurieren.

- spill_bucket – Gibt den Amazon S3-Bucket für Daten an, die die Lambda-Funktionsgrenzen überschreiten.
- spill_prefix – (Optional) Ist standardmäßig ein Unterordner im angegebenen `spill_bucket` genannt `athena-federation-spill`. Wir empfehlen Ihnen, einen Amazon-S3-Speicher-Lebenszyklus an dieser Stelle zu konfigurieren, um die Überlaufe zu löschen, die älter als eine festgelegte Anzahl von Tagen oder Stunden sind.
- spill_put_request_headers – (Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Andere mögliche Header finden Sie unter PutObject in der API-Referenz zu Amazon Simple Storage Service.

- kms_key_id – (Optional) Standardmäßig werden alle Daten, die an Amazon S3 gesendet werden, mit dem AES-GCM-authentifizierten Verschlüsselungsmodus und einem zufällig generierten Schlüssel verschlüsselt. Damit Ihre Lambda-Funktion stärkere Verschlüsselungsschlüssel verwendet, die von KMS generiert werden, wie`a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, können Sie eine ID einer Verschlüsselung angeben.
- disable_spill_encryption – (Optional) Bei Einstellung auf `True`, wird die Spill-Verschlüsselung deaktiviert. Die Standardeinstellung ist `False`, sodass Daten, die an S3 übertrragen werden, mit AES-GCM verschlüsselt werden - entweder mit einem zufällig generierten Schlüssel oder mit KMS zum Generieren von Schlüsseln. Das Deaktivieren der Überlauf-Verschlüsselung kann die Leistung verbessern, insbesondere wenn Ihr Überlauf-Standort eine serverseitige Verschlüsselung verwendet.
- disable_glue – (Optional) Falls vorhanden und auf „true“ (wahr) gesetzt, versucht der Konnektor nicht, zusätzliche Metadaten aus AWS Glue abzurufen.
- query_timeout_cluster – Der Timeout-Zeitraum in Sekunden für Cluster-Integritätsabfragen, die bei der Generierung parallel Scans verwendet werden.
- query_timeout_search – Die Zeitüberschreitungsdauer in Sekunden für Suchanfragen, die beim Abrufen von Dokumenten aus einem Index verwendet werden.
- auto_discover_endpoint – Boolesch. Der Standardwert ist `true`. Wenn Sie den Amazon OpenSearch Service verwenden und diesen Parameter auf „true“ (wahr) setzen, kann der Konnektor Ihre Domains und Endpunkte automatisch ermitteln, indem er die entsprechenden Beschreibungs- oder Listen-API-Vorgänge im OpenSearch-Dienst aufruft. Für jede andere Art von OpenSearch-Instance (z. B. selbst gehostete) müssen Sie die zugehörigen Domainendpunkte in der `domain_mapping`-Variable angeben. Bei `auto_discover_endpoint=true` verwendet der Konnektor AWS-Anmeldeinformationen für die Authentifizierung beim OpenSearch Service. Andernfalls ruft der Konnektor Benutzername und Passwort aus AWS Secrets Manager durch die `domain_mapping`-Variable ab.
- domain_mapping – Wird nur verwendet, wenn `auto_discover_endpoint` auf „false“ (falsch) gesetzt ist und die Zuordnung zwischen Domainnamen und den zugehörigen Endpunkten definiert. Die `domain_mapping`-Variable kann mehrere OpenSearch-Endpunkte in folgendem Format aufnehmen:

  ```
  domain1=endpoint1,domain2=endpoint2,domain3=endpoint3,...
  ```

  Für die Authentifizierung an einem OpenSearch-Endpunkt unterstützt der Konnektor Ersetzungszeichenfolgen, die mit dem Format `${SecretName}:` injiziert werden, wobei der

Benutzername und Passwort von AWS Secrets Manager abgerufen werden. Der Doppelpunkt (:) am Ende des Ausdrucks dient als Trennzeichen vom Rest des Endpunkts.

> **Important**
>
> Als bewährte Sicherheitsmethode sollten Sie keine fest kodierten Anmeldeinformationen in Ihren Umgebungsvariablen oder Verbindungszeichenfolgen verwenden. Informationen zum Verschieben von hartkodierten Geheimnissen nach AWS Secrets Manager finden Sie unter Verschieben von hartkodierten Geheimnissen nach AWS Secrets Manager im AWS Secrets Manager-Benutzerhandbuch.

Im folgenden Beispiel wird das `opensearch-creds`-Secret verwendet.

```
movies=https://${opensearch-creds}:search-movies-ne...qu.us-east-1.es.amazonaws.com
```

Bei der Ausführung wird `${opensearch-creds}` wie im folgenden Beispiel als Benutzername und -Passwort wiedergegeben.

```
movies=https://myusername@mypassword:search-movies-ne...qu.us-east-1.es.amazonaws.com
```

Im `domain_mapping`-Parameter kann jedes Domain-Endpunkt-Paar ein anderes Secret verwenden. Das Secret selbst muss im Format *`user_name`*@*`password`* angegeben werden. Obwohl das Passwort eingebettete @-Zeichen enthalten kann, dient das erste@ als Trennzeichen von *`user_name`*.

Es ist auch wichtig zu beachten, dass das Komma (,) und das Gleichheitszeichen (=) von diesem Konnektor als Trennzeichen für die Domain-Endpunkt-Paare verwendet werden. Aus diesem Grund sollten Sie sie nicht innerhalb des gespeicherten Secrets verwenden.

Einrichten von Datenbanken und Tabellen in AWS Glue

Der Konnektor erhält Metadateninformationen mithilfe von AWS Glue oder OpenSearch. Sie können eine AWS Glue-Tabelle als ergänzende Metadatendefinitionsquelle einrichten. Um diese Funktion zu aktivieren, definieren Sie eine AWS Glue-Datenbank und -Tabelle, die mit der Domain und dem Index der Quelle übereinstimmen, die Sie ergänzen. Der Konnektor kann auch die in der OpenSearch-

Instance gespeicherten Metadatendefinitionen nutzen, indem er das Mapping für den angegebenen Index abruft.

Definieren von Metadaten für Arrays in OpenSearch

OpenSearch hat keinen dedizierten Array-Datentyp. Jedes Feld kann null oder mehr Werte enthalten, sofern sie vom gleichen Datentyp sind. Wenn Sie OpenSearch als Ihre Metadatendefinitionsquelle verwenden möchten, müssen Sie eine `_meta`-Eigenschaft für alle mit Athena verwendeten Indizes für die Felder definieren, die als Liste oder Array betrachtet werden sollen. Wenn Sie diesen Schritt nicht ausführen, geben Abfragen nur das erste Element im Listenfeld aus. Wenn Sie die `_meta`-Eigenschaft angeben, sollten Feldnamen sollten vollständig für verschachtelte JSON-Strukturen qualifiziert sein (z. B. `address.street`, wobei `street` ein verschachteltes Feld innerhalb einer `address`-Struktur ist).

Im folgenden Beispiel werden `actor`- und `genre`-Listen in der `movies`-Tabelle definiert.

```
PUT movies/_mapping
{
  "_meta": {
    "actor": "list",
    "genre": "list"
  }
}
```

Datentypen

Der OpenSearch-Konnektor kann Metadatendefinitionen entweder aus AWS Glue oder der OpenSearch-Instance extrahieren. Der Konnektor verwendet das Mapping in der folgenden Tabelle, um die Definitionen in Apache Arrow-Datentypen zu konvertieren, einschließlich der im folgenden Abschnitt aufgeführten Punkte.

OpenSearch	Apache Arrow	AWS Glue
Text, Schlüsselwort, binär	VARCHAR	string
long	BIGINT	bigint
scaled_float	BIGINT	SCALED_FLOAT (...)
Ganzzahl	INT	int

OpenSearch	Apache Arrow	AWS Glue
short	SMALLINT	smallint
Byte	TINYINT	tinyint
double	FLOAT8	double
float, half_float	FLOAT4	float
Boolean	BIT	boolesch
Datum, date_nanos	DATUMMILLI	timestamp
JSON-Struktur	STRUCT	STRUCT
_meta (Weitere Informati onen finden Sie im Abschnitt Definieren von Metadaten für Arrays in OpenSearch.)	LIST	ARRAY

Hinweise zu Datentypen

- Derzeit unterstützt der Konnektor nur OpenSearch und AWS Glue-Datentypen, die in der obigen Tabelle aufgeführt sind.
- Ein `scaled_float` ist eine Gleitkommazahl, die mit einem festen doppelten Skalierungsfaktor skaliert und als `BIGINT` in Apache Arrow dargestellt wird. Beispielsweise wird 0,756 mit einem Skalierungsfaktor von 100 auf 76 gerundet.
- Zum Definieren eines `scaled_float` in AWS Glue, müssen Sie den `array`-Spaltentyp auswühlen und das Feld mit dem Format SCALED_FLOAT (*`scaling_factor`*) deklarieren.

  Das folgende Beispiele sind gültig:

```
SCALED_FLOAT(10.51)
SCALED_FLOAT(100)
SCALED_FLOAT(100.0)
```

  Das folgende Beispiele sind nicht gültig:

```
SCALED_FLOAT(10.)
SCALED_FLOAT(.5)
```

- Bei der Konvertierung von `date_nanos` zu `DATEMILLI` werden Nanosekunden auf die nächste Millisekunde gerundet. Zulässige Werte für `date` und `date_nanos` beinhalten, sind aber nicht beschränkt auf, die folgenden Formate:

```
"2020-05-18T10:15:30.123456789"
"2020-05-15T06:50:01.123Z"
"2020-05-15T06:49:30.123-05:00"
1589525370001 (epoch milliseconds)
```

- Eine OpenSearch `binary` ist eine Zeichenfolgendarstellung eines Binärwerts, der mit `Base64` kodiert und in ein `VARCHAR` umgewandelt wird.

Ausführen von SQL-Abfragen

Im Folgenden finden Sie Beispiele für DDL-Abfragen, die Sie mit diesem Konnektor verwenden können. In den Beispielen entspricht *`function_name`* dem Namen Ihrer Lambda-Funktion, *`domain`* ist der Name der Domain, die Sie abfragen möchten, und *`index`* ist der Name Ihres Index.

```
SHOW DATABASES in `lambda:function_name`
```

```
SHOW TABLES in `lambda:function_name`.domain
```

```
DESCRIBE `lambda:function_name`.domain.index
```

Leistung

Der Athena-OpenSearch-Konnektor unterstützt Shard-basierte parallele Scans. Der Konnektor verwendet aus der OpenSearch-Instance abgerufene Clusterintegritätsinformationen, um mehrere Anforderungen für eine Dokumentsuchabfrage zu generieren Die Anforderungen werden für jeden Shard aufgeteilt und gleichzeitig ausgeführt.

Der Konnektor gibt auch Prädikate als Teil seiner Dokumentensuchabfragen aus. Die folgende Beispielabfrage und das folgende Prädikat zeigen, wie der Konnektor Prädikat-Pushdown verwendet.

Abfragen

```
SELECT * FROM "lambda:elasticsearch".movies.movies
WHERE year >= 1955 AND year <= 1962 OR year = 1996
```

Prädikat

```
(_exists_:year) AND year:([1955 TO 1962] OR 1996)
```

Weitere Informationen finden Sie auch unter

- Einen Artikel über die Verwendung des OpenSearch-Konnektors von Amazon Athena zur Abfrage von Daten in Amazon OpenSearch Service und Amazon S3 in einer einzigen Abfrage finden Sie unter Daten in Amazon OpenSearch Service mit SQL von Amazon Athena abfragen im AWS-Big-Data-Blog.
- Weitere Informationen zu diesem Konnektor finden Sie unter der entsprechenden Seite auf GitHub.com.

Amazon Athena Oracle Konnektor

Der Amazon-Athena-Konnektor für Oracle ermöglicht es Amazon Athena, SQL-Abfragen für Daten auszuführen, die in Oracle gespeichert sind, die On-Premises oder auf Amazon EC2 oder Amazon RDS ausgeführt werden. Sie können den Konnektor auch verwenden, um Daten auf Oracle Exadata abzufragen.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Einschränkungen

- Schreiboperationen wie DDL werden nicht unterstützt.
- In einem Multiplexer-Setup werden der Überlauf-Bucket und das Präfix von allen Datenbank-Instances gemeinsam genutzt.
- Alle relevanten Lambda-Grenzwerte. Weitere Informationen finden Sie unter Lambda quotas (Lambda-Kontingente) im AWS Lambda -Entwicklerhandbuch.

Bedingungen

Die folgenden Begriffe beziehen sich auf den Oracle-Konnektor.

- Datenbank-Instance – Jede Instance einer Datenbank, die On-Premises, in Amazon EC2 oder auf Amazon RDS bereitgestellt wird.
- Handler – Ein Lambda-Handler, der auf Ihre Datenbank-Instance zugreift. Ein Handler kann für Metadaten oder für Datensätze verwendet werden.
- Metadaten-Handler – Ein Lambda-Handler, der Metadaten von Ihrer Datenbank-Instance abruft.
- Record Handler – Ein Lambda-Handler, der Datensätze aus Ihrer Datenbank-Instance abruft.
- Composite Handler – Ein Lambda-Handler, der sowohl Metadaten als auch Datensätze aus Ihrer Datenbank-Instance abruft.
- Eigenschaft oder Parameter – Eine Datenbankeigenschaft, die von Handlern zum Extrahieren von Datenbankinformationen verwendet wird. Sie konfigurieren diese Eigenschaften als Lambda-Umgebungsvariablen.
- Verbindungszeichenfolge – Eine Textzeichenfolge, die verwendet wird, um eine Verbindung zu einer Datenbank-Instance herzustellen.
- Katalog — Ein nicht bei Athena registrierter AWS Glue Katalog, der ein erforderliches Präfix für die `connection_string` Immobilie ist.
- Multiplex-Handler – Ein Lambda-Handler, der mehrere Datenbankverbindungen akzeptieren und verwenden kann.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den Oracle-Konnektor zu konfigurieren.

Verbindungszeichenfolge

Verwenden Sie eine JDBC-Verbindungszeichenfolge im folgenden Format, um eine Verbindung zu einer Datenbank-Instance herzustellen.

```
oracle://${jdbc_connection_string}
```

> **Note**
>
> Wenn Ihr Passwort Sonderzeichen enthält (z. B. `some.password`), schließen Sie Ihr Passwort in doppelte Anführungszeichen ein, wenn Sie es an die Verbindungszeichenfolge übergeben (z. B. `"some.password"`). Wenn Sie dies nicht tun, kann dies zu einem Fehler Angabe einer ungültigen Oracle-URL führen.

Verwenden eines Multiplexing-Handlers

Sie können einen Multiplexer verwenden, um mit einer einzigen Lambda-Funktion eine Verbindung zu mehreren Datenbank-Instances herzustellen. Anfragen werden anhand des Katalognamens weitergeleitet. Verwenden Sie die folgenden Klassen in Lambda.

Handler	Klasse
Composite Handler	`OracleMuxCompositeHandler`
Metadaten-Handler	`OracleMuxMetadataHandler`
Record Handler	`OracleMuxRecordHandler`

Multiplex-Handler-Parameter

Parameter	Beschreibung
`$`*`catalog`*`_connection_string`	Erforderlich Eine Verbindungszeichenfolge einer Datenbank-Instance. Stellen Sie der Umgebungsvariablen den Namen des in Athena verwendeten Katalogs voran. Wenn zum Beispiel der bei Athena registrierte Katalog `myoraclecatalog` ist, dann lautet der Name der Umgebungsvariablen `myoraclecatalog_connection_string` .
`default`	Erforderlich Die standardmäßige Verbindungszeichenfolge. Diese Zeichenfolge wird verwendet, wenn der Katalog `lambda:${`*`AWS_LAMBDA_FUNCTION_NAME`*`}` ist.

Die folgenden Beispieleigenschaften gelten für eine Oracle MUX Lambda-Funktion, die zwei Datenbankinstanzen unterstützt: `oracle1` (die Standardeinstellung) und `oracle2`.

Eigenschaft	Wert
`default`	`oracle://jdbc:oracle:thin:${Test/RDS/Oracle1}@//oracle1.hostname:port/servicename`
`oracle_catalog1_connection_string`	`oracle://jdbc:oracle:thin:${Test/RDS/Oracle1}@//oracle1.hostname:port/servicename`
`oracle_catalog2_connection_string`	`oracle://jdbc:oracle:thin:${Test/RDS/Oracle2}@//oracle2.hostname:port/servicename`

Bereitstellen von Anmeldeinformationen

Um einen Benutzernamen und ein Kennwort für Ihre Datenbank in Ihrer JDBC-Verbindungszeichenfolge anzugeben, können Sie Eigenschaften von Verbindungszeichenfolgen oder AWS Secrets Manager verwenden.

- Verbindungszeichenfolge – Ein Benutzername und ein Kennwort können als Eigenschaften in der JDBC-Verbindungszeichenfolge angegeben werden.

  > **Important**
  >
  > Als bewährte Sicherheitsmethode sollten Sie keine fest kodierten Anmeldeinformationen in Ihren Umgebungsvariablen oder Verbindungszeichenfolgen verwenden. Informationen zum Verschieben von hartcodierten Geheimnissen nach finden Sie im AWS Secrets Manager Benutzerhandbuch unter Verschieben von hartcodierten AWS Secrets Manager Geheimnissen nach.AWS Secrets Manager

- AWS Secrets Manager— Um die Athena Federated Query-Funktion verwenden zu können AWS Secrets Manager, muss die mit Ihrer Lambda-Funktion verbundene VPC über Internetzugang oder einen VPC-Endpunkt verfügen, um eine Verbindung zu Secrets Manager herzustellen.

Sie können den Namen eines Geheimnisses in AWS Secrets Manager Ihre JDBC-Verbindungszeichenfolge eingeben. Der Konnektor ersetzt den geheimen Namen durch `username`- und `password`-Werte von Secrets Manager.

Für Amazon RDS-Datenbank-Instances ist diese Unterstützung eng integriert. Wenn Sie Amazon RDS verwenden, empfehlen wir dringend, eine Rotation der Anmeldeinformationen zu verwenden AWS Secrets Manager . Wenn Ihre Datenbank Amazon RDS nicht verwendet, speichern Sie die Anmeldeinformationen als JSON im folgenden Format:

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

> **Note**
>
> Wenn Ihr Passwort Sonderzeichen enthält (z. B. `some.password`), setzen Sie Ihr Passwort in doppelte Anführungszeichen, wenn Sie es in Secrets Manager speichern (z. B. `"some.password"`). Wenn Sie dies nicht tun, kann dies zu einem Fehler Angabe einer ungültigen Oracle-URL führen.

Beispiel einer Verbindungszeichenfolge mit einem geheimen Namen

Die folgende Zeichenfolge hat den geheimen Namen `${Test/RDS/Oracle}`.

```
oracle://jdbc:oracle:thin:${Test/RDS/Oracle}@//hostname:port/servicename
```

Der Konnektor verwendet den geheimen Namen, um Secrets abzurufen und den Benutzernamen und das Kennwort bereitzustellen, wie im folgenden Beispiel gezeigt.

```
oracle://jdbc:oracle:thin:username/password@//hostname:port/servicename
```

Derzeit erkennt der Oracle-Konnektor die `UID`- und `PWD`-JDBC-Eigenschaften.

Verwenden eines einzelnen Verbindungs-Handlers

Sie können die folgenden Einzelverbindungsmetadaten und Record Handler verwenden, um eine Verbindung zu einer einzelnen Oracle-Instance herzustellen.

Handler-Typ	Klasse
Composite Handler	`OracleCompositeHandler`
Metadaten-Handler	`OracleMetadataHandler`
Record Handler	`OracleRecordHandler`

Parameter für Einzelverbindungs-Handler

Parameter	Beschreibung
`default`	Erforderlich Die standardmäßige Verbindungszeichenfolge.

Die Einzelverbindungs-Handler unterstützen eine Datenbank-Instance und müssen einen `default`-Verbindungszeichenfolgenparameter bereitstellen. Alle anderen Verbindungszeichenfolgen werden ignoriert.

Der Konnektor unterstützt SSL-basierte Verbindungen zu Amazon RDS-Instances. Die Unterstützung ist auf das Transport Layer Security (TLS)-Protokoll und auf die Authentifizierung des Servers durch den Client beschränkt. Gegenseitige Authentifizierung wird in Amazon RDS nicht unterstützt. Die zweite Zeile in der folgenden Tabelle zeigt die Syntax für die Verwendung von SSL.

Die folgende Beispieleigenschaft gilt für eine einzelne Oracle-Instance, die von einer Lambda-Funktion unterstützt wird.

Eigenschaft	Wert
`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=)))`

Überlauf-Parameter

Das Lambda-SDK kann Daten an Amazon S3 übertragen. Alle Datenbank-Instances, auf die mit derselben Lambda-Funktion zugegriffen wird, werden an denselben Speicherort verschoben.

Parameter	Beschreibung
`spill_bucket`	Erforderlich Überlauf-Bucket-Name.
`spill_prefix`	Erforderlich Schlüssel-Prefix für den Überlauf-Bucket.
`spill_put_request_headers`	(Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Weitere mögliche Header finden Sie PutObject in der Amazon Simple Storage Service API-Referenz.

Datentypunterstützung

Die folgende Tabelle zeigt die entsprechenden Datentypen für JDBC, Oracle und Arrow.

JDBC	Oracle	Arrow
Boolesch	boolesch	Bit
Ganzzahl	N/A	Tiny
Short	smallint	Smallint
Ganzzahl	Ganzzahl	Int
Long	bigint	Bigint
float	float4	Float4
Double	float8	Float8
Datum	date	DateDay

JDBC	Oracle	Arrow
Zeitstempel	Zeitstempel	DateMilli
String	Text	Varchar
Bytes	bytes	Varbinary
BigDecimal	numeric(p,s)	Dezimal
ARRAY	N.z. (siehe Hinweis)	Auflisten

Partitionen und Splits

Partitionen werden verwendet, um zu bestimmen, wie Splits für den Konnektor generiert werden. Athena konstruiert eine synthetische Säule vom Typ `varchar`, die das Partitionierungsschema für die Tabelle darstellt, das dem Konnektor beim Generieren von Splits hilft. Der Konnektor ändert nicht die eigentliche Tabellendefinition.

Leistung

Oracle unterstützt native Partitionen. Der Athena-Oracle-Konnektor kann Daten von diesen Partitionen parallel abrufen. Wenn Sie sehr große Datenmengen mit einheitlicher Partitionsverteilung abfragen möchten, wird eine native Partitionierung dringend empfohlen. Die Auswahl einer Teilmenge von Spalten beschleunigt die Abfragelaufzeit erheblich und reduziert die gescannten Daten. Der Oracle-Konnektor ist aufgrund der Gleichzeitigkeit widerstandsfähig gegenüber Drosselung. Allerdings sind die Abfragelaufzeiten in der Regel lang.

Der Athena-Oracle-Konnektor führt einen Prädikat-Pushdown durch, um die von der Abfrage durchsuchten Daten zu reduzieren. Einfache Prädikate und komplexe Ausdrücke werden an den Konnektor übertragen, um die Menge der gescannten Daten zu reduzieren und die Laufzeit der Abfrageausführung zu verkürzen.

Prädikate

Ein Prädikat ist ein Ausdruck in der `WHERE`-Klausel einer SQL-Abfrage, der einen booleschen Wert ergibt und Zeilen auf der Grundlage mehrerer Bedingungen filtert. Der Athena-Oracle-Konnektor kann diese Ausdrücke kombinieren und sie direkt an Oracle weiterleiten, um die Funktionalität zu erweitern und die Menge der gescannten Daten zu reduzieren.

Die folgenden Athena-Oracle-Konnektor-Operatoren unterstützen Prädikat-Pushdown:

- Boolean: UND, ODER, NICHT
- Gleichheit: GLEICH, NICHT GLEICH, WENIGER_ALS, WENIGER_ODER_GLEICH, GRÖSSER_ALS, GRÖSSER_ODER_GLEICH, IST_NULL
- Arithmetik: ADDIEREN, SUBTRAHIEREN, MULTIPLIZIEREN, DIVIDIEREN, NEGIEREN
- Andere: WIE_MUSTER, IN

Beispiel für einen kombinierten Pushdown

Kombinieren Sie für erweiterte Abfragefunktionen die Pushdown-Typen wie im folgenden Beispiel:

```
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%');
```

Lizenzinformationen

Durch die Verwendung dieses Connectors erkennen Sie die Einbindung von Komponenten von Drittanbietern an. Eine Liste dieser Komponenten finden Sie in der Datei pom.xml für diesen Connector und stimmen den Bedingungen der jeweiligen Drittanbieterlizenzen zu, die in der Datei LICENSE.txt auf GitHub .com enthalten sind.

Weitere Informationen finden Sie auch unter

Die neuesten Informationen zur JDBC-Treiberversion finden Sie in der Datei pom.xml für den Oracle-Connector auf GitHub .com.

Weitere Informationen zu diesem Connector finden Sie auf GitHubder entsprechenden Website unter .com.

Amazon Athena PostgreSQL Konnektor

Der Amazon-Athena-PostgreSQL-Konnektor ermöglicht Athena den Zugriff auf Ihre PostgreSQL-Datenbanken.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Einschränkungen

- Schreiboperationen wie DDL werden nicht unterstützt.
- In einem Multiplexer-Setup werden der Überlauf-Bucket und das Präfix von allen Datenbank-Instances gemeinsam genutzt.
- Alle relevanten Lambda-Grenzwerte. Weitere Informationen finden Sie unter Lambda quotas (Lambda-Kontingente) im AWS Lambda-Entwicklerhandbuch.

Bedingungen

Die folgenden Begriffe beziehen sich auf den PostgreSQL-Konnektor.

- Datenbank-Instance – Jede Instance einer Datenbank, die On-Premises, in Amazon EC2 oder auf Amazon RDS bereitgestellt wird.
- Handler – Ein Lambda-Handler, der auf Ihre Datenbank-Instance zugreift. Ein Handler kann für Metadaten oder für Datensätze verwendet werden.
- Metadaten-Handler – Ein Lambda-Handler, der Metadaten von Ihrer Datenbank-Instance abruft.
- Record Handler – Ein Lambda-Handler, der Datensätze aus Ihrer Datenbank-Instance abruft.
- Composite Handler – Ein Lambda-Handler, der sowohl Metadaten als auch Datensätze aus Ihrer Datenbank-Instance abruft.
- Eigenschaft oder Parameter – Eine Datenbankeigenschaft, die von Handlern zum Extrahieren von Datenbankinformationen verwendet wird. Sie konfigurieren diese Eigenschaften als Lambda-Umgebungsvariablen.
- Verbindungszeichenfolge – Eine Textzeichenfolge, die verwendet wird, um eine Verbindung zu einer Datenbank-Instance herzustellen.
- Katalog – Ein Nicht-AWS Glue-Katalog, der bei Athena registriert ist und ein erforderliches Präfix für die `connection_string`-Eigenschaft darstellt.

- Multiplex-Handler – Ein Lambda-Handler, der mehrere Datenbankverbindungen akzeptieren und verwenden kann.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den PostgreSQL-Konnektor zu konfigurieren.

Verbindungszeichenfolge

Verwenden Sie eine JDBC-Verbindungszeichenfolge im folgenden Format, um eine Verbindung zu einer Datenbank-Instance herzustellen.

```
postgres://${jdbc_connection_string}
```

Verwenden eines Multiplexing-Handlers

Sie können einen Multiplexer verwenden, um mit einer einzigen Lambda-Funktion eine Verbindung zu mehreren Datenbank-Instances herzustellen. Anfragen werden anhand des Katalognamens weitergeleitet. Verwenden Sie die folgenden Klassen in Lambda.

Handler	Klasse
Composite Handler	`PostGreSqlMuxCompositeHandler`
Metadaten-Handler	`PostGreSqlMuxMetadataHandler`
Record Handler	`PostGreSqlMuxRecordHandler`

Multiplex-Handler-Parameter

Parameter	Beschreibung
$*catalog*_connection_string	Erforderlich. Eine Verbindungszeichenfolge einer Datenbank-Instance. Stellen Sie der Umgebungsvariablen den Namen des in Athena verwendeten Katalogs voran. Wenn zum Beispiel der bei Athena registrierte Katalog `mypostgrescatalog` ist,

Parameter	Beschreibung
	dann lautet der Name der Umgebungsvariablen `mypostgrescatalog_connection_string` .
`default`	Erforderlich. Die standardmäßige Verbindungszeichenfolge. Diese Zeichenfolge wird verwendet, wenn der Katalog `lambda:${AWS_LAMBDA_FUNCTION_NAME}` ist.

Die folgenden Beispieleigenschaften gelten für eine PostgreSQL MUX Lambda-Funktion, die zwei Datenbankinstanzen unterstützt: `postgres1` (die Standardeinstellung) und `postgres2`.

Property (Eigenschaft)	Wert
`default`	`postgres://jdbc:postgresql://postgres1.host:5432/default?${Test/RDS/PostGres1}`
`postgres_catalog1_connection_string`	`postgres://jdbc:postgresql://postgres1.host:5432/default?${Test/RDS/PostGres1}`
`postgres_catalog2_connection_string`	`postgres://jdbc:postgresql://postgres2.host:5432/default?user=sample&password=sample`

Bereitstellen von Anmeldeinformationen

Um einen Benutzernamen und ein Kennwort für Ihre Datenbank in Ihrer JDBC-Verbindungszeichenfolge anzugeben, können Sie Eigenschaften von Verbindungszeichenfolgen oder AWS Secrets Manager verwenden.

- Verbindungszeichenfolge – Ein Benutzername und ein Kennwort können als Eigenschaften in der JDBC-Verbindungszeichenfolge angegeben werden.

> **Important**
>
> Als bewährte Sicherheitsmethode sollten Sie keine fest kodierten Anmeldeinformationen in Ihren Umgebungsvariablen oder Verbindungszeichenfolgen verwenden. Informationen zum Verschieben von hartkodierten Geheimnissen nach AWS Secrets Manager finden Sie unter Verschieben von hartkodierten Geheimnissen nach AWS Secrets Manager im AWS Secrets Manager-Benutzerhandbuch.

- AWS Secrets Manager – Um das Athena-Federated-Query-Feature mit AWS Secrets Manager zu verwenden, sollte die mit Ihrer Lambda-Funktion verbundene VPC über einen Internetzugang oder einen VPC-Endpunkt verfügen, um eine Verbindung zu Secrets Manager herzustellen.

  Sie können den Namen eines Secrets in AWS Secrets Manager in Ihrer JDBC-Verbindungszeichenfolge eingeben. Der Konnektor ersetzt den geheimen Namen durch `username`- und `password`-Werte von Secrets Manager.

  Für Amazon RDS-Datenbank-Instances ist diese Unterstützung eng integriert. Wenn Sie Amazon RDS verwenden, empfehlen wir dringend die Verwendung von AWS Secrets Manager und Wechsel der Anmeldeinformationen. Wenn Ihre Datenbank Amazon RDS nicht verwendet, speichern Sie die Anmeldeinformationen als JSON im folgenden Format:

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

Beispiel einer Verbindungszeichenfolge mit einem geheimen Namen

Die folgende Zeichenfolge hat den geheimen Namen `${Test/RDS/PostGres1}`.

```
postgres://jdbc:postgresql://postgres1.host:5432/default?...&${Test/RDS/PostGres1}&...
```

Der Konnektor verwendet den geheimen Namen, um Secrets abzurufen und den Benutzernamen und das Kennwort bereitzustellen, wie im folgenden Beispiel gezeigt.

```
postgres://jdbc:postgresql://postgres1.host:5432/
default?...&user=sample2&password=sample2&...
```

Derzeit erkennt der PostgreSQL-Konnektor die `user`- und `password`-JDBC-Eigenschaften.

Aktivieren von SSL

Um SSL in Ihrer PostgreSQL-Verbindung zu unterstützen, fügen Sie Folgendes an Ihre Verbindungszeichenfolge an:

```
&sslmode=verify-ca&sslfactory=org.postgresql.ssl.DefaultJavaSSLFactory
```

Beispiel

Die folgende Beispiel-Verbindungszeichenfolge verwendet kein SSL.

```
postgres://jdbc:postgresql://example-asdf-aurora-postgres-endpoint:5432/asdf?
user=someuser&password=somepassword
```

Um SSL zu aktivieren, ändern Sie die Zeichenfolge wie folgt.

```
postgres://jdbc:postgresql://example-asdf-aurora-postgres-
endpoint:5432/asdf?user=someuser&password=somepassword&sslmode=verify-
ca&sslfactory=org.postgresql.ssl.DefaultJavaSSLFactory
```

Verwenden eines einzelnen Verbindungs-Handlers

Sie können die folgenden Einzelverbindungsmetadaten und Record Handler verwenden, um eine Verbindung zu einer einzelnen PostgreSQL-Instance herzustellen.

Handler-Typ	Klasse
Composite Handler	`PostGreSqlCompositeHandler`
Metadaten-Handler	`PostGreSqlMetadataHandler`
Record Handler	`PostGreSqlRecordHandler`

Parameter für Einzelverbindungs-Handler

Parameter	Beschreibung
`default`	Erforderlich. Die standardmäßige Verbindungszeichenfolge.

Die Einzelverbindungs-Handler unterstützen eine Datenbank-Instance und müssen einen `default`-Verbindungszeichenfolgenparameter bereitstellen. Alle anderen Verbindungszeichenfolgen werden ignoriert.

Die folgende Beispieleigenschaft gilt für eine einzelne PostgreSQL-Instance, die von einer Lambda-Funktion unterstützt wird.

Property (Eigenschaft)	Wert
`default`	`postgres://jdbc:postgresql://postgres1.host:5432/default?secret=${Test/RDS/PostgreSQL1}`

Überlauf-Parameter

Das Lambda-SDK kann Daten an Amazon S3 übertragen. Alle Datenbank-Instances, auf die mit derselben Lambda-Funktion zugegriffen wird, werden an denselben Speicherort verschoben.

Parameter	Beschreibung
`spill_bucket`	Erforderlich. Überlauf-Bucket-Name.
`spill_prefix`	Erforderlich. Schlüssel-Prefix für den Überlauf-Bucket.
`spill_put_request_headers`	(Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Andere mögliche Header finden Sie unter PutObject in der API-Referenz zu Amazon Simple Storage Service.

Datentypunterstützung

Die folgende Tabelle zeigt die entsprechenden Datentypen für JDBC, PostGreSQL und Arrow.

JDBC	PostGreSQL	Arrow
Boolesch	Boolesch	Bit
Ganzzahl	–	Tiny
Short	smallint	Smallint
Ganzzahl	Ganzzahl	Int
Long	bigint	Bigint
float	float4	Float4
Double	float8	Float8
Datum	date	Dateday
Zeitstempel	timestamp	DateMilli
Zeichenfolge	text	Varchar
Bytes	bytes	Varbinary
BigDecimal	numeric(p,s)	Dezimal
ARRAY	N.z. (siehe Hinweis)	Auflisten

Note

Der ARRAY-Type wird für den PostgreSQL-Konnektor mit den folgenden Einschränkungen unterstützt: Multidimensionale Arrays (`<data_type>[][]` oder verschachtelte Arrays) werden nicht unterstützt. Spalten mit nicht unterstützten ARRAY-Datentypen werden in ein Array von Zeichenfolgen-Elementen konvertiert (`array<varchar>`).

Partitionen und Splits

Partitionen werden verwendet, um zu bestimmen, wie Splits für den Konnektor generiert werden. Athena konstruiert eine synthetische Säule vom Typ `varchar`, die das Partitionierungsschema für

die Tabelle darstellt, das dem Konnektor beim Generieren von Splits hilft. Der Konnektor ändert nicht die eigentliche Tabellendefinition.

Leistung

PostgreSQL unterstützt native Partitionen. Der Athena-PostgreSQL-Konnektor kann Daten von diesen Partitionen parallel abrufen. Wenn Sie sehr große Datenmengen mit einheitlicher Partitionsverteilung abfragen möchten, wird eine native Partitionierung dringend empfohlen.

Der Athena-PostgreSQL-Konnektor führt einen Prädikat-Pushdown durch, um die Anzahl der von der Abfrage gescannten Daten zu reduzieren. `LIMIT`-Klauseln, einfache Prädikate und komplexe Ausdrücke werden an den Konnektor übertragen, um die Menge der gescannten Daten zu reduzieren und die Laufzeit der Abfrage zu verkürzen. Die Auswahl einer Teilmenge von Spalten führt jedoch manchmal zu einer längeren Laufzeit der Abfrageausführung.

LIMIT-Klauseln

Eine `LIMIT N`-Anweisung reduziert die von der Abfrage durchsuchten Daten. Mit `LIMIT N`-Pushdown gibt der Konnektor nur `N` Zeilen an Athena zurück.

Prädikate

Ein Prädikat ist ein Ausdruck in der `WHERE`-Klausel einer SQL-Abfrage, der einen booleschen Wert ergibt und Zeilen auf der Grundlage mehrerer Bedingungen filtert. Der Athena-PostgreSQL-Konnektor kann diese Ausdrücke kombinieren und sie direkt an PostgreSQL weiterleiten, um die Funktionalität zu erweitern und die Menge der gescannten Daten zu reduzieren.

Die folgenden Athena-PostgreSQL-Konnektor-Operatoren unterstützen Prädikat-Pushdown:

- Boolean: UND, ODER, NICHT
- Gleichheit: GLEICH, NICHT-GLEICH, WENIGER_ALS, WENIGER_ODER-GLEICH, GRÖSSER_ALS, GRÖSSER_ODER-GLEICH, IST_UNTERSCHIEDEN VON, NULL_WENN, IST_NULL
- Arithmetik: ADDIEREN, SUBTRAHIEREN, MULTIPLIZIEREN, DIVIDIEREN, MODULIEREN, NEGIEREN
- Andere: WIE_MUSTER, IN

Beispiel für einen kombinierten Pushdown

Kombinieren Sie für erweiterte Abfragefunktionen die Pushdown-Typen wie im folgenden Beispiel:

```
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;
```

Weitere Informationen finden Sie auch unter

Die neuesten Informationen zur JDBC-Treiberversion finden Sie in der pom.xml-Datei für den PostgreSQL-Konnektor auf GitHub.com.

Weitere Informationen zu diesem Konnektor finden Sie unter der entsprechenden Seite auf GitHub.com.

Amazon Athena Redis Konnektor

Der Amazon Athena Redis-Konnektor ermöglicht Amazon Athena die Kommunikation mit Ihren Redis-Instances, sodass Sie Ihre Redis-Daten mit SQL abfragen können. Sie können die AWS Glue Data Catalog verwenden, um Ihre Redis-Schlüssel-Wert-Paare virtuellen Tabellen zuzuordnen.

Im Gegensatz zu herkömmlichen relationalen Datenspeichern gibt es bei Redis das Konzept von Tabelle oder Spalte nicht. Stattdessen bietet Redis Schlüssel-Wert-Zugriffsmuster an, bei denen der Schlüssel im Wesentlichen ein `string`und der Wert ein `string`, `z-set`, oder `hmap` ist.

Sie können das AWS Glue Data Catalog verwenden, um ein Schema zu erstellen und virtuelle Tabellen zu konfigurieren. Spezielle Tabelleneigenschaften teilen dem Athena-Redis-Konnektor mit, wie Sie Ihre Redis-Schlüssel und -Werte einer Tabelle zuordnen. Weitere Informationen finden Sie unter Einrichten von Datenbanken und Tabellen in AWS Glue an späterer Stelle in diesem Dokument.

Wenn Sie Lake Formation in Ihrem Konto aktiviert haben, muss die IAM-Rolle für Ihren Athena-Verbund-Lambda-Konnektor, den Sie im AWS Serverless Application Repository bereitstellen, in Lake Formation über Lesezugriff auf das AWS Glue Data Catalog verfügen.

Der Amazon-Athena-Redis-Konnektor unterstützt Amazon MemoryDB für Redis und Amazon ElastiCache für Redis.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor

bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

- Richten Sie eine VPC und eine Sicherheitsgruppe ein, bevor Sie diesen Konnektor verwenden. Weitere Informationen finden Sie unter Erstellen einer VPC für einen Datenquellen-Connector.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den Redis-Konnektor zu konfigurieren.

- spill_bucket – Gibt den Amazon S3-Bucket für Daten an, die die Lambda-Funktionsgrenzen überschreiten.
- spill_prefix – (Optional) Ist standardmäßig ein Unterordner im angegebenen `spill_bucket` genannt `athena-federation-spill`. Wir empfehlen Ihnen, einen Amazon-S3-Speicher-Lebenszyklus an dieser Stelle zu konfigurieren, um die Überlaufe zu löschen, die älter als eine festgelegte Anzahl von Tagen oder Stunden sind.
- spill_put_request_headers – (Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Andere mögliche Header finden Sie unter PutObject in der API-Referenz zu Amazon Simple Storage Service.
- kms_key_id – (Optional) Standardmäßig werden alle Daten, die an Amazon S3 gesendet werden, mit dem AES-GCM-authentifizierten Verschlüsselungsmodus und einem zufällig generierten Schlüssel verschlüsselt. Damit Ihre Lambda-Funktion stärkere Verschlüsselungsschlüssel verwendet, die von KMS generiert werden, wie`a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, können Sie eine ID einer Verschlüsselung angeben.
- disable_spill_encryption – (Optional) Bei Einstellung auf `True`, wird die Spill-Verschlüsselung deaktiviert. Die Standardeinstellung ist `False`, sodass Daten, die an S3 übertrragen werden, mit AES-GCM verschlüsselt werden - entweder mit einem zufällig generierten Schlüssel oder mit KMS zum Generieren von Schlüsseln. Das Deaktivieren der Überlauf-Verschlüsselung kann die Leistung verbessern, insbesondere wenn Ihr Überlauf-Standort eine serverseitige Verschlüsselung verwendet.
- glue_catalog – (Optional) Verwenden Sie diese Option, um einen kontoübergreifenden AWS Glue-Katalog anzugeben. Standardmäßig versucht der Konnektor, Metadaten von seinem eigenen AWS Glue-Konto abzurufen.

Einrichten von Datenbanken und Tabellen in AWS Glue

Zur Aktivierung einer AWS Glue-Tabelle zur Verwendung mit Redis können Sie die folgenden Tabelleneigenschaften für die Tabelle festlegen: `redis-endpoint`, `redis-value-type` und entweder `redis-keys-zset` oder `redis-key-prefix`.

Darüber hinaus muss jede AWS Glue-Datenbank, die Redis-Tabellen enthält, eine `redis-db-flag` in der URI-Eigenschaft der Datenbank haben. Zum Festlegen des Wertes der `redis-db-flag`-URI-Eigenschaft, verwenden Sie die AWS Glue-Konsole zum Bearbeiten der Datenbank.

Die folgende Liste enthält Beschreibungen der Tabelleneigenschaften.

- redis-endpoint – (Erforderlich) Der *`hostname`*`:`*`port`*`:`*`password`* des Redis-Servers, der Daten für diese Tabelle enthält (z. B. `athena-federation-demo.cache.amazonaws.com:6379`). Alternativ können Sie den Endpunkt oder einen Teil des Endpunkts in AWS Secrets Manager durch die Verwendung von ${*`Secret_Name`*} als Wert der Tabelleneigenschaft speichern.

> **Note**
>
> Um das Athena-Federated-Query-Feature mit AWS Secrets Manager zu verwenden, sollte die mit Ihrer Lambda-Funktion verbundene VPC über einen Internetzugang oder einen VPC-Endpunkt verfügen, um eine Verbindung zu Secrets Manager herzustellen.

- redis-keys-zset – (Erforderlich, wenn `redis-key-prefix` nicht verwendet wird) Eine durch Kommas getrennte Liste von Schlüsseln, deren Wert ein zset (zum Beispiel `active-orders,pending-orders`) ist. Jeder der Werte in der Gruppe wird als Schlüssel behandelt, der Teil der Tabelle ist. Entweder muss die `redis-keys-zset`-Eigenschaft oder die `redis-key-prefix`-Eigenschaft festgelegt sein.
- redis-key-prefix – (Erforderlich, wenn `redis-keys-zset` nicht verwendet wird) Eine durch Kommas getrennte Liste von Schlüsselpräfixen zum Scannen nach Werten in der Tabelle (z. B. `accounts-*,acct-`). Entweder muss die `redis-key-prefix`-Eigenschaft oder die `redis-keys-zset`-Eigenschaft festgelegt sein.
- redis-value-type – (Erforderlich) Definiert, wie die Werte für die Schlüssel definiert sind, indem entweder `redis-key-prefix` oder `redis-keys-zset` Ihrer Tabelle zugeordnet werden. Ein Literal wird einer einzelnen Spalte zugeordnet. Ein zset wird auch einer einzelnen Spalte

zugeordnet, aber jeder Schlüssel kann viele Zeilen speichern. Ein Hash ermöglicht, dass jeder Schlüssel eine Zeile mit mehreren Spalten ist (z. B. ein Hash, Literal oder zset).

- redis-ssl-flag – (Optional) Bei `True` wird eine Redis-Verbindung erstellt, die SSL/TLS verwendet. Der Standardwert ist `False`.
- redis-cluster-flag – (Optional) Bei `True` wird die Unterstützung für geclusterte Redis-Instances aktiviert. Der Standardwert ist `False`.
- redis-db-number – (Optional) Gilt nur für eigenständige, nicht geclusterte Instances.) Legen Sie diese Zahl (z. B. 1, 2 oder 3) fest, um sie aus einer nicht standardmäßigen Redis-Datenbank zu lesen. Die Standardeinstellung ist Redis logische Datenbank 0. Diese Nummer bezieht sich nicht auf eine Datenbank in Athena oder AWS Glue, aber auf eine logische Redis-Datenbank. Weitere Informationen finden Sie unter SELECT Index in der MySQL-Dokumentation.

Datentypen

Der Redis-Konnektor unterstützt die folgenden Datentypen. Redis-Streams werden nicht unterstützt.

- Zeichenfolge
- Hash
- Sortierter Satz (ZSet)

Alle Redis-Werte werden abgerufen als `string`-Datentyp. Anschließend werden sie in einen der folgenden Apache Arrow-Datentypen konvertiert, basierend darauf, wie Ihre Tabellen in AWS Glue Data Catalog definiert sind.

AWS Glue-Datentyp	Apache Arrow-Datentyp
int	INT
Zeichenfolge	VARCHAR
bigint	BIGINT
double	FLOAT8
float	FLOAT4
smallint	SMALLINT

AWS Glue-Datentyp	Apache Arrow-Datentyp
tinyint	TINYINT
boolesch	BIT
Binary	VARBINARY

Erforderliche Berechtigungen

Ausführliche Informationen zu den für diesen Konnektor erforderlichen IAM-Richtlinien finden Sie im `Policies`-Abschnitt der athena-redis.yaml-Datei. In der folgenden Liste sind die erforderlichen Berechtigungen zusammengefasst.

- Amazon-S3-Schreibzugriff – Der Konnektor benötigt Schreibzugriff auf einen Speicherort in Amazon S3, um Ergebnisse aus großen Abfragen zu übertragen.
- Athena GetQueryExecution – Der Konnektor verwendet diese Berechtigung, um ein Fast-Fail durchzuführen, wenn die vorgeschaltete Athena-Abfrage beendet wurde.
- AWS Glue Data Catalog – Der Redis-Konnektor benötigt schreibgeschützten Zugriff auf AWS Glue Data Catalog, um Schemainformationen zu erhalten.
- CloudWatch Logs – Der Konnektor benötigt Zugriff auf CloudWatch Logs zum Speichern von Protokollen.
- AWS Secrets Manager-Lesezugriff – Wenn Sie Redis-Endpunktdetails in Secrets Manager speichern möchten, müssen Sie dem Konnektor Zugriff auf diese Secrets gewähren.
- Zugriff auf VPC – Der Konnektor erfordert die Fähigkeit, Schnittstellen an Ihre VPC anzuhängen und zu trennen, damit diese eine Verbindung zu dieser herstellen und mit Ihren Redis-Instances kommunizieren kann.

Leistung

Der Athena-Redis-Konnektor versucht, Abfragen für Ihre Redis-Instance entsprechend dem von Ihnen definierten Tabellentyp zu parallelisieren (z. B. zset- oder Präfix-Schlüssel).

Der Athena-Redis-Konnektor führt einen Prädikat-Pushdown durch, um die von der Abfrage durchsuchten Daten zu reduzieren. Abfragen, die ein Prädikat für den Primärschlüssel enthalten, schlagen jedoch mit einer Zeitüberschreitung fehl. `LIMIT`-Klauseln reduzieren die Menge der

gescannten Daten, aber wenn Sie kein Prädikat angeben, sollten Sie davon ausgehen, dass `SELECT`-Abfragen mit einer `LIMIT`-Klausel mindestens 16 MB Daten scannen. Der Redis-Konnektor ist aufgrund der Gleichzeitigkeit widerstandsfähig gegenüber Drosselung.

Lizenzinformationen

Das Amazon Athena Redis-Konnektor-Projekt ist lizenziert unter Apache-2.0-Lizenz.

Weitere Informationen finden Sie auch unter

Weitere Informationen zu diesem Konnektor finden Sie unter der entsprechenden Seite auf GitHub.com.

Amazon Athena Redshift Konnektor

Der Amazon-Athena-Redshift-Konnektor ermöglicht Amazon Athena den Zugriff auf Ihre Amazon Redshift-Datenbanken.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Einschränkungen

- Schreiboperationen wie DDL werden nicht unterstützt.
- In einem Multiplexer-Setup werden der Überlauf-Bucket und das Präfix von allen Datenbank-Instances gemeinsam genutzt.
- Alle relevanten Lambda-Grenzwerte. Weitere Informationen finden Sie unter Lambda quotas (Lambda-Kontingente) im AWS Lambda-Entwicklerhandbuch.
- Da Redshift keine externen Partitionen unterstützt, werden alle in einer Abfrage angegebenen Daten jedes Mal abgerufen.

Bedingungen

Die folgenden Begriffe beziehen sich auf den Redshift-Konnektor.

- Datenbank-Instance – Jede Instance einer Datenbank, die On-Premises, in Amazon EC2 oder auf Amazon RDS bereitgestellt wird.
- Handler – Ein Lambda-Handler, der auf Ihre Datenbank-Instance zugreift. Ein Handler kann für Metadaten oder für Datensätze verwendet werden.
- Metadaten-Handler – Ein Lambda-Handler, der Metadaten von Ihrer Datenbank-Instance abruft.
- Record Handler – Ein Lambda-Handler, der Datensätze aus Ihrer Datenbank-Instance abruft.
- Composite Handler – Ein Lambda-Handler, der sowohl Metadaten als auch Datensätze aus Ihrer Datenbank-Instance abruft.
- Eigenschaft oder Parameter – Eine Datenbankeigenschaft, die von Handlern zum Extrahieren von Datenbankinformationen verwendet wird. Sie konfigurieren diese Eigenschaften als Lambda-Umgebungsvariablen.
- Verbindungszeichenfolge – Eine Textzeichenfolge, die verwendet wird, um eine Verbindung zu einer Datenbank-Instance herzustellen.
- Katalog – Ein Nicht-AWS Glue-Katalog, der bei Athena registriert ist und ein erforderliches Präfix für die `connection_string`-Eigenschaft darstellt.
- Multiplex-Handler – Ein Lambda-Handler, der mehrere Datenbankverbindungen akzeptieren und verwenden kann.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den Redshift-Konnektor zu konfigurieren.

Verbindungszeichenfolge

Verwenden Sie eine JDBC-Verbindungszeichenfolge im folgenden Format, um eine Verbindung zu einer Datenbank-Instance herzustellen.

```
redshift://${jdbc_connection_string}
```

Verwenden eines Multiplexing-Handlers

Sie können einen Multiplexer verwenden, um mit einer einzigen Lambda-Funktion eine Verbindung zu mehreren Datenbank-Instances herzustellen. Anfragen werden anhand des Katalognamens weitergeleitet. Verwenden Sie die folgenden Klassen in Lambda.

Handler	Klasse
Composite Handler	`RedshiftMuxCompositeHandler`
Metadaten-Handler	`RedshiftMuxMetadataHandler`
Record Handler	`RedshiftMuxRecordHandler`

Multiplex-Handler-Parameter

Parameter	Beschreibung
`$`*`catalog`*`_connection_string`	Erforderlich. Eine Verbindungszeichenfolge einer Datenbank-Instance. Stellen Sie der Umgebungsvariablen den Namen des in Athena verwendeten Katalogs voran. Wenn zum Beispiel der bei Athena registrierte Katalog `myredshiftcatalog` ist, dann lautet der Name der Umgebungsvariablen `myredshiftcatalog_connection_string` .
`default`	Erforderlich. Die standardmäßige Verbindungszeichenfolge. Diese Zeichenfolge wird verwendet, wenn der Katalog `lambda:${`*`AWS_LAMBDA_FUNCTION_NAME`*`}` ist.

Die folgenden Beispieleigenschaften gelten für eine Redshift MUX Lambda-Funktion, die zwei Datenbankinstanzen unterstützt: `redshift1` (die Standardeinstellung) und `redshift2`.

Property (Eigenschaft)	Wert
`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}`

Property (Eigenschaft)	Wert
`redshift_catalog2_connectio n_string`	`redshift://jdbc:redshift:// redshift2.host:3333/default? user=sample2&password=sample2`

Bereitstellen von Anmeldeinformationen

Um einen Benutzernamen und ein Kennwort für Ihre Datenbank in Ihrer JDBC-Verbindungszeichenfolge anzugeben, können Sie Eigenschaften von Verbindungszeichenfolgen oder AWS Secrets Manager verwenden.

- Verbindungszeichenfolge – Ein Benutzername und ein Kennwort können als Eigenschaften in der JDBC-Verbindungszeichenfolge angegeben werden.

  > **Important**
  >
  > Als bewährte Sicherheitsmethode sollten Sie keine fest kodierten Anmeldeinformationen in Ihren Umgebungsvariablen oder Verbindungszeichenfolgen verwenden. Informationen zum Verschieben von hartkodierten Geheimnissen nach AWS Secrets Manager finden Sie unter Verschieben von hartkodierten Geheimnissen nach AWS Secrets Manager im AWS Secrets Manager-Benutzerhandbuch.

- AWS Secrets Manager – Um das Athena-Federated-Query-Feature mit AWS Secrets Manager zu verwenden, sollte die mit Ihrer Lambda-Funktion verbundene VPC über einen Internetzugang oder einen VPC-Endpunkt verfügen, um eine Verbindung zu Secrets Manager herzustellen.

  Sie können den Namen eines Secrets in AWS Secrets Manager in Ihrer JDBC-Verbindungszeichenfolge eingeben. Der Konnektor ersetzt den geheimen Namen durch `username`- und `password`-Werte von Secrets Manager.

  Für Amazon RDS-Datenbank-Instances ist diese Unterstützung eng integriert. Wenn Sie Amazon RDS verwenden, empfehlen wir dringend die Verwendung von AWS Secrets Manager und Wechsel der Anmeldeinformationen. Wenn Ihre Datenbank Amazon RDS nicht verwendet, speichern Sie die Anmeldeinformationen als JSON im folgenden Format:

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

Beispiel einer Verbindungszeichenfolge mit einem Secret-Namen

Die folgende Zeichenfolge hat den Secret-Namen ${Test/RDS/ `Redshift1`}.

```
redshift://jdbc:redshift://redshift1.host:3306/default?...&${Test/RDS/Redshift1}&...
```

Der Konnektor verwendet den Secret-Namen, um Secrets abzurufen und den Benutzernamen und das Kennwort bereitzustellen, wie im folgenden Beispiel gezeigt.

```
redshift://jdbc:redshift://redshift1.host:3306/
default?...&user=sample2&password=sample2&...
```

Derzeit erkennt der Redshift-Konnektor die `user`- und `password`-JDBC-Eigenschaften.

Überlauf-Parameter

Das Lambda-SDK kann Daten an Amazon S3 übertragen. Alle Datenbank-Instances, auf die mit derselben Lambda-Funktion zugegriffen wird, werden an denselben Speicherort verschoben.

Parameter	Beschreibung
`spill_bucket`	Erforderlich. Überlauf-Bucket-Name.
`spill_prefix`	Erforderlich. Schlüssel-Prefix für den Überlauf-Bucket.
`spill_put_request_headers`	(Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject` -Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Andere mögliche Header finden Sie unter PutObject in der API-Referenz zu Amazon Simple Storage Service.

Datentypunterstützung

Die folgende Tabelle zeigt die entsprechenden Datentypen für JDBC und Apache Arrow.

JDBC	Arrow
Boolesch	Bit

JDBC	Arrow
Ganzzahl	Tiny
Short	Smallint
Ganzzahl	Int
Long	Bigint
float	Float4
Double	Float8
Datum	Dateday
Zeitstempel	DateMilli
Zeichenfolge	Varchar
Bytes	Varbinary
BigDecimal	Dezimal
ARRAY	Auflisten

Partitionen und Splits

Redshift unterstützt keine externen Partitionen. Informationen zur leistungsbezogenen Problemen finden Sie unter Leistung.

Leistung

Der Athena-Redshift-Konnektor führt einen Prädikat-Pushdown durch, um die Anzahl der von der Abfrage gescannten Daten zu reduzieren. `LIMIT`-Klauseln, `ORDER BY`-Klauseln, einfache Prädikate und komplexe Ausdrücke werden an den Konnektor übertragen, um die Menge der gescannten Daten zu reduzieren und die Laufzeit der Abfrage zu verkürzen. Die Auswahl einer Teilmenge von Spalten führt jedoch manchmal zu einer längeren Laufzeit der Abfrageausführung. Amazon Redshift ist besonders anfällig für eine Verlangsamung der Abfrageausführung, wenn Sie mehrere Abfragen gleichzeitig ausführen.

LIMIT-Klauseln

Eine `LIMIT N`-Anweisung reduziert die von der Abfrage durchsuchten Daten. Mit `LIMIT N`-Pushdown gibt der Konnektor nur `N` Zeilen an Athena zurück.

Top-N-Abfragen

Eine Top-`N`-Abfrage gibt eine Reihenfolge der Ergebnismenge und eine Obergrenze für die Anzahl der zurückgegebenen Zeilen an. Sie können diesen Abfragetyp verwenden, um die höchsten `N`-Höchstwerte oder die höchsten `N`-Minimalwerte für Ihre Datensätze zu ermitteln. Mit `N`-Pushdown gibt der Konnektor nur `N`-geordnete Zeilen an Athena zurück.

Prädikate

Ein Prädikat ist ein Ausdruck in der `WHERE`-Klausel einer SQL-Abfrage, der einen booleschen Wert ergibt und Zeilen auf der Grundlage mehrerer Bedingungen filtert. Der Athena-Redshift-Konnektor kann diese Ausdrücke kombinieren und sie direkt an Redshift weiterleiten, um die Funktionalität zu erweitern und die Menge der gescannten Daten zu reduzieren.

Die folgenden Athena-Redshift-Konnektor-Operatoren unterstützen Prädikat-Pushdown:

- Boolean: UND, ODER, NICHT
- Gleichheit: GLEICH, NICHT-GLEICH, WENIGER_ALS, WENIGER_ODER-GLEICH, GRÖSSER_ALS, GRÖSSER_ODER-GLEICH, IST_UNTERSCHIEDEN VON, NULL_WENN, IST_NULL
- Arithmetik: ADDIEREN, SUBTRAHIEREN, MULTIPLIZIEREN, DIVIDIEREN, MODULIEREN, NEGIEREN
- Andere: WIE_MUSTER, IN

Beispiel für einen kombinierten Pushdown

Kombinieren Sie für erweiterte Abfragefunktionen die Pushdown-Typen wie im folgenden Beispiel:

```
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;
```

Einen Artikel über die Verwendung von Prädikat-Pushdown zur Verbesserung der Leistung bei Verbundabfragen, einschließlich Amazon Redshift, finden Sie unter Verbessern von Verbundabfragen mit Prädikat-Pushdown in Amazon Athena im AWS-Big-Data-Blog.

Weitere Informationen finden Sie auch unter

Die neuesten Informationen zur JDBC-Treiberversion finden Sie in der pom.xml-Datei für den Redshift-Konnektor auf GitHub.com.

Weitere Informationen zu diesem Konnektor finden Sie unter der entsprechenden Seite auf GitHub.com.

Amazon Athena SAP HANA Konnektor

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Einschränkungen

- Schreiboperationen wie DDL werden nicht unterstützt.
- In einem Multiplexer-Setup werden der Überlauf-Bucket und das Präfix von allen Datenbank-Instances gemeinsam genutzt.
- Alle relevanten Lambda-Grenzwerte. Weitere Informationen finden Sie unter Lambda quotas (Lambda-Kontingente) im AWS Lambda -Entwicklerhandbuch.
- In SAP HANA werden Objektnamen in Großbuchstaben konvertiert, wenn sie in der SAP-HANA-Datenbank gespeichert werden. Da bei Namen in Anführungszeichen jedoch zwischen Groß- und Kleinschreibung unterschieden wird, ist es möglich, dass zwei Tabellen denselben Namen in Klein- und Großbuchstaben haben (z. B. `EMPLOYEE` und `employee`).

  In Athena Federated Query werden Schematabellennamen für die Lambda-Funktion in Kleinbuchstaben bereitgestellt. Um dieses Problem zu umgehen, können Sie `@schemaCase`-Abfragehinweise zum Abrufen der Daten aus Tabellen bereitstellen, bei deren Namen zwischen Groß- und Kleinschreibung unterscheiden wird. Im Folgenden finden Sie zwei Beispielabfragen mit Abfragehinweisen.

```
SELECT *
FROM "lambda:saphanaconnector".SYSTEM."MY_TABLE@schemaCase=upper&tableCase=upper"
```

```
SELECT *
FROM "lambda:saphanaconnector".SYSTEM."MY_TABLE@schemaCase=upper&tableCase=lower"
```

Bedingungen

Die folgenden Begriffe beziehen sich auf den SAP-HANA-Konnektor.

- Datenbank-Instance – Jede Instance einer Datenbank, die On-Premises, in Amazon EC2 oder auf Amazon RDS bereitgestellt wird.
- Handler – Ein Lambda-Handler, der auf Ihre Datenbank-Instance zugreift. Ein Handler kann für Metadaten oder für Datensätze verwendet werden.
- Metadaten-Handler – Ein Lambda-Handler, der Metadaten von Ihrer Datenbank-Instance abruft.
- Record Handler – Ein Lambda-Handler, der Datensätze aus Ihrer Datenbank-Instance abruft.
- Composite Handler – Ein Lambda-Handler, der sowohl Metadaten als auch Datensätze aus Ihrer Datenbank-Instance abruft.
- Eigenschaft oder Parameter – Eine Datenbankeigenschaft, die von Handlern zum Extrahieren von Datenbankinformationen verwendet wird. Sie konfigurieren diese Eigenschaften als Lambda-Umgebungsvariablen.
- Verbindungszeichenfolge – Eine Textzeichenfolge, die verwendet wird, um eine Verbindung zu einer Datenbank-Instance herzustellen.
- Katalog — Ein nicht bei Athena registrierter AWS Glue Katalog, der ein erforderliches Präfix für die `connection_string` Immobilie ist.
- Multiplex-Handler – Ein Lambda-Handler, der mehrere Datenbankverbindungen akzeptieren und verwenden kann.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den SAP-HANA-Konnektor zu konfigurieren.

Verbindungszeichenfolge

Verwenden Sie eine JDBC-Verbindungszeichenfolge im folgenden Format, um eine Verbindung zu einer Datenbank-Instance herzustellen.

```
saphana://${jdbc_connection_string}
```

Verwenden eines Multiplexing-Handlers

Sie können einen Multiplexer verwenden, um mit einer einzigen Lambda-Funktion eine Verbindung zu mehreren Datenbank-Instances herzustellen. Anfragen werden anhand des Katalognamens weitergeleitet. Verwenden Sie die folgenden Klassen in Lambda.

Handler	Klasse
Composite Handler	`SaphanaMuxCompositeHandler`
Metadaten-Handler	`SaphanaMuxMetadataHandler`
Record Handler	`SaphanaMuxRecordHandler`

Multiplex-Handler-Parameter

Parameter	Beschreibung
$*catalog*_connection_string	Erforderlich Eine Verbindungszeichenfolge einer Datenbank-Instance. Stellen Sie der Umgebungsvariablen den Namen des in Athena verwendeten Katalogs voran. Wenn zum Beispiel der bei Athena registrierte Katalog `mysaphanacatalog` ist, dann lautet der Name der Umgebungsvariablen `mysaphanacatalog_connection_string` .
`default`	Erforderlich Die standardmäßige Verbindungszeichenfolge. Diese Zeichenfolge wird verwendet, wenn der Katalog `lambda:${`*AWS_LAMBDA_FUNCTION_NAME*`}` ist.

Die folgenden Beispieleigenschaften gelten für eine SAP HANA MUX Lambda-Funktion, die zwei Datenbankinstanzen unterstützt: `saphana1` (die Standardeinstellung) und `saphana2`.

Eigenschaft	Wert
`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`

Bereitstellen von Anmeldeinformationen

Um einen Benutzernamen und ein Kennwort für Ihre Datenbank in Ihrer JDBC-Verbindungszeichenfolge anzugeben, können Sie Eigenschaften von Verbindungszeichenfolgen oder AWS Secrets Manager verwenden.

- Verbindungszeichenfolge – Ein Benutzername und ein Kennwort können als Eigenschaften in der JDBC-Verbindungszeichenfolge angegeben werden.

  > **Important**
  >
  > Als bewährte Sicherheitsmethode sollten Sie keine fest kodierten Anmeldeinformationen in Ihren Umgebungsvariablen oder Verbindungszeichenfolgen verwenden. Informationen zum Verschieben von hartcodierten Geheimnissen nach finden Sie im AWS Secrets Manager Benutzerhandbuch unter Verschieben von hartcodierten AWS Secrets Manager Geheimnissen nach.AWS Secrets Manager

- AWS Secrets Manager— Um die Athena Federated Query-Funktion verwenden zu können AWS Secrets Manager, muss die mit Ihrer Lambda-Funktion verbundene VPC über Internetzugang oder einen VPC-Endpunkt verfügen, um eine Verbindung zu Secrets Manager herzustellen.

Sie können den Namen eines Geheimnisses in AWS Secrets Manager Ihre JDBC-Verbindungszeichenfolge eingeben. Der Konnektor ersetzt den geheimen Namen durch `username`- und `password`-Werte von Secrets Manager.

Für Amazon RDS-Datenbank-Instances ist diese Unterstützung eng integriert. Wenn Sie Amazon RDS verwenden, empfehlen wir dringend, eine Rotation der Anmeldeinformationen zu verwenden AWS Secrets Manager . Wenn Ihre Datenbank Amazon RDS nicht verwendet, speichern Sie die Anmeldeinformationen als JSON im folgenden Format:

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

Beispiel einer Verbindungszeichenfolge mit einem geheimen Namen

Die folgende Zeichenfolge hat den geheimen Namen `${Test/RDS/Saphana1}`.

```
saphana://jdbc:sap://saphana1.host:port/?${Test/RDS/Saphana1}&...
```

Der Konnektor verwendet den geheimen Namen, um Secrets abzurufen und den Benutzernamen und das Kennwort bereitzustellen, wie im folgenden Beispiel gezeigt.

```
saphana://jdbc:sap://saphana1.host:port/?user=sample2&password=sample2&...
```

Derzeit erkennt der SAP-HANA-Konnektor die `user`- und `password`-JDBC-Eigenschaften.

Verwenden eines einzelnen Verbindungs-Handlers

Sie können die folgenden Einzelverbindungsmetadaten und Record Handler verwenden, um eine Verbindung zu einer einzelnen SAP HANA-Instance herzustellen.

Handler-Typ	Klasse
Composite Handler	`SaphanaCompositeHandler`
Metadaten-Handler	`SaphanaMetadataHandler`
Record Handler	`SaphanaRecordHandler`

Parameter für Einzelverbindungs-Handler

Parameter	Beschreibung
`default`	Erforderlich Die standardmäßige Verbindungszeichenfolge.

Die Einzelverbindungs-Handler unterstützen eine Datenbank-Instance und müssen einen `default`-Verbindungszeichenfolgenparameter bereitstellen. Alle anderen Verbindungszeichenfolgen werden ignoriert.

Die folgende Beispieleigenschaft gilt für eine einzelne SAP HANA-Instance, die von einer Lambda-Funktion unterstützt wird.

Eigenschaft	Wert
`default`	`saphana://jdbc:sap://saphana1.host:port/?secret=Test/RDS/Saphana1`

Überlauf-Parameter

Das Lambda-SDK kann Daten an Amazon S3 übertragen. Alle Datenbank-Instances, auf die mit derselben Lambda-Funktion zugegriffen wird, werden an denselben Speicherort verschoben.

Parameter	Beschreibung
`spill_bucket`	Erforderlich Überlauf-Bucket-Name.
`spill_prefix`	Erforderlich Schlüssel-Prefix für den Überlauf-Bucket.
`spill_put_request_headers`	(Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Weitere mögliche Header finden Sie PutObject in der Amazon Simple Storage Service API-Referenz.

Datentypunterstützung

Die folgende Tabelle zeigt die entsprechenden Datentypen für JDBC und Apache Arrow.

JDBC	Arrow
Boolesch	Bit
Ganzzahl	Tiny
Short	Smallint
Ganzzahl	Int
Long	Bigint
float	Float4
Double	Float8
Datum	DateDay
Zeitstempel	DateMilli
String	Varchar
Bytes	Varbinary
BigDecimal	Dezimal
ARRAY	Auflisten

Datentypkonvertierungen

Zusätzlich zu den Konvertierungen von JDBC in Arrow führt der Konnektor bestimmte andere Konvertierungen durch, um die SAP-HANA-Quelle und die Athena-Datentypen kompatibel zu machen. Diese Konvertierungen tragen dazu bei, dass Abfragen erfolgreich ausgeführt werden. Die folgende Tabelle zeigt diese Konvertierungen.

Quelldatentyp (SAP HANA)	Konvertierter Datentyp (Athena)
DECIMAL	BIGINT
INTEGER	INT
DATUM	DATEDAY
TIMESTAMP (ZEITSTEMPEL)	DATEMILLI

Alle anderen nicht unterstützten Datentypen werden in `VARCHAR` konvertiert.

Partitionen und Splits

Eine Partition wird durch eine einzelne Partitionsspalte vom Typ `Integer` dargestellt. Die Spalte enthält Partitionsnamen der Partitionen, die in einer SAP HANA-Tabelle definiert sind. Für eine Tabelle, die keine Partitionsnamen hat, wird * ausgegeben, was einer einzelnen Partition entspricht. Eine Partition entspricht einem Split.

Name	Typ	Beschreibung
PART_ID	Ganzzahl	Benannte Partition in SAP HANA.

Leistung

SAP HANA unterstützt native Partitionen. Der Athena-SAP-HANA-Konnektor kann Daten von diesen Partitionen parallel abrufen. Wenn Sie sehr große Datenmengen mit einheitlicher Partitionsverteilung abfragen möchten, wird eine native Partitionierung dringend empfohlen. Die Auswahl einer Teilmenge von Spalten beschleunigt die Abfragelaufzeit erheblich und reduziert die gescannten Daten.
Der Konnektor zeigt aufgrund der Gleichzeitigkeit eine erhebliche Drosselung und manchmal Abfragefehler.

Der Athena-SAP-HANA-Konnektor führt einen Prädikat-Pushdown durch, um die Anzahl der von der Abfrage gescannten Daten zu reduzieren. `LIMIT`-Klauseln, einfache Prädikate und komplexe Ausdrücke werden an den Konnektor übertragen, um die Menge der gescannten Daten zu reduzieren und die Laufzeit der Abfrage zu verkürzen.

LIMIT-Klauseln

Eine `LIMIT N`-Anweisung reduziert die von der Abfrage durchsuchten Daten. Mit `LIMIT N`-Pushdown gibt der Konnektor nur `N` Zeilen an Athena zurück.

Prädikate

Ein Prädikat ist ein Ausdruck in der `WHERE`-Klausel einer SQL-Abfrage, der einen booleschen Wert ergibt und Zeilen auf der Grundlage mehrerer Bedingungen filtert. Der Athena-SAP-HANA-Konnektor kann diese Ausdrücke kombinieren und sie direkt an SAP HANA weiterleiten, um die Funktionalität zu erweitern und die Menge der gescannten Daten zu reduzieren.

Die folgenden Athena-SAP-HANA-Konnektor-Operatoren unterstützen Prädikat-Pushdown:

- Boolean: UND, ODER, NICHT
- Gleichheit: GLEICH, NICHT-GLEICH, WENIGER_ALS, WENIGER_ODER-GLEICH, GRÖSSER_ALS, GRÖSSER_ODER-GLEICH, IST_UNTERSCHIEDEN VON, NULL_WENN, IST_NULL
- Arithmetik: ADDIEREN, SUBTRAHIEREN, MULTIPLIZIEREN, DIVIDIEREN, MODULIEREN, NEGIEREN
- Andere: WIE_MUSTER, IN

Beispiel für einen kombinierten Pushdown

Kombinieren Sie für erweiterte Abfragefunktionen die Pushdown-Typen wie im folgenden Beispiel:

```
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;
```

Lizenzinformationen

Durch die Verwendung dieses Connectors erkennen Sie die Einbindung von Komponenten von Drittanbietern an. Eine Liste dieser Komponenten finden Sie in der Datei pom.xml für diesen Connector und stimmen den Bedingungen der jeweiligen Drittanbieterlizenzen zu, die in der Datei LICENSE.txt auf GitHub .com enthalten sind.

Weitere Informationen finden Sie auch unter

Die neuesten Informationen zur JDBC-Treiberversion finden Sie in der Datei pom.xml für den SAP HANA-Connector auf GitHub .com.

Weitere Informationen zu diesem Connector finden Sie auf GitHubder entsprechenden Website unter .com.

Amazon Athena Snowflake Konnektor

Der Amazon-Athena-Konnektor für Snowflake ermöglicht es Amazon Athena, SQL-Abfragen für Daten auszuführen, die in Snowflake gespeichert sind.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Einschränkungen

- Schreiboperationen wie DDL werden nicht unterstützt.
- In einem Multiplexer-Setup werden der Überlauf-Bucket und das Präfix von allen Datenbank-Instances gemeinsam genutzt.
- Alle relevanten Lambda-Grenzwerte. Weitere Informationen finden Sie unter Lambda quotas (Lambda-Kontingente) im AWS Lambda -Entwicklerhandbuch.
- Derzeit werden Snowflake-Ansichten mit Einzelaufteilung unterstützt.
- Da bei Objektnamen in Snowflake zwischen Groß- und Kleinschreibung unterschieden wird, können zwei Tabellen denselben Namen in Klein- und Großbuchstaben haben (z. B. `EMPLOYEE` und `employee`). In Athena Federated Query werden Schematabellennamen für die Lambda-Funktion in Kleinbuchstaben bereitgestellt. Um dieses Problem zu umgehen, können Sie `@schemaCase`-Abfragehinweise zum Abrufen der Daten aus Tabellen bereitstellen, bei deren Namen zwischen Groß- und Kleinschreibung unterscheiden wird. Im Folgenden finden Sie zwei Beispielabfragen mit Abfragehinweisen.

```
SELECT *
```

```
FROM "lambda:snowflakeconnector".SYSTEM."MY_TABLE@schemaCase=upper&tableCase=upper"
```

```
SELECT *
FROM "lambda:snowflakeconnector".SYSTEM."MY_TABLE@schemaCase=upper&tableCase=lower"
```

Bedingungen

Die folgenden Begriffe und Konzepte beziehen sich auf den Snowflake-Konnektor.

- Datenbank-Instance – Jede Instance einer Datenbank, die On-Premises, in Amazon EC2 oder auf Amazon RDS bereitgestellt wird.
- Handler – Ein Lambda-Handler, der auf Ihre Datenbank-Instance zugreift. Ein Handler kann für Metadaten oder für Datensätze verwendet werden.
- Metadaten-Handler – Ein Lambda-Handler, der Metadaten von Ihrer Datenbank-Instance abruft.
- Record Handler – Ein Lambda-Handler, der Datensätze aus Ihrer Datenbank-Instance abruft.
- Composite Handler – Ein Lambda-Handler, der sowohl Metadaten als auch Datensätze aus Ihrer Datenbank-Instance abruft.
- Eigenschaft oder Parameter – Eine Datenbankeigenschaft, die von Handlern zum Extrahieren von Datenbankinformationen verwendet wird. Sie konfigurieren diese Eigenschaften als Lambda-Umgebungsvariablen.
- Verbindungszeichenfolge – Eine Textzeichenfolge, die verwendet wird, um eine Verbindung zu einer Datenbank-Instance herzustellen.
- Katalog — Ein nicht bei Athena registrierter AWS Glue Katalog, der ein erforderliches Präfix für die `connection_string` Immobilie ist.
- Multiplex-Handler – Ein Lambda-Handler, der mehrere Datenbankverbindungen akzeptieren und verwenden kann.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den Snowflake-Konnektor zu konfigurieren.

Verbindungszeichenfolge

Verwenden Sie eine JDBC-Verbindungszeichenfolge im folgenden Format, um eine Verbindung zu einer Datenbank-Instance herzustellen.

```
snowflake://${jdbc_connection_string}
```

Verwenden eines Multiplexing-Handlers

Sie können einen Multiplexer verwenden, um mit einer einzigen Lambda-Funktion eine Verbindung zu mehreren Datenbank-Instances herzustellen. Anfragen werden anhand des Katalognamens weitergeleitet. Verwenden Sie die folgenden Klassen in Lambda.

Handler	Klasse
Composite Handler	`SnowflakeMuxCompositeHandler`
Metadaten-Handler	`SnowflakeMuxMetadataHandler`
Record Handler	`SnowflakeMuxRecordHandler`

Multiplex-Handler-Parameter

Parameter	Beschreibung
$*catalog*_connection_string	Erforderlich Eine Verbindungszeichenfolge einer Datenbank-Instance. Stellen Sie der Umgebungsvariablen den Namen des in Athena verwendeten Katalogs voran. Wenn zum Beispiel der bei Athena registrierte Katalog `mysnowflakecatalog` ist, dann lautet der Name der Umgebungsvariablen `mysnowflakecatalog_connection_string` .
`default`	Erforderlich Die standardmäßige Verbindungszeichenfolge. Diese Zeichenfolge wird verwendet, wenn der Katalog `lambda:${AWS_LAMBDA_FUNCTION_NAME}` ist.

Die folgenden Beispieleigenschaften beziehen sich auf eine Snowflake MUX Lambda-Funktion, die zwei Datenbank-Instances unterstützt: `snowflake1` (Standard) und `snowflake2`.

Eigenschaft	Wert
`default`	`snowflake://jdbc:snowflake://snowflake1.host:port/?warehouse=warehousename&db=db1&schema=schema1&${Test/RDS/Snowflake1}`
`snowflake_catalog1_connection_string`	`snowflake://jdbc:snowflake://snowflake1.host:port/?warehouse=warehousename&db=db1&schema=schema1${Test/RDS/Snowflake1}`
`snowflake_catalog2_connection_string`	`snowflake://jdbc:snowflake://snowflake2.host:port/?warehouse=warehousename&db=db1&schema=schema1&user=sample2&password=sample2`

Bereitstellen von Anmeldeinformationen

Um einen Benutzernamen und ein Kennwort für Ihre Datenbank in Ihrer JDBC-Verbindungszeichenfolge anzugeben, können Sie Eigenschaften von Verbindungszeichenfolgen oder AWS Secrets Manager verwenden.

- Verbindungszeichenfolge – Ein Benutzername und ein Kennwort können als Eigenschaften in der JDBC-Verbindungszeichenfolge angegeben werden.

> **Important**
>
> Als bewährte Sicherheitsmethode sollten Sie keine fest kodierten Anmeldeinformationen in Ihren Umgebungsvariablen oder Verbindungszeichenfolgen verwenden. Informationen zum Verschieben von hartcodierten Geheimnissen nach finden Sie im AWS Secrets Manager Benutzerhandbuch unter Verschieben von hartcodierten AWS Secrets Manager Geheimnissen nach.AWS Secrets Manager

- AWS Secrets Manager— Um die Athena Federated Query-Funktion verwenden zu können AWS Secrets Manager, muss die mit Ihrer Lambda-Funktion verbundene VPC über Internetzugang oder einen VPC-Endpunkt verfügen, um eine Verbindung zu Secrets Manager herzustellen.

  Sie können den Namen eines Geheimnisses in AWS Secrets Manager Ihre JDBC-Verbindungszeichenfolge eingeben. Der Konnektor ersetzt den geheimen Namen durch `username`- und `password`-Werte von Secrets Manager.

  Für Amazon RDS-Datenbank-Instances ist diese Unterstützung eng integriert. Wenn Sie Amazon RDS verwenden, empfehlen wir dringend, eine Rotation der Anmeldeinformationen zu verwenden AWS Secrets Manager . Wenn Ihre Datenbank Amazon RDS nicht verwendet, speichern Sie die Anmeldeinformationen als JSON im folgenden Format:

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

Beispiel einer Verbindungszeichenfolge mit einem geheimen Namen

Die folgende Zeichenfolge hat den geheimen Namen `${Test/RDS/Snowflake1}`.

```
snowflake://jdbc:snowflake://snowflake1.host:port/?
warehouse=warehousename&db=db1&schema=schema1${Test/RDS/Snowflake1}&...
```

Der Konnektor verwendet den geheimen Namen, um Secrets abzurufen und den Benutzernamen und das Kennwort bereitzustellen, wie im folgenden Beispiel gezeigt.

```
snowflake://jdbc:snowflake://snowflake1.host:port/
warehouse=warehousename&db=db1&schema=schema1&user=sample2&password=sample2&...
```

Derzeit erkennt Snowflake die `user`- und `password`-JDBC-Eigenschaften. Es akzeptiert auch den Benutzernamen und das Passwort im Format *`Nutzername`*`/`*`Passwort`* ohne die Schlüssel `user` oder `password`.

Verwenden eines einzelnen Verbindungs-Handlers

Sie können die folgenden Metadaten und Record Handler für eine einzelne Verbindung verwenden, um eine Verbindung zu einer einzelnen Snowflake-Instance herzustellen.

Handler-Typ	Klasse
Composite Handler	`SnowflakeCompositeHandler`
Metadaten-Handler	`SnowflakeMetadataHandler`
Record Handler	`SnowflakeRecordHandler`

Parameter für Einzelverbindungs-Handler

Parameter	Beschreibung
`default`	Erforderlich Die standardmäßige Verbindungszeichenfolge.

Die Einzelverbindungs-Handler unterstützen eine Datenbank-Instance und müssen einen `default`-Verbindungszeichenfolgenparameter bereitstellen. Alle anderen Verbindungszeichenfolgen werden ignoriert.

Die folgende Beispieleigenschaft gilt für eine einzelne Snowflake-Instance, die von einer Lambda-Funktion unterstützt wird.

Eigenschaft	Wert
`default`	`snowflake://jdbc:snowflake://snowflake1.host:port/?secret=Test/RDS/Snowflake1`

Überlauf-Parameter

Das Lambda-SDK kann Daten an Amazon S3 übertragen. Alle Datenbank-Instances, auf die mit derselben Lambda-Funktion zugegriffen wird, werden an denselben Speicherort verschoben.

Parameter	Beschreibung
`spill_bucket`	Erforderlich Überlauf-Bucket-Name.

Parameter	Beschreibung
`spill_prefix`	Erforderlich Schlüssel-Prefix für den Überlauf-Bucket.
`spill_put_request_headers`	(Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Weitere mögliche Header finden Sie PutObject in der Amazon Simple Storage Service API-Referenz.

Datentypunterstützung

Die folgende Tabelle zeigt die entsprechenden Datentypen für JDBC und Apache Arrow.

JDBC	Arrow
Boolesch	Bit
Ganzzahl	Tiny
Short	Smallint
Ganzzahl	Int
Long	Bigint
float	Float4
Double	Float8
Datum	DateDay
Zeitstempel	DateMilli
String	Varchar
Bytes	Varbinary
BigDecimal	Dezimal

JDBC	Arrow
ARRAY	Auflisten

Datentypkonvertierungen

Zusätzlich zu den Konvertierungen von JDBC in Arrow führt der Konnektor bestimmte andere Konvertierungen durch, um die Snowflake-Quelle und die Athena-Datentypen kompatibel zu machen. Diese Konvertierungen tragen dazu bei, dass Abfragen erfolgreich ausgeführt werden. Die folgende Tabelle zeigt diese Konvertierungen.

Quelldatentyp (Snowflake)	Konvertierter Datentyp (Athena)
TIMESTAMP (ZEITSTEMPEL)	TIMESTAMPMILLI
DATUM	TIMESTAMPMILLI
INTEGER	INT
DECIMAL	BIGINT
TIMESTAMP_NTZ	TIMESTAMPMILLI

Alle anderen nicht unterstützten Datentypen werden in `VARCHAR` konvertiert.

Partitionen und Splits

Partitionen werden verwendet, um zu bestimmen, wie Splits für den Konnektor generiert werden. Athena konstruiert eine synthetische Säule vom Typ `varchar`, die das Partitionierungsschema für die Tabelle darstellt, das dem Konnektor beim Generieren von Splits hilft. Der Konnektor ändert nicht die eigentliche Tabellendefinition.

Leistung

Verwenden Sie nach Möglichkeit Filter in Abfragen, um eine optimale Leistung zu erzielen. Darüber hinaus empfehlen wir dringend die native Partitionierung, um riesige Datensätze mit einheitlicher Partitionsverteilung abzurufen. Die Auswahl einer Teilmenge von Spalten beschleunigt die Abfragelaufzeit erheblich und reduziert die gescannten Daten. Der Snowflake-Konnektor ist aufgrund der Gleichzeitigkeit widerstandsfähig gegenüber Drosselung.

Der Athena-Snowflake-Konnektor führt einen Prädikat-Pushdown durch, um die Anzahl der von der Abfrage gescannten Daten zu reduzieren. `LIMIT`-Klauseln, einfache Prädikate und komplexe Ausdrücke werden an den Konnektor übertragen, um die Menge der gescannten Daten zu reduzieren und die Laufzeit der Abfrage zu verkürzen.

LIMIT-Klauseln

Eine `LIMIT N`-Anweisung reduziert die von der Abfrage durchsuchten Daten. Mit `LIMIT N`-Pushdown gibt der Konnektor nur `N` Zeilen an Athena zurück.

Prädikate

Ein Prädikat ist ein Ausdruck in der `WHERE`-Klausel einer SQL-Abfrage, der einen booleschen Wert ergibt und Zeilen auf der Grundlage mehrerer Bedingungen filtert. Der Athena-Snowflake-Konnektor kann diese Ausdrücke kombinieren und sie direkt an Snowflake weiterleiten, um die Funktionalität zu verbessern und die Menge der gescannten Daten zu reduzieren.

Die folgenden Athena-Snowflake-Konnektor-Operatoren unterstützen Prädikat-Pushdown:

- Boolean: UND, ODER, NICHT
- Gleichheit: GLEICH, NICHT-GLEICH, WENIGER_ALS, WENIGER_ODER-GLEICH, GRÖSSER_ALS, GRÖSSER_ODER-GLEICH, IST_UNTERSCHIEDEN VON, NULL_WENN, IST_NULL
- Arithmetik: ADDIEREN, SUBTRAHIEREN, MULTIPLIZIEREN, DIVIDIEREN, MODULIEREN, NEGIEREN
- Andere: WIE_MUSTER, IN

Beispiel für einen kombinierten Pushdown

Kombinieren Sie für erweiterte Abfragefunktionen die Pushdown-Typen wie im folgenden Beispiel:

```
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;
```

Lizenzinformationen

Durch die Verwendung dieses Connectors erkennen Sie die Einbindung von Komponenten von Drittanbietern an. Eine Liste dieser Komponenten finden Sie in der Datei pom.xml für diesen Connector und stimmen den Bedingungen der jeweiligen Drittanbieterlizenzen zu, die in der Datei LICENSE.txt auf GitHub .com enthalten sind.

Weitere Informationen finden Sie auch unter

Die neuesten Informationen zur JDBC-Treiberversion finden Sie in der Datei pom.xml für den Snowflake-Connector auf .com. GitHub

Weitere Informationen zu diesem Connector finden Sie auf der entsprechenden Website unter .com. GitHub

Amazon Athena Microsoft SQL Server Konnektor

Der Amazon-Athena-Konnektor für Microsoft SQL Server ermöglicht es Amazon Athena, SQL-Abfragen für Ihre Daten auszuführen, die in Microsoft SQL Server unter Verwendungt von JDBC gespeichert sind.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Einschränkungen

- Schreiboperationen wie DDL werden nicht unterstützt.
- In einem Multiplexer-Setup werden der Überlauf-Bucket und das Präfix von allen Datenbank-Instances gemeinsam genutzt.
- Alle relevanten Lambda-Grenzwerte. Weitere Informationen finden Sie unter Lambda quotas (Lambda-Kontingente) im AWS Lambda -Entwicklerhandbuch.
- Unter Filterbedingungen müssen Sie die `Date`- und `Timestamp`-Datentypen in den entsprechenden Datentyp umwandeln.
- So suchen Sie nach negativen Werten des Typs `Real` und `Float`, verwenden Sie den <=- oder >=-Operator.

- Die Datentypen `binary`, `varbinary`, `image` und `rowversion` werden nicht unterstützt.

Bedingungen

Die folgenden Begriffe beziehen sich auf den SQL Server-Konektor.

- Datenbank-Instance – Jede Instance einer Datenbank, die On-Premises, in Amazon EC2 oder auf Amazon RDS bereitgestellt wird.
- Handler – Ein Lambda-Handler, der auf Ihre Datenbank-Instance zugreift. Ein Handler kann für Metadaten oder für Datensätze verwendet werden.
- Metadaten-Handler – Ein Lambda-Handler, der Metadaten von Ihrer Datenbank-Instance abruft.
- Record Handler – Ein Lambda-Handler, der Datensätze aus Ihrer Datenbank-Instance abruft.
- Composite Handler – Ein Lambda-Handler, der sowohl Metadaten als auch Datensätze aus Ihrer Datenbank-Instance abruft.
- Eigenschaft oder Parameter – Eine Datenbankeigenschaft, die von Handlern zum Extrahieren von Datenbankinformationen verwendet wird. Sie konfigurieren diese Eigenschaften als Lambda-Umgebungsvariablen.
- Verbindungszeichenfolge – Eine Textzeichenfolge, die verwendet wird, um eine Verbindung zu einer Datenbank-Instance herzustellen.
- Katalog — Ein nicht bei Athena registrierter AWS Glue Katalog, der ein erforderliches Präfix für die `connection_string` Immobilie ist.
- Multiplex-Handler – Ein Lambda-Handler, der mehrere Datenbankverbindungen akzeptieren und verwenden kann.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den SQL-Server-Konnektor zu konfigurieren.

Verbindungszeichenfolge

Verwenden Sie eine JDBC-Verbindungszeichenfolge im folgenden Format, um eine Verbindung zu einer Datenbank-Instance herzustellen.

```
sqlserver://${jdbc_connection_string}
```

Verwenden eines Multiplexing-Handlers

Sie können einen Multiplexer verwenden, um mit einer einzigen Lambda-Funktion eine Verbindung zu mehreren Datenbank-Instances herzustellen. Anfragen werden anhand des Katalognamens weitergeleitet. Verwenden Sie die folgenden Klassen in Lambda.

Handler	Klasse
Composite Handler	`SqlServerMuxCompositeHandler`
Metadaten-Handler	`SqlServerMuxMetadataHandler`
Record Handler	`SqlServerMuxRecordHandler`

Multiplex-Handler-Parameter

Parameter	Beschreibung
`$`*`catalog`*`_connection_string`	Erforderlich Eine Verbindungszeichenfolge einer Datenbank-Instance. Stellen Sie der Umgebungsvariablen den Namen des in Athena verwendeten Katalogs voran. Wenn zum Beispiel der bei Athena registrierte Katalog `mysqlservercatalog` ist, dann lautet der Name der Umgebungsvariablen `mysqlservercatalog_connection_string` .
`default`	Erforderlich Die standardmäßige Verbindungszeichenfolge. Diese Zeichenfolge wird verwendet, wenn der Katalog `lambda:${`*`AWS_LAMBDA_FUNCTION_NAME`*`}` ist.

Die folgenden Beispieleigenschaften beziehen sich auf eine SqlServer MUX-Lambda-Funktion, die zwei Datenbankinstanzen unterstützt: `sqlserver1` (Standard) und. `sqlserver2`

Eigenschaft	Wert
`default`	`sqlserver://jdbc:sqlserver://sqlserver1.`*`hostname`*`:`*`port`*`;databaseName=`*`<database_name>`*`;${`*`secret1_name`*`}`

Eigenschaft	Wert
`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`*` }`

Bereitstellen von Anmeldeinformationen

Um einen Benutzernamen und ein Kennwort für Ihre Datenbank in Ihrer JDBC-Verbindungszeichenfolge anzugeben, können Sie Eigenschaften von Verbindungszeichenfolgen oder AWS Secrets Manager verwenden.

- Verbindungszeichenfolge – Ein Benutzername und ein Kennwort können als Eigenschaften in der JDBC-Verbindungszeichenfolge angegeben werden.

  > **Important**
  >
  > Als bewährte Sicherheitsmethode sollten Sie keine fest kodierten Anmeldeinformationen in Ihren Umgebungsvariablen oder Verbindungszeichenfolgen verwenden. Informationen zum Verschieben Ihrer hartcodierten Geheimnisse nach finden Sie im AWS Secrets Manager Benutzerhandbuch unter Verschieben von hartcodierten AWS Secrets Manager Geheimnissen nach.AWS Secrets Manager

- AWS Secrets Manager— Um die Athena Federated Query-Funktion verwenden zu können AWS Secrets Manager, muss die mit Ihrer Lambda-Funktion verbundene VPC über Internetzugang oder einen VPC-Endpunkt verfügen, um eine Verbindung zu Secrets Manager herzustellen.

  Sie können den Namen eines Geheimnisses in AWS Secrets Manager Ihre JDBC-Verbindungszeichenfolge eingeben. Der Konnektor ersetzt den geheimen Namen durch `username`- und `password`-Werte von Secrets Manager.

  Für Amazon RDS-Datenbank-Instances ist diese Unterstützung eng integriert. Wenn Sie Amazon RDS verwenden, empfehlen wir dringend, eine Rotation der Anmeldeinformationen zu verwenden

AWS Secrets Manager . Wenn Ihre Datenbank Amazon RDS nicht verwendet, speichern Sie die Anmeldeinformationen als JSON im folgenden Format:

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

Beispiel einer Verbindungszeichenfolge mit einem geheimen Namen

Die folgende Zeichenfolge hat den geheimen Namen `${secret_name}`.

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

Der Konnektor verwendet den geheimen Namen, um Secrets abzurufen und den Benutzernamen und das Kennwort bereitzustellen, wie im folgenden Beispiel gezeigt.

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

Verwenden eines einzelnen Verbindungs-Handlers

Sie können die folgenden Einzelverbindungsmetadaten und Record Handler verwenden, um eine Verbindung zu einer einzelnen SQL Server-Instance herzustellen.

Handler-Typ	Klasse
Composite Handler	`SqlServerCompositeHandler`
Metadaten-Handler	`SqlServerMetadataHandler`
Record Handler	`SqlServerRecordHandler`

Parameter für Einzelverbindungs-Handler

Parameter	Beschreibung
`default`	Erforderlich Die standardmäßige Verbindungszeichenfolge.

Die Einzelverbindungs-Handler unterstützen eine Datenbank-Instance und müssen einen `default`-Verbindungszeichenfolgenparameter bereitstellen. Alle anderen Verbindungszeichenfolgen werden ignoriert.

Die folgende Beispieleigenschaft gilt für eine einzelne SQL Server-Instance, die von einer Lambda-Funktion unterstützt wird.

Eigenschaft	Wert
`default`	`sqlserver://jdbc:sqlserver://` *`hostname`*`:`*`port`*`;database Name=` *`<database_name>`* `;${`*`secret_name`*` }`

Überlauf-Parameter

Das Lambda-SDK kann Daten an Amazon S3 übertragen. Alle Datenbank-Instances, auf die mit derselben Lambda-Funktion zugegriffen wird, werden an denselben Speicherort verschoben.

Parameter	Beschreibung
`spill_bucket`	Erforderlich Überlauf-Bucket-Name.
`spill_prefix`	Erforderlich Schlüssel-Prefix für den Überlauf-Bucket.
`spill_put_request_headers`	(Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Weitere mögliche Header finden Sie PutObject in der Amazon Simple Storage Service API-Referenz.

Datentypunterstützung

Die folgende Tabelle zeigt die entsprechenden Datentypen für SQL Server und Apache Arrow.

SQL Server	Arrow
Bit	TINYINT
tinyint	SMALLINT
smallint	SMALLINT
int	INT
bigint	BIGINT
Dezimalwert	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)
char[n]	VARCHAR
varchar[n/max]	VARCHAR

SQL Server	Arrow
nchar[n]	VARCHAR
nvarchar[n/max]	VARCHAR
text	VARCHAR
ntext	VARCHAR

Partitionen und Splits

Eine Partition wird durch eine einzelne Partitionsspalte vom Typ `varchar` dargestellt. Im Fall des SQL-Server-Konnektors bestimmt eine Partitionsfunktion, wie Partitionen auf die Tabelle angewendet werden. Die Informationen zur Partitionsfunktion und zum Spaltennamen werden aus der SQL Server-Metadatentabelle abgerufen. Eine benutzerdefinierte Abfrage ruft dann die Partition ab. Splits werden basierend auf der Anzahl der empfangenen unterschiedlichen Partitionen erstellt.

Leistung

Die Auswahl einer Teilmenge von Spalten beschleunigt die Abfragelaufzeit erheblich und reduziert die gescannten Daten. Der SQL-Server-Konnektor ist aufgrund der Gleichzeitigkeit widerstandsfähig gegenüber Drosselung.

Der Athena-SQL-Server-Konnektor führt einen Prädikat-Pushdown durch, um die von der Abfrage durchsuchten Daten zu verringern. Einfache Prädikate und komplexe Ausdrücke werden an den Konnektor übertragen, um die Menge der gescannten Daten zu reduzieren und die Laufzeit der Abfrageausführung zu verkürzen.

Prädikate

Ein Prädikat ist ein Ausdruck in der `WHERE`-Klausel einer SQL-Abfrage, der einen booleschen Wert ergibt und Zeilen auf der Grundlage mehrerer Bedingungen filtert. Der Athena-SQL-Server-Konnektor kann diese Ausdrücke kombinieren und sie direkt an SQL Server weiterleiten, um die Funktionalität zu erweitern und die Menge der gescannten Daten zu reduzieren.

Die folgenden Athena-SQL-Server-Konnektor-Operatoren unterstützen Prädikat-Pushdown:

- Boolean: UND, ODER, NICHT

- Gleichheit: GLEICH, NICHT-GLEICH, WENIGER_ALS, WENIGER_ODER-GLEICH, GRÖSSER_ALS, GRÖSSER_ODER-GLEICH, IST_UNTERSCHIEDEN VON, NULL_WENN, IST_NULL
- Arithmetik: ADDIEREN, SUBTRAHIEREN, MULTIPLIZIEREN, DIVIDIEREN, MODULIEREN, NEGIEREN
- Andere: WIE_MUSTER, IN

Beispiel für einen kombinierten Pushdown

Kombinieren Sie für erweiterte Abfragefunktionen die Pushdown-Typen wie im folgenden Beispiel:

```
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%');
```

Lizenzinformationen

Durch die Verwendung dieses Connectors erkennen Sie die Einbindung von Komponenten von Drittanbietern an. Eine Liste dieser Komponenten finden Sie in der Datei pom.xml für diesen Connector und stimmen den Bedingungen der jeweiligen Drittanbieterlizenzen zu, die in der Datei LICENSE.txt auf GitHub .com enthalten sind.

Weitere Informationen finden Sie auch unter

Die neuesten Informationen zur JDBC-Treiberversion finden Sie in der Datei pom.xml für den SQL Server-Connector auf GitHub .com.

Weitere Informationen zu diesem Connector finden Sie auf GitHubder entsprechenden Website unter .com.

Amazon Athena Teradata Konnektor

Der Amazon-Athena-Konnektor für Teradata ermöglicht es Athena, SQL-Abfragen für Daten auszuführen, die in Ihren Teradata-Datenbanken gespeichert sind.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor

bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Einschränkungen

- Schreiboperationen wie DDL werden nicht unterstützt.
- In einem Multiplexer-Setup werden der Überlauf-Bucket und das Präfix von allen Datenbank-Instances gemeinsam genutzt.
- Alle relevanten Lambda-Grenzwerte. Weitere Informationen finden Sie unter Lambda quotas (Lambda-Kontingente) im AWS Lambda -Entwicklerhandbuch.

Bedingungen

Die folgenden Begriffe beziehen sich auf den Teradata-Konnektor.

- Datenbank-Instance – Jede Instance einer Datenbank, die On-Premises, in Amazon EC2 oder auf Amazon RDS bereitgestellt wird.
- Handler – Ein Lambda-Handler, der auf Ihre Datenbank-Instance zugreift. Ein Handler kann für Metadaten oder für Datensätze verwendet werden.
- Metadaten-Handler – Ein Lambda-Handler, der Metadaten von Ihrer Datenbank-Instance abruft.
- Record Handler – Ein Lambda-Handler, der Datensätze aus Ihrer Datenbank-Instance abruft.
- Composite Handler – Ein Lambda-Handler, der sowohl Metadaten als auch Datensätze aus Ihrer Datenbank-Instance abruft.
- Eigenschaft oder Parameter – Eine Datenbankeigenschaft, die von Handlern zum Extrahieren von Datenbankinformationen verwendet wird. Sie konfigurieren diese Eigenschaften als Lambda-Umgebungsvariablen.
- Verbindungszeichenfolge – Eine Textzeichenfolge, die verwendet wird, um eine Verbindung zu einer Datenbank-Instance herzustellen.
- Katalog — Ein nicht bei Athena registrierter AWS Glue Katalog, der ein erforderliches Präfix für die `connection_string` Immobilie ist.
- Multiplex-Handler – Ein Lambda-Handler, der mehrere Datenbankverbindungen akzeptieren und verwenden kann.

Voraussetzung für Lambda-Ebenen

Um den Teradata-Konnektor mit Athena zu verwenden, müssen Sie eine Lambda-Ebene erstellen, die den Teradata-JDBC-Treiber enthält. Eine Lambda-Ebene ist ein `.zip`-Dateiarchiv, das zusätzlichen Code für eine Lambda-Funktion enthält. Wenn Sie den Teradata-Konnektor für Ihr Konto bereitstellen, geben Sie den ARN der Ebene an. Dadurch wird die Lambda-Ebene mit dem Teradata-JDBC-Treiber an den Teradata-Konnektor angeschlossen, damit Sie ihn mit Athena verwenden können.

Weitere Informationen zu Lambda-Ebenen finden Sie unter Erstellen und Freigeben von Lambda-Ebenen im Entwicklerhandbuch für AWS Lambda .

So erstellen Sie eine Lambda-Ebene für den Teradata-Konnektor

1. Navigieren Sie zur Downloadseite des Teradata-JDBC-Treibers unter https://downloads.teradata.com/download/connectivity/jdbc-driver.
2. Laden Sie den Teradata-JDBC-Treiber herunter. Auf der Website müssen Sie ein Konto erstellen und eine Lizenzvereinbarung zum Herunterladen der Datei akzeptieren.
3. Extrahieren Sie die `terajdbc4.jar`-Datei aus der Archivdatei, die Sie heruntergeladen haben.
4. Erstellen Sie die folgende Ordnerstruktur und platzieren Sie die `.jar`-Datei drin.

   ```
   java\lib\terajdbc4.jar
   ```
5. Erstellen Sie eine `.zip`-Datei der gesamten Ordnerstruktur, die die `terajdbc4.jar`-Datei enthält.
6. Melden Sie sich bei der an AWS Management Console und öffnen Sie die AWS Lambda Konsole unter https://console.aws.amazon.com/lambda/.
7. Wählen Sie im Navigationsbereich Layers (Ebenen) und dann Create layer (Ebene erstellen) aus.
8. Geben Sie für Name einen Namen für Ihre Ebene ein (z. B.: `TeradataJava11LambdaLayer`).
9. Stellen Sie sicher, dass die Option Upload a .zip file (eine ZIP-Datei hochladen) ausgewählt ist.
10. Wählen Sie Upload (Hochladen) und laden Sie dann den gezippten Ordner hoch, der den Teradata-JDBC-Treiber enthält.
11. Wählen Sie Erstellen.
12. Kopieren Sie auf der Detailseite für die Ebene den Ebenen-ARN, indem Sie oben auf der Seite das Zwischenablagesymbol auswählen.
13. Speichern Sie den ARN als Referenz.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den Teradata-Konnektor zu konfigurieren.

Verbindungszeichenfolge

Verwenden Sie eine JDBC-Verbindungszeichenfolge im folgenden Format, um eine Verbindung zu einer Datenbank-Instance herzustellen.

```
teradata://${jdbc_connection_string}
```

Verwenden eines Multiplexing-Handlers

Sie können einen Multiplexer verwenden, um mit einer einzigen Lambda-Funktion eine Verbindung zu mehreren Datenbank-Instances herzustellen. Anfragen werden anhand des Katalognamens weitergeleitet. Verwenden Sie die folgenden Klassen in Lambda.

Handler	Klasse
Composite Handler	`TeradataMuxCompositeHandler`
Metadaten-Handler	`TeradataMuxMetadataHandler`
Record Handler	`TeradataMuxRecordHandler`

Multiplex-Handler-Parameter

Parameter	Beschreibung
$*catalog*_connection_string	Erforderlich Eine Verbindungszeichenfolge einer Datenbank-Instance. Stellen Sie der Umgebungsvariablen den Namen des in Athena verwendeten Katalogs voran. Wenn zum Beispiel der bei Athena registrierte Katalog `myteradatacatalog` ist, dann lautet der Name der Umgebungsvariablen `myteradatacatalog_connection_string` .

Parameter	Beschreibung
`default`	Erforderlich Die standardmäßige Verbindungszeichenfolge. Diese Zeichenfolge wird verwendet, wenn der Katalog `lambda:${` *AWS_LAMBDA_FUNCTION_NAME* `}` ist.

Die folgenden Beispieleigenschaften gelten für eine Teradata MUX Lambda-Funktion, die zwei Datenbank-Instances unterstützt: `teradata1` (die Standardeinstellung) und `teradata2`.

Eigenschaft	Wert
`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`

Bereitstellen von Anmeldeinformationen

Um einen Benutzernamen und ein Kennwort für Ihre Datenbank in Ihrer JDBC-Verbindungszeichenfolge anzugeben, können Sie Eigenschaften von Verbindungszeichenfolgen oder AWS Secrets Manager verwenden.

- Verbindungszeichenfolge – Ein Benutzername und ein Kennwort können als Eigenschaften in der JDBC-Verbindungszeichenfolge angegeben werden.

> **Important**
>
> Als bewährte Sicherheitsmethode sollten Sie keine fest kodierten Anmeldeinformationen in Ihren Umgebungsvariablen oder Verbindungszeichenfolgen verwenden. Informationen zum Verschieben Ihrer hartcodierten Geheimnisse nach AWS Secrets Manager finden Sie AWS Secrets Manager im AWS Secrets Manager Benutzerhandbuch unter Verschieben von hartcodierten Geheimnissen nach.

- AWS Secrets Manager— Um die Athena Federated Query-Funktion verwenden zu können AWS Secrets Manager, muss die mit Ihrer Lambda-Funktion verbundene VPC über Internetzugang oder einen VPC-Endpunkt verfügen, um eine Verbindung zu Secrets Manager herzustellen.

  Sie können den Namen eines Geheimnisses in AWS Secrets Manager Ihre JDBC-Verbindungszeichenfolge eingeben. Der Konnektor ersetzt den geheimen Namen durch `username`- und `password`-Werte von Secrets Manager.

  Für Amazon RDS-Datenbank-Instances ist diese Unterstützung eng integriert. Wenn Sie Amazon RDS verwenden, empfehlen wir dringend, eine Rotation der Anmeldeinformationen zu verwenden AWS Secrets Manager . Wenn Ihre Datenbank Amazon RDS nicht verwendet, speichern Sie die Anmeldeinformationen als JSON im folgenden Format:

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

Beispiel einer Verbindungszeichenfolge mit einem geheimen Namen

Die folgende Zeichenfolge hat den geheimen Namen `${Test/RDS/Teradata1}`.

```
teradata://jdbc:teradata1.host/TMODE=ANSI,CHARSET=UTF8,DATABASE=TEST,${Test/RDS/
Teradata1}&...
```

Der Konnektor verwendet den geheimen Namen, um Secrets abzurufen und den Benutzernamen und das Kennwort bereitzustellen, wie im folgenden Beispiel gezeigt.

```
teradata://jdbc:teradata://teradata1.host/
TMODE=ANSI,CHARSET=UTF8,DATABASE=TEST,...&user=sample2&password=sample2&...
```

Derzeit erkennt Teradata die `user`- und `password`-JDBC-Eigenschaften. Es akzeptiert auch den Benutzernamen und das Passwort im Format *`Nutzername`*`/`*`Passwort`* ohne die Schlüssel `user` oder `password`.

Verwenden eines einzelnen Verbindungs-Handlers

Sie können die folgenden Einzelverbindungsmetadaten und Record Handler verwenden, um eine Verbindung zu einer einzelnen Teradata-Instance herzustellen.

Handler-Typ	Klasse
Composite Handler	`TeradataCompositeHandler`
Metadaten-Handler	`TeradataMetadataHandler`
Record Handler	`TeradataRecordHandler`

Parameter für Einzelverbindungs-Handler

Parameter	Beschreibung
`default`	Erforderlich Die standardmäßige Verbindungszeichenfolge.

Die Einzelverbindungs-Handler unterstützen eine Datenbank-Instance und müssen einen `default`-Verbindungszeichenfolgenparameter bereitstellen. Alle anderen Verbindungszeichenfolgen werden ignoriert.

Die folgende Beispieleigenschaft gilt für eine einzelne Teradata-Instance, die von einer Lambda-Funktion unterstützt wird.

Eigenschaft	Wert
`default`	`teradata://jdbc:teradata://teradata1.host/TMODE=ANSI,CHARSET=UTF8,DATABASE=TEST,secret=Test/RDS/Teradata1`

Überlauf-Parameter

Das Lambda-SDK kann Daten an Amazon S3 übertragen. Alle Datenbank-Instances, auf die mit derselben Lambda-Funktion zugegriffen wird, werden an denselben Speicherort verschoben.

Parameter	Beschreibung
`spill_bucket`	Erforderlich Überlauf-Bucket-Name.
`spill_prefix`	Erforderlich Schlüssel-Prefix für den Überlauf-Bucket.
`spill_put_request_headers`	(Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Weitere mögliche Header finden Sie PutObject in der Amazon Simple Storage Service API-Referenz.

Datentypunterstützung

Die folgende Tabelle zeigt die entsprechenden Datentypen für JDBC und Apache Arrow.

JDBC	Arrow
Boolesch	Bit
Ganzzahl	Tiny
Short	Smallint
Ganzzahl	Int
Long	Bigint
float	Float4
Double	Float8
Datum	DateDay

JDBC	Arrow
Zeitstempel	DateMilli
String	Varchar
Bytes	Varbinary
BigDecimal	Dezimal
ARRAY	Auflisten

Partitionen und Splits

Eine Partition wird durch eine einzelne Partitionsspalte vom Typ `Integer` dargestellt. Die Spalte enthält Partitionsnamen der Partitionen, die in einer Teradata-Tabelle definiert sind. Für eine Tabelle, die keine Partitionsnamen hat, wird * ausgegeben, was einer einzelnen Partition entspricht. Eine Partition entspricht einem Split.

Name	Typ	Beschreibung
Partition	Ganzzahl	Benannte Partition in Teradata.

Leistung

Teradata unterstützt native Partitionen. Der Athena-Teradata-Konnektor kann Daten von diesen Partitionen parallel abrufen. Wenn Sie sehr große Datenmengen mit einheitlicher Partitionsverteilung abfragen möchten, wird eine native Partitionierung dringend empfohlen. Die Auswahl einer Teilmenge von Spalten verlangsamt die Abfragelaufzeit erheblich. Der Konnektor zeigt aufgrund der Gleichzeitigkeit eine gewisse Drosselung.

Der Athena-Teradata-Konnektor führt einen Prädikat-Pushdown durch, um die von der Abfrage durchsuchten Daten zu reduzieren. Einfache Prädikate und komplexe Ausdrücke werden an den Konnektor übertragen, um die Menge der gescannten Daten zu reduzieren und die Laufzeit der Abfrageausführung zu verkürzen.

Prädikate

Ein Prädikat ist ein Ausdruck in der `WHERE`-Klausel einer SQL-Abfrage, der einen booleschen Wert ergibt und Zeilen auf der Grundlage mehrerer Bedingungen filtert. Der Athena-Teradata-Konnektor kann diese Ausdrücke kombinieren und sie direkt an Teradata weiterleiten, um die Funktionalität zu erweitern und die Menge der gescannten Daten zu reduzieren.

Die folgenden Athena-Teradata-Konnektor-Operatoren unterstützen Prädikat-Pushdown:

- Boolean: UND, ODER, NICHT
- Gleichheit: GLEICH, NICHT-GLEICH, WENIGER_ALS, WENIGER_ODER_GLEICH, GRÖSSER_ALS, GRÖSSER_ODER_GLEICH, NULL_WENN, IST_NULL
- Arithmetik: ADDIEREN, SUBTRAHIEREN, MULTIPLIZIEREN, DIVIDIEREN, MODULIEREN, NEGIEREN
- Andere: WIE_MUSTER, IN

Beispiel für einen kombinierten Pushdown

Kombinieren Sie für erweiterte Abfragefunktionen die Pushdown-Typen wie im folgenden Beispiel:

```
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%');
```

Lizenzinformationen

Durch die Verwendung dieses Connectors erkennen Sie die Einbindung von Komponenten von Drittanbietern an. Eine Liste dieser Komponenten finden Sie in der Datei pom.xml für diesen Connector und stimmen den Bedingungen der jeweiligen Drittanbieterlizenzen zu, die in der Datei LICENSE.txt auf GitHub .com enthalten sind.

Weitere Informationen finden Sie auch unter

Die neuesten Informationen zur JDBC-Treiberversion finden Sie in der Datei pom.xml für den Teradata-Connector auf .com. GitHub

Weitere Informationen zu diesem Connector finden Sie auf der entsprechenden Website unter .com. GitHub

Amazon Athena Timestream Konnektor

Der Amazon-Athena-Timestream-Konnektor ermöglicht Amazon Athena die Kommunikation mit Amazon Timestream, wodurch auf Ihre Zeitreihendaten über Amazon Athena zugegriffen werden kann. Sie können AWS Glue Data Catalog optional als Quelle für ergänzende Metadaten verwenden.

Amazon Timestream ist eine schnelle, skalierbare, vollständig verwaltete, speziell entwickelte Zeitreihendatenbank, mit der Sie mühelos Milliarden von Zeitreihendatenpunkten pro Tag speichern und analysieren können. Timestream spart Ihnen Zeit und Kosten bei der Verwaltung des Lebenszyklus von Zeitreihendaten, indem aktuelle Daten im Arbeitsspeicher aufbewahrt und historische Daten basierend auf benutzerdefinierten Richtlinien in eine kostenoptimierte Speicherebene verschoben werden.

Wenn Sie Lake Formation in Ihrem Konto aktiviert haben, muss die IAM-Rolle für Ihren Athena-Verbund-Lambda-Konnektor, den Sie im AWS Serverless Application Repository bereitstellen, in Lake Formation über Lesezugriff auf das AWS Glue Data Catalog verfügen.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den Timestream-Konnektor zu konfigurieren.

- spill_bucket – Gibt den Amazon S3-Bucket für Daten an, die die Lambda-Funktionsgrenzen überschreiten.
- spill_prefix – (Optional) Ist standardmäßig ein Unterordner im angegebenen `spill_bucket` genannt `athena-federation-spill`. Wir empfehlen Ihnen, einen Amazon-S3-Speicher-Lebenszyklus an dieser Stelle zu konfigurieren, um die Überlaufe zu löschen, die älter als eine festgelegte Anzahl von Tagen oder Stunden sind.
- spill_put_request_headers – (Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Andere mögliche Header finden Sie unter PutObject in der API-Referenz zu Amazon Simple Storage Service.

- kms_key_id – (Optional) Standardmäßig werden alle Daten, die an Amazon S3 gesendet werden, mit dem AES-GCM-authentifizierten Verschlüsselungsmodus und einem zufällig generierten Schlüssel verschlüsselt. Damit Ihre Lambda-Funktion stärkere Verschlüsselungsschlüssel verwendet, die von KMS generiert werden, wie`a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, können Sie eine ID einer Verschlüsselung angeben.
- disable_spill_encryption – (Optional) Bei Einstellung auf `True`, wird die Spill-Verschlüsselung deaktiviert. Die Standardeinstellung ist `False`, sodass Daten, die an S3 übertrragen werden, mit AES-GCM verschlüsselt werden - entweder mit einem zufällig generierten Schlüssel oder mit KMS zum Generieren von Schlüsseln. Das Deaktivieren der Überlauf-Verschlüsselung kann die Leistung verbessern, insbesondere wenn Ihr Überlauf-Standort eine serverseitige Verschlüsselung verwendet.
- glue_catalog – (Optional) Verwenden Sie diese Option, um einen kontoübergreifenden AWS Glue-Katalog anzugeben. Standardmäßig versucht der Konnektor, Metadaten von seinem eigenen AWS Glue-Konto abzurufen.

Einrichten von Datenbanken und Tabellen in AWS Glue

Sie können AWS Glue Data Catalog optional als Quelle für ergänzende Metadaten verwenden. Für die Aktivierung einer AWS Glue-Tabelle für die Verwendung mit Timestream benötigen Sie eine AWS Glue-Datenbank und -Tabelle mit Namen, die mit der Timestream-Datenbank und der -Tabelle übereinstimmen, für die Sie zusätzliche Metadaten bereitstellen möchten.

> **Note**
>
> Verwenden Sie für Ihre Datenbanknamen und -tabellen nur Kleinbuchstaben, um eine optimale Leistung zu erzielen. Bei der Verwendung gemischter Groß- und Kleinschreibung führt der Konnektor eine Suche ohne Berücksichtigung der Groß- und Kleinschreibung durch, die rechenintensiver ist.

Zur Konfigurierung einer AWS Glue-Tabelle für die Verwendung mit Timestream müssen Sie ihre Tabelleneigenschaften in AWS Glue einrichten.

Verwenden Sie eine AWS Glue-Tabelle für ergänzende Metadaten wie folgt

1. Bearbeiten Sie die Tabelle in der AWS Glue-Konsole, um die folgenden Tabelleneigenschaften hinzuzufügen:

- timestream-metadata-flag – Diese Eigenschaft teilt dem Timestream-Konnektor mit, dass der Konnektor die Tabelle für ergänzende Metadaten verwenden kann. Sie können einen beliebigen Wert für `timestream-metadata-flag` angeben, solange die `timestream-metadata-flag`-Eigenschaft in der Liste der Tabelleneigenschaften vorhanden ist.
- _view_template – Wenn Sie AWS Glue für ergänzende Metadaten verwenden, können Sie diese Tabelleneigenschaft verwenden und eine beliebige Timestream-SQL als Ansicht angeben. Der Athena-Timestream-Konnektor verwendet das SQL aus der Ansicht zusammen mit Ihrem SQL von Athena, um Ihre Abfrage auszuführen. Dies ist nützlich, wenn Sie ein Feature von Timestream SQL verwenden möchten, die sonst in Athena nicht verfügbar ist.

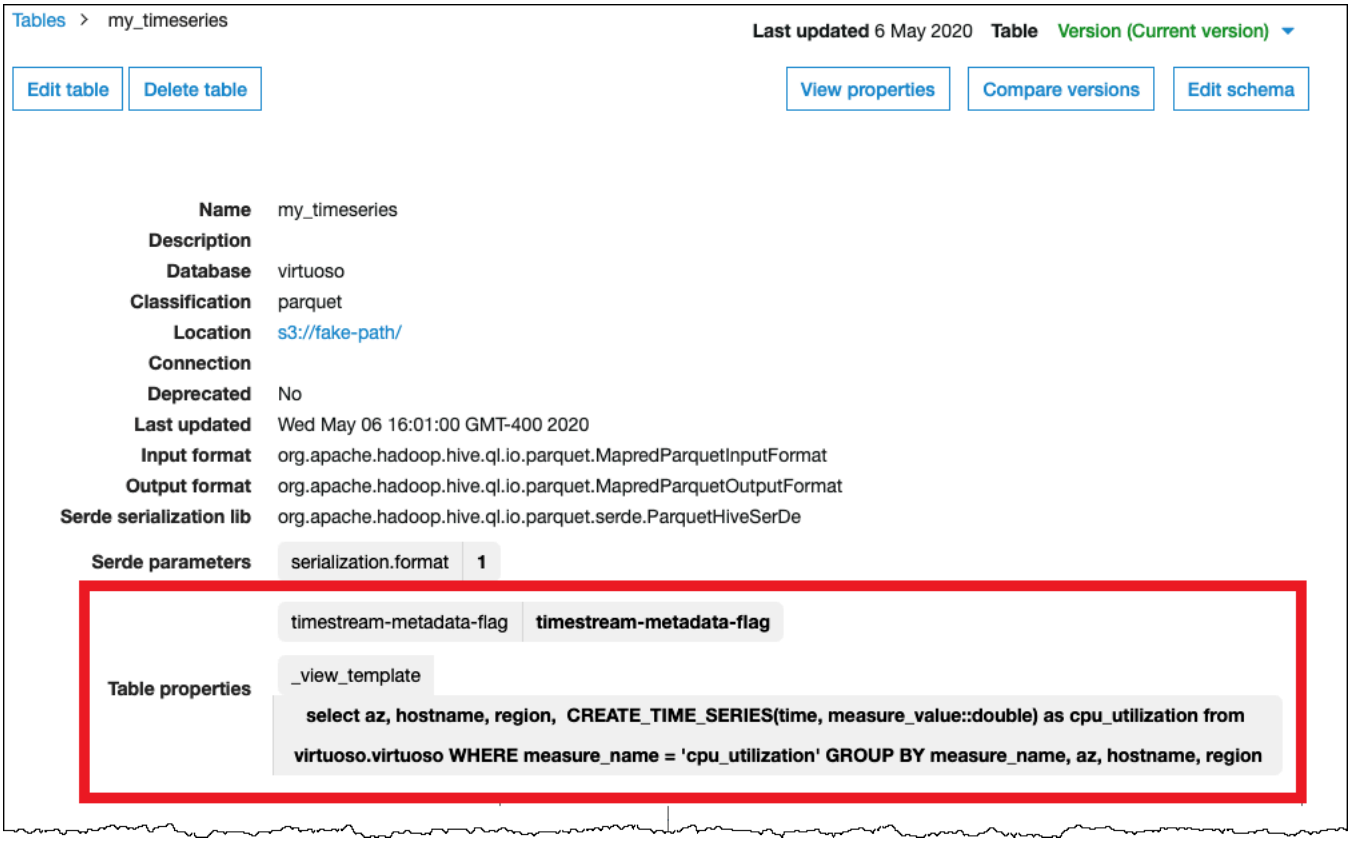
2. Achten Sie darauf, Datentypen zu verwenden, die für AWS Glue geeignet sind, wie in diesem Dokument aufgeführt.

Datentypen

Derzeit unterstützt der Timestream-Konnektor nur eine Teilmenge der in Timestream verfügbaren Datentypen, insbesondere: die Skalarwerte `varchar`, `double` und `timestamp`.

Um den `timeseries`-Datentyp abzufragen, müssen Sie eine Ansicht in AWS Glue-Tabelleneigenschaften konfigurieren, die die `CREATE_TIME_SERIES`-Funktion von Timestream verwenden. Sie müssen auch ein Schema für die Ansicht bereitstellen, das die Syntax `ARRAY<STRUCT<time:timestamp,measure_value::double:double>>` als Typ für eine Ihrer Zeitreihenspalten verwendet. Achten Sie darauf, dass Sie `double` mit dem entsprechenden Skalartyp für Ihre Tabelle ersetzen.

Das folgende Image zeigt ein Beispiel für die AWS Glue-Tabelleneigenschaften, die zum Einrichten einer Ansicht über eine Zeitreihe konfiguriert sind.



Erforderliche Berechtigungen

Ausführliche Informationen zu den für diesen Konnektor erforderlichen IAM-Richtlinien finden Sie im `Policies`-Abschnitt der athena-timestream.yaml-Datei. In der folgenden Liste sind die erforderlichen Berechtigungen zusammengefasst.

- Amazon-S3-Schreibzugriff – Der Konnektor benötigt Schreibzugriff auf einen Speicherort in Amazon S3, um Ergebnisse aus großen Abfragen zu übertragen.
- Athena GetQueryExecution – Der Konnektor verwendet diese Berechtigung, um ein Fast-Fail durchzuführen, wenn die vorgeschaltete Athena-Abfrage beendet wurde.
- AWS Glue Data Catalog – Der Timestream-Konnektor benötigt schreibgeschützten Zugriff auf AWS Glue Data Catalog, um Schemainformationen zu erhalten.
- CloudWatch Logs – Der Konnektor benötigt Zugriff auf CloudWatch Logs zum Speichern von Protokollen.
- Timestream Access – Zum Ausführen von Timestream-Abfragen.

Leistung

Wir empfehlen die `LIMIT`-Klausel zu verwenden, um die zurückgegebenen Daten (nicht die gescannten Daten) auf weniger als 256 MB zu begrenzen, um sicherzustellen, dass interaktive Abfragen leistungsfähig sind.

Der Athena-Timestream-Konnektor führt ein Prädikat-Pushdown durch, um die von der Abfrage gescannten Daten zu reduzieren. `LIMIT`-Klauseln reduzieren die Menge der gescannten Daten, aber wenn Sie kein Prädikat angeben, sollten Sie davon ausgehen, dass `SELECT`-Abfragen mit einer `LIMIT`-Klausel mindestens 16 MB Daten scannen. Die Auswahl einer Teilmenge von Spalten beschleunigt die Abfragelaufzeit erheblich und reduziert die gescannten Daten. Der Timestream-Konnektor ist aufgrund der Gleichzeitigkeit widerstandsfähig gegenüber Drosselung.

Lizenzinformationen

Das Timestream-Konnektor-Projekt von Amazon Athena ist unter der Apache-2.0-Lizenz lizenziert.

Weitere Informationen finden Sie auch unter

Weitere Informationen zu diesem Konnektor finden Sie unter der entsprechenden Seite auf GitHub.com.

Amazon Athena TPC Benchmark DS (TPC-DS) Konnektor

Der Amazon-Athena-TPC-DS-Konnektor ermöglicht Amazon Athena die Kommunikation mit einer Quelle zufällig erzeugter TPC-Benchmark-DS-Daten für Benchmarking und Funktionstests von Athena Federation. Der Athena-TPC-DS-Konnektor generiert eine TPC-DS-konforme Datenbank mit einem von vier Skalierungsfaktoren. Wir empfehlen die Verwendung dieses Konnektors nicht als Alternative zu Amazon-S3-basierten Data Lake-Leistungstests.

Voraussetzungen

- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.

Parameter

Verwenden Sie die Lambda-Umgebungsvariablen in diesem Abschnitt, um den TPC-DS-Konnektor zu konfigurieren.

- spill_bucket – Gibt den Amazon S3-Bucket für Daten an, die die Lambda-Funktionsgrenzen überschreiten.
- spill_prefix – (Optional) Ist standardmäßig ein Unterordner im angegebenen `spill_bucket` genannt `athena-federation-spill`. Wir empfehlen Ihnen, einen Amazon-S3-Speicher-Lebenszyklus an dieser Stelle zu konfigurieren, um die Überlaufe zu löschen, die älter als eine festgelegte Anzahl von Tagen oder Stunden sind.
- spill_put_request_headers – (Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Andere mögliche Header finden Sie unter PutObject in der API-Referenz zu Amazon Simple Storage Service.
- kms_key_id – (Optional) Standardmäßig werden alle Daten, die an Amazon S3 gesendet werden, mit dem AES-GCM-authentifizierten Verschlüsselungsmodus und einem zufällig generierten Schlüssel verschlüsselt. Damit Ihre Lambda-Funktion stärkere Verschlüsselungsschlüssel verwendet, die von KMS generiert werden, wie`a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, können Sie eine ID einer Verschlüsselung angeben.
- disable_spill_encryption – (Optional) Bei Einstellung auf `True`, wird die Spill-Verschlüsselung deaktiviert. Die Standardeinstellung ist `False`, sodass Daten, die an S3 übertrragen werden, mit AES-GCM verschlüsselt werden - entweder mit einem zufällig generierten Schlüssel oder mit KMS zum Generieren von Schlüsseln. Das Deaktivieren der Überlauf-Verschlüsselung kann die Leistung verbessern, insbesondere wenn Ihr Überlauf-Standort eine serverseitige Verschlüsselung verwendet.

Testen von Datenbanken und Tabellen

Der Athena TPC-DS-Konnektor generiert eine TPC-DS-konforme Datenbank mit einem der vier Skalierungsfaktoren `tpcds1`, `tpcds10`, `tpcds100`, `tpcds250` oder `tpcds1000`.

Zusammenfassung der Tabellen

Eine vollständige Liste der Testdatentabellen und -spalten erhalten Sie, wenn Sie `SHOW TABLES`- oder `DESCRIBE TABLE`-Abfragen ausführen. Die folgende Zusammenfassung der Tabellen dient der Übersichtlichkeit.

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. Bestand
13. item
14. promotion
15. Grund
16. ship_mode
17. Store
18. store_returns
19. store_sales
20. time_dim
21. warehouse
22. web_page
23. web_returns
24. web_sales
25. Web_site

Informationen zu TPC-DS-Abfragen, die mit diesem generierten Schema und den Daten kompatibel sind, finden Sie im athena-tpcds/src/main/resources/queries/-Verzeichnis auf GitHub.

Beispielabfrage

Die folgenden `SELECT`-Abfragebeispiele fragen den `tpcds`-Katalog für demografische Kundendaten in bestimmten Landkreisen ab.

```
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
```

Erforderliche Berechtigungen

Ausführliche Informationen zu den für diesen Konnektor erforderlichen IAM-Richtlinien finden Sie im `Policies`-Abschnitt der athena-tpcds.yaml-Datei. In der folgenden Liste sind die erforderlichen Berechtigungen zusammengefasst.

- Amazon-S3-Schreibzugriff – Der Konnektor benötigt Schreibzugriff auf einen Speicherort in Amazon S3, um Ergebnisse aus großen Abfragen zu übertragen.
- Athena GetQueryExecution – Der Konnektor verwendet diese Berechtigung, um ein Fast-Fail durchzuführen, wenn die vorgeschaltete Athena-Abfrage beendet wurde.

Leistung

Der Athena-TPC-DS-Konnektor versucht, Abfragen basierend auf dem von Ihnen gewählten Skalierungsfaktor zu parallelisieren. Der Prädikat-Pushdown wird innerhalb der Lambda-Funktion ausgeführt.

Lizenzinformationen

Das TPC-DS-Konnektor-Projekt von Amazon Athena ist unter der Apache-2.0-Lizenz lizenziert.

Weitere Informationen finden Sie auch unter

Weitere Informationen zu diesem Konnektor finden Sie unter der entsprechenden Seite auf GitHub.com.

Amazon Athena Vertica Konnektor

Vertica ist eine spaltenbasierte Datenbankplattform, die in der Cloud oder On-Premises bereitgestellt werden kann, die Data Warehouses im Exabyte-Maßstab unterstützt. Sie können den Amazon-Athena-Vertica-Konnektor in Verbundabfragen verwenden, um Vertica-Datenquellen von Athena abzufragen. Sie können beispielsweise analytische Abfragen über ein Data Warehouse in Vertica und einen Data Lake in Amazon S3 ausführen.

Voraussetzungen

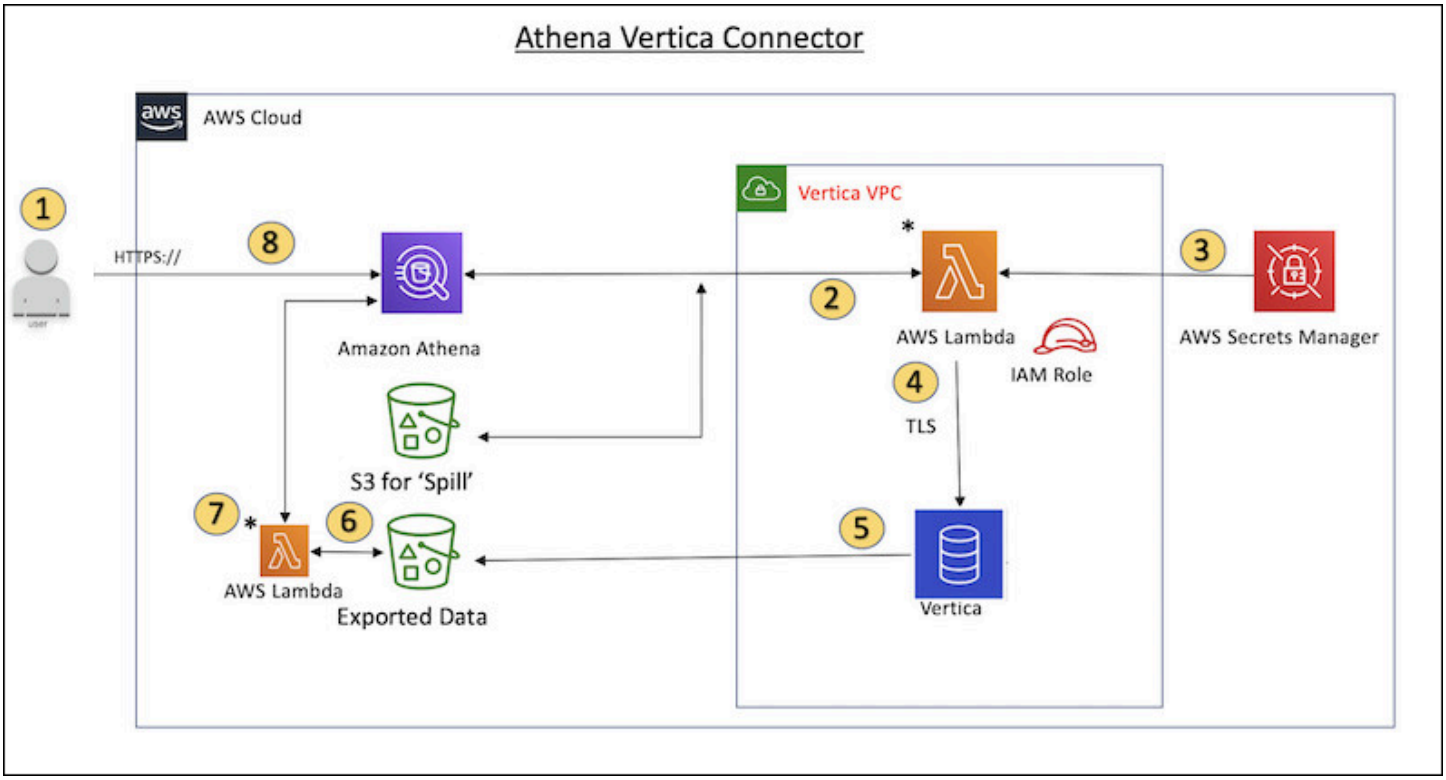
- Stellen Sie den Konnektor für Ihr AWS-Konto mithilfe der Athena-Konsole oder AWS Serverless Application Repository bereit. Weitere Informationen finden Sie unter Datenquellen-Konnektor bereitstellen oder Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository.
- Richten Sie eine VPC und eine Sicherheitsgruppe ein, bevor Sie diesen Konnektor verwenden. Weitere Informationen finden Sie unter Erstellen einer VPC für einen Datenquellen-Connector.

Einschränkungen

- Weil der Athena-Vertica-Konnektor Amazon S3 Select verwendet, um Parquet-Dateien aus Amazon S3 zu lesen, kann die Leistung des Konnektors langsam sein. Wenn Sie große Tabellen abfragen, empfehlen wir, dass Sie eine CREATE TABLE AS (SELECT ...)-Abfrage und SQL-Prädikate verwenden.
- Aufgrund eines bekannten Problems in Athena Federated Query veranlasst der Konnektor derzeit, dass Vertica alle Spalten der abgefragten Tabelle nach Amazon S3 exportiert, aber nur die abgefragten Spalten sind in den Ergebnissen auf der Athena-Konsole sichtbar.
- Schreiboperationen wie DDL werden nicht unterstützt.
- Alle relevanten Lambda-Grenzwerte. Weitere Informationen finden Sie unter Lambda quotas (Lambda-Kontingente) im AWS Lambda-Entwicklerhandbuch.

Workflow

Das folgende Diagramm zeigt den Arbeitsablauf einer Abfrage, die den Vertica-Konnektor verwendet.



1. Eine SQL-Abfrage wird für eine oder mehrere Tabellen in Vertica ausgegeben.
2. Der Konnektor analysiert die SQL-Abfrage, um den entsprechenden Teil über die JDBC-Verbindung an Vertica zu senden.
3. Die Verbindungszeichenfolgen verwenden den Benutzernamen und das Passwort, die in AWS Secrets Manager gespeichert sind, um Zugang zu Vertica zu erhalten.
4. Der Konnektor umschließt die SQL-Abfrage mit einem Vertica-EXPORT-Befehl, wie im folgenden Beispiel.

```
EXPORT TO PARQUET (directory = 's3://bucket_name/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 verarbeitet die SQL-Abfrage und sendet die Ergebnismenge an einen Amazon-S3-Bucket. Für einen besseren Durchsatz verwendet Vertica die `EXPORT`-Option, um den Schreibvorgang mehrerer Parquet-Dateien zu parallelisieren.
6. Athena durchsucht den Amazon-S3-Bucket, um die Anzahl der Dateien zu ermitteln, die für die Ergebnismenge gelesen werden sollen.
7. Athena ruft die Lambda-Funktion mehrfach auf und verwendet Amazon S3 Select, um die Parquet-Dateien aus der Ergebnismenge zu lesen. Durch mehrere Aufrufe kann Athena das Lesen der Amazon-S3-Dateien parallelisieren und einen Durchsatz von bis zu 100 GB pro Sekunde erreichen.
8. Athena verarbeitet die von Vertica zurückgegebenen Daten mit aus dem Data Lake gescannten Daten und gibt das Ergebnis zurück.

Bedingungen

Die folgenden Begriffe beziehen sich auf den Vertica-Konnektor.

- Datenbank-Instance – Jede Instance einer Vertica-Datenbank, die auf Amazon EC2 bereitgestellt wird.
- Handler – Ein Lambda-Handler, der auf Ihre Datenbank-Instance zugreift. Ein Handler kann für Metadaten oder für Datensätze verwendet werden.
- Metadaten-Handler – Ein Lambda-Handler, der Metadaten von Ihrer Datenbank-Instance abruft.
- Record Handler – Ein Lambda-Handler, der Datensätze aus Ihrer Datenbank-Instance abruft.
- Composite Handler – Ein Lambda-Handler, der sowohl Metadaten als auch Datensätze aus Ihrer Datenbank-Instance abruft.
- Eigenschaft oder Parameter – Eine Datenbankeigenschaft, die von Handlern zum Extrahieren von Datenbankinformationen verwendet wird. Sie konfigurieren diese Eigenschaften als Lambda-Umgebungsvariablen.
- Verbindungszeichenfolge – Eine Textzeichenfolge, die verwendet wird, um eine Verbindung zu einer Datenbank-Instance herzustellen.
- Katalog – Ein Nicht-AWS Glue-Katalog, der bei Athena registriert ist und ein erforderliches Präfix für die `connection_string`-Eigenschaft darstellt.

Parameter

Der Amazon Athena Vertica Konnektor stellt Konfigurationsoptionen über Lambda-Umgebungsvariablen zur Verfügung. Sie können die folgenden Lambda-Umgebungsvariablen verwenden, um den Konnektor zu konfigurieren.

- AthenaCatalogName – Name der Lambda-Funktion
- ExportBucket – Amazon-S3-Bucket, in den die Vertica-Abfrageergebnisse exportiert werden.
- SpillBucket – Der Name des Amazon-S3-Buckets, in den diese Funktion einen Überlauf von Daten ausführen kann.
- SpillPrefix – Das Präfix für die `SpillBucket`-Stelle, an die diese Funktion einen Überlauf von Daten ausführen kann.
- SecurityGroupIds – Eine oder mehrere IDs, die der Sicherheitsgruppe entsprechen, die auf die Lambda-Funktion angewendet werden soll (z. B. `sg1`, `sg2` oder `sg3`).
- SubnetIds – Eine oder mehrere Subnetz-IDs, die dem Subnetz entsprechen, das die Lambda-Funktion für den Zugriff auf Ihre Datenquelle verwenden kann (z. B. `subnet1` oder `subnet2`).
- SecretNameOrPrefix – Der Name oder das Präfix einer Gruppe von Namen in Secrets Manager, auf die diese Funktion Zugriff hat (z. B. `vertica-*`).
- VerticaConnectionString – Die standardmäßig zu verwendenden Vertica-Verbindungsdetails, wenn keine katalogspezifische Verbindung definiert ist. Die Zeichenkette kann optional AWS Secrets Manager-Syntax (z. B. `${secret_name}`) verwenden.
- VPC ID – Die VPC-ID, die an die Lambda-Funktion angehängt werden soll.

Verbindungszeichenfolge

Verwenden Sie eine JDBC-Verbindungszeichenfolge im folgenden Format, um eine Verbindung zu einer Datenbank-Instance herzustellen.

```
jdbc:vertica://host_name:port/database?user=vertica-username&password=vertica-password
```

Verwenden eines einzelnen Verbindungs-Handlers

Sie können die folgenden Einzelverbindungsmetadaten und Record Handler verwenden, um eine Verbindung zu einer einzelnen Vertica-Instance herzustellen.

Handler-Typ	Klasse
Composite Handler	`VerticaCompositeHandler`
Metadaten-Handler	`VerticaMetadataHandler`
Record Handler	`VerticaRecordHandler`

Parameter für Einzelverbindungs-Handler

Parameter	Beschreibung
`default`	Erforderlich. Die standardmäßige Verbindungszeichenfolge.

Die Einzelverbindungs-Handler unterstützen eine Datenbank-Instance und müssen einen `default`-Verbindungszeichenfolgenparameter bereitstellen. Alle anderen Verbindungszeichenfolgen werden ignoriert.

Bereitstellen von Anmeldeinformationen

Um einen Benutzernamen und ein Kennwort für Ihre Datenbank in Ihrer JDBC-Verbindungszeichenfolge anzugeben, können Sie Eigenschaften von Verbindungszeichenfolgen oder AWS Secrets Manager verwenden.

- Verbindungszeichenfolge – Ein Benutzername und ein Kennwort können als Eigenschaften in der JDBC-Verbindungszeichenfolge angegeben werden.

> **Important**
>
> Als bewährte Sicherheitsmethode sollten Sie keine fest kodierten Anmeldeinformationen in Ihren Umgebungsvariablen oder Verbindungszeichenfolgen verwenden. Informationen zum Verschieben von hartkodierten Geheimnissen nach AWS Secrets Manager finden Sie unter Verschieben von hartkodierten Geheimnissen nach AWS Secrets Manager im AWS Secrets Manager-Benutzerhandbuch.

- AWS Secrets Manager – Um das Athena-Federated-Query-Feature mit AWS Secrets Manager zu verwenden, sollte die mit Ihrer Lambda-Funktion verbundene VPC über einen Internetzugang oder einen VPC-Endpunkt verfügen, um eine Verbindung zu Secrets Manager herzustellen.

  Sie können den Namen eines Secrets in AWS Secrets Manager in Ihrer JDBC-Verbindungszeichenfolge eingeben. Der Konnektor ersetzt den geheimen Namen durch `username`- und `password`-Werte von Secrets Manager.

  Für Amazon RDS-Datenbank-Instances ist diese Unterstützung eng integriert. Wenn Sie Amazon RDS verwenden, empfehlen wir dringend die Verwendung von AWS Secrets Manager und Wechsel der Anmeldeinformationen. Wenn Ihre Datenbank Amazon RDS nicht verwendet, speichern Sie die Anmeldeinformationen als JSON im folgenden Format:

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

Beispiel für eine Verbindungszeichenfolge mit geheimen Namen

Die folgende Zeichenfolge hat die geheimen Namen ${`vertica-username`} und ${`vertica-password`}.

```
jdbc:vertica://host_name:port/database?user=${vertica-username}&password=${vertica-password}
```

Der Konnektor verwendet den geheimen Namen, um Secrets abzurufen und den Benutzernamen und das Kennwort bereitzustellen, wie im folgenden Beispiel gezeigt.

```
jdbc:vertica://host_name:port/database?user=sample-user&password=sample-password
```

Derzeit erkennt der Vertica-Konnektor die `vertica-username`- und `vertica-password`-JDBC-Eigenschaften.

Überlauf-Parameter

Das Lambda-SDK kann Daten an Amazon S3 übertragen. Alle Datenbank-Instances, auf die mit derselben Lambda-Funktion zugegriffen wird, werden an denselben Speicherort verschoben.

Parameter	Beschreibung
`spill_bucket`	Erforderlich. Überlauf-Bucket-Name.
`spill_prefix`	Erforderlich. Schlüssel-Prefix für den Überlauf-Bucket.
`spill_put_request_headers`	(Optional) Eine JSON-codierte Zuordnung von Anforderungsheadern und Werten für die Amazon-S3-`putObject`-Anforderung, die für den Überlauf verwendet wird (z. B. `{"x-amz-server-side-encryption" : "AES256"}`). Andere mögliche Header finden Sie unter PutObject in der API-Referenz zu Amazon Simple Storage Service.

Datentypunterstützung

In der folgenden Tabelle sind die unterstützten Datentypen für den Vertica-Konnektor aufgeführt.

Boolesch
BigInt
Short
Ganzzahl
Long
Gleitkommazahl
Double
Datum
Varchar
Bytes
BigDecimal

Boolesch
TimeStamp als Varchar

Leistung

Die Lambda-Funktion führt Projektions-Pushdown durch, um die von der Abfrage gescannten Daten zu reduzieren. `LIMIT`-Klauseln reduzieren die Menge der gescannten Daten, aber wenn Sie kein Prädikat angeben, sollten Sie davon ausgehen, dass `SELECT`-Abfragen mit einer `LIMIT`-Klausel mindestens 16 MB Daten scannen. Der Vertica-Konnektor ist aufgrund der Gleichzeitigkeit widerstandsfähig gegenüber Drosselung.

Lizenzinformationen

Durch die Verwendung dieses Konnektors erkennen Sie die Aufnahme von Komponenten von Drittanbietern an. Eine Liste dieser Komponenten finden Sie in der pom.xml-Datei für diesen Konnektor. Zudem stimmen Sie den Bedingungen der jeweiligen Drittanbieterlizenzen zu, die in der LICENSE.txt-Datei auf GitHub.com aufgeführt werden.

Weitere Informationen finden Sie auch unter

Aktuelle Informationen zur JDBC-Treiberversion finden Sie in der pom.xml-Datei für den Vertica-Konnektor auf GitHub.com.

Weitere Informationen zu diesem Konnektor finden Sie unter der entsprechenden Seite auf GitHub.com und Querying a Vertica data source in Amazon Athena using the Athena Federated Query SDK (Abfragen einer Vertica-Datenquelle in Amazon Athena mithilfe des Athena-Federated-Query-SDK) im AWS-Big-Data-Blog.

Datenquellen-Konnektor bereitstellen

Die Vorbereitung der Erstellung von Verbundabfragen stellt einen zweiteiligen Prozess dar: Bereitstellung eines Lambda-Funktions-Datenquellen-Connectors und Verbindung der Lambda-Funktion mit einer Datenquelle. In diesem Prozess geben Sie der Lambda-Funktion einen Namen, den Sie später in der Athena-Konsole auswählen können, und geben dem Connector einen Namen, auf den Sie in Ihren SQL-Abfragen verweisen können.

> **Note**
>
> Um das Athena-Federated-Query-Feature mit AWS Secrets Manager zu verwenden, müssen Sie einen privaten Amazon-VPC-Endpunkt für Secrets Manager konfigurieren. Weitere Informationen finden Sie unter Erstellen eines privaten Secrets-Manager-VPC-Endpunkts im Benutzerhandbuch für AWS Secrets Manager.

Themen

Verwenden der Athena-Konsole

Um einen Datenquellen-Connector auszuwählen, zu benennen und bereitzustellen, verwenden Sie die Athena- und Lambda-Konsolen in einem integrierten Prozess.

So stellen Sie einen Datenquellen-Connector bereit

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.

3. Klicken Sie im Navigationsbereich auf Data sources (Datenquellen).
4. Wählen Sie auf der Seite Datenquellen die Option Datenquellen erstellen aus.
5. Wählen Sie für Auswahl einer Datenquelle unter Berücksichtigung der folgenden Richtlinien die Datenquelle aus, die Athena abfragen soll:

   - Wählen Sie eine Verbundabfrageoption aus, die Ihrer Datenquelle entspricht. Athena hat Datenquellen-Connectors vorentwickelt, die Sie für Quellen wie MySQL, Amazon DocumentDB und PostgreSQL konfigurieren können.
   - Wählen Sie S3 – AWS Glue Data Catalog, wenn Sie Daten in Amazon S3 abfragen möchten und keinen Apache-Hive-Metastore oder eine der anderen Verbundabfrage-Datenquellenoptionen auf dieser Seite verwenden. Athena benutzt den AWS Glue Data Catalog, um Metadaten und Schemainformationen für Datenquellen in Amazon S3 zu speichern. Dies ist die Standardoption (nicht verbunden). Weitere Informationen finden Sie unter Verwenden von AWS Glue zum Herstellen einer Verbindung mit Datenquellen in Amazon S3.
   - Wählen Sie S3 – Apache-Hive-Metastore, um Datensätze in Amazon S3 abzufragen, die einen Apache-Hive-Metastore verwenden. Weitere Informationen zu dieser Option finden Sie unter Verbinden von Athena mit einem Apache Hive Metastore.
   - Wählen Sie Benutzerdefinierter oder freigegebener Connector, wenn Sie Ihren eigenen Datenquellen-Connector für die Verwendung mit Athena erstellen möchten. Hinweise zum Schreiben eines Datenquellen-Connectors finden Sie unter Schreiben eines Datenquellen-Konnektors mithilfe des Athena Query Federation SDK.

Dieses Tutorial wählt Amazon CloudWatch Logs als Verbunddatenquelle.

6. Wählen Sie Next (Weiter).
7. Geben Sie auf der Seite Datenquellendetails eingeben für Datenquellenname den Namen ein, den Sie in Ihren SQL-Anweisungen verwenden möchten, wenn Sie die Datenquelle von Athena abfragen (z. B `CloudWatchLogs`). Der Name kann bis zu 127 Zeichen lang sein und muss innerhalb Ihres Kontos eindeutig sein. Er kann nicht mehr geändert werden, nachdem Sie ihn erstellt haben. Gültige Zeichen sind a-z, A-Z, 0-9, _ (Unterstrich), @ (At-Zeichen) und - (Bindestrich). Die Namen `awsdatacatalog`, `hive`, `jmx` und `system` sind von Athena reserviert und können nicht für Datenquellennamen verwendet werden.
8. Für Lambda-Funktion, wählen Sie Erstellen einer Lambda-Funktion aus. Anschließend wird die Funktionsseite für den von Ihnen ausgewählten Connector in der AWS Lambda-Konsole geöffnet. Die Seite enthält detaillierte Informationen zum Connector.
9. Lesen Sie unter Anwendungseinstellungen die Beschreibung für jede Anwendungseinstellung sorgfältig durch und geben Sie dann Werte ein, die Ihren Anforderungen entsprechen.

   Die angezeigten Anwendungseinstellungen variieren je nach Connector für die Datenquelle. Folgende Mindesteinstellungen sind erforderlich:

   - AthenaCatalogName – Ein Name für die Lambda-Funktion, der die Datenquelle angibt, auf die sie gerichtet ist, etwa `cloudwatchlogs`.
   - SpillBucket – Ein Amazon-S3-Bucket in Ihrem Konto zum Speichern von Daten, die die Begrenzung der Antwortgröße der Lambda-Funktion überschreiten.

   > **Note**
   >
   > Daten aus Datenlecks werden in nachfolgenden Ausführungen nicht wiederverwendet und können nach 12 Stunden sicher gelöscht werden. Athena löscht diese Daten nicht für Sie. Um diese Objekte zu verwalten, sollten Sie eine Richtlinie zum Objektlebenszyklus hinzufügen, die alte Daten aus Ihrem Amazon-S3-Spill-Bucket löscht. Weitere Informationen finden Sie unter Managing your storage lifecycle (Verwaltung des Speicherlebenszyklus) im Amazon-S3-Benutzerhandbuch.

10. Wählen Sie Ich bestätige, dass diese App benutzerdefinierte IAM-Rollen und Ressourcenrichtlinien erstellt. Um weitere Informationen zu erhalten, wählen Sie den Link Info .

11. Wählen Sie Bereitstellen aus. Wenn die Bereitstellung abgeschlossen ist, erscheint die Lambda-Funktion im Abschnitt Ressourcen in der Lambda-Konsole.

Eine Verbindung mit der Datenquelle herstellen

Nachdem Sie den Datenquellen-Connector für Ihr Konto bereitgestellt haben, können Sie Athena mit ihm verbinden.

So verbinden Sie Athena mithilfe eines Connectors, den Sie in Ihrem Konto bereitgestellt haben, mit einer Datenquelle

1. Kehren Sie zur Seite Datenquellendetails eingeben der Athena-Konsole zurück.
2. Im Abschnitt Verbindungsdetails wählen Sie das Aktualisierungssymbol neben dem Suchfeld Suchen oder eine Lambda-Funktion eingeben.
3. Wählen Sie den Namen der Funktion aus, die Sie gerade in der Lambda-Konsole erstellt haben. Der ARN der Lambda-Funktion wird angezeigt.
4. (Optional) Fügen Sie für Tags Schlüssel-Wert-Paare hinzu, die mit dieser Datenquelle verknüpft werden sollen. Weitere Informationen zu Tags erhalten Sie unter Markieren von Athena-Ressourcen.
5. Wählen Sie Next (Weiter).
6. Auf der Seite Überprüfen und erstellen prüfen Sie die Datenquellendetails und wählen Sie dann Datenquelle erstellen aus.
7. Der Abschnitt Datenquellendetails auf der Seite für Ihre Datenquelle zeigt Informationen über Ihren neuen Connector an. Sie können den Connector jetzt in Ihren Athena-Abfragen verwenden.

   Informationen zur Verwendung von Datenkonnektoren in Abfragen finden Sie unter Verbundabfragen ausführen.

Bereitstellen eines Datenquellen-Connectors mit AWS Serverless Application Repository

Um einen Datenquellen-Connector bereitzustellen, können Sie die AWS Serverless Application Repository verwenden anstatt mit der Athena-Konsole zu beginnen. Verwenden Sie AWS Serverless Application Repository, um den Connector zu suchen, den Sie verwenden möchten, die Parameter anzugeben, die der Connector benötigt, und den Connector dann für Ihr Konto bereitzustellen. Nachdem Sie den Connector bereitgestellt haben, verwenden Sie die Athena-Konsole, um die Datenquelle für Athena verfügbar zu machen.

Bereitstellen des Connectors für Ihr Konto

So verwenden Sie den AWS Serverless Application Repository zum Bereitstellen eines Datenquellen-Connectors für Ihr Konto:

1. Melden Sie sich an der AWS Management Console an und öffnen Sie Serverless App Repository (Serverless-App-Repository).
2. Wählen Sie im Navigationsbereich Available applications (Verfügbare Anwendungen) aus.
3. Wählen Sie die Option Apps anzeigen, die benutzerdefinierte IAM-Rollen oder Ressourcenrichtlinien erstellen.
4. Geben Sie im Suchfeld den Namen des Connectors ein. Die Liste vorab erstellter Athena-Daten-Connector finden Sie unter Verfügbare Datenquellenkonnektoren.
5. Wählen Sie den Namen des Connectors aus. Wenn Sie einen Connector auswählen, wird die Seite Anwendungsdetails der Lambda-Funktion in der AWS Lambda-Konsole geöffnet.
6. Geben Sie auf der rechten Seite der Detailseite für Application settings (Anwendungseinstellungen) die erforderlichen Informationen ein. Zu den mindestens erforderlichen Einstellungen gehören die folgenden. Informationen zu den verbleibenden konfigurierbaren Optionen für Daten-Connectors , die von Athena erstellt wurden, finden Sie im entsprechenden Thema Verfügbare Connectors auf GitHub.

   - AthenaCatalogName – Ein Name für die Lambda-Funktion in Kleinbuchstaben, der die Datenquelle angibt, auf die sie gerichtet ist, etwa `cloudwatchlogs`.
   - SpillBucket – Geben Sie einen Amazon-S3-Bucket in Ihrem Konto an, um Daten von großen Antwortnutzlasten zu empfangen, die die Größenbeschränkungen für Lambda-Funktionsantworten überschreiten.
7. Wählen Sie Ich bestätige, dass diese App benutzerdefinierte IAM-Rollen und Ressourcenrichtlinien erstellt. Um weitere Informationen zu erhalten, wählen Sie den Link Info .
8. Wählen Sie unten rechts im Abschnitt Anwendungseinstellungen Bereitstellen. Wenn die Bereitstellung abgeschlossen ist, erscheint die Lambda-Funktion im Abschnitt Ressourcen in der Lambda-Konsole.

Den Connector in Athena verfügbar machen

Nun sind Sie bereit, die Athena-Konsole zu verwenden, um die Datenquelle für Athena verfügbar zu machen.

Um die Datenquelle für Athena verfügbar zu machen

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.

3. Klicken Sie im Navigationsbereich auf Data sources (Datenquellen).
4. Wählen Sie auf der Seite Datenquellen die Option Datenquellen erstellen aus.
5. Für Auswählen einer Datenquelle wählen Sie die Datenquelle aus, für die Sie einen Connector in AWS Serverless Application Repository erstellt haben. Dieses Tutorial nutzt Amazon CloudWatch Logs als Verbunddatenquelle.
6. Wählen Sie Next (Weiter).
7. Geben Sie auf der Seite Datenquellendetails eingeben für Datenquellenname den Namen ein, den Sie in Ihren SQL-Anweisungen verwenden möchten, wenn Sie die Datenquelle von Athena abfragen (z. B `CloudWatchLogs`). Der Name kann bis zu 127 Zeichen lang sein und muss innerhalb Ihres Kontos eindeutig sein. Er kann nicht mehr geändert werden, nachdem Sie ihn erstellt haben. Gültige Zeichen sind a-z, A-Z, 0-9, _ (Unterstrich), @ (At-Zeichen) und - (Bindestrich). Die Namen `awsdatacatalog`, `hive`, `jmx` und `system` sind von Athena reserviert und können nicht für Datenquellennamen verwendet werden.
8. Im Abschnitt Verbindungsdetails verwenden Sie das Feld Auswählen oder Eingeben einer Lambda-Funktion, um den Namenn der Funktion, die Sie soeben erstellt haben, auszuwählen. Der ARN der Lambda-Funktion wird angezeigt.

9. (Optional) Fügen Sie für Tags Schlüssel-Wert-Paare hinzu, die mit dieser Datenquelle verknüpft werden sollen. Weitere Informationen zu Tags erhalten Sie unter Markieren von Athena-Ressourcen.
10. Wählen Sie Next (Weiter).
11. Auf der Seite Überprüfen und erstellen prüfen Sie die Datenquellendetails und wählen Sie dann Datenquelle erstellen aus.
12. Der Abschnitt Datenquellendetails auf der Seite für Ihre Datenquelle zeigt Informationen über Ihren neuen Connector an. Sie können den Connector jetzt in Ihren Athena-Abfragen verwenden.

    Informationen zur Verwendung von Datenkonnektoren in Abfragen finden Sie unter Verbundabfragen ausführen.

Erstellen einer VPC für einen Datenquellen-Connector

Einige Athena-Datenquellen-Connectors benötigen eine VPC und eine Sicherheitsgruppe. In diesem Thema wird gezeigt, wie Sie eine VPC mit einem Subnetz und einer Sicherheitsgruppe für die VPC erstellen. Im Rahmen dieses Prozesses rufen Sie die IDs für VPC, Subnetz und Sicherheitsgruppe ab, die Sie erstellen. Diese IDs sind erforderlich, wenn Sie Ihren Connector für die Verwendung mit Athena konfigurieren.

So erstellen Sie eine VPC für einen Athena-Datenquellen-Connector

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc/.
2. Stellen Sie im Navigationsbereich sicher, dass New VPC Experience (Neues VPC-Erlebnis) ausgewählt ist.
3. Wählen Sie Create VPC aus.
4. Unter VPC Settings (VPC-Einstellungen) wählen Sie bei Resources to create (zu erstellende Ressourcen) VPC and more (VPC und mehr) aus.
5. Geben Sie für Auto-generate (Automatisch generieren) einen Wert ein, der verwendet wird, um Namensschilder für alle Ressourcen in Ihrer VPC zu generieren.
6. Wählen Sie Create VPC aus.
7. Wählen Sie View VPC (VPC anzeigen).
8. Kopieren Sie im Abschnitt Details für VPC ID Ihre VPC-ID zur späteren Referenz.

Jetzt können Sie die Subnetz-ID für die VPC abrufen, die Sie gerade erstellt haben.

So rufen Sie Ihre VPC-Subnetz-ID ab

1. Wählen Sie im Navigationsbereich der VPC-Konsole Subnets (Subnetze).
2. Wählen Sie den Namen aus, der dem von Ihnen erstellten Subnetz entspricht.
3. Kopieren Sie im Abschnitt Details für Subnet ID Ihre Subnetz-ID zur späteren Referenz.

Erstellen Sie als nächstes eine VPC-Sicherheitsgruppe für Ihre VPC.

So erstellen Sie eine Sicherheitsgruppe für Ihre VPC

1. Wählen Sie im Navigationsbereich der VPC-Konsole Security (Sicherheit), Security Groups (Sicherheitsgruppen).
2. Wählen Sie Create security group (Sicherheitsgruppe erstellen).
3. Geben Sie auf der Seite Create security group (Sicherheitsgruppe erstellen) die folgenden Informationen ein:
   - Geben Sie für Security group name (Sicherheitsgruppen-Name) einen Namen für Ihre Sicherheitsgruppe ein.
   - Geben Sie für Description (Beschreibung) eine Beschreibung für die Sicherheitsgruppe. Dies ist ein Pflichtfeld.
   - Geben Sie für VPC die VPC-ID der VPC ein, die Sie für Ihren Datenquellen-Connector erstellt haben.
   - Fügen Sie für Inbound rules (eingehende Regeln) und Outbound rules (ausgehende Regeln) alle erforderlichen Regeln für ein- und ausgehenden Datenverkehr hinzu.
4. Wählen Sie Create security group (Sicherheitsgruppe erstellen).
5. Kopieren Sie auf der Seite Details für die Sicherheitsgruppe die Security group ID (Sicherheitsgruppen-ID) zur späteren Referenz.

Aktivieren von kontoübergreifenden Verbundabfragen

Verbundabfrage ermöglicht es Ihnen, andere Datenquellen als Amazon S3 mithilfe von Datenquellen-Connectors abzufragen, die auf AWS Lambda bereitgestellt sind. Das kontoübergreifende Verbundabfragefeature ermöglicht es, dass sich die Lambda-Funktion und die abzufragenden Datenquellen in verschiedenen Konten befinden.

Als Datenadministrator können Sie kontoübergreifende Verbundabfragen aktivieren, indem Sie Ihren Daten-Connector mit dem Konto eines Datenanalysten freigeben oder als Datenanalyst einen freigegebenen Lambda-ARN von einem Datenadministrator verwenden, um es Ihrem Konto hinzuzufügen. Wenn Konfigurationsänderungen an einem Connector im Ursprungskonto vorgenommen werden, wird die aktualisierte Konfiguration automatisch auf die freigegebenen Instances des Connectors in den Konten anderer Benutzer angewendet.

Überlegungen und Einschränkungen

- Das kontoübergreifende Verbundabfragefeature ist für Nicht-Hive-Metastore-Daten-Connectors verfügbar, die eine Lambda-basierte Datenquelle verwenden.
- Das Feature ist nicht verfügbar für den AWS Glue Data Catalog-Datenquellentyp. Weitere Informationen zum kontoübergreifenden Zugriff auf AWS Glue Data Catalog, siehe Kontoübergreifender Zugriff auf AWS Glue -Datenkataloge.
- Wenn die Antwort der Lambda-Funktion Ihres Konnektors die Beschränkung der Lambda-Antwortgröße von 6 MB überschreitet, verschlüsselt Athena die Antwort automatisch, stapelt sie und spillt sie an einen von Ihnen konfigurierten Amazon-S3-Bucket weiter. Die Entität, die die Athena-Abfrage ausführt, muss Zugriff auf den Spill-Standort haben, damit Athena die übergebenen Daten lesen kann. Wir empfehlen, dass Sie eine Amazon-S3-Lebenszyklusrichtlinie einrichten, um Objekte vom Spill-Speicherort zu löschen, da die Daten nach Abschluss der Abfrage nicht benötigt werden.
- Die Verwendung von Verbundabfragen über mehrere AWS-Regionen wird nicht unterstützt.

Erforderliche Berechtigungen

- Damit das Datenadministratorkonto A eine Lambda-Funktion mit dem Datenanalystenkonto B teilen kann, erfordert Konto B die Lambda-Aufruffunktion und den Spill-Bucket-Zugriff. Dementsprechend sollte Konto A der Lambda-Funktion und dem Prinzipal Zugriff auf seinen Spill-Bucket in Amazon S3 eine ressourcenbasierte Richtlinie hinzufügen.
  1. Die folgende Richtlinie erteilt Lambda-Funktionsberechtigungen für Konto B für eine Lambda-Funktion in Konto A aufzurufen.

```
{
  "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. Die folgende Richtlinie ermöglicht den Spill-Bucket-Zugriff auf den Prinzipal in Konto 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. Wenn die Lambda-Funktion den Spill-Bucket mit einem AWS KMS-Schlüssel anstelle der vom Verbund-SDK angebotenen Standardverschlüsselung verschlüsselt, muss die AWS KMS-Schlüsselrichtlinie in Konto A dem Benutzer in Konto B Zugriff gewähren, wie im folgenden Beispiel.

```
{
    "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.
}
```

- Damit Konto A seinen Connector mit Konto B teilen kann, muss Konto B eine Rolle namens `AthenaCrossAccountCreate-account-A-id` erstellen, die Konto A durch Aufrufen der AWS-Sicherheitstokenservice-AssumeRole-API-Aktion annimmt.

  Die folgende Richtlinie, die die `CreateDataCatalog`-Aktion zulässt, sollte in Konto B erstellt und der `AthenaCrossAccountCreate-account-A-id`-Rolle hinzugefügt werden, die Konto B für Konto A erstellt.

  ```
  {
   "Effect": "Allow",
   "Action": "athena:CreateDataCatalog",
   "Resource": "arn:aws:athena:*:account-B-id:datacatalog/*"
  }
  ```

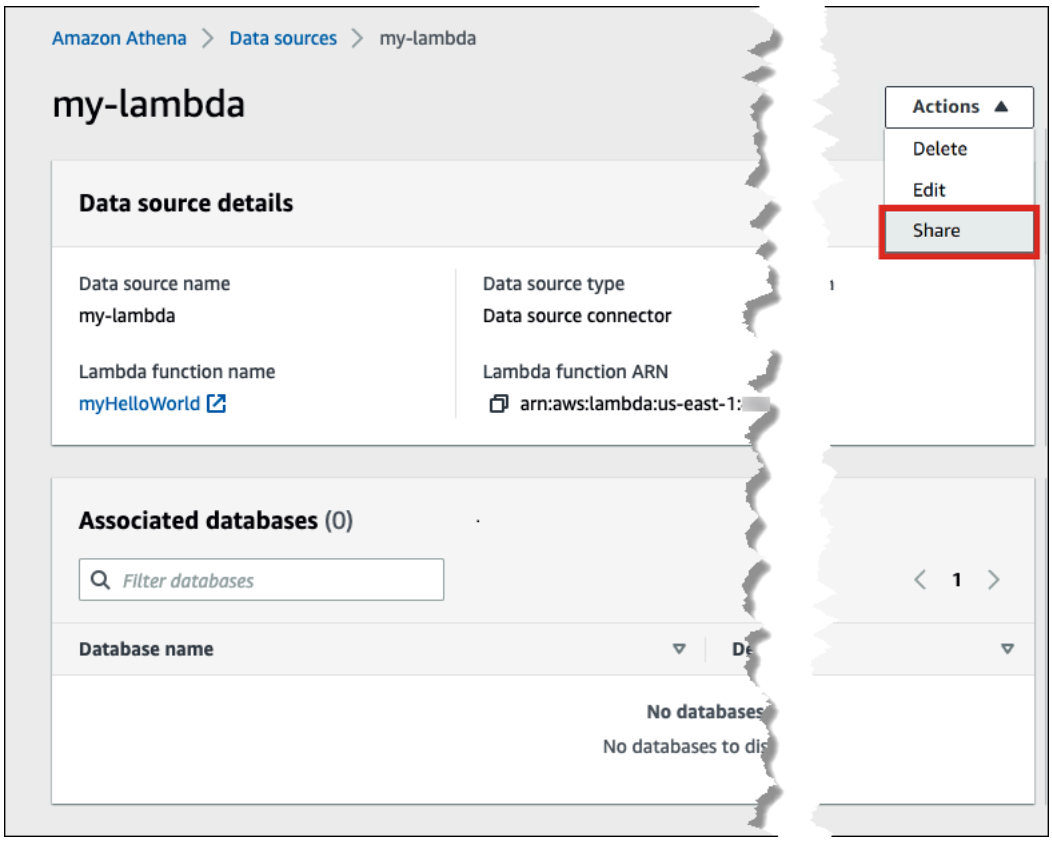
Freigeben einer Datenquelle in Konto A mit Konto B

Nachdem Berechtigungen vorhanden sind, können Sie die Data sources (Datenquellen)-Seite in der Athena-Konsole verwenden, um einen Daten-Connector in Ihrem Konto (Konto A) mit einem anderen Konto (Konto B) zu teilen. Konto A behält die volle Kontrolle und den Besitz des Connectors. Wenn Konto A Konfigurationsänderungen am Connector vornimmt, gilt die aktualisierte Konfiguration für den freigegebenen Connector in Konto B.

So teilen Sie eine Lambda-Datenquelle in Konto A mit Konto B

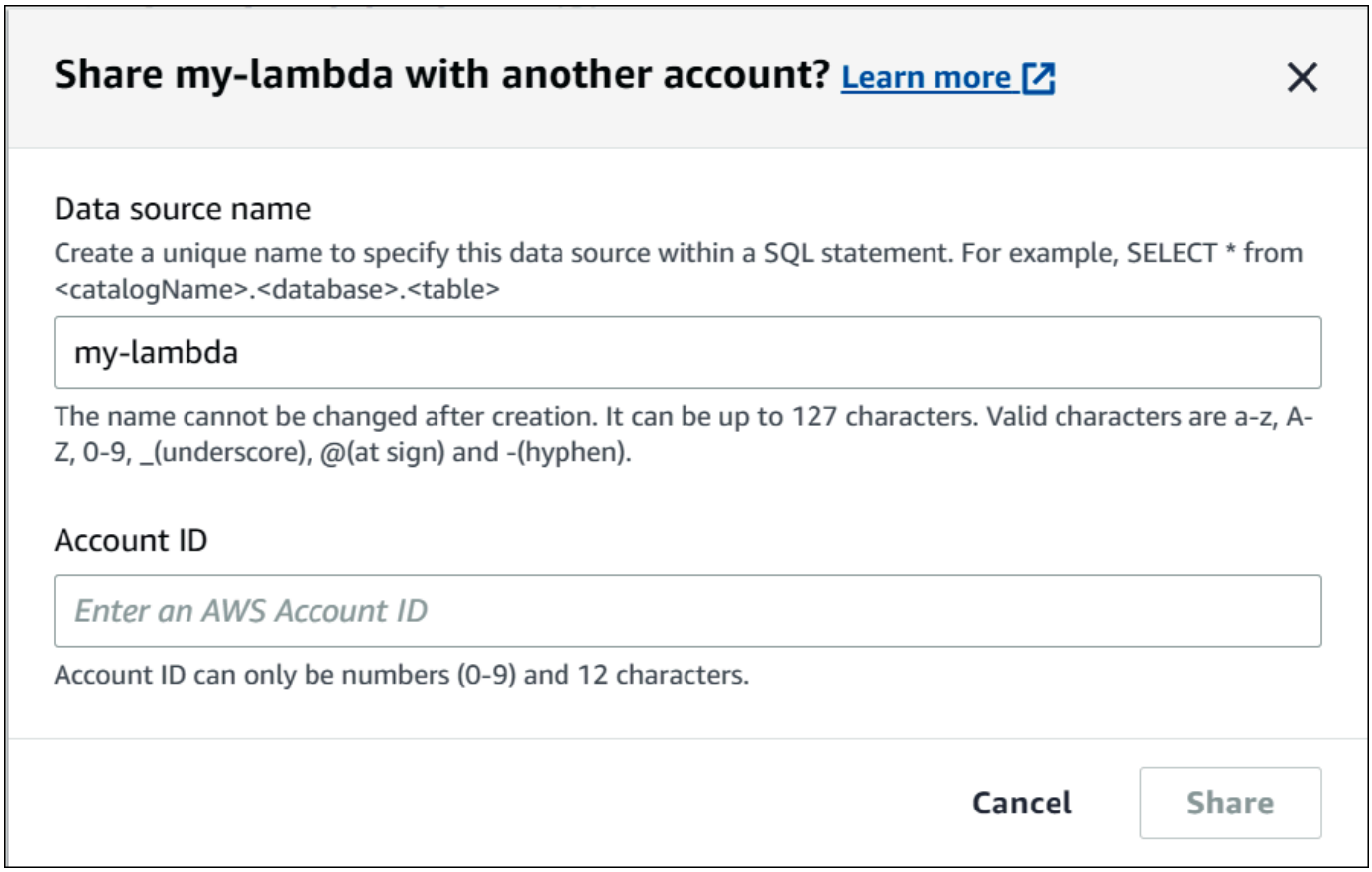
1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.

3. Wählen Sie Data sources (Datenquellen) aus.
4. Wählen Sie auf der Seite Data sources (Datenquellen) den Link des Connectors aus, den Sie freigeben möchten.
5. Wählen Sie auf der Detailseite für eine Lambda-Datenquelle die Option Freigeben in der oberen rechten Ecke aus.



6. Geben Sie im Dialogfeld ***Lambda-Name*** mit einem anderen Konto teilen die erforderlichen Informationen ein.

   - Geben Sie für Data source name (Datenquellenname) den Namen der kopierten Datenquelle ein, wie er im anderen Konto angezeigt werden soll.
   - Geben Sie bei Konto-ID die ID des Kontos ein, mit dem Sie Ihre Datenquelle teilen möchten (in diesem Fall Konto B).



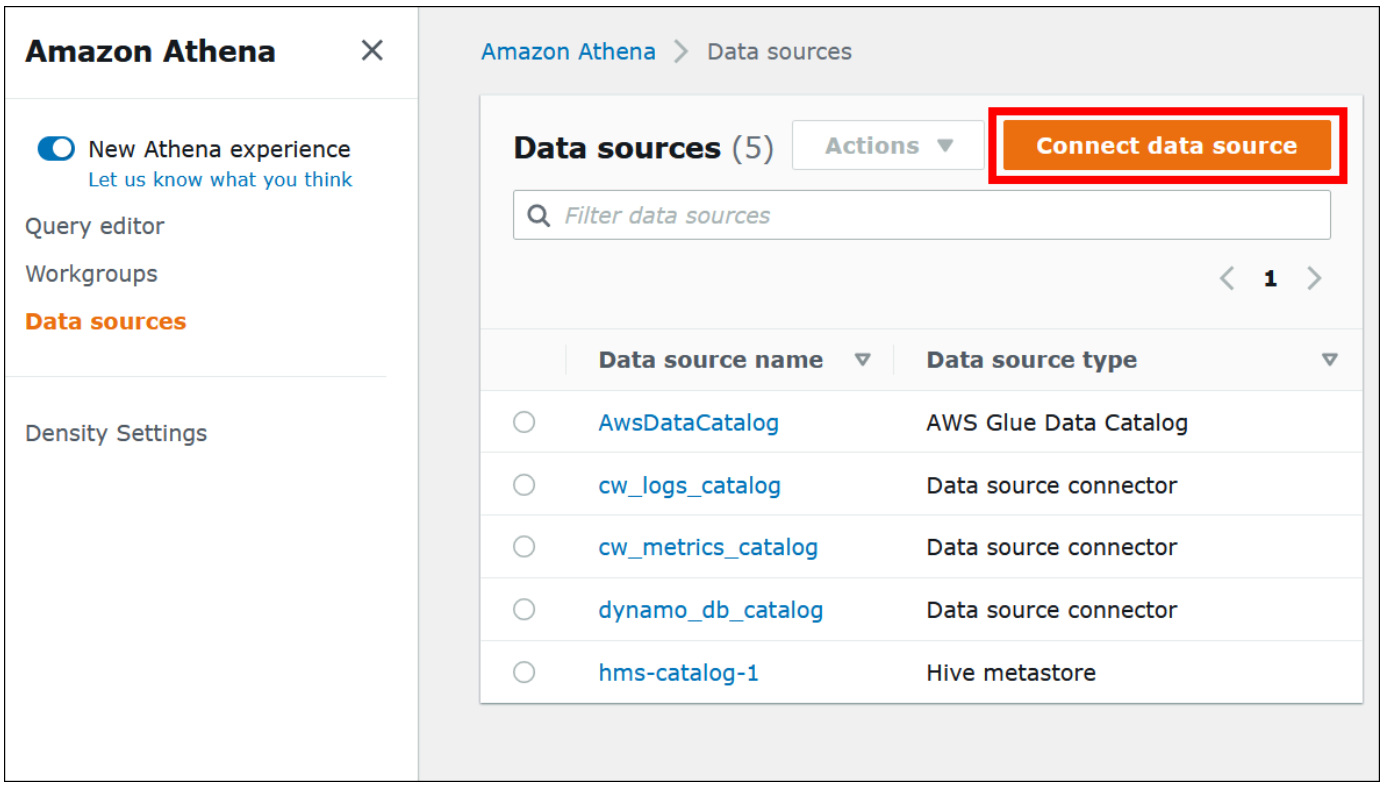
7. Wählen Sie Share (Teilen) aus. Der von Ihnen angegebene Connector für freigegebene Daten wird in Konto B erstellt. Konfigurationsänderungen am Connector in Konto A gelten für den Connector in Konto B.

Hinzufügen einer freigegebenen Datenquelle aus Konto A zu Konto B

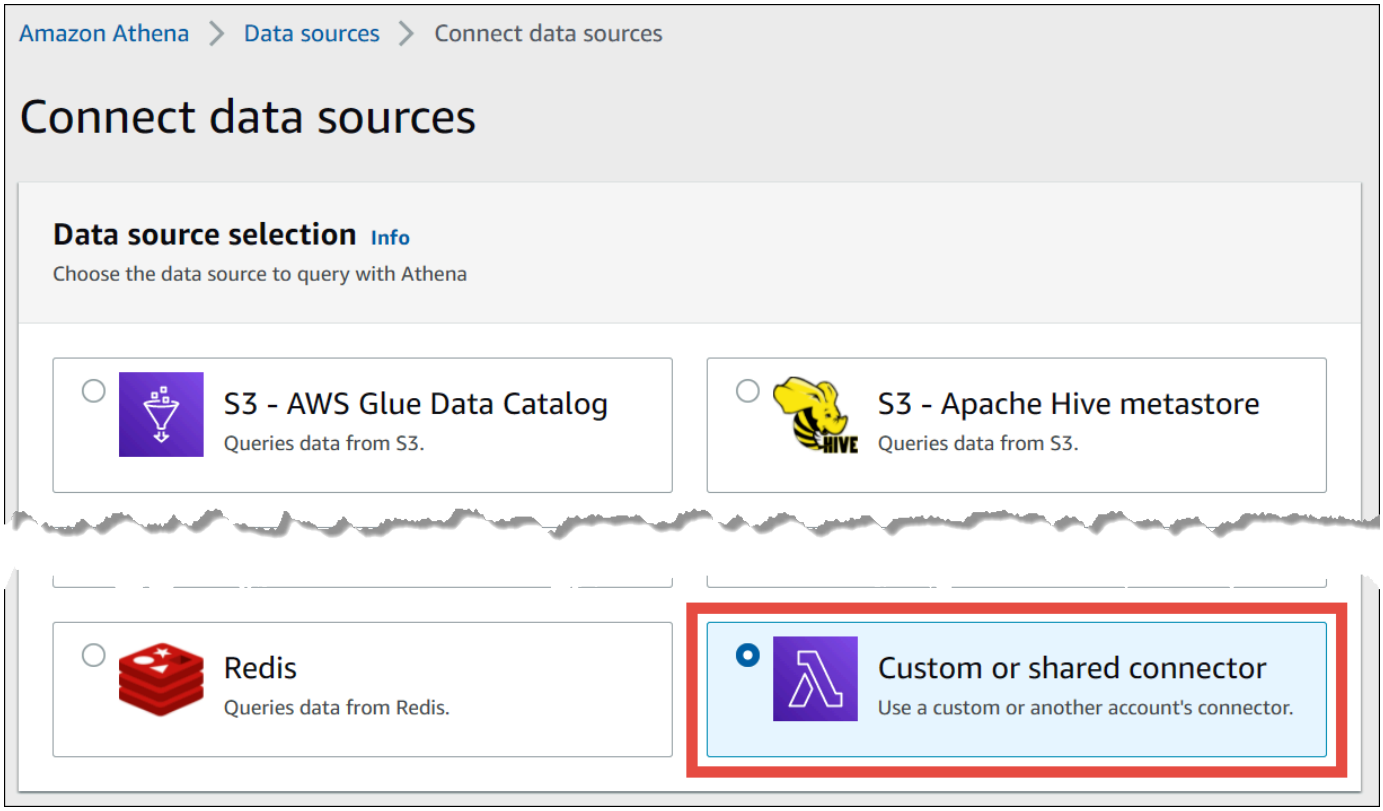
Als Datenanalyst erhalten Sie möglicherweise den ARN eines Connectors, den Sie von einem Datenadministrator zu Ihrem Konto hinzufügen können. Sie können die Seite Data sources (Datenquellen) der Athena-Konsole verwenden, um den von Ihrem Administrator bereitgestellten Lambda-ARN zu Ihrem Konto hinzuzufügen.

So fügen Sie den Lambda-ARN eines freigegebenen Daten-Connectors zu Ihrem Konto hinzu

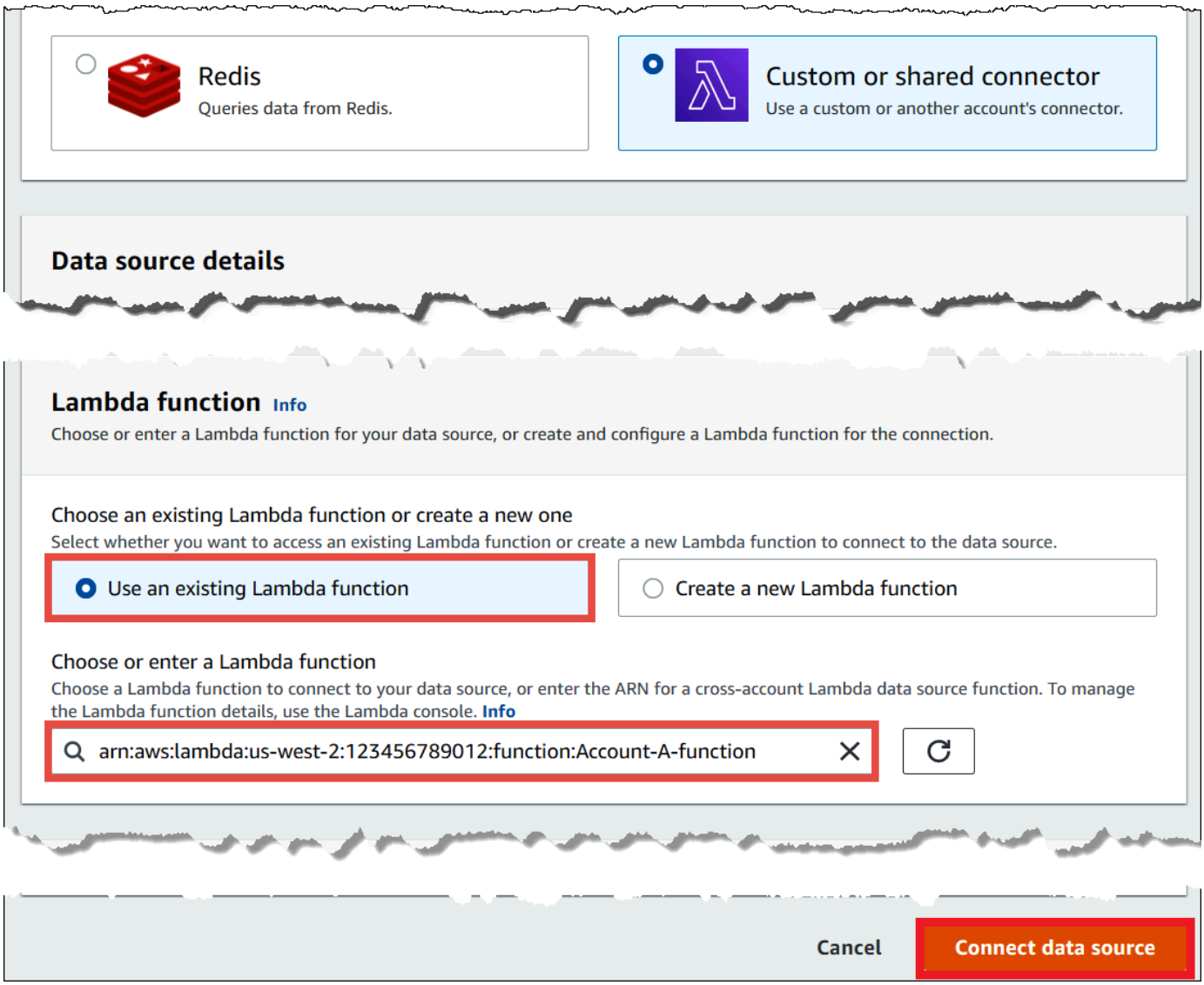
1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn Sie die neue Konsolenerfahrung verwenden und der Navigationsbereich nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.
3. Wählen Sie Data sources (Datenquellen).
4. Wählen Sie auf der Seite Data sources (Datenquellen) die Option Connect data source (Datenquelle verbinden) aus.



5. Wählen Sie Custom or shared connector (Benutzerdefinierter oder freigegebener Connector) aus.



6. Stellen Sie sicher, dass im Abschnitt Lambda-Funktion die Option Eine vorhandene Lambda-Funktion verwenden ausgewählt ist.



7. Geben Sie für Lambda-Funktion auswählen oder eingeben den Lambda-ARN von Konto A ein.
8. Wählen Sie Connect data source (Datenquelle verbinden) aus.

Fehlerbehebung

Wenn Sie eine Fehlermeldung erhalten, dass Konto A nicht über die Berechtigung verfügt, eine Rolle in Konto B zu übernehmen, stellen Sie sicher, dass der Name der in Konto B erstellten Rolle korrekt geschrieben ist und dass die richtige Richtlinie angehängt ist.

Schreiben eines Datenquellen-Konnektors

Athena empfiehlt, die von Ihnen verwendeten Datenquellen-Konnektoren regelmäßig auf die neueste Version zu aktualisieren, um von neuen Features und Verbesserungen zu profitieren. Um zu beginnen, müssen Sie die neueste Versionsnummer finden.

Die neueste Version von Athena-Verbundabfrage finden

Die neueste Versionsnummer der Athena-Datenquellen-Konnektors entspricht der neuesten Version von Athena-Verbundabfrage. In bestimmten Fällen können die GitHub-Versionen etwas neuer sein als die, die auf der AWS Serverless Application Repository (SAR) verfügbar sind.

Die neueste Versionsnummer von Athena-Verbundabfrage finden

1. Besuchen Sie die GitHub-URL https://github.com/awslabs/aws-athena-query-federation/releases/latest.
2. Notieren Sie sich die Versionsnummer in der Überschrift der Hauptseite im folgenden Format:

   Release v *year.week_of_year.iteration_of_week* of Athena Query Federation

   Die Versionsnummer für Version 2023.8.3 von Athena-Verbundabfrage lautet beispielsweise 2023.8.3.

Ressourcennamen finden und notieren

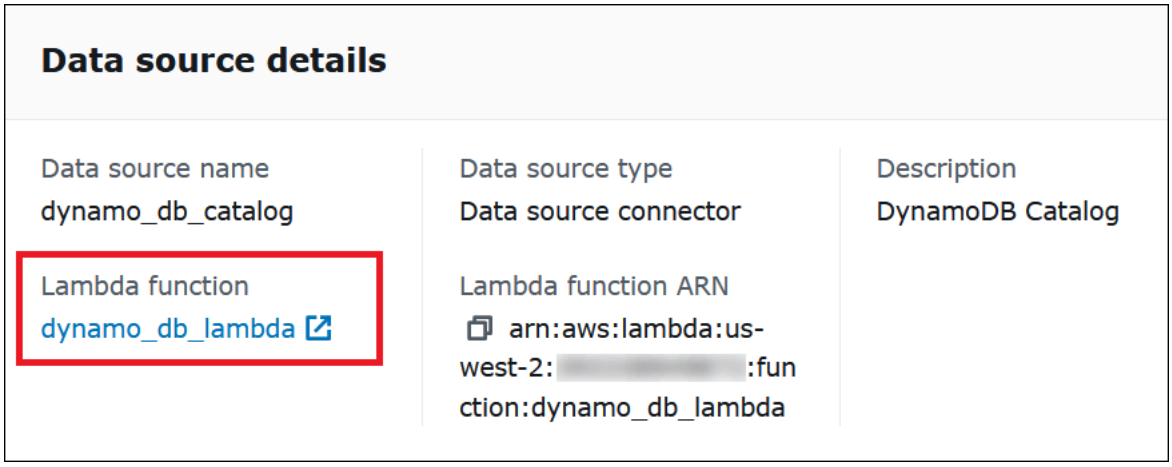
Zur Vorbereitung des Upgrades müssen Sie die folgenden Informationen finden und sich diese notieren:

1. Der Lambda-Funktionsname für den Konnektor.
2. Umgebungsvariablen für die Lambda-Funktion.
3. Der Name der Lambda-Anwendung, die die Lambda-Funktion für den Konnektor verwaltet.

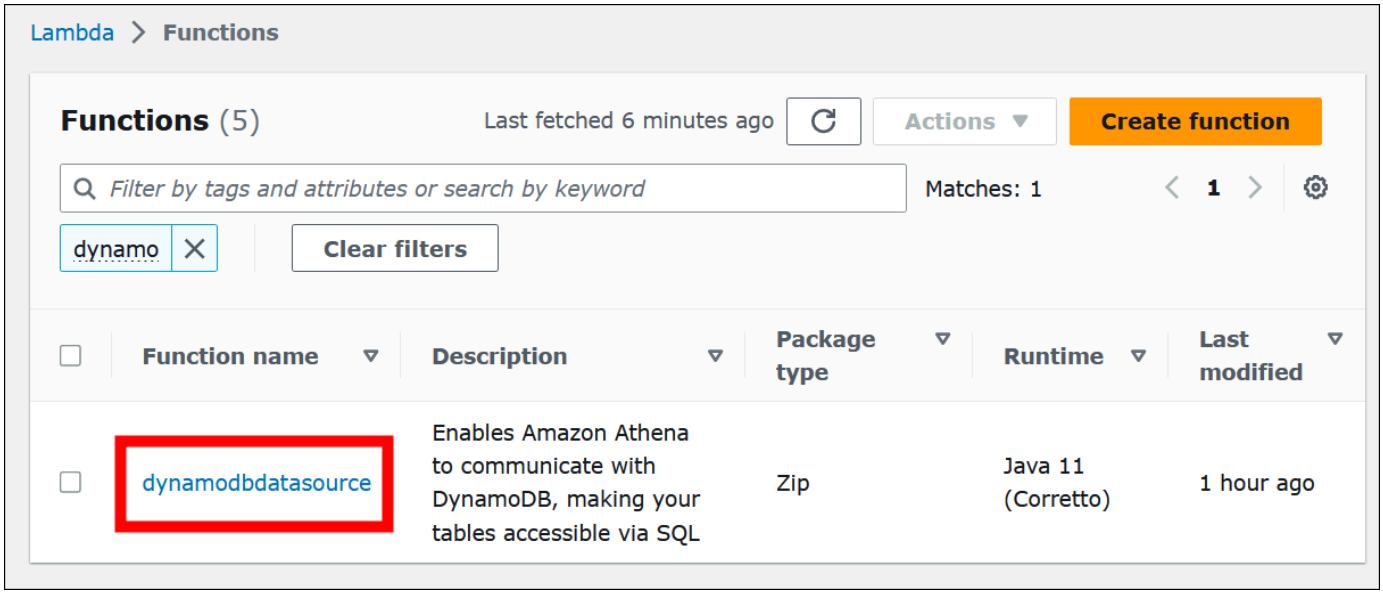
Wie Sie Ressourcennamen von der Athena-Konsole aus suchen

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.

3. Klicken Sie im Navigationsbereich auf Data sources (Datenquellen).
4. Wählen Sie in der Spalte Datenquellenname den Link zur Datenquelle für Ihren Konnektor aus.
5. Wählen Sie im Abschnitt Datenquellendetails unter Lambda-Funktion den Link zu Ihrer Lambda-Funktion aus.



6. Notieren Sie sich auf der Seite Funktionen in der Spalte Funktionsname den Funktionsnamen für Ihren Konnektor.



7. Wählen Sie den Link mit dem Funktionsnamen.
8. Wählen Sie im Abschnitt Funktionsübersicht die Registerkarte Konfiguration aus.
9. Wählen Sie im Bereich auf der linken Seite Umgebungsvariablen aus.
10. Notieren Sie sich im Abschnitt Umgebungsvariablen die Schlüssel und ihre entsprechenden Werte.
11. Scrollen Sie zum Seitenanfang.
12. In der Nachricht Diese Funktion gehört zu einer Anwendung. Klicken Sie hier, um sie zu verwalten, und wählen Sie den Link Hier klicken.
13. Notieren Sie sich auf der Seite serverlessrepo-***your_application_name*** den NamenIhrer Anwendung ohne serverlessrepo. Wenn der Anwendungsname beispielsweise serverlessrepo-DynamoDbTestApp lautet, dann lautet Ihr Anwendungsname DynamoDbTestApp.
14. Bleiben Sie auf der Lambda-Konsolenseite für Ihre Anwendung und fahren Sie dann mit den Schritten unter Suchen der Version des Konnektors fort, den Sie verwenden.

Suchen Sie die Version des Konnektors, den Sie verwenden

Gehen Sie folgendermaßen vor, um die Version des von Ihnen verwendeten Konnektors zu ermitteln.

Wie Sie die Version des Konnektors, den Sie verwenden, suchen

1. Wählen Sie auf der Konsolenseite für Ihre Lambda-Anwendung die Registerkarte Bereitstellungen aus.
2. Erweitern Sie auf der Registerkarte Bereitstellungen SAM-Vorlage.

3. Suchen Sie nach CodeURI.
4. Suchen Sie im Feld Schlüssel unter CodeURI nach der folgenden Zeichenfolge:

```
applications-connector_name-
versions-year.week_of_year.iteration_of_week/hash_number
```

Das folgende Beispiel zeigt eine Zeichenfolge für den CloudWatch-Konnektor:

```
applications-AthenaCloudwatchConnector-versions-2021.42.1/15151159...
```

5. Notieren Sie den Wert für das *year.week_of_year.iteration_of_week* (zum Beispiel 2021.42.1). Dies ist die Version für Ihren Konnektor.

Die neue Version Ihres Konnektors bereitstellen

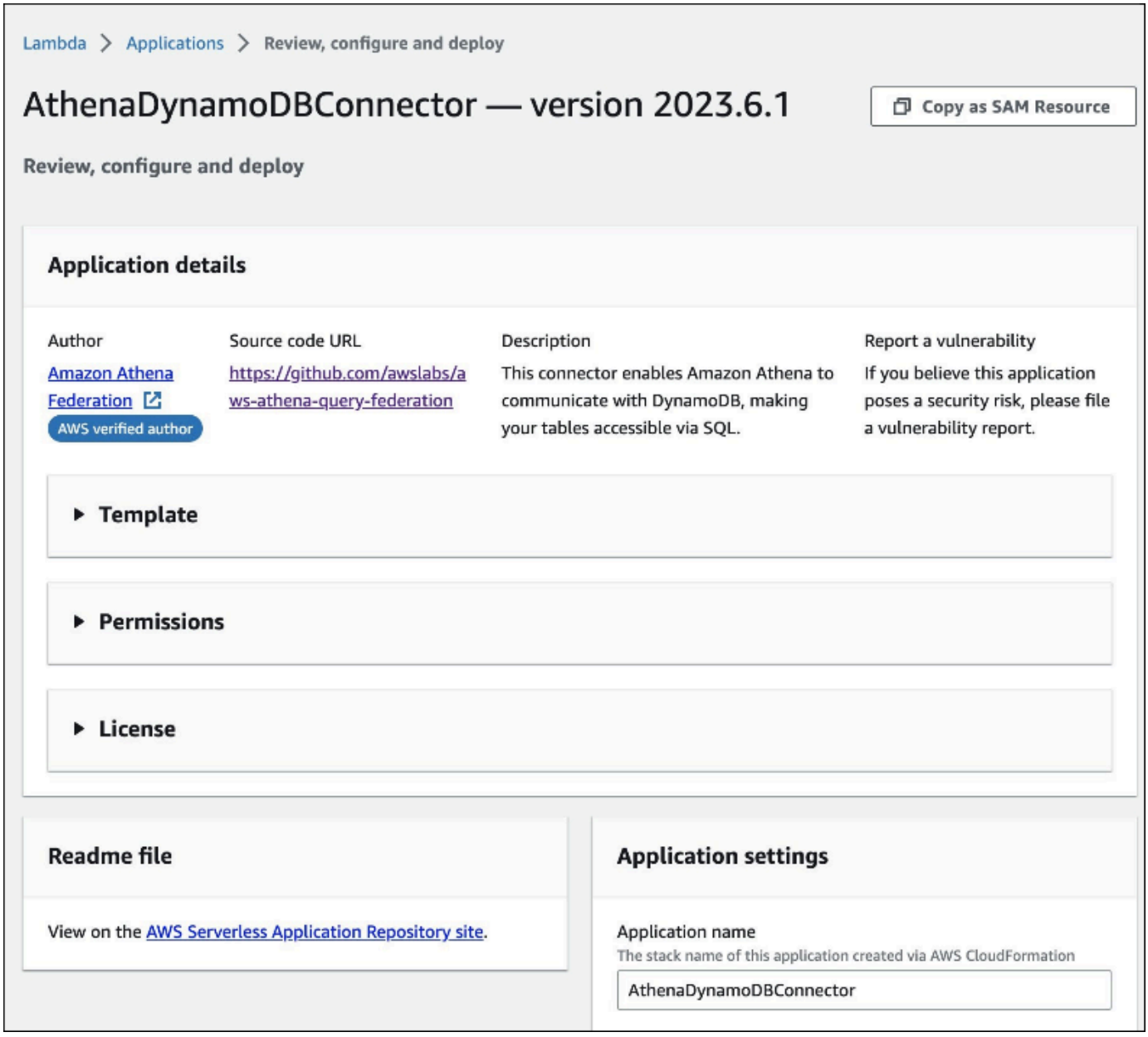
Gehen Sie wie folgt vor, um eine neue Version Ihres Konnektors bereitzustellen.

Wie Sie die neue Version Ihres Konnektors bereitstellen

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.

3. Klicken Sie im Navigationsbereich auf Data sources (Datenquellen).
4. Wählen Sie auf der Seite Datenquellen die Option Datenquellen erstellen aus.

5. Wählen Sie die Datenquelle aus, für die Sie ein Upgrade durchführen möchten, und wählen Sie dann Weiter aus.
6. Wählen Sie auf der Seite Verbindungsdetails die Option Neue Lambda-Funktion erstellen aus. Dadurch wird die Lambda-Konsole geöffnet, in der Sie Ihre aktualisierte Anwendung bereitstellen können.



7. Da Sie keine neue Datenquelle erstellen, können Sie die Registerkarte Athena-Konsole schließen.
8. Führen Sie Lambda-Konsolenseite für den Konnektor die folgenden Schritte aus:

a. Stellen Sie sicher, dass Sie das Präfix serverlessrepo- aus Ihrem Anwendungsnamen entfernt haben, und kopieren Sie dann den Anwendungsnamen in das Feld Anwendungsname.

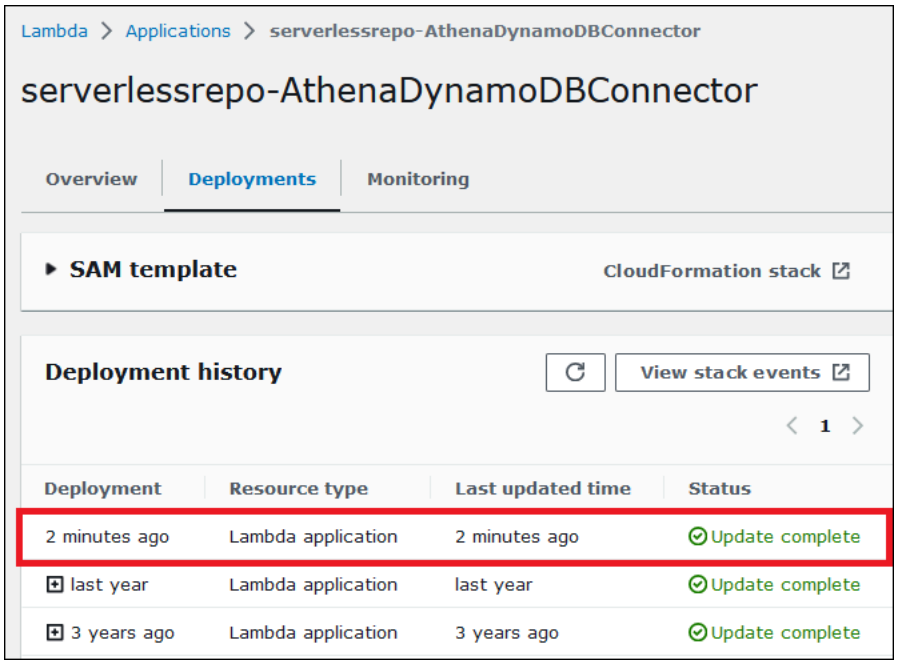
b. Kopieren Sie Ihren Lambda-Funktionsnamen in das Feld AthenaCatalogName. Einige Konnektoren nennen dieses Feld LambdaFunctionName.

c. Kopieren Sie die Umgebungsvariablen, die Sie aufgezeichnet haben, in die entsprechenden Felder.

9. Wählen Sie Ich bestätige, dass diese Anwendung benutzerdefinierte IAM-Rollen und Ressourcenrichtlinien erstellt und dann Bereitstellen.

10. Um zu überprüfen, ob Ihre Anwendung aktualisiert wurde, wählen Sie die Registerkarte Bereitstellungen.

    Im Abschnitt Bereitstellungsverlauf wird angezeigt, dass Ihr Update abgeschlossen ist.



11. Um die neue Versionsnummer zu bestätigen, können Sie die SAM-Vorlage wie zuvor erweitern, nach CodeURI suchen und die Versionsnummer des Konnektors im Feld Schlüssel überprüfen.

Sie können jetzt Ihren aktualisierten Konnektor verwenden, um Athena-Verbundabfragen zu erstellen.

Verbundabfragen ausführen

Wenn Sie einen oder mehrere Daten-Connector konfiguriert und in Ihrem Konto bereitgestellt haben, können Sie diese in Ihren Athena-Abfragen verwenden.

Abfragen einer einzelnen Datenquelle

In den Beispielen in diesem Abschnitt wird davon ausgegangen, dass Sie den Amazon Athena CloudWatch Konnektor für Ihr Konto konfiguriert und bereitgestellt haben. Sie verwenden den gleichen Ansatz für Abfragen für andere Connectors.

So erstellen Sie eine Athena-Abfrage, die den CloudWatch-Connector verwendet

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Erstellen Sie im Athena-Abfrage-Editor eine SQL-Abfrage mit der folgenden Syntax in der `FROM`-Klausel.

```
MyCloudwatchCatalog.database_name.table_name
```

Beispiele

Im folgenden Beispiel wird der Athena-CloudWatch-Connector verwendet, um eine Verbindung mit der `all_log_streams`-Ansicht in der `/var/ecommerce-engine/order-processor`-CloudWatch-Logs-Protokollgruppe herzustellen. Die Ansicht `all_log_streams` ist eine Ansicht aller Protokollstreams in der Protokollgruppe. Die Beispielabfrage begrenzt die Anzahl der zurückgegebenen Zeilen auf 100.

```
SELECT *
FROM "MyCloudwatchCatalog"."/var/ecommerce-engine/order-processor".all_log_streams
LIMIT 100;
```

Im folgenden Beispiel werden Informationen aus derselben Ansicht wie im vorherigen Beispiel analysiert. Im Beispiel werden die Bestell-ID und die Protokollebene extrahiert und alle Nachrichten mit der Ebene `INFO` herausgefiltert.

```
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'
```

Abfragen mehrerer Datenquellen

Stellen Sie sich als komplexeres Beispiel ein E-Commerce-Unternehmen vor, das die folgenden Datenquellen verwendet, um Daten zu Kundenkäufen zu speichern:

- Amazon RDS für MySQL zum Speichern von Produktkatalogdaten
- Amazon DocumentDB zur Speicherung von Kundenkontodaten wie E-Mail- und Versandadressen
- Amazon DynamoDB zum Speichern von Versand- und Tracking-Daten von Bestellungen

Stellen Sie sich vor, ein Datenanalyst für diese E-Commerce-Anwendung erfährt, dass die Versandzeit in einigen Regionen durch die lokalen Wetterbedingungen beeinflusst wurde. Der Analyst möchte wissen, wie viele Bestellungen verzögert sind, wo sich die betroffenen Kunden befinden und welche Produkte am stärksten betroffen sind. Anstatt die Informationsquellen separat zu untersuchen, verwendet der Analyst Athena, um die Daten in einer einzigen Verbundabfrage zusammenzuführen.

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
```

Verbundansichten abfragen

Bei der Abfrage von Verbundquellen können Sie Ansichten verwenden, um die zugrunde liegenden Datenquellen zu verschleiern oder komplexe Verknüpfungen vor anderen Analysten zu verbergen, die die Daten abfragen.

Überlegungen und Einschränkungen

- Für Verbundansichten ist Athena-Engine-Version 3 erforderlich.
- Verbundene Ansichten werden in AWS Glue und nicht mit der zugrunde liegenden Datenquelle gespeichert.
- Ansichten, die mit Verbundkatalogen erstellt wurden, müssen die Syntax voll qualifizierter Namen verwenden, wie im folgenden Beispiel:

```
"ddbcatalog"."default"."customers"
```

- Benutzer, die Abfragen an Verbundquellen ausführen, müssen über die Berechtigung verfügen, die Verbundquellen abzufragen.
- Die `athena:GetDataCatalog`-Berechtigung ist für Verbundansichten erforderlich. Weitere Informationen finden Sie unter Beispiel für IAM-Berechtigungsrichtlinien zum Zulassen von Athena Federated Query.

Beispiele

Im folgenden Beispiel wird eine Ansicht mit dem Namen `customers` auf Daten erstellt, die in einer Verbunddatenquelle gespeichert sind.

Example

```
CREATE VIEW customers AS
SELECT *
FROM my_federated_source.default.table
```

Die folgende Beispielabfrage zeigt eine Abfrage, die auf die `customers`-Ansicht statt auf die zugrunde liegende Verbunddatenquelle verweist.

Example

```
SELECT id, SUM(order_amount)
FROM customers
GROUP by 1
ORDER by 2 DESC
LIMIT 50
```

Das folgende Beispiel erstellt eine Ansicht namens `order_summary`, die Daten aus einer Verbunddatenquelle und aus einer Amazon-S3-Datenquelle kombiniert. Aus der Verbundquelle, die bereits in Athena erstellt wurde, verwendet die Ansicht die Tabellen `person` und `profile`. In Amazon S3 verwendet die Ansicht die `purchase`- und `payment`-Tabellen. Um auf Amazon S3 zu verweisen, verwendet die Anweisung das Schlüsselwort `awsdatacatalog`. Beachten Sie, dass die föderierte Datenquelle die vollqualifizierte Namenssyntax *`federated_source_name`* verwendet. *`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
```

Weitere Informationen finden Sie auch unter

- Ein Beispiel für eine Verbundansicht, die von ihrer ursprünglichen Quelle entkoppelt ist und für On-Demand-Analysen in einem Mehrbenutzermodell verfügbar ist, finden Sie unter Erweitern Sie Ihr Data Mesh mit Amazon Athena und Verbundansichten im AWS-Big-Data-Blog.
- Weitere Informationen zur Arbeit mit Ansichten in Athena finden Sie unter Arbeiten mit Ansichten.

Qualifizierer für Athena und verbundene Tabellennamen

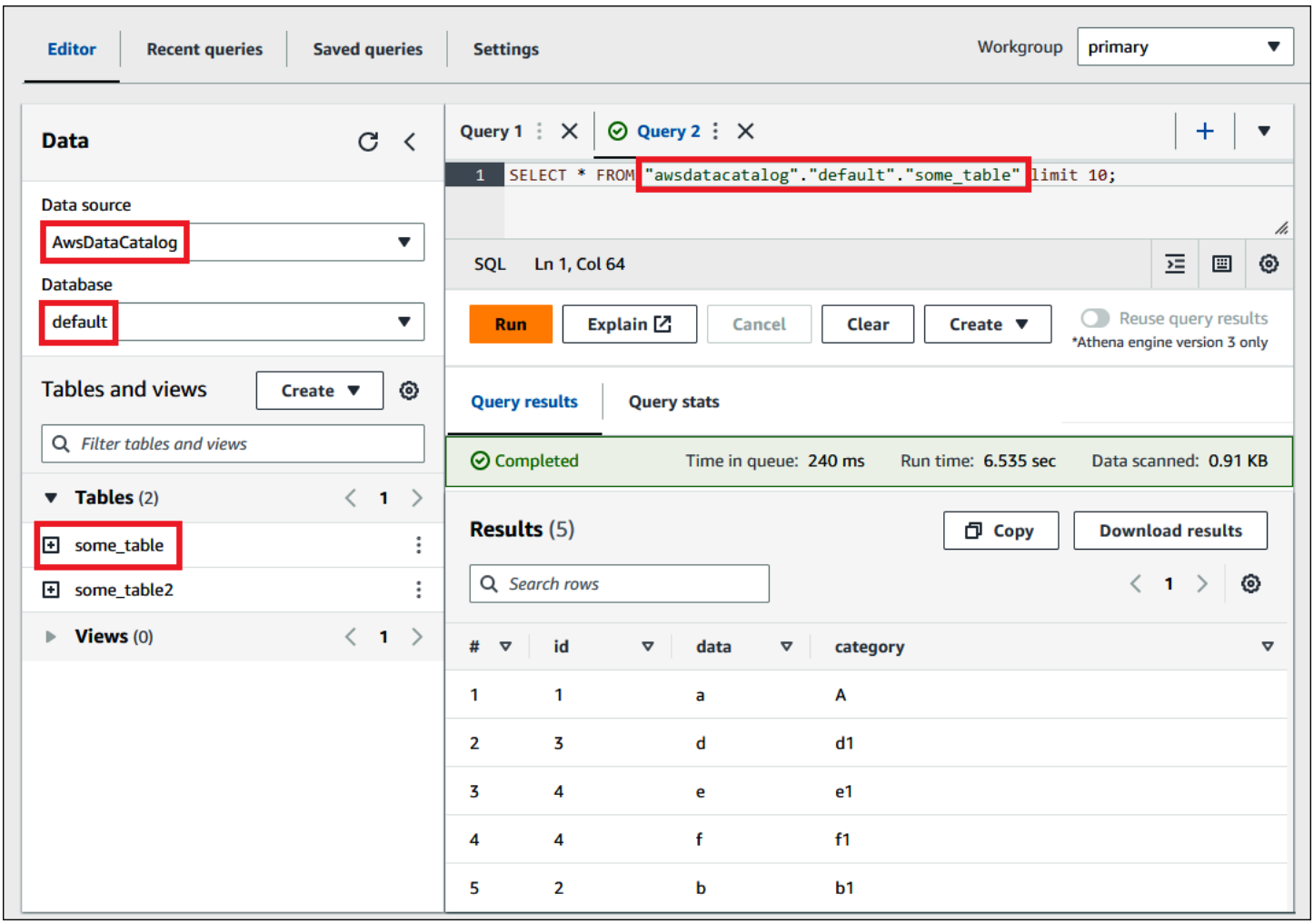
Athena verwendet die folgenden Begriffe, um sich auf Hierarchien von Datenobjekten zu beziehen:

- Datenquelle – eine Gruppe von Datenbanken
- Datenbank – eine Gruppe von Tabellen
- Tabelle – Daten, die als Gruppe von Zeilen oder Spalten organisiert sind

Manchmal werden diese Objekte auch mit alternativen, aber gleichwertigen Namen bezeichnet, z. B. den folgenden:

- Eine Datenquelle wird manchmal auch als Katalog bezeichnet.
- Eine Datenbank wird manchmal auch als Schema bezeichnet.

Die folgende Beispielabfrage in der Athena-Konsole verwendet die `awsdatacatalog`-Datenquelle, die `default`-Datenbank und die `some_table`-Tabelle.



Begriffe in verbundenen Datenquellen

Wenn Sie verbundene Datenquellen abfragen, beachten Sie, dass die zugrunde liegende Datenquelle möglicherweise nicht dieselbe Terminologie wie Athena verwendet. Beachten Sie diesen Unterschied, wenn Sie Ihre Verbundabfragen schreiben. In den folgenden Abschnitten wird beschrieben, wie Datenobjektbegriffe in Athena denen in verbundenen Datenquellen entsprechen.

Amazon Redshift

Eine Datenbank von Amazon Redshift ist eine Gruppe von Redshift-Schemas, die eine Gruppe von Redshift-Tabellen enthält.

Athena	Redshift
Redshift-Datenquelle	Eine Lambda-Funktion des Redshift-Konnektors, die so konfiguriert ist, dass sie auf eine Redshift-`database` verweist.
`data_source.database.table`	`database.schema.table`

Beispielabfrage

```
SELECT * FROM
Athena_Redshift_connector_data_source.Redshift_schema_name.Redshift_table_name
```

Weitere Informationen zu diesem Konnektor finden Sie unter Amazon Athena Redshift Konnektor.

Cloudera Hive

Ein Cloudera-Hive-Server oder -Cluster ist eine Gruppe von Cloudera-Hive-Datenbanken, die eine Gruppe von Cloudera-Hive-Tabellen enthält.

Athena	Hive
Cloudera-Hive-Datenquelle	Lambda-Funktion für Cloudera-Hive-Konnektor konfiguriert, um auf eine Cloudera-Hive-`server` zu zeigen.
`data_source.database.table`	`server.database.table`

Beispielabfrage

```
SELECT * FROM
Athena_Cloudera_Hive_connector_data_source.Cloudera_Hive_database_name.Cloudera_Hive_table_name
```

Weitere Informationen zu diesem Konnektor finden Sie unter Amazon Athena Cloudera Hive Konnektor.

Cloudera Impala

Ein Impala-Server oder -Cluster ist eine Gruppe von Impala-Datenbanken, die eine Gruppe von Impala-Tabellen enthält.

Athena	Impala
Impala-Datenquelle	Die Lambda-Funktion des Impala-Konnektors ist so konfiguriert, dass sie auf einen Impala-`server` zeigt.
`data_source.database.table`	`server.database.table`

Beispielabfrage

```
SELECT * FROM
Athena_Impala_connector_data_source.Impala_database_name.Impala_table_name
```

Weitere Informationen zu diesem Konnektor finden Sie unter Amazon Athena Cloudera Impala Connector.

MySQL

Ein MySQL-Server ist eine Gruppe von MySQL-Datenbanken, die eine Gruppe von MySQL-Tabellen enthält.

Athena	MySQL
MySQL-Datenquelle	Die Lambda-Funktion des MySQL-Konnektors ist so konfiguriert, dass sie auf einen MySQL-`server` zeigt.
`data_source.database.table`	`server.database.table`

Beispielabfrage

```
SELECT * FROM
Athena_MySQL_connector_data source.MySQL_database_name.MySQL_table_name
```

Weitere Informationen zu diesem Konnektor finden Sie unter Amazon Athena MySQL Konnektor.

Oracle

Ein Oracle-Server (oder Datenbank) ist eine Gruppe von Oracle-Schemas, die eine Gruppe von Oracle-Tabellen enthält.

Athena	Oracle
Oracle-Datenquelle	Die Lambda-Funktion des Oracle-Konnektors ist so konfiguriert, dass sie auf einen Oracle-`server` zeigt.
`data_source.database.table`	`server.schema.table`

Beispielabfrage

```
SELECT * FROM
Athena_Oracle_connector_data_source.Oracle_schema_name.Oracle_table_name
```

Weitere Informationen zu diesem Konnektor finden Sie unter Amazon Athena Oracle Konnektor.

Postgres

Ein Postgres-Server (oder -Cluster) ist eine Gruppe von Postgres-Datenbanken. Eine Postgres-Datenbank ist eine Gruppe von Postgres-Schemas, die eine Gruppe von Postgres-Tabellen enthält.

Athena	Postgres
Postgres-Datenquelle	Die Lambda-Funktion des Postgres-Konnektors ist so konfiguriert, dass sie auf ein Postgres-`server` und -`database` zeigt.
`data_source.database.table`	`server.database.schema.table`

Beispielabfrage

```
SELECT * FROM
```

```
Athena_Postgres_connector_data_source.Postgres_schema_name.Postgres_table_name
```

Weitere Informationen zu diesem Konnektor finden Sie unter Amazon Athena PostgreSQL Konnektor.

Schreiben eines Datenquellen-Konnektors mithilfe des Athena Query Federation SDK

Um eigene Datenquellen-Connectors zu schreiben, können Sie den Athena Query Federation SDK verwenden. Der Athena Query Federation SDK definiert einen Satz von Schnittstellen und Wire-Protokollen, mit denen Sie Athena ermöglichen können, Teile des Abfrageausführungsplans an Code zu delegieren, den Sie schreiben und bereitstellen. Das SDK enthält eine Connector-Suite und einen Beispiel-Connector.

Sie können die vorgefertigten Connectors von Amazon Athena auch für Ihren eigenen Gebrauch anpassen. Sie können eine Kopie des Quellcodes aus GitHub ändern und dann das Connector Publish Tool verwenden, um ein eigenes AWS Serverless Application Repository-Paket zu erstellen. Nachdem Sie den Connector auf diese Weise bereitgestellt haben, können Sie ihn in Ihren Athena-Abfragen verwenden.

Informationen zum Herunterladen des SDK und detaillierte Anweisungen zum Schreiben eines eigenen Connectors finden Sie unter Beispiel-Athena-Connector auf GitHub.

Athena-Datenquellenkonnektoren für Apache Spark

Einige Athena-Datenquellenkonnektoren sind als Spark-DSV2-Konnektoren verfügbar. Die Namen der Spark-DSV2-Konnektoren haben ein `-dsv2`-Suffix (z. B. `athena-dynamodb-dsv2`).

Im Folgenden finden Sie die derzeit verfügbaren DSV2-Konnektoren, ihren `.format()`-Spark-Klassennamen und Links zur entsprechenden Amazon-Athena-Verbundabfragen-Dokumentation:

DSV2-Konnektor	Spark .format() class name	Dokumentation
athena-cloudwatch-dsv2	`com.amazonaws.athena.connectors.dsv2.cloudwatch.CloudwatchTableProvider`	CloudWatch
athena-cloudwatch-	`com.amazonaws.athena.connectors.dsv2`	CloudWatch-Metriken

DSV2-Konnektor	Spark .format() class name	Dokumentation
metrics-dsv2	`.cloudwatch.metrics.CloudwatchMetricsTableProvider`	
athena-aws-cmdb-dsv2	`com.amazonaws.athena.connectors.dsv2.aws.cmdb.AwsCmdbTableProvider`	CMDB
athena-dynamodb-dsv2	`com.amazonaws.athena.connectors.dsv2.dynamodb.DDBTableProvider`	DynamoDB

Um `.jar`-Dateien für die DSV2-Konnektoren herunterzuladen, besuchen Sie die GitHub-Seite Amazon-Athena-Verbundabfragen-DSV2 und sehen Sie sich den Abschnitt Releases, Release-***<version>***, Assets an.

Das JAR für Spark angeben

Um die Athena-DSV2-Konnektoren mit Spark zu verwenden, senden Sie die `.jar`-Datei für den Konnektor an die Spark-Umgebung, die Sie verwenden. In den folgenden Abschnitten werden spezifische Fälle beschrieben.

Athena für Spark

Informationen zum Hinzufügen von benutzerdefinierten `.jar`-Dateien und zur benutzerdefinierten Konfiguration zu Amazon Athena für Apache Spark finden Sie unter Hinzufügen von JAR-Dateien und benutzerdefinierte Spark-Konfiguration.

General Spark

Um die `.jar`-Konnektor-Datei an Spark zu übergeben, verwenden Sie den `spark-submit`-Befehl und geben Sie die Datei `.jar` in der Option `--jars` an, wie im folgenden Beispiel:

```
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

Um einen `spark-submit`-Befehl mit dem Parameter `--jars` auf Amazon EMR auszuführen, müssen Sie Ihrem Amazon-EMR-Spark-Cluster einen Schritt hinzufügen. Einzelheiten zur Verwendung von `spark-submit` auf Amazon EMR finden Sie unter Spark-Schritt hinzufügen im Amazon-EMR-Versionshandbuch.

AWS Glue ETL Spark

Für AWS Glue ETL können Sie die GitHub.com-URL der `.jar`-Datei an das `--extra-jars`-Argument des `aws glue start-job-run`-Befehls übergeben. In der AWS Glue-Dokumentation wird beschrieben, dass der Parameter `--extra-jars` einen Amazon-S3-Pfad verwendet, aber der Parameter kann auch eine HTTPS-URL verwenden. Weitere Informationen finden Sie in der Aufgabenparameter-Referenz im AWS Glue-Entwicklerhandbuch.

Den Konnektor auf Spark abfragen

Verwenden Sie die Funktion `spark.sql()`, um das Äquivalent Ihrer vorhandenen Athena-Verbundabfrage auf Apache Spark einzureichen. Nehmen wir beispielsweise an, die folgende Athena-Abfrage wurde mit Apache Spark erstellt.

```
SELECT somecola, somecolb, somecolc
FROM ddb_datasource.some_schema_or_glue_database.some_ddb_or_glue_table
WHERE somecola > 1
```

Verwenden Sie den folgenden Code, um dieselbe Abfrage auf Spark mithilfe des Konnektors von Amazon Athena DynamoDB DSV2 durchzuführen:

```
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
''')
```

Parameter festlegen

Die DSV2-Versionen der Athena-Datenquellenkonnektoren verwenden dieselben Parameter wie die entsprechenden Athena-Datenquellen-Konnektoren. Informationen zu den Parametern finden Sie in der Dokumentation für den entsprechenden Athena-Datenquellenkonnektor.

Verwenden Sie in Ihrem PySpark-Code die folgende Syntax, um Ihre Parameter zu konfigurieren.

```
spark.read.option("athena.connectors.conf.parameter", "value")
```

Der folgende Code setzt zum Beispiel den Parameter des Amazon-Athena-DynamoDB-Konnektors `disable_projection_and_casing` auf `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())
```

IAM-Richtlinien für den Zugriff auf Datenkataloge

Verwenden Sie zur Steuerung des Zugriffs auf Datenkataloge IAM-Berechtigungen auf Ressourcenebene oder identitätsbasierte IAM-Richtlinien.

Das folgende Verfahren gilt speziell für Athena.

IAM-spezifische Informationen finden Sie unter den Links am Ende dieses Abschnitts. Weitere Informationen zu JSON-Datenkatalog-Beispielrichtlinien finden Sie unter Datenkatalog-Beispielrichtlinien.

Verwenden Sie den visuellen Editor in der IAM-Konsole, um eine Datenkatalogrichtlinie zu erstellen

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.

2. Wählen Sie im Navigationsbereich auf der linken Seite Policies (Richtlinien) und dann Create Policy (Richtlinie erstellen) aus.
3. Wählen Sie auf der Registerkarte Visual editor (Visueller Editor) die Option Choose a service (Wählen Sie einen Service) aus. Wählen Sie dann Athena aus, um es der Richtlinie hinzuzufügen.
4. Wählen Sie Select actions (Aktionen auswählen) und dann die Aktionen aus, die Sie der Richtlinie hinzufügen möchten. Im visuellen Editor werden die in Athena verfügbaren Aktionen angezeigt. Weitere Informationen finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Athena in der Service-Autorisierungs-Referenz.
5. Wählen Sie Add actions (Aktionen hinzufügen) aus, um eine bestimmte Aktion einzugeben, oder verwenden Sie Platzhalter (*), um mehrere Aktionen anzugeben.

   Standardmäßig lässt die Richtlinie, die Sie erstellen, die Aktionen zu, die Sie auswählen. Wenn Sie eine oder mehrere Aktionen auswählen, die Berechtigungen auf Ressourcenebene für die `datacatalog`-Ressource in Athena unterstützen, listet der Editor die `datacatalog`-Ressource auf.
6. Wählen Sie Resources (Ressourcen) aus, um die Datenkataloge für Ihre Richtlinie anzugeben. JSON-Datenkatalog-Beispielrichtlinien finden Sie unter Datenkatalog-Beispielrichtlinien.
7. Geben Sie die `datacatalog`-Ressource wie folgt an:

   ```
   arn:aws:athena:<region>:<user-account>:datacatalog/<datacatalog-name>
   ```
8. Wählen Sie Review policy (Richtlinie prüfen) aus und geben Sie Name und Description (Beschreibung) (optional) für die zu erstellende Richtlinie ein. Prüfen Sie die Richtlinienübersicht, um sicherzustellen, dass Sie die beabsichtigten Berechtigungen erteilt haben.
9. Wählen Sie Create policy (Richtlinie erstellen) aus, um Ihre neue Richtlinie zu speichern.
10. Fügen Sie diese identitätsbasierte Richtlinie einem Benutzer, einer Gruppe oder Rolle an und geben Sie die `datacatalog`-Ressourcen an, auf die diese zugreifen dürfen.

Weitere Informationen finden Sie in den folgenden Themen in der Service Authorization Reference und im IAM-Benutzerhandbuch:

- Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Athena
- Erstellen von Richtlinien mit dem visuellen Editor
- Hinzufügen und Entfernen von IAM-Richtlinien

- Steuern des Zugriffs auf Ressourcen

JSON-Datenkatalog-Beispielrichtlinien finden Sie unter Datenkatalog-Beispielrichtlinien.

Weitere Informationen zu AWS Glue-Berechtigungen und AWS Glue-Crawler-Berechtigungen finden Sie unter Einrichten von IAM-Berechtigungen für AWS Glue und Voraussetzungen für Crawler im Entwicklerhandbuch für AWS Glue.

Eine vollständige Liste mit Amazon-Athena-Aktionen finden Sie unter den Namen von API-Aktionen in der Amazon-Athena-API-Referenz.

Datenkatalog-Beispielrichtlinien

In diesem Abschnitt finden Sie Beispielrichtlinien, die Sie verwenden können, um verschiedene Aktionen zu Datenkatalogen zu aktivieren.

Ein Datenkatalog ist eine IAM-Ressource, die von Athena verwaltet wird. Wenn Ihre Datenkatalogrichtlinie Aktionen verwendet, die `datacatalog` als Eingabe verwenden, müssen Sie daher den ARN des Datenkatalogs wie folgt angeben:

```
"Resource": [arn:aws:athena:<region>:<user-account>:datacatalog/<datacatalog-name>]
```

Der `<datacatalog-name>` ist der Name Ihres Datenkatalogs. Geben Sie diesen beispielsweise für einen Datenkatalog mit dem Namen `test_datacatalog` wie folgt als Ressource an:

```
"Resource": ["arn:aws:athena:us-east-1:123456789012:datacatalog/test_datacatalog"]
```

Eine vollständige Liste mit Amazon-Athena-Aktionen finden Sie unter den Namen von API-Aktionen in der Amazon-Athena-API-Referenz. Weitere Informationen zu IAM-Richtlinien finden Sie unter Erstellen von Richtlinien mit dem visuellen Editor im IAM-Benutzerhandbuch. Weitere Informationen zum Erstellen von IAM-Richtlinien für Arbeitsgruppen finden Sie unter IAM-Richtlinien für den Zugriff auf Datenkataloge.

- 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 Beispielrichtlinie für vollständigen Zugriff auf alle Datenkataloge

Die folgende Richtlinie erlaubt den vollständigen Zugriff auf alle möglicherweise in dem Konto vorhandenen Datenkatalogressourcen. Wir empfehlen, diese Richtlinie für die Benutzer in Ihrem Konto zu verwenden, die Datenkataloge für alle anderen Benutzer verwalten müssen.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "athena:*"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

Example Beispielrichtlinie für vollständigen Zugriff auf einen angegebenen Datenkatalog

Die folgende Richtlinie gewährt vollständigen Zugriff auf eine einzelne Datenkatalogressource mit dem Namen `datacatalogA`. Sie können diese Richtlinie für Benutzer mit vollständiger Kontrolle über einen bestimmten Datenkatalog verwenden.

```
{
   "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 Beispielrichtlinie für die Abfrage eines angegebenen Datenkatalogs

In der folgenden Richtlinie ist ein Benutzer berechtigt, Abfragen in der angegebenen `datacatalogA` auszuführen. Der Benutzer darf keine Managementaufgaben für die Arbeitsgruppe selbst durchführen, sie also beispielsweise nicht aktualisieren oder löschen.

```
{
   "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 Beispielrichtlinie für Verwaltungsvorgänge in einem angegebenen Datenkatalog

In der folgenden Richtlinie ist ein Benutzer berechtigt, einen Datenkatalog `datacatalogA` zu erstellen, zu löschen, Details darüber abzurufen und den Datenkatalog zu aktualisieren.

```
{
    "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 Beispielrichtlinie für das Auflisten von Datenkatalogen

Mit der folgenden Richtlinie erhalten alle Benutzer die Berechtigung, alle Datenkataloge aufzulisten:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListDataCatalogs"
            ],
            "Resource": "*"
        }
    ]
}
```

Example Beispielrichtlinie für Metadatenoperationen in Datenkatalogen

Die folgende Richtlinie ermöglicht Metadatenoperationen in Datenkatalogen:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:GetDatabase",
                "athena:GetTableMetadata",
                "athena:ListDatabases",
                "athena:ListTableMetadata"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

Verwalten von Datenquellen

Sie können die Seite Data Sources (Datenquellen) der Athena-Konsole verwenden, um die von Ihnen erstellten Datenquellen zu verwalten.

So zeigen Sie eine Datenquelle an:

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.

3. Klicken Sie im Navigationsbereich auf Data sources (Datenquellen).
4. Wählen Sie aus der Liste der Datenquellen den Namen der Datenquelle aus, die Sie anzeigen möchten.

> **Note**
>
> Die Elemente in der Spalte Data source name (Datenquellenname) entsprechen der Ausgabe der ListDataCatalogs-API-Aktion und des list-data-catalogs-CLI-Befehls.

So bearbeiten Sie eine Datenquelle:

1. Öffnen Sie die Seite Data sources (Datenquellen) und führen Sie eine der folgenden Aktionen aus:
   - Wählen Sie die Schaltfläche neben dem Katalognamen und anschließend Actions (Aktionen), Edit (Bearbeiten) aus.
   - Wählen Sie den Namen der Datenquelle aus. Wählen Sie dann auf der Detailseite Actions (Aktionen), Edit (Bearbeiten) aus.
2. Auf der Seite Edit (Bearbeiten) können Sie eine andere Lambda-Funktion für die Datenquelle auswählen, die Beschreibung ändern oder benutzerdefinierte Tags hinzufügen. Weitere Informationen zu Tags erhalten Sie unter Markieren von Athena-Ressourcen.
3. Wählen Sie Save (Speichern) aus.
4. Um Ihre AwsDataCatalog-Datenquelle zu bearbeiten, wählen Sie den AwsDataCatalog-Link, um die Detailseite zu öffnen. Wählen Sie dann auf der Detailseite den Link zur AWS Glue-Konsole, in der Sie Ihren Katalog bearbeiten können.

So teilen Sie eine Datenquelle

Weitere Informationen zum Teilen von Datenquellen finden Sie unter den folgenden Links.

- Informationen zu Nicht-Hive-Lambda-basierten Datenquellen finden Sie unter Aktivieren von kontoübergreifenden Verbundabfragen.
- Für AWS Glue Data Catalogs, siehe Kontoübergreifender Zugriff auf AWS Glue -Datenkataloge.

So löschen Sie eine Datenquelle:

1. Öffnen Sie die Seite Data sources (Datenquellen) und führen Sie eine der folgenden Aktionen aus:
   - Wählen Sie die Schaltfläche neben dem Katalognamen und anschließend Actions (Aktionen), Delete (Löschen) aus.
   - Wählen Sie den Namen der Datenquelle aus und wählen Sie dann auf der Detailseite Actions (Aktionen), Delete (Löschen) aus.

> **Note**
>
> Die AwsDataCatalog ist die Standarddatenquelle in Ihrem Konto und kann nicht gelöscht werden.

Sie werden gewarnt, dass beim Löschen einer Datenquelle der entsprechende Datenkatalog, Tabellen und Ansichten aus dem Abfrage-Editor entfernt werden. Gespeicherte Abfragen, die die Datenquelle verwendet haben, werden nicht mehr in Athena ausgeführt.

2. Geben Sie zum Bestätigen des Löschvorgangs den Namen der Datenquelle ein und wählen Sie dann Delete (Löschen) aus.

Verwenden von Amazon DataZone in Athena

Sie können Amazon DataZone verwenden, um Daten in großem Maßstab über Unternehmensgrenzen hinweg zu teilen, zu suchen und zu erkunden. DataZone vereinfacht Ihr Erlebnis mit AWS-Analytics-Services wie Athena, AWS Glue und AWS Lake Formation. Wenn Sie beispielsweise Petabyte an Daten in verschiedenen Datenquellen haben, können Sie Amazon DataZone verwenden, um Gruppen von Personen, Daten und Tools zu erstellen, die auf geschäftlichen Anwendungsfällen basieren. Weitere Informationen finden Sie unter Was ist Amazon DataZone?.

In Athena können Sie den Abfrageeditor verwenden, um auf DataZone-Umgebungen zuzugreifen und diese abzufragen. Eine DataZone-Umgebung spezifiziert eine Kombination aus DataZone-Projekt und -Domain. Wenn Sie eine DataZone-Umgebung von der Athena-Konsole aus verwenden, übernehmen Sie die IAM-Rolle der DataZone-Umgebung, und Sie sehen nur die Datenbanken und Tabellen, die zu dieser Umgebung gehören. Die Berechtigungen werden durch die Rollen bestimmt, die Sie in DataZone angeben.

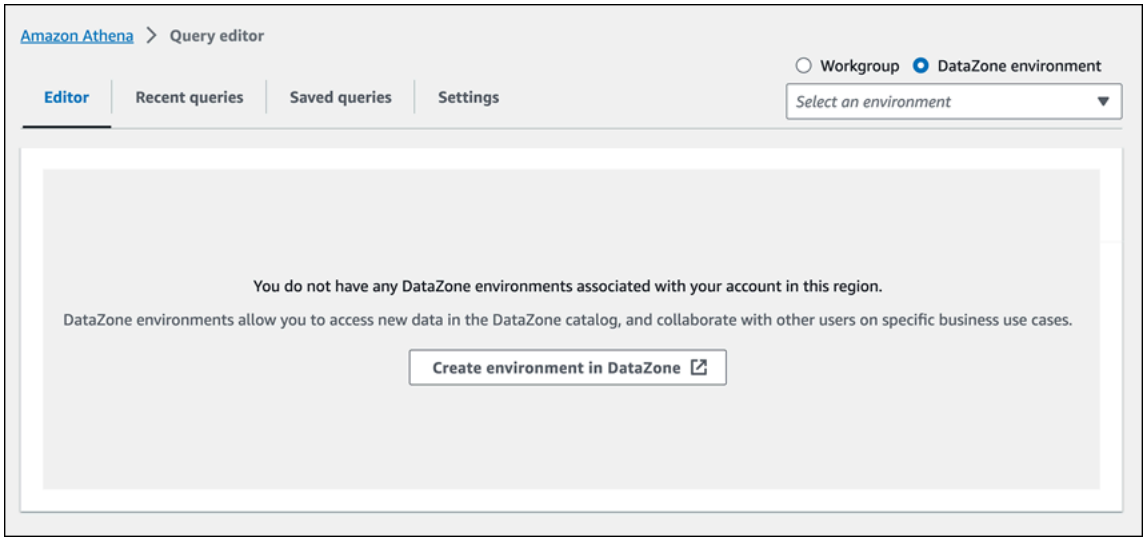
In Athena können Sie die DataZone-Umgebungsauswahl auf der Seite des Abfrageeditors verwenden, um eine DataZone-Umgebung auszuwählen.

So öffnen Sie eine DataZone-Umgebung in Athena

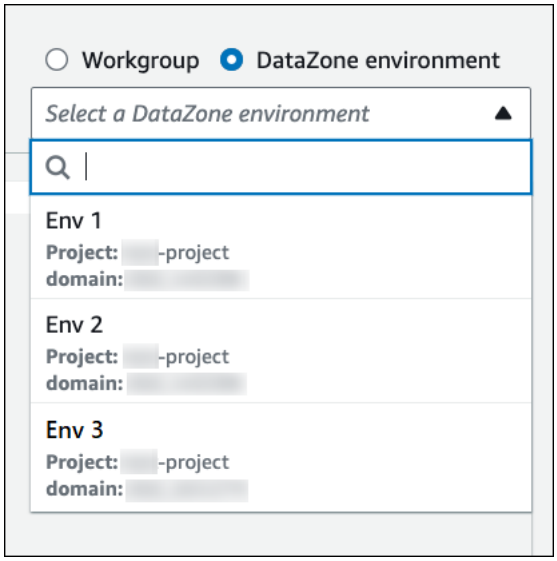
1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.

2. Wählen Sie oben rechts in der Athena-Konsole neben Arbeitsgruppe die Option DataZone-Umgebung.

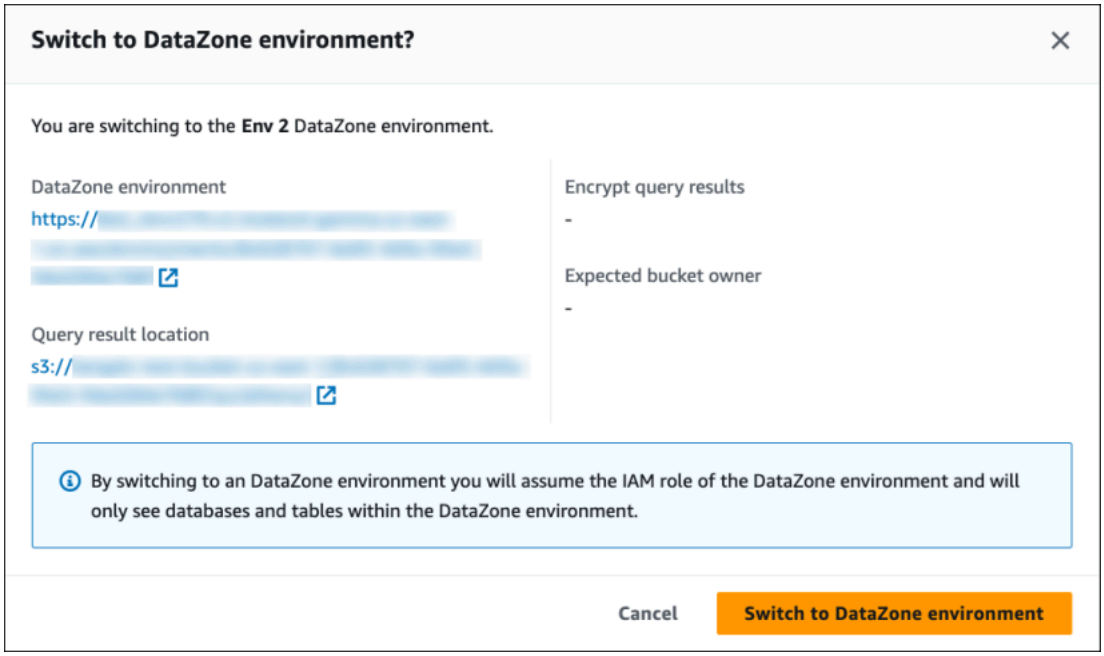
> **Note**
>
> Die Option DataZone-Umgebung ist nur verfügbar, wenn eine oder mehrere Domains in DataZone verfügbar sind.



3. Verwenden Sie die Umgebungsauswahl zur Auswahl einer Umgebung.



4. Vergewissern Sie sich im Dialogfeld Zur DataZone-Umgebung wechseln, dass es sich um die gewünschte Umgebung handelt, und wählen Sie dann Zur DataZone-Umgebung wechseln.



Weitere Informationen zu den ersten Schritten mit DataZone und Athena finden Sie im Tutorial Erste Schritte im Amazon-DataZone-Benutzerhandbuch.

Herstellen einer Verbindung mit Amazon Athena mit ODBC- und JDBC-Treibern

Wenn Sie Ihre Daten mit Business-Intelligence-Tools durchsuchen und darstellen möchten, müssen Sie einen ODBC (Open Database Connectivity)- oder JDBC (Java Database Connectivity)-Treiber herunterladen, installieren und konfigurieren.

Themen

- Herstellen einer Verbindung zu Amazon Athena mit JDBC
- Herstellen einer Verbindung mit Amazon Athena mit ODBC

Weitere Informationen finden Sie in den folgenden AWS-Wissenscenter- und AWS-Big-Data-Blog-Themen:

- Wie kann ich meine IAM-Rollen-Anmeldeinformationen verwenden oder zu einer anderen IAM-Rolle wechseln, wenn ich eine Verbindung mit Athena über den JDBC-Treiber herstelle?
- Einrichten der Vertrauensstellung zwischen ADFS und AWS und Verwenden von Active-Directory-Anmeldeinformationen zum Herstellen einer Verbindung mit Amazon Athena mit ODBC-Treiber

Herstellen einer Verbindung zu Amazon Athena mit JDBC

Amazon Athena bietet zwei JDBC-Treiber, die Versionen 2.x und 3.x. Der Athena-JDBC-3.x-Treiber ist der Treiber der neuen Generation, der eine bessere Leistung und Kompatibilität bietet. Der JDBC-3.x-Treiber unterstützt das Lesen von Abfrageergebnissen direkt aus Amazon S3, was die Leistung von Anwendungen verbessert, die große Abfrageergebnisse verbrauchen. Der neue Treiber hat auch weniger Abhängigkeiten von Drittanbietern, was die Integration mit BI-Tools und benutzerdefinierten Anwendungen erleichtert. In den meisten Fällen können Sie den neuen Treiber ohne oder mit minimalen Änderungen an der vorhandenen Konfiguration verwenden.

- Informationen zum Herunterladen des neuesten JDBC-3.x-Treibers finden Sie unter Athena-JDBC-3.x-Treiber.
- Informationen zum Herunterladen des neuesten JDBC-2.x-Treibers finden Sie unter Athena-JDBC-2.x-Treiber.

Themen

Athena-JDBC-3.x-Treiber

Sie können einen Athena-JDBC-Treiber verwenden, um von vielen SQL-Client-Tools von Drittanbietern und von benutzerdefinierten Anwendungen eine Verbindung zu Amazon Athena herzustellen.

Systemanforderungen

- Laufzeitumgebung Java 8 (oder höher)
- Mindestens 20 MB verfügbarer Speicherplatz

Überlegungen und Einschränkungen

Im Folgenden finden Sie einige Überlegungen und Einschränkungen für den Athena-JDBC-3.x-Treiber.

- Protokollierung – Der 3.x-Treiber verwendet SLF4J, eine Abstraktionsschicht, die die Nutzung eines von mehreren Protokollierungssystemen zur Laufzeit ermöglicht.

- Verschlüsselung – Wenn Sie den S3-Fetcher mit der `CSE_KMS`-Verschlüsselungsoption verwenden, kann der Amazon-S3-Client das im Amazon-S3-Bucket gespeicherte Ergebnis nicht entschlüsseln. Wenn Sie `CSE_KMS`-Verschlüsselung benötigen, können Sie den Streaming-Fetcher weiterhin verwenden. Die Unterstützung der `CSE_KMS`-Verschlüsselung mit dem Amazon-S3-Fetcher ist geplant.

JDBC-3.x-Treiber-Download

Dieser Abschnitt enthält Download- und Lizenzinformationen für den JDBC-3.x-Treiber.

> **Important**
>
> Beachten Sie bei der Verwendung des JDBC-3.x-Treibers unbedingt die folgenden Anforderungen:
>
> - Open port 444 – Halten Sie Port 444, den Athena zum Streamen von Abfrageergebnissen verwendet, für ausgehenden Datenverkehr geöffnet. Wenn Sie einen PrivateLink Endpunkt verwenden, um eine Verbindung zu Athena herzustellen, stellen Sie sicher, dass die mit dem PrivateLink Endpunkt verbundene Sicherheitsgruppe für eingehenden Datenverkehr auf Port 444 geöffnet ist.
> - athena:GetQueryResultsStream policy – Fügen Sie die `athena:GetQueryResultsStream` Richtlinienaktion zu den IAM-Prinzipalen hinzu, die den JDBC-Treiber verwenden. Diese Richtlinienaktion wird nicht direkt mit der API bereitgestellt. Sie wird nur mit dem JDBC-Treiber als Teil der Unterstützung von Streaming-Ergebnissen verwendet. Eine Beispielrichtlinie finden Sie unter AWS Verwaltete Richtlinie: AWSQuicksightAthenaAccess.

Um den 3.x-JDBC-Treiber von Amazon Athena herunterzuladen, klicken Sie auf die folgenden Links.

JDBC-Treiber-uber-jar

Der folgende Download packt den Treiber und all seine Abhängigkeiten in derselben `.jar`-Datei. Dieser Download wird häufig für SQL-Clients von Drittanbietern verwendet.

Uber jar

JDBC-Treiber-lean-jar

Der folgende Download ist eine `.zip`-Datei, die die Lean-`.jar` für den Treiber und separate `.jar`-Dateien für die Abhängigkeiten des Treibers enthält. Dieser Download wird häufig für benutzerdefinierte Anwendungen verwendet, deren Abhängigkeiten möglicherweise mit den vom Treiber verwendeten Abhängigkeiten in Konflikt stehen. Dieser Download ist nützlich, wenn Sie auswählen möchten, welche der Treiberabhängigkeiten in das Lean-Jar-Format aufgenommen und welche ausgeschlossen werden sollen, falls Ihre benutzerdefinierte Anwendung bereits eine oder mehrere davon enthält.

Lean jar

License

Der folgende Link enthält die Lizenzvereinbarung für den JDBC-3.x-Treiber.

Lizenz

Themen

- Erste Schritte mit dem JDBC-3.x-Treiber
- Amazon-Athena-JDBC-3.x-Verbindungsparameter
- Andere JDBC-3.x-Konfigurationen
- Versionshinweise zu Amazon Athena JDBC 3.x

Erste Schritte mit dem JDBC-3.x-Treiber

Verwenden Sie die Informationen in diesem Abschnitt, um mit dem JDBC-3.x-Treiber von Amazon Athena zu beginnen.

Themen

- Installationsanleitungen
- Ausführen des Treibers
- Konfigurieren des Treibers
- Upgrade vom Athena-JDBC-v2-Treiber

Installationsanleitungen

Sie können den JDBC-3.x-Treiber in einer benutzerdefinierten Anwendung oder in einem SQL-Client eines Drittanbieters verwenden.

In einer benutzerdefinierten Anwendung

Laden Sie die `.zip`-Datei herunter, die die Treiber-jar und ihre Abhängigkeiten enthält. Jede Abhängigkeit hat ihre eigene `.jar`-Datei. Fügen Sie in Ihrer benutzerdefinierten Anwendung die Treiber-jar als Abhängigkeit hinzu. Fügen Sie die Abhängigkeiten der Treiber-jar selektiv hinzu, je nachdem, ob Sie diese Abhängigkeiten bereits aus einer anderen Quelle zu Ihrer Anwendung hinzugefügt haben.

In einem SQL-Client eines Drittanbieters

Laden Sie die Treiber-uber-jar herunter und fügen Sie sie dem SQL-Client eines Drittanbieters hinzu, indem Sie den Anweisungen für diesen Client folgen.

Ausführen des Treibers

Verwenden Sie zum Ausführen des Treibers eine benutzerdefinierte Anwendung oder einen SQL-Client eines Drittanbieters.

In einer benutzerdefinierten Anwendung

Verwenden Sie die JDBC-Schnittstelle, um von einem Programm aus mit dem JDBC-Treiber zu interagieren. Der folgende Code zeigt eine benutzerdefinierte Java-Beispielanwendung.

```
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://samplebucket");
    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
}
```

In einem SQL-Client eines Drittanbieters

Folgen Sie der Dokumentation für den SQL-Client, den Sie verwenden. In der Regel verwenden Sie die grafische Benutzeroberfläche des SQL-Clients, um die Abfrage einzugeben und zu senden, und die Abfrageergebnisse werden in derselben Oberfläche angezeigt.

Konfigurieren des Treibers

Sie können Verbindungsparameter verwenden, um den Amazon-Athena-JDBC-Treiber zu konfigurieren. Informationen zu unterstützten Verbindungsparametern finden Sie unter Amazon-Athena-JDBC-3.x-Verbindungsparameter.

In einer benutzerdefinierten Anwendung

Führen Sie einen der folgenden Schritte aus, um die Verbindungsparameter für den JDBC-Treiber in einer benutzerdefinierten Anwendung festzulegen:

- Fügen Sie die Parameternamen und ihre Werte zu einem `Properties`-Objekt hinzu. Wenn Sie `Connection#connect` aufrufen, übergeben Sie dieses Objekt zusammen mit der URL. Ein Beispiel dafür finden Sie in der Java-Beispielanwendung in Ausführen des Treibers.
- Verwenden Sie in der Verbindungszeichenfolge (der URL) das folgende Format, um die Parameternamen und ihre Werte direkt nach dem Protokollpräfix hinzuzufügen.

  ```
  <parameterName>=<parameterValue>;
  ```

  Verwenden Sie am Ende jedes Paars aus Parameternamen und Parameterwerten ein Semikolon und setzen Sie nach dem Semikolon kein Leerzeichen, wie im folgenden Beispiel.

  ```
  String url = "jdbc:athena://WorkGroup=primary;Region=us-east-1;...;";AthenaDriver
   driver = new AthenaDriver();Connection connection = driver.connect(url, null);
  ```

  > **Note**
  >
  > Wenn ein Parameter sowohl in der Verbindungszeichenfolge als auch im `Properties`-Objekt angegeben ist, hat der Wert in der Verbindungszeichenfolge Vorrang. Es wird nicht empfohlen, an beiden Stellen denselben Parameter anzugeben.

- Fügen Sie die Parameterwerte als Argumente zu den Methoden von `AthenaDataSource` hinzu, wie im folgenden Beispiel.

```
AthenaDataSource dataSource = new AthenaDataSource();
    dataSource.setWorkGroup("primary");
    dataSource.setRegion("us-east-2");
    ...
    Connection connection = dataSource.getConnection();
    ...
```

In einem SQL-Client eines Drittanbieters

Folgen Sie den Anweisungen des SQL-Clients, den Sie verwenden. In der Regel bietet der Client eine grafische Benutzeroberfläche zur Eingabe der Parameternamen und ihrer Werte.

Upgrade vom Athena-JDBC-v2-Treiber

Die meisten Verbindungsparameter der JDBC-Version 3 sind abwärtskompatibel mit dem JDBC-Treiber der Version 2 (Simba). Das bedeutet, dass eine Verbindungszeichenfolge der Version 2 mit Version 3 des Treibers wiederverwendet werden kann. Einige Verbindungsparameter haben sich jedoch geändert. Diese Änderungen werden hier beschrieben. Wenn Sie auf den JDBC-Treiber der Version 3 aktualisieren, aktualisieren Sie Ihre vorhandene Konfiguration, falls erforderlich.

Treiberklasse

Bei einigen BI-Tools werden Sie aufgefordert, die Treiberklasse aus der JDBC-Treiber-`.jar`-Datei anzugeben. Die meisten Tools finden diese Klasse automatisch. Der vollständig qualifizierte Name der Klasse im Treiber der Version 3 lautet `com.amazon.athena.jdbc.AthenaDriver`. Im Treiber der Version 2 war die Klasse `com.simba.athena.jdbc.Driver`.

Verbindungszeichenfolge

Der Treiber der Version 3 verwendet `jdbc:athena://` für das Protokoll am Anfang der URL der JDBC-Verbindungszeichenfolge. Der Treiber der Version 3 unterstützt auch das Protokoll der Version 2`jdbc:awsathena://`, aber die Verwendung des Protokolls der Version 2 ist veraltet. Um undefiniertes Verhalten zu vermeiden, akzeptiert Version 3 keine Verbindungszeichenfolgen, die mit beginnen, `jdbc:awsathena://` wenn Version 2 (oder ein anderer Treiber, der Verbindungszeichenfolgen akzeptiert, die mit beginnen`jdbc:awsathena://`) bei der DriverManager Klasse registriert wurde.

Anmeldeinformationsanbieter

Der Treiber der Version 2 verwendet vollqualifizierte Namen, um verschiedene Anbieter von Anmeldeinformationen zu identifizieren, z. B. `com.simba.athena.amazonaws.auth.DefaultAWSCredentialsProviderChain`. Der Treiber der Version 3 verwendet kürzere Namen, z. B. `DefaultChain`. Die neuen Namen werden in den entsprechenden Abschnitten für jeden Anbieter von Anmeldeinformationen beschrieben.

Anbieter für benutzerdefinierte Anmeldeinformationen, die für den Treiber der Version 2 geschrieben wurden, müssen geändert werden, damit der Treiber der Version 3 die AwsCredentialsProvider Schnittstelle von der neuen AWS SDK for Java anstelle der AWSCredentialsProvider Schnittstelle von der alten implementieren kann AWS SDK for Java.

Protokollebene

Die folgende Tabelle zeigt die Unterschiede in den `LogLevel`-Parametern in den JDBC-Treibern Version 2 und Version 3.

JDBC-Treiberversion	Parametername	Parametertyp	Standardwert	Mögliche Werte	Beispiel für Verbindungszeichenfolgen
v2	`LogLevel`	Optional	0	0-6	`LogLevel=6;`
v3	`LogLevel`	Optional	TRACE	OFF, ERROR, WARN, INFO, DEBUG, TRACE	`LogLevel=INFO;`

Abrufen der Abfrage-ID

Im Treiber der Version 2 entpacken Sie eine `Statement`-Instance nach `com.interfaces.core.IStatementQueryInfoProvider`, eine Schnittstelle, die über zwei Methoden verfügt: `#getPReparedQueryId` und `#getQueryId`. Sie können diese Methoden verwenden, um die Abfrageausführungs-ID einer Abfrage abzurufen, die ausgeführt wurde.

Im Treiber der Version 3 entpacken Sie die Instances `Statement`, `PreparedStatement`, und `ResultSet` in die Schnittstelle `com.amazon.athena.jdbc.AthenaResultSet`. Die Schnittstelle hat eine Methode: `#getQueryExecutionId`.

Amazon-Athena-JDBC-3.x-Verbindungsparameter

Unterstützte Verbindungsparameter sind hier in drei Abschnitte unterteilt: Grundlegende Verbindungsparameter, Erweiterte Verbindungsparameter und Authentifizierungsverbindungsparameter. Die Abschnitte „Erweiterte Verbindungsparameter“ und „Authentifizierungsparameter“ enthalten Unterabschnitte, in denen verwandte Parameter zusammengefasst sind.

Themen

Grundlegende Verbindungsparameter

In den folgenden Abschnitten werden die grundlegenden Verbindungsparameter für den JDBC-3.x-Treiber beschrieben.

Region

Die AWS-Region , in der Abfragen ausgeführt werden. Eine Liste der unterstützten Regionen finden Sie unter Endpunkte und Kontingente von Amazon Athena.

Parameter name	Alias	Parametertyp	Standardwert
Region	AwsRegion (veraltet)	Obligatorisch (wird jedoch, falls nicht angegeben, mit der durchsuchtDefaultAwsRegionProviderChain)	Keine

Katalog

Der Katalog, der die Datenbanken und Tabellen enthält, auf die mit dem Treiber zugegriffen wird. Weitere Informationen zu Katalogen finden Sie unter DataCatalog.

Parametername	Alias	Parametertyp	Standardwert
Katalog	Keine	Optional	AwsDataCatalog

Datenbank

Die Datenbank, in der Abfragen ausgeführt werden. Tabellen, die nicht explizit mit einem Datenbanknamen qualifiziert sind, werden in diese Datenbank aufgelöst. Hinweise zu Datenbanken finden Sie unter Datenbank.

Parametername	Alias	Parametertyp	Standardwert
Datenbank	Schema	Optional	default

Arbeitsgruppe

Die Arbeitsgruppe, in der Abfragen ausgeführt werden. Weitere Informationen zu Arbeitsgruppen finden Sie unter WorkGroup.

Parametername	Alias	Parametertyp	Standardwert
WorkGroup	Keine	Optional	primary

Ausgabeort

Der Ort in Amazon S3, an dem die Abfrageergebnisse gespeichert werden. Informationen zum Ausgabespeicherort finden Sie unter ResultConfiguration.

Parametername	Alias	Parametertyp	Standardwert
OutputLocation	S3OutputLocation (veraltet)	Obligatorisch (es sei denn, die Arbeitsgruppe gibt einen Ausgabespeicherort an)	Keine

Erweiterte Verbindungsparameter

In den folgenden Abschnitten werden die erweiterten Verbindungsparameter für den JDBC-3.x-Treiber beschrieben.

Themen

Ergebnisverschlüsselungsparameter

Beachten Sie folgende Punkte:

- Der AWS KMS Schlüssel muss angegeben werden, wenn `SSE_KMS` oder `EncryptionOption` ist`CSE_KMS`.
- Der AWS KMS Schlüssel kann nicht angegeben werden, wenn er `EncryptionOption` nicht angegeben `EncryptionOption` ist oder wann`SSE_S3`.

Verschlüsselungsoption

Die Art der Verschlüsselung, die für Abfrageergebnisse verwendet werden soll, wenn sie in Amazon S3 gespeichert werden. Informationen zur Verschlüsselung von Abfrageergebnissen finden Sie EncryptionConfigurationin der Amazon Athena API-Referenz.

Parametername	Alias	Parametertyp	Standardwert	Mögliche Werte
EncryptionOption	S3 OutputEnc Option (veraltet)	Optional	Keine	SSE_S3, SSE_KMS, CSE_KMS

KMS-Schlüssel

Der KMS-Schlüssel-ARN oder die -ID, falls `SSE_KMS` oder `CSE_KMS`, wird als Verschlüsselungsoption ausgewählt. Weitere Informationen finden Sie EncryptionConfigurationin der Amazon Athena API-Referenz.

Parametername	Alias	Parametertyp	Standardwert
KmsKey	S3 OutputEnc KMSKey (veraltet)	Optional	Keine

Parameter zum Abrufen von Ergebnissen

Ergebnis-Fetcher

Der Fetcher, der zum Herunterladen von Abfrageergebnissen verwendet wird.

Der standardmäßige Ergebnis-Fetcher, `S3`, lädt Abfrageergebnisse direkt von Amazon S3 herunter, ohne die Athena-APIs zu verwenden. Dies ist in den meisten Fällen die schnellste Option. Diese Option ist nicht verfügbar, wenn Ihre Abfrageergebnisse mit `CSE_KMS` verschlüsselt sind oder wenn die Richtlinie, die dem Benutzer den Zugriff auf Abfrageergebnisse ermöglicht, nur Anrufe von Athena mit `s3:CalledVia` erlaubt.

Parametername	Alias	Parametertyp	Standardwert	Mögliche Werte
ResultFetcher	Keine	Optional	S3	GetQueryResultsS3, GetQueryResultsStream

> **Note**
>
> Im JDBC 2.x-Treiber konfiguriert die `UseResultsetStreaming = 1` Einstellung den Treiber so, dass er die Result Set-Streaming-API verwendet. Im JDBC 3.x-Treiber lautet die entsprechende Einstellung. `ResultFetcher=GetQueryResultsStream`

Abrufgröße

Der Wert dieses Parameters wird als Minimum für interne Puffer und als Zielseitengröße beim Abrufen von Ergebnissen verwendet. Der Wert 0 (Null) bedeutet, dass der Treiber seine Standardwerte wie unten beschrieben verwenden soll. Der maximale Wert beträgt 1 000 000.

Parametername	Alias	Parametertyp	Standardwert
FetchSize	RowsToFetchPerBlock (veraltet)	Optional	0

- Der `GetQueryResults`-Fetcher verwendet immer eine Seitengröße von 1 000. Dies ist der maximale Wert, der vom API-Aufruf unterstützt wird. Wenn die Abrufgröße höher als 1 000 ist, werden mehrere aufeinanderfolgende API-Aufrufe ausgeführt, um den Puffer über dem Mindestwert zu füllen.
- Der `GetQueryResultsStream`-Fetcher verwendet die konfigurierte Abrufgröße als Seitengröße oder standardmäßig 10 000.
- Der `S3`-Fetcher verwendet die konfigurierte Abrufgröße als Seitengröße oder standardmäßig 10 000.

Parameter für die Wiederverwendung von Abfrageergebnissen

Wiederverwendung von Ergebnissen aktivieren

Gibt an, ob frühere Ergebnisse für dieselbe Abfrage wiederverwendet werden können, wenn eine Abfrage ausgeführt wird. Hinweise zur Wiederverwendung von Abfrageergebnissen finden Sie unter. ResultReuseByAgeConfiguration

Parametername	Alias	Parametertyp	Standardwert
EnableResultReuseByAge	Keine	Optional	FALSE

Höchstalter für die Wiederverwendung von Ergebnissen

Das maximale Alter eines früheren Abfrageergebnisses in Minuten, das Athena für die Wiederverwendung berücksichtigen soll. Hinweise zur maximalen Wiederverwendung von Ergebnissen finden Sie unter ResultReuseByAgeConfiguration.

Parametername	Alias	Parametertyp	Standardwert
MaxResultReuseAgeInMinutes	Keine	Optional	60

Abfrageparameter für die Abfrageausführung

Minimales Abfrageintervall für die Ausführung von Abfragen

Die minimale Zeit in Millisekunden, die gewartet werden muss, bevor Athena nach dem Status der Abfrageausführung gefragt wird.

Parametername	Alias	Parametertyp	Standardwert
MinQueryExecutionPollingIntervalMillis	MinQueryExecutionPollingInterval (veraltet)	Optional	5

Maximales Abfrageintervall für die Ausführung von Abfragen

Die maximale Zeit in Millisekunden, die gewartet werden muss, bevor Athena nach dem Status der Abfrageausführung gefragt wird.

Parametername	Alias	Parametertyp	Standardwert
MaxQueryExecutionPollingIntervalMillis	MaxQueryExecutionPollingInterval (veraltet)	Optional	5000

Multiplikator für das Abfrageintervall bei der Ausführung von Abfragen

Der Faktor für die Verlängerung des Abfragezeitraums. Standardmäßig beginnt die Abfrage mit dem Wert für `MinQueryExecutionPollingIntervalMillis` und verdoppelt sich bei jeder Abfrage, bis der Wert für `MaxQueryExecutionPollingIntervalMillis` erreicht ist.

Parametername	Alias	Parametertyp	Standardwert
QueryExecutionPollingIntervalMultiplier	Keine	Optional	2

Parameter für Endpunktüberschreibung

Athena-Endpunktüberschreibung

Der Endpunkt, den der Treiber für API-Aufrufe an Athena verwendet.

Beachten Sie folgende Punkte:

- Wenn die Protokolle `https://` oder `http://` in der angegebenen URL nicht angegeben sind, fügt der Treiber das Präfix `https://` ein.
- Wenn dieser Parameter nicht angegeben ist, verwendet der Treiber einen Standardendpunkt.

Parametername	Alias	Parametertyp	Standardwert
AthenaEndpoint	EndpointOverride (veraltet)	Optional	Keine

Athena-Streaming-Service-Endpunktüberschreibung

Der Endpunkt, den der Treiber zum Herunterladen von Abfrageergebnissen verwendet, wenn er den Athene-Streaming-Service verwendet. Der Athena-Streaming-Service ist über Port 444 verfügbar.

Beachten Sie folgende Punkte:

- Wenn die Protokolle `https://` oder `http://` in der angegebenen URL nicht angegeben sind, fügt der Treiber das Präfix `https://` ein.
- Wenn in der angegebenen URL kein Port angegeben ist, fügt der Treiber den Streaming-Service-Port 444 ein.
- Wenn der `AthenaStreamingEndpoint`-Parameter nicht angegeben ist, verwendet der Treiber die `AthenaEndpoint`-Überschreibung. Wenn weder die Überschreibung `AthenaStreamingEndpoint` noch `AthenaEndpoint` angegeben sind, verwendet der Treiber einen Standard-Streaming-Endpunkt.

Parametername	Alias	Parametertyp	Standardwert
AthenaStreamingEndpoint	StreamingEndpointOverride (veraltet)	Optional	Keine

LakeFormation Endpunkt überschreiben

Der Endpunkt, den der Treiber für den Lake Formation Formation-Dienst verwendet, wenn er die AWS Lake Formation AssumeDecoratedRoleWithSAML-API zum Abrufen temporärer Anmeldeinformationen verwendet. Wenn dieser Parameter nicht angegeben ist, verwendet der Treiber einen Lake-Formation-Standardendpunkt.

Beachten Sie folgende Punkte:

- Wenn die Protokolle `https://` oder `http://` in der angegebenen URL nicht angegeben sind, fügt der Treiber das Präfix `https://` ein.

Parametername	Alias	Parametertyp	Standardwert
LakeFormationEndpoint	LfEndpointOverride (veraltet)	Optional	Keine

Überschreibung von S3-Endpunkten

Der Endpunkt, den der Treiber zum Herunterladen von Abfrageergebnissen verwendet, wenn er den Amazon-S3-Fetcher verwendet. Wenn dieser Parameter nicht angegeben ist, verwendet der Treiber einen standardmäßigen Amazon-S3-Endpunkt.

Beachten Sie folgende Punkte:

- Wenn die Protokolle `https://` oder `http://` in der angegebenen URL nicht angegeben sind, fügt der Treiber das Präfix `https://` ein.

Parametername	Alias	Parametertyp	Standardwert
S3Endpoint	None	Optional	Keine

Überschreibung von STS-Endpunkten

Der Endpunkt, den der Treiber für den AWS STS Dienst verwendet, wenn er die AWS STS AssumeRoleWithSAML-API zum Abrufen temporärer Anmeldeinformationen verwendet. Wenn dieser Parameter nicht angegeben ist, verwendet der Treiber einen AWS STS Standardendpunkt.

Beachten Sie folgende Punkte:

- Wenn die Protokolle `https://` oder `http://` in der angegebenen URL nicht angegeben sind, fügt der Treiber das Präfix `https://` ein.

Parametername	Alias	Parametertyp	Standardwert
StsEndpoint	StsEndpointOverride(veraltet)	Optional	Keine

Proxy-Konfigurationsparameter

Proxy-Host

Die URL des Proxy-Hosts. Verwenden Sie diesen Parameter, wenn Athena-Anfragen über einen Proxy laufen sollen.

> **Note**
>
> Stellen Sie sicher, dass Sie das Protokoll `https://` oder `http://` am Anfang der URL für `ProxyHost` angeben.

Parametername	Alias	Parametertyp	Standardwert
ProxyHost	Keine	Optional	Keine

Proxy-Port

Der Port, der auf dem Proxy-Host verwendet werden soll. Verwenden Sie diesen Parameter, wenn Athena-Anfragen über einen Proxy laufen sollen.

Parametername	Alias	Parametertyp	Standardwert
ProxyPort	Keine	Optional	Keine

Proxy-Benutzername

Der Benutzername für die Authentifizierung mit dem Proxy-Server. Verwenden Sie diesen Parameter, wenn Athena-Anfragen über einen Proxy laufen sollen.

Parametername	Alias	Parametertyp	Standardwert
ProxyUsername	ProxyUID (veraltet)	Optional	Keine

Proxy-Passwort

Das Passwort für die Authentifizierung mit dem Proxy-Server. Verwenden Sie diesen Parameter, wenn Athena-Anfragen über einen Proxy laufen sollen.

Parametername	Alias	Parametertyp	Standardwert
ProxyPassword	ProxyPWD (veraltet)	Optional	Keine

Hosts ohne Proxy

Eine Gruppe von Hostnamen, zu denen der Treiber eine Verbindung herstellt, ohne einen Proxy zu verwenden, wenn die Proxyfunktion aktiviert ist (d. h. wenn die Verbindungsparameter `ProxyHost` und `ProxyPort` festgelegt sind). Die Hosts sollten durch das Pipe-Zeichen (`|`) getrennt werden (z. B. `host1.com|host2.com`).

Parametername	Alias	Parametertyp	Standardwert
ProxyExemptHosts	NonProxyHosts	Optional	Keine

Proxy für Identitätsanbieter aktiviert

Gibt an, ob ein Proxy verwendet werden soll, wenn der Treiber eine Verbindung zu einem Identitätsanbieter herstellt.

Parametername	Alias	Parametertyp	Standardwert
ProxyEnabledForIdP	UseProxyForIdP	Optional	FALSE

Protokollieren von Parametern

In diesem Abschnitt werden Parameter im Zusammenhang mit der Protokollierung beschrieben.

Protokollebene

Gibt die Ebene für die Treiberprotokollierung an. Es wird nichts protokolliert, es sei denn, der Parameter `LogPath` ist ebenfalls festgelegt.

> **Note**
>
> Wir empfehlen, nur den Parameter `LogPath` festzulegen, sofern Sie keine besonderen Anforderungen haben. Wenn Sie nur den Parameter `LogPath` festlegen, wird die Protokollierung aktiviert und die `TRACE`-Standardprotokollebene verwendet. Die Protokollebene `TRACE` bietet die detaillierteste Protokollierung.

Parametername	Alias	Parametertyp	Standardwert	Mögliche Werte
LogLevel	Keine	Optional	TRACE	OFF, ERROR, WARN, INFO, DEBUG, TRACE

Protokollpfad

Der Pfad zu einem Verzeichnis auf dem Computer, auf dem der Treiber ausgeführt wird, in dem Treiberprotokolle gespeichert werden. Eine Protokolldatei mit einem eindeutigen Namen wird im angegebenen Verzeichnis erstellt. Wenn diese Option festgelegt ist, wird die Treiberprotokollierung aktiviert.

Parametername	Alias	Parametertyp	Standardwert
LogPath	Keine	Optional	Keine

Anwendungsname

Der Name der Anwendung, die den Treiber verwendet. Wenn ein Wert für diesen Parameter angegeben wird, ist der Wert in der User-Agent-Zeichenfolge der API-Aufrufe enthalten, die der Treiber an Athena sendet.

> **Note**
>
> Sie können den Namen der Anwendung auch festlegen, indem Sie `setApplicationName` im Objekt `DataSource` aufrufen.

Parametername	Alias	Parametertyp	Standardwert
ApplicationName	Keine	Optional	Keine

Verbindungstest

Wenn auf TRUE festgelegt, führt der Treiber jedes Mal, wenn eine JDBC-Verbindung hergestellt wird, einen Verbindungstest durch, auch wenn keine Abfrage für die Verbindung ausgeführt wird.

Parametername	Alias	Parametertyp	Standardwert
ConnectionTest	Keine	Optional	TRUE

> **Note**
>
> Ein Verbindungstest sendet eine `SELECT 1`-Anfrage an Athena, um zu überprüfen, ob die Verbindung korrekt konfiguriert wurde. Das bedeutet, dass zwei Dateien (der Ergebnissatz und die Metadaten) in Amazon S3 gespeichert werden und zusätzliche Gebühren gemäß der Amazon-Athena-Preisrichtlinie anfallen können.

Anzahl der Wiederholungen

Gibt an, wie oft der Treiber maximal eine wiederholbare Anforderung an Athena senden sollte.

Parametername	Alias	Parametertyp	Standardwert
NumRetries	MaxErrorRetry (veraltet)	Optional	Keine

Authentifizierungsverbindungsparameter

Der Athena-JDBC-3.x-Treiber unterstützt mehrere Authentifizierungsmethoden. Die erforderlichen Verbindungsparameter hängen von der verwendeten Authentifizierungsmethode ab.

Themen

- IAM-Anmeldeinformationen
- Standard-Anmeldeinformationen
- AWS -Anmeldeinformationen eines Konfigurationsprofils
- Anmeldeinformationen eines Instance-Profils
- Benutzerdefinierte Anmeldeinformationen
- JWT-Anmeldeinformationen
- Azure-AD-Anmeldeinformationen
- Okta-Anmeldeinformationen
- Ping-Anmeldeinformationen
- AD-FS-Anmeldeinformationen
- Anmeldeinformationen für Browser Azure AD
- Browser-SAML-Anmeldeinformationen

IAM-Anmeldeinformationen

Sie können Ihre IAM-Anmeldeinformationen mit dem JDBC-Treiber verwenden, um eine Verbindung mit Amazon Athena herzustellen, indem Sie die folgenden Verbindungsparameter festlegen.

Benutzer

Ihre AWS Zugriffsschlüssel-ID. Weitere Informationen zu Zugriffsschlüsseln finden Sie unter AWS - Sicherheitsanmeldeinformationen im IAM-Benutzerhandbuch.

Parametername	Alias	Parametertyp	Standardwert
Benutzer	AccessKeyId	Erforderlich	Keine

Passwort

Ihre AWS geheime Schlüssel-ID. Weitere Informationen zu Zugriffsschlüsseln finden Sie unter AWS - Sicherheitsanmeldeinformationen im IAM-Benutzerhandbuch.

Parametername	Alias	Parametertyp	Standardwert
Passwort	SecretAccessKey	Optional	Keine

Sitzungs-Token

Wenn Sie temporäre AWS Anmeldeinformationen verwenden, müssen Sie ein Sitzungstoken angeben. Weitere Informationen zu temporären Anmeldeinformationen finden Sie unter Temporäre Sicherheitsanmeldeinformationen in IAM im IAM-Benutzerhandbuch.

Parametername	Alias	Parametertyp	Standardwert
SessionToken	Keine	Optional	Keine

Standard-Anmeldeinformationen

Sie können die Standardanmeldeinformationen verwenden, die Sie auf Ihrem Client-System konfigurieren, um eine Verbindung zu Amazon Athena herzustellen, indem Sie die folgenden Verbindungsparameter festlegen. Informationen zur Verwendung von Standard-Anmeldeinformationen finden Sie unter Verwendung der Standard-Anbieterkette für Anmeldeinformationen im AWS SDK for Java -Entwicklerhandbuch.

Anmeldeinformationsanbieter

Der Anbieter für Anmeldeinformationen, der zur Authentifizierung von Anforderungen an AWSverwendet wird. Stellen Sie den Wert dieses Parameters auf `DefaultChain` ein.

Parametername	Alias	Parametertyp	Standardwert	Zu verwenden der Wert
CredentialsProvider	AWSCredentialsProviderClass (veraltet)	Erforderlich	Keine	`DefaultChain`

AWS -Anmeldeinformationen eines Konfigurationsprofils

Sie können Anmeldeinformationen verwenden, die in einem - AWS Konfigurationsprofil gespeichert sind, indem Sie die folgenden Verbindungsparameter festlegen. AWS -Konfigurationsprofile werden normalerweise in Dateien im -`~/.aws`Verzeichnis gespeichert). Informationen zu AWS -Konfigurationsprofilen finden Sie unter Verwenden von Profilen im AWS SDK for Java -Entwicklerhandbuch.

Anmeldeinformationsanbieter

Der Anbieter für Anmeldeinformationen, der zur Authentifizierung von Anforderungen an AWSverwendet wird. Stellen Sie den Wert dieses Parameters auf `ProfileCredentials` ein.

Parametername	Alias	Parametertyp	Standardwert	Zu verwendender Wert
CredentialsProvider	AWSCredentialsProviderClass (veraltet)	Erforderlich	Keine	`ProfileCredentials`

Profilname

Der Name des AWS Konfigurationsprofils, dessen Anmeldeinformationen zur Authentifizierung der Anfrage an Athena verwendet werden sollen.

Parametername	Alias	Parametertyp	Standardwert
ProfileName	Keine	Erforderlich	Keine

> **Note**
>
> Der Profilname kann auch als Wert des `CredentialsProviderArguments`-Parameters angegeben werden, obwohl diese Verwendung veraltet ist.

Anmeldeinformationen eines Instance-Profils

Dieser Authentifizierungstyp wird in Amazon-EC2-Instances verwendet. Ein einer Amazon-EC2-Instance hinzugefügtes Instance-Profil. Die Verwendung eines Anbieters für Instance-Profilanmeldeinformationen delegiert die Verwaltung von AWS Anmeldeinformationen an den Amazon EC2 Instance Metadata Service. Dadurch müssen Entwickler keine Anmeldeinformationen dauerhaft auf der Amazon-EC2-Instance speichern oder sich Gedanken über die Rotation oder Verwaltung temporärer Anmeldeinformationen machen.

Anmeldeinformationsanbieter

Der Anbieter für Anmeldeinformationen, der zur Authentifizierung von Anforderungen an AWSverwendet wird. Stellen Sie den Wert dieses Parameters auf `InstanceProfile` ein.

Parametername	Alias	Parametertyp	Standardwert	Zu verwenden der Wert
CredentialsProvider	AWSCredentialsProviderClass (veraltet)	Erforderlich	Keine	`InstanceProfile`

Benutzerdefinierte Anmeldeinformationen

Sie können diesen Authentifizierungstyp verwenden, um Ihre eigenen Anmeldeinformationen bereitzustellen, indem Sie eine Java-Klasse verwenden, die die AwsCredentialsProvider Schnittstelle implementiert.

Anmeldeinformationsanbieter

Der Anbieter für Anmeldeinformationen, der zur Authentifizierung von Anforderungen an AWSverwendet wird. Legen Sie den Wert dieses Parameters auf den vollqualifizierten Klassennamen der benutzerdefinierten Klasse fest, die die AwsCredentialsProvider Schnittstelle implementiert. Zur Laufzeit muss sich diese Klasse im Java-Klassenpfad der Anwendung befinden, die den JDBC-Treiber verwendet.

Parametername	Alias	Parametertyp	Standardwert	Zu verwendender Wert
CredentialsProvider	AWSCredentialsProviderClass (veraltet)	Erforderlich	Keine	Der vollqualifizierte Klassenname der benutzerdefinierten Implementierung von `AwsCredentialsProvider`

Argumente für den Anmeldeinformationsanbieter

Eine durch Komma getrennte Liste der Zeichenkettenargumente für die benutzerdefinierten Anmeldeinformationen.

Parametername	Alias	Parametertyp	Standardwert
CredentialsProviderArguments	AwsCredentialsProviderArguments (veraltet)	Optional	Keine

JWT-Anmeldeinformationen

Mit diesem Authentifizierungstyp können Sie ein von einem externen Identitätsanbieter abgerufenes JSON-Webtoken (JWT) als Verbindungsparameter für die Authentifizierung bei Athena verwenden. Der externe Anmeldeinformationsanbieter muss bereits mit verbunden sein AWS.

Anmeldeinformationsanbieter

Der Anbieter für Anmeldeinformationen, der zur Authentifizierung von Anforderungen an AWSverwendet wird. Stellen Sie den Wert dieses Parameters auf `JWT` ein.

Parametername	Alias	Parametertyp	Standardwert	Zu verwenden der Wert
CredentialsProvider	AWSCredentialsProviderClass (veraltet)	Erforderlich	Keine	`JWT`

JWT-Web-Identitätstoken

Das JWT-Token, das von einem externen föderierten Identitätsanbieter abgerufen wurde. Dieses Token wird zur Authentifizierung bei Athena verwendet.

Parametername	Alias	Parametertyp	Standardwert
JwtWebIdentityToken	web_identity_token (veraltet)	Erforderlich	Keine

JWT-Rollen-ARN

Der angenommene Amazon-Ressourcenname (ARN) der Rolle. Informationen zur Übernahme von Rollen finden Sie unter AssumeRole in der APIAWS Security Token Service -Referenz zu .

Parametername	Alias	Parametertyp	Standardwert
JwtRoleArn	role_arn (veraltet)	Erforderlich	Keine

JWT-Rollensitzungsname

Der Name der Sitzung, wenn Sie JWT-Anmeldeinformationen für die Authentifizierung verwenden. Der Name kann ein beliebiger Name sein, den Sie wählen.

Parametername	Alias	Parametertyp	Standardwert
JwtRoleSessionName	role_session_name (veraltet)	Erforderlich	Keine

Azure-AD-Anmeldeinformationen

Ein SAML-basierter Authentifizierungsmechanismus, der die Authentifizierung bei Athena mit dem Azure-AD-Identitätsanbieter ermöglicht. Bei dieser Methode wird davon ausgegangen, dass bereits ein Verbund zwischen Athena und Azure AD eingerichtet wurde.

> **Note**
>
> Einige der Parameternamen in diesem Abschnitt haben Aliase. Die Aliase sind funktionale Äquivalente der Parameternamen und wurden aus Gründen der Abwärtskompatibilität mit dem JDBC 2.x-Treiber bereitgestellt. Da die Parameternamen verbessert wurden, sodass sie

einer klareren und einheitlicheren Benennungskonvention folgen, empfehlen wir, sie anstelle der Aliase zu verwenden, die abgeschafft wurden.

Anmeldeinformationsanbieter

Der Anbieter für Anmeldeinformationen, der zur Authentifizierung von Anforderungen an AWSverwendet wird. Stellen Sie den Wert dieses Parameters auf `AzureAD` ein.

Parametername	Alias	Parametertyp	Standardwert	Zu verwenden der Wert
CredentialsProvider	AWSCredentialsProviderClass (veraltet)	Erforderlich	Keine	`AzureAD`

Benutzer

Die E-Mail-Adresse des Azure-AD-Benutzers, der für die Authentifizierung mit Azure AD verwendet werden soll.

Parametername	Alias	Parametertyp	Standardwert
Benutzer	UID (veraltet)	Erforderlich	Keine

Passwort

Das Passworts für den Azure-AD-Benutzer.

Parametername	Alias	Parametertyp	Standardwert
Passwort	PWD (veraltet)	Erforderlich	Keine

Azure-AD-Mandanten-ID

Die Mandanten-ID Ihrer Azure-AD-Anwendung.

Parametername	Alias	Parametertyp	Standardwert
AzureAdTenantId	tenant_id (veraltet)	Erforderlich	Keine

Azure-AD-Client-ID

Die Client-ID Ihrer Azure-AD-Anwendung.

Parametername	Alias	Parametertyp	Standardwert
AzureAdClientId	client_id (veraltet)	Erforderlich	Keine

Azure-AD-Client-Secret

Das Client-Secret Ihrer Azure-AD-Anwendung.

Parametername	Alias	Parametertyp	Standardwert
AzureAdClientSecret	client_secret (veraltet)	Erforderlich	Keine

Bevorzugte Rolle

Der angenommene Amazon-Ressourcenname (ARN) der Rolle. Weitere Informationen zu ARN-Rollen finden Sie unter AssumeRole in der APIAWS Security Token Service -Referenz zu .

Parametername	Alias	Parametertyp	Standardwert
PreferredRole	preferred_role (veraltet)	Optional	Keine

Rollensitzungsdauer

Die Dauer der Rollen-Sitzung in Sekunden. Weitere Informationen finden Sie unter AssumeRole in der AWS Security Token Service -API-Referenz.

Parametername	Alias	Parametertyp	Standardwert
RoleSessionDuration	Duration (veraltet)	Optional	3600

Lake Formation aktiviert

Gibt an, ob die API-Aktion AssumeDecoratedRoleWithSAML Lake Formation verwendet werden soll, um temporäre IAM-Anmeldeinformationen anstelle der AssumeRoleWithSAML AWS STS -API-Aktion abzurufen.

Parametername	Alias	Parametertyp	Standardwert
LakeFormationEnabled	Keine	Optional	FALSE

Okta-Anmeldeinformationen

Ein SAML-basierter Authentifizierungsmechanismus, der die Authentifizierung bei Athena mit dem Okta-Identitätsanbieter ermöglicht. Bei dieser Methode wird davon ausgegangen, dass bereits ein Verbund zwischen Athena und Okta eingerichtet wurde.

Anmeldeinformationsanbieter

Der Anbieter für Anmeldeinformationen, der zur Authentifizierung von Anforderungen an AWSverwendet wird. Stellen Sie den Wert dieses Parameters auf `Okta` ein.

Parametername	Alias	Parametertyp	Standardwert	Zu verwenden der Wert
CredentialsProvider	AWSCredentialsProviderClass (veraltet)	Erforderlich	Keine	`Okta`

Benutzer

Die E-Mail-Adresse des Okta-Benutzers, der für die Authentifizierung bei Okta verwendet werden soll.

Parametername	Alias	Parametertyp	Standardwert
Benutzer	UID (veraltet)	Erforderlich	Keine

Passwort

Das Passwort für den Okta-Benutzer.

Parametername	Alias	Parametertyp	Standardwert
Passwort	PWD (veraltet)	Erforderlich	Keine

Okta-Hostname

Die URL für Ihre Okta-Organisation. Sie können den Parameter `idp_host` aus der URL Link einbetten in Ihrer Okta-Anwendung extrahieren. Informationen zu den erforderlichen Schritten finden Sie unter Abrufen von ODBC-Konfigurationsinformationen von Okta. Das erste Segment nach `https://` bis einschließlich `okta.com`, ist Ihr IdP-Host (zum Beispiel `trial-1234567.okta.com` für eine URL, die mit `https://trial-1234567.okta.com` beginnt).

Parametername	Alias	Parametertyp	Standardwert
OktaHostName	IdP_Host (veraltet)	Erforderlich	Keine

Okta-Anwendungs-ID

Der zweiteilige Identifikator für Ihre Anwendung. Sie können die Anwendungs-ID aus der URL Link einbetten in Ihrer Okta-Anwendung extrahieren. Informationen zu den erforderlichen Schritten finden Sie unter Abrufen von ODBC-Konfigurationsinformationen von Okta. Die Anwendungs-ID besteht aus den letzten beiden Segmenten der URL, einschließlich des Schrägstrichs in der Mitte. Bei den Segmenten handelt es sich um zwei 20-stellige Zeichenfolgen, die eine Mischung aus Zahlen sowie Groß- und Kleinbuchstaben (z. B. `Abc1de2fghi3J45kL678/abc1defghij2klmNo3p4`) enthalten.

Parametername	Alias	Parametertyp	Standardwert
OktaAppId	App_ID (veraltet)	Erforderlich	Keine

Okta-Anwendungsname

Der Name Ihrer Okta-Anwendung.

Parametername	Alias	Parametertyp	Standardwert
OktaAppName	App_Name (veraltet)	Erforderlich	Keine

Okta-MFA-Typ

Wenn Sie Okta so eingerichtet haben, dass eine Multi-Faktor-Authentifizierung (MFA) erforderlich ist, müssen Sie je nach dem zweiten Faktor, den Sie verwenden möchten, den Okta-MFA-Typ und zusätzliche Parameter angeben.

Der Okta-MFA-Typ ist der zweite Authentifizierungsfaktortyp (nach dem Passwort), der für die Authentifizierung bei Okta verwendet wird. Zu den unterstützten zweiten Faktoren gehören Push-Benachrichtigungen, die über die Okta-Verify-App gesendet werden, und temporäre Einmalpasswörter (TOTPs), die von Okta Verify, Google Authenticator generiert oder per SMS gesendet werden. Die Sicherheitsrichtlinien der einzelnen Organisationen bestimmen, ob MFA für die Benutzeranmeldung erforderlich ist oder nicht.

Parametername	Alias	Parametertyp	Standardwert	Mögliche Werte
OktaMfaType	okta_mfa_type (veraltet)	Erforderlich, wenn Okta so eingerichtet ist, dass MFA erforderlich ist	Keine	`oktaverifywithpush` , `oktaverifywithtotp` , `googleauthenticator` , `smsauthentication`

Okta-Telefonnummer anrufen

Die Telefonnummer, an die Okta ein temporäres Einmalpasswort per SMS sendet, wenn der `smsauthentication`-MFA-Typ ausgewählt wird. Bei der Telefonnummer muss es sich um eine US-amerikanische oder kanadische Telefonnummer handeln.

Parametername	Alias	Parametertyp	Standardwert
OktaPhoneNumber	okta_phone_number (veraltet)	Erforderlich, wenn OktaMfaType ist `smsauthentication`	Keine

Okta-MFA-Wartezeit

Die Wartezeit in Sekunden, bis der Benutzer eine Push-Benachrichtigung von Okta bestätigt, bevor der Treiber eine Timeout-Ausnahme auslöst.

Parametername	Alias	Parametertyp	Standardwert
OktaMfaWaitTime	okta_mfa_wait_time (veraltet)	Optional	60

Bevorzugte Rolle

Der angenommene Amazon-Ressourcenname (ARN) der Rolle. Weitere Informationen zu ARN-Rollen finden Sie unter AssumeRole in der APIAWS Security Token Service -Referenz zu .

Parametername	Alias	Parametertyp	Standardwert
PreferredRole	preferred_role (veraltet)	Optional	Keine

Rollensitzungsdauer

Die Dauer der Rollen-Sitzung in Sekunden. Weitere Informationen finden Sie unter AssumeRole in der AWS Security Token Service -API-Referenz.

Parametername	Alias	Parametertyp	Standardwert
RoleSessionDuration	Duration (veraltet)	Optional	3600

Lake Formation aktiviert

Gibt an, ob die API-Aktion AssumeDecoratedRoleWithSAML Lake Formation verwendet werden soll, um temporäre IAM-Anmeldeinformationen anstelle der AssumeRoleWithSAML AWS STS -API-Aktion abzurufen.

Parametername	Alias	Parametertyp	Standardwert
LakeFormationEnabled	Keine	Optional	FALSE

Ping-Anmeldeinformationen

Ein SAML-basierter Authentifizierungsmechanismus, der die Authentifizierung bei Athena mit dem Ping–Federate-Identitätsanbieter ermöglicht. Bei dieser Methode wird davon ausgegangen, dass bereits ein Verbund zwischen Athena und Ping Federate eingerichtet wurde.

Anmeldeinformationsanbieter

Der Anbieter für Anmeldeinformationen, der zur Authentifizierung von Anforderungen an AWSverwendet wird. Stellen Sie den Wert dieses Parameters auf `Ping` ein.

Parametername	Alias	Parametertyp	Standardwert	Zu verwenden der Wert
CredentialsProvider	AWSCredentialsProviderClass (veraltet)	Erforderlich	Keine	`Ping`

Benutzer

Die E-Mail-Adresse des Ping-Federate-Benutzers, der für die Authentifizierung mit Ping Federate verwendet werden soll.

Parametername	Alias	Parametertyp	Standardwert
Benutzer	UID (veraltet)	Erforderlich	Keine

Passwort

Das Passwort für den Ping-Federate-Benutzer.

Parametername	Alias	Parametertyp	Standardwert
Passwort	PWD (veraltet)	Erforderlich	Keine

PingHostName

Die Adresse für Ihren Ping-Server. Um Ihre Adresse zu finden, rufen Sie die folgende URL auf und sehen Sie sich das Feld SSO-Anwendungsendpunkt an.

```
https://your-pf-host-#:9999/pingfederate/your-pf-app#/spConnections
```

Parametername	Alias	Parametertyp	Standardwert
PingHostName	IdP_Host (veraltet)	Erforderlich	Keine

PingPortNumber

Die Portnummer, die für die Verbindung mit Ihrem IdP-Host verwendet werden soll.

Parametername	Alias	Parametertyp	Standardwert
PingPortNumber	IdP_Port (veraltet)	Erforderlich	Keine

PingPartnerSpId

Die Adresse des Serviceanbieters. Um die Adresse des Serviceanbieters zu finden, besuchen Sie die folgende URL und sehen Sie sich das Feld SSO Application Endpoint an.

```
https://your-pf-host-#:9999/pingfederate/your-pf-app#/spConnections
```

Parametername	Alias	Parametertyp	Standardwert
PingPartnerSpId	Partner_SPID (veraltet)	Erforderlich	Keine

Bevorzugte Rolle

Der angenommene Amazon-Ressourcenname (ARN) der Rolle. Weitere Informationen zu ARN-Rollen finden Sie unter AssumeRole in der APIAWS Security Token Service -Referenz zu .

Parametername	Alias	Parametertyp	Standardwert
PreferredRole	preferred_role (veraltet)	Optional	Keine

Rollensitzungsdauer

Die Dauer der Rollen-Sitzung in Sekunden. Weitere Informationen finden Sie unter AssumeRole in der AWS Security Token Service -API-Referenz.

Parametername	Alias	Parametertyp	Standardwert
RoleSessionDuration	Duration (veraltet)	Optional	3600

Lake Formation aktiviert

Gibt an, ob die API-Aktion AssumeDecoratedRoleWithSAML Lake Formation verwendet werden soll, um temporäre IAM-Anmeldeinformationen anstelle der AssumeRoleWithSAML AWS STS -API-Aktion abzurufen.

Parametername	Alias	Parametertyp	Standardwert
LakeFormationEnabled	Keine	Optional	FALSE

AD-FS-Anmeldeinformationen

Ein SAML-basierter Authentifizierungsmechanismus, der die Authentifizierung bei Athena mithilfe von Microsoft Active Directory Federation Services (AD FS) ermöglicht. Bei dieser Methode wird davon ausgegangen, dass der Benutzer bereits einen Verbund zwischen Athena und AD FS eingerichtet hat.

Anmeldeinformationsanbieter

Der Anbieter für Anmeldeinformationen, der zur Authentifizierung von Anforderungen an AWSverwendet wird. Stellen Sie den Wert dieses Parameters auf `ADFS` ein.

Parametername	Alias	Parametertyp	Standardwert	Zu verwenden der Wert
CredentialsProvider	AWSCredentialsProviderClass (veraltet)	Erforderlich	Keine	`ADFS`

Benutzer

Die E-Mail-Adresse des AD-FS-Benutzers, der für die Authentifizierung mit AD FS verwendet werden soll.

Parametername	Alias	Parametertyp	Standardwert
Benutzer	UID (veraltet)	Erforderlich für die formularbasierte Authentifizierung. Optional für die integrierte Windows-Authentifizierung.	Keine

Passwort

Das Passwort für den AD-FS-Benutzer.

Parametername	Alias	Parametertyp	Standardwert
Passwort	PWD (veraltet)	Erforderlich für die formularbasierte Authentifizierung. Optional für die integrierte Windows-Authentifizierung.	Keine

ADFS-Hostname

Die Adresse für Ihren AD-FS-Server.

Parametername	Alias	Parametertyp	Standardwert
AdfsHostName	IdP_Host (veraltet)	Erforderlich	Keine

ADFS-Portnummer

Die Portnummer, die für die Verbindung mit Ihrem AD-FS-Server verwendet werden soll.

Parametername	Alias	Parametertyp	Standardwert
AdfsPortNumber	IdP_Port (veraltet)	Erforderlich	Keine

ADFS-Vertrauensseite

Die vertrauenswürdige vertrauende Stelle. Verwenden Sie diesen Parameter, um die URL des AD-FS-Endpunkts der vertrauenden Stelle zu überschreiben.

Parametername	Alias	Parametertyp	Standardwert
AdfsRelyingParty	LoginToRP (veraltet)	Optional	`urn:amazon:webservices`

ADFS WIA aktiviert

Boolesch. Verwenden Sie diesen Parameter, um die Windows Integrated Authentication (WIA) mit AD FS zu aktivieren.

Parametername	Alias	Parametertyp	Standardwert
AdfsWiaEnabled	none	Optional	FALSE

Bevorzugte Rolle

Der angenommene Amazon-Ressourcenname (ARN) der Rolle. Weitere Informationen zu ARN-Rollen finden Sie unter `AssumeRole` in der APIAWS Security Token Service -Referenz zu .

Parametername	Alias	Parametertyp	Standardwert
PreferredRole	preferred_role (veraltet)	Optional	Keine

Rollensitzungsdauer

Die Dauer der Rollen-Sitzung in Sekunden. Weitere Informationen finden Sie unter `AssumeRole` in der AWS Security Token Service -API-Referenz.

Parametername	Alias	Parametertyp	Standardwert
RoleSessionDuration	Duration (veraltet)	Optional	3600

Lake Formation aktiviert

Gibt an, ob die `AssumeDecoratedRoleWithSAML` Lake-Formation-API-Aktion verwendet werden soll, um temporäre IAM-Anmeldeinformationen anstelle der `AssumeRoleWithSAML` AWS STS API-Aktion abzurufen.

Parametername	Alias	Parametertyp	Standardwert
LakeFormationEnabled	none	Optional	FALSE

Anmeldeinformationen für Browser Azure AD

Browser Azure AD ist ein SAML-basierter Authentifizierungsmechanismus, der mit dem Azure-AD-Identitätsprovider zusammenarbeitet und Multi-Faktor-Authentifizierung unterstützt. Im Gegensatz zum standardmäßigen Azure-AD-Authentifizierungsmechanismus erfordert dieser Mechanismus keinen Benutzernamen, kein Passwort und kein Client-Secret in den Verbindungsparametern. Wie der standardmäßige Azure AD-Authentifizierungsmechanismus geht auch Browser Azure AD davon aus, dass der Benutzer bereits einen Verbund zwischen Athena und Azure AD eingerichtet hat.

Anmeldeinformationsanbieter

Der Anbieter für Anmeldeinformationen, der zur Authentifizierung von Anforderungen an AWSverwendet wird. Stellen Sie den Wert dieses Parameters auf `BrowserAzureAD` ein.

Parametername	Alias	Parametertyp	Standardwert	Zu verwenden der Wert
CredentialsProvider	AWSCredentialsProviderClass (veraltet)	Erforderlich	Keine	`BrowserAzureAD`

Azure-AD-Mandanten-ID

Die Mandanten-ID Ihrer Azure-AD-Anwendung

Parametername	Alias	Parametertyp	Standardwert
AzureAdTenantId	tenant_id (veraltet)	Erforderlich	Keine

Azure-AD-Client-ID

Die Client-ID Ihrer Azure-AD-Anwendung

Parametername	Alias	Parametertyp	Standardwert
AzureAdClientId	client_id (veraltet)	Erforderlich	Keine

Identitätsanbieter-Reaktions-Timeout

Die Dauer in Sekunden, bevor der Treiber nicht mehr auf die SAML-Antwort von Azure AD wartet.

Parametername	Alias	Parametertyp	Standardwert
IdpResponseTimeout	idp_response_timeout (veraltet)	Optional	120

Bevorzugte Rolle

Der angenommene Amazon-Ressourcenname (ARN) der Rolle. Weitere Informationen zu ARN-Rollen finden Sie unter AssumeRole in der APIAWS Security Token Service -Referenz zu .

Parametername	Alias	Parametertyp	Standardwert
PreferredRole	preferred_role (veraltet)	Optional	Keine

Rollensitzungsdauer

Die Dauer der Rollen-Sitzung in Sekunden. Weitere Informationen finden Sie unter AssumeRole in der AWS Security Token Service -API-Referenz.

Parametername	Alias	Parametertyp	Standardwert
RoleSessionDuration	Duration (veraltet)	Optional	3600

Lake Formation aktiviert

Gibt an, ob die API-Aktion AssumeDecoratedRoleWithSAML Lake Formation verwendet werden soll, um temporäre IAM-Anmeldeinformationen anstelle der AssumeRoleWithSAML AWS STS -API-Aktion abzurufen.

Parametername	Alias	Parametertyp	Standardwert
LakeFormationEnabled	Keine	Optional	FALSE

Browser-SAML-Anmeldeinformationen

Browser SAML ist ein generisches Authentifizierungs-Plugin, das mit SAML-basierten Identitätsanbietern zusammenarbeiten kann und Multi-Faktor-Authentifizierung unterstützt.

Anmeldeinformationsanbieter

Der Anbieter für Anmeldeinformationen, der zur Authentifizierung von Anforderungen an AWSverwendet wird. Stellen Sie den Wert dieses Parameters auf `BrowserSaml` ein.

Parametername	Alias	Parametertyp	Standardwert	Zu verwenden der Wert
CredentialsProvider	AWSCredentialsProviderClass (veraltet)	Erforderlich	Keine	`BrowserSaml`

Single-Sign-On-Anmelde-URL

Die Single-Sign-On-URL für Ihre Anwendung beim SAML-basierten Identitätsanbieter.

Parametername	Alias	Parametertyp	Standardwert
SsoLoginUrl	login_url (veraltet)	Erforderlich	Keine

Listener-Port

Die Portnummer, die verwendet wird, um auf die SAML-Antwort zu warten. Dieser Wert sollte mit der URL übereinstimmen, mit der Sie den SAML-basierten Identitätsanbieter konfiguriert haben (z. B. `http://localhost:7890/athena`).

Parametername	Alias	Parametertyp	Standardwert
ListenPort	listen_port (veraltet)	Optional	`7890`

Identitätsanbieter-Reaktions-Timeout

Die Dauer in Sekunden, bevor der Treiber nicht mehr auf die SAML-Antwort von Azure AD wartet.

Parametername	Alias	Parametertyp	Standardwert
IdpResponseTimeout	idp_response_timeout (veraltet)	Optional	120

Bevorzugte Rolle

Der angenommene Amazon-Ressourcenname (ARN) der Rolle. Weitere Informationen zu ARN-Rollen finden Sie unter AssumeRole in der APIAWS Security Token Service -Referenz zu .

Parametername	Alias	Parametertyp	Standardwert
PreferredRole	preferred_role (veraltet)	Optional	Keine

Rollensitzungsdauer

Die Dauer der Rollen-Sitzung in Sekunden. Weitere Informationen finden Sie unter AssumeRole in der AWS Security Token Service -API-Referenz.

Parametername	Alias	Parametertyp	Standardwert
RoleSessionDuration	Duration (veraltet)	Optional	3600

Lake Formation aktiviert

Gibt an, ob die API-Aktion AssumeDecoratedRoleWithSAML Lake Formation verwendet werden soll, um temporäre IAM-Anmeldeinformationen anstelle der AssumeRoleWithSAML AWS STS -API-Aktion abzurufen.

Parametername	Alias	Parametertyp	Standardwert
LakeFormationEnabled	Keine	Optional	FALSE

Andere JDBC-3.x-Konfigurationen

In den folgenden Abschnitten werden einige zusätzliche Konfigurationseinstellungen für den JDBC-3.x-Treiber beschrieben.

Netzwerk-Timeout

Die Zeit in Millisekunden, die der Treiber auf eine Antwort wartet, wenn er einen API-Aufruf an Athena tätigt. Nach Ablauf dieser Zeit löst der Treiber eine Timeout-Ausnahme aus.

Das Netzwerk-Timeout kann nicht als Verbindungsparameter festgelegt werden. Um ihn festzulegen, rufen Sie die `setNetworkTimeout`-Methode für ein JDBC-`Connection`-Objekt auf. Dieser Wert kann während des Lebenszyklus der JDBC-Verbindung geändert werden. Der Standardwert für diesen Parameter ist `infinity`.

Im folgenden Beispiel wird der Netzwerk-Timeout auf 5 000 Millisekunden festgelegt.

```
...
AthenaDriver driver = new AthenaDriver();
Connection connection = driver.connect(url, connectionParameters);
connection.setNetworkTimeout(null, 5000);
...
```

Zeitbeschränkung für Abfragen

Die Zeit in Sekunden, die der Treiber auf den Abschluss einer Abfrage in Athena wartet, nachdem eine Abfrage gestellt wurde. Nach Ablauf dieser Zeit versucht der Treiber, die übermittelte Abfrage abzubrechen, und löst eine Timeout-Ausnahme aus.

Das Abfrage-Timeout kann nicht als Verbindungsparameter festgelegt werden. Um ihn festzulegen, rufen Sie die `setQueryTimeout`-Methode für ein JDBC-`Statement`-Objekt auf. Dieser Wert kann während des Lebenszyklus der JDBC-Anweisung geändert werden. Der Standardwert für diesen Parameter ist `0` (Null). Ein Wert von `0` bedeutet, dass Abfragen ausgeführt werden können, bis sie abgeschlossen sind (abhängig von Service Quotas).

Im folgenden Beispiel wird der Netzwerk-Timeout auf 5 Sekunden festgelegt.

```
...
AthenaDriver driver = new AthenaDriver();
Connection connection = driver.connect(url, connectionParameters);
Statement statement = connection.createStatement();
statement.setQueryTimeout(5);
...
```

Versionshinweise zu Amazon Athena JDBC 3.x

Diese Versionshinweise enthalten Details zu Verbesserungen, Funktionen, bekannten Problemen und Workflow-Änderungen im Amazon Athena-JDBC-3.x-Treiber.

3.1.0

Veröffentlicht am 2024-02-15

Amazon Athena-JDBC-Treiberversion 3.1.0 bietet Unterstützung für Microsoft Active Directory Federation Services (AD FS) Windows Integrierte Authentifizierung und formularbasierte Authentifizierung. Die Version 3.1.0 enthält auch andere kleinere Verbesserungen und Fehlerbehebungen.

Informationen zum Herunterladen des JDBC-v3-Treibers finden Sie unter JDBC-3.x-Treiber-Download.

Athena-JDBC-2.x-Treiber

Sie können eine JDBC-Verbindung verwenden, um Athena mit Business-Intelligence-Tools und anderen Anwendungen wie SQL Workbench zu verbinden. Verwenden Sie dazu die Links zu Amazon S3 auf dieser Seite, um den Athena-JDBC-2.x-Treiber herunterzuladen, zu installieren und zu konfigurieren. Informationen zum Erstellen der JDBC-Verbindungs-URL finden Sie im herunterladbaren Installations- und Konfigurationshandbuch für JDBC und im . Weitere Informationen zu Berechtigungen finden Sie unter Zugriff über JDBC- und ODBC-Verbindungen. Wenn Sie Feedback zum JDBC-Treiber senden möchten, senden Sie eine E-Mail an athena-

feedback@amazon.com. Ab Version 2.0.24 stehen zwei Versionen des Treibers zur Verfügung: eine enthält die AWS-SDK und eine nicht.

> **Important**
>
> Beachten Sie bei der Verwendung des JDBC-Treibers unbedingt die folgenden Anforderungen:
>
> - Open port 444 – Halten Sie Port 444, den Athena zum Streamen von Abfrageergebnissen verwendet, für ausgehenden Datenverkehr geöffnet. Wenn Sie einen PrivateLink-Endpunkt für die Verbindung mit Athena verwenden, stellen Sie sicher, dass die Sicherheitsgruppe, die an den PrivatLink-Endpunkt angeschlossen ist, für eingehenden Datenverkehr an Port 444 geöffnet ist. Wenn Port 444 blockiert ist, erhalten Sie möglicherweise die Fehlermeldung [Simba][AthenaJDBC](100123) Ein Fehler ist aufgetreten. Ausnahme während der Spalten-Initialisierung.
> - athena:GetQueryResultsStream-Richtlinie – Fügen Sie die `athena:GetQueryResultsStream`-Richtlinienaktion den IAM-Prinzipalen hinzu, die den JDBC-Treiber verwenden. Diese Richtlinienaktion wird nicht direkt mit der API bereitgestellt. Sie wird nur mit dem JDBC-Treiber als Teil der Unterstützung von Streaming-Ergebnissen verwendet. Eine Beispielrichtlinie finden Sie unter AWS Verwaltete Richtlinie: AWSQuicksightAthenaAccess.
> - Verwenden des JDBC-Treibers für mehrere Datenkataloge – Um den JDBC-Treiber für mehrere Datenkataloge mit Athena zu verwenden (z. B. bei Verwendung eines externen Hive-Metastores oder Verbundabfragen), fügen Sie `MetadataRetrievalMethod=ProxyAPI` in Ihre JDBC-Verbindungszeichenfolge ein.
> - 4.1-Treiber – Ab 2023 wird die Treiberunterstützung für JDBC-Version 4.1 eingestellt. Es werden keine weiteren Updates veröffentlicht. Wenn Sie einen JDBC-4.1-Treiber verwenden, wird die Migration zum 4.2-Treiber dringend empfohlen.

JDBC-2.x-Treiber mit AWS-SDK

Die JDBC-Treiber-Version 2.1.3 entspricht dem Datenstandard JDBC-API 4.2 und erfordert JDK 8.0 oder höher. Informationen zum Überprüfen der verwendeten Version der Java-Laufzeitumgebung (JRE) finden Sie in der Java-Dokumentation.

Verwenden Sie den folgenden Link, um die `.jar`-Datei des JDBC 4.2-Treibers herunterzuladen.

- AthenaJDBC42-2.1.3.1002.jar

Der folgende `.zip`-Dateidownload enthält die `.jar`-Datei für JDBC 4.2 und beinhaltet das AWS-SDK und die dazugehörige Dokumentation, Versionshinweise, Lizenzen und Vereinbarungen.

- SimbaAthenaJDBC-2.1.3.1002.zip

JDBC-2.x-Treiber ohne AWS-SDK

Die JDBC-Treiber-Version 2.1.3 entspricht dem Datenstandard JDBC-API 4.2 und erfordert JDK 8.0 oder höher. Informationen zum Überprüfen der verwendeten Version der Java-Laufzeitumgebung (JRE) finden Sie in der Java-Dokumentation.

Verwenden Sie den folgenden Link, um die JDBC 4.2 `.jar`-Treiberdatei ohne das AWS-SDK herunterzuladen.

- AthenaJDBC42-2.1.3.1003.jar

Der folgende `.zip`-Dateidownload enthält die `.jar`-Datei für JDBC 4.2 und die dazugehörige Dokumentation, Versionshinweise, Lizenzen und Vereinbarungen. Es ist kein AWS-SDK enthalten.

- SimbaAthenaJDBC-2.1.3.1003.zip

JDBC-2.x-Treiber – Versionshinweise, Lizenzvereinbarung und Anmerkungen

Nachdem Sie die benötigte Version heruntergeladen haben, lesen Sie die Versionshinweise und überprüfen Sie die Lizenzvereinbarung und Anmerkungen.

- Versionshinweise
- Lizenzvereinbarung
- Mitteilungen
- Drittanbieterlizenzen

Dokumentation zum JDBC-2.x-Treiber

Laden Sie die folgende Dokumentation für den Treiber herunter:

- Installations- und Konfigurationsleitfaden für JDBC-Treiber. Verwenden Sie dieses Handbuch für die Installation und Konfiguration des Treibers.
- Migrationsleitfaden für JDBC-Treiber. Verwenden Sie dieses Handbuch für die Migration von früheren Versionen auf die aktuelle Version.

Herstellen einer Verbindung mit Amazon Athena mit ODBC

Amazon Athena bietet zwei ODBC-Treiber, die Versionen 1.x und 2.x. Der Athena-ODBC-2.x-Treiber ist eine neue Alternative, die Linux-, macOS-ARM-, macOS-Intel- und Windows-64-Bit-Systeme unterstützt. Der Athena-2.x-Treiber unterstützt alle Authentifizierungs-Plugins, die der 1.x-ODBC-Treiber unterstützt, und fast alle Verbindungsparameter sind abwärtskompatibel.

- Informationen zum Herunterladen des neuesten ODBC-2.x-Treibers finden Sie unter ODBC-2.x-Verbindungen von Amazon Athena konfigurieren.
- Informationen zum Herunterladen des neuesten ODBC-1.x-Treibers finden Sie unter Athena-ODBC-1.x-Treiber.

Themen

ODBC-2.x-Verbindungen von Amazon Athena konfigurieren

Sie können eine ODBC-Verbindung verwenden, um von vielen SQL-Client-Tools und -Anwendungen von Drittanbietern eine Verbindung zu Amazon Athena herzustellen. Sie richten die ODBC-Verbindung auf Ihrem Client-Computer ein.

Überlegungen und Einschränkungen

- Informationen zur Migration vom Athena-ODBC-1.x-Treiber zum Athena-2.x-ODBC-Treiber finden Sie unter Migration zum ODBC-2.x-Treiber.
- Wenn Sie den S3-Fetcher mit der `CSE_KMS`-Verschlüsselungsoption verwenden, kann der Amazon-S3-Client das im Amazon-S3-Bucket gespeicherte Ergebnis nicht entschlüsseln. Um das Problem zu umgehen, verwenden Sie die Athena-Streaming-API-Option für den Abruf der Ergebnismenge.

ODBC-2.x-Treiber-Download

Um den Amazon Athena-2.x-ODBC-Treiber herunterzuladen, besuchen Sie die Links auf dieser Seite.

> **Important**
>
> Beachten Sie bei der Verwendung des ODBC-2.x-Treibers unbedingt die folgenden Anforderungen:
>
> - Open port 444 – Halten Sie Port 444, den Athena zum Streamen von Abfrageergebnissen verwendet, für ausgehenden Datenverkehr geöffnet. Wenn Sie einen PrivateLink Endpunkt verwenden, um eine Verbindung zu Athena herzustellen, stellen Sie sicher, dass die mit dem PrivateLink Endpunkt verbundene Sicherheitsgruppe für eingehenden Datenverkehr auf Port 444 geöffnet ist.
> - athena:GetQueryResultsStream policy – Fügen Sie die `athena:GetQueryResultsStream` Richtlinienaktion zu den IAM-Prinzipalen hinzu, die den ODBC-Treiber verwenden. Diese Richtlinienaktion wird nicht direkt mit der API bereitgestellt. Sie wird nur mit dem JDBC-Treiber als Teil der Unterstützung von Streaming-Ergebnissen verwendet. Eine Beispielrichtlinie finden Sie unter AWS Verwaltete Richtlinie: AWSQuicksightAthenaAccess.

Linux

Treiberversion	Download-Link
ODBC 2.0.2.2 für Linux 64-Bit	Linux 64-Bit-ODBC-Treiber 2.0.2.2

macOS (ARM)

Treiberversion	Download-Link
ODBC 2.0.2.2 für macOS 64-Bit (ARM)	macOS 64-Bit-ODBC-Treiber 2.0.2.2 (ARM)

macOS (Intel)

Treiberversion	Download-Link
ODBC 2.0.2.2 für macOS 64-Bit (Intel)	macOS 64-Bit-ODBC-Treiber 2.0.2.2 (Intel)

Windows

Treiberversion	Download-Link
ODBC 2.0.2.2 für Windows 64-Bit	Windows 64-Bit-ODBC-Treiber 2.0.2.2

Themen

Erste Schritte mit dem ODBC-2.x-Treiber

Verwenden Sie die Informationen in diesem Abschnitt, um mit dem ODBC-2.x-Treiber von Amazon Athena zu beginnen. Der Treiber wird auf den Betriebssystemen Windows, Linux und macOS unterstützt.

Themen

Windows

Wenn Sie einen Windows-Client-Computer für den Zugriff auf Amazon Athena verwenden möchten, ist der Amazon Athena-ODBC-Treiber erforderlich.

Windows-Systemanforderungen

Installieren Sie den ODBC-Treiber von Amazon Athena auf Client-Computern, die direkt auf Amazon-Athena-Datenbanken zugreifen, anstatt einen Webbrowser zu verwenden.

Das von Ihnen verwendete Windows-System muss die folgenden Anforderungen erfüllen:

- Sie haben Administratorrechte
- Eines der folgenden Betriebssysteme:
  - Windows 11, 10 oder 8.1
  - Windows Server 2019, 2016 oder 2012
- Mindestens 100 MB verfügbarer Speicherplatz
- Microsoft Visual C++ Redistributable for Visual Studio für 64-Bit-Windows installiert.

Installieren des Amazon-Athena-ODBC-Treibers

Wie Sie den ODBC-Treiber von Amazon Athena für Windows herunterzuladen und installieren

1. Laden Sie die `AmazonAthenaODBC-2.x.x.x.msi`-Installationsdateien herunter.
2. Starten Sie die Installationsdatei und wählen Sie dann Weiter.
3. Um die Bedingungen der Lizenzvereinbarung zu akzeptieren, aktivieren Sie das Kontrollkästchen und wählen Sie dann Weiter.
4. Um den Installationsort zu ändern, wählen Sie Durchsuchen, navigieren Sie zum gewünschten Ordner und wählen Sie dann OK.
5. Um den Installationsort zu akzeptieren, wählen Sie Weiter.
6. Wählen Sie Installieren aus.
7. Nach abgeschlossener Installation wählen Sie Beenden aus.

Möglichkeiten, die Optionen für die Treiberkonfiguration festzulegen

Um das Verhalten des ODBC-Treibers von Amazon Athena unter Windows zu steuern, können Sie die Treiberkonfigurationsoptionen auf folgende Weise festlegen:

- Im ODBC-Datenquellenadministratorprogramm, wenn Sie einen Datenquellennamen (DSN) konfigurieren.
- Durch Hinzufügen oder Ändern von Windows-Registrierungsschlüsseln an der folgenden Stelle:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI\YOUR_DSN_NAME
```

- Durch das Einstellen der Treiberoptionen in der Verbindungszeichenfolge, wenn Sie eine Verbindung programmgesteuert herstellen.

Konfiguration eines Datenquellennamens unter Windows

Nachdem Sie den ODBC-Treiber heruntergeladen und installiert haben, fügen Sie dem Clientcomputer oder der Amazon-EC2-Instance einen Datenquellennamen (DSN) hinzu. SQL-Client-Tools verwenden diese Datenquelle, um eine Verbindung mit Amazon Athena herzustellen.

So erstellen Sie einen DSN-Eintrag

1. Klicken Sie im Windows-Startmenü mit der rechten Maustaste auf ODBC-Datenquellen (64 Bit) und wählen Sie dann Mehr, Als Administrator ausführen aus.
2. Wählen Sie im ODBC-Datenquelle-Administrator die Registerkarte Treiber aus und suchen Sie den Treiberordner.
3. Vergewissern Sie sich, dass Amazon Athena ODBC (x64) in der Spalte Name vorhanden ist.
4. Führen Sie eine der folgenden Aktionen aus:

   - Um den Treiber für alle Benutzer auf dem Computer zu konfigurieren, wählen Sie die Registerkarte System DSN. Da Anwendungen, die ein anderes Konto zum Laden von Daten verwenden, möglicherweise keine Benutzer-DSNs von einem anderen Konto erkennen können, empfehlen wir die System-DSN-Konfigurationsoption.

     > **Note**
     >
     > Für die Verwendung der System-DSN-Option sind Administratorrechte erforderlich.

   - Um den Treiber nur für Ihr Benutzerkonto zu konfigurieren, wählen Sie die Registerkarte Benutzer-DSN.
5. Wählen Sie Hinzufügen aus. Das Fenster Neue Datenquelle erstellen wird geöffnet.
6. Wählen Sie den (Amazon-Athena-ODBC-Treiber (x64) und wählen Sie dann Fertigstellen.

7. Geben Sie im Dialogfeld Amazon-Athena-ODBC-Konfiguration die folgenden Informationen ein. Weitere Informationen zu diesen Optionen finden Sie unter Wichtigste ODBC-2.x-Verbindungsparameter.

   - Geben Sie unter Datenquellenname einen Namen ein, unter dem Sie die Datenquelle identifizieren möchten.
   - Geben Sie unter Beschreibung eine Beschreibung zur schnellen Erkennung der Datenquelle ein.
   - Geben Sie unter Region den Namen der AWS-Region ein, in der Sie Athena verwenden werden (z. B. **`us-west-1`**).
   - Geben Sie unter Katalog den Namen des Amazon-Athena-Katalogs ein. Der Standardwert ist AwsDataCatalog, der von verwendet wird AWS Glue.
   - Geben Sie unter Datenbank den Namen der Amazon-Athena-Datenbank ein. Der Standardwert ist Standard.
   - Geben Sie für Arbeitsgruppe den Namen der Amazon-Athena-Arbeitsgruppe ein. Die Standardeinstellung ist primär.
   - Geben Sie für den Speicherort der S3-Ausgabe den Speicherort in Amazon S3 an, an dem die Abfrageergebnisse gespeichert werden sollen (z. B. `s3://DOC-EXAMPLE-BUCKET/`).
   - (Optional) Wählen Sie für Verschlüsselungsoptionen eine Verschlüsselungsoption. Der Standardwert ist `NOT_SET`.
   - (Optional) Wählen Sie für KMS-Schlüssel bei Bedarf einen KMS-Schlüssel aus.

8. Um Konfigurationsoptionen für die IAM-Authentifizierung anzugeben, wählen Sie Authentifizierungsoptionen.

9. Geben Sie die folgenden Informationen ein:

   - Wählen Sie als Authentifizierungstyp die Option IAM-Anmeldeinformationen aus. Dies ist die Standardeinstellung. Weitere Informationen zu den verfügbaren Authentifizierungstypen finden Sie unter Authentifizierungsoptionen.
   - Geben Sie in das Feld Benutzername einen Benutzernamen ein.
   - Geben Sie in das Feld Passwort ein Passwort ein.
   - Geben Sie für Sitzungstoken ein Sitzungstoken ein, wenn Sie temporäre AWS Anmeldeinformationen verwenden möchten. Informationen zu temporären Anmeldeinformationen finden Sie unter Verwenden temporärer Anmeldeinformationen mit - AWS Ressourcen im IAM-Benutzerhandbuch.

10. Wählen Sie OK aus.
11. Wählen Sie unten im Dialogfeld Amazon-Athena-ODBC-Konfiguration die Option Test aus. Wenn der Client-Computer erfolgreich eine Verbindung zu Amazon Athena herstellt, meldet das Feld Verbindungstest Verbindung erfolgreich. Wenn nicht, meldet das Feld Verbindung fehlgeschlagen mit entsprechenden Fehlerinformationen.
12. Wählen Sie OK, um den Verbindungstest zu beenden. Die von Ihnen erstellte Datenquelle wird in der Liste der Datenquellennamen erscheinen.

Unter Windows eine Verbindung ohne DSN verwenden

Sie können eine Verbindung ohne DSN verwenden, um eine Verbindung zu einer Datenbank ohne Datenquellennamen (DSN) herzustellen. Das folgende Beispiel zeigt eine Verbindungszeichenfolge für den Amazon Athena ODBC (x64)-ODBC-Treiber, der eine Verbindung zu Amazon Athena herstellt.

```
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

Wenn Sie einen Linux-Client-Computer für den Zugriff auf Amazon Athena verwenden möchten, ist der Amazon Athena-ODBC-Treiber erforderlich.

Linux-Systemanforderungen

Jeder Linux-Client-Computer, auf dem Sie den Treiber installieren, muss die folgenden Anforderungen erfüllen.

- Sie haben Root-Zugriff.
- Verwenden Sie eine der folgenden Linux-Distributionen:
  - Red Hat Enterprise Linux (RHEL) 7 oder 8
  - CentOS 7 oder 8.
- Stellen Sie 100 MB Festplattenspeicher zur Verfügung.
- Verwenden Sie Version 2.3.1 oder höher von unixODBC .

- Verwenden Sie Version 2.26 oder höher der GNU C Library (glibc).

Installieren des ODBC-Daten-Connectors unter Linux

Gehen Sie wie folgt vor, um den Amazon Athena-ODBC-Treiber auf einem Linux-Betriebssystem zu installieren.

So installieren Sie den Amazon Athena-ODBC-Treiber unter Linux

1. Geben Sie einen der folgenden Befehle ein:

   ```
   sudo rpm -Uvh AmazonAthenaODBC-2.X.Y.Z.rpm
   ```

   or

   ```
   sudo yum --nogpgcheck localinstall AmazonAthenaODBC-2.X.Y.Z.rpm
   ```

2. Geben Sie nach Abschluss der Installation einen der folgenden Befehle ein, um zu überprüfen, ob der Treiber installiert ist:

   - ```
     yum list | grep amazon-athena-odbc-driver
     ```

     Ausgabe:

     ```
     amazon-athena-odbc-driver.x86_64 2.0.2.1-1.amzn2int installed
     ```

   - ```
     rpm -qa | grep amazon
     ```

     Ausgabe:

     ```
     amazon-athena-odbc-driver-2.0.2.1-1.amzn2int.x86_64
     ```

Konfigurieren eines Datenquellennamens unter Linux

Nachdem der Treiber installiert wurde, finden Sie `.odbcinst.ini` Beispieldateien `.odbc.ini` und am folgenden Speicherort:

- `/opt/athena/odbc/ini/`.

Verwenden Sie die `.ini` Dateien an diesem Speicherort als Beispiele für die Konfiguration des Amazon Athena-ODBC-Treibers und des Datenquellennamens (DSN).

> **Note**
>
> Standardmäßig verwenden ODBC-Treibermanager die ausgeblendeten Konfigurationsdateien `.odbc.ini` und `.odbcinst.ini`, die sich im Home-Verzeichnis befinden.

Führen Sie die folgenden Schritte aus, um den Pfad zu den `.odbcinst.ini` Dateien `.odbc.ini` und mit unixODBC anzugeben.

So geben Sie ODBC-**`.ini`**Dateispeicherorte mit unixODBC an

1. Legen Sie `ODBCINI` auf den vollständigen Pfad und Dateinamen der `odbc.ini` Datei fest, wie im folgenden Beispiel.

   ```
   export ODBCINI=/opt/athena/odbc/ini/odbc.ini
   ```

2. Legen Sie `ODBCSYSINI` auf den vollständigen Pfad des Verzeichnisses fest, das die `odbcinst.ini` Datei enthält, wie im folgenden Beispiel.

   ```
   export ODBCSYSINI=/opt/athena/odbc/ini
   ```

3. Geben Sie den folgenden Befehl ein, um zu überprüfen, ob Sie den unixODBC-Treibermanager und die richtigen `odbc*.ini` Dateien verwenden:

   ```
   username % odbcinst -j
   ```

   Beispielausgabe für

   ```
   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. Wenn Sie einen Datenquellennamen (DSN) verwenden möchten, um eine Verbindung zu Ihrem Datenspeicher herzustellen, konfigurieren Sie die `odbc.ini` Datei so, dass Datenquellennamen (DSNs) definiert werden. Legen Sie die Eigenschaften in der `odbc.ini` Datei fest, um einen DSN zu erstellen, der die Verbindungsinformationen für Ihren Datenspeicher angibt, wie im folgenden Beispiel.

```
[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://odbc-temp-test-folder-out/
```

5. Konfigurieren Sie die `odbcinst.ini` Datei wie im folgenden Beispiel.

```
[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. Nachdem Sie den Amazon Athena-ODBC-Treiber installiert und konfiguriert haben, verwenden Sie das unixODBC-`isql`Befehlszeilen-Tool, um die Verbindung zu überprüfen, wie im folgenden Beispiel.

```
username % isql -v "athena_odbc_test"
+---------------------------------------+
```

```
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
SQL>
```

macOS

Wenn Sie einen macOS-Client-Computer für den Zugriff auf Amazon Athena verwenden möchten, ist der Amazon Athena-ODBC-Treiber erforderlich.

macOS-Systemanforderungen

Jeder macOS-Computer, auf dem Sie den Treiber installieren, muss die folgenden Anforderungen erfüllen.

- Verwenden Sie macOS Version 14 oder höher.
- Stellen Sie 100 MB Festplattenspeicher zur Verfügung.
- Verwenden Sie Version 3.52.16 oder höher von iODBC .

Installieren des ODBC-Daten-Connectors unter macOS

Gehen Sie wie folgt vor, um den Amazon Athena-ODBC-Treiber für macOS-Betriebssysteme herunterzuladen und zu installieren.

So laden Sie den Amazon Athena-ODBC-Treiber für macOS herunter und installieren ihn

1. Laden Sie die `-.pkg`Paketdatei herunter.
2. Doppelklicken Sie auf die Datei `.pkg`.
3. Führen Sie die Schritte im Assistenten aus, um den Treiber zu installieren.
4. Drücken Sie auf der Seite Lizenzvereinbarung auf Weiter und wählen Sie dann Zustimmen aus.
5. Wählen Sie Installieren aus.
6. Nach abgeschlossener Installation wählen Sie Beenden aus.
7. Geben Sie den folgenden Befehl ein, um zu überprüfen, ob der Treiber installiert ist:

```
> pkgutil --pkgs | grep athenaodbc
```

Je nach System kann die Ausgabe wie eine der folgenden aussehen.

```
com.amazon.athenaodbc-x86_64.Config
com.amazon.athenaodbc-x86_64.Driver
```

or

```
com.amazon.athenaodbc-arm64.Config
com.amazon.athenaodbc-arm64.Driver
```

Konfigurieren eines Datenquellennamens unter macOS

Nachdem der Treiber installiert wurde, finden Sie `.odbcinst.ini` Beispieldateien `.odbc.ini` und an den folgenden Speicherorten:

- Intel-Prozessorcomputer: `/opt/athena/odbc/x86_64/ini/`
- ARM-Prozessorcomputer: `/opt/athena/odbc/arm64/ini/`

Verwenden Sie die `.ini` Dateien an diesem Speicherort als Beispiele für die Konfiguration des Amazon Athena-ODBC-Treibers und des Datenquellennamens (DSN).

> **Note**
>
> Standardmäßig verwenden ODBC-Treibermanager die ausgeblendeten Konfigurationsdateien `.odbc.ini` und `.odbcinst.ini`, die sich im Home-Verzeichnis befinden.

Führen Sie die folgenden Schritte aus, um den Pfad zu den `.odbcinst.ini` Dateien `.odbc.ini` und mit dem iODBC-Treibermanager anzugeben.

So geben Sie ODBC-**`.ini`**Dateispeicherorte mit dem iODBC-Treibermanager an

1. Legen Sie `ODBCINI` auf den vollständigen Pfad und Dateinamen der `odbc.ini`-Datei fest.

- Verwenden Sie für macOS-Computer mit Intel-Prozessoren die folgende Syntax.

```
export ODBCINI=/opt/athena/odbc/x86_64/ini/odbc.ini
```

- Verwenden Sie für macOS-Computer mit ARM-Prozessoren die folgende Syntax.

```
export ODBCINI=/opt/athena/odbc/arm64/ini/odbc.ini
```

2. Legen Sie `ODBCSYSINI` auf den vollständigen Pfad und Dateinamen der `odbcinst.ini`-Datei fest.

- Verwenden Sie für macOS-Computer mit Intel-Prozessoren die folgende Syntax.

```
export ODBCSYSINI=/opt/athena/odbc/x86_64/ini/odbcinst.ini
```

- Verwenden Sie für macOS-Computer mit ARM-Prozessoren die folgende Syntax.

```
export ODBCSYSINI=/opt/athena/odbc/arm64/ini/odbcinst.ini
```

3. Wenn Sie einen Datenquellennamen (DSN) verwenden möchten, um eine Verbindung zu Ihrem Datenspeicher herzustellen, konfigurieren Sie die `odbc.ini` Datei so, dass Datenquellennamen (DSNs) definiert werden. Legen Sie die Eigenschaften in der `odbc.ini` Datei fest, um einen DSN zu erstellen, der die Verbindungsinformationen für Ihren Datenspeicher angibt, wie im folgenden Beispiel.

```
[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://odbc-temp-test-folder-out/
```

4. Konfigurieren Sie die `odbcinst.ini` Datei wie im folgenden Beispiel.

```
[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. Nachdem Sie den Amazon Athena-ODBC-Treiber installiert und konfiguriert haben, verwenden Sie das `iodbctest` Befehlszeilen-Tool, um die Verbindung zu überprüfen, wie im folgenden Beispiel.

```
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>
```

Athena-ODBC-2.x-Verbindungsparameter

Die Optionen im Dialogfeld ODBC-Konfiguration in Amazon Athena umfassen Authentifizierungsoptionen, erweiterte Optionen, Protokollierungsoptionen, Endpunktüberschreibungen und Proxyoptionen. Ausführliche Informationen zu den einzelnen Themen finden Sie unter den entsprechenden Links.

- Wichtigste ODBC-2.x-Verbindungsparameter
- Authentifizierungsoptionen
- Erweiterte Optionen
- Protokollierungsoptionen
- Überschreibungen von Endpunkten
- Proxy-Optionen

Wichtigste ODBC-2.x-Verbindungsparameter

In den folgenden Abschnitten wird jeder der wichtigsten Verbindungsparameter beschrieben.

Datenquellenname

Gibt den Namen Ihrer Datenquelle an.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
DSN	Optional für Verbindungstypen ohne DSN	none	`DSN=AmazonAthena0dbcUsWest1;`

Beschreibung

Enthält eine Beschreibung Ihrer Datenquelle.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
Beschreibung	Optional	none	`Description=Connection to Amazon Athena us-west-1;`

Katalog

Gibt den Namen des Datenkatalogs an. Weitere Informationen finden Sie unter DataCatalog in der Referenz für Amazon-Athena-API.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
Katalog	Optional	`AwsDataCatalog`	`Catalog=AwsDataCatalog;`

Region

Festlegen von AWS-Region. Weitere Informationen über AWS-Regionen finden Sie unter Regionen und Availability Zones.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
AwsRegion	zwingend erforderlich	none	`AwsRegion =us-west-1`

Datenbank

Legt den Datenbanknamen fest. Weitere Informationen finden Sie unter Database in der in der Referenz für Amazon-Athena-API..

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
Schema	Optional	`default`	`Schema=default;`

Arbeitsgruppe

Gibt den Namen der Arbeitsgruppe an. Weitere Informationen finden Sie unter WorkGroup in der in der Referenz für Amazon-Athena-API..

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
Arbeitsgruppe	Optional	`primary`	`Workgroup =primary;`

Ausgabeort

Gibt den Ort in Amazon S3 an, an dem die Abfrageergebnisse gespeichert werden. Weitere Informationen zum Ausgabeort finden Sie unter ResultConfiguration in der Referenz für Amazon-Athena-API..

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
S3OutputLocation	zwingend erforderlich	none	`S3OutputLocation=s3://` *`DOC-BEISPIEL-BUCKET`* `/;`

Verschlüsselungsoptionen

Name des Dialogparameters: Verschlüsselungsoptionen

Gibt die Verschlüsselungsoption an. Weitere Informationen zu Verschlüsselungsoptionen finden Sie unter EncryptionConfiguration in der Referenz für Amazon-Athena-API..

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Mögliche Werte	Beispiel für Verbindungszeichenfolgen
S3OutputEncOption	Optional	none	NOT_SET, SSE_S3, SSE_KMS, CSE_KMS	S3OutputEncOption=SSE_S3;

KMS-Schlüssel

Gibt einen KMS-Schlüssel für die Verschlüsselung an. Weitere Informationen zu Verschlüsselungsoptionen für KMS-Schlüssel finden Sie unter EncryptionConfiguration in der Referenz für Amazon-Athena-API..

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
S3OutputEncKMSKey	Optional	none	S3OutputEncKMSKey=your_key;

Verbindungstest

ODBC Data Source Administrator bietet eine Testoption, mit der Sie Ihre ODBC-2.x-Verbindung zu Amazon Athena testen können. Informationen zu den erforderlichen Schritten finden Sie unter Konfiguration eines Datenquellennamens unter Windows. Wenn Sie eine Verbindung testen, ruft der ODBC-Treiber die API-Aktion GetWorkGroup von Athena auf. Der Aufruf verwendet den Authentifizierungstyp und den entsprechenden Anbieter von Anmeldeinformationen, den Sie zum Abrufen der Anmeldeinformationen angegeben haben. Der Verbindungstest ist kostenlos, wenn Sie den ODBC-2.x-Treiber verwenden. Der Test generiert keine Abfrageergebnisse in Ihrem Amazon-S3-Bucket.

Authentifizierungsoptionen

Sie können mit den folgenden Authentifizierungstypen eine Verbindung zu Amazon Athena herstellen. Für alle Typen lautet der Name der Verbindungszeichenfolge `AuthenticationType`,

der Parametertyp ist `Required` und der Standardwert ist `IAM Credentials`. Informationen zu den Parametern für die einzelnen Authentifizierungstypen finden Sie unter dem entsprechenden Link. Allgemeine Authentifizierungsparameter finden Sie unter Allgemeine Authentifizierungsparameter.

Authentifizierungstyp	Beispiel für Verbindungszeichenfolgen
IAM-Anmeldeinformationen	`AuthenticationType=IAM Credentials;`
IAM-Profil	`AuthenticationType=IAM Profile;`
AD FS	`AuthenticationType=ADFS;`
Azure AD	`AuthenticationType=AzureAD;`
Azure AD Browser	`AuthenticationType=BrowserAzureAD;`
SAML Browser	`AuthenticationType=BrowserSAML;`
Browser SSO OIDC	`AuthenticationType=BrowserSSOOIDC;`
Standard-Anmeldeinformationen	`AuthenticationType=Default Credentials;`
Externe Anmeldeinformationen	`AuthenticationType=External Credentials;`
Instance-Profil	`AuthenticationType=Instance Profile;`
JWT	`AuthenticationType=JWT;`
Okta	`AuthenticationType=Okta;`
Ping	`AuthenticationType=Ping;`

IAM-Anmeldeinformationen

Sie können Ihre IAM-Anmeldeinformationen verwenden, um mit dem ODBC-Treiber eine Verbindung zu Amazon Athena herzustellen, indem Sie die in diesem Abschnitt beschriebenen Verbindungszeichenfolgenparameter verwenden.

Authentifizierungstyp

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
AuthenticationType	Erforderlich	`IAM Credentials`	`AuthenticationType=IAM Credentials;`

Benutzer-ID

Ihre AWS-Zugriffsschlüssel-ID. Weitere Informationen zu Zugriffsschlüsseln finden Sie unter AWS-Sicherheitsanmeldeinformationen im IAM-Benutzerhandbuch.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
Benutzerkennung (UID)	Erforderlich	none	`UID=AKIAIOSFODNN7EXAMPLE;`

Passwort

Ihre geheime AWS-Schlüssel-ID. Weitere Informationen zu Zugriffsschlüsseln finden Sie unter AWS-Sicherheitsanmeldeinformationen im IAM-Benutzerhandbuch.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
PWD	Erforderlich	none	`PWD=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKE;`

Sitzungs-Token

Wenn Sie temporäre AWS-Anmeldeinformationen verwenden, müssen Sie den Sitzungs-Token angeben. Weitere Informationen zu temporären Anmeldeinformationen finden Sie unter Temporäre Sicherheitsanmeldeinformationen in IAM im IAM-Benutzerhandbuch.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
SessionToken	Optional	none	`SessionToken=AQoDYXdzEJr...<remainder of session token>;`

IAM-Profil

Sie können ein benanntes Profil konfigurieren, um mithilfe des ODBC-Treibers eine Verbindung zu Amazon Athena herzustellen. Um die in Ihrem Profil von Hosting-Amazon-EC2-Instance verfügbaren Anmeldeinformationen zu verwenden, setzen Sie den `credential_source`-Parameter auf `Ec2InstanceMetadata`. Wenn Sie einen Anbieter für benutzerdefinierte Anmeldeinformationen in einem benannten Profil verwenden möchten, geben Sie einen Wert für den `plugin_name`-Parameter in Ihrer Profilkonfiguration an.

Authentifizierungstyp

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
AuthenticationType	Erforderlich	`IAM Credentials`	`AuthenticationType=IAM Profile;`

AWS-Profil

Der Profilname, der für Ihre ODBC-Verbindung verwendet werden soll. Weitere Informationen zu Profilen finden Sie unter Benannte Profile verwenden im AWS Command Line Interface-Benutzerhandbuch.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
AWS-Profil	Erforderlich	none	`AWSProfile=default;`

Bevorzugte Rolle

Der angenommene Amazon-Ressourcenname (ARN) der Rolle. Der bevorzugte Rollenparameter wird verwendet, wenn der Anbieter für benutzerdefinierte Anmeldeinformationen durch den `plugin_name`-Parameter in Ihrer Profilkonfiguration angegeben wird. Weitere Informationen über ARN-Rollen finden Sie unter AssumeRole in der AWS Security Token Service-API-Referenz.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
preferred_role	Optional	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Sitzungsdauer

Die Dauer der Rollen-Sitzung in Sekunden. Weitere Informationen zur Sitzungsdauer finden Sie unter AssumeRole in der AWS Security Token Service-API-Referenz. Der Parameter für die Sitzungsdauer wird verwendet, wenn der Anbieter für benutzerdefinierte Anmeldeinformationen durch den `plugin_name`-Parameter in Ihrer Profilkonfiguration angegeben wird.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
duration	Optional	`900`	`duration=900;`

Name des Plug-ins

Gibt den Namen eines Anbieters für benutzerdefinierte Anmeldeinformationen an, der in einem benannten Profil verwendet wird. Dieser Parameter kann dieselben Werte annehmen wie die Werte im Feld Authentifizierungstyp des ODBC-Datenquellenadministrators, wird aber nur für die `AWSProfile`-Konfiguration verwendet.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
plugin_name	Optional	`none`	`plugin_name=AzureAD;`

AD FS

AD FS ist ein SAML-basiertes Authentifizierungs-Plugin, das mit dem Active Directory Federation Service (AD FS)-Identitätsanbieter zusammenarbeitet. Das Plugin unterstützt die integrierte Windows-Authentifizierung und die formularbasierte Authentifizierung. Wenn Sie die integrierte Windows-Authentifizierung verwenden, können Sie den Benutzernamen und das Kennwort weglassen. Weitere Informationen zur Konfiguration von AD FS und Athena finden Sie unter Konfigurieren des Verbundzugriffs auf Amazon Athena für Microsoft-AD-FS-Benutzer mithilfe eines ODBC-Clients.

Authentifizierungstyp

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
AuthenticationType	Erforderlich	`IAM Credentials`	`AuthenticationType=ADFS;`

Benutzer-ID

Ihr Benutzername für die Verbindung zum AD-FS-Server. Wenn Sie die integrierte Windows-Authentifizierung verwenden, können Sie den Benutzernamen weglassen. Wenn Ihre AD-FS-Einrichtung einen Benutzernamen erfordert, müssen Sie ihn im Verbindungsparameter angeben.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
Benutzerkennung (UID)	Optional für die integrierte Windows-Authentifizierung	`none`	`UID=domain\username;`

Passwort

Ihr Passwort für die Verbindung zum AD-FS-Server. Wie im Feld für den Benutzernamen können Sie den Benutzernamen weglassen, wenn Sie die integrierte Windows-Authentifizierung verwenden. Wenn Ihre AD-FS-Einrichtung ein Passwort erfordert, müssen Sie es im Verbindungsparameter angeben.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
PWD	Optional für die integrierte Windows-Authentifizierung	`none`	`PWD=password_3EXAMPLE;`

Bevorzugte Rolle

Der angenommene Amazon-Ressourcenname (ARN) der Rolle. Wenn Ihre SAML-Assertion mehrere Rollen hat, können Sie diesen Parameter angeben, um die Rolle auszuwählen, die übernommen werden soll. Diese Rolle sollte in der SAML-Assertion enthalten sein. Weitere Informationen über ARN-Rollen finden Sie unter AssumeRole in der AWS Security Token Service-API-Referenz.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
preferred_role	Optional	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Sitzungsdauer

Die Dauer der Rollen-Sitzung in Sekunden. Weitere Informationen zur Sitzungsdauer finden Sie unter AssumeRole in der AWS Security Token Service-API-Referenz.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
duration	Optional	`900`	`duration=900;`

IdP-Host

Der Name des AD-FS-Servicehosts.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
idp_host	Erfordert	none	`idp_host=<server-name>.<company.com>;`

IdP-Port

Der Port, der für die Verbindung mit dem AD-FS-Host verwendet werden soll.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
idp_port	Erforderlich	none	`idp_port=443;`

LoginToRP

Die vertrauenswürdige vertrauende Stelle. Verwenden Sie diesen Parameter, um die URL des AD-FS-Endpunkts der vertrauenden Stelle zu überschreiben.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
LoginToRP	Optional	`urn:amazon:webservices`	`LoginToRP=trustedparty;`

Azure AD

Azure AD ist ein SAML-basiertes Authentifizierungs-Plugin, das mit dem Azure-AD-Identitätsanbieter funktioniert. Dieses Plugin bietet keine Unterstützung für Multi-Factor Authentication (MFA). Wenn Sie MFA-Unterstützung benötigen, sollten Sie stattdessen das `BrowserAzureAD`-Plugin verwenden.

Authentifizierungstyp

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
AuthenticationType	Erforderlich	`IAM Credentials`	`AuthenticationType=AzureAD;`

Bevorzugte Rolle

Der angenommene Amazon-Ressourcenname (ARN) der Rolle. Weitere Informationen zu ARN-Rollen finden Sie unter AssumeRole in der AWS Security Token Service-API-Referenz.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
preferred_role	Optional	none	`preferred_role=arn:aws:iam::123456789012:id/user1;`

Sitzungsdauer

Die Dauer der Rollen-Sitzung in Sekunden. Weitere Informationen finden Sie unter AssumeRole in der AWS Security Token Service-API-Referenz.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
duration	Optional	`900`	`duration=900;`

Tenant-ID

Gibt die Tenant-ID Ihrer Anwendung an.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
idp_tenant	Erforderlich	none	`idp_tenant=123zz112z-z12d-1z1f-11zz-f111aa111234;`

Client-ID

Gibt die Client-ID Ihrer Anwendung an.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
Client-ID	Erforderlich	none	`client_id=9178ac27-a1bc-1a2b-1a2b-a123abcd1234;`

Clientschlüssel

Gibt Ihr Client-Geheimnis an.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
client_secret	Erforderlich	none	`client_secret=zG12q~.xzG1xxxZ1wX1.~ZzXXX1XxkHZizeT1zzZ;`

Azure AD Browser

Browser Azure AD ist ein SAML-basiertes Authentifizierungs-Plugin, das mit dem Azure-AD-Identitätsprovider zusammenarbeitet und Multi-Faktor-Authentifizierung unterstützt. Im Gegensatz zum standardmäßigen Azure-AD-Plugin erfordert dieses Plugin keinen Benutzernamen, kein Passwort und keinen geheimen Clientschlüssel in den Verbindungsparametern.

Authentifizierungstyp

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
AuthenticationType	Erforderlich	`IAM Credentials`	`AuthenticationType=BrowserAzureAD;`

Bevorzugte Rolle

Der angenommene Amazon-Ressourcenname (ARN) der Rolle. Wenn Ihre SAML-Assertion mehrere Rollen hat, können Sie diesen Parameter angeben, um die Rolle auszuwählen, die übernommen werden soll. Die angegebene Rolle sollte in der SAML-Assertion vorhanden sein. Weitere Informationen über ARN-Rollen finden Sie unter AssumeRole in der AWS Security Token Service-API-Referenz.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
preferred_role	Optional	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Sitzungsdauer

Die Dauer der Rollen-Sitzung in Sekunden. Weitere Informationen zur Sitzungsdauer finden Sie unter AssumeRole in der AWS Security Token Service-API-Referenz.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
duration	Optional	`900`	`duration=900;`

Tenant-ID

Gibt die Tenant-ID Ihrer Anwendung an.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
idp_tenant	Erforderlich	none	`idp_tenant=123zz112z-z12d-1z1f-11zz-f111aa111234;`

Client-ID

Gibt die Client-ID Ihrer Anwendung an.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
Client-ID	Erforderlich	none	`client_id=9178ac27-a1bc-1a2b-1a2b-a123abcd1234;`

Zeitüberschreitung

Die Dauer in Sekunden, bevor das Plugin nicht mehr auf die SAML-Antwort von Azure AD wartet.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
timeout	Optional	`120`	`timeout=90;`

Azure-Dateicache aktivieren

Aktiviert einen Cache für temporäre Anmeldeinformationen. Mit diesem Verbindungsparameter können temporäre Anmeldeinformationen zwischengespeichert und zwischen mehreren Prozessen wiederverwendet werden. Verwenden Sie diese Option, um die Anzahl der geöffneten Browserfenster zu reduzieren, wenn Sie BI-Tools wie Microsoft Power BI verwenden.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
browser_azure_cache	Optional	1	`browser_azure_cache=0;`

SAML Browser

Browser SAML ist ein generisches Authentifizierungs-Plugin, das mit SAML-basierten Identitätsanbietern zusammenarbeiten kann und Multi-Faktor-Authentifizierung unterstützt. Ausführliche Informationen zur Konfiguration finden Sie unter Konfigurieren von Single Sign-On mit ODBC, SAML 2.0 und dem Okta-Identitätsanbieter.

Authentifizierungstyp

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
AuthenticationType	Erforderlich	`IAM Credentials`	`AuthenticationType=BrowserSAML;`

Bevorzugte Rolle

Der angenommene Amazon-Ressourcenname (ARN) der Rolle. Wenn Ihre SAML-Assertion mehrere Rollen hat, können Sie diesen Parameter angeben, um die Rolle auszuwählen, die übernommen werden soll. Diese Rolle sollte in der SAML-Assertion vorhanden sein. Weitere Informationen über ARN-Rollen finden Sie unter AssumeRole in der AWS Security Token Service-API-Referenz.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
preferred_role	Optional	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Sitzungsdauer

Die Dauer der Rollen-Sitzung in Sekunden. Weitere Informationen finden Sie unter AssumeRole in der AWS Security Token Service-API-Referenz.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
duration	Optional	`900`	`duration=900;`

URL zur Anmeldung

Die Single-Sign-On-URL, die für Ihre Anwendung angezeigt wird.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
login_url	Erforderlich	none	`login_url=https://trial-1234567.okta.com/app/trial-1234567_oktabrowsersaml_1/zzz4izzzAzDFBzZz1234/sso/saml;`

Listener-Port

Die Portnummer, die verwendet wird, um auf die SAML-Antwort zu warten. Dieser Wert sollte mit der IAM-Identity-Center-URL übereinstimmen, mit der Sie den IdP konfiguriert haben (z. B. `http://localhost:7890/athena`).

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
listen_port	Optional	`7890`	`listen_port=7890;`

Zeitüberschreitung

Die Dauer in Sekunden, bis das Plugin nicht mehr auf die SAML-Antwort des Identitätsanbieters wartet.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
timeout	Optional	`120`	`timeout=90;`

Browser SSO OIDC

Browser SSO OIDC ist ein Authentifizierungs-Plugin, das mit AWS IAM Identity Center funktioniert. Informationen zur Aktivierung und Verwendung von IAM Identity Center finden Sie unter Schritt 1: IAM Identity Center aktivieren im AWS IAM Identity Center-Benutzerhandbuch.

Authentifizierungstyp

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
AuthenticationType	Erforderlich	`IAM Credentials`	`AuthenticationType=BrowserSSOOIDC;`

Start-URL von IAM Identity Center

Die URL für das AWS-Zugriffsportal. Die API-Aktion StartDeviceAuthorization von IAM Identity Center verwendet diesen Wert für den Parameter `startUrl`.

Die URL des AWS-Zugriffsportals kopieren

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die AWS IAM Identity Center-Konsole unter https://console.aws.amazon.com/singlesignon/.
2. Wählen Sie im Navigationsbereich Settings (Einstellungen).
3. Wählen Sie auf der Seite Einstellungen unter Identitätsquelle das Zwischenablagesymbol für die AWS-Zugangsportal-URL aus.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
sso_oidc_start_url	Erforderlich	none	`sso_oidc_start_url=https://app_id.awsapps.com/start;`

Region des IAM-Identitätszentrums

Die AWS-Region, in der Ihr SSO konfiguriert ist. Die `SSOOIDCClient`- und `SSOClient`-AWS-SDK-Clients verwenden diesen Wert für den `region`-Parameter.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
sso_oidc_region	Erforderlich	none	`sso_oidc_region=us-east-1;`

Bereiche

Die Liste der Bereiche, die vom Client definiert sind. Bei der Autorisierung schränkt diese Liste die Berechtigungen ein, wenn ein Zugriffstoken gewährt wird. Die API-Aktion RegisterClient von IAM Identity Center verwendet diesen Wert für den Parameter `scopes`.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
sso_oidc_scopes	Optional	none	`sso_oidc_scopes=scope1,scope2,scope3;`

Konto-ID

Die Kennung für das AWS-Konto, das dem Benutzer zugewiesen ist. Die API GetRoleCredentials von IAM Identity Center verwendet diesen Wert für den Parameter `accountId`.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
sso_oidc_account_id	Erforderlich	none	`sso_oidc_account_id=123456789123;`

Rollenname

Der freundliche Name der Rolle, die dem Benutzer zugewiesen ist. Der Name, den Sie für diesen Berechtigungssatz angeben, wird im AWS-Zugangsportal als verfügbare Rolle angezeigt. Die API-Aktion GetRoleCredentials von IAM Identity Center verwendet diesen Wert für den Parameter `roleName`.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
sso_oidc_role_name	Erforderlich	none	`sso_oidc_role_name=AthenaReadAccess;`

Zeitüberschreitung

Die Anzahl der Sekunden, für die die SSO-API für Abfragen nach dem Zugriffstoken suchen soll.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
sso_oidc_timeout	Optional	`120`	`sso_oidc_timeout=60;`

Dateicache aktivieren

Aktiviert einen Cache für temporäre Anmeldeinformationen. Mit diesem Verbindungsparameter können temporäre Anmeldeinformationen zwischengespeichert und zwischen mehreren Prozessen wiederverwendet werden. Verwenden Sie diese Option, um die Anzahl der geöffneten Browserfenster zu reduzieren, wenn Sie BI-Tools wie Microsoft Power BI verwenden.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
sso_oidc_cache	Optional	1	`sso_oidc_cache=0;`

Standard-Anmeldeinformationen

Sie können die Standard-Anmeldeinformationen verwenden, die Sie auf Ihrem Client-System konfigurieren, um eine Verbindung zu Amazon Athena herzustellen. Informationen zur Verwendung von Standard-Anmeldeinformationen finden Sie unter Verwendung der Standard-Anbieterkette für Anmeldeinformationen im AWS SDK for Java-Entwicklerhandbuch.

Authentifizierungstyp

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
AuthenticationType	Erforderlich	`IAM Credentials`	`AuthenticationType=Default Credentials;`

Externe Anmeldeinformationen

Externe Anmeldeinformationen sind ein generisches Authentifizierungs-Plugin, mit dem Sie eine Verbindung zu jedem externen SAML-basierten Identitätsanbieter herstellen können. Um das Plugin zu verwenden, übergeben Sie eine ausführbare Datei, die eine SAML-Antwort zurückgibt.

Authentifizierungstyp

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
AuthenticationType	Erforderlich	`IAM Credentials`	`AuthenticationType=External Credentials;`

Pfad der ausführbaren Datei

Der Pfad zur ausführbaren Datei, die der Logik Ihres benutzerdefinierten SAML-basierten Anmeldeinformationsanbieters entspricht. Die Ausgabe der ausführbaren Datei muss die geparste SAML-Antwort des Identitätsanbieters sein.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
ExecutablePath	Erforderlich	none	ExecutablePath=C:\Users*user_name* *external_credential.exe*

Liste der Argumente

Die Liste der Argumente, die Sie an die ausführbare Datei übergeben wollen.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
ArgumentList	Optional	none	ArgumentList= *arg1 arg2 arg3*

Instance-Profil

Dieser Authentifizierungstyp wird in EC2-Instances verwendet und über den Amazon-EC2-Metadaten-Service bereitgestellt.

Authentifizierungstyp

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
AuthenticationType	Erforderlich	IAM Credentials	AuthenticationType=Instance Profile;

JWT

Das JWT-Plugin (JSON Web Token) bietet eine Schnittstelle, die JSON-Web-Tokens verwendet, um eine Amazon IAM-Rolle zu übernehmen. Die Konfiguration hängt vom jeweiligen Identitätsanbieter

ab. Informationen zur Konfiguration des Verbunds für Google Cloud und AWS finden Sie unter Konfigurieren Sie Workload-Identitätsverbund mit AWS oder Azure in der Google Cloud-Dokumentation.

Authentifizierungstyp

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
AuthenticationType	Erforderlich	`IAM Credentials`	`AuthenticationType=JWT;`

Bevorzugte Rolle

Der angenommene Amazon-Ressourcenname (ARN) der Rolle. Weitere Informationen über ARN-Rollen finden Sie unter AssumeRole in der AWS Security Token Service-API-Referenz.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
preferred_role	Optional	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Sitzungsdauer

Die Dauer der Rollen-Sitzung in Sekunden. Weitere Informationen zur Sitzungsdauer finden Sie unter AssumeRole in der AWS Security Token Service-API-Referenz.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
duration	Optional	`900`	`duration=900;`

JSON Web Token

Das JSON-Webtoken, das zum Abrufen temporärer IAM-Anmeldeinformationen mithilfe der AWS STS-API-Aktion AssumeRoleWithWebIdentity verwendet wird. Informationen zum Generieren von JSON-Webtoken für Nutzer der Google Cloud Platform (GCP) finden Sie in der Google-Cloud-Dokumentation unter Verwenden von JWT-OAuth-Token.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
`web_identity_token`	Erforderlich	`none`	`web_identity_token=eyJhbGc...<remainder of token>;`

Rollensitzungsname

Ein Name für die Sitzung. Eine gängige Methode besteht darin, den Namen oder die Kennung des Benutzers Ihrer Anwendung als Namen für die Rollensitzung zu verwenden. Dadurch werden die temporären Anmeldeinformationen, die Ihre Anwendung verwendet, komfortabel dem entsprechenden Benutzer zugeordnet.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
role_session_name	Erforderlich	`none`	`role_session_name=familiarname;`

Okta

Okta ist ein SAML-basiertes Authentifizierungs-Plugin, das mit dem Okta-Identitätsanbieter zusammenarbeitet. Informationen zur Konfiguration des Verbunds für Okta und Amazon Athena finden Sie unter Konfigurieren von SSO für ODBC mit dem Okta-Plug-In und einem Okta-Identitätsanbieter.

Authentifizierungstyp

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
AuthenticationType	Erforderlich	`IAM Credentials`	`AuthenticationType=Okta;`

Benutzer-ID

Ihr Okta-Benutzername.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
Benutzerkennung (UID)	Erforderlich	none	`UID=jane.doe@org.com;`

Passwort

Ihr Okta-Benutzer-Passwort.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
PWD	Erforderlich	none	`PWD=oktauserpasswordexample;`

Bevorzugte Rolle

Der angenommene Amazon-Ressourcenname (ARN) der Rolle. Weitere Informationen über ARN-Rollen finden Sie unter AssumeRole in der AWS Security Token Service-API-Referenz.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
preferred_role	Optional	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Sitzungsdauer

Die Dauer der Rollen-Sitzung in Sekunden. Weitere Informationen finden Sie unter AssumeRole in der AWS Security Token Service-API-Referenz.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
duration	Optional	`900`	`duration=900;`

IdP-Host

Die URL für Ihre Okta-Organisation. Sie können den Parameter `idp_host` aus der URL Link einbetten in Ihrer Okta-Anwendung extrahieren. Informationen zu den erforderlichen Schritten finden Sie unter Abrufen von ODBC-Konfigurationsinformationen von Okta. Das erste Segment nach `https://` bis einschließlich `okta.com`, ist Ihr IdP-Host (zum Beispiel `http://trial-1234567.okta.com`).

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
idp_host	Erforderlich	`None`	`idp_host=dev-99999999.okta.com;`

IdP-Port

Die Portnummer, die für die Verbindung mit Ihrem IdP-Host verwendet werden soll.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
idp_port	Erforderlich	None	`idp_port=443;`

Okta-App-ID

Der zweiteilige Identifikator für Ihre Anwendung. Sie können den Parameter `app_id` aus der URL Link einbetten in Ihrer Okta-Anwendung extrahieren. Informationen zu den erforderlichen Schritten finden Sie unter Abrufen von ODBC-Konfigurationsinformationen von Okta. Die Anwendungs-ID besteht aus den letzten beiden Segmenten der URL, einschließlich des Schrägstrichs in der Mitte. Bei den Segmenten handelt es sich um zwei 20-stellige Zeichenfolgen, die eine Mischung aus Zahlen sowie Groß- und Kleinbuchstaben (z. B. `Abc1de2fghi3J45kL678/abc1defghij2klmNo3p4`) enthalten.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
app_id	Erforderlich	None	`app_id=0oa25kx8ze9A3example/alnexamplea0piaWa0g7;`

Name der Okta-Anwendung

Der Name der Okta-Anwendung.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
app_name	Erforderlich	None	app_name=amazon_aws_redshift;

Okta-Wartezeit

Gibt die Wartezeit in Sekunden auf den Multifaktor-Authentifizierungscode (MFA) an.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
okta_mfa_wait_time	Optional	10	okta_mfa_wait_time=20;

Okta-MFA-Typ

Der MFA-Faktortyp. Unterstützte Typen sind Google Authenticator, SMS (Okta), Okta Verify with Push und Okta Verify with TOTP. Die Sicherheitsrichtlinien der einzelnen Organisationen bestimmen, ob MFA für die Benutzeranmeldung erforderlich ist oder nicht.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Mögliche Werte	Beispiel für Verbindungszeichenfolgen
okta_mfa_type	Optional	None	googleauthenticator, smsauthentication, oktaverifywithpush, oktaverifywithtotp	okta_mfa_type=oktaverifywithpush;

Okta-Telefonnummer anrufen

Die Telefonnummer, die für die Authentifizierung mit AWS SMS verwendet werden soll. Dieser Parameter ist nur für die Multifaktor-Anmeldung erforderlich. Wenn Ihre Handynummer bereits registriert ist oder wenn die Sicherheitsrichtlinie keine AWS SMS-Authentifizierung verwendet, können Sie dieses Feld ignorieren.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
okta_mfa_phone_number	Für die MFA-Anmeldung erforderlich, andernfalls optional	None	`okta_mfa_phone_number=19991234567;`

Okta-Dateicache aktivieren

Aktiviert einen Cache für temporäre Anmeldeinformationen. Mit diesem Verbindungsparameter können temporäre Anmeldeinformationen zwischengespeichert und zwischen den verschiedenen Prozessen, die von BI-Anwendungen geöffnet werden, wiederverwendet werden. Verwenden Sie diese Option, um das Drosselungslimit der Okta-API zu umgehen.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
okta_cache	Optional	`0`	`okta_cache=1;`

Ping

Ping ist ein SAML-basiertes Plugin, das mit dem PingFederate-Identitätsanbieter zusammenarbeitet.

Authentifizierungstyp

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
AuthenticationType	Erforderlich	`IAM Credentials`	`AuthenticationType=Ping;`

Benutzer-ID

Der Benutzername für den PingFederate-Server.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
Benutzerkennung (UID)	Erforderlich	none	`UID=pinguser@domain.com;`

Passwort

Das Passwort für den PingFederate-Server.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
PWD	Erforderlich	none	`PWD=pingpassword;`

Bevorzugte Rolle

Der angenommene Amazon-Ressourcenname (ARN) der Rolle. Wenn Ihre SAML-Assertion mehrere Rollen hat, können Sie diesen Parameter angeben, um die Rolle auszuwählen, die übernommen werden soll. Diese Rolle sollte in der SAML-Assertion vorhanden sein. Weitere Informationen über ARN-Rollen finden Sie unter AssumeRole in der AWS Security Token Service-API-Referenz.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
preferred_role	Optional	none	`preferred_role=arn:aws:iam::123456789012:id/user1;`

Sitzungsdauer

Die Dauer der Rollen-Sitzung in Sekunden. Weitere Informationen zur Sitzungsdauer finden Sie unter AssumeRole in der AWS Security Token Service-API-Referenz.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
duration	Optional	`900`	`duration=900;`

IdP-Host

Die Adresse für Ihren Ping-Server. Um Ihre Adresse zu finden, rufen Sie die folgende URL auf und sehen Sie sich das Feld SSO-Anwendungsendpunkt an.

```
https://your-pf-host-#:9999/pingfederate/your-pf-app#/spConnections
```

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
idp_host	Erforderlich	none	`idp_host=ec2-1-83-65-12.compute-1.amazonaws.com;`

IdP-Port

Die Portnummer, die für die Verbindung mit Ihrem IdP-Host verwendet werden soll.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
idp_port	Erforderlich	None	idp_port=443;

Partner-SPID

Die Adresse des Serviceanbieters. Um die Adresse des Serviceanbieters zu finden, besuchen Sie die folgende URL und sehen Sie sich das Feld SSO Application Endpoint an.

```
https://your-pf-host-#:9999/pingfederate/your-pf-app#/spConnections
```

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
partner_spid	Erforderlich	None	partner_spid=https://us-east-1.signin.aws.amazon.com/platform/saml/<...>;

Ping-URI-Parameter

Übergibt ein URI-Argument für eine Authentifizierungsanforderung an Ping. Verwenden Sie diesen Parameter, um die Einzelrollenbeschränkung von Lake Formation zu umgehen. Konfigurieren Sie Ping so, dass der übergebene Parameter erkannt wird, und stellen Sie sicher, dass die übergebene Rolle in der Liste der dem Benutzer zugewiesenen Rollen vorhanden ist. Senden Sie dann eine einzelne Rolle in der SAML-Assertion.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
ping_uri_param	Optional	None	ping_uri_param=role=my_iam_role;

Allgemeine Authentifizierungsparameter

Die Parameter in diesem Abschnitt sind für alle Authentifizierungstypen geläufig, wie angegeben.

Proxy für IdP verwenden

Ermöglicht die Kommunikation zwischen dem Treiber und dem IdP über den Proxy. Diese Option ist für die folgenden Authentifizierungs-Plugins verfügbar:

- AD FS
- Azure AD
- Azure AD Browser
- Browser SSO OIDC
- JWT
- Okta
- Ping

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
UseProxyForIdP	Optional	`0`	`UseProxyForIdP=1;`

Lake Formation verwenden

Verwendet die API-Aktion AssumeDecoratedRoleWithSAML von Lake Formation, um temporäre IAM-Anmeldeinformationen abzurufen, anstatt der AWS STS-API-Aktion AssumeRoleWithSAML. Diese Option ist für die Azure-AD-, Browser-Azure-AD-, Browser-SAML-, Okta-, Ping- und AD-FS-Authentifizierungs-Plugins verfügbar.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
LakeformationEnabled	Optional	`0`	`LakeformationEnabled=1;`

SSL unsicher (IdP)

Deaktiviert SSL bei der Kommunikation mit dem IdP. Diese Option ist für die Azure-AD-, Browser-Azure-AD-, Okta-, Ping- und AD-FS-Authentifizierungs-Plugins verfügbar.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
SSL_Insecure	Optional	`0`	`SSL_Insecure=1;`

Überschreibungen von Endpunkten

Athena-Endpunktüberschreibung

Die `endpointOverride` `ClientConfiguration`-Klasse verwendet diesen Wert und überschreibt den Standard-HTTP-Endpunkt für den Amazon-Athena-Client. Weitere Informationen finden Sie unter AWS-Client-Konfiguration im Entwicklerhandbuch für AWS SDK for C++.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
EndpointOverride	Optional	none	`EndpointOverride=athena.us-west-2.amazonaws.com;`

Athena-Streaming-Endpunkt überschreiben

Die `ClientConfiguration.endpointOverride`-Methode verwendet diesen Wert und überschreibt den Standard-HTTP-Endpunkt für den Amazon-Athena-Streaming-Client. Weitere Informationen finden Sie unter AWS-Client-Konfiguration im Entwicklerhandbuch für AWS SDK for C++. Der Athena Streaming-Service ist über Port 444 verfügbar.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
Streaming EndpointOverride	Optional	none	`StreamingEndpointOverride=athena.us-west-1.amazonaws.com:444;`

Überschreibung von AWS STS-Endpunkten

Die `ClientConfiguration.endpointOverride`-Methode verwendet diesen Wert und überschreibt den Standard-HTTP-Endpunkt für den AWS STS-Client. Weitere Informationen finden Sie unter AWS-Client-Konfiguration im Entwicklerhandbuch für AWS SDK for C++.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
StsEndpointOverride	Optional	none	`StsEndpointOverride=sts.us-west-1.amazonaws.com;`

Überschreiben von Endpunkten in Lake Formation

Die `ClientConfiguration.endpointOverride`-Methode verwendet diesen Wert und überschreibt den Standard-HTTP-Endpunkt für den Lake-Formation-Client. Weitere Informationen finden Sie unter AWS-Client-Konfiguration im Entwicklerhandbuch für AWS SDK for C++.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
LakeFormationEndpointOverride	Optional	none	`LakeFormationEndpointOverride=lakeformation.us-west-1.amazonaws.com;`

Überschreibung von SSO-Endpunkten

Die `ClientConfiguration.endpointOverride`-Methode verwendet diesen Wert und überschreibt den Standard-HTTP-Endpunkt für den SSO-Client. Weitere Informationen finden Sie unter AWS-Client-Konfiguration im Entwicklerhandbuch für AWS SDK for C++.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
SSOEndpointOverride	Optional	none	`SSOEndpointOverride=portal.sso.us-east-2.amazonaws.com;`

Überschreibung von SSO-OIDC-Endpunkten

Die `ClientConfiguration.endpointOverride`-Methode verwendet diesen Wert und überschreibt den Standard-HTTP-Endpunkt für den SSO-OIDC-Client. Weitere Informationen finden Sie unter AWS-Client-Konfiguration im Entwicklerhandbuch für AWS SDK for C++.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
SSOOIDCEndpointOverride	Optional	none	`SSOOIDCEndpointOverride=oidc.us-east-2.amazonaws.com`

Erweiterte Optionen

Abrufgröße

Die maximale Anzahl der Ergebnisse (Reihen), die bei dieser Anfrage zurückzugeben sind. Informationen zu Parametern finden Sie unter GetQuery MaxResults. Für die Streaming-API ist der Höchstwert 10 000 000.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
RowsToFetchPerBlock	Optional	`1000` für Nicht-Streaming `20000` für Streaming	`RowsToFetchPerBlock=20000;`

Wiederverwendung von Ergebnissen aktivieren

Gibt an, ob frühere Abfrageergebnisse wiederverwendet werden können, wenn die Abfrage ausgeführt wird. Informationen zu Parametern finden Sie unter ResultReuseByAgeConfiguration.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
EnableResultReuse	Optional	`0`	`EnableResultReuse=1;`

Höchstalter für die Wiederverwendung von Ergebnissen

Gibt in Minuten das maximale Alter eines vorherigen Abfrageergebnisses an, das Athena bei der Wiederverwendung berücksichtigen sollte. Informationen zu Parametern finden Sie unter ResultReuseByAgeConfiguration.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
ReusedResultMaxAgeInMinutes	Optional	`60`	`ReusedResultMaxAgeInMinutes=90;`

AP-Streaming aktivieren

Wählt aus, ob die Athena-Streaming-API zum Abrufen der Ergebnismenge verwendet werden soll.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
UseResultsetStreaming	Optional	`0`	`UseResultsetStreaming=1;`

S3-Fetcher aktivieren

Ruft das von Athena generierte Ergebnis aus dem Amazon-S3-Bucket ab, indem es direkt mit Amazon S3 interagiert.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
EnableS3Fetcher	Optional	`1`	`EnableS3Fetcher=1;`

Mehrere S3-Threads verwenden

Ruft Daten mithilfe mehrerer Threads von Amazon S3 ab. Wenn diese Option aktiviert ist, wird die im Amazon-S3-Bucket gespeicherte Ergebnisdatei parallel über mehrere Threads abgerufen.

Aktivieren Sie diese Option nur, wenn Sie über eine gute Netzwerkbandbreite verfügen. In unseren Messungen auf einer c5.2xlarge-Instance in EC2 erreichte beispielsweise ein Single-Thread-S3-Client 1 Gbit/s, während S3-Clients mit mehreren Threads einen Netzwerkdurchsatz von 4 Gbit/s erreichten.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
UseMultipleS3-Threads	Optional	`0`	`UseMultipleS3Threads=1;`

Einen einzigen Katalog und ein Schema verwenden

Standardmäßig fragt der ODBC-Treiber Athena ab, um die Liste der verfügbaren Kataloge und Schemas abzurufen. Diese Option zwingt den Treiber, den Katalog und das Schema zu verwenden, welche im Konfigurationsdialogfeld des ODBC-Datenquellenadministrators oder in den Verbindungsparametern angegeben sind.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
UseSingleCatalogAndSchema	Optional	`0`	`UseSingleCatalogAndSchema=1;`

Externe Kataloge abfragen

Gibt an, ob der Treiber externe Kataloge von Athena abfragen muss. Weitere Informationen finden Sie unter Migration zum ODBC-2.x-Treiber.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
QueryExternalCatalogs	Optional	`0`	`QueryExternalCatalogs=1;`

SSL verifizieren

Steuert, ob SSL-Zertifikate verifiziert werden sollen, wenn Sie das AWS SDK verwenden. Dieser Wert wird an den `ClientConfiguration.verifySSL`-Parameter übergeben. Weitere Informationen finden Sie unter AWS -Client-Konfiguration im Entwicklerhandbuch für AWS SDK for C++ .

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
VerifySSL	Optional	1	`VerifySSL=0;`

Größe des S3-Ergebnisblocks

Gibt die Größe des Blocks in Byte an, der für eine einzelne Amazon S3 GetObjectS3-API-Anfrage heruntergeladen werden soll. Der Standardwert ist 67 108 864 (64 MB). Die zulässigen Mindest- und Höchstwerte sind 10 485 760 (10 MB) und 2 146 435 072 (etwa 2 GB).

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
S3 ResultBlockSize	Optional	`67108864`	`S3ResultBlockSize=268435456;`

Länge der Zeichenkettenspalte

Gibt die Spaltenlänge für Spalten mit dem `string` Datentyp an. Da Athena den Apache Hive-Zeichenkettendatentyp verwendet, für den es keine definierte Genauigkeit gibt, ist die von Athena gemeldete Standardlänge 2147483647 (). `INT_MAX` Da BI-Tools normalerweise Speicher für Spalten vorab zuweisen, kann dies zu einem hohen Speicherverbrauch führen. Um dies zu vermeiden, begrenzt der Athena ODBC-Treiber die angegebene Genauigkeit für Spalten des `string` Datentyps und macht den `StringColumnLength` Verbindungsparameter verfügbar, sodass der Standardwert geändert werden kann.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
StringColumnLength	Optional	255	`StringColumnLength=65535;`

Spaltenlänge eines komplexen Typs

Gibt die Spaltenlänge für Spalten mit komplexen Datentypen wie `mapstruct`, und an`array`. Zum Beispiel StringColumnLengthmeldet Athena eine Genauigkeit von 0 für Spalten mit komplexen Datentypen. Der Athena ODBC-Treiber legt die Standardgenauigkeit für Spalten mit komplexen Datentypen fest und macht den `ComplexTypeColumnLength` Verbindungsparameter verfügbar, sodass der Standardwert geändert werden kann.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
ComplexTypeColumnLength	Optional	65535	`ComplexTypeColumnLength=123456;`

Vertrauenswürdiges CA-Zertifikat

Weist den HTTP-Client an, wo er den Vertrauensspeicher für Ihr SSL-Zertifikat findet. Dieser Wert wird an den `ClientConfiguration.caFile`-Parameter übergeben. Weitere Informationen finden Sie unter AWS -Client-Konfiguration im Entwicklerhandbuch für AWS SDK for C++ .

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
TrustedCerts	Optional	`%INSTALL_PATH%/bin`	`TrustedCerts=C:\\Program Files\\Amazon Athena ODBC Driver\\bin\\cacert.pem;`

Minimaler Abfragezeitraum

Gibt den Mindestwert in Millisekunden an, der gewartet werden muss, bevor Athena den Status der Abfrageausführung abfragt.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
MinQueryExecutionPollingInterval	Optional	`100`	`MinQueryExecutionPollingInterval=200;`

Maximaler Abfragezeitraum

Gibt den Höchstwert in Millisekunden an, der gewartet werden muss, bevor Athena den Status der Abfrageausführung abfragt.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
MaxQueryExecutionPollingInterval	Optional	`60000`	`MaxQueryExecutionPollingInterval=1000;`

Multiplikator für Abfragen

Gibt den Faktor für die Verlängerung des Abfragezeitraums an. Standardmäßig beginnt die Abfrage mit dem Wert für den minimalen Abfragezeitraum und verdoppelt sich bei jeder Abfrage, bis der Wert für den maximalen Abfragezeitraum erreicht ist.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
QueryExecutionPollingIntervalMultiplier	Optional	2	`QueryExecutionPollingIntervalMultiplier=2;`

Max. Abfragedauer

Gibt den Höchstwert in Millisekunden an, den ein Treiber bei Athena für den Abfrageausführungsstatus abfragen kann.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
MaxPollDuration	Optional	`1800000`	`MaxPollDuration=1800000;`

Verbindungstimeout

Die Zeit (in Millisekunden), die die HTTP-Verbindung wartet, um eine Verbindung herzustellen. Dieser Wert ist für den `ClientConfiguration.connectTimeoutMs`-Athena-Client festgelegt. Wenn nichts angegeben ist, wird der Standardwert verwendet. Weitere Informationen zu Verbindungsparametern finden Sie unter Client-Konfiguration im AWS SDK for Java - Entwicklerhandbuch.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
ConnectionTimeout	Optional	`0`	`ConnectionTimeout=2000;`

Anforderungs-Timeout

Gibt das Socket-Lese-Timeout für HTTP-Clients an. Dieser Wert ist für den `ClientConfiguration.requestTimeoutMs`-Parameter des Athena-Clients festgelegt. Weitere Informationen finden Sie unter Client-Konfiguration im Entwicklerhandbuch für AWS SDK for Java .

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
RequestTimeout	Optional	`10000`	`RequestTimeout=30000;`

Proxy-Optionen

Proxy-Host

Wenn Benutzer einen Proxy verwenden müssen, verwenden Sie diesen Parameter, um den Proxyhost festzulegen. Dies entspricht dem Parametertyp `ClientConfiguration.proxyHost` im AWS-SDK. Weitere Informationen finden Sie unter AWS-Client-Konfiguration im Entwicklerhandbuch für AWS SDK for C++.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
ProxyHost	Optional	none	`ProxyHost =127.0.0.1;`

Proxy-Port

Verwenden Sie diesen Parameter, um den Proxy-Port festzulegen. Dies entspricht dem Parametertyp `ClientConfiguration.proxyPort` im AWS-SDK. Weitere Informationen finden Sie unter AWS-Client-Konfiguration im Entwicklerhandbuch für AWS SDK for C++.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
ProxyPort	Optional	none	`ProxyPort=8888;`

Proxy-Benutzername

Verwenden Sie diesen Parameter, um den Proxy-Benutzernamen festzulegen. Dies entspricht dem Parametertyp `ClientConfiguration.proxyUserName` im AWS-SDK. Weitere Informationen finden Sie unter AWS-Client-Konfiguration im Entwicklerhandbuch für AWS SDK for C++.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
ProxyUID	Optional	none	`ProxyUID= username;`

Proxy-Passwort

Verwenden Sie diesen Parameter, um das Proxy-Passwort festzulegen. Dies entspricht dem Parametertyp `ClientConfiguration.proxyPassword` im AWS-SDK. Weitere Informationen finden Sie unter AWS-Client-Konfiguration im Entwicklerhandbuch für AWS SDK for C++.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
ProxyPWD	Optional	none	`ProxyPWD=password;`

Host, der kein Proxy ist

Verwenden Sie diesen optionalen Parameter, um einen Host anzugeben, zu dem der Treiber eine Verbindung herstellt, ohne einen Proxy zu verwenden. Dies entspricht dem Parametertyp `ClientConfiguration.nonProxyHosts` im AWS-SDK. Weitere Informationen finden Sie unter AWS-Client-Konfiguration im Entwicklerhandbuch für AWS SDK for C++.

Der Verbindungsparameter `NonProxyHost` wird an die Option `CURLOPT_NOPROXY` curl übergeben. Informationen zum `CURLOPT_NOPROXY`-Format finden Sie unter CURLOPT_NOPROXY in der curl-Dokumentation.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
NonProxyHost	Optional	none	`NonProxyHost=.amazonaws.com,localhost,.example.net,.example.com;`

Verwenden Sie einen Proxy

Aktiviert Benutzerverkehr über den angegebenen Proxy.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
UseProxy	Optional	none	`UseProxy=1;`

Protokollierungsoptionen

Administratorrechte sind erforderlich, um die hier beschriebenen Einstellungen zu ändern. Um die Änderungen vorzunehmen, können Sie das Dialogfeld Protokollierungsoptionen des ODBC-Datenquellenadministrators verwenden oder die Windows-Registrierung direkt ändern.

Protokollebene

Diese Option aktiviert ODBC-Treiberprotokolle. In Windows können Sie die Registrierung oder ein Dialogfeld verwenden, um die Protokollierung zu aktivieren oder zu deaktivieren. Die Option befindet sich im folgenden Registrierungspfad:

```
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Amazon Athena\ODBC\Driver
```

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
LogLevel	Optional	`0`	`LogLevel=1;`

Protokollpfad

Gibt den Pfad zu der Datei an, in der die ODBC-Treiberprotokolle gespeichert sind. Sie können die Registrierung oder ein Dialogfeld verwenden, um diesen Wert festzulegen. Die Option befindet sich im folgenden Registrierungspfad:

```
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Amazon Athena\ODBC\Driver
```

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
LogPath	Optional	none	`LogPath=C:\Users\ username\projects \internal\trunk\;`

AWS Logger verwenden

Gibt an, ob die AWS-SDK-Protokollierung aktiviert ist. Geben Sie 1 an, um zu aktivieren, 0, um zu deaktivieren.

Name der Verbindungszeichenfolge	Parametertyp	Standardwert	Beispiel für Verbindungszeichenfolgen
UseAwsLogger	Optional	`0`	`UseAwsLogger=1;`

Migration zum ODBC-2.x-Treiber

Da die meisten Athena-ODBC-2.x-Verbindungsparameter abwärtskompatibel mit dem ODBC-1.x-Treiber sind, können Sie den größten Teil Ihrer vorhandenen Verbindungszeichenfolge mit dem Athena-ODBC-2.x-Treiber wiederverwenden. Die folgenden Verbindungsparameter müssen jedoch geändert werden.

Protokollebene

Während der aktuelle ODBC-Treiber eine Reihe verfügbarer Protokollierungsoptionen bietet, angefangen bei `LOG_OFF (0)` bis `LOG_TRACE (6)`, hat der ODBC-Treiber von Amazon Athena nur zwei Werte: 0 (deaktiviert) und 1 (aktiviert).

Weitere Hinweise zur Protokollierung des ODBC-2.x-Treibers finden Sie unter Protokollierungsoptionen.

	ODBC-1.x-Treiber	ODBC-2.x-Treiber
Name der Verbindungszeichenfolge	`LogLevel`	`LogLevel`
Parametertyp	Optional	Optional
Standardwert	`0`	`0`
Mögliche Werte	`0-6`	`0,1`
Beispiel für Verbindungszeichenfolgen	`LogLevel=6;`	`LogLevel=1;`

MetadataRetrievalMethod

Der aktuelle ODBC-Treiber bietet mehrere Optionen zum Abrufen der Metadaten von Athena. Der ODBC-Treiber von Amazon Athena veraltet und `MetadataRetrievalMethod` verwendet immer die Amazon-Athena-API, um Metadaten zu extrahieren.

Athena führt die Kennzeichnung `QueryExternalCatalogs` für die Abfrage externer Kataloge ein. Um externe Kataloge mit dem aktuellen ODBC-Treiber abzufragen, setzen Sie `MetadataRetrievalMethod` auf `ProxyAPI`. Um externe Kataloge mit dem ODBC-Treiber von Athena abzufragen, setzen Sie `QueryExternalCatalogs` auf `1`.

	ODBC-1.x-Treiber	ODBC-2.x-Treiber
Name der Verbindungszeichenfolge	`MetadataRetrievalMethod`	`QueryExternalCatalogs`
Parametertyp	Optional	Optional
Standardwert	`Auto`	`0`
Mögliche Werte	`Auto`, `AWS Glue`, `ProxyAPI`, `Query`	`0`,`1`
Beispiel für Verbindungszeichenfolgen	`MetadataRetrievalMethod=ProxyAPI;`	`QueryExternalCatalogs=1;`

Verbindungstest

Wenn Sie eine ODBC-1.x-Treiberverbindung testen, führt der Treiber eine `SELECT 1`-Abfrage aus, die zwei Dateien in Ihrem Amazon-S3-Bucket generiert: eine für die Ergebnismenge und eine für die Metadaten. Die Testverbindung wird gemäß den Amazon-Athena-Preisrichtlinien berechnet.

Wenn Sie eine ODBC-2.x-Treiber-Verbindung testen, ruft der Treiber die API-Aktion GetWorkGroup von Athena auf. Der Aufruf verwendet den Authentifizierungstyp und den entsprechenden Anbieter von Anmeldeinformationen, den Sie zum Abrufen der Anmeldeinformationen angegeben haben. Der Verbindungstest ist kostenlos, wenn Sie den ODBC-2.x-Treiber verwenden, und der Test generiert keine Abfrageergebnisse in Ihrem Amazon-S3-Bucket.

Problembehandlung beim ODBC-2.x-Treiber

Wenn Sie Probleme mit dem ODBC-Treiber von Amazon Athena haben, können Sie sich an AWS Support wenden (wählen Sie im AWS Management Console Support, Support-Center).

Stellen Sie sicher, dass Sie die folgenden Informationen und alle zusätzlichen Details angeben, die dem Support-Team helfen, Ihren Anwendungsfall zu verstehen.

- Beschreibung – (Erforderlich) Eine Beschreibung, die detaillierte Informationen zu Ihrem Anwendungsfall und dem Unterschied zwischen dem erwarteten und dem beobachteten Verhalten enthält. Fügen Sie alle Informationen hinzu, die den Support-Technikern helfen können, das Problem einfach zu lösen. Wenn das Problem zeitweise auftritt, geben Sie die Daten, Zeitstempel oder Intervalle an, an denen das Problem aufgetreten ist.
- Versionsinformationen – (Erforderlich) Informationen über die Treiberversion, das Betriebssystem und die Anwendungen, die Sie verwendet haben. Zum Beispiel „ODBC-Treiberversion 1.2.3, Windows 10 (x64), Power BI“.
- Protokolldateien – (Erforderlich) Die Mindestanzahl von ODBC-Treiberprotokolldateien, die erforderlich sind, um das Problem zu verstehen. Weitere Hinweise zu Protokollierungsoptionen des ODBC-2.x-Treibers finden Sie unter Protokollierungsoptionen.
- Verbindungszeichenfolge – (Erforderlich) Ihre ODBC-Verbindungszeichenfolge oder ein Screenshot des Dialogfelds, in dem die von Ihnen verwendeten Verbindungsparameter angezeigt werden. Weitere Informationen zu Verbindungsparametern finden Sie unter Athena-ODBC-2.x-Verbindungsparameter.
- Problemschritte – (optional) Fügen Sie nach Möglichkeit Schritte oder ein eigenständiges Programm hinzu, mit dem das Problem reproduziert werden kann.
- Fehlerinformationen abfragen – (Optional) Wenn Sie Fehler im Zusammenhang mit DML- oder DDL-Abfragen haben, geben Sie die folgenden Informationen an:
  - Eine vollständige oder vereinfachte Version der fehlgeschlagenen DML- oder DDL-Abfrage.
  - Die verwendete Konto-ID und die AWS-Region, sowie die ID für die Ausführung der Abfrage.
- SAML-Fehler – (Optional) Wenn Sie ein Problem mit der Authentifizierung mit SAML-Assertion haben, geben Sie die folgenden Informationen an:
  - Der Identitätsanbieter und das Authentifizierungs-Plugin, die verwendet wurden.
  - Ein Beispiel mit dem SAML-Token.

Versionshinweise von Amazon Athena ODBC 2.x

Diese Versionshinweise enthalten Einzelheiten zu Verbesserungen, Features, bekannten Problemen und Workflow-Änderungen im Amazon Athena-ODBC-2.x-Treiber.

2.0.2.2

Veröffentlicht am 2024-02-13

Der Amazon Athena-ODBC-v2.0.2.2-Treiber enthält die folgenden Verbesserungen und Korrekturen.

Verbesserungen

- Es wurden zwei Verbindungsparameter hinzugefügt, `StringColumnLength` und `ComplexTypeColumnLength`, mit denen Sie die Standardspaltenlänge für Zeichenfolgen- und komplexe Datentypen ändern können. Weitere Informationen finden Sie unter Länge der Zeichenkettenspalte und Spaltenlänge eines komplexen Typs.
- Unterstützung für die Betriebssysteme Linux und macOS (Intel und ARM) wurde hinzugefügt. Weitere Informationen finden Sie unter Linux und macOS.
- AWS-SDK-CPP wurde auf die Tag-Version 1.11.245 aktualisiert.
- Die curl-Bibliothek wurde auf die Version 8.6.0 aktualisiert.

Behobene Probleme

- Es wurde ein Problem behoben, das dazu führte, dass falsche Werte in Ergebnissatzmetadaten für zeichenfolgenähnliche Datentypen in der Spalte Präzision gemeldet wurden.

Informationen zum Herunterladen des ODBC-v2-Treibers finden Sie unter ODBC-2.x-Treiber-Download. Verbindungsinformationen finden Sie unter ODBC-2.x-Verbindungen von Amazon Athena konfigurieren.

2.0.2.1

Veröffentlicht am 07.12.2023

Der ODBC-v2.0.2.1-Treiber von Amazon Athena enthält die folgenden Verbesserungen und Korrekturen.

Verbesserungen

- Verbesserte Thread-Sicherheit des ODBC-Treibers für alle Schnittstellen.
- Wenn die Protokollierung aktiviert ist, werden Datum/Uhrzeit-Werte jetzt mit Millisekunden-Präzision aufgezeichnet.
- Bei der Authentifizierung mit dem Browser SSO OIDC-Plugin wird nun das Terminal geöffnet, um dem Benutzer den Gerätecode anzuzeigen.

Behobene Probleme

- Es wurde ein Problem mit der Speicherfreigabe behoben, das beim Parsen von Ergebnissen aus der Streaming-API auftrat.
- Anfragen für die Schnittstellen `SQLTablePrivileges()`, `SQLSpecialColumns()`, `SQLProcedureColumns()` und `SQLProcedures()` geben nun leere Ergebnismengen zurück.

Informationen zum Herunterladen des ODBC-v2-Treibers finden Sie unter ODBC-2.x-Treiber-Download. Verbindungsinformationen finden Sie unter ODBC-2.x-Verbindungen von Amazon Athena konfigurieren.

2.0.2.0

Veröffentlicht am 17.10.2023

Der Amazon-Athena-ODBC-v2.0.2.0-Treiber enthält die folgenden Verbesserungen und Korrekturen.

Verbesserungen

- Das Datei-Cache-Feature wurde für die browserbasierten Authentifizierungs-Plugins Browser Azure AD, Browser SSO OIDC und Okta hinzugefügt.

  BI-Tools wie Power BI und browserbasierte Plugins verwenden mehrere Browserfenster. Mit dem neuen Datei-Cache-Verbindungsparameter können temporäre Anmeldeinformationen zwischengespeichert und zwischen den verschiedenen Prozessen, die von BI-Anwendungen geöffnet werden, wiederverwendet werden.
- Anwendungen können jetzt Informationen über die Ergebnismenge abfragen, nachdem eine Anweisung vorbereitet wurde.

- Die standardmäßigen Verbindungs- und Anforderungs-Timeouts wurden für die Verwendung in langsameren Client-Netzwerken erhöht. Weitere Informationen finden Sie unter Verbindungstimeout und Anforderungs-Timeout.
- Endpunktüberschreibungen wurden für SSO und SSO OIDC hinzugefügt. Weitere Informationen finden Sie unter Überschreibungen von Endpunkten.
- Es wurde ein Verbindungsparameter hinzugefügt, um ein URI-Argument für eine Authentifizierungsanforderung an Ping zu übergeben. Sie können diesen Parameter verwenden, um die Einzelrollenbeschränkung von Lake Formation zu umgehen. Weitere Informationen finden Sie unter Ping-URI-Parameter.

Behobene Probleme

- Es wurde ein Problem mit einem Ganzzahlüberlauf behoben, das bei der Verwendung des zeilenbasierten Bindungsmechanismus auftrat.
- Das Timeout wurde aus der Liste der erforderlichen Verbindungsparameter für das Browser-SSO-OIDC-Authentifizierungs-Plugin entfernt.
- Fehlende Schnittstellen für `SQLStatistics()`, `SQLPrimaryKeys()`, `SQLForeignKeys()` und `SQLColumnPrivileges()` wurden hinzugefügt und es wurde die Möglichkeit hinzugefügt, auf Anfrage leere Ergebnissätze zurückzugeben.

Informationen zum Herunterladen des neuesten ODBC-v2-Treibers finden Sie unter ODBC-2.x-Treiber-Download. Verbindungsinformationen finden Sie unter ODBC-2.x-Verbindungen von Amazon Athena konfigurieren.

2.0.1.1

Veröffentlicht am 10.08.2023

Der Amazon-Athena-ODBC-v2.0.1.1-Treiber enthält die folgenden Verbesserungen und Korrekturen.

Verbesserungen

- URI-Protokollierung zum Okta-Authentifizierungs-Plugin hinzugefügt.
- Der bevorzugte Rollenparameter wurde dem Plug-in für externe Anmeldeinformationen hinzugefügt.
- Es wurde eine Behandlung für das Profilpräfix im Profilnamen der AWS -Konfigurationsdatei hinzugefügt.

Behobene Probleme

- Es wurde ein AWS-Region Nutzungsproblem behoben, das bei der Arbeit mit Lake Formation und AWS STS Clients auftrat.
- Fehlende Partitionsschlüssel wurden in der Liste der Tabellenspalten wiederhergestellt.
- Der fehlende `BrowserSSOOIDC`-Authentifizierungstyp wurde dem AWS -Profil hinzugefügt.

Informationen zum Herunterladen des neuesten ODBC-v2-Treibers finden Sie unter ODBC-2.x-Treiber-Download.

2.0.1.0

Veröffentlicht am 29.06.2023

Amazon Athena veröffentlicht den ODBCv2.0.1.0-Treiber.

Athena hat einen neuen ODBC-Treiber veröffentlicht, der die Verbindung, Abfrage und Visualisierung von Daten aus kompatiblen SQL-Entwicklungs- und Business-Intelligence-Anwendungen verbessert. Die neueste Version des Athena-ODBC-Treibers unterstützt die Feature des vorhandenen Treibers und ist einfach zu aktualisieren. Die neue Version bietet Unterstützung für die Authentifizierung von Benutzern über AWS IAM Identity Center. Es bietet auch die Möglichkeit, Abfrageergebnisse aus Amazon S3 zu lesen, wodurch Ihnen Abfrageergebnisse früher zur Verfügung stehen können.

Weitere Informationen finden Sie unter ODBC-2.x-Verbindungen von Amazon Athena konfigurieren.

Athena-ODBC-1.x-Treiber

Verwenden Sie die Links auf dieser Seite, um die Lizenzvereinbarung für den 1.x-ODBC-Treiber von Amazon Athena, die ODBC-Treiber und die ODBC-Dokumentation herunterzuladen. Weitere Informationen zu der ODBC-Verbindungszeichenfolge finden Sie in der PDF-Datei „ODBC Driver Installation and Configuration Guide“ (Installations- und Konfigurationshandbuch zum ODBC-Treiber), die Sie von dieser Seite herunterladen können. Weitere Informationen zu Berechtigungen finden Sie unter Zugriff über JDBC- und ODBC-Verbindungen.

> **Important**
>
> Beachten Sie bei der Verwendung des ODBC-1.x-Treibers unbedingt die folgenden Anforderungen:

- Open port 444 – Halten Sie Port 444, den Athena zum Streamen von Abfrageergebnissen verwendet, für ausgehenden Datenverkehr geöffnet. Wenn Sie einen PrivateLink-Endpunkt für die Verbindung mit Athena verwenden, stellen Sie sicher, dass die Sicherheitsgruppe, die an den PrivatLink-Endpunkt angeschlossen ist, für eingehenden Datenverkehr an Port 444 geöffnet ist.
- athena:GetQueryResultsStream-Richtlinie – Fügen Sie die `athena:GetQueryResultsStream`-Richtlinienaktion den IAM-Prinzipalen hinzu, die den ODBC-Treiber verwenden. Diese Richtlinienaktion wird nicht direkt mit der API bereitgestellt. Sie wird nur mit dem JDBC-Treiber als Teil der Unterstützung von Streaming-Ergebnissen verwendet. Eine Beispielrichtlinie finden Sie unter AWS Verwaltete Richtlinie: AWSQuicksightAthenaAccess.

Windows

Treiberversion	Download-Link
ODBC 1.2.1.1000 für Windows 32-Bit	Windows-32-Bit-ODBC-Treiber 1.2.1.1000
ODBC 1.2.1.1000 für Windows 64-Bit	Windows-64-Bit-ODBC-Treiber 1.2.1.1000

Linux

Treiberversion	Download-Link
ODBC 1.2.1.1000 für Linux 32-Bit	ODBC-Treiber 1.2.1.1000 für Linux 32-Bit
ODBC 1.2.1.1000 für Linux 64-Bit	ODBC-Treiber 1.2.1.1000 für Linux 64-Bit

OSX

Treiberversion	Download-Link
ODBC 1.2.1.1000 für OSX	ODBC-Treiber 1.2.1.1000 für OSX

Dokumentation

Inhalt	Link zur Dokumentation
Amazon-Athena-ODBC-Treiber-Lizenzvereinbarung	Lizenzvereinbarung
Dokumentation für ODBC 1.2.1.1000	Installations- und Konfigurationshandbuch für ODBC-Treiber-Version 1.2.1.1000
Versionshinweise für ODBC 1.2.1.1000	ODBC-Treiber Versionshinweise für Version 1.2.1.1000

Hinweise zum ODBC-Treiber

Herstellen einer Verbindung ohne Verwendung eines Proxys

Wenn Sie bestimmte Hosts angeben möchten, mit denen der Treiber eine Verbindung ohne Verwendung eines Proxys herstellt, können Sie die optionale `NonProxyHost`-Eigenschaft in Ihrer ODBC-Verbindungszeichenfolge verwenden.

Die `NonProxyHost`-Eigenschaft gibt eine durch Kommas getrennte Liste von Hosts an, auf die der Connector zugreifen kann, ohne den Proxyserver zu durchlaufen, wenn eine Proxyverbindung aktiviert ist, wie im folgenden Beispiel:

```
.amazonaws.com,localhost,.example.net,.example.com
```

Der Verbindungsparameter `NonProxyHost` wird an die Option `CURLOPT_NOPROXY` curl übergeben. Informationen zum `CURLOPT_NOPROXY`-Format finden Sie unter CURLOPT_NOPROXY in der curl-Dokumentation.

Konfigurieren des Verbundzugriffs auf Amazon Athena für Microsoft-AD-FS-Benutzer mithilfe eines ODBC-Clients

Um den Verbundzugriff auf Amazon Athena für Benutzer von Microsoft Active Directory Federation Services (AD FS) mithilfe eines ODBC-Clients einzurichten, stellen Sie zunächst eine Vertrauensstellung zwischen AD FS und Ihrem AWS-Konto her. Mit dieser Vertrauensstellung können sich Ihre AD-Benutzer zusammenschließen zu AWS, um ihre AD-Anmeldeinformationen zu verwenden und Berechtigungen einer AWS Identity and Access Management (IAM)-Rolle für den Zugriff auf AWS-Ressourcen wie die Athena-API zu übernehmen.

Um diese Vertrauensstellung zu erstellen, fügen Sie Ihrem AWS-Konto AD FS als SAML-Anbieter hinzu und erstellen eine IAM-Rolle, die Verbundbenutzer annehmen können. Auf der AD-FS-Seite fügen Sie AWS als vertrauende Partei hinzu und erstellen Sie SAML-Antragsregeln, um die richtigen Benutzerattribute für die Autorisierung an AWS zu senden (insbesondere Athena und Amazon S3).

Die Konfiguration des AD-FS-Zugriffs auf Athena umfasst die folgenden Hauptschritte:

1. Einrichtung eines IAM-SAML-Anbieters und einer Rolle

2. Konfigurieren von AD FS

3. Erstellen von Active-Directory-Benutzern und -Gruppen

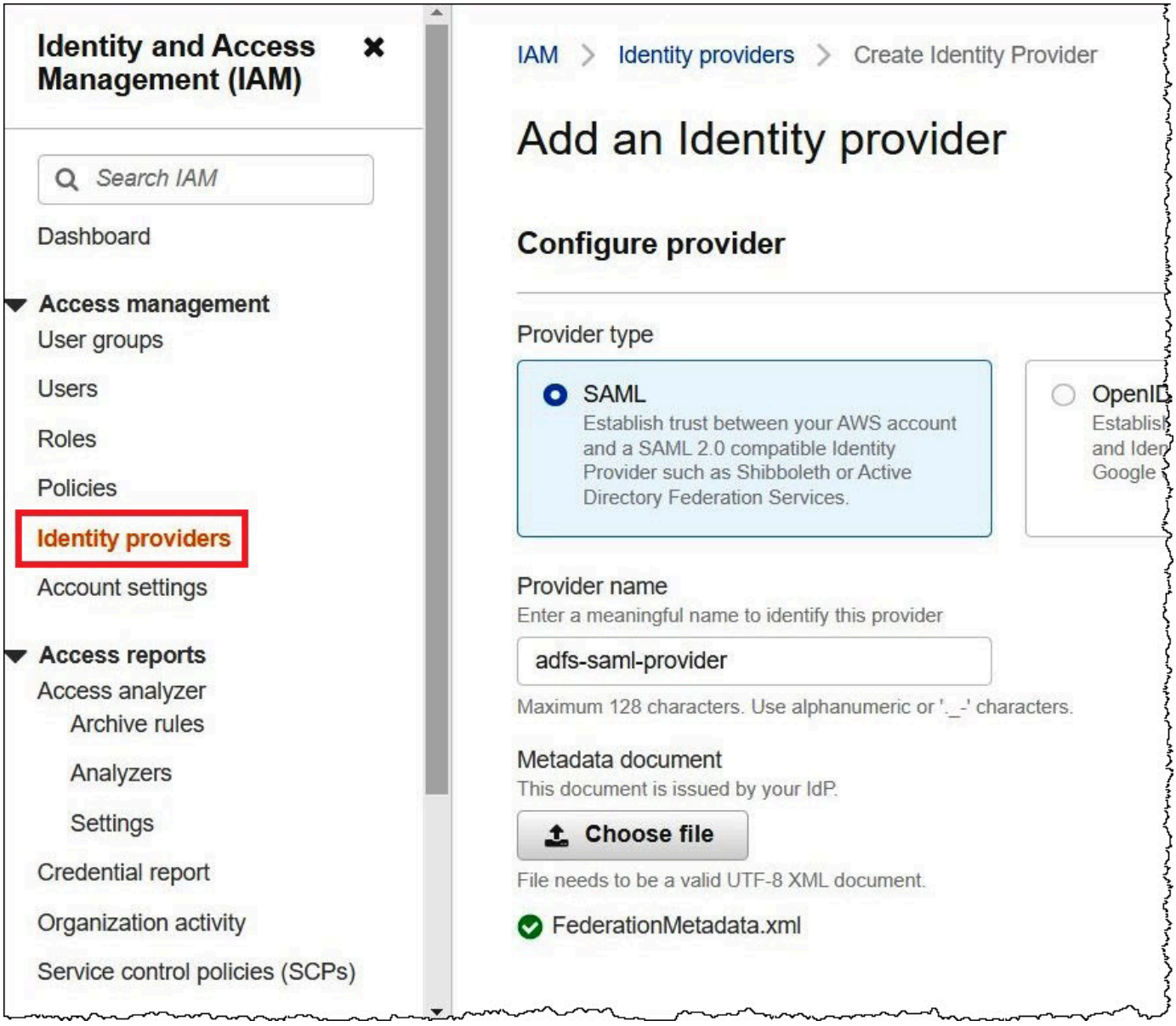
4. Konfigurieren der AD-FS-ODBC-Verbindung zu Athena

1. Einrichtung eines IAM-SAML-Anbieters und einer Rolle

In diesem Abschnitt fügen Sie Ihrem AWS-Konto AD FS als SAML-Anbieter hinzu und erstellen eine IAM-Rolle, die Ihre Verbundbenutzer annehmen können.

So richten Sie einen SAML-Anbieter ein

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Identitätsanbieter.
3. Wählen Sie Add provider (Anbieter hinzufügen) aus.
4. Wählen Sie als Provider type (Anbietertyp) SAML aus.



5. Geben Sie für Anbietername **`adfs-saml-provider`** ein.
6. Geben Sie in einem Browser die folgende Adresse ein, um die Verbund-XML-Datei für Ihren AD-FS-Server herunterzuladen. Um diesen Schritt auszuführen, muss Ihr Browser über Zugriff auf den AD-FS-Server verfügen.

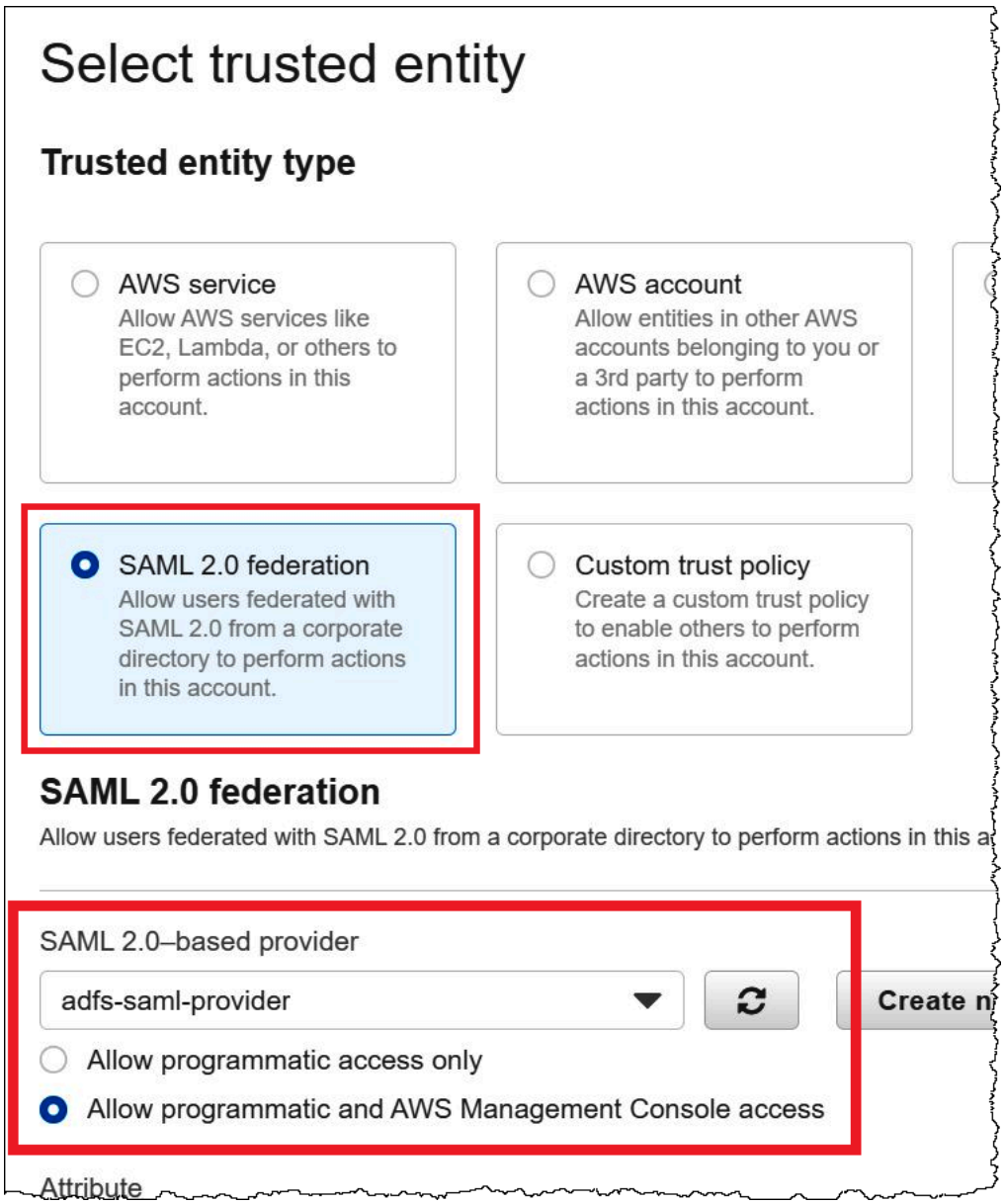
```
https://adfs-server-name/federationmetadata/2007-06/federationmetadata.xml
```

7. Wählen Sie in der IAM-Konsole unter Metadata document (Metadatendokument) die Option Choose file (Datei auswählen) aus und laden Sie anschließend die Verbund-Metadatendatei in AWS hoch.
8. Wählen Sie abschließend Add provider (Anbieter hinzufügen).

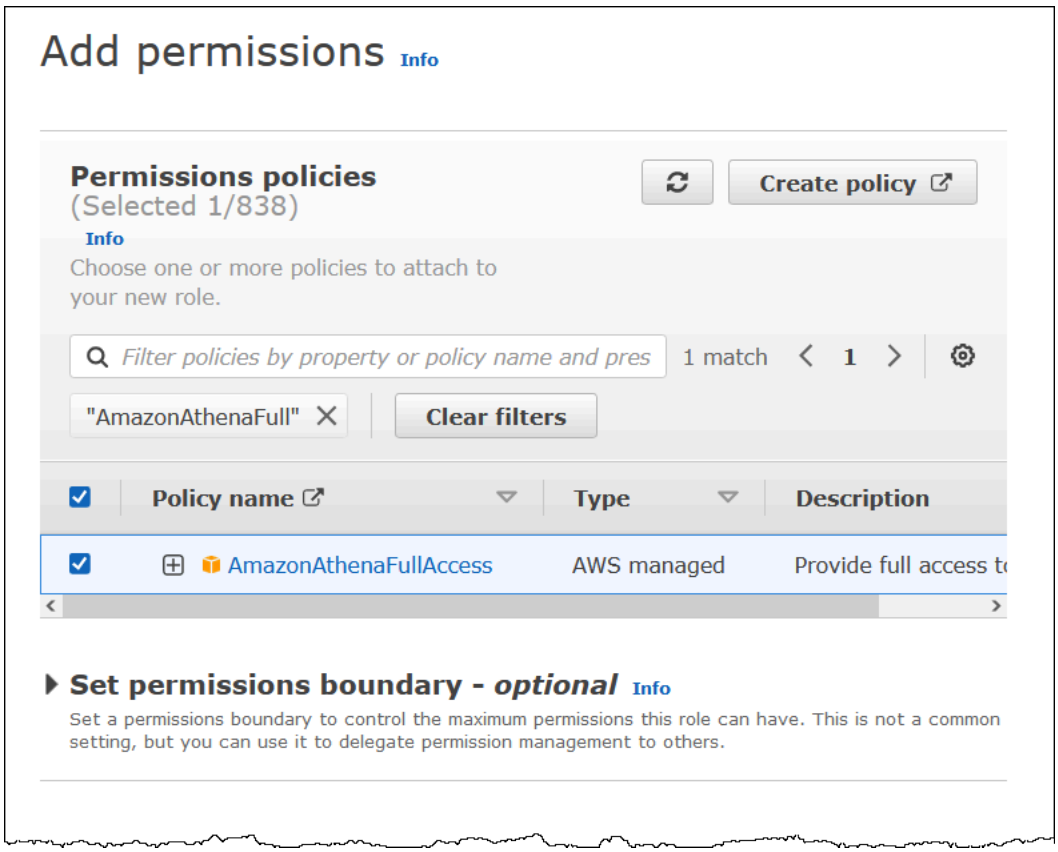
Als Nächstes erstellen Sie die IAM-Rolle, die Ihre Verbundbenutzer annehmen können.

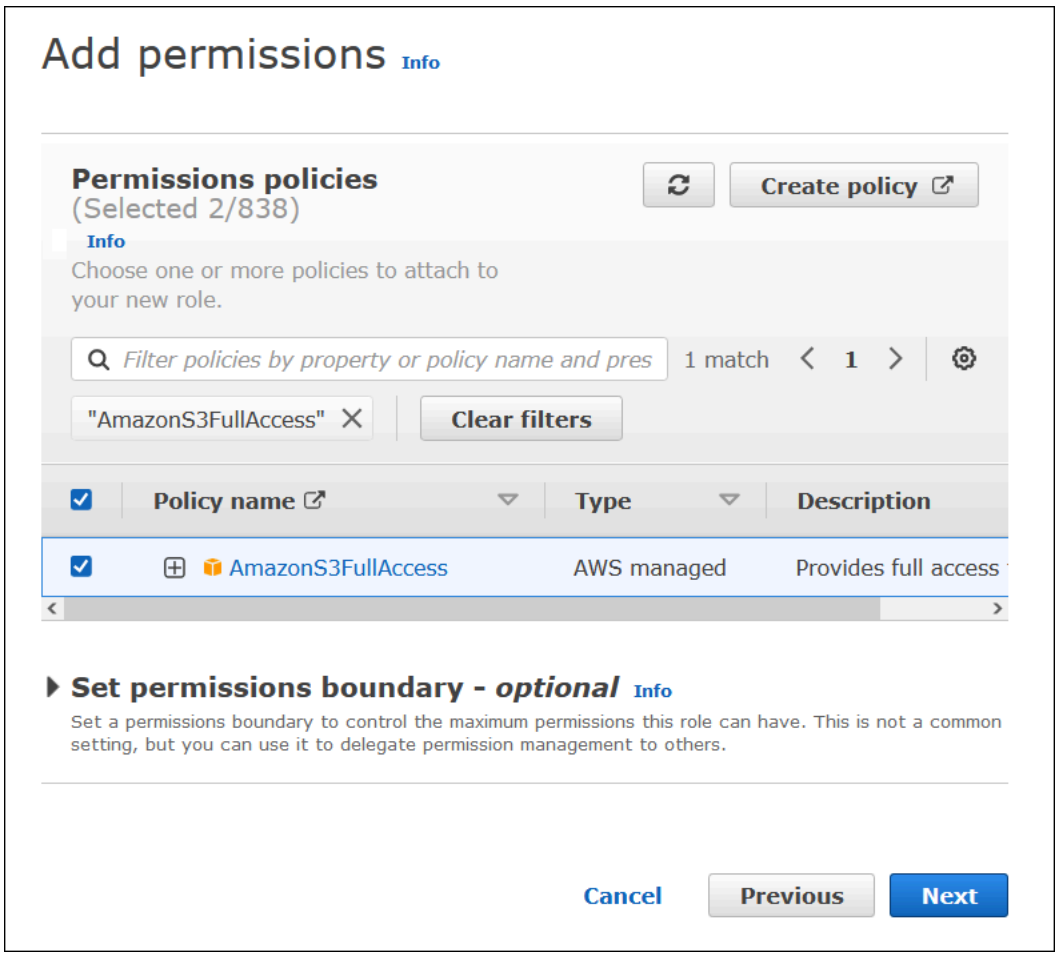
So erstellen Sie eine IAM-Rolle für Verbundbenutzer

1. Wählen Sie im Navigationsbereich der IAM-Konsole Roles (Rollen) aus.
2. Wählen Sie Create role (Rolle erstellen) aus.
3. Wählen Sie als Trusted entity type (Typ der vertrauenswürdigen Entität) die Option SAML 2.0 federation (SAML 2.0-Verbund) aus.
4. Wählen Sie für SAML 2.0-based provider (SAML 2.0-basierte Anbieter) den von Ihnen erstellten Anbieter adfs-saml-provider aus.
5. Wählen Sie Zugriff programmgesteuert und über die AWS-Managementkonsole zulassen und wählen Sie anschließend Weiter aus.



6. Filtern Sie auf der Seite Add permissions (Berechtigungen hinzufügen) nach den IAM-Berechtigungsrichtlinien, die Sie für diese Rolle benötigen, und aktivieren Sie dann die entsprechenden Kontrollkästchen. In diesem Tutorial werden die `AmazonAthenaFullAccess`- und `AmazonS3FullAccess`-Richtlinien angefügt.


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.

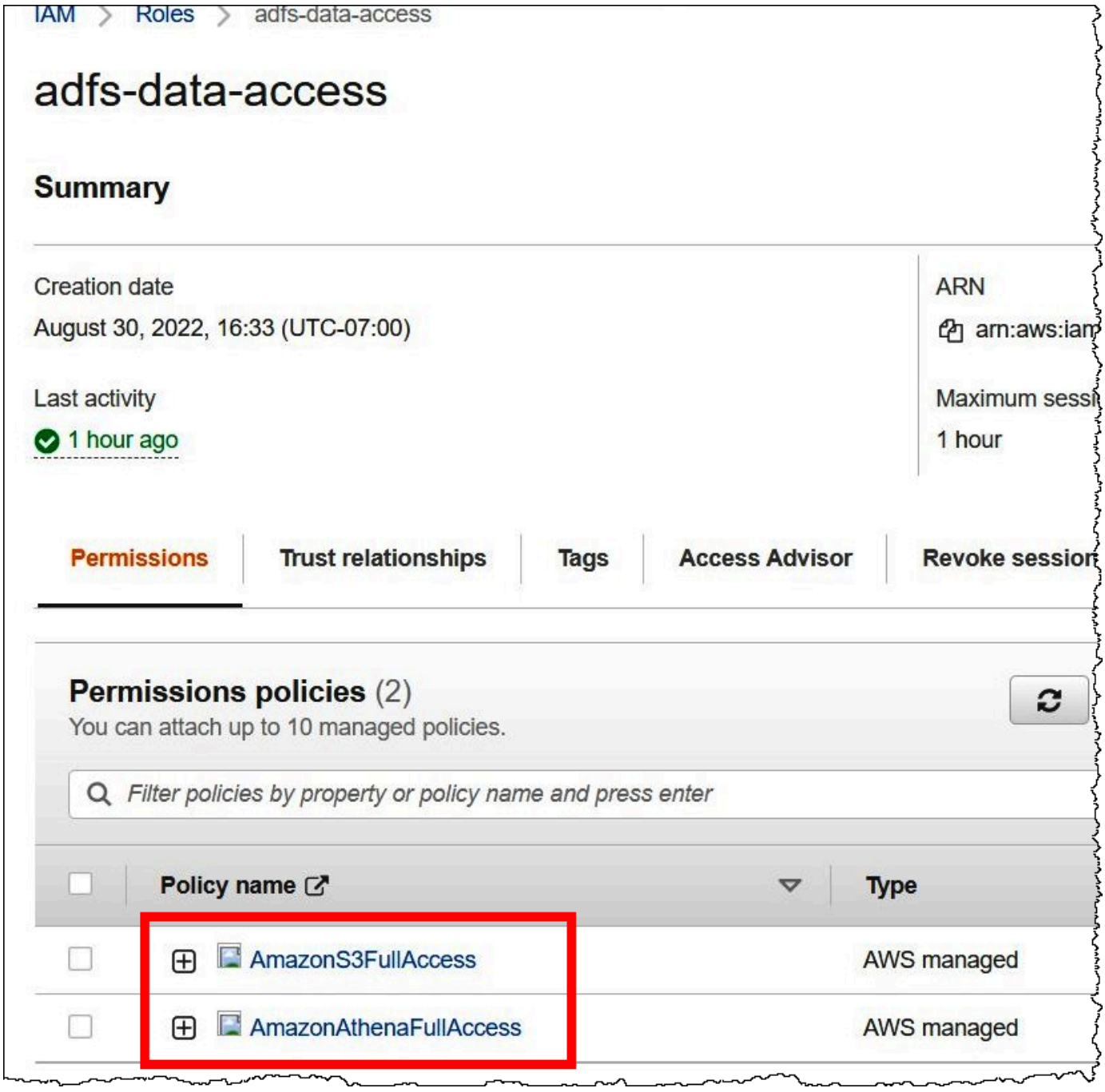


7. Wählen Sie Next (Weiter).
8. Geben Sie auf der Seite Name, review, and create (Benennen, überprüfen und erstellen) für Role name (Rollenname) einen Namen für die Rolle ein. Dieses Tutorial verwendet den Namen adfs-data-access.

   In Step 1: Select trusted entities (In Schritt 1: Vertrauenswürdige Entitäten auswählen) sollte das Feld Principal (Prinzipal) automatisch mit `"Federated:" "arn:aws:iam::account_id:saml-provider/adfs-saml-provider"` ausgefüllt werden. Das `Condition`-Feld sollte `"SAML:aud"` und `"https://signin.aws.amazon.com/saml"` enthalten.

Step 2: Add permissions (Schritt 2: Berechtigungen hinzufügen) zeigt die Richtlinien an, die Sie der Rolle angefügt haben.

9. Wählen Sie Create role (Rolle erstellen) aus. Eine Banner-Meldung bestätigt die Erstellung der Rolle.
10. Wählen Sie auf der Seite Roles (Rollen) den Namen der von Ihnen soeben erstellten Rolle aus. Auf der Übersichtsseite für die Rolle werden die angefügten Richtlinien angezeigt.



2. Konfigurieren von AD FS

Jetzt können Sie AWS als vertrauende Partei hinzufügen und SAML-Antragsregeln schreiben, sodass Sie die richtigen Benutzerattribute zur Autorisierung an AWS senden können.

Der SAML-basierte Verbund verfügt über zwei Teilnehmerparteien: den IdP (Active Directory) und die vertrauende Partei (AWS), d. h. den Service oder die Anwendung, die die Authentifizierung durch IdP verwendet.

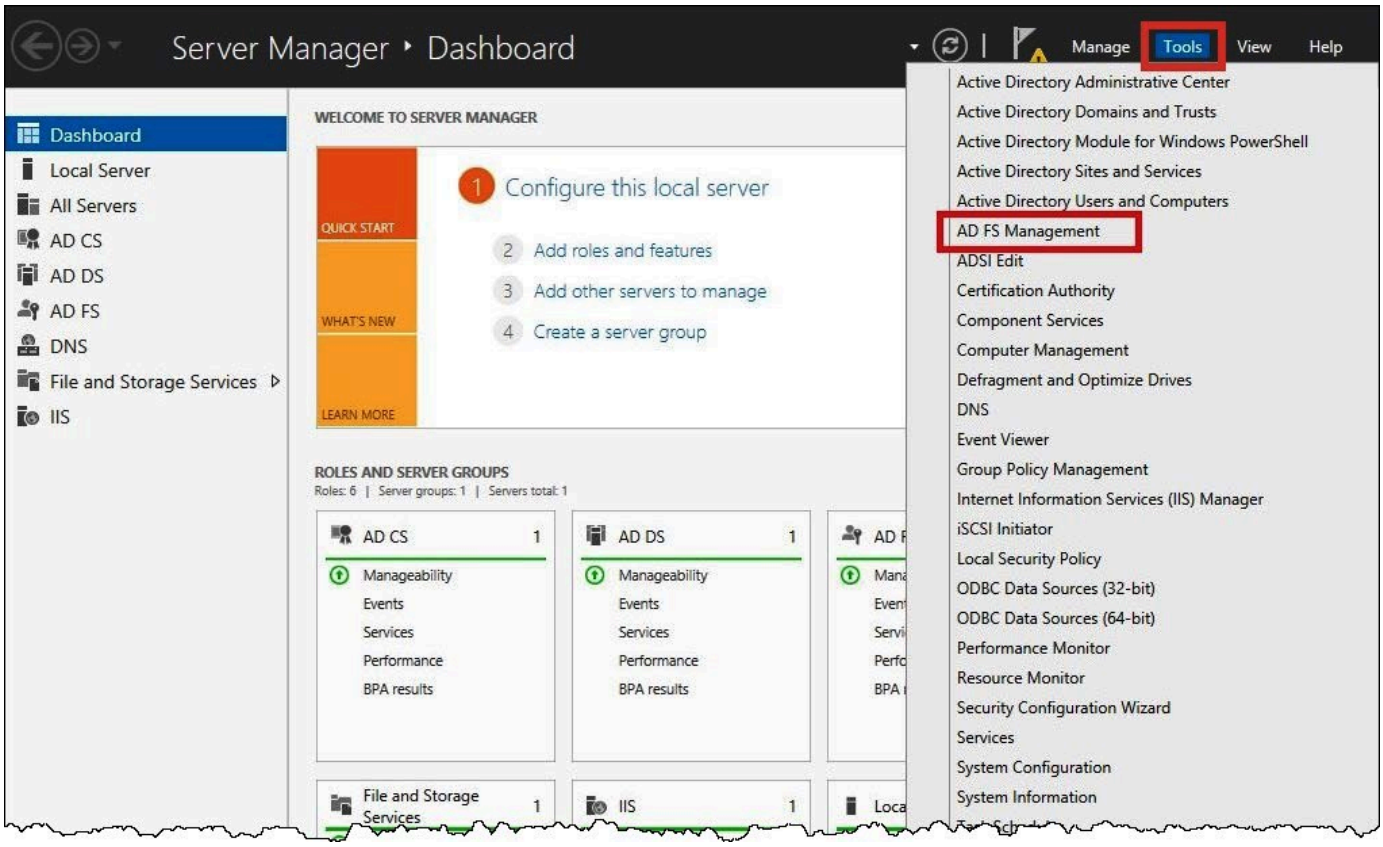
Um AD FS zu konfigurieren, fügen Sie zuerst eine Vertrauensstellung der vertrauenden Partei hinzu und konfigurieren dann SAML-Antragsregeln für die vertrauende Partei. AD FS verwendet Antragsregeln, um eine SAML-Aussage zu bilden, die an eine vertrauende Partei gesendet wird. Die SAML-Aussage besagt, dass die Informationen über den AD-Benutzer wahr sind und dass der Benutzer authentifiziert wurde.

Hinzufügen einer Vertrauensstellung der vertrauenden Partei

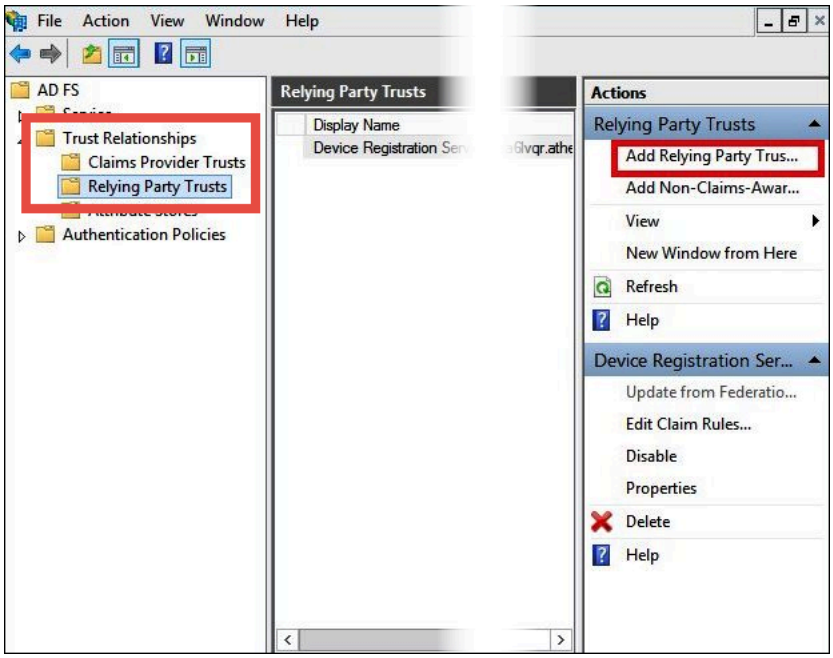
Um eine Vertrauensstellung der vertrauenden Partei in AD FS hinzuzufügen, verwenden Sie den AD-FS-Server-Manager.

So fügen Sie eine Vertrauensstellung der vertrauenden Partei in AD FS hinzu

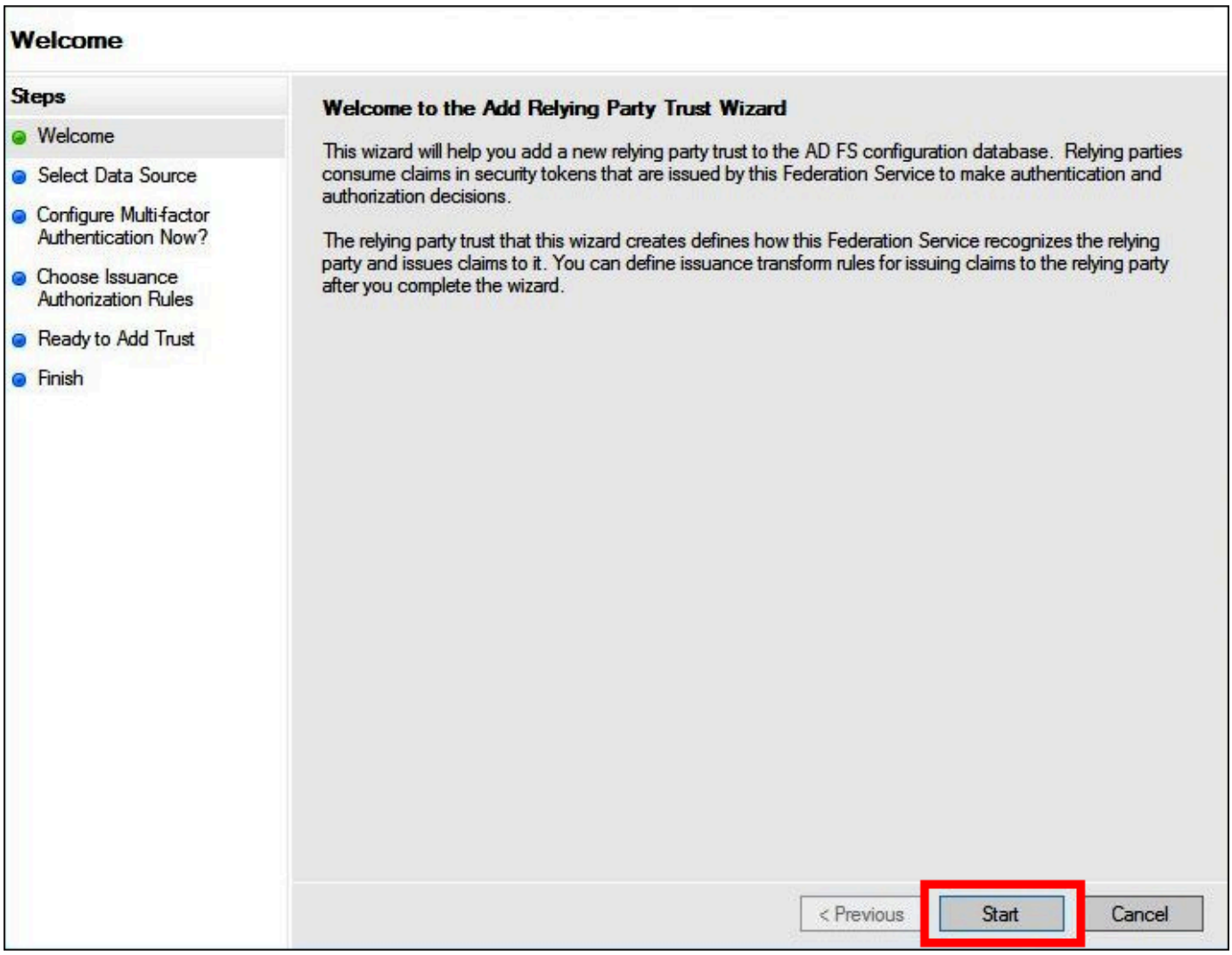
1. Melden Sie sich beim AD-FS-Server an.
2. Öffnen Sie im Start-Menü den Server Manager (Server-Manager).
3. Wählen Sie Tools und anschließend AD FS Management (AD-FS-Verwaltung) aus.



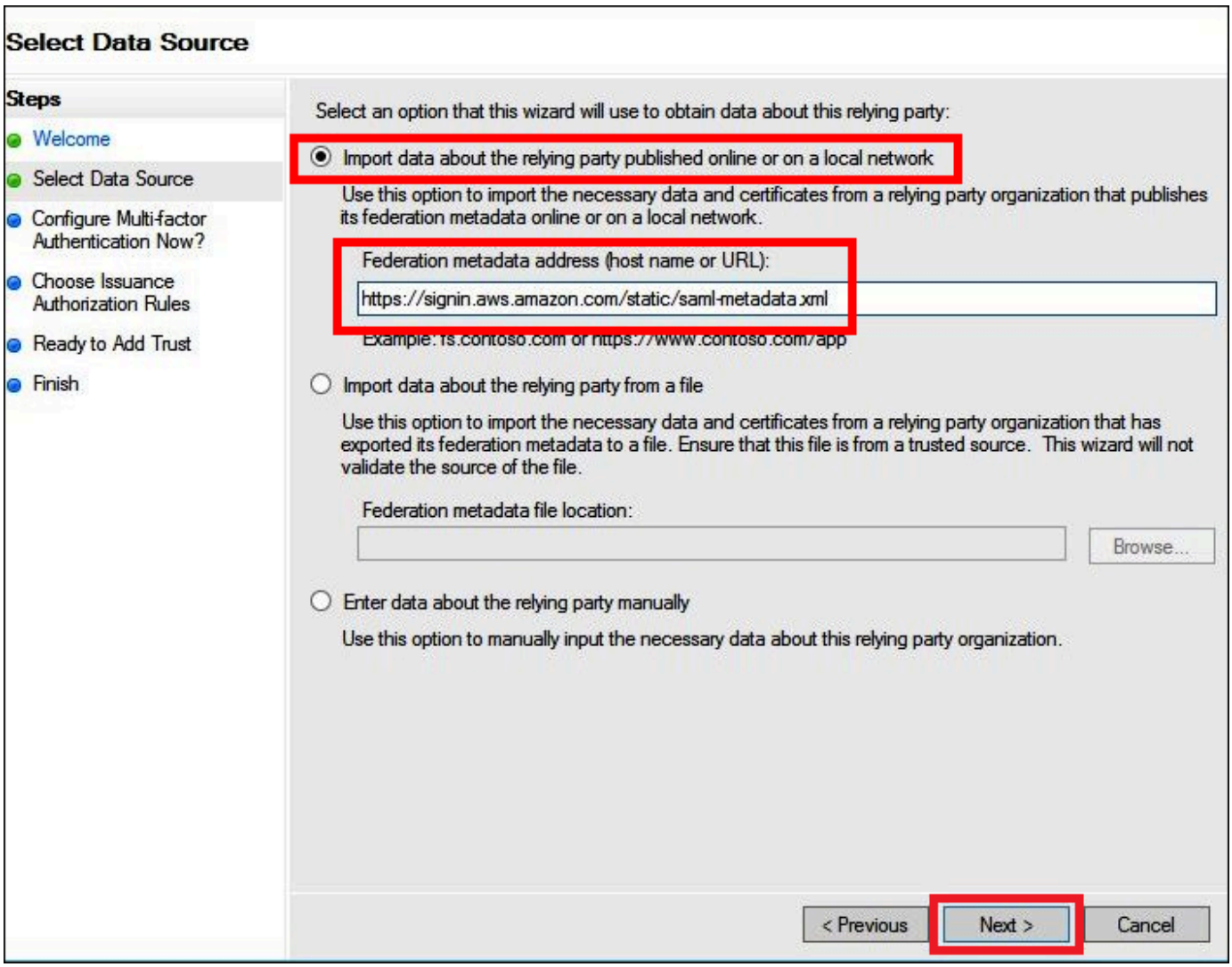
4. Wählen Sie im Navigationsbereich unter Trust Relationships (Vertrauensstellungen) die Option Relying Party Trusts (Vertrauensstellungen der vertrauenden Partei) aus.
5. Wählen Sie unter Aktionen die Option Add Relying Party Trust (Vertrauen für die vertrauende Partei hinzufügen) aus.



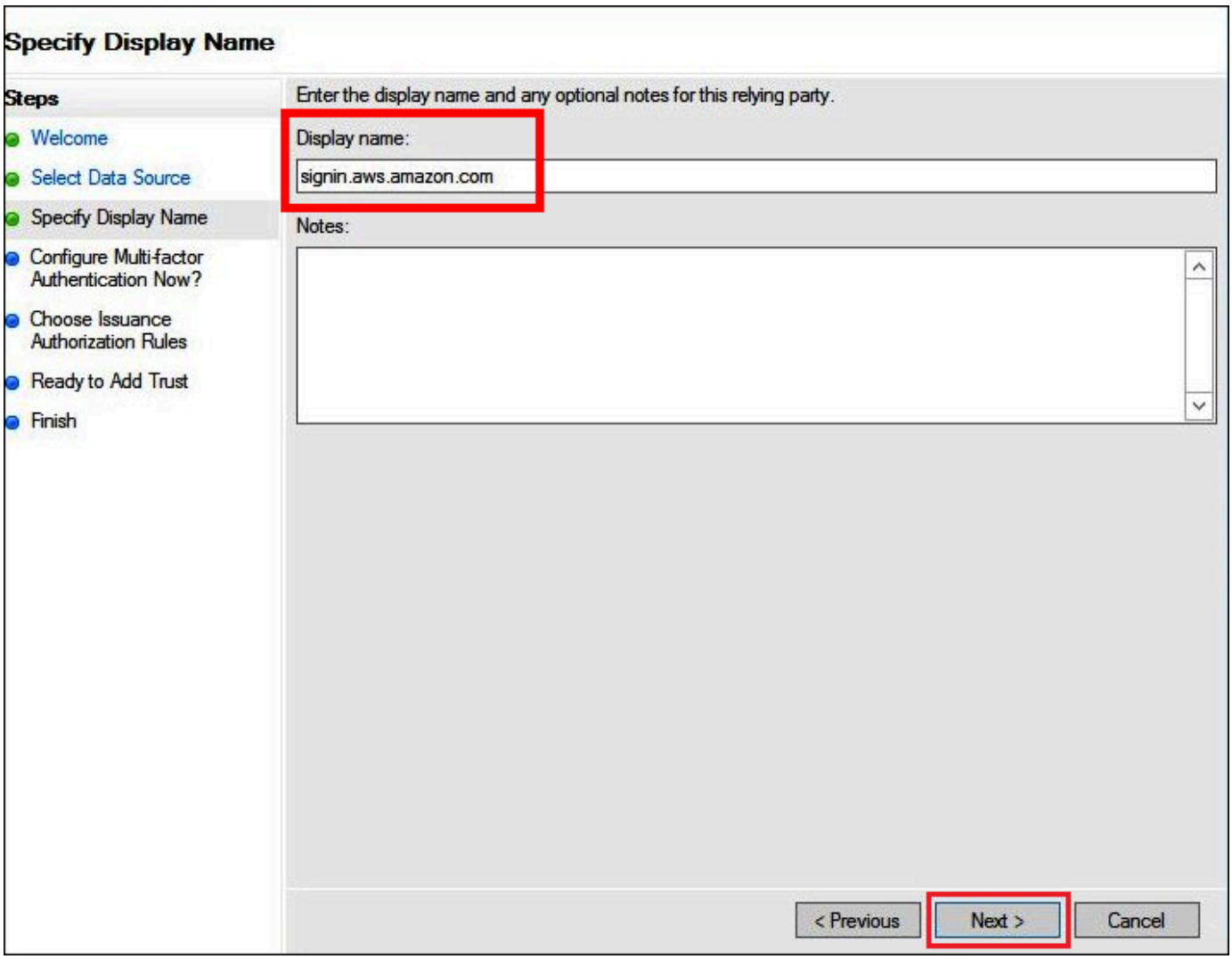
6. Wählen Sie auf der Seite Add Relying Party Trust Wizard (Assistent zum Hinzufügen vertrauender Parteien) die Option Start.



7. Wählen Sie auf dem Bildschirm Select Data Source (Datenquelle auswählen) die Option Import data about the relying party published online or on a local network (Daten über die vertrauende Partei importieren, die online oder in einem lokalen Netzwerk veröffentlicht wurden) aus.
8. Geben Sie für Federation metadata address (host name or URL) (Verbund-Metadatenadresse (Hostname oder URL)) die URL **`https://signin.aws.amazon.com/static/saml-metadata.xml`** ein
9. Wählen Sie Next (Weiter).

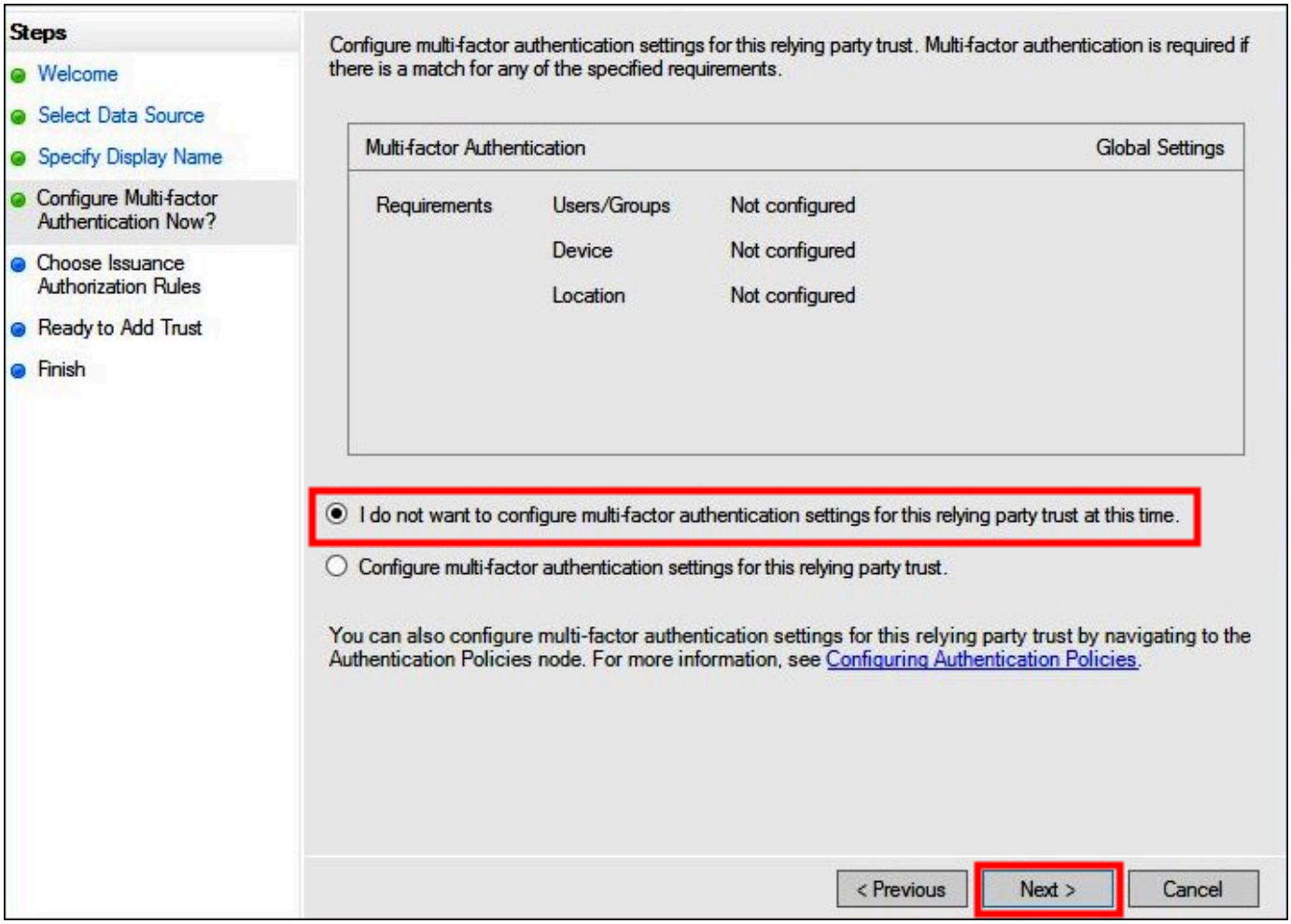


10. Geben Sie auf der Seite Specify Display Name (Anzeigenamen angeben) für Display name (Anzeigename) einen Anzeigenamen für Ihre vertrauende Partei ein, und wählen Sie dann Next (Weiter) aus.



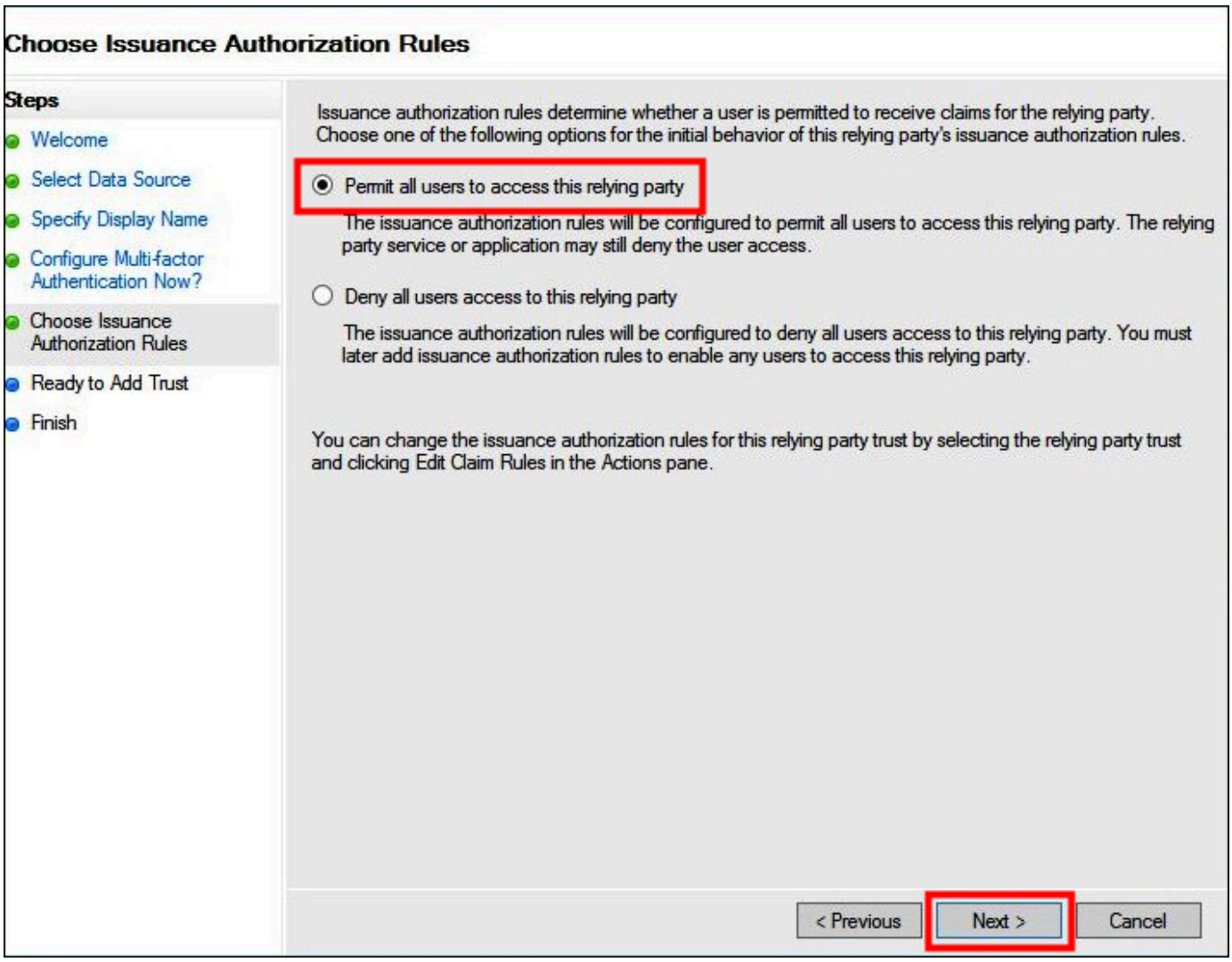
11. Auf der Seite Configure Multi-factor Authentication Now (Mehrstufige Authentifizierung jetzt konfigurieren) wählt dieses Tutorial I do not want to configure multi-factor authentication for this relying party trust at this time (Ich möchte die mehrstufige Authentifizierung für diese Vertrauensstellung der vertrauenden Partei zu diesem Zeitpunkt nicht konfigurieren) aus.

    Zur Verbesserung der Sicherheit und zum Schutz Ihrer AWS-Ressourcen empfehlen wir, dass Sie die Multifaktor-Authentifizierung konfigurieren. Da es nur einen Beispieldatensatz verwendet, aktiviert dieses Tutorial keine Multi-Faktor-Authentifizierung.

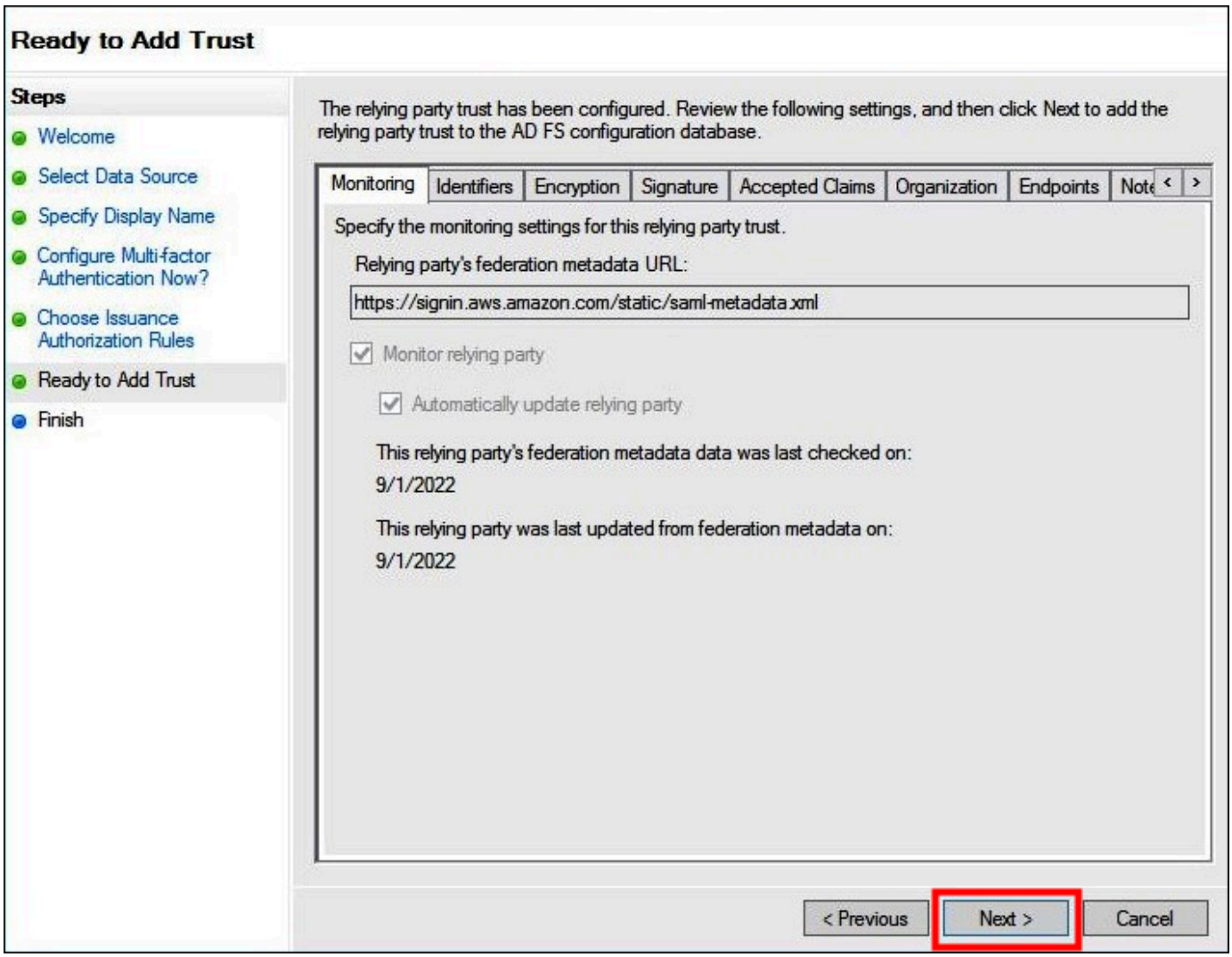


12. Wählen Sie Next (Weiter).
13. Wählen Sie auf der Seite Choose Issuance Authorization Rules (Ausgabeberechtigungsregeln auswählen) die Option Permit all users to access this relying party (Allen Benutzern den Zugriff auf diese vertrauende Partei gewähren) aus.

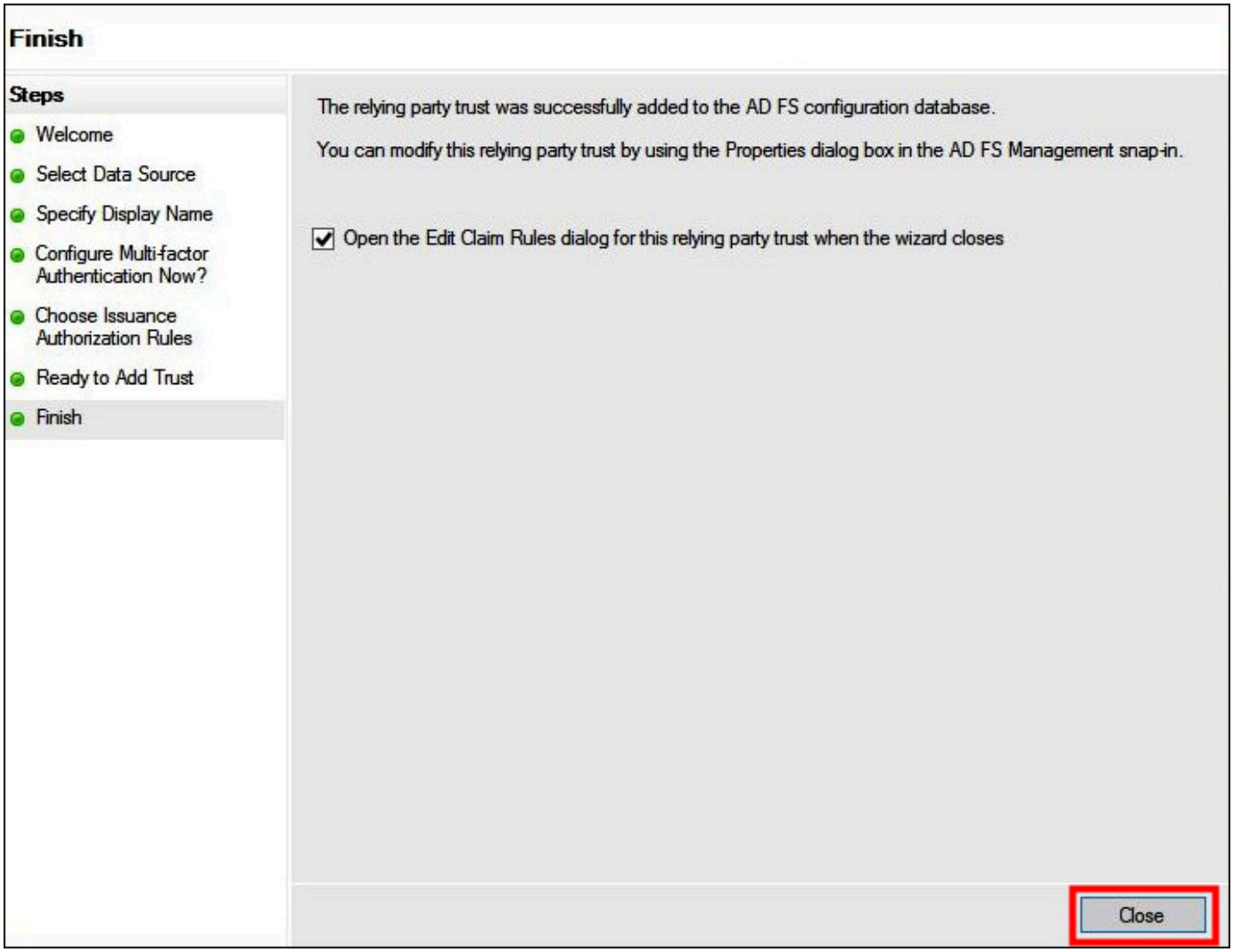
    Mit dieser Option können alle Benutzer in Active Directory AD FS mit AWS als vertrauende Partei verwenden. Sie sollten Ihre Sicherheitsanforderungen berücksichtigen und diese Konfiguration entsprechend anpassen.



14. Wählen Sie Next (Weiter).
15. Wählen Sie auf der Seite Ready to Add Trust (Bereit zum Hinzufügen der Vertrauensstellung) Next (Weiter) aus, um die Vertrauensstellung der vertrauenden Partei zur AD-FS-Konfigurationsdatenbank hinzuzufügen.



16. Wählen Sie auf der Seite Finish (Fertigstellen) die Option Close (Schließen) aus.



Konfigurieren von SAML-Antragsregeln für die vertrauende Partei

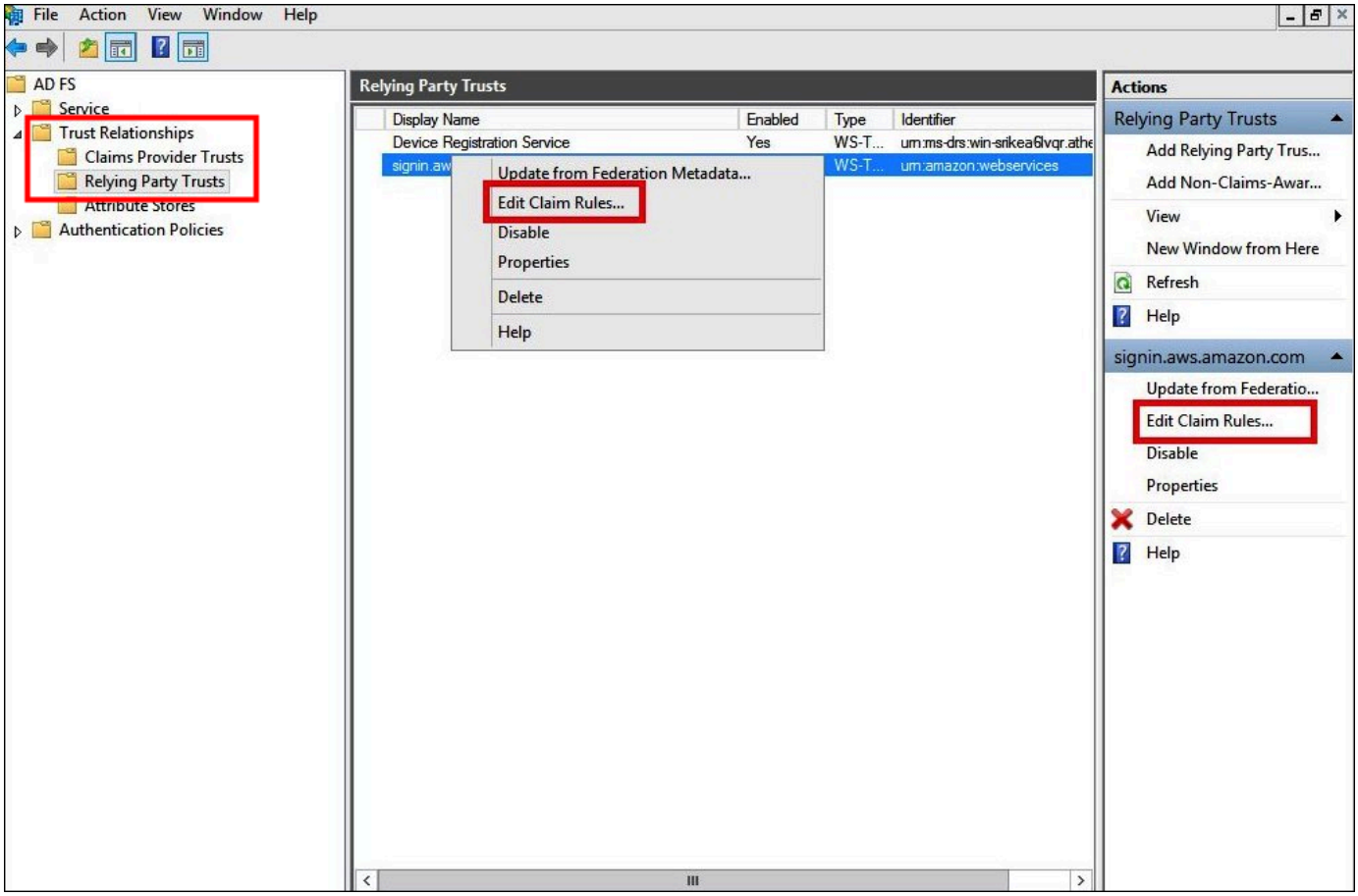
In dieser Aufgabe erstellen Sie zwei Sätze von Antragsregeln.

Der erste Satz, die Regeln 1-4, enthält AD-FS-Antragsregeln, die erforderlich sind, um eine IAM-Rolle basierend auf der AD-Gruppenmitgliedschaft anzunehmen. Dies sind die gleichen Regeln, die Sie erstellen, wenn Sie Verbundzugriff auf AWS Management Console einrichten möchten.

Beim zweiten Satz, den Regeln 5-6, handelt es sich um Beanspruchungsregeln, die für die Athena-Zugriffskontrolle erforderlich sind.

So erstellen Sie AD-FS-Antragsregeln

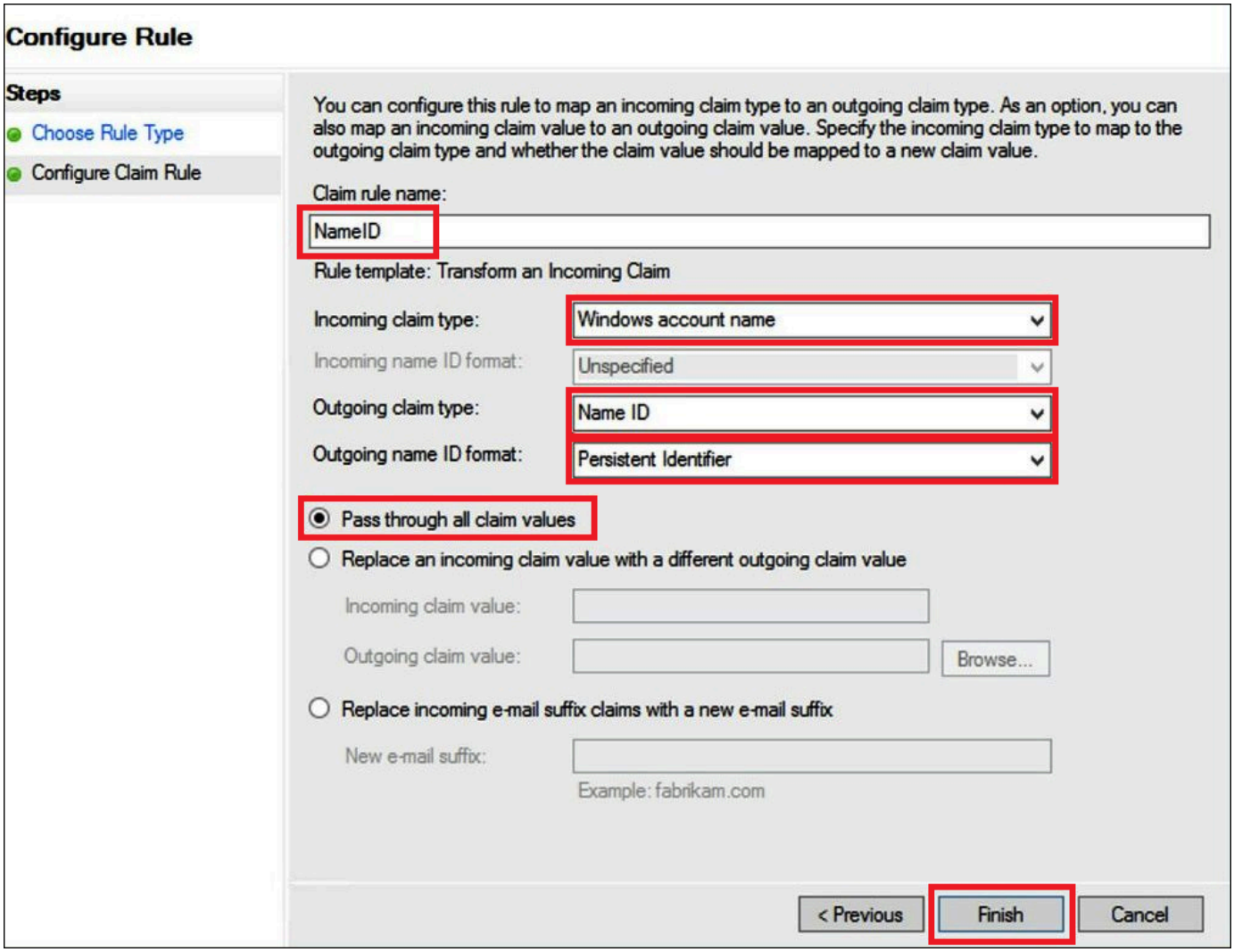
1. Wählen Sie im Navigationsbereich der AD-FS-Verwaltungskonsole Trust Relationships (Vertrauensbeziehungen), Relying Party Trusts (Vertrauensstellungen der vertrauenden Partei) aus.
2. Suchen Sie die vertrauende Partei, die Sie im vorherigen Abschnitt erstellt haben.
3. Klicken Sie mit der rechten Maustaste auf die vertrauende Partei und wählen Sie Edit Claim Rules (Antragsregeln bearbeiten) oder wählen Sie die Option Edit Claim Rules (Antragsregeln bearbeiten) im Menü Actions (Aktionen) aus.



4. Klicken Sie auf Add Rule (Regel hinzufügen).
5. Geben Sie auf der Seite Configure Rule (Regel konfigurieren) des Assistenten zum Hinzufügen von Transformations-Antragsregeln die folgenden Informationen ein, um Antragsregeln 1 zu erstellen, und wählen Sie anschließend Finish (Fertig stellen) aus.

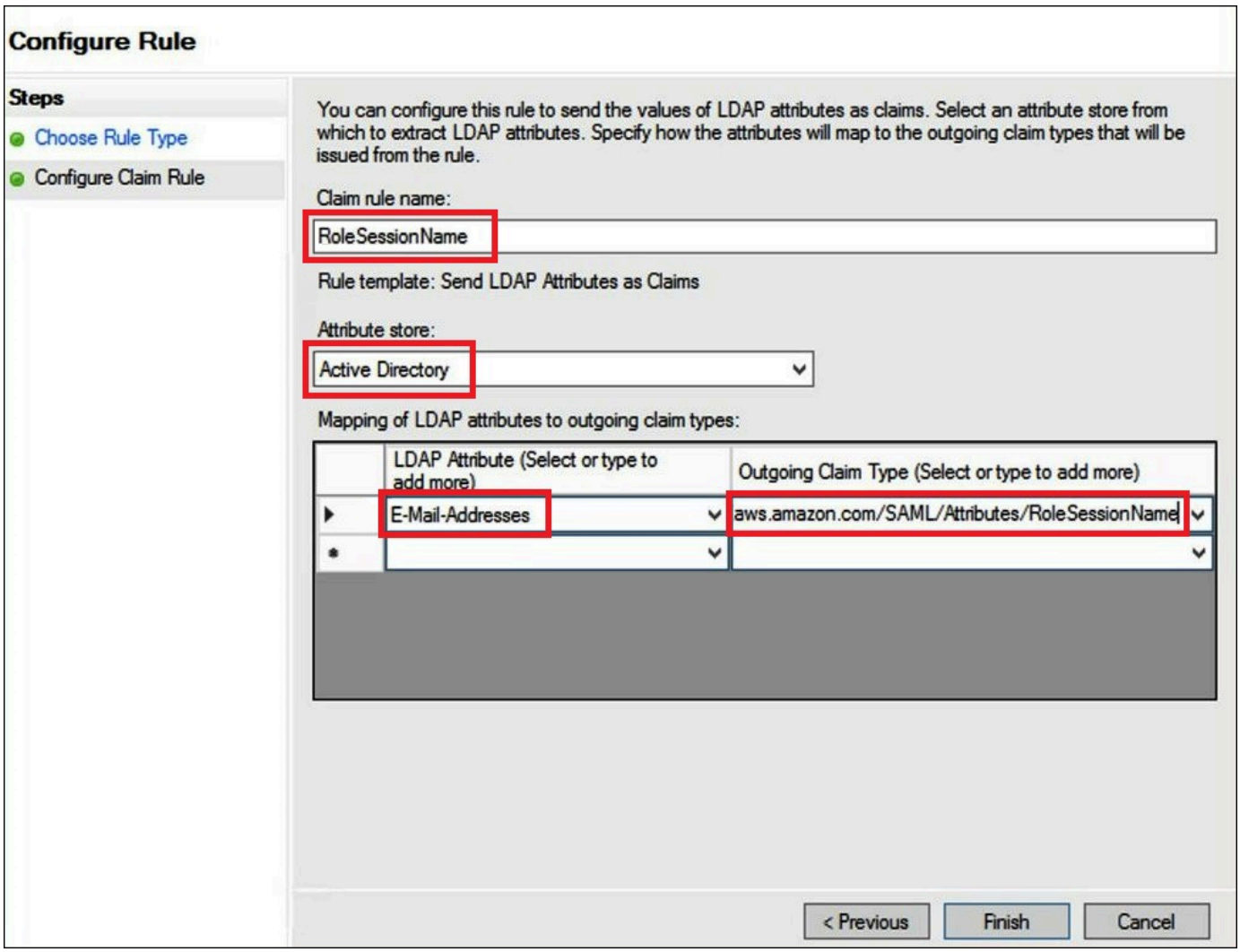
   - Geben Sie unter Claim rule name (Name der Antragsregel) **`NameID`** ein.
   - Verwenden Sie für Rule template (Regelvorlage) die Option Transform an Incoming Claim (Einen eingehenden Antrag transformieren).

- Wählen Sie unter Incoming claim type (Typ des eingehenden Antrags) die Option Windows Account Name (Windows-Kontoname).
- Wählen Sie unter Outgoing claim type (Typ des ausgehenden Antrags) die Option Name ID (Namens-ID) aus.
- Wählen Sie unter Outgoing name ID format (Format der ausgehenden Namens-ID) die Option Persistent identifier (Persistente ID).
- Wählen Sie Pass through all claim values (Alle Antragswerte weiterleiten) aus.



6. Wählen Sie Add Rule (Regel hinzufügen) aus und geben Sie dann die folgenden Informationen ein, um Antragsregel 2 zu erstellen. Wählen Sie anschließend Finish (Fertig stellen) aus.
   - Geben Sie unter Claim rule name (Name der Antragsregel) **`RoleSessionName`** ein.

- Verwenden Sie für Rule template (Regelvorlage) die Option Send LDAP Attribute as Claims (LDAP-Attribut als Anträge senden) aus.
- Wählen Sie unter Attribute store (Attributspeicher) Active Directory.
- Fügen Sie für Mapping of LDAP attributes to outgoing claim types (Zuordnung von LDAP-Attributen zu ausgehenden Antragstypen) das Attribut **`E-Mail-Addresses`** hinzu. Geben Sie für Outgoing Claim Type (Art des ausgehenden Antrags) **`https://aws.amazon.com/SAML/Attributes/RoleSessionName`** ein.

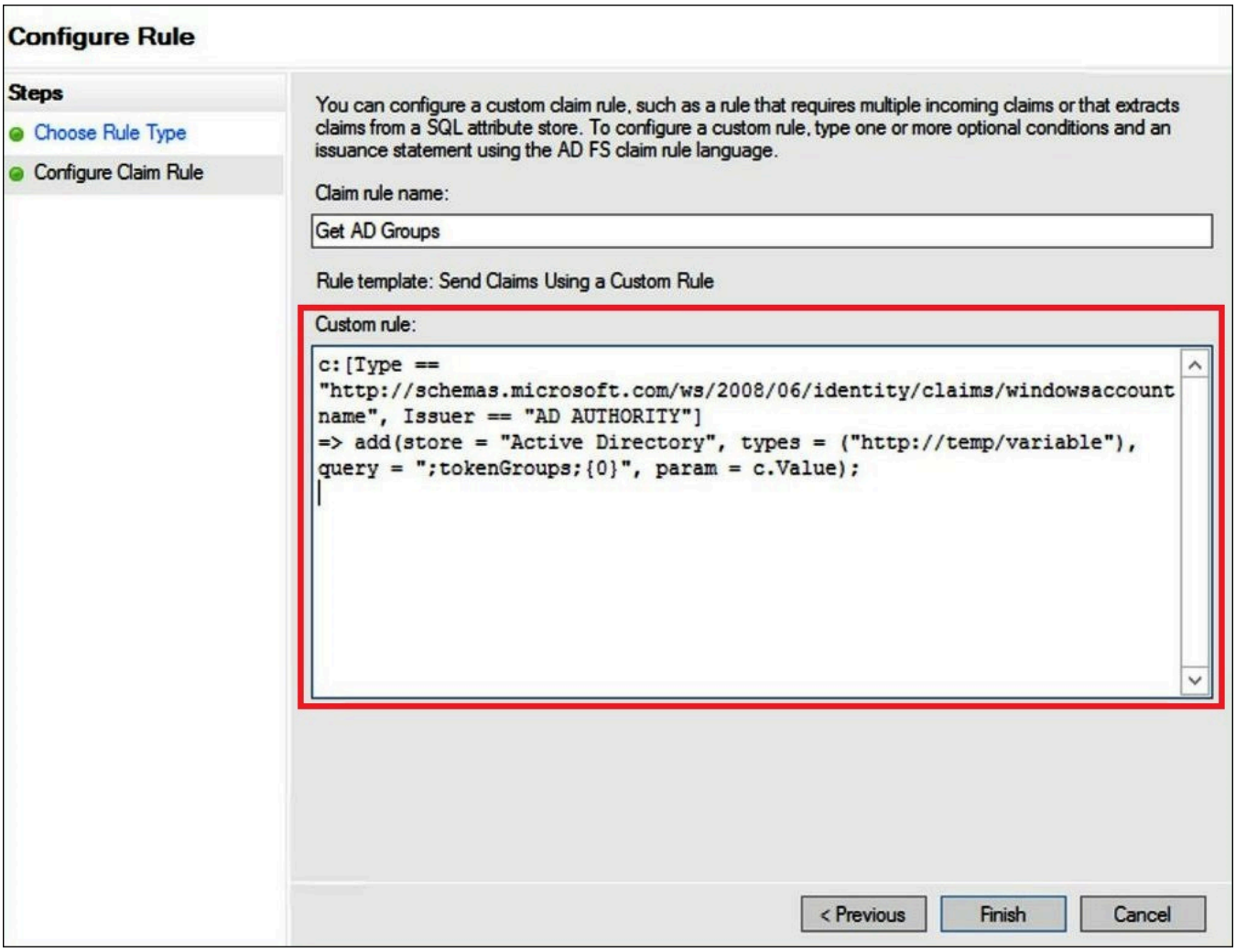


7. Wählen Sie Add Rule (Regel hinzufügen) aus und geben Sie dann die folgenden Informationen ein, um Antragsregel 3 zu erstellen. Wählen Sie dann Finish (Fertig stellen) aus.

   - Geben Sie unter Claim rule name (Name der Antragsregel) **`Get AD Groups`** ein.

- Verwenden Sie für die Regelvorlage die Option Send Claims Using a Custom Rule (Anträge mit einer benutzerdefinierten Regel senden).
- Geben Sie unter Custom rule (Benutzerdefinierte Regel) den folgenden Code ein:

```
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);
```

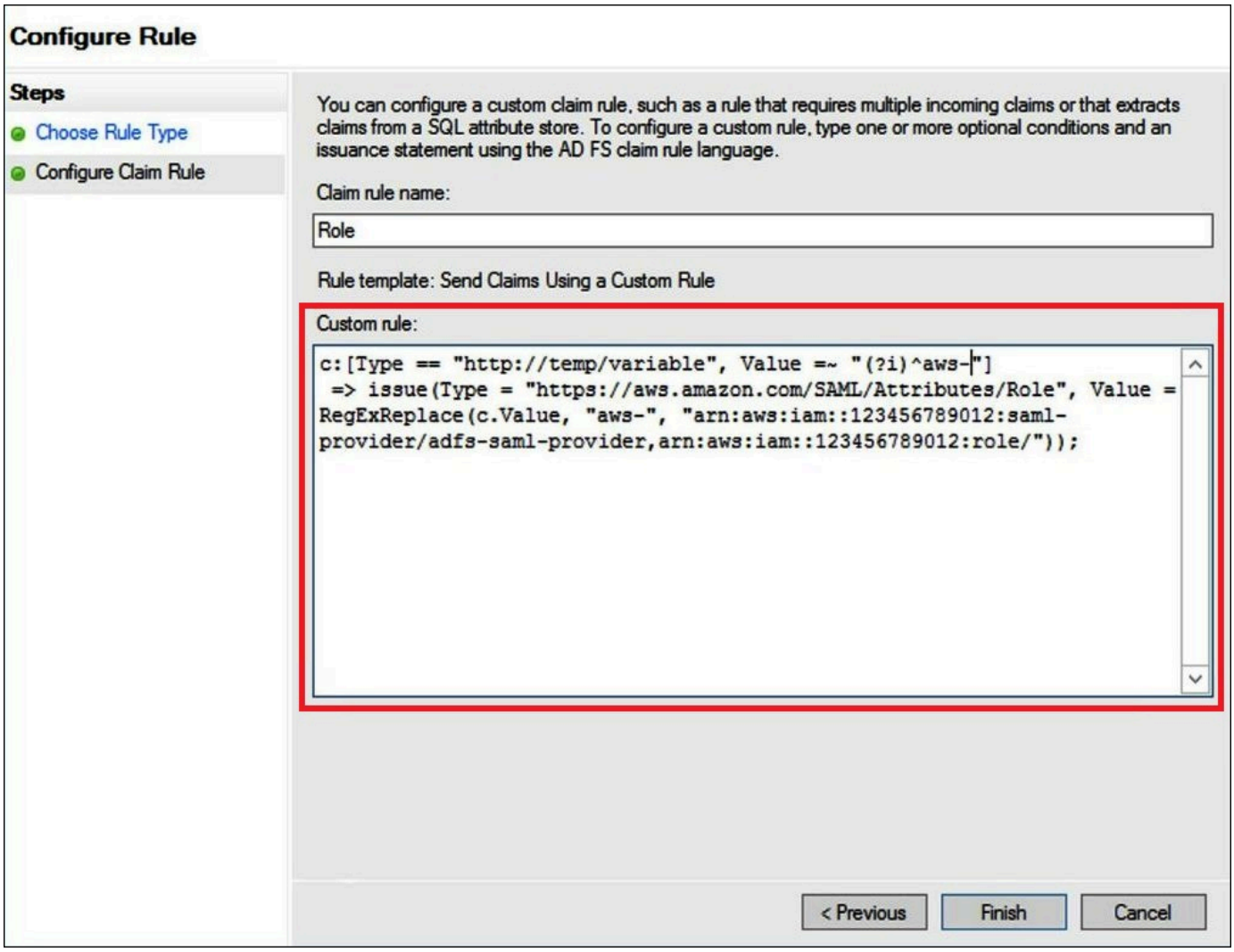


8. Klicken Sie auf Add Rule (Regel hinzufügen). Geben Sie die folgenden Informationen ein, um Antragsregel 4 zu erstellen und wählen Sie anschließend Finish (Fertig stellen) aus.

   - Geben Sie unter Claim rule name (Name der Antragsregel) **`Role`** ein.

- Verwenden Sie für Rule template (Regelvorlage) die Option Send LDAP Attribute as Claims (LDAP-Attribut als Anträge senden) aus.
- Geben Sie für Custom Rule (Benutzerdefinierte Regel) den folgenden Code mit Ihrer Kontonummer und dem Namen des zuvor erstellten SAML-Anbieters ein:

```
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/"));
```

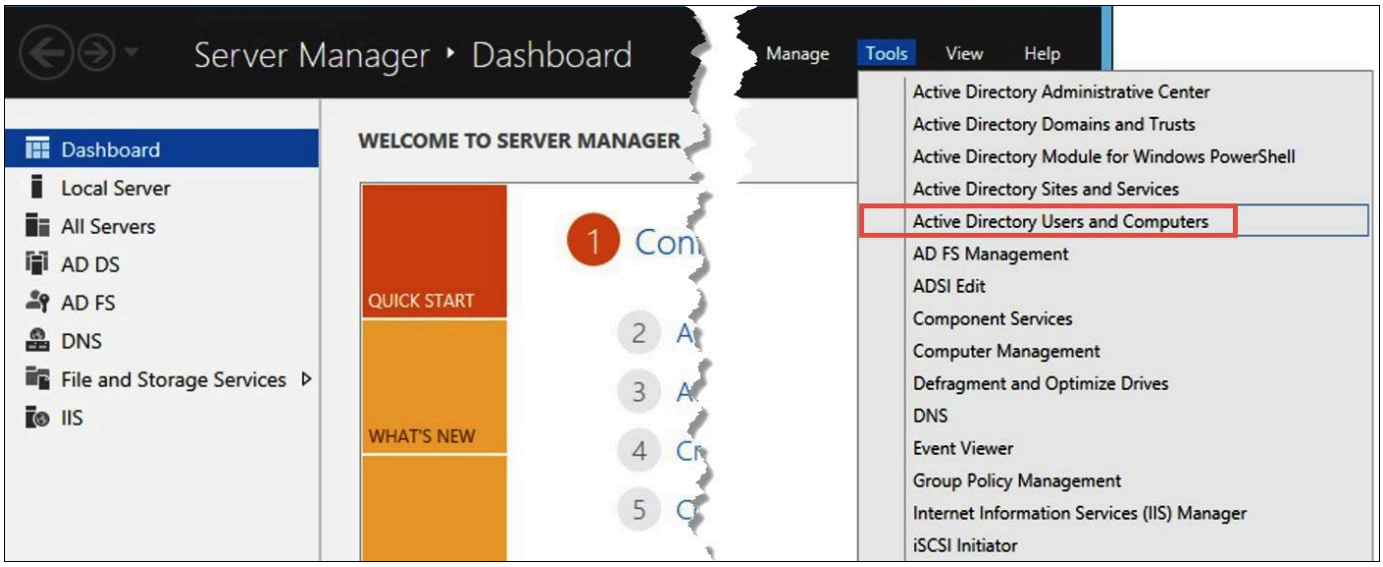


3. Erstellen von Active-Directory-Benutzern und -Gruppen

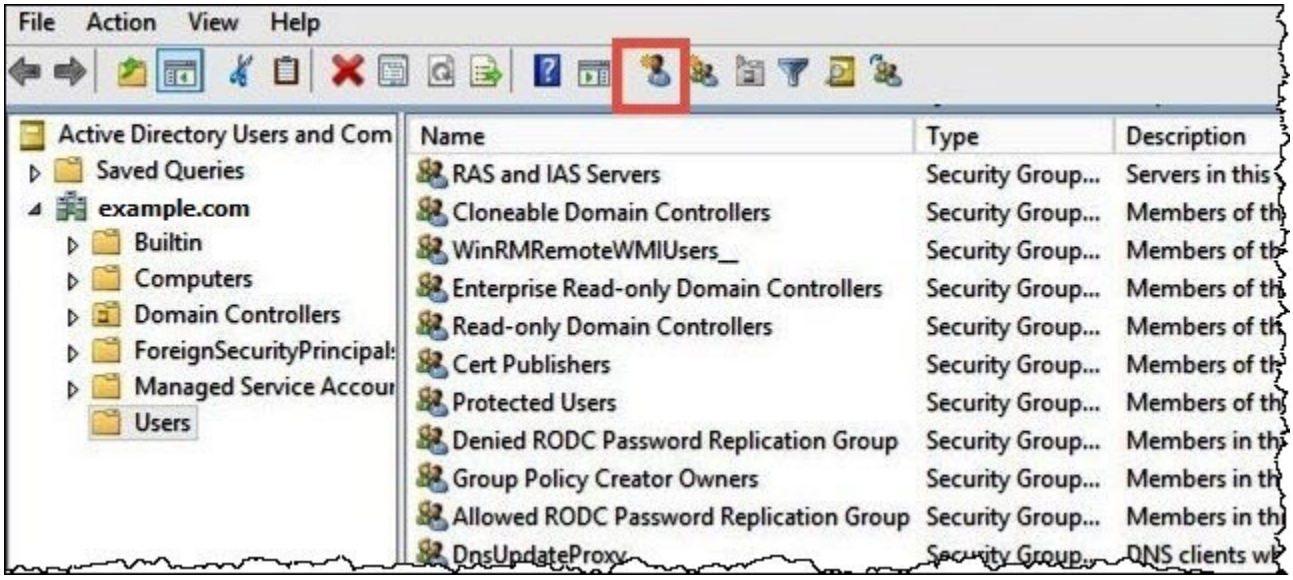
Jetzt können Sie AD-Benutzer erstellen, die auf Athena zugreifen, und AD-Gruppen erstellen, in denen sie platziert werden, sodass Sie die Zugriffsebenen nach Gruppen steuern können. Nachdem Sie AD-Gruppen erstellt haben, die Datenzugriffsmuster kategorisieren, fügen Sie Ihre Benutzer zu diesen Gruppen hinzu.

So erstellen Sie AD-Benutzer für den Zugriff auf Athena

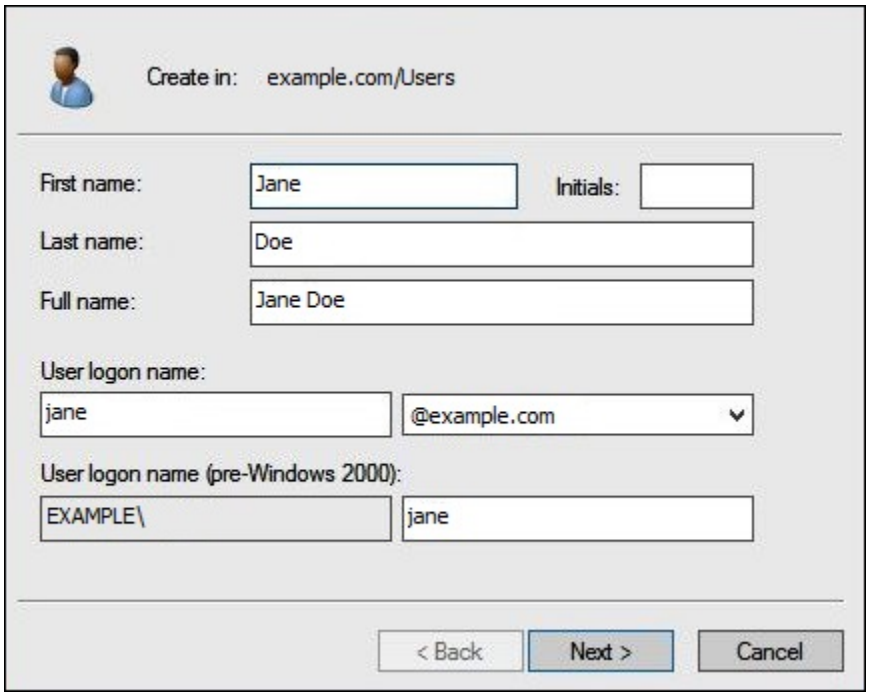
1. Wählen Sie im Dashboard des Server-Managers Tools und dann Active Directory Users and Computers (Active-Directory-Benutzer und -Computer) aus.



2. Klicken Sie im Navigationsbereich auf Users (Benutzer).
3. Wählen Sie in der Symbolleiste Active Directory Users and Computers (Active-Directory-Benutzer und -Computer) die Option Create user (Benutzer erstellen) aus.

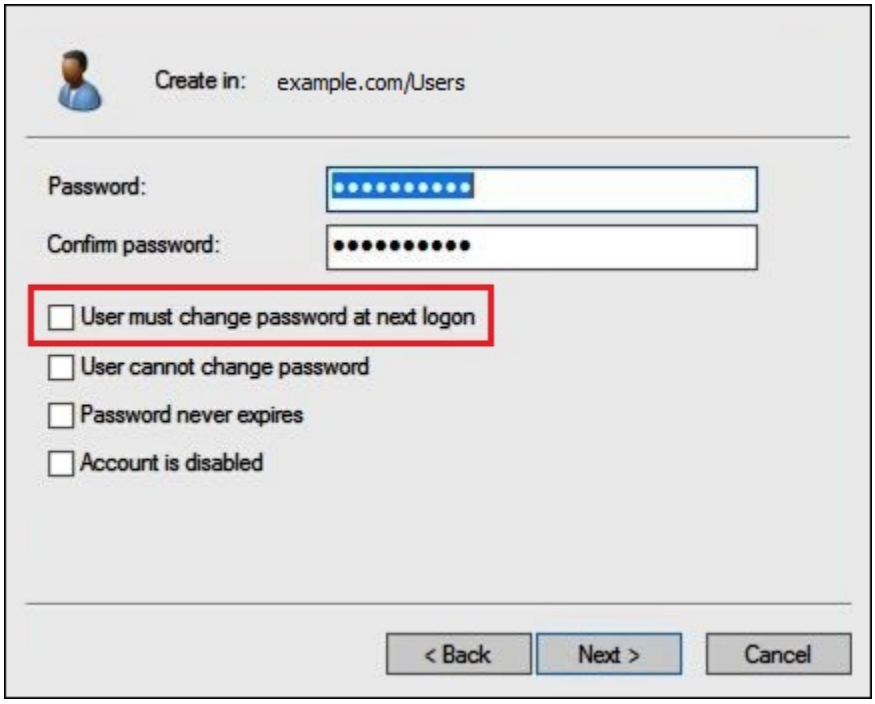


4. Geben Sie im Dialogfeld New Object – User (Neues Objekt – Benutzer)unter First name (Vorname), Last name (Nachname) und Full name (Vollständiger Name) einen Namen ein. In diesem Tutorial wird ein **`Jane Doe`** verwendet.

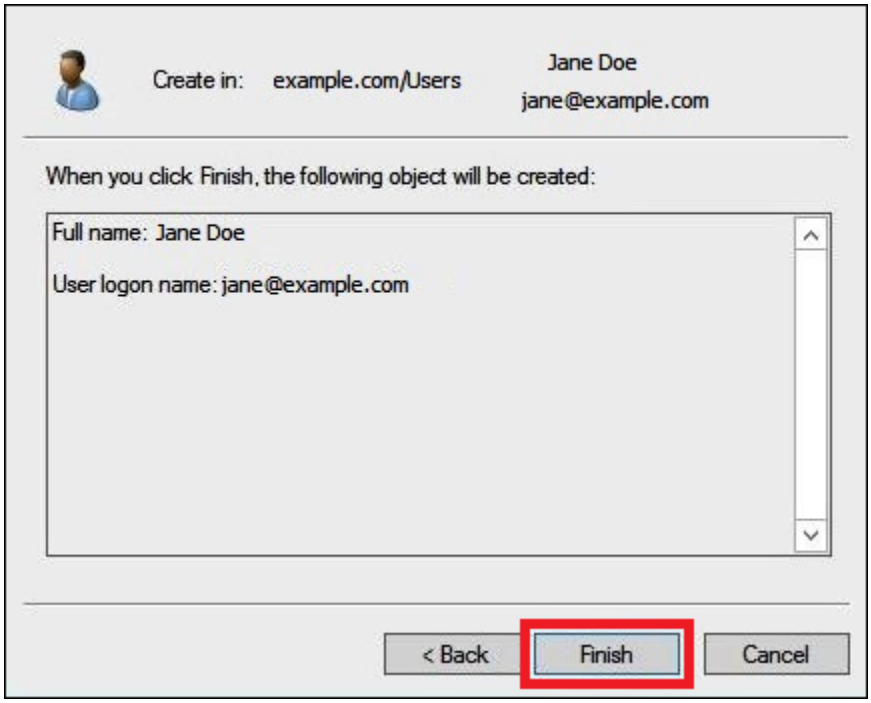


5. Wählen Sie Next (Weiter).
6. Geben Sie unter Password (Passwort) ein Passwort ein; geben Sie es zur Bestätigung erneut ein.

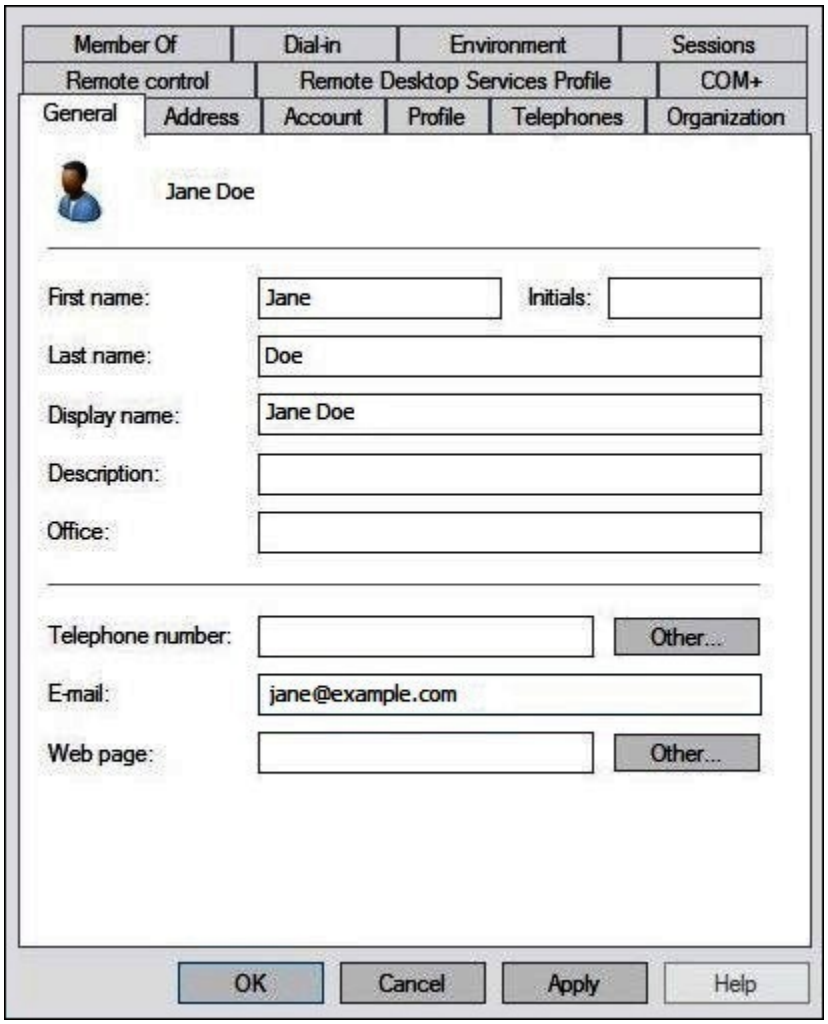
Der Einfachheit halber wird in diesem Tutorial die Option User must change password at next sign on (Benutzer muss das Passwort bei der nächsten Anmeldung ändern) deaktiviert. In realen Szenarien sollten Sie von neu erstellten Benutzern verlangen, dass sie ihr Passwort ändern.



7. Wählen Sie Next (Weiter).
8. Wählen Sie Finish (Abschließen).



9. Wählen Sie unter Active Directory Users and Computers (Active-Directory-Benutzer und -Computer) den Benutzernamen aus.
10. Geben Sie im Dialogfeld Properties (Eigenschaften) des Benutzers unter E-Mail eine E-Mail-Adresse ein. In diesem Tutorial wird ein **`jane@example.com`** verwendet.



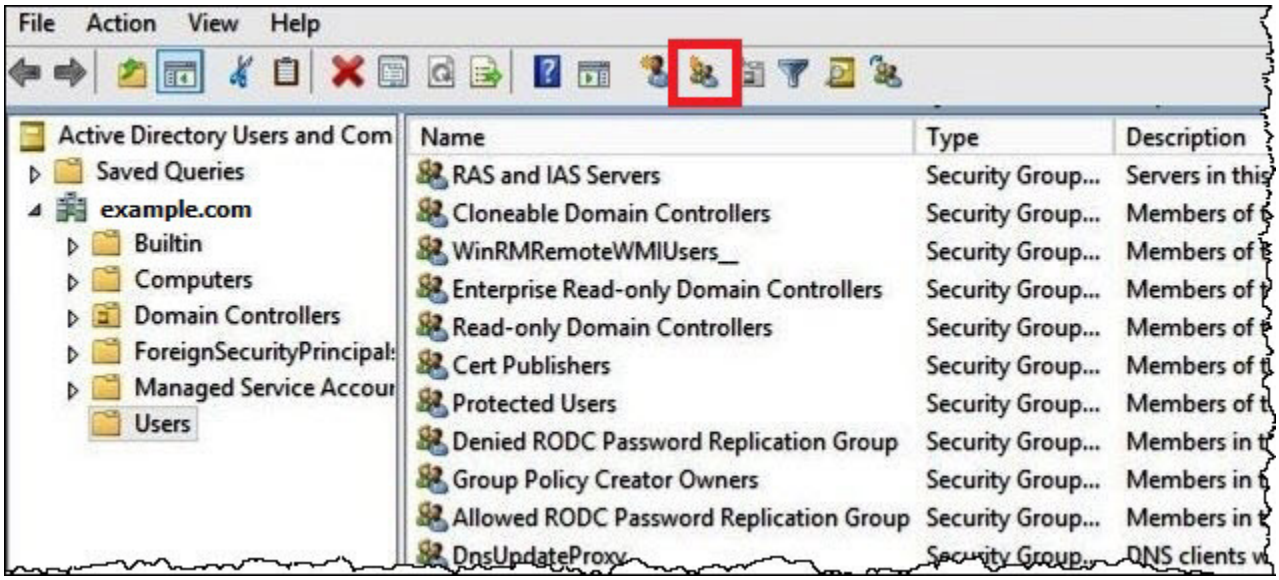
11. Wählen Sie OK.

Erstellen von AD-Gruppen zur Darstellung von Datenzugriffsmustern

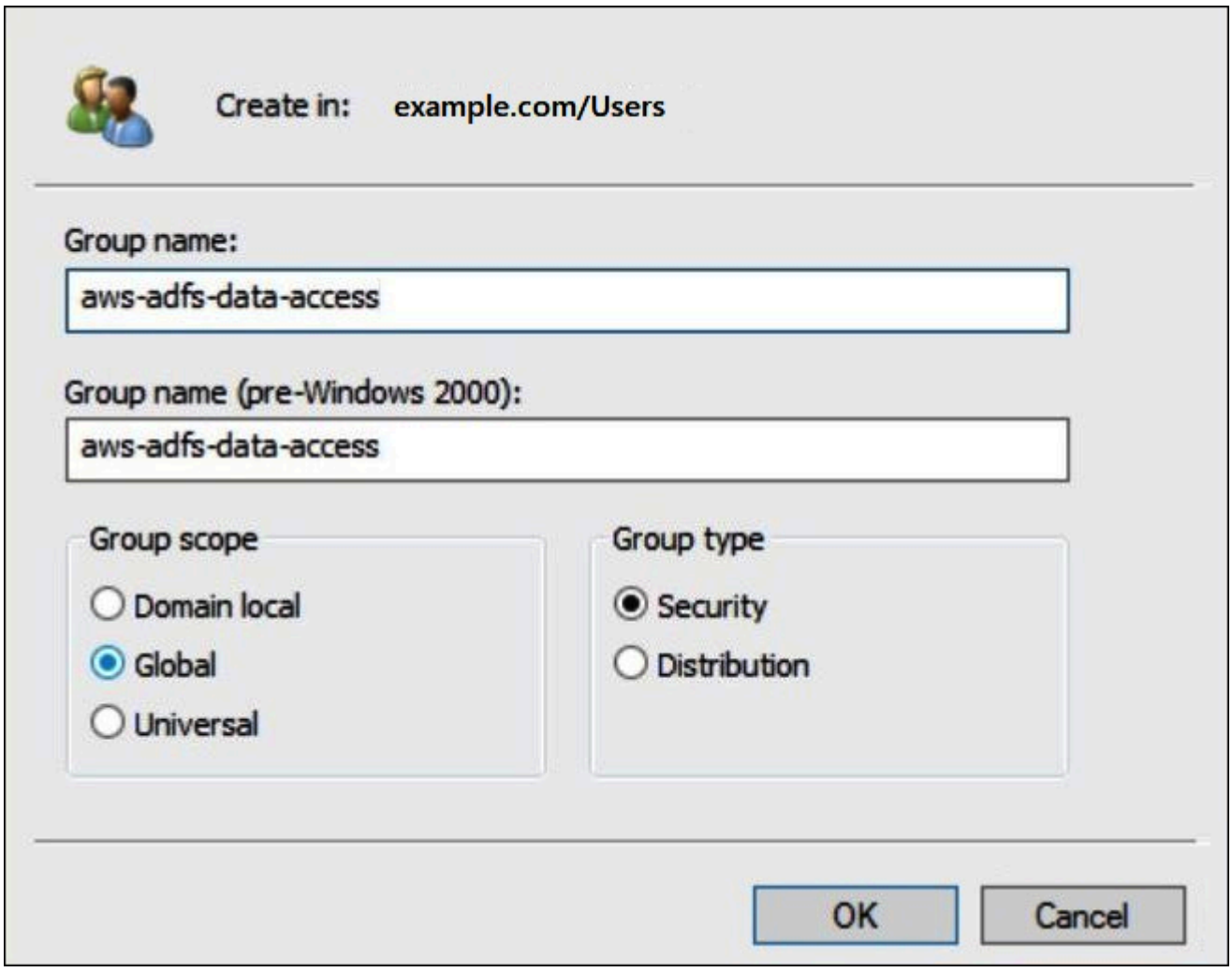
Sie können AD-Gruppen erstellen, deren Mitglieder die `adfs-data-access`-IAM-Rolle übernehmen, wenn sich diese bei AWS anmelden. Im folgenden Beispiel wird eine AD-Gruppe mit dem Namen aws-adfs-data-access erstellt.

So erstellen Sie eine AD-Gruppe

1. Wählen Sie im Server-Manager-Dashboard im Menü Tools die Option Active Directory Users and Computers (Active-Directory-Benutzer und -Computer) aus.
2. Wählen Sie in der Symbolleiste die Option Create new group (Neue Gruppe erstellen) aus.



3. Geben Sie in das Dialogfeld New Object – Group (Neues Objekt – Gruppe) die folgenden Informationen ein:
   - Geben Sie für Group name (Gruppenname) **`aws-adfs-data-access`** ein.
   - Wählen Sie als Group scope (Gruppenbereich) die Option Global aus.
   - Wählen Sie als Group type (Gruppentyp) die Option Security (Sicherheit) aus.



4. Wählen Sie OK.

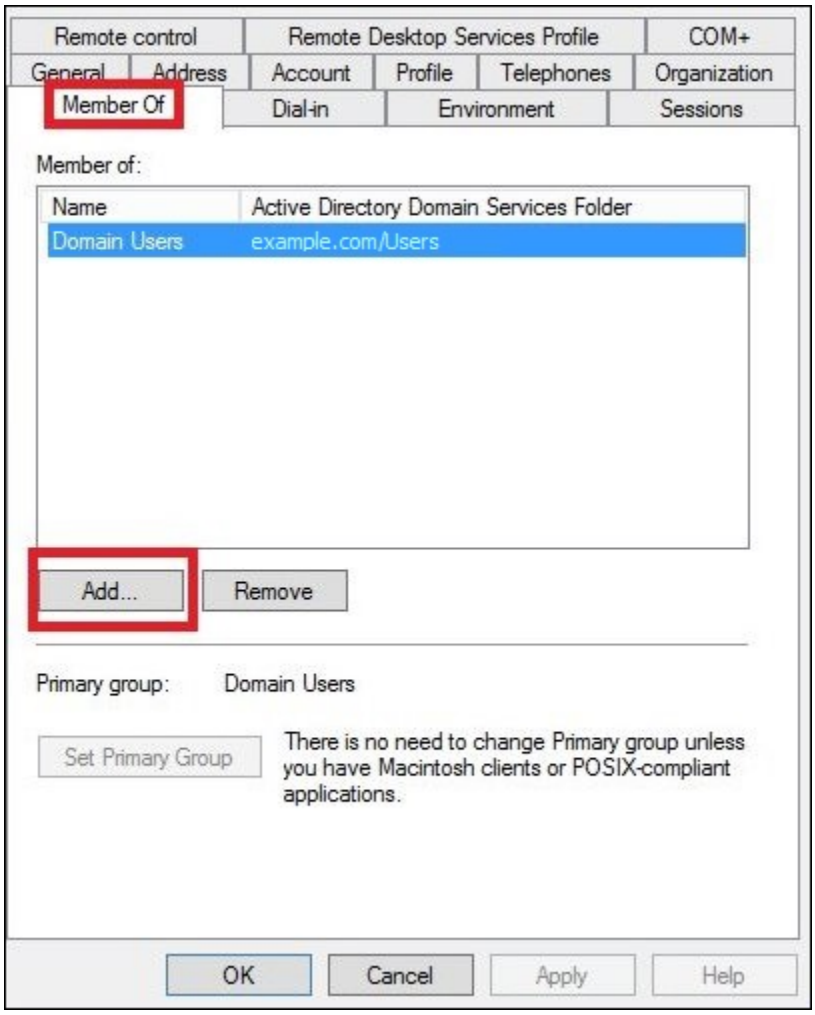
AD-Benutzer zu entsprechenden Gruppen hinzufügen

Nachdem Sie nun sowohl einen AD-Benutzer als auch eine AD-Gruppe erstellt haben, können Sie den Benutzer der Gruppe hinzufügen.

So fügen Sie einen AD-Benutzer zu einer AD-Gruppe hinzu

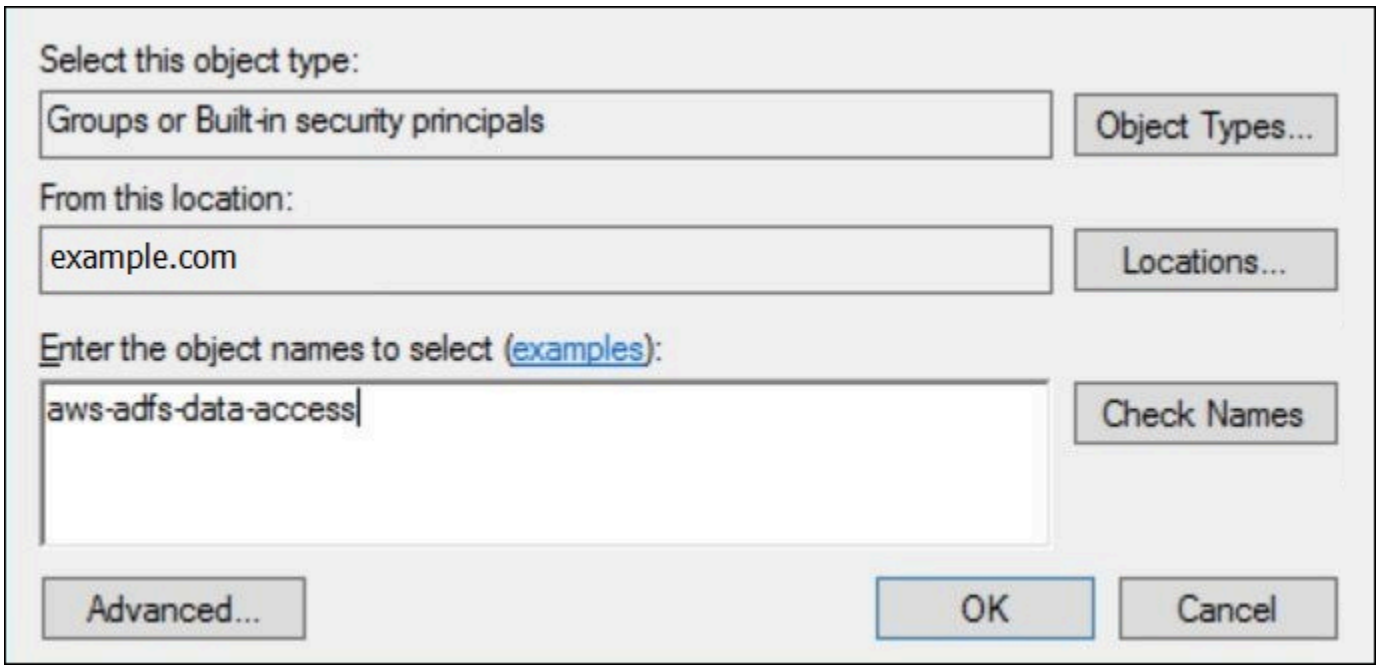
1. Wählen Sie im Server-Manager-Dashboard im Menü Tools die Option Active Directory Users and Computers (Active-Directory-Benutzer und -Computer) aus.
2. Wählen Sie für First name (Vorname) und Last name (Nachname) einen Benutzer aus (z. B. Jane Doe (Erika Mustermann)).

3. Wählen Sie im Dialogfeld Properties (Eigenschaften) des Benutzers auf der Registerkarte Member Of (Mitglied von) die Option Add (Hinzufügen) aus.



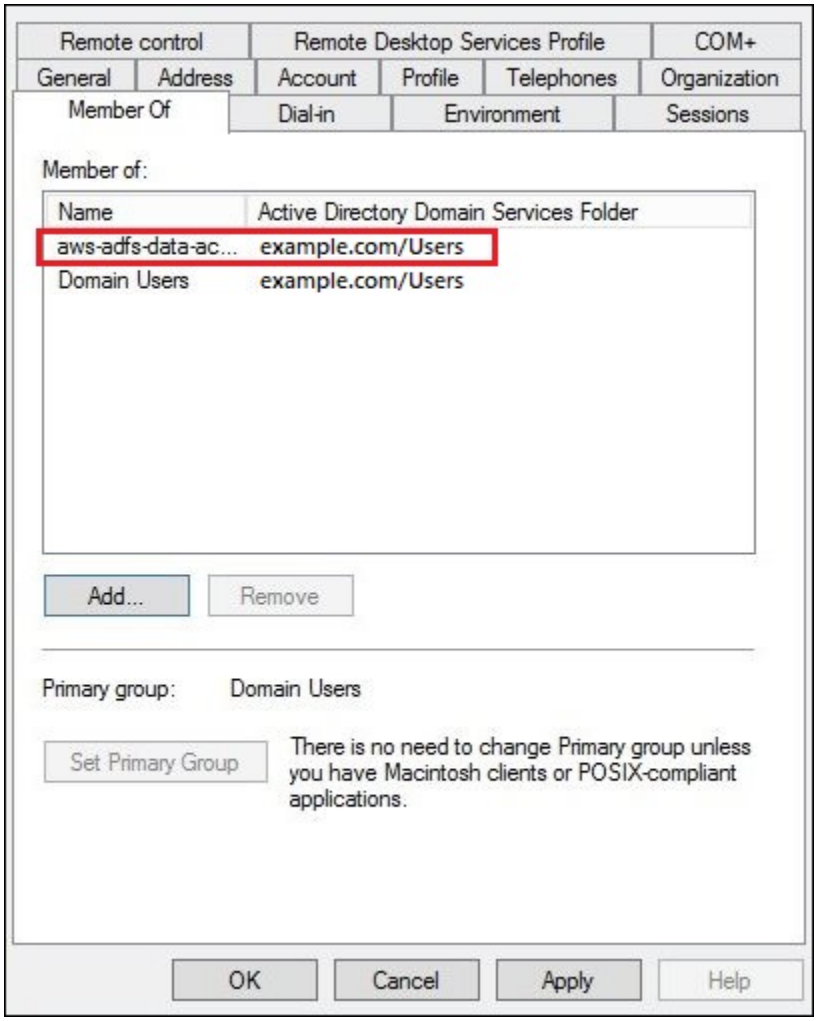
4. Fügen Sie je nach Bedarf eine oder mehrere AD-FS-Gruppen hinzu. In diesem Tutorial wird die Gruppe aws-adfs-data-access hinzugefügt.

5. Geben Sie im Dialogfeld Select Groups (Gruppen auswählen) unter Enter the object names to select (Zu verwendende Objektnamen eingeben) den Namen der AD-FS-Gruppe ein, die Sie erstellt haben (z. B. **`aws-adfs-data-access`**), und wählen Sie anschließend Check Names (Namen überprüfen) aus.



6. Wählen Sie OK.

   Im Dialogfeld Properties (Eigenschaften) für den Benutzer wird der Name der AD-Gruppe in der Liste Member of (Mitglied von) angezeigt.



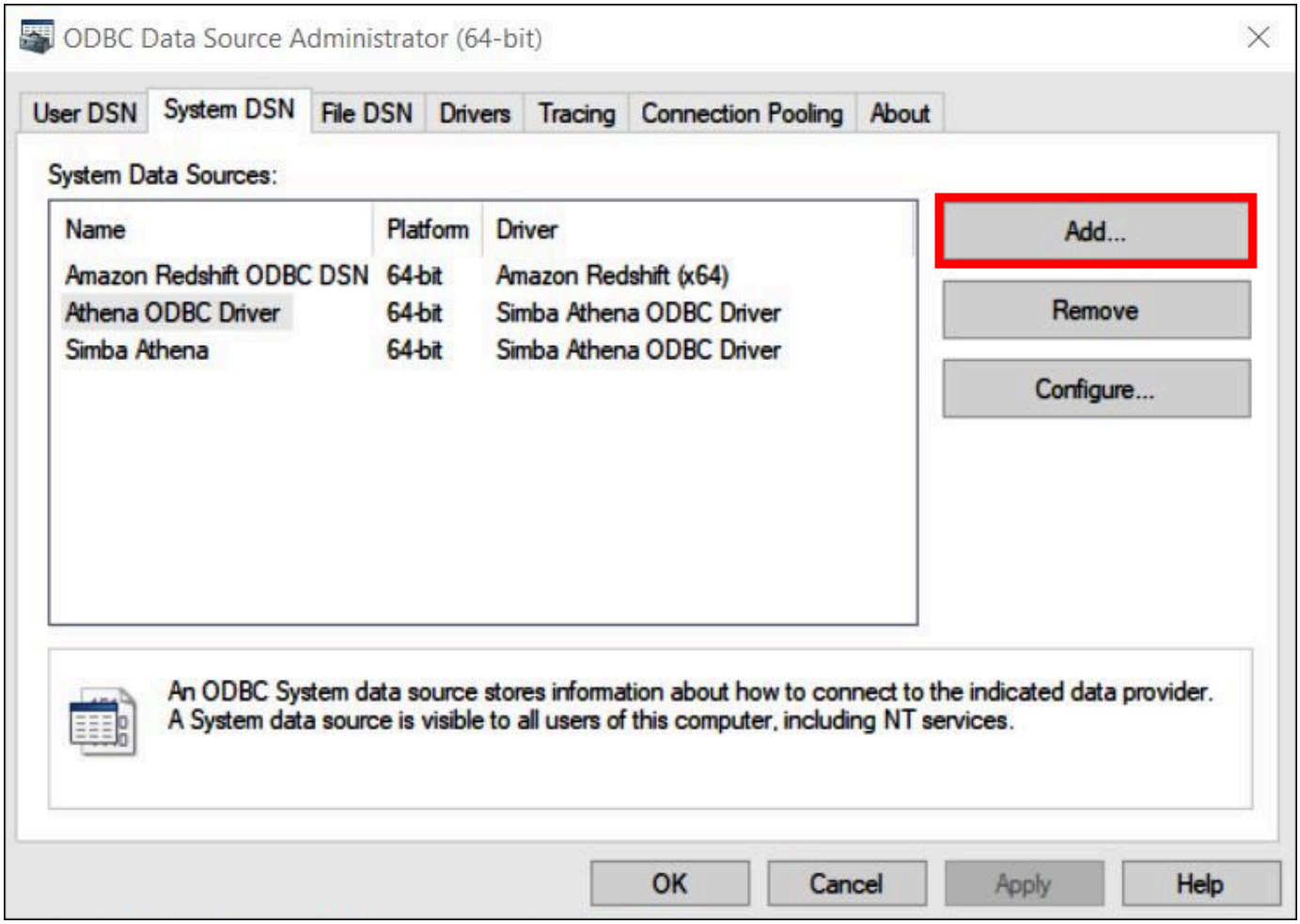
7. Wählen Sie Apply (Anwenden) und anschließend OK aus.

4. Konfigurieren der AD-FS-ODBC-Verbindung zu Athena

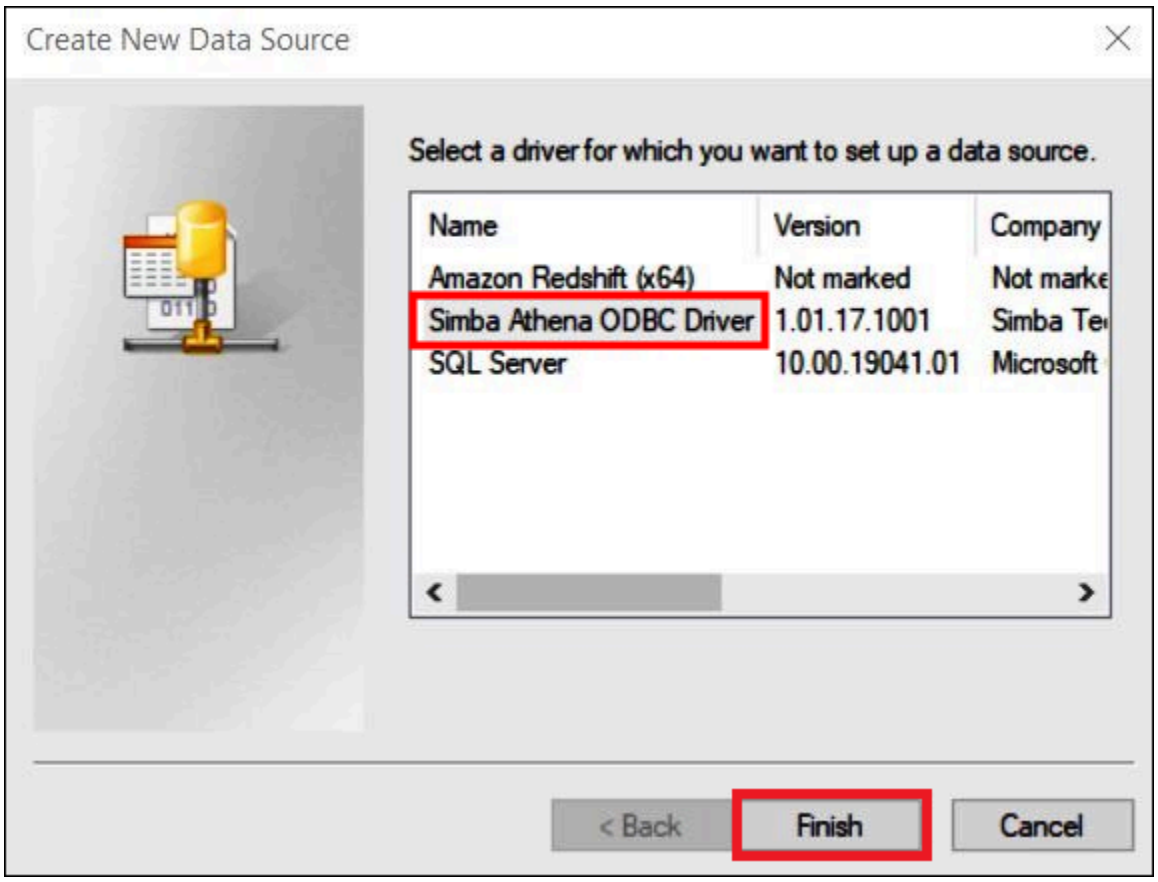
Nachdem Sie Ihre AD-Benutzer und -Gruppen erstellt haben, können Sie das ODBC-Datenquellenprogramm in Windows verwenden, um Ihre Athena-ODBC-Verbindung für AD FS zu konfigurieren.

So konfigurieren Sie die AD FS-ODBC-Verbindung zu Athena

1. Installieren Sie den ODBC-Treiber für Athena. Die Downloadlinks finden Sie unter Herstellen einer Verbindung mit Amazon Athena mit ODBC.
2. Wählen Sie in Windows Start, dann ODBC Data Sources (ODBC-Datenquellen) aus.
3. Wählen Sie in ODBC-Datenquellen-Administrator Hinzufügen (Add) aus.

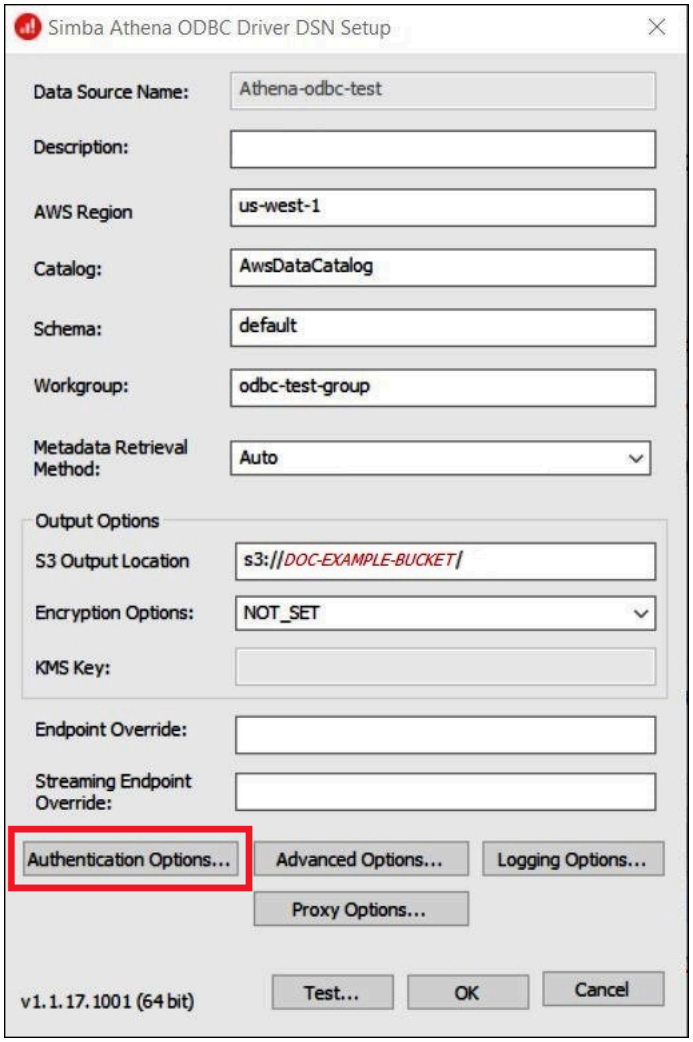


4. Wählen Sie im Dialogfeld Create New Data Source (Neue Datenquelle erstellen) die Option Simba Athena ODBC Driver (Simba-Athena-ODBC-Treiber) und anschließend Finish (Fertig stellen) aus.



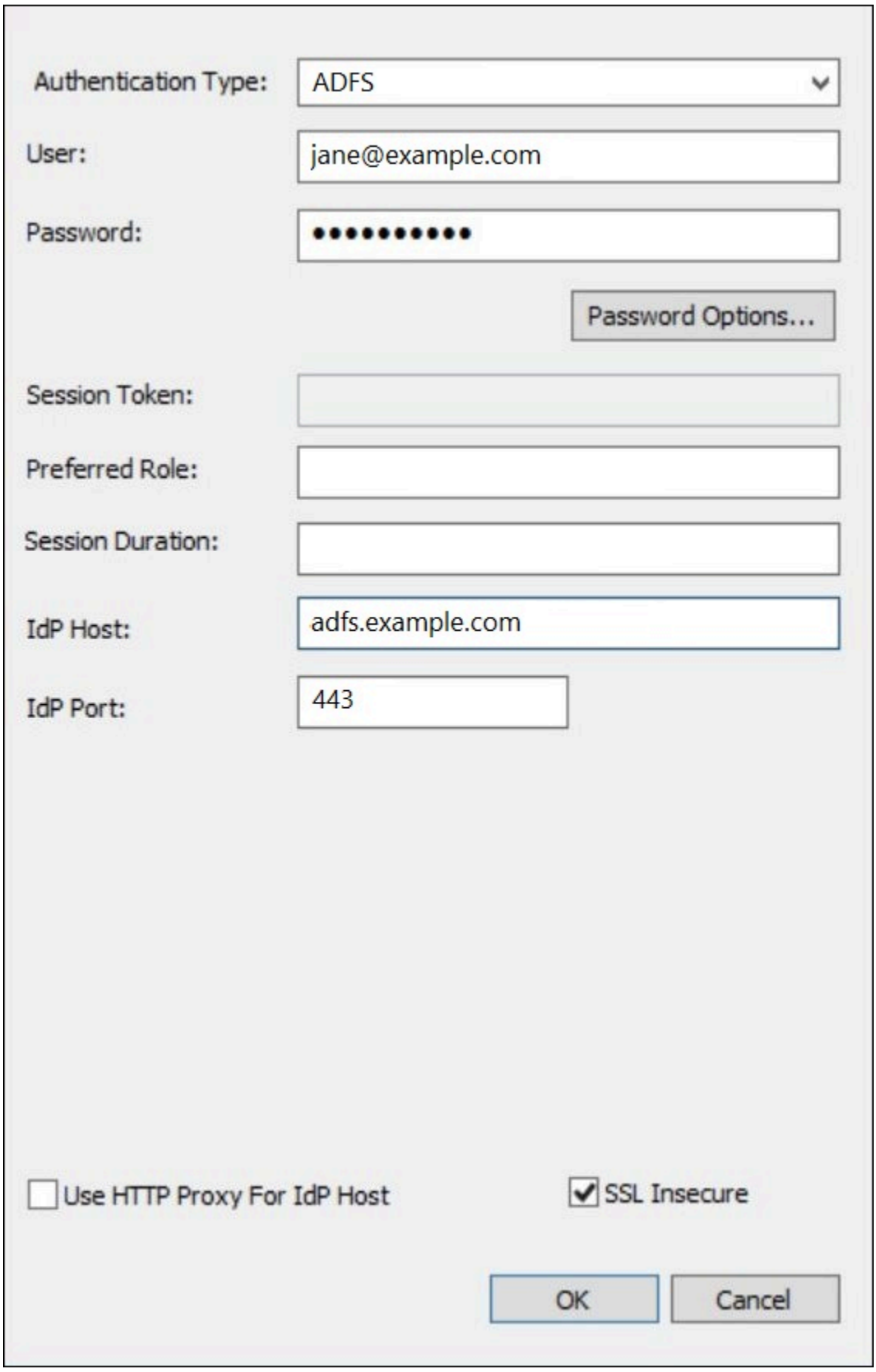
5. Geben Sie im Dialogfeld Simba Athena ODBC Driver DSN Setup (DSN-Setup des Simba-Athena-ODBC-Treibers) die folgenden Werte ein:

   - Geben Sie in Data Source Name (Datenquellenname) einen Namen für Ihre Datenquelle ein (z. B. **`Athena-odbc-test`**).
   - Geben Sie in das Feld Bezeichnung (Description) eine Beschreibung für Ihre Datenquelle ein.
   - Geben Sie für die AWS-Region die AWS-Region, die Sie verwenden (z. B. **`us-west-1`**) ein.
   - Geben Sie für S3-Ausgabespeicherort den Amazon-S3-Pfad ein, in dem Ihre Ausgabe gespeichert werden soll.



6. Wählen Sie Authentifizierungsoptionen aus.
7. Geben Sie im Dialogfeld Authentication Options (Authentifizierungsoptionen) die folgenden Werte an:

   - Wählen Sie für Authentication Type (Authentifizierungstyp) ADFS aus.
   - Geben Sie unter User (Benutzer) die E-Mail-Adresse des Benutzers ein (z. B. **`jane@example.com`**).
   - Geben Sie unter Password (Passwort) das ADFS-Passwort des Benutzers ein.
   - Geben Sie für IdP Host (IdP-Host) den AD-FS-Servernamen ein (z. B. **`adfs.example.com`**).
   - Verwenden Sie für IdP Port (IdP-Anschluss) den Standardwert 443 aus.
   - Wählen Sie die Option SSL Insecure aus.


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. Wählen Sie OK, um die Authentication Options (Authentifizierungsoptionen) zu schließen.
9. Wählen Sie Test, um die Verbindung zu testen, oder OK, um den Vorgang abzuschließen.

Konfigurieren von SSO für ODBC mit dem Okta-Plug-In und einem Okta-Identitätsanbieter

Auf dieser Seite wird beschrieben, wie Sie den Amazon-Athena-ODBC-Treiber und das Okta-Plug-In konfigurieren, um mithilfe des Okta-Identitätsanbieters Funktionen für Single-Sign-On (SSO) hinzuzufügen.

Voraussetzungen

Für das Ausfüllen der Schritte in diesem Tutorial benötigen Sie Folgendes:

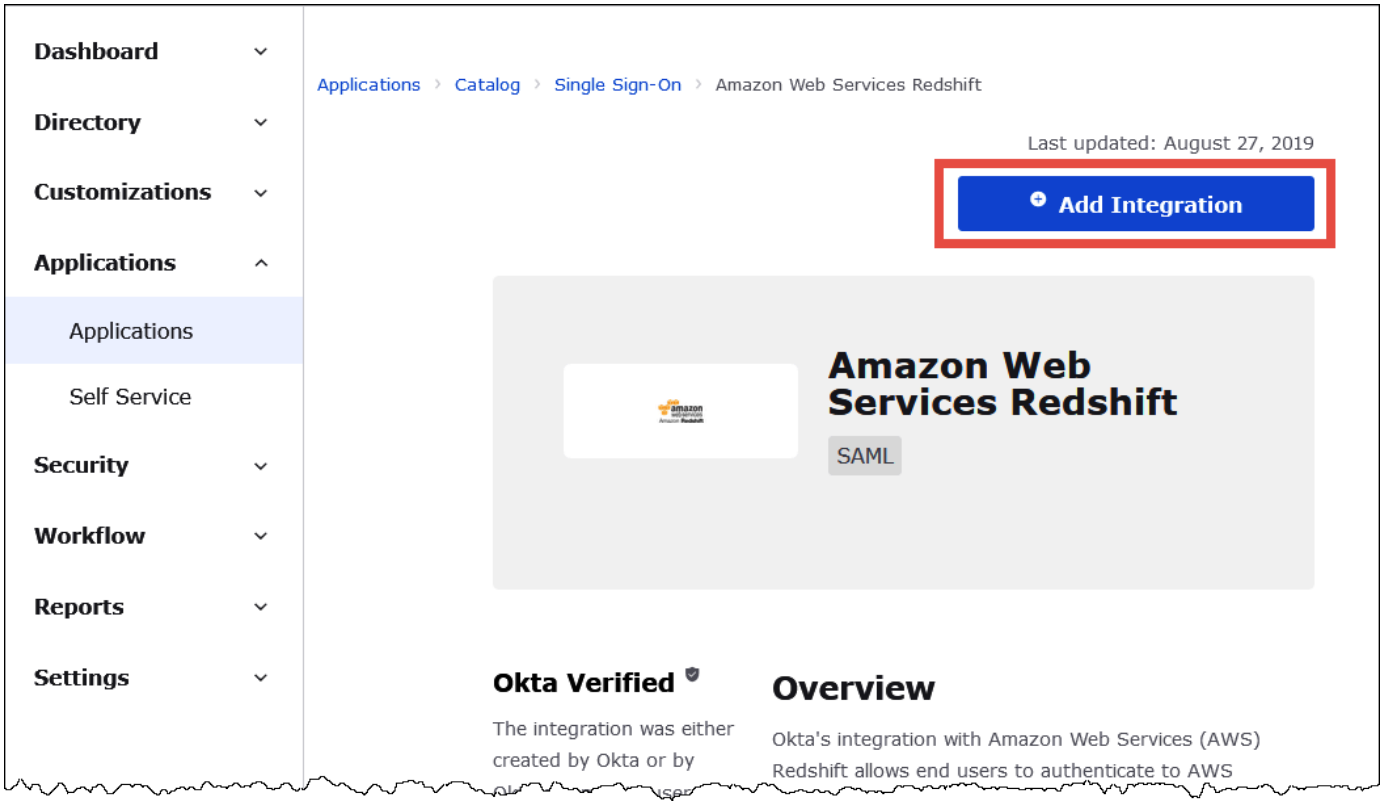
- Amazon-Athena-ODBC-Treiber Die Downloadlinks finden Sie unter Herstellen einer Verbindung mit Amazon Athena mit ODBC.
- Eine IAM-Rolle, die Sie mit SAML verwenden möchten. Weitere Informationen finden Sie unter Erstellen einer Rolle für SAML-2.0-basierten Verbund im IAM-Benutzerhandbuch.
- Ein Okta-Konto. Weitere Informationen finden Sie unter okta.com.

Erstellen einer App-Integration in Okta

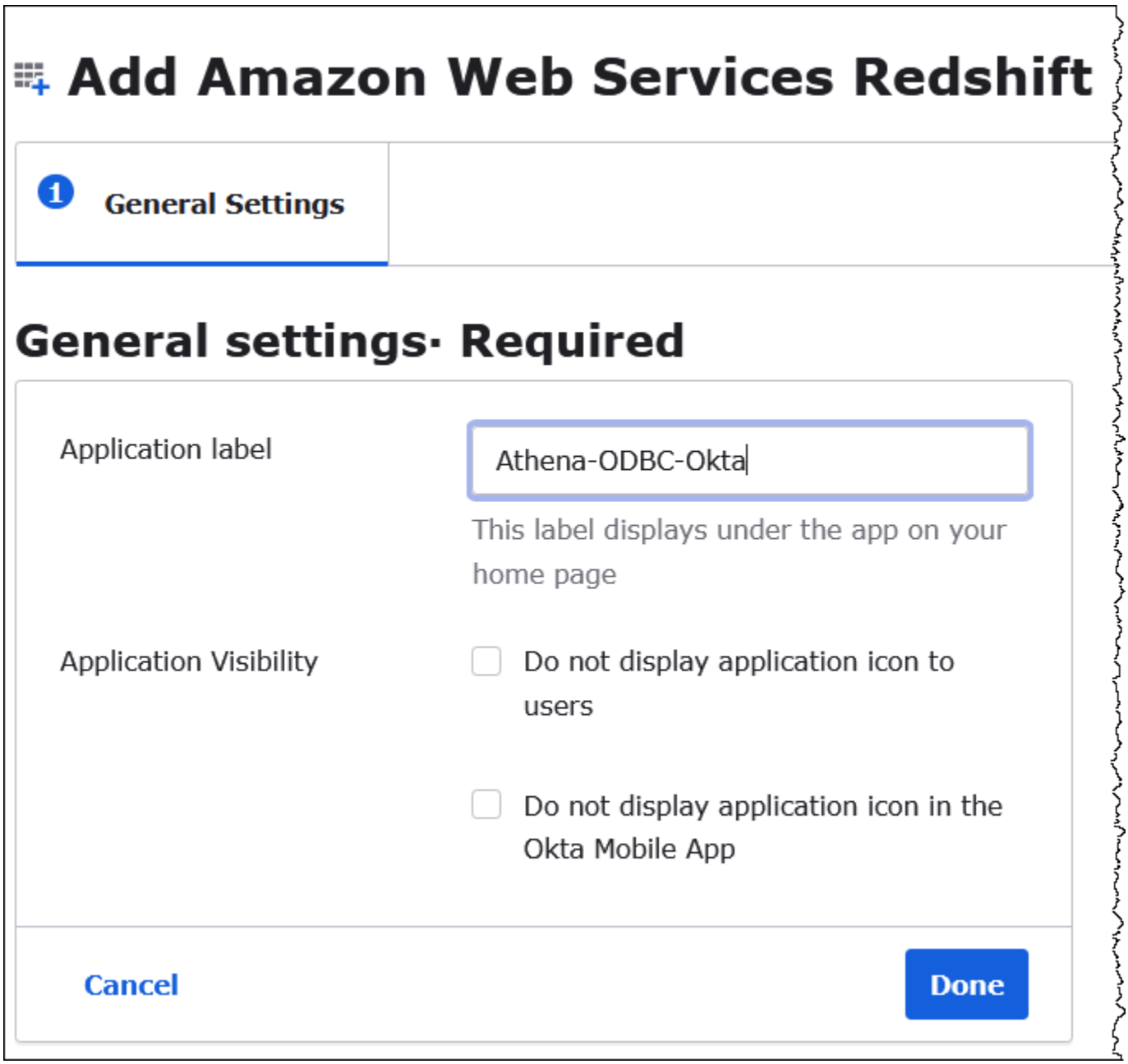
Verwenden Sie zunächst das Okta-Dashboard, um eine SAML-2.0-App für Single-Sign-On bei Athena zu erstellen und zu konfigurieren. Sie können eine vorhandene Redshift-Anwendung in Okta verwenden, um den Zugriff auf Athena zu konfigurieren.

So erstellen Sie eine App-Integration in Okta

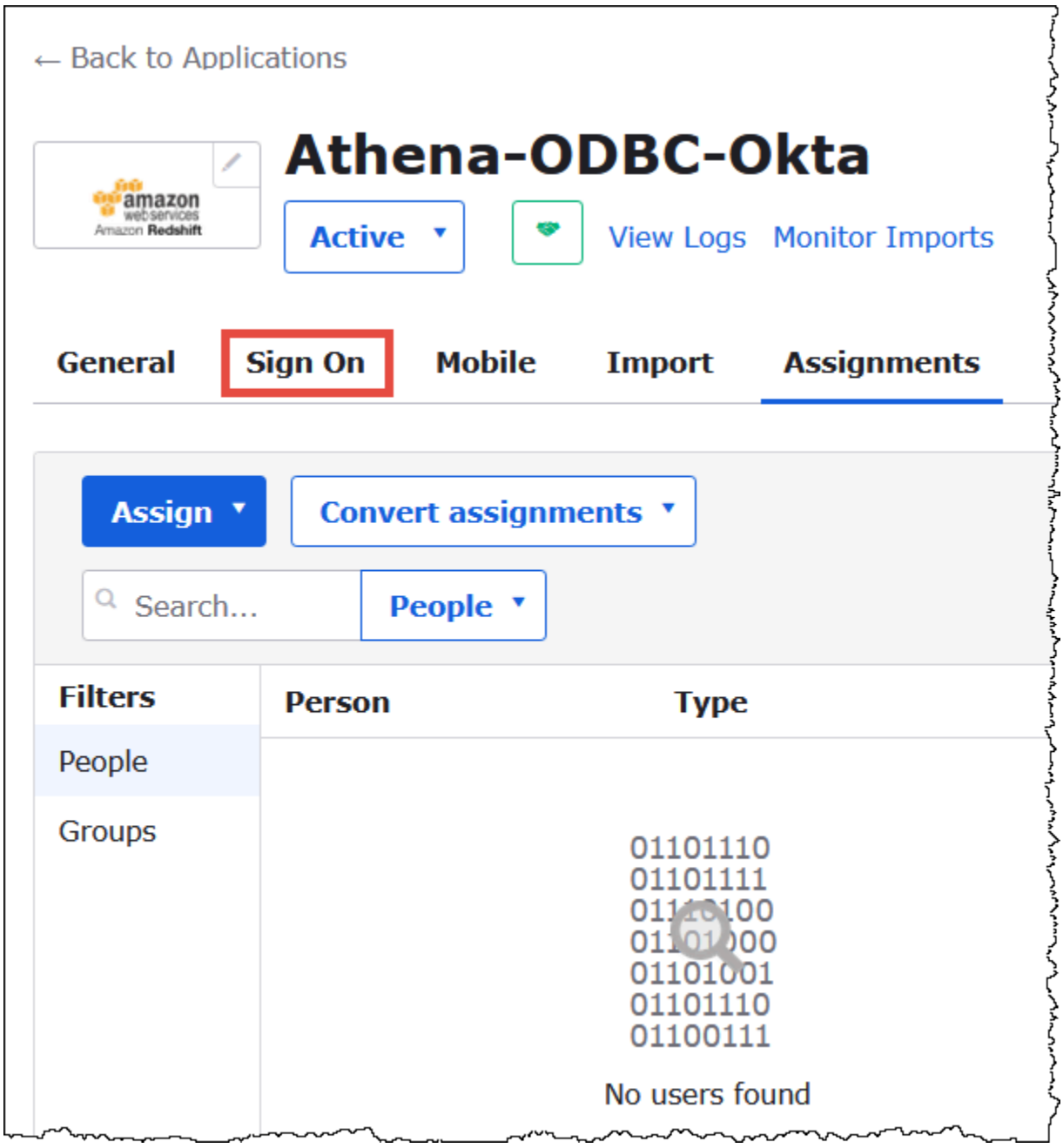
1. Melden Sie sich auf der Admin-Seite für Ihr Konto auf Okta.com an.
2. Wählen Sie die Anwendung im Navigationsbereich unter Applications (Anwendungen) Applications (Anwendungen) aus.
3. Wählen Sie auf der Seite Applications (Anwendungen) die Option Browse App Catalog (App-Katalog durchsuchen) aus.
4. Wählen Sie auf der Seite Browse App Integration Catalog (App-Integrationskatalog durchsuchen) im Abschnitt Use Case (Anwendungsfall) die Option All Integrations (Alle Integrationen) aus.
5. Geben Sie im Suchfeld den Wert Amazon Web Services Redshift ein und wählen Sie danach Amazon Web Services Redshift SAML aus.
6. Wählen Sie Add Integration (Integration hinzufügen) aus.



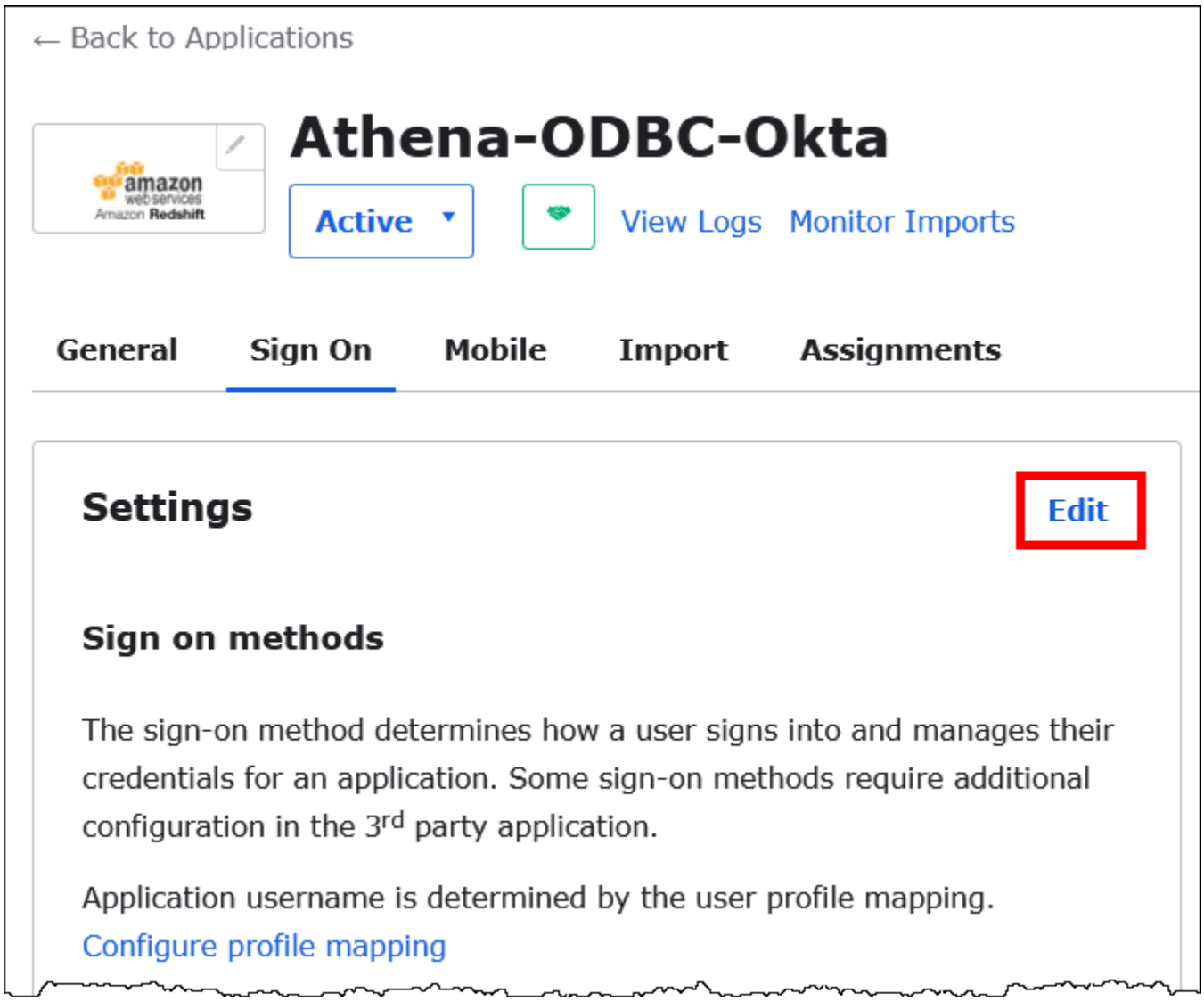
7. Geben Sie im Abschnitt General Settings Required (Allgemeine Einstellungen erforderlich) für Application label (Anwendungsbezeichnung) einen Namen für die Anwendung ein. In diesem Tutorial wird der Name Athena-ODBC-Okta verwendet.



8. Wählen Sie Done (Erledigt) aus.
9. Wählen Sie auf der Seite für Ihre Okta-Anwendung (z. B.Athena-ODBC-Okta) die Option Sign On (Anmelden) aus.



10. Wählen Sie im Abschnitt Settings (Einstellungen) die Option Edit (Bearbeiten) aus.



11. Konfigurieren Sie im Abschnitt Advanced Sign-on Settings (Erweiterte Anmeldeeinstellungen) die folgenden Werte.

    - Geben Sie für IdP ARN and Role ARN (Identitätsanbieter-ARN und Rollen-ARN) Ihren AWS-Identitätsanbieter-ARN und Rollen-ARN als kommagetrennte Werte ein. Informationen zum IAM-Rollenformat finden Sie unter Konfigurieren von SAML-Assertions für die Authentifizierungsantwort im IAM-Benutzerhandbuch.
    - Geben Sie für Session Duration (Sitzungsdauer) einen Wert zwischen 900 und 43 200 Sekunden ein. In diesem Tutorial wird der Standardwert 3 600 (1 Stunde) verwendet.

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

Die Einstellungen DbUser Format (DbUser-Format), AutoCreate (Automatisch erstellen) und Allowed DBGroups (Zulässige Datenbankgruppen) werden von Athena nicht verwendet. Sie müssen sie nicht konfigurieren.

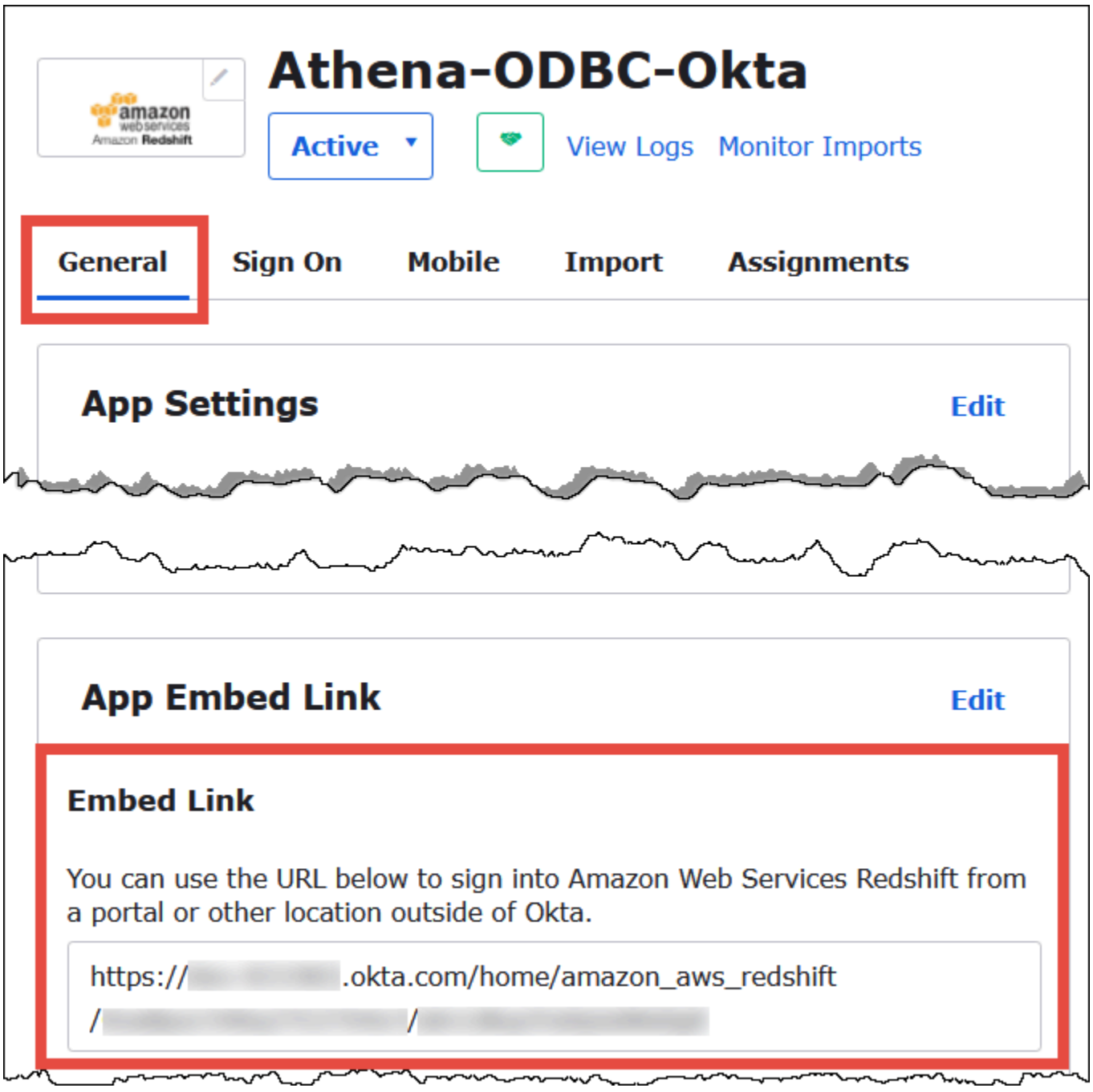
12. Wählen Sie Save (Speichern).

Abrufen von ODBC-Konfigurationsinformationen von Okta

Nachdem Sie die Okta-Anwendung erstellt haben, können Sie die ID und die Identitätsanbieter-Host-URL der Anwendung abrufen. Sie benötigen diese später, wenn Sie ODBC für die Verbindung mit Athena konfigurieren.

So rufen Sie ODBC-Konfigurationsinformationen von Okta ab

1. Wählen Sie das Symbol General (Allgemeines) und scrollen Sie dann nach unten zu Ihrer Okta-Anwendung zum Abschnitt App Embed Link (App-Einbettungslink).



Die Embed Link-URL (Einbettungs-Link-URL) hat das folgende Format:

```
https://trial-1234567.okta.com/home/amazon_aws_redshift/Abc1de2fghi3J45kL678/
abc1defghij2klmNo3p4
```

2. Extrahieren Sie aus Ihrer Embed Link-URL (Einbettungs-Link-URL) die folgenden Informationen und speichern Sie sie:

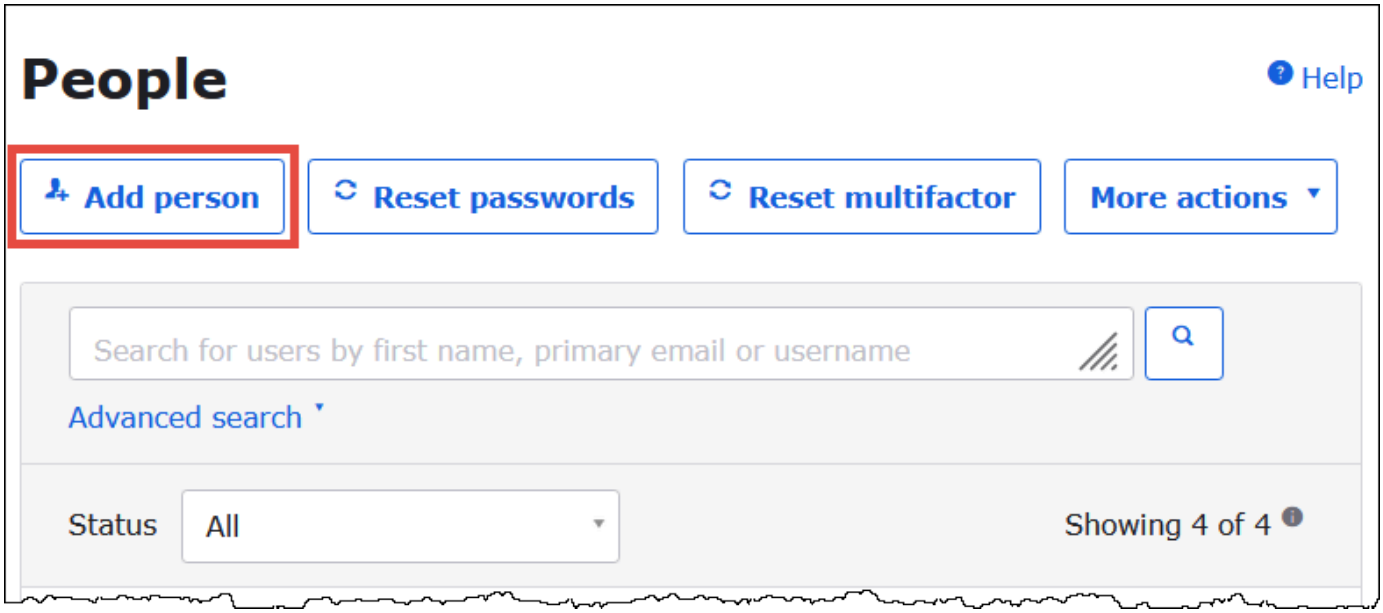
- Das erste Segment nach `https://`, bis einschließlich `okta.com` (Beispiel: trial-1234567.okta.com). Das ist Ihr Identitätsanbieter-Host.
- Die letzten beiden Segmente der URL, einschließlich des Schrägstrichs in der Mitte. Bei den Segmenten handelt es sich um zwei 20-stellige Zeichenfolgen, die eine Mischung aus Zahlen sowie Groß- und Kleinbuchstaben (z. B.ABC1DE2FGHI3J45KL678/ ABC1DEFGHIJ2KLMNO3P4) enthalten. Dies ist Ihre Anwendungs-ID.

Hinzufügen eines Benutzers zur Okta-Anwendung

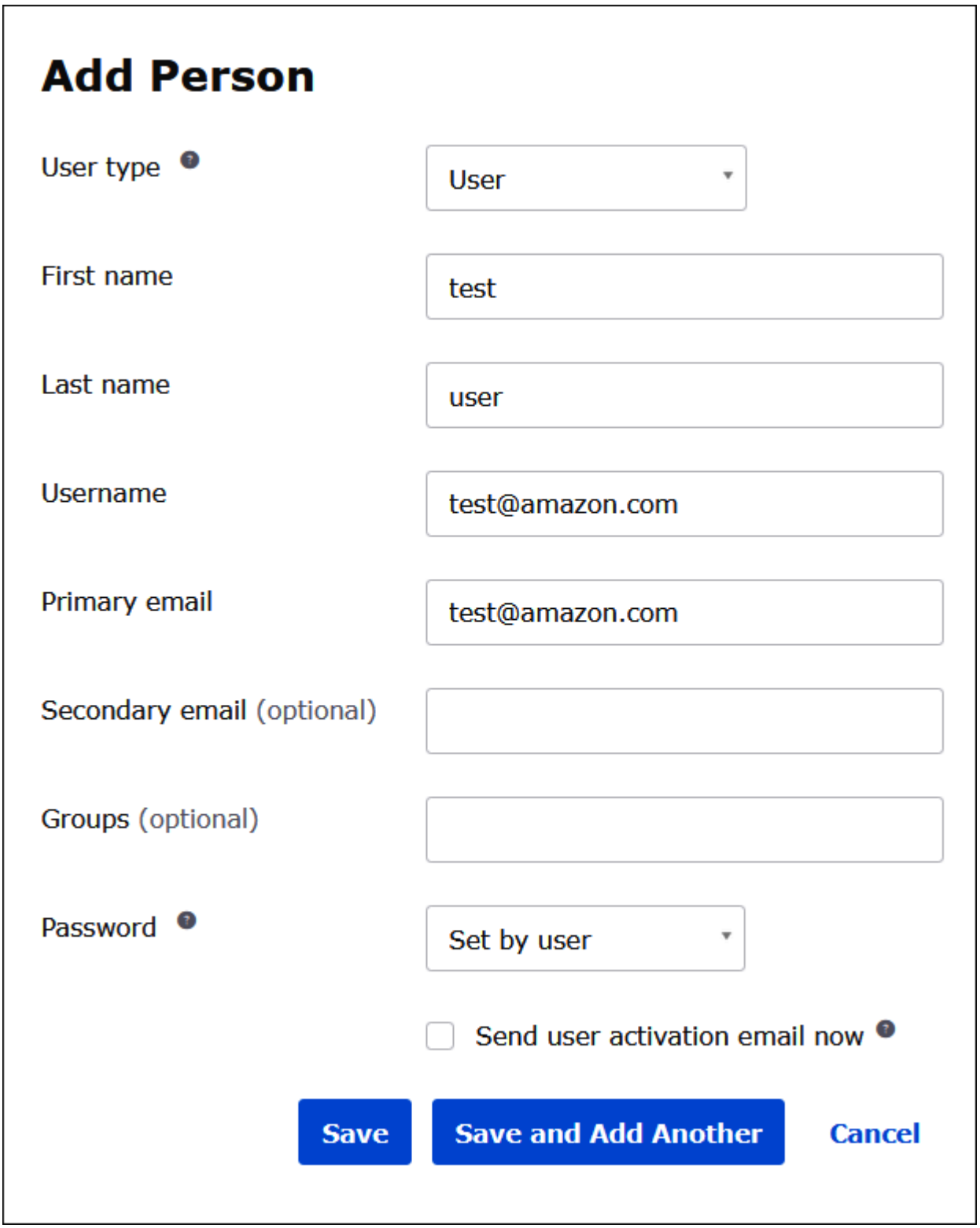
Jetzt können Sie Ihrer Okta-Anwendung einen Benutzer hinzufügen.

So fügen Sie einen Benutzer zur Okta-Anwendung hinzu

1. Wählen Sie im linken Navigationsbereich Verzeichnis und dann Personen aus.
2. Wählen Sie Person hinzufügen aus.



3. Geben Sie im Dialogfeld Add Person (Person hinzufügen) die folgenden Informationen ein:
   - Geben Sie die Werte für Vorname und Nachname ein. In diesem Tutorial wird ein **`test user`** verwendet.
   - Geben Sie Werte für Username (Benutzername) und Primary email (Primäre E-Mail-Adresse) ein. In diesem Tutorial wird **`test@amazon.com`** für beides verwendet. Die Sicherheitsanforderungen für Passwörter können variieren.

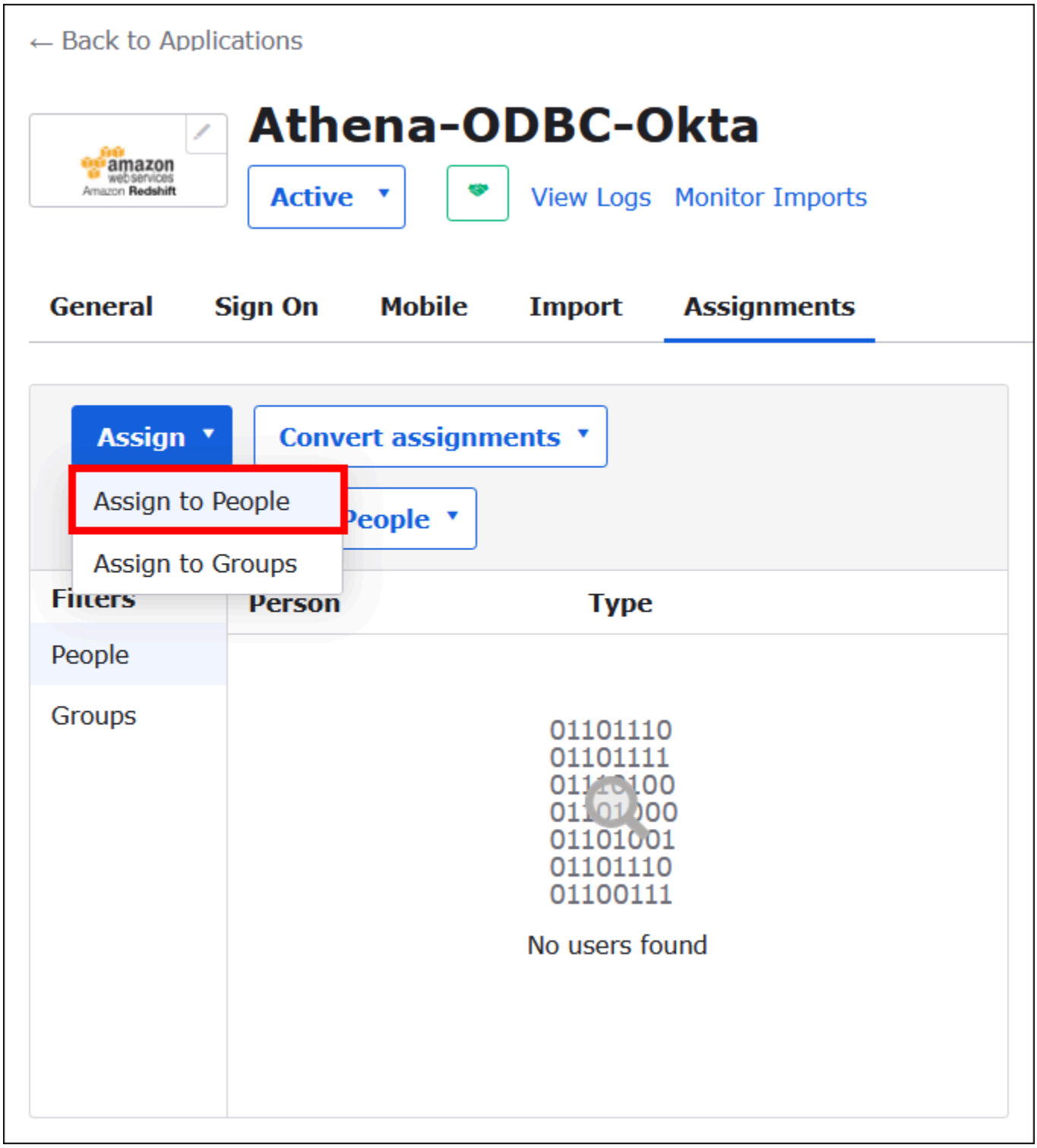


4. Wählen Sie Save (Speichern).

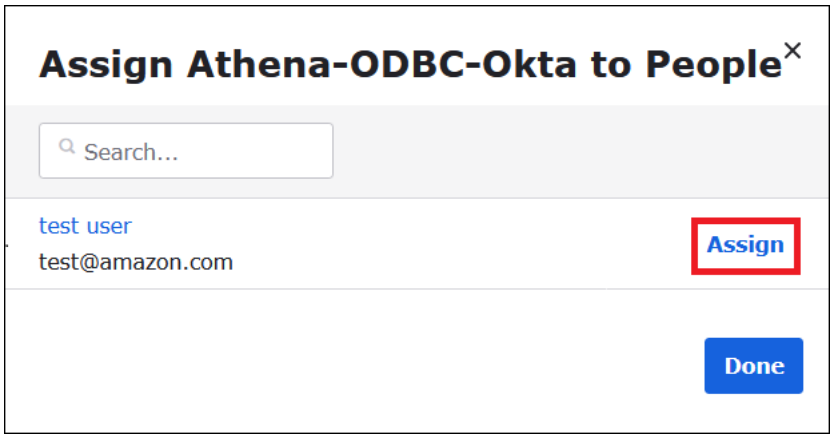
Jetzt können Sie den erstellten Benutzer Ihrer Anwendung zuweisen.

So weisen Sie den Benutzer Ihrer Anwendung zu:

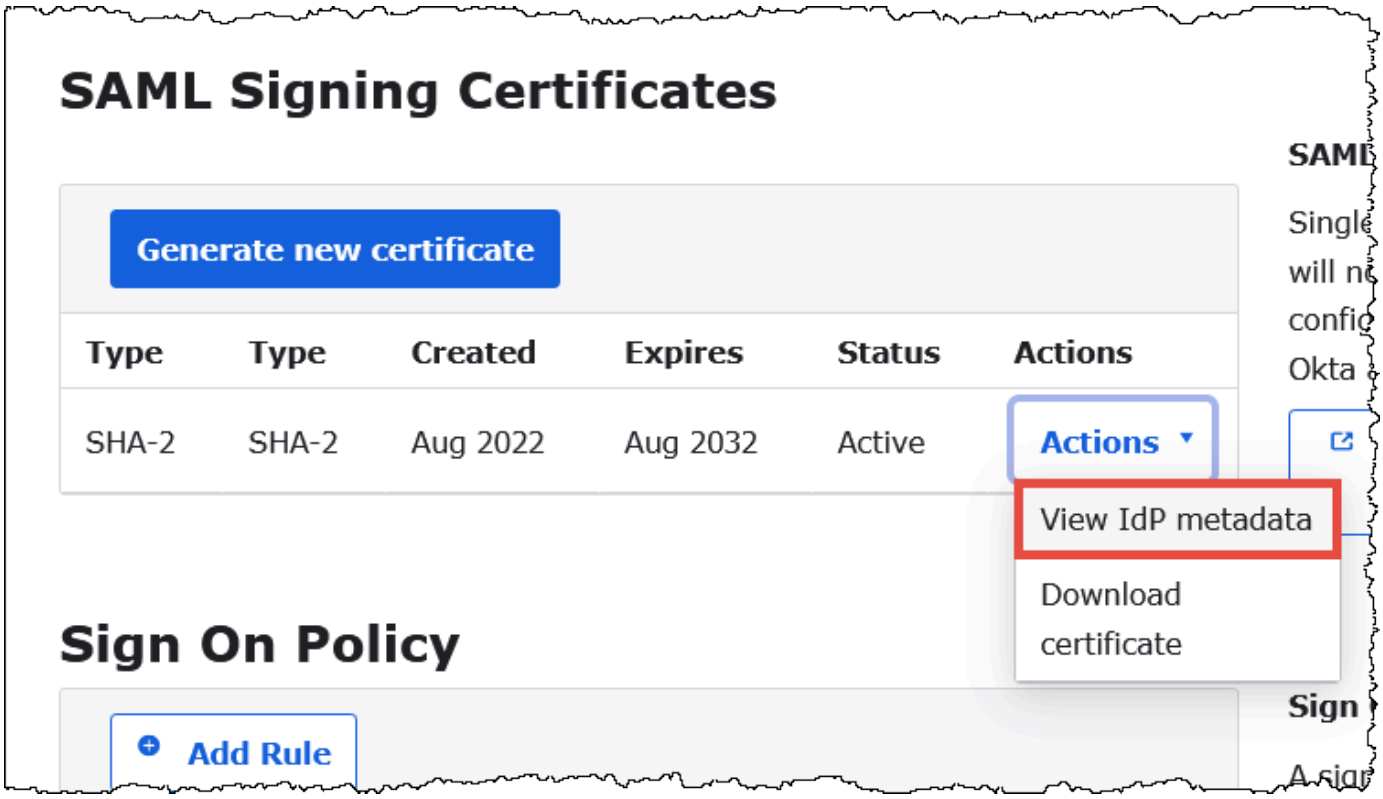
1. Wählen Sie im Navigationsbereich Applications (Anwendungen), Applications (Anwendungen) und dann den Namen Ihrer Anwendung aus (z. B. Athena-ODBC-Okta).
2. Klicken Sie auf Assign (Zuweisen) und danach auf Assign to People (Personen zuweisen).



3. Wählen Sie die Option Assign (Zuweisen) für Ihren Benutzer aus und klicken Sie dann auf Done (Erledigt).



4. Klicken Sie an der Eingabeaufforderung auf Save and Go Back (Speichern und zurückkehren). Das Dialogfeld zeigt den Status des Benutzers als Assigned (Zugewiesen) an.
5. Wählen Sie Done (Erledigt) aus.
6. Wählen Sie die Registerkarte Sign On (Anmelden) aus.
7. Scrollen Sie auf der Seite nach unten zum Abschnitt SAML Signing Certificates (SAML-Signaturzertifikate).
8. Wählen Sie Actions (Aktionen).
9. Öffnen Sie das Kontextmenü (rechte Maustaste) für View IdP metadata (Identitätsanbieter-Metadaten anzeigen) und wählen Sie dann die Browseroption zum Speichern der Datei aus.
10. Speichern Sie die Datei mit der `.xml`-Dateierweiterung.

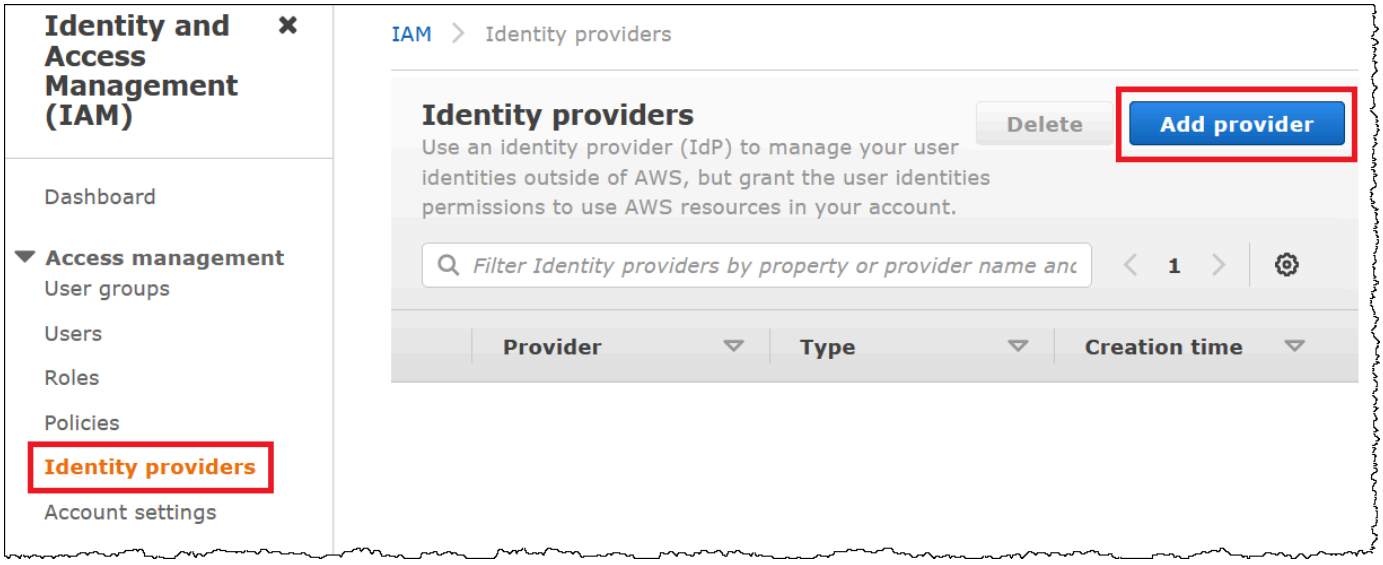


Erstellen eines AWS-SAML-Identitätsanbieters und einer Rolle

Jetzt können Sie die Metadaten-XML-Datei in die IAM-Konsole in AWS hochladen. Sie verwenden diese Datei, um einen AWS-SAML-Identitätsanbieter und eine Rolle zu erstellen. Verwenden Sie ein AWS-Services-Administratorkonto, um diese Schritte auszuführen.

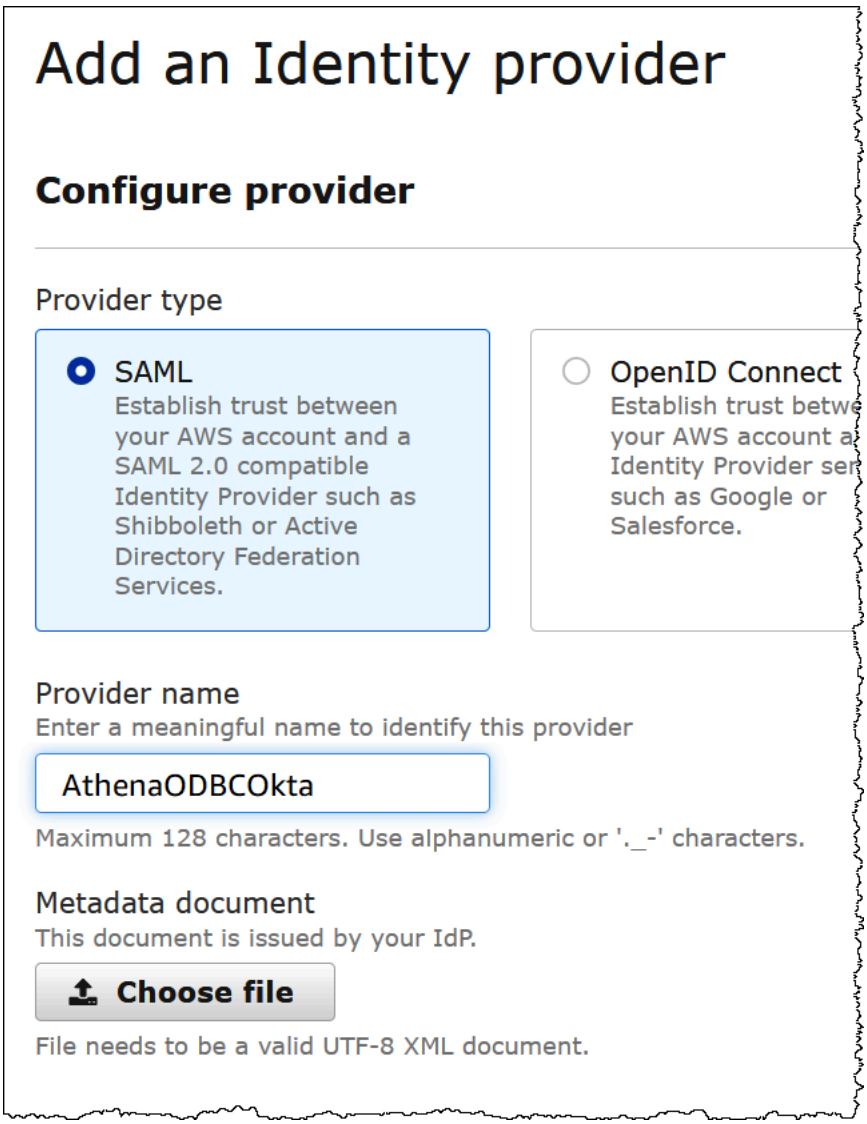
So erstellen Sie einen SAML-Identitätsanbieter und eine Rolle in AWS

1. Melden Sie sich bei AWS Management Console an, und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/IAM/.
2. Wählen Sie im Navigationsbereich Identity providers (Identitätsanbieter) und dann Add provider (Anbieter hinzufügen) aus.



3. Geben Sie auf der Seite Add an Identity provider (Identitätsanbieter hinzufügen) für Configure provider (Anbieter konfigurieren) die folgenden Informationen ein.

   - Wählen Sie als Provider type (Anbietertyp) SAML aus.
   - Geben Sie für Provider name (Anbietername) einen Namen für Ihren Anbieter ein (z. B. **`AthenaODBCOkta`**).
   - Verwenden Sie für das Metadatendokument die Option Datei auswählen, um die heruntergeladene XML-Metadatendatei des Identitätsanbieters (IdP) hochzuladen.



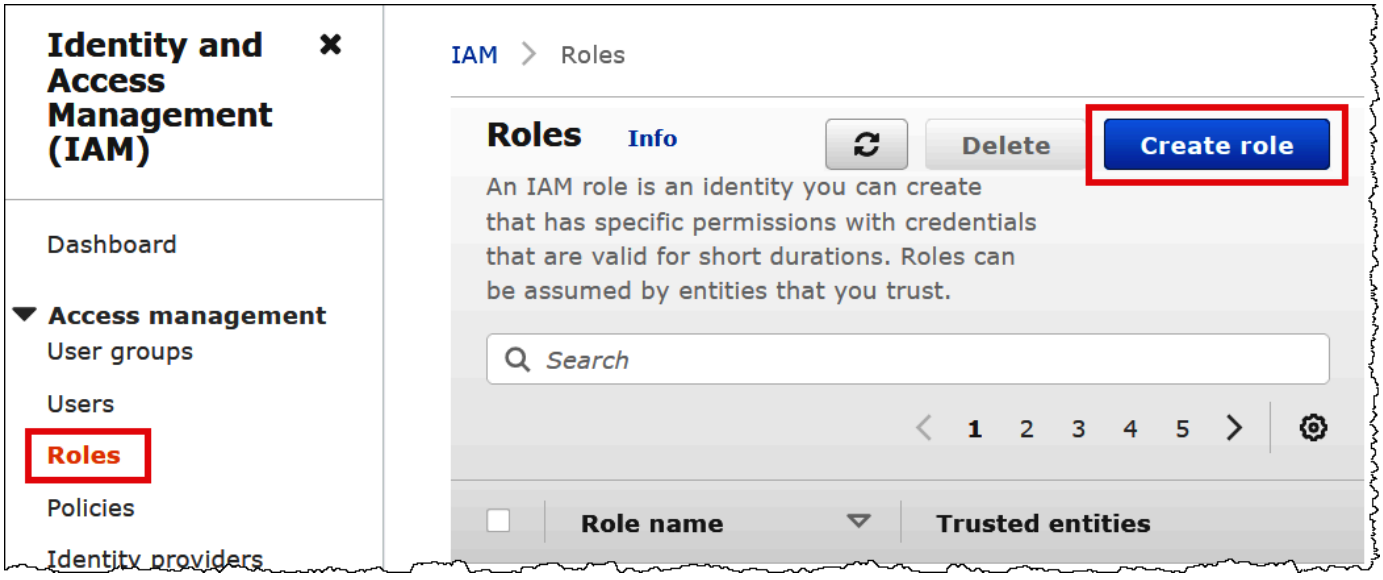
4. Wählen Sie Add provider (Anbieter hinzufügen) aus.

Erstellen einer IAM-Rolle für den Athena- und Amazon S3-Zugriff

Jetzt können Sie eine IAM-Rolle für den Zugriff auf Athena und Amazon S3 erstellen. Sie weisen diese Rolle Ihrem Benutzer zu. So können Sie dem Benutzer Single-Sign-On-Zugriff auf Athena gewähren.

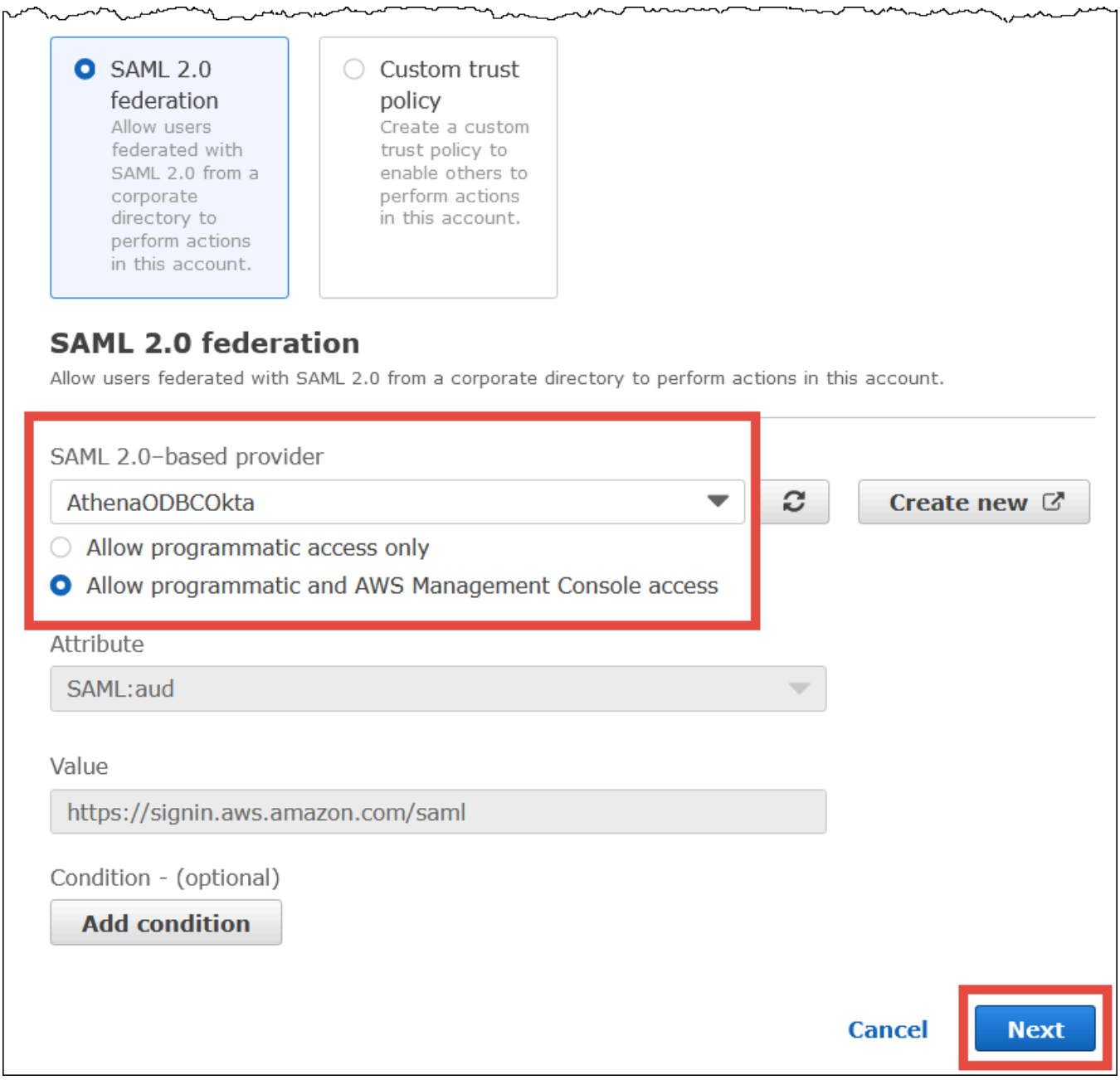
So erstellen Sie eine IAM-Rolle für Ihren Benutzer

1. Wählen Sie im Navigationsbereich der IAM-Konsole Rollen und dann Rolle erstellen aus.

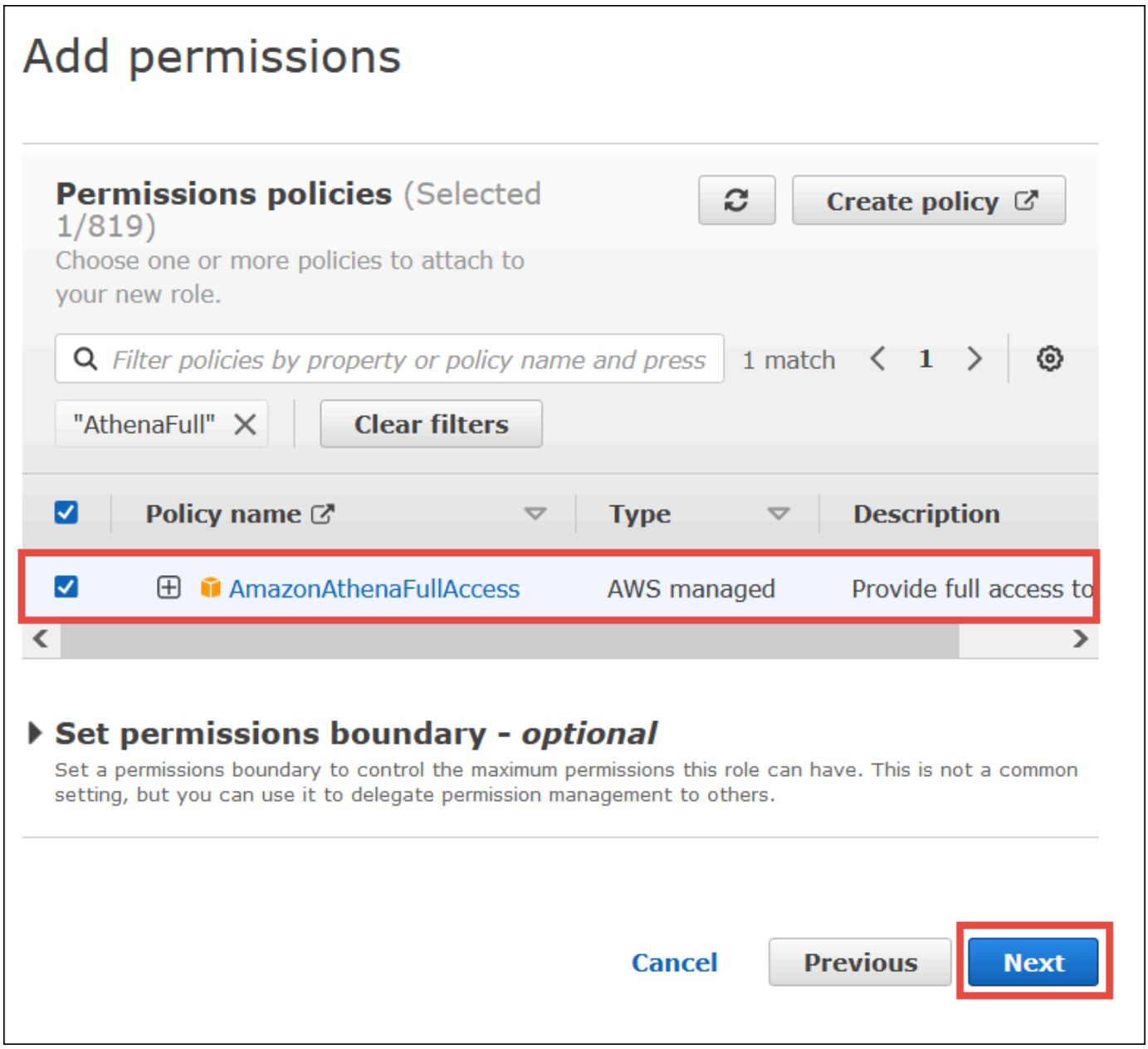


2. Wählen Sie auf der Seite Create role (Rolle erstellen) die folgenden Optionen aus:

   - Wählen Sie für Typ der vertrauenswürdigen Entität auswählen die Option SAML-2.0-Verbund aus.
   - Wählen Sie für SAML 2.0–based provider (SAML 2.0-basierter Anbieter) den von Ihnen erstellten SAML-Identitätsanbieter aus (z. B.Athenao DBCOKTA).
   - Wählen Sie Allow programmatic and AWS Management Console access (Programmgesteuerten und -Zugriff erlauben) aus.



3. Wählen Sie Next (Weiter).
4. Geben Sie auf der Seite Add Permissions (Berechtigungen hinzufügen) für Filter policies (Richtlinien filtern) den Wert **`AthenaFull`** ein, und drücken Sie die Eingabetaste.
5. Wählen Sie die verwaltete `AmazonAthenaFullAccess`-Richtlinie und dann Next aus.



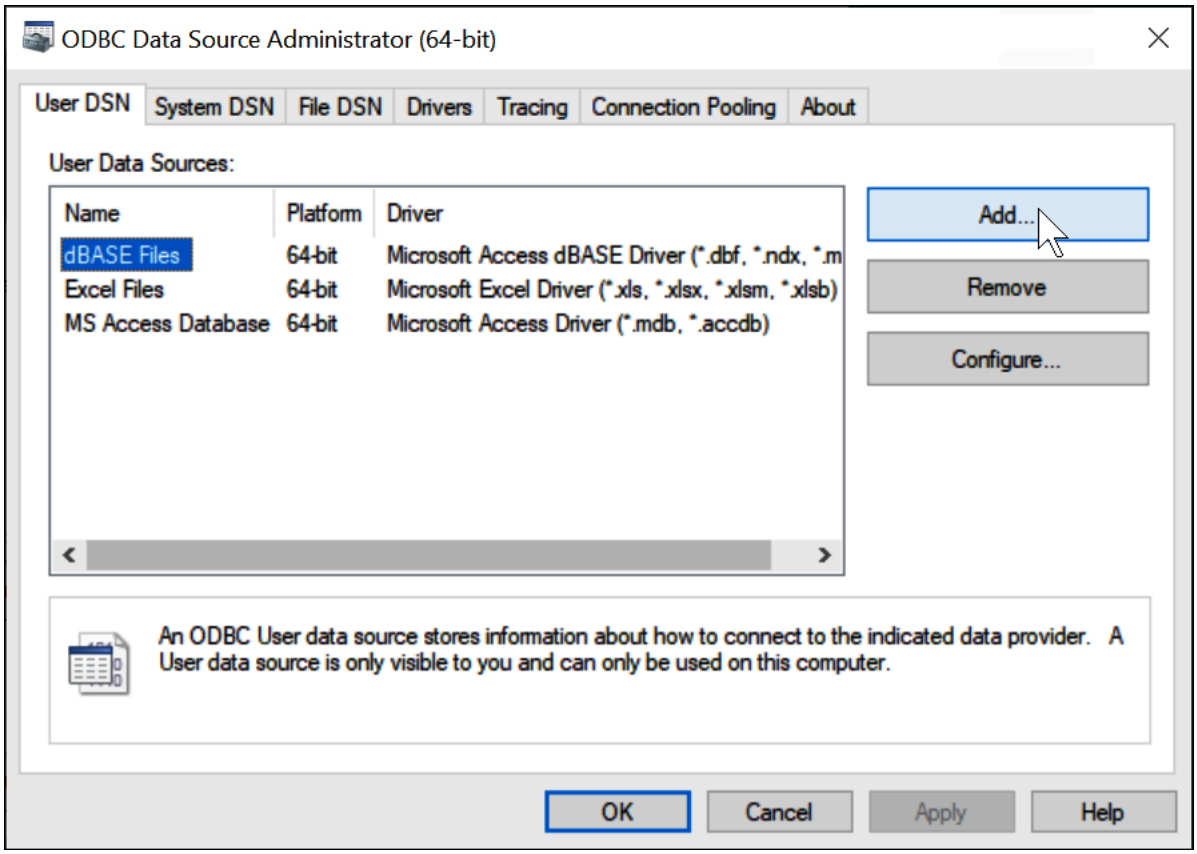
6. Geben Sie auf der Seite Name, review, and create (Benennen, prüfen und erstellen) für Role name (Rollenname) einen Namen für die Rolle ein (z. B. **`Athena-ODBC-OktaRole`**), und wählen Sie dann Create role (Rolle erstellen) aus.

Konfigurieren der Okta-ODBC-Verbindung zu Athena

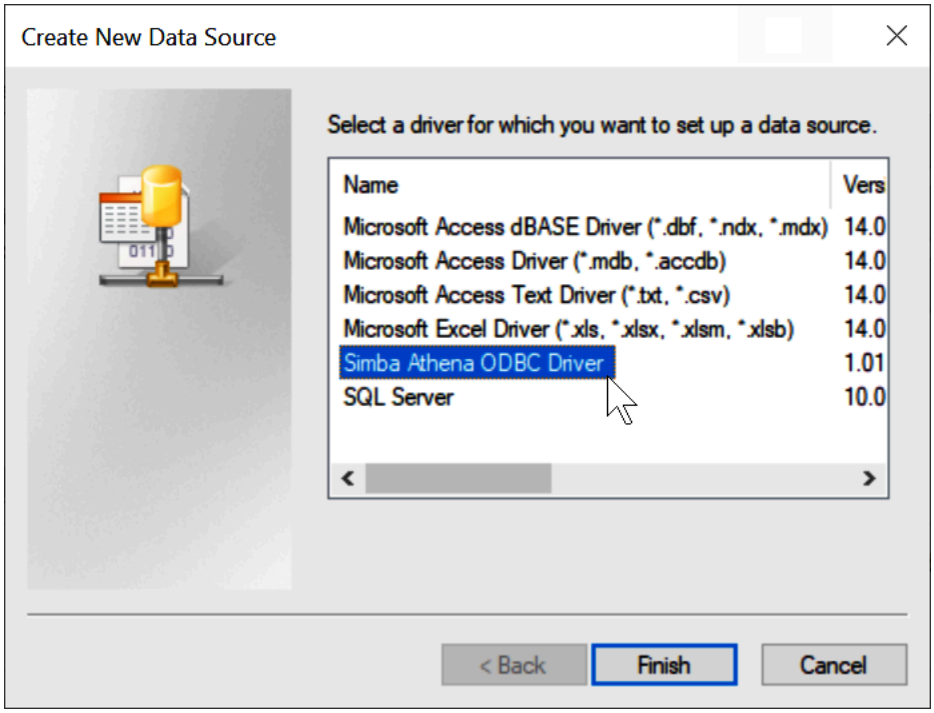
Jetzt können Sie die Okta-ODBC-Verbindung zu Athena mithilfe des ODBC-Datenquellen-Programms in Windows konfigurieren.

So konfigurieren Sie Ihre Okta-ODBC-Verbindung zu Athena

1. Starten Sie unter Windows das ODBC-Datenquellen-Programm.
2. Wählen Sie in ODBC-Datenquellen-Administrator Hinzufügen (Add) aus.

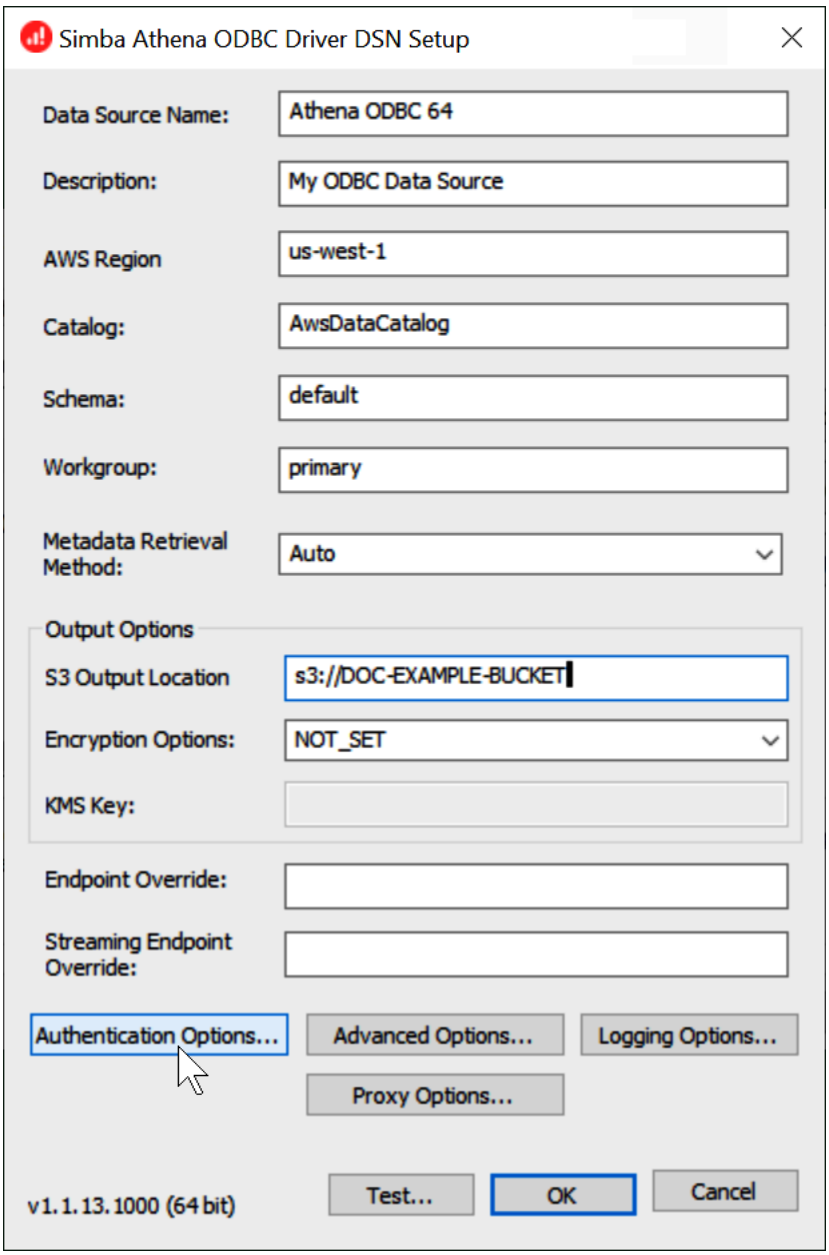


3. Wählen Sie den Simba-Athena-ODBC-Treiber und dann Beenden (Finish) aus.



4. Geben Sie in SSN-Einrichtung für Simba-Athena-ODBC-Treiber die beschriebenen Werte ein.

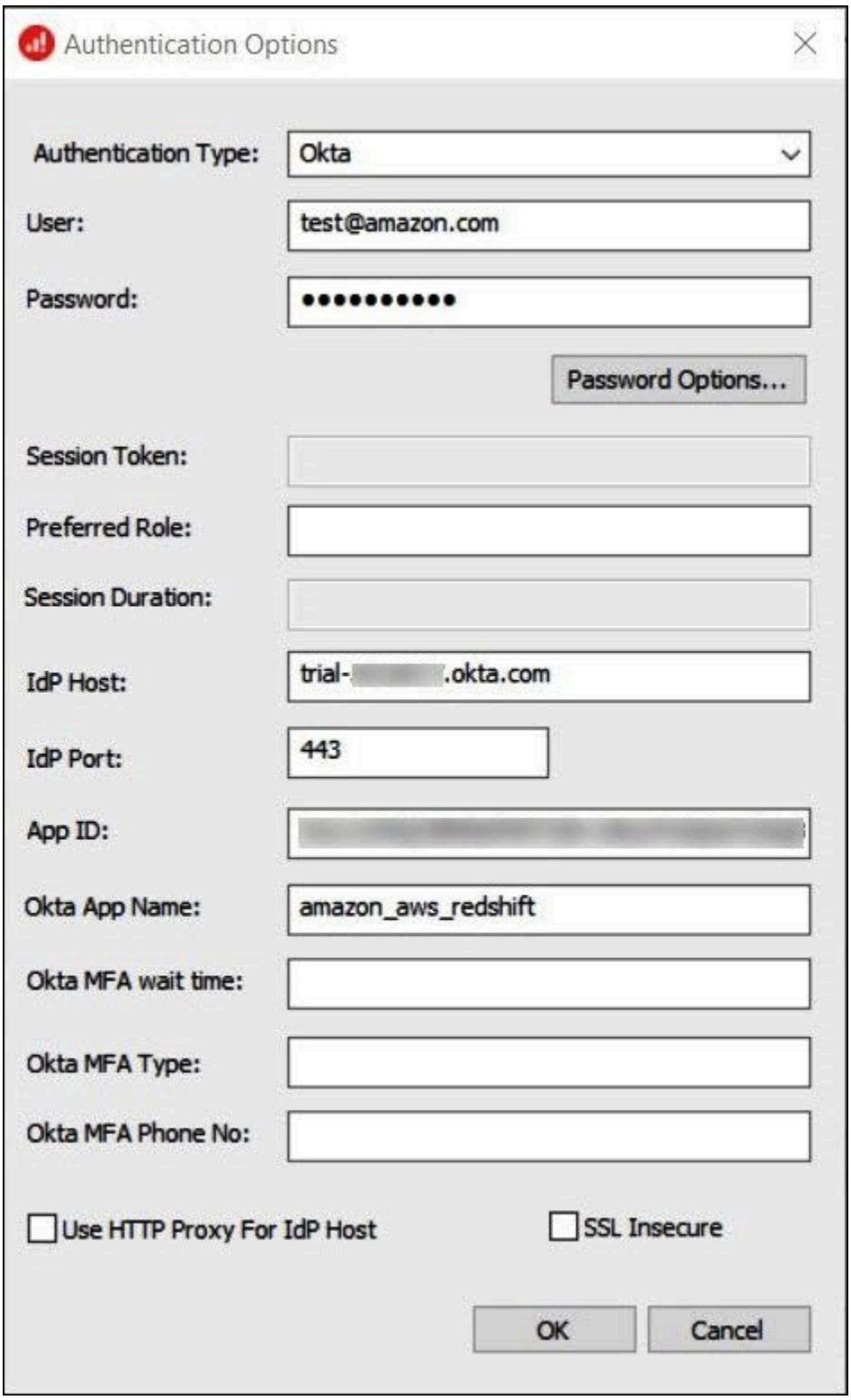
   - Geben Sie in Data Source Name (Datenquellenname) einen Namen für Ihre Datenquelle ein (z. B. **`Athena ODBC 64`**).
   - Geben Sie in das Feld Bezeichnung (Description) eine Beschreibung für Ihre Datenquelle ein.
   - Geben Sie für die AWS-Region die AWS-Region ein, die Sie verwenden (z. B. **`us-west-1`**).
   - Geben Sie für S3-Ausgabespeicherort den Amazon-S3-Pfad ein, in dem Ihre Ausgabe gespeichert werden soll.



5. Wählen Sie Authentifizierungsoptionen aus.
6. Wählen Sie im Dialogfeld Authentifizierungsoptionen die folgenden Werte aus oder geben Sie sie ein.

   - Wählen Sie für Authentication Type (Authentifizierungstyp) Okta aus.
   - Geben Sie unter User (Benutzer) Ihren Okta-Benutzernamen ein.
   - Geben Sie unter Password (Passwort) Ihr Okta-Passwort ein.
   - Geben Sie für IdP Host (Identitätsanbieter-Host) den Wert ein, den Sie zuvor aufgezeichnet haben (z. B.**`trial-1234567.okta.com`**).

- Geben Sie für IdP Port (Identitätsanbieter-Port) **`443`** ein.
- Geben Sie für App-ID (App-ID) den Wert ein, den Sie zuvor aufgezeichnet haben (die letzten beiden Segmente Ihres Okta-Einbettungslinks).
- Geben Sie für Okta App Name (Okta-App-Name) den Wert **`amazon_aws_redshift`** ein.



7. Wählen Sie OK.
8. Wählen Sie Test, um die Verbindung zu testen, oder OK, um den Vorgang abzuschließen.

Konfigurieren von Single Sign-On mit ODBC, SAML 2.0 und dem Okta-Identitätsanbieter

Um eine Verbindung zu Datenquellen herzustellen, können Sie Amazon Athena mit Identitätsanbietern (IDPs) wie PingOne, Okta, OneLogin und anderen verwenden. Beginnend mit Athena-ODBC-Treiberversion 1.1.13 und Athena-JDBC-Treiberversion 2.0.25 ist ein Browser SAML-Plug-In enthalten, das Sie für die Arbeit mit jedem SAML-2.0-Anbieter konfigurieren können. In diesem Thema erfahren Sie, wie Sie den Amazon-Athena-ODBC-Treiber und das browserbasierte SAML-Plug-In so konfigurieren, dass mithilfe des Okta-Identitätsanbieters Funktionen für Single Sign-On (SSO) hinzugefügt werden.

Voraussetzungen

Für das Ausfüllen der Schritte in diesem Tutorial benötigen Sie Folgendes:

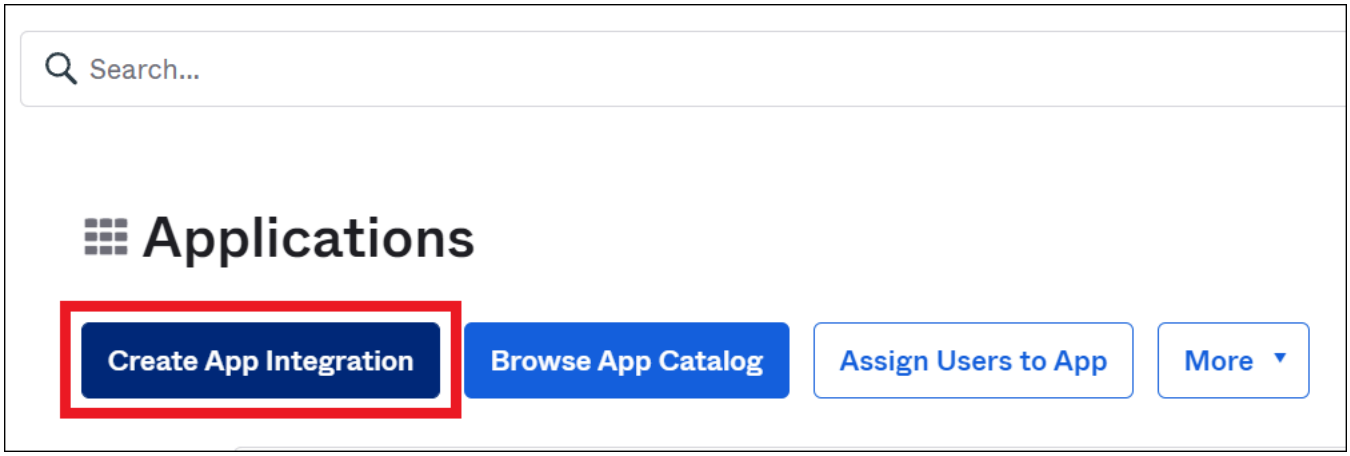
- Athena-ODBC-Treiber, Version 1.1.13 oder höher. Die Versionen 1.1.13 und höher enthalten Browser-SAML-Unterstützung. Download-Links finden Sie unter Verbindung zu Amazon Athena mit ODBC.
- Eine IAM-Rolle, die Sie mit SAML verwenden möchten. Weitere Informationen finden Sie unter Erstellen einer Rolle für SAML-2.0-basierten Verbund im IAM-Benutzerhandbuch.
- Ein Okta-Konto. Weitere Informationen finden Sie unter okta.com.

Erstellen einer App-Integration in Okta

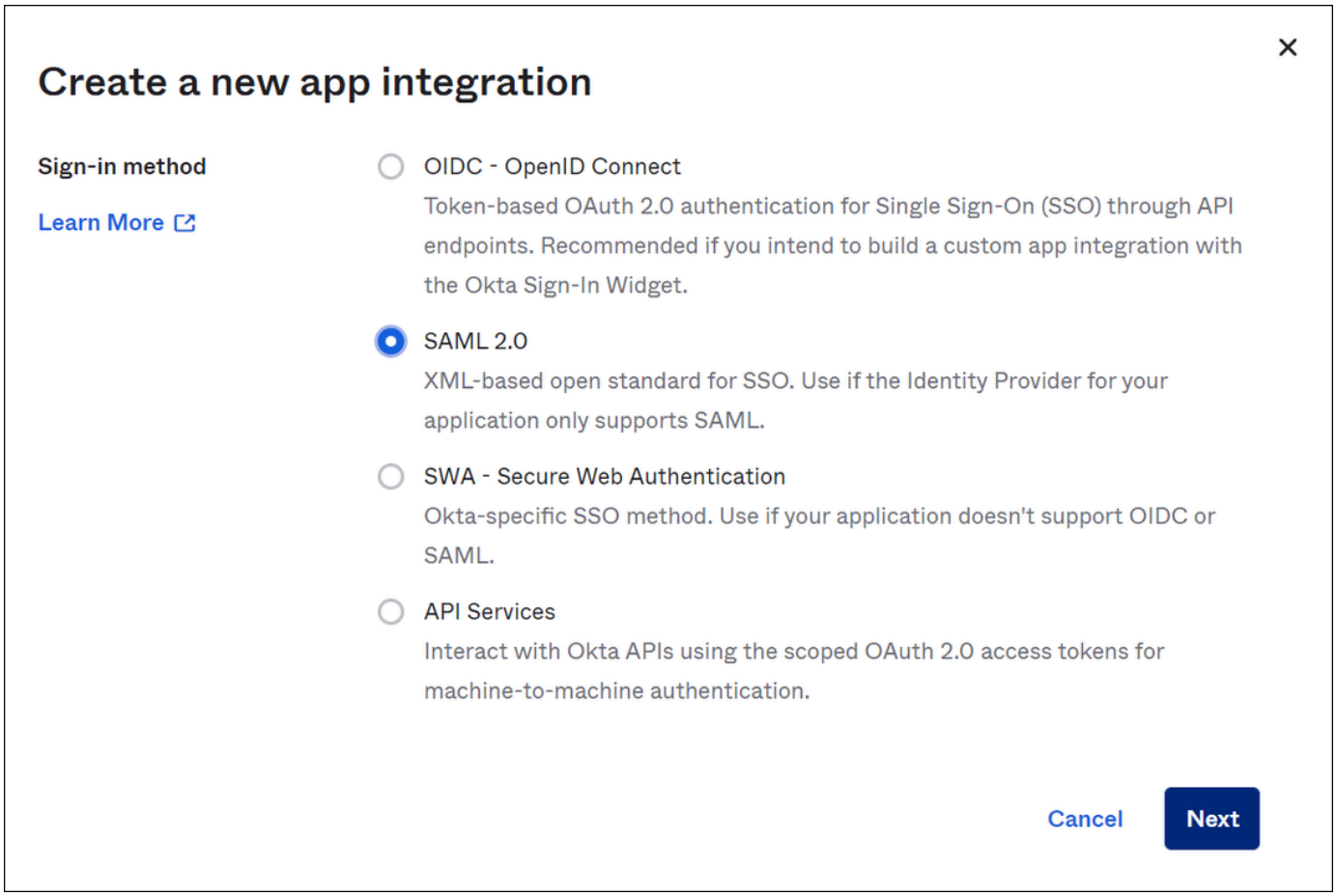
Verwenden Sie zunächst das Okta-Dashboard, um eine SAML-2.0-App für Single Sign-On bei Athena zu erstellen und zu konfigurieren.

So richten Sie mit dem Okta-Dashboard Single Sign-On für Athena ein

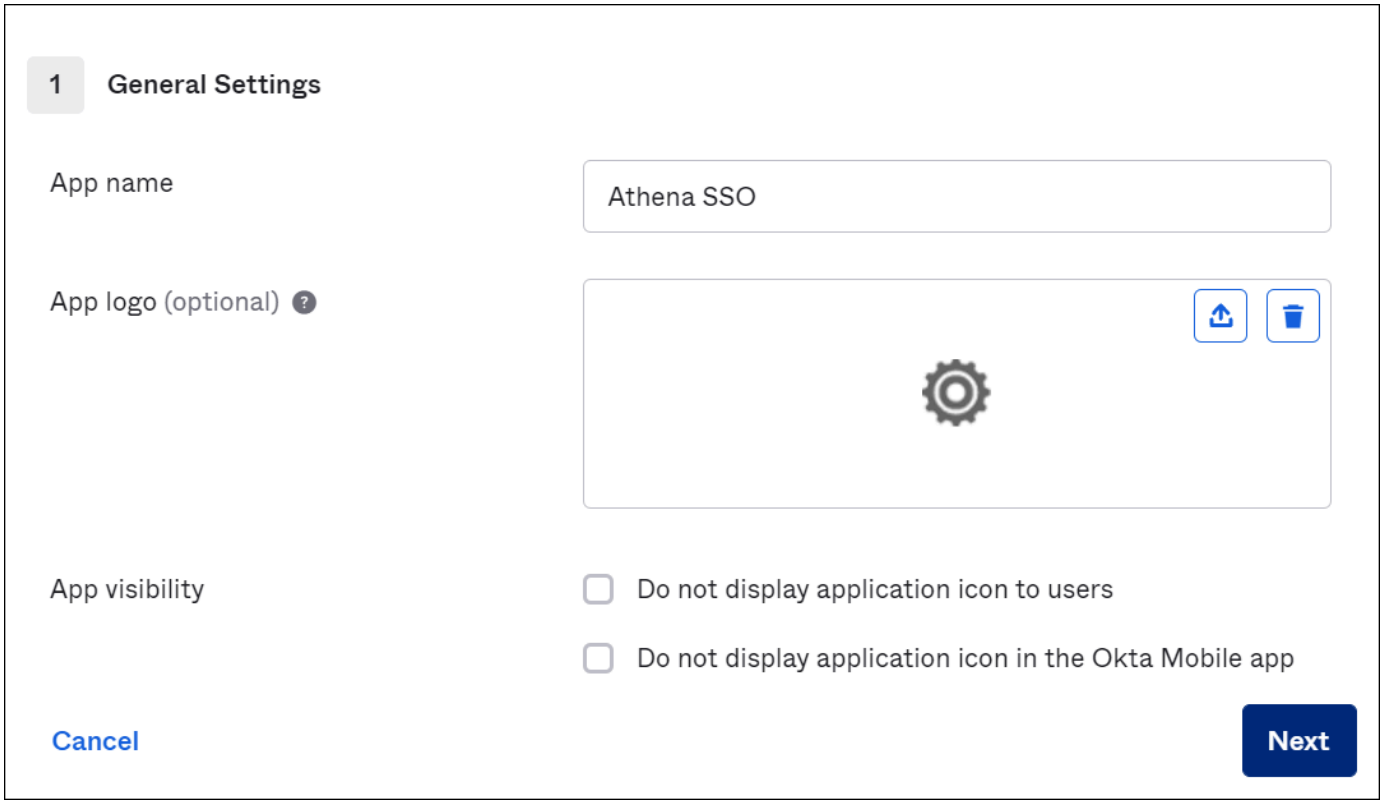
1. Melden Sie sich auf der Okta-Admin-Seite in `okta.com` an.
2. Wählen Sie die Anwendung im Navigationsbereich unter Applications (Anwendungen) Applications (Anwendungen) aus.
3. Wählen Sie auf der Seite Applications (Anwendungen) die Option Create App Integration (App-Integration erstellen) aus.



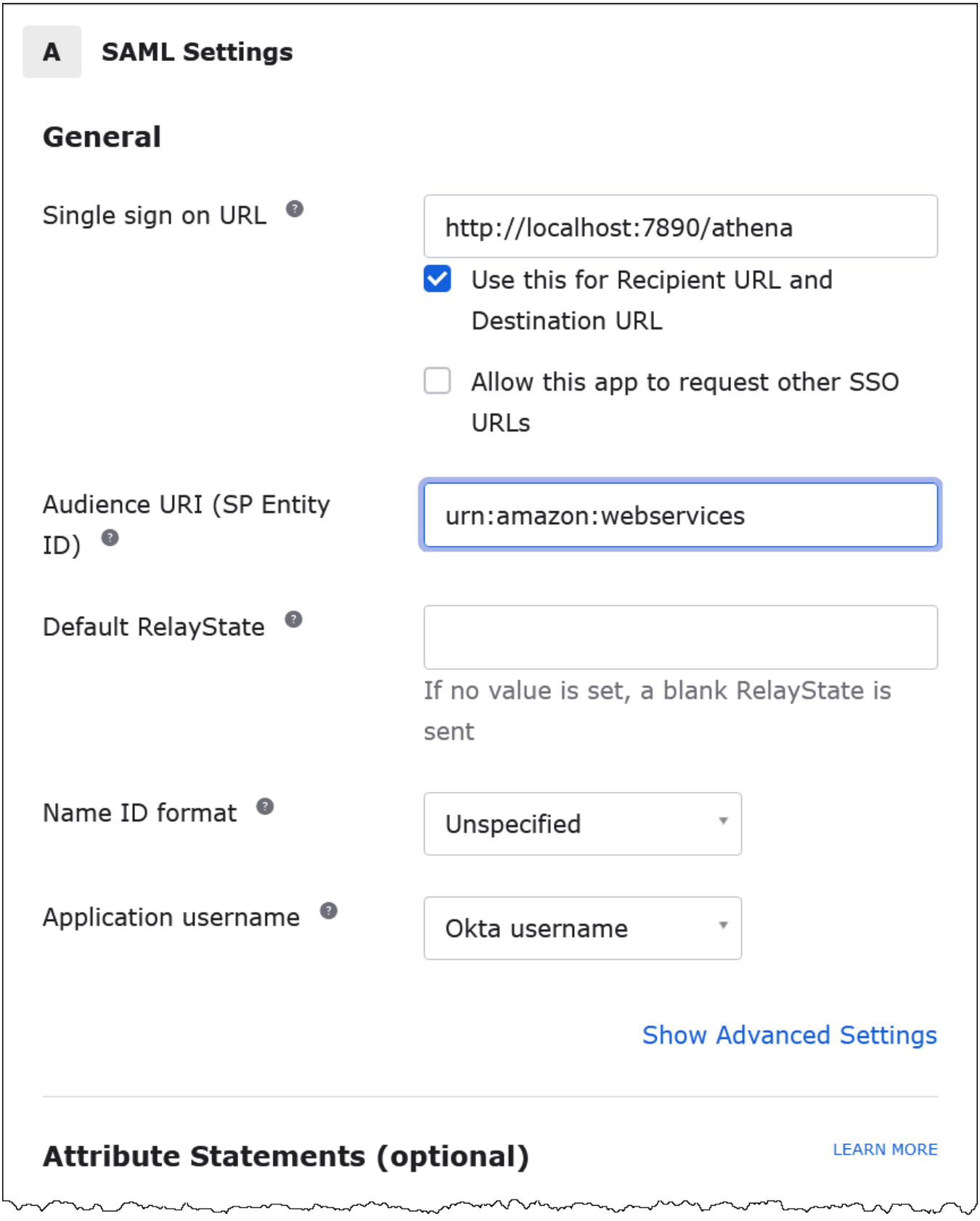
4. Wählen Sie im Dialogfeld Create a new app integration (Erstellen Sie eine neue App-Integration) für Sign-in method (Anmelde-Methode) SAML 2.0 aus und danach wählen Sie Next (Weiter).



5. Geben Sie auf der Seite SAML-Integration erstellen im Abschnitt Allgemeine Einstellungen einen Namen für die Anwendung ein. In diesem Tutorial wird der Name Athena SSO verwendet.



6. Wählen Sie Next (Weiter).
7. Geben Sie auf der Seite SAML konfigurieren im Abschnitt SAML-Einstellungen die folgenden Werte ein:
   - Geben Sie für Single-Sign-On-URL **`http://localhost:7890/athena`** ein
   - Geben Sie für Zielgruppen-URI **`urn:amazon:webservices`** ein


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. Geben Sie für Attributanweisungen (optional) die folgenden beiden Name/Wert-Paare ein. Dies sind erforderliche Mapping-Attribute.

   - Geben Sie in das Feld Name die folgende URL ein:

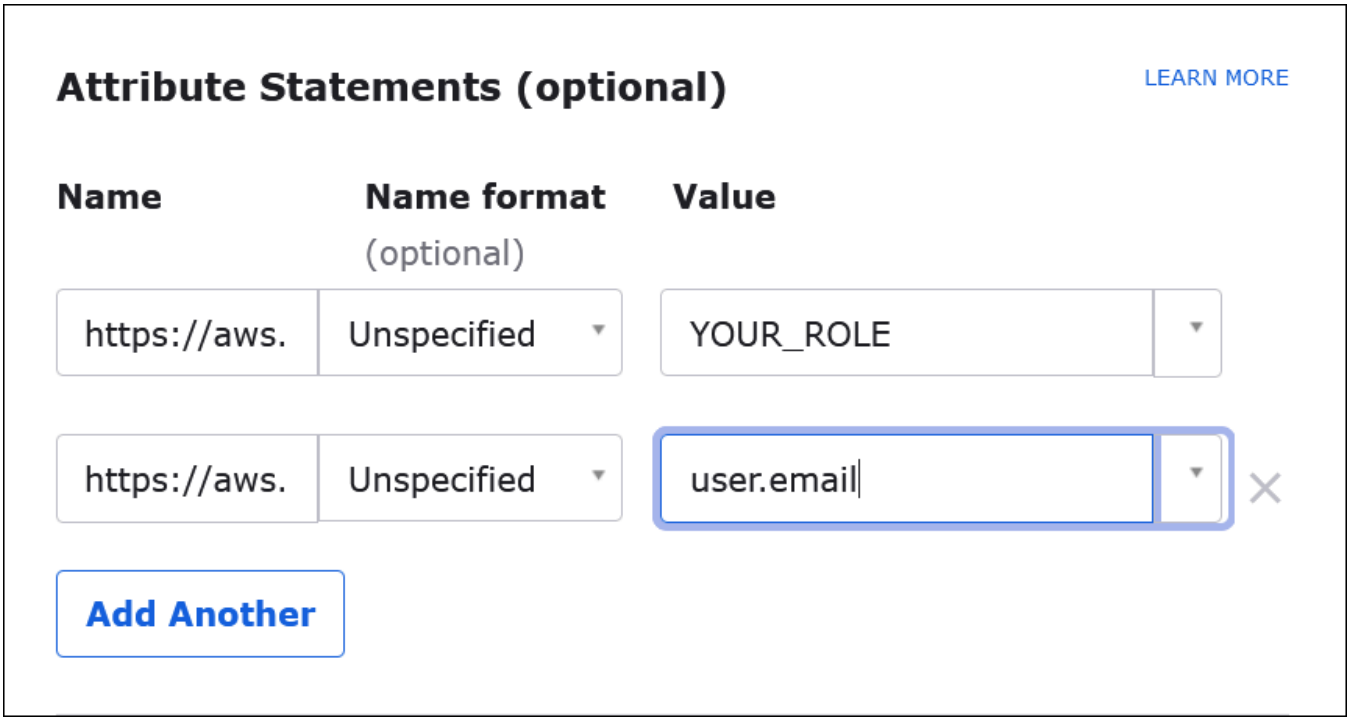
     `https://aws.amazon.com/SAML/Attributes/Role`

     Geben Sie für Wert den Namen Ihrer IAM-Rolle ein. Informationen zum IAM-Rollenformat finden Sie unter Konfigurieren von SAML-Assertions für die Authentifizierungsantwort im IAM-Benutzerhandbuch.

   - Geben Sie in das Feld Name die folgende URL ein:

     `https://aws.amazon.com/SAML/Attributes/RoleSessionName`

     Geben Sie für Wert **`user.email`** ein.



9. Wählen Sie Next (Weiter) und danach Finish (Beenden).

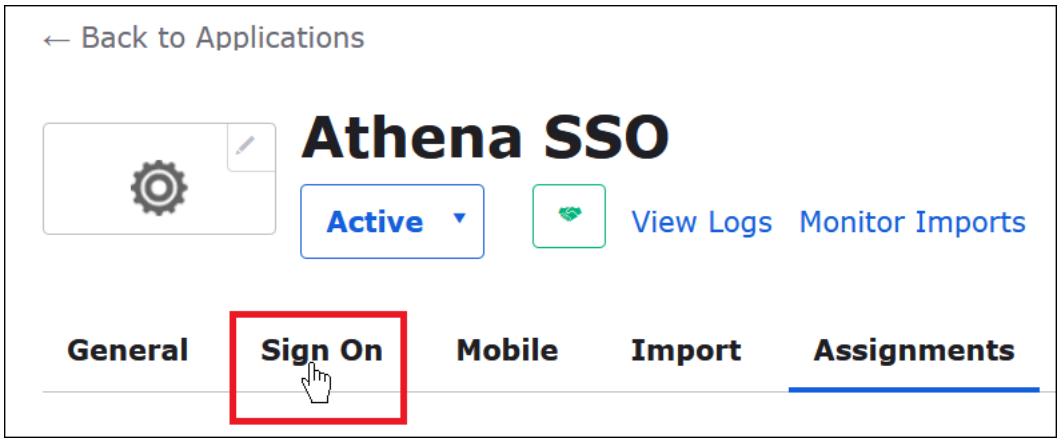
   Wenn Okta die Anwendung erstellt, wird auch Ihre Anmelde-URL erstellt, die Sie als nächstes abrufen werden.

Abrufen der Anmelde-URL aus dem Okta-Dashboard

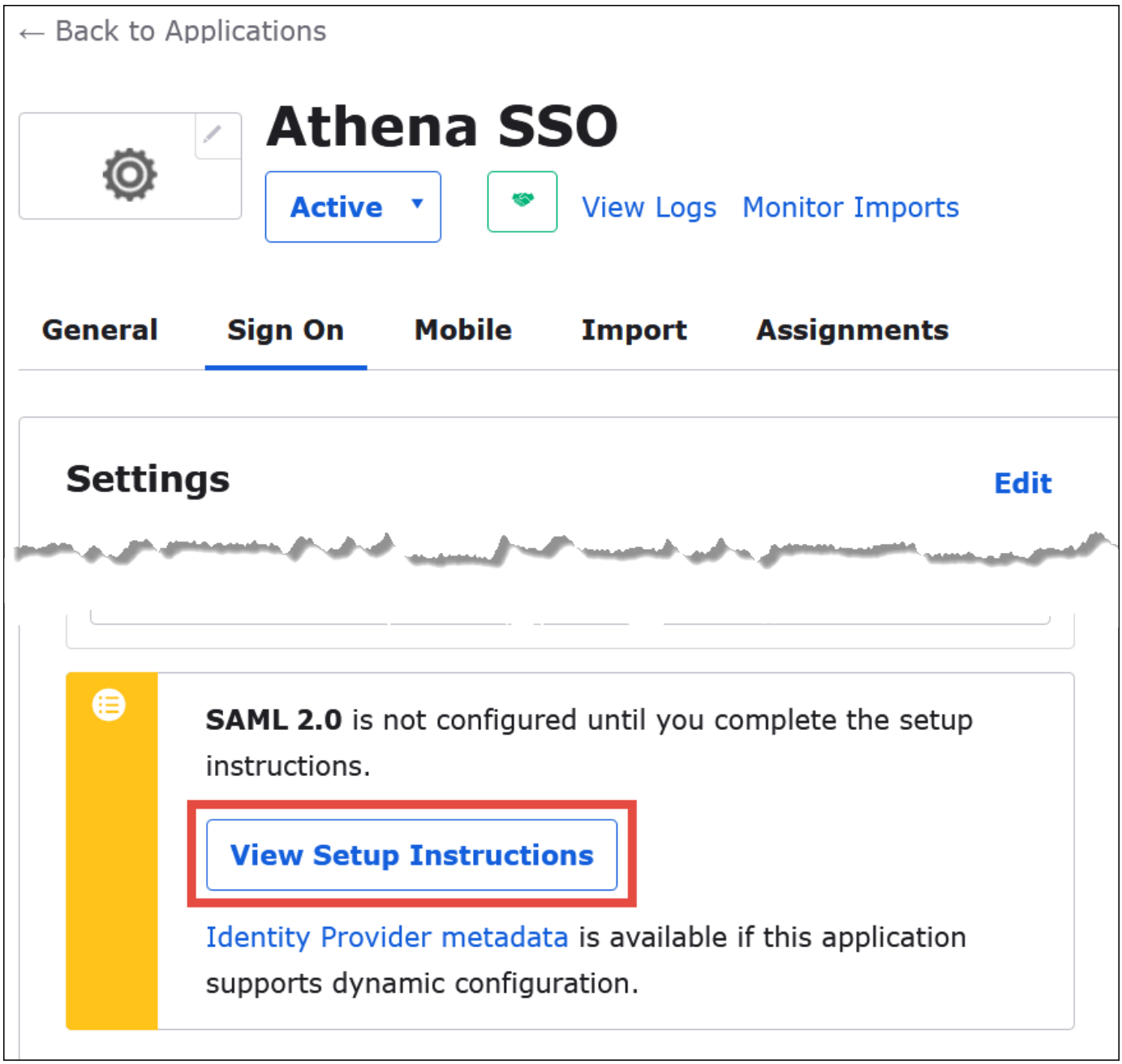
Nachdem Ihre Anwendung erstellt wurde, können Sie ihre Anmelde-URL und andere Metadaten über das Okta-Dashboard abrufen.

Abrufen der Anmelde-URL aus dem Okta-Dashboard

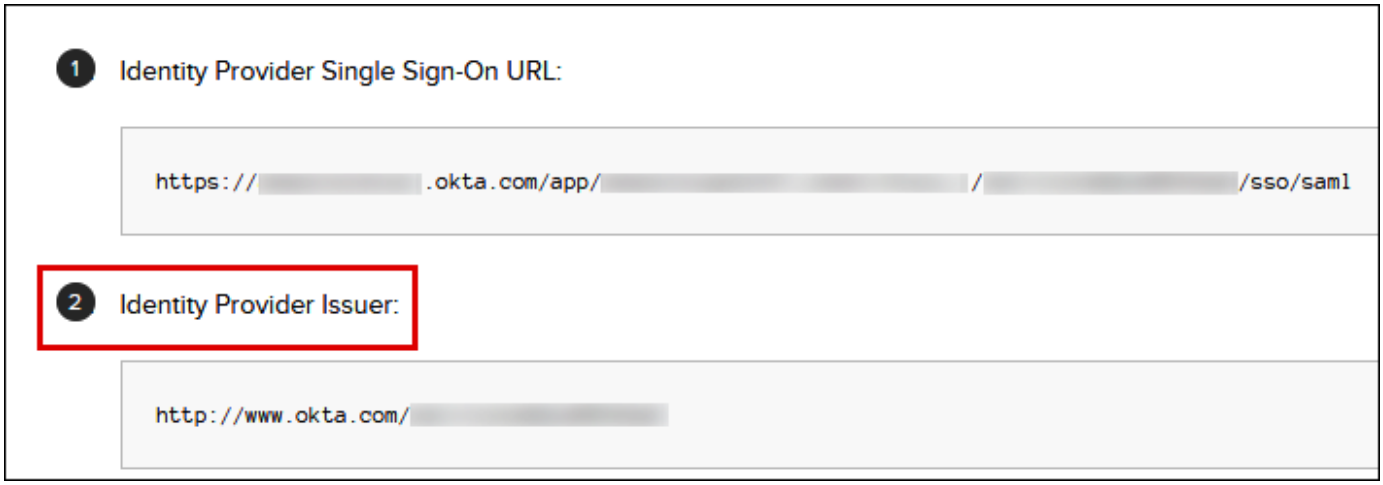
1. Wählen Sie die Anwendung im Okta-Navigationsbereich unter Applications (Anwendungen), Applications (Anwendungen) aus.
2. Wählen Sie die Anwendung aus, deren Anmelde-URL Sie finden möchten (z. B. AthenaSSO).
3. Wählen Sie auf der Seite für Ihre Anwendung Anmelden aus.



4. Wählen Sie View Setup Instructions (Einrichtungsanweisungen anzeigen) aus.



5. Suchen Sie auf der Seite So konfigurieren Sie SAML 2.0 für Athena SSO nach der URL Identitätsanbieter-Aussteller. Einige Stellen im Okta-Dashboard beziehen sich auf diese URL als SAML-Aussteller-ID.



6. Kopieren oder speichern Sie den Wert für URL des Identitätsanbieters Single Sign-On.

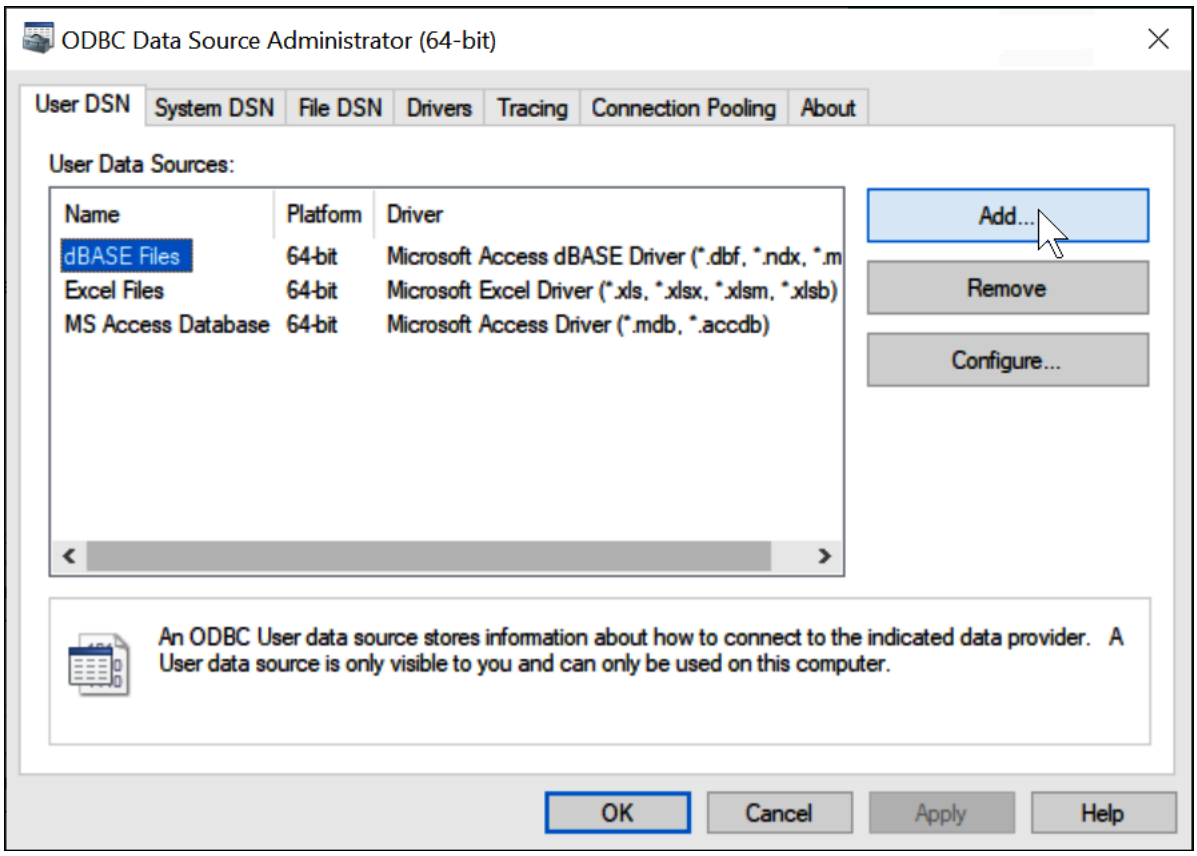
   Wenn Sie im nächsten Abschnitt die ODBC-Verbindung konfigurieren, geben Sie diesen Wert als Anmeldungs-URL-Verbindungsparameter für das SAML-Plug-In des Browsers.

Konfigurieren der Browser-SAML-ODBC-Verbindung zu Athena

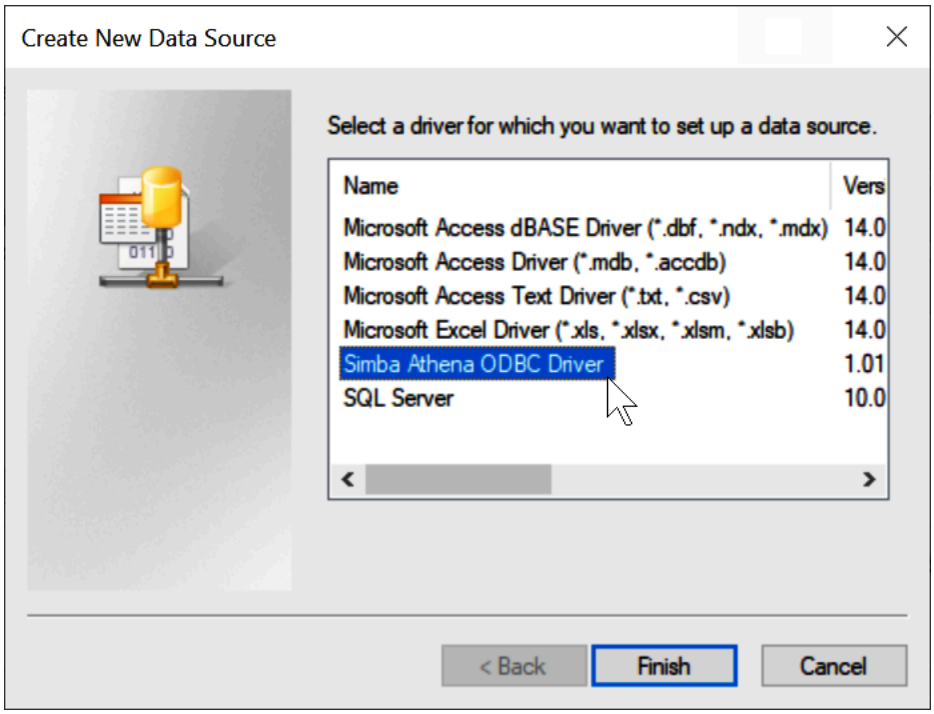
Jetzt können Sie die SAML-Verbindung des Browsers zu Athena mithilfe des ODBC-Datenquellenprogramms in Windows konfigurieren.

Konfigurieren der Browser-SAML-ODBC-Verbindung zu Athena

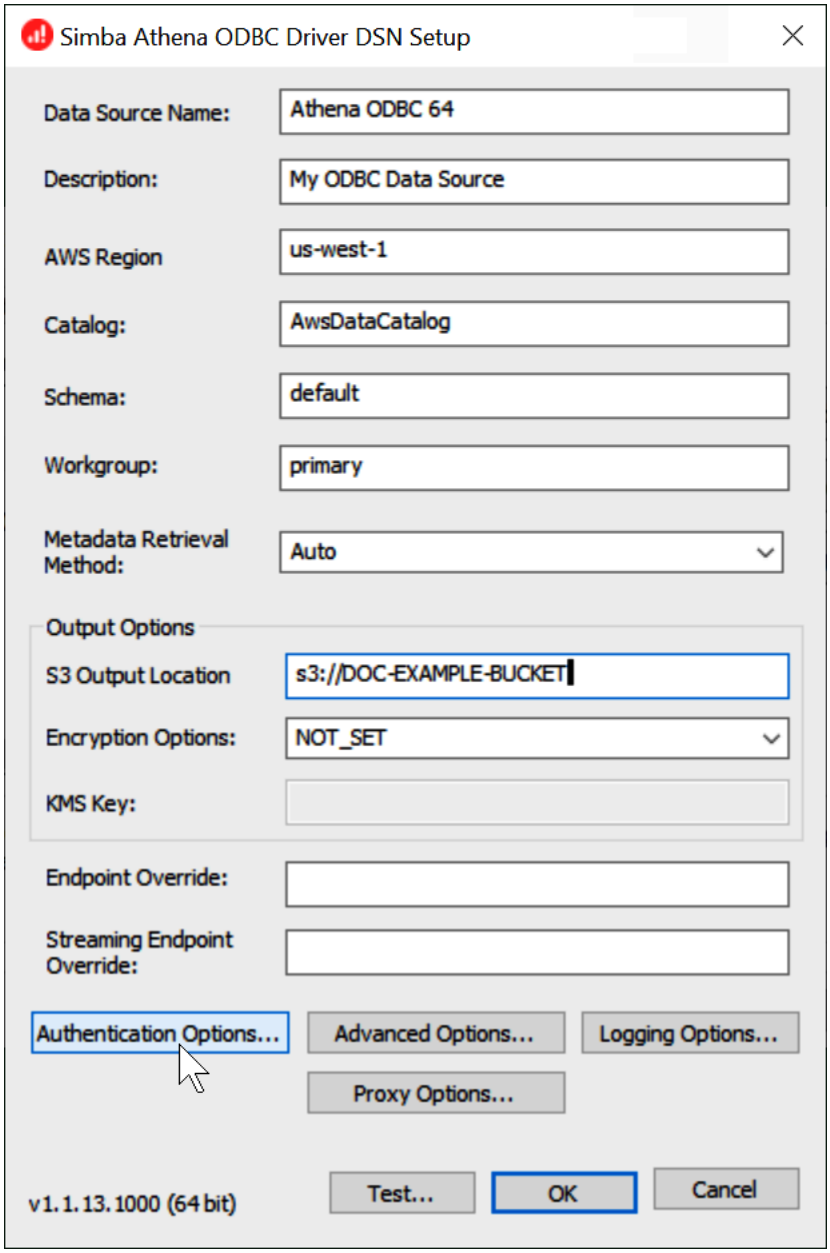
1. Starten Sie unter Windows das ODBC-Datenquellen-Programm.
2. Wählen Sie in ODBC-Datenquellen-Administrator Hinzufügen (Add) aus.



3. Wählen Sie den Simba-Athena-ODBC-Treiber und dann Beenden (Finish) aus.



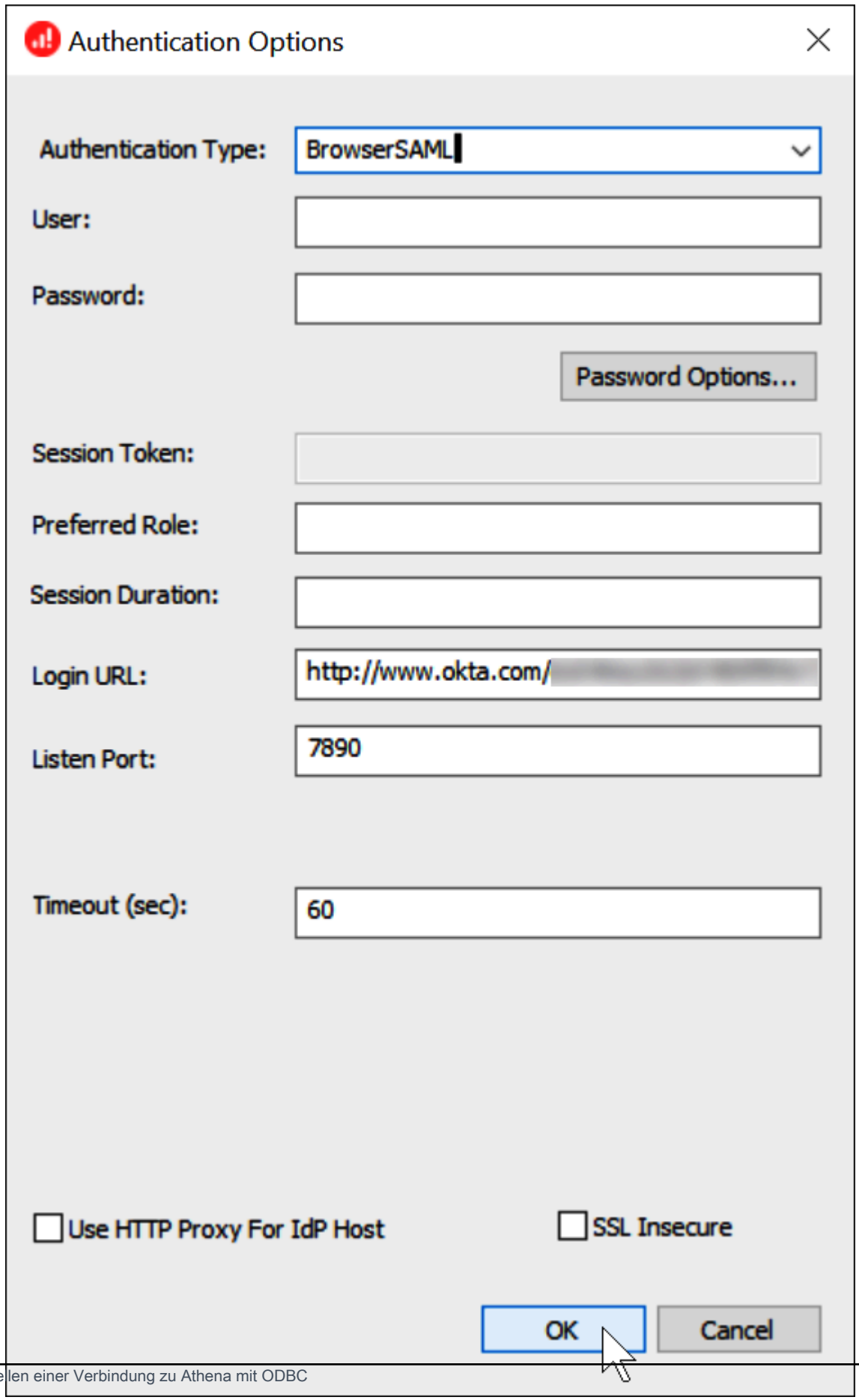
4. Geben Sie in SSN-Einrichtung für Simba-Athena-ODBC-Treiber die beschriebenen Werte ein.



- Geben Sie in Data Source Name (Datenquellenname) einen Namen für Ihre Datenquelle ein (z. B. Athena ODBC 64).
- Geben Sie in das Feld Bezeichnung (Description) eine Beschreibung für Ihre Datenquelle ein.
- Geben Sie für die AWS-Region die AWS-Region, die Sie verwenden (z. B. **`us-west-1`**) ein.
- Geben Sie für S3-Ausgabespeicherort den Amazon-S3-Pfad ein, in dem Ihre Ausgabe gespeichert werden soll.

5. Wählen Sie Authentifizierungsoptionen aus.

6. Wählen Sie im Dialogfeld Authentifizierungsoptionen die folgenden Werte aus oder geben Sie sie ein.


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

- Wählen Sie für Authentifizierungstyp BrowserSAML aus.
- Geben Sie für Anmeldungs-URL die Single-Sign-On-URL des Identitätsanbieters ein, die Sie vom Okta-Dashboard abgerufen haben.
- Geben Sie unter Listen-Port 7890 ein.
- Geben Sie für Timeout (Sek) einen Wert für einen Verbindungs-Timeout in Sekunden ein.

7. Wählen Sie OK, um die Authentication Options (Authentifizierungsoptionen) zu schließen.
8. Wählen Sie Test, um die Verbindung zu testen, oder OK, um den Vorgang abzuschließen.

Verwenden des Amazon-Athena-Power-BI-Connectors

Auf Windows-Betriebssystemen können Sie den Microsoft-Power-BI-Connector für Amazon Athena verwenden, um Daten von Amazon Athena in Microsoft-Power-BI-Desktop zu analysieren. Informationen zu Power BI finden Sie unter Microsoft power BI. Nachdem Sie Inhalte im Power-BI-Service veröffentlicht haben, können Sie die Version von Power-BI-Gateway vom Juli 2021 oder höher verwenden, um die Inhalte durch on-demand oder geplante Aktualisierungen auf dem neuesten Stand zu halten.

Voraussetzungen

Stellen Sie vor Beginn sicher, dass Ihre Umgebung die folgenden Anforderungen erfüllt. Der Amazon-Athena-ODBC-Treiber ist erforderlich.

- AWS-Konto
- Berechtigungen zur Nutzung von Athena
- Amazon-Athena-ODBC-Treiber
- Power-BI-Desktop

Unterstützte Funktionen

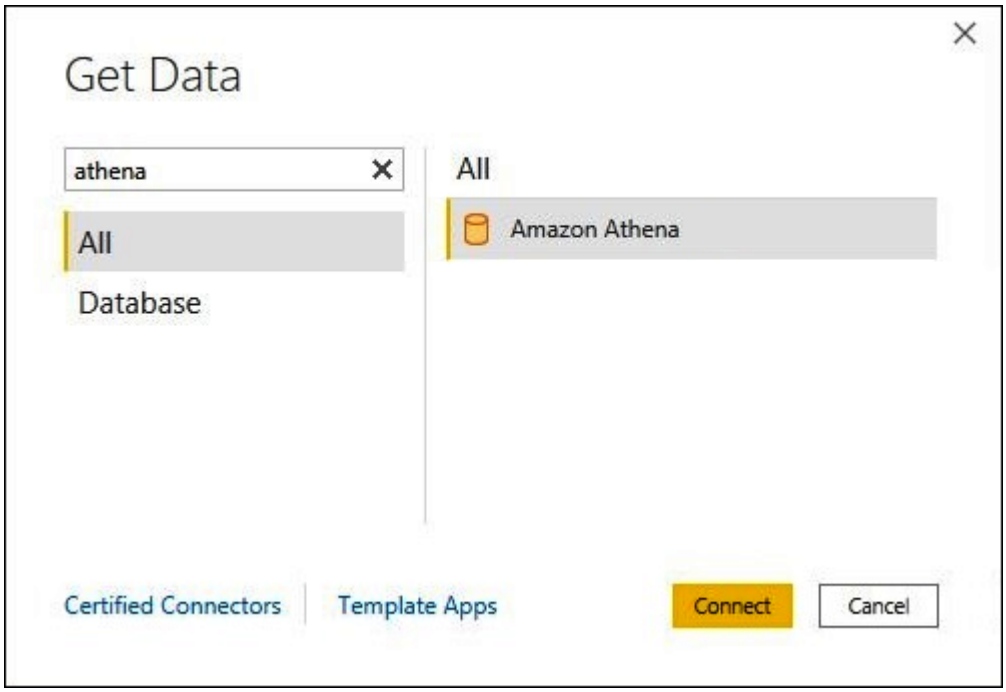
- Import – Ausgewählte Tabellen und Spalten werden zur Abfrage in Power-BI-Desktop importiert.
- DirectQuery – Es werden keine Daten importiert oder in Power-BI-Desktop kopiert. Power-BI-Desktop fragt die zugrunde liegende Datenquelle direkt ab.
- Power-BI-Gateway – Ein On-Premises-Daten-Gateway in Ihrem AWS-Konto, die wie eine Brücke zwischen dem Microsoft-Power-BI-Service und Athena funktioniert. Das Gateway ist erforderlich, um Ihre Daten im Microsoft-Power-BI-Service anzuzeigen.

Mit Amazon Athena verbinden

Führen Sie folgende Schritte aus, um Power-BI-Desktop mit Ihren Amazon-Athena-Daten zu verbinden.

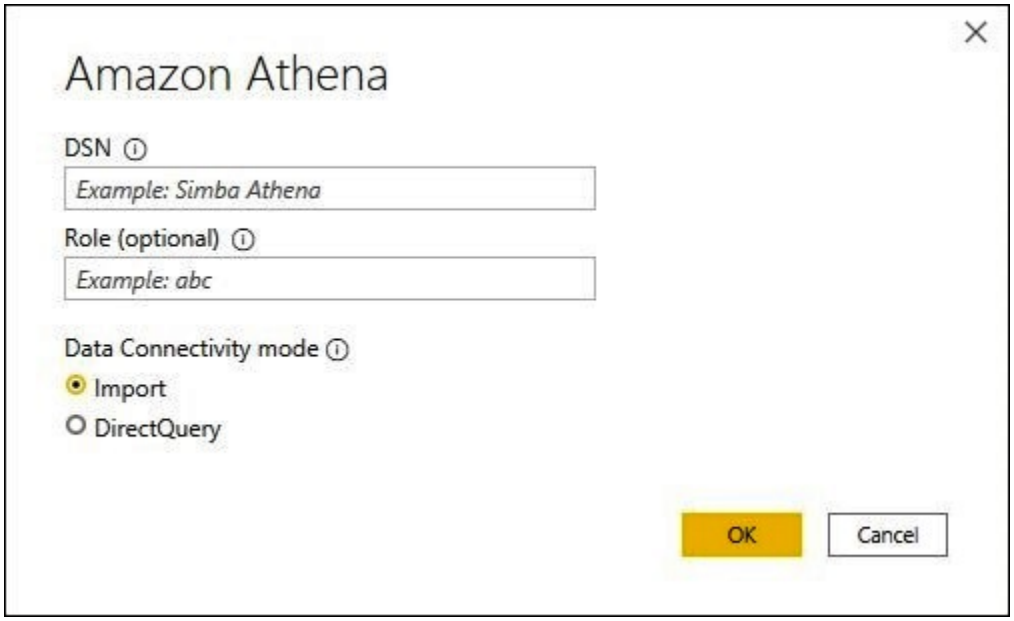
So stellen Sie eine Verbindung mit Athena-Daten über Power-BI-Desktop her

1. Starten Sie Power-BI-Desktop.
2. Führen Sie eine der folgenden Aktionen aus:
   - Wählen Sie File (Datei), Get Data (Abrufen von Daten)
   - Wählen Sie im Menüband Start die Option Daten abrufen aus.
3. Geben Sie in das Suchfeld Athena ein.
4. Wählen Sie Amazon Athena und dann Verbinden aus.

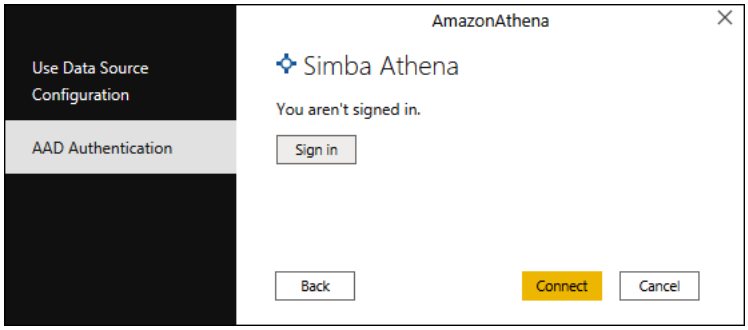


5. Geben Sie auf der Verbindungsseite von Amazon Athena die folgenden Informationen ein.
   - Geben Sie für DSN den Namen des ODBC-DSN ein, den Sie verwenden möchten. Anweisungen zum Konfigurieren Ihres DSN finden Sie in der ODBC-Treiberdokumentation.
   - Wählen Sie für den Datenkonnektivitätsmodus einen Modus aus, der für Ihren Anwendungsfall geeignet ist, und befolgen Sie diese allgemeinen Richtlinien:

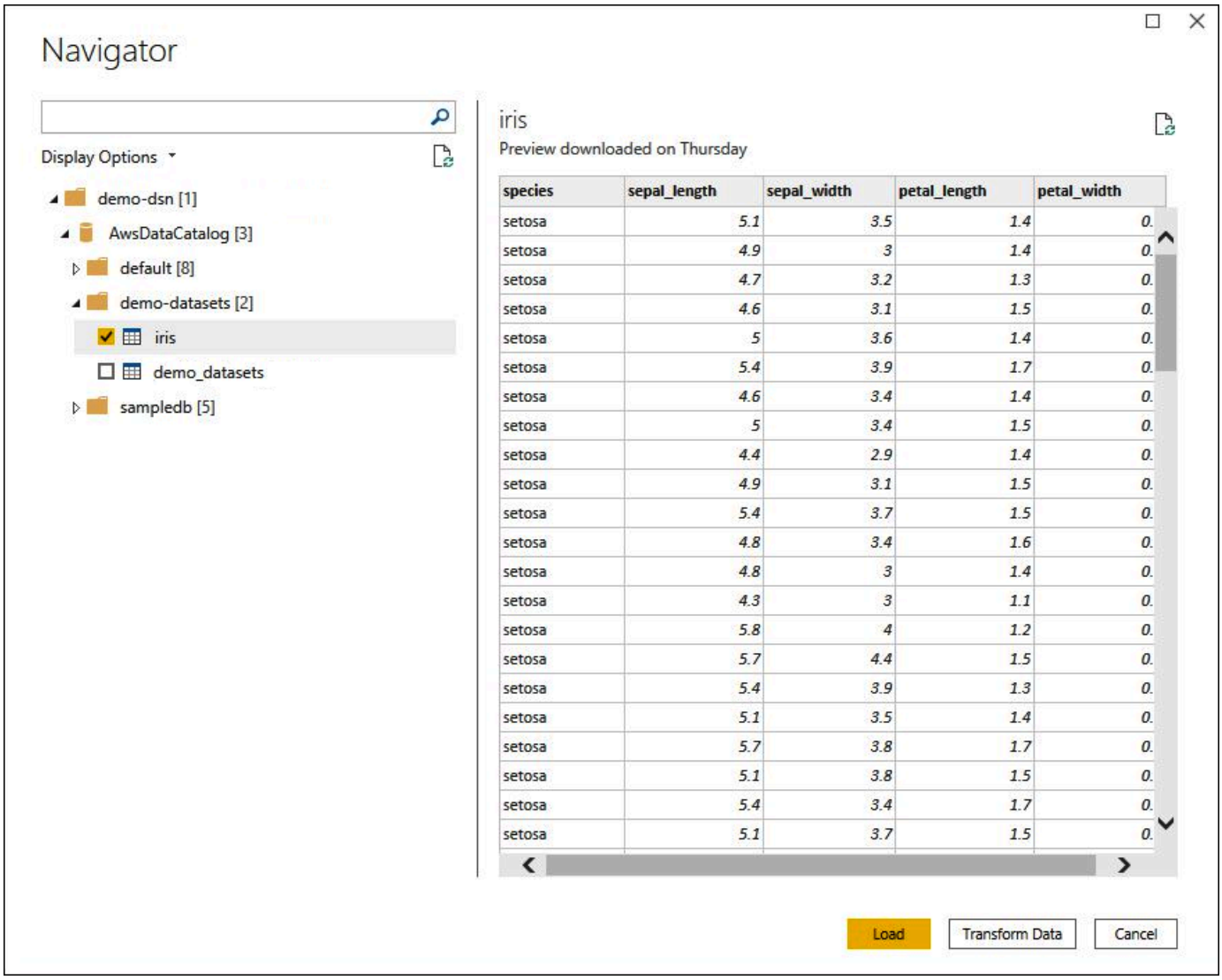
- Wählen Sie für kleinere Datensätze Importieren. Wenn Sie den Importmodus verwenden, arbeitet Power BI mit Athena zusammen, um den Inhalt des gesamten Datensatzes für die Verwendung in Ihren Visualisierungen zu importieren.
- Wählen Sie für größere Datensätze DirectQuery aus. Im DirectQuery-Modus werden keine Daten auf Ihre Arbeitsstation heruntergeladen. Während Sie eine Visualisierung erstellen oder mit ihr interagieren, arbeitet Microsoft Power BI mit Athena zusammen, um die zugrunde liegende Datenquelle dynamisch abzufragen, sodass Sie immer die aktuellen Daten anzeigen. Weitere Informationen zu DirectQuery finden Sie unter Verwenden von DirectQuery in Power-BI-Desktop in der Microsoft-Dokumentation.



6. Wählen Sie OK.
7. Wählen Sie bei der Aufforderung zum Konfigurieren der Datenquellenauthentifizierung entweder Datenquellenkonfiguration verwenden oder AAD-Authentifizierung aus und wählen Sie dann Verbinden aus.

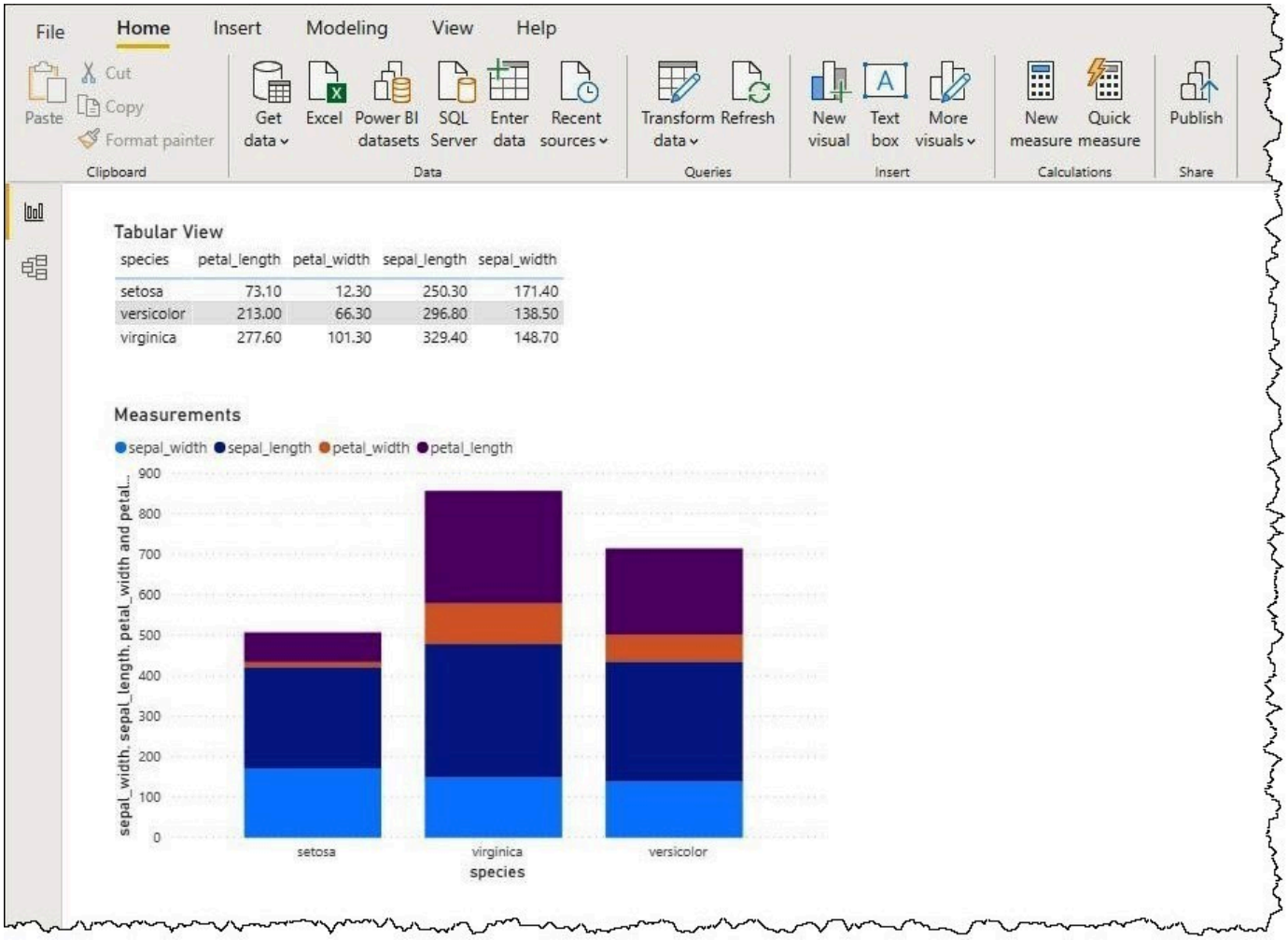


Ihr Datenkatalog, Ihre Datenbanken und Tabellen werden im Dialogfeld Navigator angezeigt.



species	sepal_length	sepal_width	petal_length	petal_width
setosa	5.1	3.5	1.4	0.
setosa	4.9	3	1.4	0.
setosa	4.7	3.2	1.3	0.
setosa	4.6	3.1	1.5	0.
setosa	5	3.6	1.4	0.
setosa	5.4	3.9	1.7	0.
setosa	4.6	3.4	1.4	0.
setosa	5	3.4	1.5	0.
setosa	4.4	2.9	1.4	0.
setosa	4.9	3.1	1.5	0.
setosa	5.4	3.7	1.5	0.
setosa	4.8	3.4	1.6	0.
setosa	4.8	3	1.4	0.
setosa	4.3	3	1.1	0.
setosa	5.8	4	1.2	0.
setosa	5.7	4.4	1.5	0.
setosa	5.4	3.9	1.3	0.
setosa	5.1	3.5	1.4	0.
setosa	5.7	3.8	1.7	0.
setosa	5.1	3.8	1.5	0.
setosa	5.4	3.4	1.7	0.
setosa	5.1	3.7	1.5	0.

8. Aktivieren Sie im Bereich Anzeigeoptionen das Kontrollkästchen für den Datensatz, den Sie verwenden möchten.
9. Wenn Sie den Datensatz vor dem Importieren transformieren möchten, gehen Sie zum unteren Rand des Dialogfelds und wählen Sie Daten transformieren. Dadurch wird der Power-Query-Editor geöffnet, sodass Sie den Datensatz, den Sie verwenden möchten, filtern und verfeinern können.
10. Wählen Sie Load (Laden) aus. Nachdem der Ladevorgang abgeschlossen ist, können Sie Visualisierungen wie im folgenden Image erstellen. Wenn Sie DirectQuery als Importmodus ausgewählt haben, gibt Power BI eine Abfrage an Athena für die angeforderte Visualisierung aus.



Einrichten eines On-Premises-Gateways

Sie können Dashboards und Datensätze im Power-BI-Service veröffentlichen, damit andere Benutzer über Web-, mobile und eingebettete Apps mit ihnen interagieren können. Um Ihre Daten im Microsoft-Power-BI-Service anzuzeigen, installieren Sie das On-Premises-Daten-Gateway von Microsoft Power BI in Ihrem AWS-Konto. Das Gateway funktioniert wie eine Brücke zwischen dem Microsoft-Power-BI-Service und Athena.

So laden Sie ein On-Premises-Daten-Gateways herunter, installieren und testen Sie es

1. Besuchen Sie die Download-Seite für das Microsoft-Power-BI-Gateway und wählen Sie entweder den persönlichen Modus oder den Standardmodus aus. Der persönliche Modus ist nützlich, um den Athena-Connector lokal zu testen. Der Standardmodus ist in einer Mehrbenutzerproduktionseinstellung geeignet.

2. Informationen zum Installieren eines On-Premises-Gateways (im persönlichen oder Standardmodus) finden Sie unter Installieren eines On-Premises-Daten-Gateways in der Microsoft-Dokumentation.
3. Führen Sie zum Testen des Gateways die Schritte unter Verwenden von benutzerdefinierten Daten-Connector mit dem On-Premises-Daten-Gateway in der Microsoft-Dokumentation aus.

Weitere Informationen zu On-Premises-Daten-Gateways finden Sie in den folgenden Microsoft-Ressourcen.

- Was ist ein On-Premises-Daten-Gateway?
- Anleitung zum Bereitstellen eines Daten-Gateways für Power BI

Ein Beispiel für die Konfiguration von Power-BI-Gateway für die Verwendung mit Athena finden Sie im AWS-Big-Data-Blogbeitrag Dashboards schnell in Microsoft Power BI mit Amazon Athena erstellen.

Erstellen von Datenbanken und Tabellen

Amazon Athena unterstützt ein Subset an DDL-(Data-Definition-Language)-Anweisungen sowie ANSI-SQL-Funktionen und -Operatoren, um externe Tabellen, deren Daten in Amazon Simple Storage gespeichert sind, zu definieren und abzufragen.

Wenn Sie eine Datenbank und eine Tabelle in Athena erstellen, beschreiben Sie das Schema und den Speicherort der Daten, damit Sie die darin enthaltenen Daten für Echtzeitabfragen nutzen können.

Zur Optimierung der Abfrageleistung und zur Kostensenkung wird empfohlen, die Daten zu partitionieren und spaltenbasierte Open-Source-Formate wie Apache Parquet oder ORC zum Speichern in Amazon S3 zu verwenden.

Themen

- Spaltenbasierte Speicherformate
- Konvertieren in spaltenbasierte Formate
- Daten in Athena partitionieren
- Partitionsprojektion mit Amazon Athena

Erstellen von Datenbanken in Athena

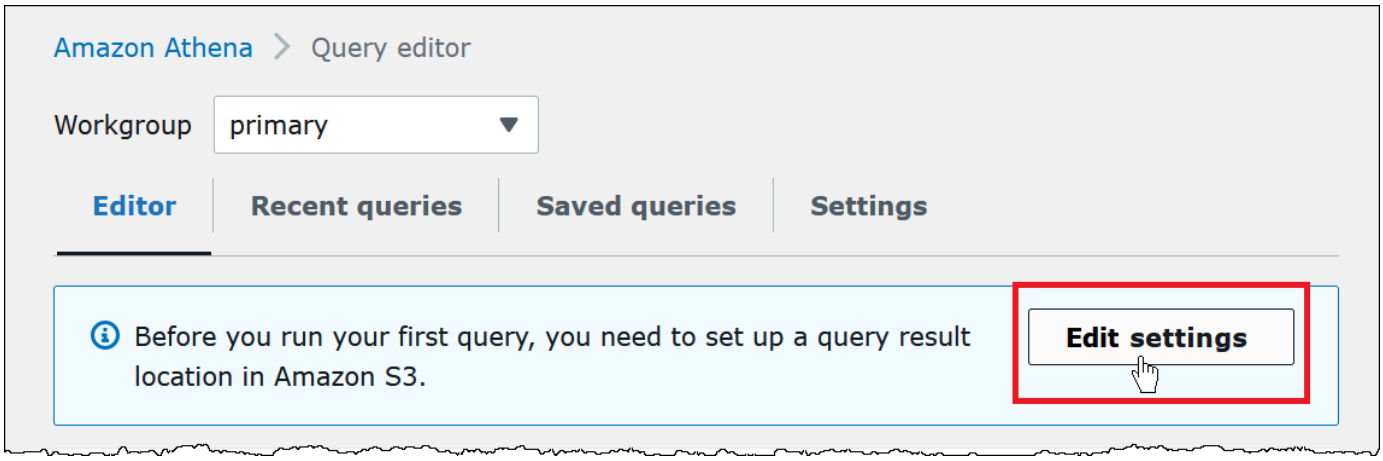
Eine Datenbank in Athena ist eine logische Gruppierung für Tabellen, die Sie darin erstellen.

Voraussetzungen

Wenn Sie noch keinen Abfrageausgabespeicherort in Amazon S3 eingerichtet haben, führen Sie dazu die folgenden erforderlichen Schritte aus.

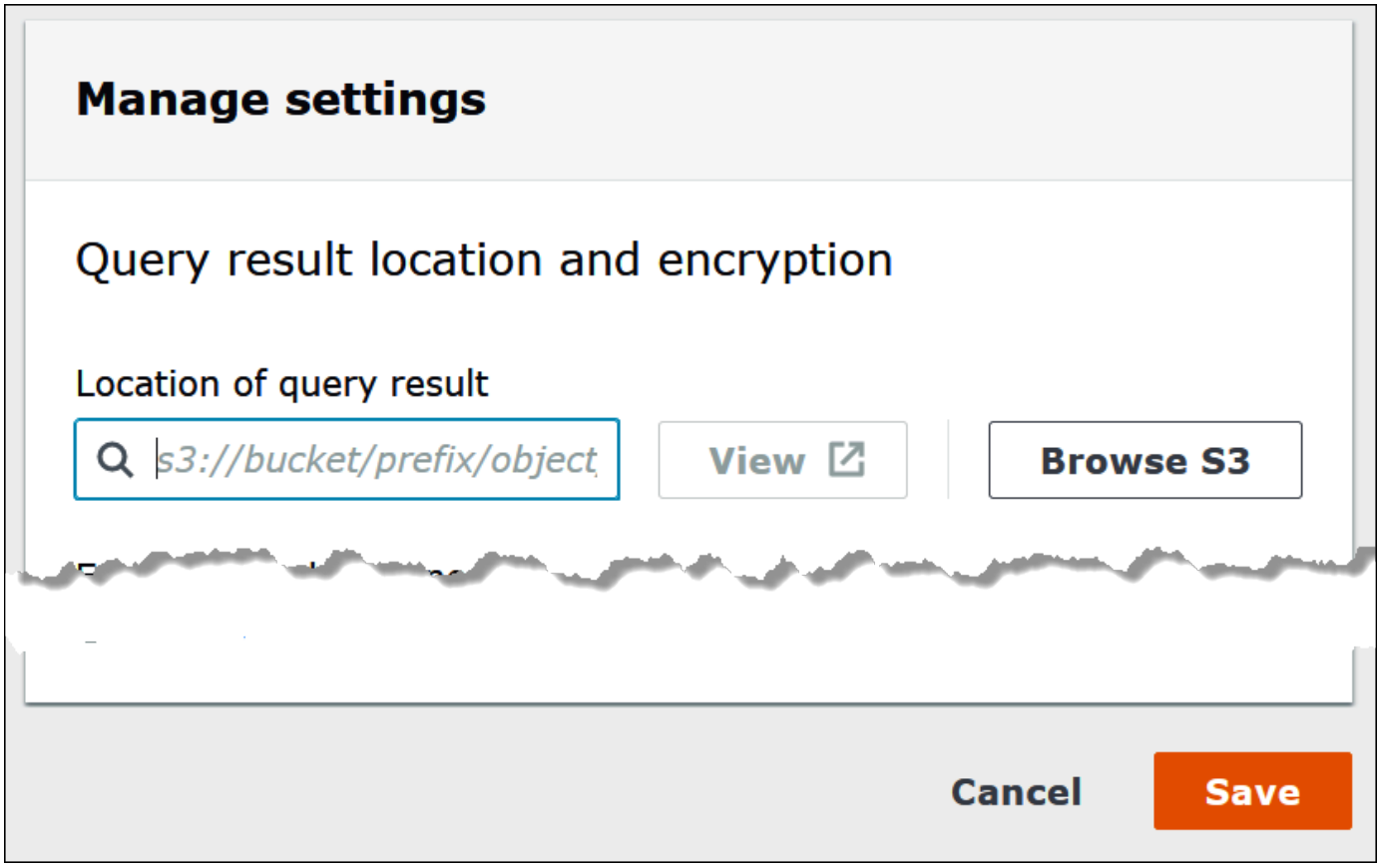
Wie Sie einen Abfrageausgabespeicherort erstellen

1. Führen Sie unter Verwendung derselben AWS-Region und desselben Kontos, das Sie für Athena verwenden, die Schritte aus (z. B. mithilfe der Amazon-S3-Konsole), um einen Bucket in Amazon S3 zu erstellen, der Ihre Athena-Abfrageergebnisse enthält. Sie konfigurieren diesen Bucket als Speicherort für die Abfrageausgabe.
2. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
3. Wenn Sie die Athena-Konsole in Ihrer aktuellen AWS-Region zum ersten Mal besuchen, wählen Sie Erkunden des Abfrage-Editors, um den Abfrage-Editor zu öffnen. Andernfalls wird Athena im Abfrage-Editor geöffnet.
4. Wählen Sie Edit Settings (Einstellungen bearbeiten), um einen Speicherort für Abfrageergebnisse in Amazon S3 einzurichten.

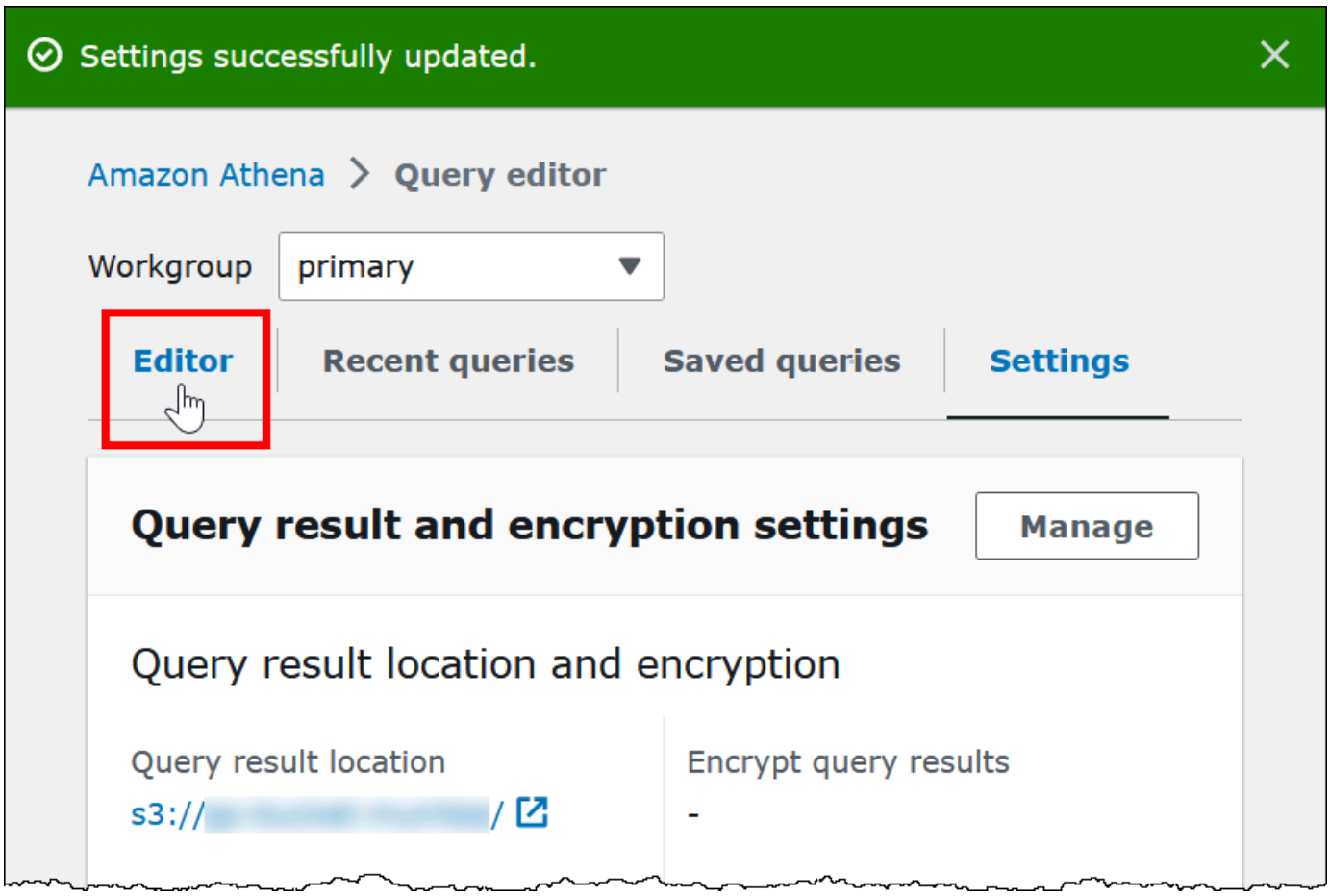


5. Führen Sie für Einstellungen verwalten einen der folgenden Schritte aus:

   - Geben Sie im Feld Speicherort der Abfrageergebnisse den Pfad zu dem Bucket ein, den Sie in Amazon S3 für Ihre Abfrageergebnisse erstellt haben. Stellen Sie dem Pfad einen Präfix mit `s3://` aus.
   - Wählen Sie das Symbol S3 durchsuchen, wählen Sie den Amazon-S3-Bucket aus, den Sie in Ihrer aktuellen Region erstellt haben und wählen Sie dann Auswählen.



6. Wählen Sie Save (Speichern) aus.
7. Wählen Sie Editor, um zum Abfrage-Editor zu wechseln.



Erstellen einer Datenbank

Nachdem Sie einen Speicherort für die Abfrage-Ergebnisse eingerichtet haben, ist die Datenbankerstellung im Abfrage-Editor der Athena-Konsole einfach.

So erstellen Sie eine Datenbank mit dem Athena-Abfrage-Editor

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Geben Sie auf der Registerkarte Abfrage-Editor den Hive Data Definition Language (DDL)-Befehl `CREATE DATABASE myDataBase` ein. Ersetzen Sie *myDatabase* durch den Namen, den Sie verwenden möchten. Einschränkungen für Datenbanknamen finden Sie unter Namen für Tabellen, Datenbanken und Spalten.
3. Wählen Sie Run (Ausführen) oder drücken Sie **Ctrl+ENTER**.
4. Um Ihre Datenbank zur aktuellen Datenbank zu machen, wählen Sie sie im Menü Database (Datenbank) auf der linken Seite des Abfrage-Editors aus.

Weitere Informationen zum Steuern von Berechtigungen für Athena finden Sie unter Differenzierter Zugriff auf Datenbanken und Tabellen in AWS Glue Data Catalog.

Erstellen von Tabellen in Athena

Sie können DDL-Anweisungen in der Athena-Konsole mithilfe eines JDBC- oder ODBC-Treibers oder mit dem Athena-Formular Create table (Tabelle erstellen) ausführen.

Wenn Sie ein neues Tabellenschema in Athena erstellen, speichert Athena das Schema in einem Datenkatalog und verwendet es zum Ausführen von Abfragen.

Athena verwendet einen als schema-on-read bezeichneten Ansatz. Dabei wird ein Schema bei jedem Ausführen einer Abfrage auf Ihre Daten projiziert. Dadurch müssen Daten nicht geladen oder umgewandelt werden.

Athena bearbeitet Ihre Daten in Amazon S3 nicht.

Athena definiert mithilfe von Apache Hive Tabellen und erstellt Datenbanken, die an sich ein logischer Namespace von Tabellen sind.

Wenn Sie eine Datenbank und Tabelle in Athena erstellen, beschreiben Sie einfach das Schema und den Speicherort, an dem die Tabellendaten in Amazon S3 für Abfragen beim Lesen gespeichert sind. Die Begriffe Datenbank und Tabelle haben daher eine etwas abweichende Bedeutung als in traditionellen relationalen Datenbanksystemen, da die Daten nicht zusammen mit der Schemadefinition für die Datenbank und Tabelle gespeichert werden.

Wenn Sie Daten abfragen, tun Sie dies mit Standard-SQL und die Daten werden zum Zeitpunkt der Abfrage gelesen. Eine Anleitung zum Erstellen von Datenbanken und Tabellen finden Sie in der Apache-Hive-Dokumentation. Nachfolgend finden Sie jedoch eine Anleitung speziell für Athena.

Die maximale Länge der Abfragezeichenfolge beträgt 256 KB.

Hive unterstützt mehrere Datenformate mithilfe von SerDe (Serializer/Deserializer)-Bibliotheken. Sie können auch komplexe Schemata mit regulären Ausdrücken definieren. Eine Liste der unterstützten SerDe-Bibliotheken finden Sie unter Unterstützte SerDes- und Daten-Formate.

Überlegungen und Einschränkungen

Im Folgenden finden Sie einige wichtige Einschränkungen und Überlegungen für Tabellen in Athena.

Voraussetzungen für Tabellen in Athena und Daten in Amazon S3

Beim Erstellen einer Tabelle geben Sie einen Amazon-S3-Bucket-Speicherort für die zugrunde liegenden Daten mit der Klausel `LOCATION` an. Berücksichtigen Sie dabei Folgendes:

- Athena kann nur die neueste Version der Daten in einem versionierten Amazon-S3-Bucket und keine vorherigen Versionen der Daten abfragen.
- Sie benötigen die entsprechenden Berechtigungen zum Arbeiten mit Daten im Amazon-S3-Speicherort. Weitere Informationen finden Sie unter Zugriff auf Amazon S3.
- Athena unterstützt das Abfragen von Objekten, die mit verschiedenen Speicherklassen im gleichen von der `LOCATION`-Klausel angegebenen Bucket gespeichert werden. Beispielsweise können Sie Daten in Objekten abfragen, die in verschiedenen Speicherklassen in Amazon S3 gespeichert sind (Standard, Standard IA und Intelligent-Tiering).
- Athena unterstützt Buckets mit Zahlung durch den Anforderer. Weitere Informationen zum Aktivieren der Zahlung durch den Anforderer für Buckets mit Quelldaten, die Sie in Athena abfragen möchten, finden Sie unter Erstellen von Arbeitsgruppen.
- Athena unterstützt keine Abfragen für Daten in den Speicherklassen S3 Glacier Flexible Retrieval oder S3 Glacier Deep Archive. Objekte in den Speicherklassen S3 Glacier Flexible Retrieval und S3 Glacier Deep Archive werden ignoriert. Alternativ können Sie die Speicherklasse Amazon S3 Glacier Instant Retrieval verwenden, die von Athena abgefragt werden kann. Weitere Informationen finden Sie unter Amazon-S3-Speicherklasse Glacier Instant Retrieval.

  Weitere Informationen finden Sie unter Speicherklassen, Ändern der Speicherklasse eines Objekts in Amazon S3, Übergang in die Speicherklasse GLACIER (Objektarchivierung) und Buckets mit Zahlung durch den Anforderer im Benutzerhandbuch für Amazon Simple Storage Service.
- Wenn Sie Abfragen gegen Amazon-S3-Buckets mit einer großen Anzahl von Objekten ausgeben und die Daten nicht partitioniert sind, können sich solche Abfragen auf die Grenzwerte für die Get-Anforderungsrate in Amazon S3 auswirken und zu Amazon-S3-Ausnahmen führen. Um Fehler zu vermeiden, partitionieren Sie Ihre Daten. Überlegen Sie zusätzlich, ob Sie Ihre Amazon-S3-Anforderungsraten optimieren möchten. Weitere Informationen finden Sie unter Anfragerate und Leistungsaspekte.
- Wenn Sie den API-Vorgang AWS Glue CreateTable oder die Vorlage AWS CloudFormation `AWS::Glue::Table` verwenden, um eine Tabelle zur Verwendung in Athena zu erstellen, ohne die `TableType`-Eigenschaft anzugeben und dann eine DDL-Abfrage wie `SHOW CREATE TABLE` oder `MSCK REPAIR TABLE` ausführen, können Sie die Fehlermeldung FEHLGESCHLAGEN: NullPointerException-Name is null anzeigen.

Um den Fehler zu beheben, geben Sie einen Wert für das TableInput `TableType`-Attribut als Teil des AWS Glue-API-Aufrufs `CreateTable` oder der z-AWS CloudFormation Vorlage an. Mögliche Werte für `TableType` sind `EXTERNAL_TABLE` oder `VIRTUAL_VIEW`.

Diese Anforderung gilt nur, wenn Sie eine Tabelle mit dem AWS Glue-API-Vorgang `CreateTable` oder der `AWS::Glue::Table`-Vorlage erstellen. Wenn Sie eine Tabelle für Athena mithilfe einer DDL-Anweisung oder eines AWS Glue-Crawlers erstellen, wird die `TableType`-Eigenschaft automatisch für Sie definiert.

Unterstützte Funktionen

Die in Athena-Abfragen unterstützten Funktionen entsprechen denen in Trino und Presto. Informationen zu individuellen Funktionen finden Sie unter im Funktionen- und Operatorenbereich in der Trino or Presto-Dokumentation.

Umwandlungen von Transaktionsdaten werden nicht unterstützt

Athena bietet keine Unterstützung für transaktionsbasierte Vorgänge (wie in Hive oder Presto) für Tabellendaten. Eine vollständige Liste der nicht unterstützten Schlüsselwörter finden Sie unter Nicht unterstützte DDLs.

Vorgänge, die Tabellenstatus ändern, sind ACID

Die Vorgänge zum Erstellen, Aktualisieren und Löschen von Tabellen sind garantiert ACID-kompatibel. Wenn beispielsweise mehrere Benutzer oder Clients gleichzeitig versuchen, eine vorhandene Tabelle zu erstellen oder zu bearbeiten, ist nur einer erfolgreich.

Tabellen sind EXTERNAL

Verwenden Sie außer beim Erstellen von Iceberg-Tabellen immer das `EXTERNAL`-Schlüsselwort. Wenn Sie `CREATE TABLE` ohne das `EXTERNAL`-Schlüsselwort für Nicht-Iceberg-Tabellen verwenden, gibt Athena einen Fehler aus. Wenn Sie eine Tabelle in Athena löschen, werden nur die Tabellenmetadaten gelöscht. Die Daten verbleiben in Amazon S3.

Erstellen von Tabellen mit AWS Glue oder der Athena-Konsole

Sie können Tabellen in Athena mithilfe von AWS Glue, des Formulars zum Hinzufügen von Tabellen oder durch Ausführen einer DDL-Anweisung im Athena-Abfrage-Editor erstellen.

So erstellen Sie eine Tabelle mit dem AWS Glue-Crawler

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wählen Sie im Abfrage-Editor neben Tables and views (Tabellen und Ansichten) Create (Erstellen) und danach AWS Glue-Crawler aus.
3. Führen Sie auf der Seite Add crawler (Crawler hinzufügen) der AWS Glue-Konsole die Schritte zum Hinzufügen eines Crawlers aus.

   Weitere Informationen finden Sie unter Verwenden von AWS Glue Crawlern.

So erstellen Sie eine Tabelle mit dem Athena-Formular „Tabelle erstellen“

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wählen Sie im Abfrage-Editor neben Tables and views (Tabellen und Ansichten) Create (Erstellen) und danach S3 bucket data (S3-Bucket-Daten) aus.
3. Geben Sie im Formular Create Table From S3 bucket data (Tabelle aus S3-Bucket-Daten erstellen) die Informationen zum Erstellen der Tabelle ein, und wählen Sie dann Create table (Tabelle erstellen) aus. Weitere Hinweise zu den Feldern im Formular finden Sie unter Hinzufügen einer Tabelle mit einem Formular.

So erstellen Sie eine Tabelle mit Hive DDL

1. Wählen Sie im Menü Database (Datenbank) die Datenbank aus, für die Sie eine Tabelle erstellen möchten. Wenn Sie keine Datenbank in Ihrer `CREATE TABLE`-Anweisung angeben, wird die Tabelle in der Datenbank erstellt, die derzeit im Abfrage-Editor ausgewählt ist.
2. Geben Sie eine Anweisung wie die folgende im Abfrage-Editor ein, und wählen Sie dann Run (Ausführen) aus, oder drücken Sie **`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/';
```

Tabelleninformationen anzeigen

Nachdem Sie eine Tabelle in Athena erstellt haben, wird der Name in der Tables (Tabellen)-Liste auf der linken Seite angezeigt. Um Informationen über die Tabelle anzuzeigen und zu verwalten, wählen Sie die vertikalen drei Punkte neben dem Tabellennamen in der Athena-Konsole aus.

- Vorversion – Zeigt die ersten zehn Zeilen aller Spalten an, indem die `SELECT * FROM "database_name"."table_name" LIMIT 10`-Anweisung im Athena-Abfrage-Editor ausgeführt wird.
- Generieren einer Tabellen-DDL – Generiert eine DDL-Anweisung, mit der Sie die Tabelle neu erstellen können, indem Sie die `SHOW CREATE TABLE`-*table_name*-Anweisung im Athena Abfrage-Editor verwenden.
- Laden von Partitionen – Führt die `MSCK REPAIR TABLE` *table_name*-Anweisung im Athena-Abfrage-Editor aus. Diese Option ist nur verfügbar, wenn die Tabelle Partitionen hat.
- In Editor einfügen – Fügt den Namen der Tabelle in den Abfrage-Editor am aktuellen Bearbeitungsort ein.
- Tabelle löschen – Zeigt ein Bestätigungsdialogfeld an, in dem Sie gefragt werden, ob Sie die Tabelle löschen möchten. Wenn Sie zustimmen, wird die `DROP TABLE` *table_name*-Anweisung im Athena-Abfrage-Editor ausgeführt.
- Tabelleneigenschaften – Zeigt den Namen der Tabelle, der Datenbank, die Zeit der Erstellung und ob die Tabelle verschlüsselte Daten enthält.

Namen für Tabellen, Datenbanken und Spalten

Verwenden Sie diese Tipps für die Benennung von Datenbankobjekten in Athena.

Datenbank-, Tabellen- und Spaltennamen müssen 255 Zeichen oder kürzer sein.

Datenbank-, Tabellen- und Spaltennamen müssen 255 Zeichen oder kürzer sein. Eine Überschreitung dieses Limits generiert einen Fehler wie Wert in 'name' konnte Einschränkung nicht erfüllen: Mitglied muss eine Länge von höchstens 255 haben.

Tabellen- und Tabellenspaltennamen in Athena dürfen nur Kleinbuchstaben enthalten

Athena akzeptiert gemischte Groß-/Kleinschreibung in DDL- und DML-Abfragen, aber die Namen werden beim Ausführen der Abfrage kleingeschrieben. Vermeiden Sie daher die Verwendung von Groß- und Kleinschreibung für Tabellen- oder Spaltennamen, und verlassen Sie sich nicht auf die Groß-/Kleinschreibung in Athena, um solche Namen zu unterscheiden. Wenn Sie beispielsweise eine DDL-Anweisung verwenden, um eine Spalte mit dem Namen `Castle` zu erstellen, wird die erstellte Spalte zu `castle` kleingeschrieben. Wenn Sie dann den Spaltennamen in einer DML-Abfrage als `Castle` oder `CASTLE` angeben, wird Athena den Namen klein schreiben, damit Sie die Abfrage ausführen können, aber die Spaltenüberschrift in der von Ihnen in der Abfrage gewählten Groß-/Kleinschreibung anzeigen.

Datenbank-, Tabellen- und Spaltennamen müssen mindestens 255 Zeichen lang sein.

Namen, die mit einem Unterstrich beginnen

Verwenden Sie beim Erstellen von Tabellen umgekehrte Anführungszeichen, um Tabellen-, Ansichts- oder Spaltennamen einzuschließen, die mit einem Unterstrich beginnen. Beispiele:

```
CREATE EXTERNAL TABLE IF NOT EXISTS `_myunderscoretable`(
  `_id` string, `_index` string)
LOCATION 's3://my-athena-data/'
```

Tabellen-, Ansichts- oder Spaltennamen, die mit Ziffern beginnen

Setzen Sie bei der Ausführung von `SELECT`-, `CTAS`- oder `VIEW`-Abfragen Anführungszeichen um Bezeichner wie Tabellen-, Ansichts- oder Spaltennamen, die mit einer Ziffer beginnen. Beispiele:

```
CREATE OR REPLACE VIEW "123view" AS
SELECT "123columnone", "123columntwo"
FROM "234table"
```

Spaltennamen und komplexe Typen

Bei komplexen Typen sind nur alphanumerische Zeichen, Unterstrich (_) und Punkt (.) in Spaltennamen erlaubt. Um eine Tabelle und Mappings für Schlüssel mit eingeschränkten Zeichen zu erstellen, können Sie eine benutzerdefinierte DDL-Anweisung verwenden. Weitere Informationen finden Sie im Artikel Erstellen von Tabellen in Amazon Athena aus verschachteltem JSON und Mappings mit JSONSerDe im AWS-Big-Data-Blog.

Reservierte Wörter

Bestimmte reservierte Wörter in Athena müssen mit Escape-Zeichen versehen werden. Um reservierte Schlüsselwörter in DDL-Anweisungen mit Escapezeichen zu versehen, schließen Sie sie in einfache umgekehrte Anführungszeichen (`) ein. Um reservierte Schlüsselwörter in SQL-SELECT-Anweisungen und in Abfragen in Ansichten mit Escape-Zeichen zu versehen, schließen Sie sie in doppelte Anführungszeichen (") ein.

Weitere Informationen finden Sie unter Reservierte Schlüsselwörter.

Weitere Informationen finden Sie auch unter

Die vollständige Syntax zur Datenbank- und Tabellenerstellung finden Sie auf den folgenden Seiten.

- CREATE DATABASE
- CREATE TABLE

Reservierte Schlüsselwörter

Wenn Sie Abfragen in Athena ausführen, die reservierte Schlüsselwörter enthalten, müssen Sie sie als Escapezeichen mit Sonderzeichen umgeben, um ihre Verarbeitung zu verhindern. Überprüfen Sie anhand der Listen in diesem Thema, welche Schlüsselwörter in Athena reserviert sind.

Um reservierte Schlüsselwörter in DDL-Anweisungen mit Escapezeichen zu versehen, schließen Sie sie in einfache umgekehrte Anführungszeichen (`) ein. Um reservierte Schlüsselwörter in SQL-SELECT-Anweisungen und in Abfragen in Ansichten mit Escape-Zeichen zu versehen, schließen Sie sie in doppelte Anführungszeichen (") ein.

- Liste reservierter Schlüsselwörter in DDL-Anweisungen
- Liste reservierter Schlüsselwörter in SQL-SELECT-Anweisungen
- Beispiele für Abfragen mit reservierten Wörtern

Liste reservierter Schlüsselwörter in DDL-Anweisungen

Athena verwendet die folgende Liste reservierter Schlüsselwörter in seinen DDL-Anweisungen. Wenn Sie sie verwenden, ohne sie mit einem Escapezeichen zu versehen, gibt Athena eine Fehlermeldung aus. Um sie mit Escapezeichen zu versehen, schließen Sie sie in einfache umgekehrte Anführungszeichen (`) ein.

Sie können DDL-reservierte Schlüsselwörter als Bezeichnernamen in DDL-Anweisungen nur verwenden, wenn Sie sie in einfache umgekehrte Anführungszeichen (`) einschließen.

```
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 reservierter Schlüsselwörter in SQL-SELECT-Anweisungen

Athena verwendet die folgende Liste reservierter Schlüsselwörter in `SELECT`-Anweisungen von SQL und in Abfragen in Ansichten.

Wenn Sie diese Schlüsselwörter als Bezeichner verwenden, müssen Sie sie in Ihren Abfrageanweisungen in doppelte Anführungszeichen (") einschließen.

```
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
```

Beispiele für Abfragen mit reservierten Wörtern

Die Abfrage im folgenden Beispiel verwendet einfache umgekehrte Anführungszeichen (`), um die DDL-bezogenen reservierten Schlüsselwörter partition und date, die für einen Tabellennamen und einen der Spaltennamen verwendet werden, mit Escapezeichen zu versehen:

```
CREATE EXTERNAL TABLE `partition` (
`date` INT,
col2 STRING
)
PARTITIONED BY (year STRING)
STORED AS TEXTFILE
LOCATION 's3://test_bucket/test_examples/';
```

Die Abfragen im folgenden Beispiel enthalten einen Spaltennamen mit den DDL-bezogenen reservierten Schlüsselwörtern in `ALTER TABLE ADD PARTITION`- und `ALTER TABLE DROP PARTITION`-Anweisungen. Die DDL-reservierten Schlüsselwörter stehen in einfachen umgekehrten Anführungszeichen (`):

```
ALTER TABLE test_table
ADD PARTITION (`date` = '2018-05-14')
```

```
ALTER TABLE test_table
DROP PARTITION (`partition` = 'test_partition_value')
```

Die Abfrage im folgenden Beispiel enthält ein reserviertes Schlüsselwort (End) als Bezeichner in einer `SELECT`-Anweisung. Das Schlüsselwort ist als Escapezeichen mit doppelten Anführungszeichen versehen:

```
SELECT *
```

```
FROM TestTable
WHERE "end" != nil;
```

Die Abfrage im folgenden Beispiel enthält ein reserviertes Schlüsselwort (first) als Bezeichner in einer SELECT-Anweisung. Das Schlüsselwort ist als Escapezeichen mit doppelten Anführungszeichen versehen:

```
SELECT "itemId"."first"
FROM testTable
LIMIT 10;
```

Tabellenspeicherort in Amazon S3

Wenn Sie eine CREATE TABLE Abfrage in Athena ausführen, registriert Athena Ihre Tabelle beim AWS Glue Data Catalog, in dem Athena Ihre Metadaten speichert.

Zur Angabe des Pfads zu Ihren Daten in Amazon S3 verwenden Sie die LOCATION-Eigenschaft, wie im folgenden Beispiel gezeigt:

```
CREATE EXTERNAL TABLE `test_table`(
...
)
ROW FORMAT ...
STORED AS INPUTFORMAT ...
OUTPUTFORMAT ...
LOCATION s3://bucketname/folder/
```

- Weitere Informationen finden Sie unter Bucket-Einschränkungen und -Limits im Benutzerhandbuch zu Amazon Simple Storage Service.
- Weitere Informationen zur Verwendung von Ordnern in Amazon S3 finden Sie unter Verwenden von Ordnern im Benutzerhandbuch von Amazon Simple Storage Service.

Das LOCATION in Amazon S3 gibt alle Dateien an, die Ihre Tabelle repräsentieren.

> **Important**
>
> Athena liest alle von Ihnen im angegebenen Amazon-S3-Ordner gespeicherten Daten. Wenn Sie Daten haben, die Athena nicht lesen soll, speichern Sie diese Daten nicht im selben Amazon-S3-Ordner wie die Daten, die Athena lesen soll. Wenn Sie die Partitionierung

nutzen, muss Ihr `WHERE`-Filter die Partition enthalten, damit Athena die Daten darin scannen kann. Weitere Informationen finden Sie unter Tabellenspeicherort und Partitionen.

Wenn Sie die `LOCATION` in der `CREATE TABLE` Anweisung angeben, verwenden Sie die folgenden Richtlinien:

- Verwenden Sie einen abschließenden Schrägstrich.
- Sie können einen Pfad zu einem Amazon-S3-Ordner oder einem Amazon-S3-Zugriffspunkt-Alias verwenden. Weitere Informationen über Amazon-S3-Zugriffspunkt-Aliase finden Sie unter Verwenden eines Alias im Bucket-Stil für Ihren Zugriffspunkt im Amazon-S3-Benutzerhandbuch.

Verwenden:

```
s3://bucketname/folder/
```

```
s3://access-point-name-metadata-s3alias/folder/
```

Verwenden Sie keines der folgenden Elemente für die Angabe von `LOCATION` für Ihre Daten.

- Verwenden Sie keine Dateinamen, Unterstriche, Platzhalter oder „glob“-Muster, um Dateispeicherorte anzugeben.
- Fügen Sie nicht die vollständige HTTP-Notation (z. B. `s3.amazon.com`) zum Amazon-S3-Bucket-Pfad hinzu.
- Verwenden Sie keine leeren Ordner wie `//` im Pfad, wie folgt: `S3://bucketname/folder//folder/`. Dies ist zwar ein gültiger Amazon-S3-Pfad, Athena lässt ihn jedoch nicht zu und ändert ihn zu `s3://bucketname/folder/folder/`, wobei der zusätzliche `/` entfernt wird.

  Nicht verwenden:

```
s3://path_to_bucket
s3://path_to_bucket/*
s3://path_to_bucket/mySpecialFile.dat
s3://bucketname/prefix/filename.csv
s3://test-bucket.s3.amazon.com
S3://bucket/prefix//prefix/
```

```
arn:aws:s3:::bucketname/prefix
s3://arn:aws:s3:<region>:<account_id>:accesspoint/<accesspointname>
https://<accesspointname>-<number>.s3-accesspoint.<region>.amazonaws.com
```

Tabellenspeicherort und Partitionen

Ihre Quelldaten können basierend auf einem Satz von Spalten in Amazon-S3-Ordnern gruppiert werden, die als Partitionen bezeichnet werden. Beispielsweise können diese Spalten das Jahr, den Monat und den Tag der Erstellung des jeweiligen Datensatzes repräsentieren.

Wenn Sie eine Tabelle erstellen, können Sie wählen, dass sie partitioniert sein soll. Wenn Athena eine SQL-Abfrage gegen eine nicht-partitionierte Tabelle ausführt, verwendet es die `LOCATION`-Eigenschaft aus der Tabellendefinition als Basispfad zum Auflisten und anschließenden Scannen aller verfügbaren Dateien. Bevor jedoch eine partitionierte Tabelle abgefragt werden kann, müssen Sie den AWS Glue Data Catalog mit Partitionsinformationen aktualisieren. Diese Informationen stellen das Schema der Dateien innerhalb der bestimmten Partition und die `LOCATION` der Dateien in Amazon S3 für die Partition dar.

- Informationen dazu, wie der AWS Glue Crawler Partitionen hinzufügt, finden Sie unter Wie bestimmt ein Crawler, wann Partitionen erstellt werden sollen? im AWS Glue -Entwicklerhandbuch.
- Informationen zum Konfigurieren des Crawlers zum Erstellen von Tabellen für Daten in vorhandenen Partitionen finden Sie unter Verwenden mehrerer Datenquellen mit Crawlern.
- Sie können auch Partitionen in einer Tabelle direkt in Athena erstellen. Weitere Informationen finden Sie unter Daten in Athena partitionieren.

Bei der Ausführung einer Abfrage auf einer partitionierten Tabelle prüft Athena zunächst, ob partitionierten Spalten in der `WHERE`-Klausel der Abfrage verwendet werden. Wenn partitionierte Spalten verwendet werden, fordert Athena den AWS Glue Data Catalog auf, die Partitionsspezifikation zurückzugeben, die den angegebenen Partitionsspalten entspricht. Die Partitionsspezifikation enthält die `LOCATION`-Eigenschaft, die Athena mitteilt, welches Amazon-S3-Präfix beim Lesen der Daten zu verwenden ist. In diesem Fall werden nur mit diesem Präfix gespeicherte Daten gescannt. Wenn Sie keine partitionierten Spalten in der `WHERE`-Klausel verwenden, scannt Athena alle Dateien, die zu den Partitionen der Tabelle gehören.

Beispiele für die Verwendung der Partitionierung mit Athena zur Verbesserung der Abfrageleistung und zur Senkung der Abfragekosten finden Sie unter Top 10 Tipps zur Leistungsoptimierung für Amazon Athena.

Spaltenbasierte Speicherformate

Apache Parquet und ORC sind spaltenbasierte Speicherformate, die für den schnellen Abruf von Daten optimiert sind und in analytischen AWS-Anwendungen verwendet werden.

Spaltenbasierte Speicherformate haben die folgenden Eigenschaften, wodurch sie sich für die Verwendung mit Athena eignen:

- Komprimierung nach Spalte mit dem für den Spaltendatentyp ausgewählten Komprimierungsalgorithmus, um Speicherplatz in Amazon S3 zu sparen und Festplattenspeicher und die I/O-Vorgänge während der Abfrageverarbeitung zu reduzieren.
- Prädikat-Pushdown in Parquet und ORC ermöglicht Athena-Abfragen, um nur die tatsächlich benötigten Blöcke abzurufen, wobei die Abfrageleistung verbessert wird. Wenn eine Athena-Abfrage Werte aus bestimmten Spalten von Ihren Daten erhält, verwendet es Statistiken von Datenblockprädikaten, wie z. B. Max.-/Min.-Werte, um festzustellen, ob der Block zu lesen oder zu überspringen ist.
- Das Aufteilen von Daten in Parquet und ORC ermöglicht Athena das Lesen von Daten an mehrere Leser und die Erhöhung der Parallelität während der Abfrageverarbeitung.

Um Ihre vorhandenen Rohdaten von anderen Speicherformaten für Parquet oder ORC zu konvertieren, können Sie CREATE TABLE AS SELECT (CTAS)-Abfragen in Athena ausführen und ein Datenspeicherformat als Parquet oder ORC angeben, oder verwenden Sie den AWS Glue-Crawler.

Wählen Sie zwischen Parquet und ORC

Die Wahl zwischen ORC (Optimized Row Columnar) und Parquet hängt von Ihren spezifischen Nutzungsanforderungen ab.

Apache Parquet bietet effiziente Datenkomprimierungs- und Kodierungsschemata und ist ideal für die Ausführung komplexer Abfragen und die Verarbeitung großer Datenmengen. Parquet ist für die Verwendung mit Apache Arrow optimiert. Dies kann von Vorteil sein, wenn Sie Tools verwenden, die sich auf Arrow beziehen.

ORC bietet eine effiziente Möglichkeit, Hive-Daten zu speichern. ORC-Dateien sind oft kleiner als Parquet-Dateien, und ORC-Indizes können Abfragen beschleunigen. Darüber hinaus unterstützt ORC komplexe Typen wie Strukturen, Maps und Listen.

Wenn Sie zwischen Parquet und ORC wählen, sollten Sie die folgenden Faktoren berücksichtigen:

Abfrageleistung – Da Parquet eine breitere Palette von Abfragetypen unterstützt, ist Parquet möglicherweise die bessere Wahl, wenn Sie komplexe Abfragen ausführen möchten.

Komplexe Datentypen – Wenn Sie komplexe Datentypen verwenden, ist ORC möglicherweise die bessere Wahl, da es ein breiteres Spektrum an komplexen Datentypen unterstützt.

Dateigröße – Wenn der Speicherplatz ein Problem darstellt, führt ORC in der Regel zu kleineren Dateien, wodurch die Speicherkosten gesenkt werden können.

Komprimierung – Sowohl Parquet als auch ORC bieten eine gute Komprimierung, aber welches Format für Sie am besten geeignet ist, hängt von Ihrem spezifischen Anwendungsfall ab.

Evolution – Sowohl Parquet als auch ORC unterstützen die Schemaentwicklung, was bedeutet, dass Sie im Laufe der Zeit Spalten hinzufügen, entfernen oder ändern können.

Sowohl Parquet als auch ORC sind eine gute Wahl für Big-Data-Anwendungen. Berücksichtigen Sie jedoch die Anforderungen Ihres Szenarios, bevor Sie sich entscheiden. Möglicherweise möchten Sie Benchmarks für Ihre Daten und Abfragen durchführen, um herauszufinden, welches Format für Ihren Anwendungsfall besser geeignet ist.

Konvertieren in spaltenbasierte Formate

Sie können die Leistung von Amazon-Athena-Abfragen verbessern, indem Sie Ihre Daten in ein spaltenbasiertes Open-Source-Format wie Apache Parquet oder ORC konvertieren.

Optionen zum einfachen Konvertieren von Quelldaten wie JSON oder CSV in ein Säulenformat, umfassen die Verwendung von CREATE TABLE AS-Abfragen oder das Ausführen von Aufträgen in AWS Glue.

- Sie können `CREATE TABLE AS`(CTAS)-Abfragen verwenden, um Daten in Parquet oder ORC in einem Schritt zu konvertieren. Ein Beispiel finden Sie unter Beispiel: Schreiben von Abfrageergebnissen in ein anderes Format auf der Beispiele für CTAS-Abfragen-Seite.
- Informationen zum Ausführen eines AWS Glue-Auftrags zur Umwandlung von CSV-Daten in Parquet finden Sie im Abschnitt „Daten vom CSV in Parquet-Format umwandeln“ im AWS-Blogbeitrag zu Big Data Entwickeln einer Data-Lake-Grundlage mit AWS Glue und Amazon S3. AWS Glue unterstützt die Verwendung derselben Technik zum Konvertieren von CSV-Daten in ORC- oder JSON-Daten in Parquet oder ORC.

Daten in Athena partitionieren

Durch eine Datenpartitionierung können Sie die pro Abfrage durchsuchte Datenmenge eingrenzen und so die Leistung verbessern sowie Kosten senken. Sie können Ihre Daten nach jedem beliebigen Schlüssel partitionieren. In der Regel werden die Daten zeitbasiert partitioniert und das führt häufig zu einem Multi-Level-Partitionierungsschema. Beispielsweise kann ein Kunde, bei dem stündlich Daten eingehen, diese nach Jahr, Monat, Datum und Stunde partitionieren. Ein anderer Kunde, der über Daten aus vielen verschiedenen Quellen verfügt, die jedoch nur einmal pro Tag geladen werden, kann nach einer Datenquellenkennung und einem Datum partitionieren.

Athena kann Partitionen im Apache-Hive-Stil verwenden, deren Datenpfade Schlüsselwertpaare enthalten, die durch Gleichheitszeichen verbunden sind (z. B. `country=us/...` oder `year=2021/month=01/day=26/...`). Daher enthalten die Pfade sowohl die Namen der Partitionsschlüssel als auch die Werte, die jeder Pfad darstellt. Um neue Hive-Partitionen in eine partitionierte Tabelle zu laden, können Sie den MSCK REPAIR TABLE-Befehl verwenden, der nur mit Partitionen im Hive-Stil funktioniert.

Athena kann auch Partitionierungsschemata im Nicht-Hive-Stil verwenden. Beispielsweise verwenden CloudTrail Protokolle und Firehose-Bereitstellungsdatenströme separate Pfadkomponenten für Datumsteile wie `data/2021/01/26/us/6fc7845e.json`. Bei solchen Partitionen im Nicht-Hive-Stil können Sie ALTER TABLE ADD PARTITION verwenden, um die Partitionen manuell hinzuzufügen.

Überlegungen und Einschränkungen

Beachten Sie bei der Verwendung der Partitionierung die folgenden Punkte:

- Wenn Sie eine partitionierte Tabelle abfragen und die Partition in der `WHERE`-Klausel angeben, scannt Athena nur die Daten aus dieser Partition. Weitere Informationen finden Sie unter Tabellenspeicherort und Partitionen.
- Wenn Sie Abfragen gegen Amazon-S3-Buckets mit einer großen Anzahl von Objekten ausgeben und die Daten nicht partitioniert sind, können sich solche Abfragen auf die Grenzwerte für die `GET`-Anforderungsrate in Amazon S3 auswirken und zu Amazon-S3-Ausnahmen führen. Um Fehler zu vermeiden, partitionieren Sie Ihre Daten. Überlegen Sie zusätzlich, ob Sie Ihre Amazon-S3-Anforderungsraten optimieren möchten. Weitere Informationen finden Sie unter Bewährte Methoden für Designmuster: Optimieren der Amazon-S3-Leistung .
- Partitionsspeicherorte zur Verwendung mit Athena müssen das `s3`-Protokoll verwenden (z. B. `s3://DOC-EXAMPLE-BUCKET/folder/`). In Athena führen Speicherorte, die andere Protokolle

verwenden (z. B. `s3a://DOC-EXAMPLE-BUCKET/folder/`), zu Abfragefehlern, wenn `MSCK REPAIR TABLE`-Abfragen für die enthaltenen Tabellen ausgeführt werden.

- Achten Sie darauf, dass der Amazon-S3-Pfad in Kleinbuchstaben und nicht mit der Camel-Case-Schreibweise geschrieben ist (z. B. `userid` statt `userId`). Wenn der S3-Pfad in Camel-Case geschrieben ist, fügt `MSCK REPAIR TABLE` die Partitionen nicht zu AWS Glue Data Cataloghinzu. Weitere Informationen finden Sie unter MSCK REPAIR TABLE.
- Da es sich bei `MSCK REPAIR TABLE` durchsucht sowohl einen Ordner als auch seine Unterordner, um ein übereinstimmendes Partitionsschema zu finden. Achten Sie darauf, dass die Daten für separate Tabellen in separaten Ordnerhierarchien aufbewahrt werden. Angenommen, Sie haben Daten für Tabelle A in `s3://table-a-data` und Daten für Tabelle B in `s3://table-a-data/table-b-data`. Wenn beide Tabellen nach Zeichenfolge partitioniert sind, fügt `MSCK REPAIR TABLE` die Partitionen für Tabelle B zu Tabelle A hinzu. Um dies zu vermeiden, verwenden Sie stattdessen separate Ordnerstrukturen wie `s3://table-a-data` und `s3://table-b-data`. Beachten Sie, dass dieses Verhalten mit Amazon EMR und Apache Hive konsistent ist.
- Wenn Sie die AWS Glue Data Catalog mit Athena verwenden, finden Sie unter -AWS Glue Endpunkte und -Kontingente Informationen zu Servicekontingenten für Partitionen pro Konto und pro Tabelle.
  - Obwohl Athena das Abfragen von AWS Glue Tabellen mit 10 Millionen Partitionen unterstützt, kann Athena nicht mehr als 1 Million Partitionen in einem einzigen Scan lesen. In solchen Szenarien kann die Indizierung von Partitionen von Vorteil sein. Weitere Informationen finden Sie im AWS -Big-Data-Blogartikel Verbessern der Abfrageleistung von Amazon Athena mithilfe von AWS Glue Data Catalog Partitionsindizes.
- Um eine Erhöhung des Partitionskontingents anzufordern, wenn Sie die verwenden AWS Glue Data Catalog, besuchen Sie die Service Quotas-Konsole für AWS Glue.

Erstellen und Laden einer Tabelle mit partitionierten Daten

Um eine Tabelle zu erstellen, verwenden Sie die `PARTITIONED BY`-Klausel in Ihrer CREATE TABLE-Anweisung. Die `PARTITIONED BY`-Klausel definiert die Schlüssel, mit denen Daten partitioniert werden sollen, wie im folgenden Beispiel. Die `LOCATION`-Klausel gibt den Stammspeicherort der partitionierten Daten an.

```
CREATE EXTERNAL TABLE users (
first string,
last string,
username string
```

```
)
PARTITIONED BY (id string)
STORED AS parquet
LOCATION 's3://DOC-EXAMPLE-BUCKET/folder/'
```

Nachdem Sie die Tabelle erstellt haben, laden Sie die Daten in den Partitionen für die Abfrage. Bei Partitionen im Hive-Stil führen Sie MSCK REPAIR TABLE aus. Bei Partitionen im Nicht-Hive-Stil verwenden Sie ALTER TABLE ADD PARTITION, um die Partitionen manuell hinzuzufügen.

Vorbereiten von Hive-artigen- und Nicht-Hive-artigen Daten für die Abfrage

In den folgenden Abschnitten wird gezeigt, wie Sie Hive-Stil- und Nicht-Hive-Stil-Daten für die Abfrage in Athena vorbereiten.

Szenario 1: Daten, die im Hive-Format auf Amazon S3 gespeichert sind

Die Partitionen in diesem Szenario werden in Amazon S3 in separaten Ordnern gespeichert. Hier ist beispielsweise die teilweise Auflistung für Beispiel-Anzeigenimpressionen, die vom `aws s3 ls`-Befehl ausgegeben werden, der die S3-Objekte unter einem angegebenen Präfix auflistet:

```
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/
```

Die Protokolle werden hier mit dem Spaltennamen (dt) gespeichert, der auf Datums- und Uhrzeitangaben gesetzt ist. Wenn Sie eine DDL mit dem Speicherort des übergeordneten Ordners, dem Schema und dem Namen der partitionierten Spalte angeben, kann Athena Abfragen für die Daten in diesen Unterordnern ausführen.

Erstellen der Tabelle

Um aus diesen Daten eine Tabelle zu erzeugen, erstellen Sie wie in der folgenden Athena-DDL-Anweisung eine Partition für „dt“:

```
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/' ;
```

Diese Tabelle nutzt den Hive-eigenen JSON-SerDe (Serializer/Deserializer) zum Lesen der in Amazon S3 gespeicherten JSON-Daten. Weitere Informationen zu den unterstützten Formaten finden Sie unter Unterstützte SerDes- und Daten-Formate.

MSCK REPAIR TABLE ausführen

Führen Sie nach dem Ausführen der `CREATE TABLE`-Abfrage den `MSCK REPAIR TABLE`-Befehl im Athena-Abfrageeditor aus, um die Partitionen zu laden, wie im folgenden Beispiel.

```
MSCK REPAIR TABLE impressions
```

Nachdem Sie diesen Befehl ausführen, sind die Daten für die Abfrage bereit.

Abfragen der Daten

Führen Sie mithilfe der partitionierten Spalte eine Datenabfrage für die Tabelle „impressions“ aus. Ein Beispiel:

```
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
```

Diese Abfrage sollte Ergebnisse ähnlich den folgenden anzeigen:

```
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
```

Szenario 2: Daten werden nicht im Hive-Format partitioniert

Im folgenden Beispiel zeigt der `aws s3 ls`-Befehl ELB-Protokolle an, die in Amazon S3 gespeichert sind. Beachten Sie, dass das Datenlayout keine `key=value`-Paare verwendet und daher nicht im Hive-Format vorliegt. (Die `--recursive`-Option für den `aws s3 ls`-Befehl gibt an, dass alle Dateien oder Objekte unter dem angegebenen Verzeichnis oder Präfix aufgelistet werden.)

```
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_$folder$
```

ALTER TABLE ADD PARTITION ausführen

Da die Daten nicht im Hive-Format vorliegen, können Sie den `MSCK REPAIR TABLE`-Befehl nicht verwenden, um der Tabelle die Partitionen hinzuzufügen, nachdem Sie sie erstellt haben. Stattdessen können Sie den ALTER TABLE ADD PARTITION-Befehl verwenden, um jede Partition manuell hinzuzufügen. Um beispielsweise die Daten in s3://athena-examples-*myregion*/elb/plaintext/2015/01/01/ zu laden, können Sie die folgende Abfrage ausführen. Beachten Sie, dass

nicht für jeden Amazon-S3-Ordner eine separate Partitionsspalte erforderlich ist und dass sich der Partitionsschlüsselwert vom Amazon-S3-Schlüssel unterscheiden kann.

```
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/'
```

Wenn bereits eine Partition vorhanden ist, erhalten Sie die Fehlermeldung Partition bereits vorhanden. Um diesen Fehler zu vermeiden, können Sie die `IF NOT EXISTS`-Klausel verwenden. Weitere Informationen finden Sie unter ALTER TABLE ADD PARTITION. Um eine Partition zu entfernen, können Sie ALTER TABLE DROP PARTITION verwenden.

Partitionsprojektion

Um zu vermeiden, dass Partitionen verwaltet werden müssen, können Sie die Partitionsprojektion verwenden. Die Partitionsprojektion ist eine Option für hoch partitionierte Tabellen, deren Struktur im Voraus bekannt ist. Bei der Partitionsprojektion werden Partitionswerte und Speicherorte aus Tabelleneigenschaften berechnet, die Sie konfigurieren, anstatt aus einem Metadaten-Repository zu lesen. Da die Berechnungen im Arbeitsspeicher schneller sind als die Remote-Suche, kann die Verwendung der Partitionsprojektion die Abfragelaufzeiten erheblich verkürzen.

Weitere Informationen finden Sie unter Partitionsprojektion mit Amazon Athena.

Weitere Ressourcen

- Informationen zu Partitionierungsoptionen für Firehose-Daten finden Sie unter Beispiel für Amazon Data Firehose.
- Sie können den JDBC-Treiber verwenden, um das Hinzufügen von Partitionen zu automatisieren.
- Sie können CTAS und INSERT INTO verwenden, um eine Datenmenge zu partitionieren. Weitere Informationen finden Sie unter Verwenden von CTAS und INSERT INTO für ETL und Datenanalyse.

Partitionsprojektion mit Amazon Athena

Sie können die Partitionsprojektion in Athena verwenden, um die Abfrageverarbeitung von hochpartitionierten Tabellen zu beschleunigen und die Partitionsverwaltung zu automatisieren.

Bei der Partitionsprojektion berechnet Athena Partitionswerte und Speicherorte aus Tabelleneigenschaften, die Sie direkt in Ihrer Tabelle in AWS Glue konfigurieren. Die

Tabelleneigenschaften ermöglichen es Athena, die erforderlichen Partitionsinformationen zu „projizieren“ oder zu ermitteln, anstatt eine zeitaufwändigere Suche nach Metadaten in AWS Glue Data Catalog durchführen zu müssen. Da In-Memory-Operationen oft schneller sind als Remote-Operationen, kann die Partitionsprojektion die Laufzeit von Abfragen für hochpartitionierte Tabellen reduzieren. Je nach den spezifischen Merkmalen der Abfrage und den zugrunde liegenden Daten kann die Partitionsprojektion die Abfragelaufzeit für Abfragen, die auf den Abruf von Partitionsmetadaten beschränkt sind, erheblich verkürzen.

Beschneidung und Projektion für stark partitionierte Tabellen

Die Partitionsbeschneidung erfasst Metadaten und „beschneidet“ sie nur für die Partitionen, die für Ihre Abfrage gelten. Dies beschleunigt häufig Abfragen. Athena verwendet die Partitionsbeschneidung für alle Tabellen mit Partitionsspalten, einschließlich der Tabellen, die für die Partitionsprojektion konfiguriert sind.

Normalerweise führt Athena bei der Verarbeitung von Abfragen eine `GetPartitions`-Abfrage an AWS Glue Data Catalog durch, bevor die Partitionsbeschneidung durchgeführt wird. Wenn eine Tabelle über eine große Anzahl von Partitionen verfügt, kann die Verwendung von `GetPartitions` die Leistung negativ beeinflussen. Um dies zu vermeiden, können Sie die Partitionsprojektion verwenden. Die Partitionsprojektion ermöglicht es Athena, Aufrufe von `GetPartitions` zu vermeiden, da die Partitionsprojektionskonfiguration Athena alle notwendigen Informationen gibt, um die Partitionen selbst zu erstellen.

Verwenden der Partitionsprojektion

Um die Partitionsprojektion zu verwenden, geben Sie die Bereiche von Partitionswerten und Projektionstypen für jede Partitionsspalte in den Tabelleneigenschaften im AWS Glue Data Catalog oder im externen Hive-Metastorean. Diese benutzerdefinierten Eigenschaften in der Tabelle ermöglichen es Athena zu wissen, welche Partitionsmuster zu erwarten sind, wenn eine Abfrage für die Tabelle ausgeführt wird. Während der Abfrageausführung verwendet Athena diese Informationen, um die Partitionswerte zu projizieren, anstatt sie aus dem AWS Glue Data Catalog oder dem externen Hive-Metastore abzurufen. Dies reduziert nicht nur die Ausführungszeit der Abfrage, sondern automatisiert auch das Partitionsmanagement, da dadurch keine manuelle Erstellung von Partitionen in Athena, AWS Glue oder Ihrem externen Hive-Metastore erforderlich ist.

> **Important**
>
> Wenn Sie die Partitionsprojektion für eine Tabelle aktivieren, ignoriert Athena alle Partitionsmetadaten, die in der Tabelle im AWS Glue Data Catalog oder Hive-Metastore registriert sind.

Anwendungsfälle

Szenarien, in denen Partitionsprojektion sinnvoll ist, sind die etwa folgenden:

- Abfragen für eine hochpartitionierte Tabelle werden nicht so schnell abgeschlossen, wie Sie dies wünschen.
- Sie fügen regelmäßig Partitionen zu Tabellen hinzu, wenn neue Datums- oder Zeitpartitionen in Ihren Daten erstellt werden. Mit der Partitionsprojektion konfigurieren Sie relative Datumsbereiche, die beim Eintreffen neuer Daten verwendet werden können.
- Sie haben stark partitionierte Daten in Amazon S3. Die Daten sind nicht für die Modellierung in Ihrem AWS Glue Data Catalog oder Hive-Metastore geeignet und Ihre Abfragen lesen nur kleine Teile davon.

Projizierbare Partitionsstrukturen

Die Partitionsprojektion kann am einfachsten konfiguriert werden, wenn Ihre Partitionen einem vorhersehbaren Muster folgen, wie z. B.

- Ganzzahlen – Jede fortlaufende Folge von ganzen Zahlen wie `[1, 2, 3, 4, ..., 1000]` oder `[0500, 0550, 0600, ..., 2500]`.
- Datumsangaben – Jede fortlaufende Folge von Datumsangaben oder Datum/Uhrzeitangaben wie `[20200101, 20200102, ..., 20201231]` oder `[1-1-2020 00:00:00, 1-1-2020 01:00:00, ..., 12-31-2020 23:00:00]`.
- Aufzählungswerte – Ein endlicher Satz von Aufzählungswerten wie Flughafencodes oder AWS-Regionen.
- AWS-Service-Protokolle – AWS-Service -Protokolle verfügen in der Regel über eine bekannte Struktur, deren Partitionsschema Sie in AWS Glue angeben können und die Athena daher für die Partitionsprojektion verwenden kann.

Anpassen der Partitionspfadvorlage

Standardmäßig erstellt Athena Partitionsspeicherorte mithilfe des Formulars `s3://<bucket>/<table-root>/partition-col-1=<partition-col-1-val>/partition-col-2=<partition-col-2-val>/`, aber wenn Ihre Daten anders organisiert sind, bietet Athena einen Mechanismus zum Anpassen dieser Pfadvorlage. Informationen zu den erforderlichen Schritten finden Sie unter Angeben von benutzerdefinierten S3-Speicherorten.

Überlegungen und Einschränkungen

Beachten Sie die folgenden Überlegungen:

- Die Partitionsprojektion beseitigt die Notwendigkeit, Partitionen manuell in AWS Glue oder einem externen Hive-Metastore anzugeben.
- Wenn Sie die Partitionsprojektion für eine Tabelle aktivieren, ignoriert Athena alle Partitionsmetadaten im AWS Glue Data Catalog oder im externen Hive-Metastore für diese Tabelle.
- Wenn eine projizierte Partition in Amazon S3 nicht vorhanden ist, projiziert Athena die Partition weiter aus. Athena löst keinen Fehler aus, aber es werden keine Daten zurückgegeben. Wenn jedoch zu viele Ihrer Partitionen leer sind, kann die Leistung im Vergleich zu herkömmlichen AWS Glue-Partitionen langsamer sein. Wenn mehr als die Hälfte der projizierten Partitionen leer ist, wird empfohlen, herkömmliche Partitionen zu verwenden.
- Abfragen für Werte, die außerhalb der für die Partitionsprojektion definierten Bereichsgrenzen liegen, geben keinen Fehler zurück. Stattdessen wird die Abfrage ausgeführt, gibt aber null Zeilen aus. Wenn Sie beispielsweise zeitbezogene Daten haben, die im Jahr 2020 beginnen und als `'projection.timestamp.range'='2020/01/01,NOW'` definiert sind, wird eine Abfrage wie `SELECT * FROM table-name WHERE timestamp = '2019/02/02'` erfolgreich abgeschlossen, aber Nullzeilen zurückgeben.
- Die Partitionsprojektion ist nur verwendbar, wenn die Tabelle über Athena abgefragt wird. Wenn dieselbe Tabelle durch einen anderen Service wie Amazon Redshift Spectrum, Athena for Spark oder Amazon EMR gelesen wird, werden die Standard-Partitionsmetadaten verwendet.
- Da die Partitionsprojektion ein Nur-DML-Feature ist, listet `SHOW PARTITIONS` Partitionen nicht auf, die von Athena projiziert, aber nicht im AWS Glue-Katalog oder im externen Hive-Metastore registriert sind.
- Athena verwendet die Tabelleneigenschaften von Ansichten nicht als Konfiguration für die Partitionsprojektion. Um diese Einschränkung zu umgehen, konfigurieren und aktivieren Sie die Partitionsprojektion in den Tabelleneigenschaften für die Tabellen, auf die die Ansichten verweisen.

- Lake-Formation-Datenfilter können nicht mit Partitionsprojektion in Athena verwendet werden.

Video

Das folgende Video zeigt, wie Sie die Partitionsprojektion verwenden, um die Leistung Ihrer Abfragen in Athena zu verbessern.

Partitionsprojektion mit Amazon Athena

Themen

- Einrichten der Partitionsprojektion
- Unterstützte Typen für die Partitionsprojektion
- Dynamische ID-Partitionierung
- Beispiel für Amazon Data Firehose

Einrichten der Partitionsprojektion

Das Einrichten der Partitionsprojektion in den Eigenschaften einer Tabelle ist ein zweistufiger Prozess:

1. Geben Sie die Datenbereiche und relevanten Muster für jede Partitionsspalte an oder verwenden Sie eine benutzerdefinierte Vorlage.
2. Aktivieren Sie die Partitionsprojektion für die Tabelle.

> **Note**
>
> Bevor Sie einer vorhandenen Tabelle Partitionsprojektionseigenschaften hinzufügen, muss die Partitionsspalte, für die Sie Partitionsprojektionseigenschaften einrichten, bereits im Tabellenschema vorhanden sein. Wenn die Partitionsspalte noch nicht vorhanden ist, müssen Sie der vorhandenen Tabelle manuell eine Partitionsspalte hinzufügen. AWS Glue führt diesen Schritt nicht automatisch für Sie durch.

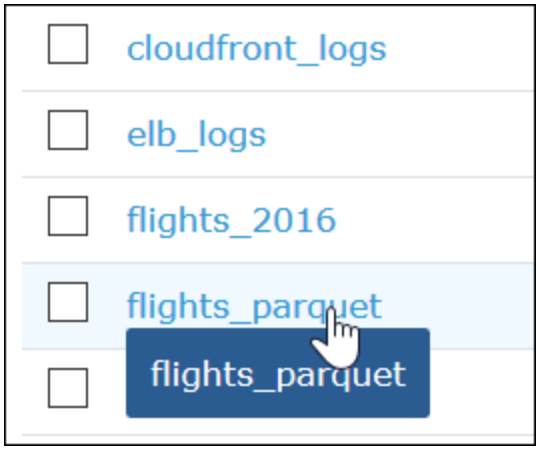
In diesem Abschnitt wird gezeigt, wie Sie die Tabelleneigenschaften für AWS Glue festlegen. Um dies zu tun, können Sie die AWS Glue-Konsole, Athena CREATE TABLE-Abfragen oder AWS Glue API-

Operationen verwenden. Im folgenden Verfahren wird gezeigt, wie die Eigenschaften in der AWS Glue-Konsole festgelegt werden.

So konfigurieren und aktivieren Sie die Partitionsprojektion mit der AWS Glue-Konsole

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die AWS Glue-Konsole unter https://console.aws.amazon.com/glue/.
2. Wählen Sie die Registerkarte Tables (Tabellen).

   Auf der Registerkarte Tables (Tabellen) können Sie vorhandene Tabellen bearbeiten oder mit Add tables (Tabellen hinzufügen) neue Tabellen erstellen. Informationen zum manuellen Hinzufügen von Tabellen oder zur Verwendung eines Crawlers dafür finden Sie unter Arbeiten mit Tabellen in der AWS Glue-Konsole im AWS Glue-Entwicklerhandbuch.
3. Wählen Sie in der Liste der Tabellen den Link für die Tabelle aus, die Sie bearbeiten möchten.

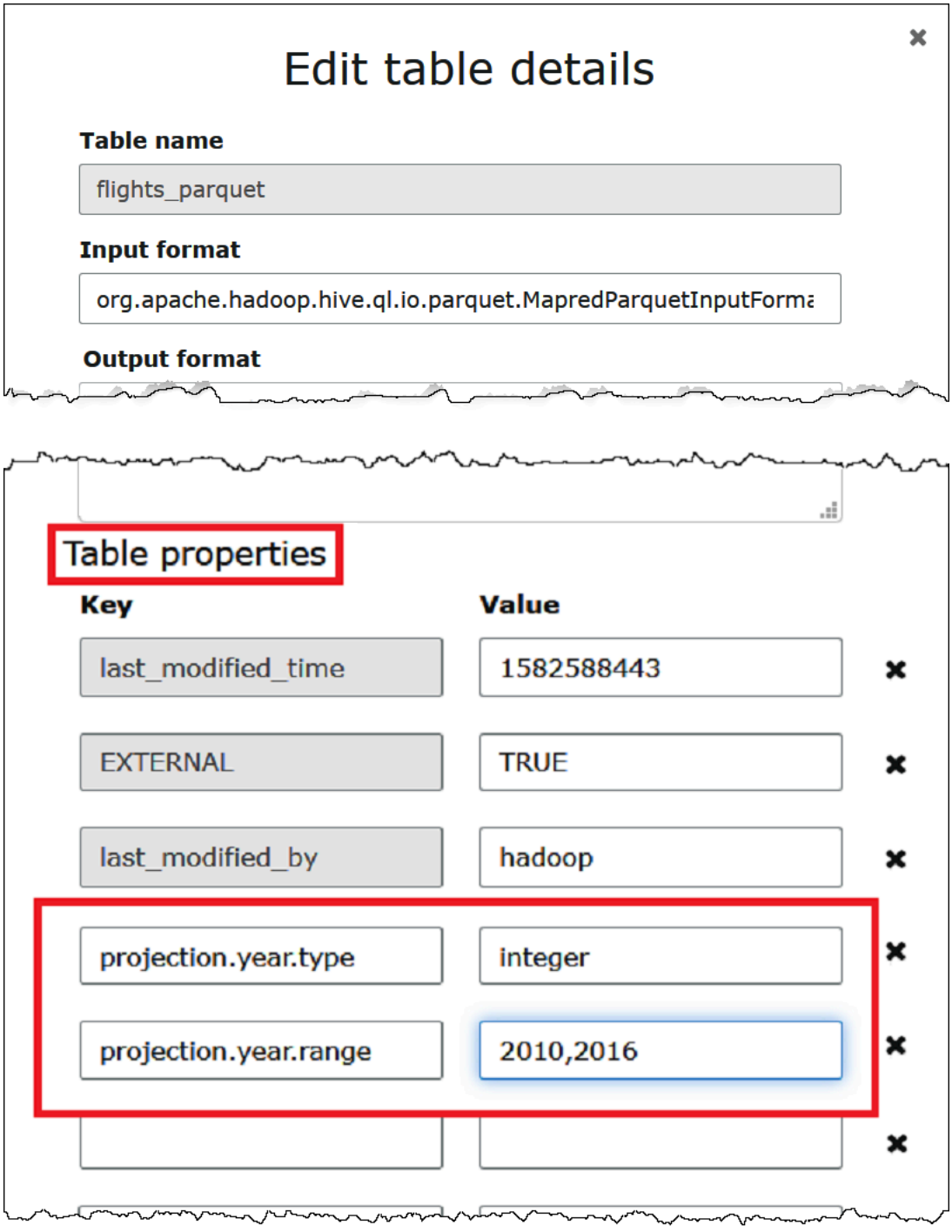


4. Wählen Sie Actions (Aktionen) und Edit table (Tabelle bearbeiten) aus.
5. Fügen Sie auf der Seite Edit table (Tabelle bearbeiten) im Abschnitt Table properties (Tabelleneigenschaften) für jede partitionierte Spalte das folgende Schlüssel-Wert-Paar hinzu:

   a. Fügen Sie für Key (Schlüssel) `projection.`*`columnName`*`.type` hinzu.

   b. Fügen Sie für Value (Wert) einen der unterstützten Typen hinzu: `enum`, `integer`, `date` oder `injected`. Weitere Informationen finden Sie unter Unterstützte Typen für die Partitionsprojektion.
6. Fügen Sie nach den Anweisungen in Unterstützte Typen für die Partitionsprojektion zusätzliche Schlüssel-Wert-Paare gemäß Ihren Konfigurationsanforderungen hinzu.

In der folgenden Beispieltabellenkonfiguration wird die `year`-Spalte für die Partitionsprojektion konfiguriert, wobei die Werte, die zurückgegeben werden können, auf einen Bereich von 2010 bis 2016 beschränkt werden.

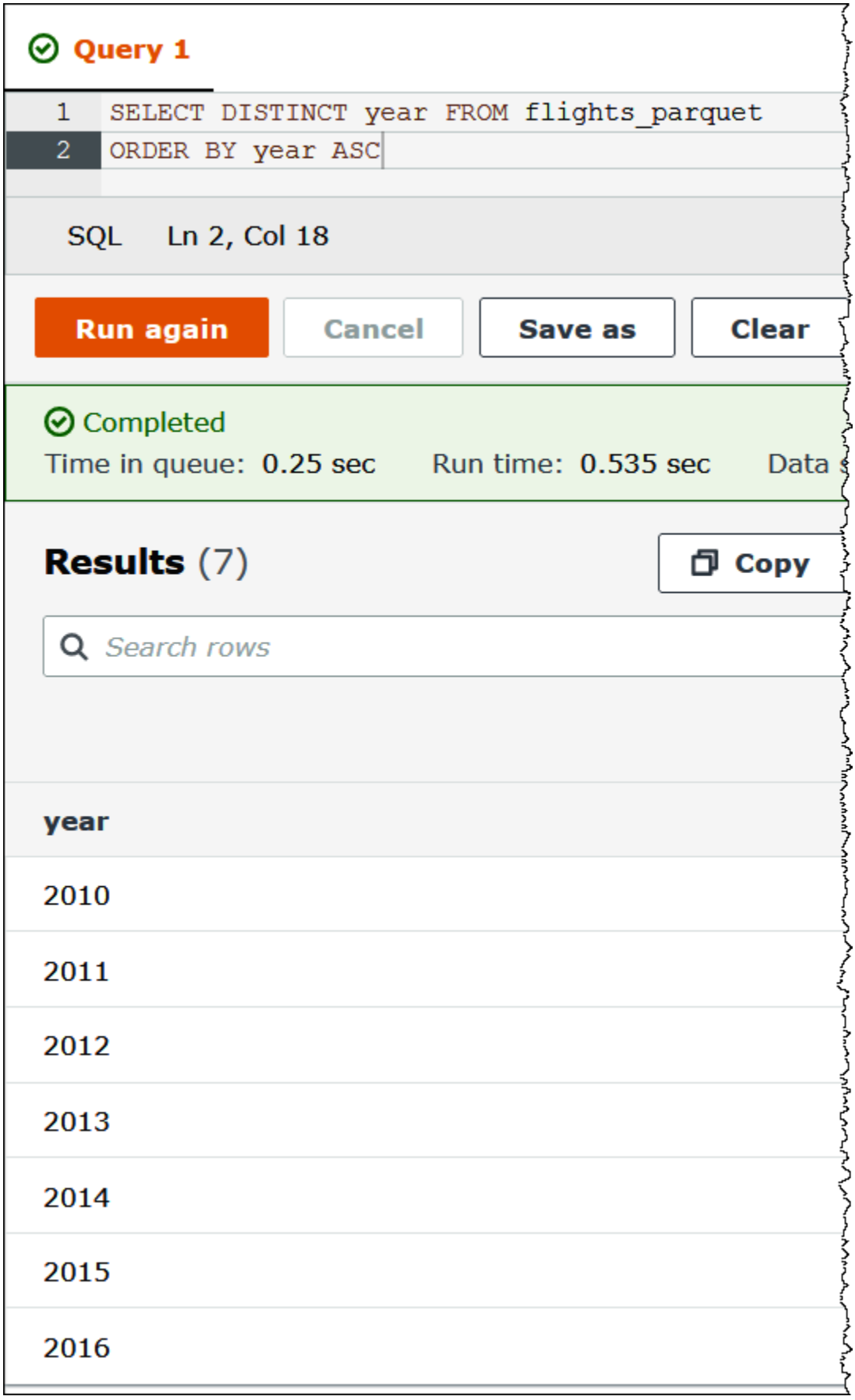


7. Fügen Sie ein Schlüssel-Wert-Paar hinzu, um die Partitionsprojektion zu aktivieren. Geben Sie für Key (Schlüssel) den Wert `projection.enabled` und für dessen Value (Wert) `true` ein.

> **Note**
>
> Sie können die Partitionsprojektion für diese Tabelle jederzeit deaktivieren, indem Sie `projection.enabled` auf `false` setzen.

8. Klicken Sie auf Save , sobald Sie fertig sind.
9. Prüfen Sie im Athena-Abfrage-Editor die Spalten, die Sie für die Tabelle konfiguriert haben.

   Die folgende Beispielabfrage verwendet `SELECT DISTINCT`, um die eindeutigen Werte aus der `year`-Spalte zurückzugeben. Die Datenbank enthält Daten von 1987 bis 2016, aber die `projection.year.range`-Eigenschaft beschränkt die zurückgegebenen Werte auf die Jahre 2010 bis 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**
>
> Wenn Sie `projection.enabled` auf `true` festlegen, aber nicht mindestens eine Partitionsspalte konfigurieren, erhalten Sie eine Fehlermeldung wie die folgende: `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).`

Angeben von benutzerdefinierten S3-Speicherorten

Wenn Sie Tabelleneigenschaften in AWS Glue bearbeiten, können Sie auch eine benutzerdefinierte Amazon-S3-Pfadvorlage für die projizierten Partitionen angeben. Eine benutzerdefinierte Vorlage ermöglicht Athena die ordnungsgemäße Zuweisung von Partitionswerten zu benutzerdefinierten Amazon-S3-Dateispeicherorten, die keinem typischen `.../column=value/...`-Muster folgen.

Die Verwendung einer benutzerdefinierten Vorlage ist optional. Wenn Sie jedoch eine benutzerdefinierte Vorlage verwenden, muss die Vorlage einen Platzhalter für jede Partitionsspalte enthalten. Vorlagenspeicherorte müssen mit einem Schrägstrich enden, damit die partitionierten Datendateien pro Partition in einem „Ordner“ gespeichert werden.

So geben Sie eine benutzerdefinierte Partitionsspeicherortvorlage an:

1. Befolgen Sie die Schritte zum Konfigurieren und Aktivieren der Partitionsprojektion mithilfe der AWS Glue-Konsole und fügen Sie ein zusätzliches Schlüssel-Wert-Paar hinzu, das eine benutzerdefinierte Vorlage wie folgt angibt:

   a. Geben Sie für Key (Schlüssel) `storage.location.template` ein.

   b. Geben Sie für Value (Wert) einen Speicherort an, der für jede Partitionsspalte einen Platzhalter enthält. Stellen Sie sicher, dass jeder Platzhalter (und der S3-Pfad selbst) durch einen einzelnen Schrägstrich beendet wird.

      In den folgenden Beispielvorlagenwerten wird von einer Tabelle mit den Partitionsspalten `a`, `b` und `c` ausgegangen.

      ```
      s3://bucket/table_root/a=${a}/${b}/some_static_subdirectory/${c}/
      ```

```
s3://bucket/table_root/c=${c}/${b}/some_static_subdirectory/${a}/${b}/${c}/
${c}/
```

Für dieselbe Tabelle ist der folgende Beispielvorlagenwert ungültig, da er keinen Platzhalter für die Spalte `c` enthält.

```
s3://bucket/table_root/a=${a}/${b}/some_static_subdirectory/
```

2. Wählen Sie Apply (Anwenden) aus.

Unterstützte Typen für die Partitionsprojektion

Eine Tabelle kann eine beliebige Kombination aus den Partitionsspaltentypen `enum`, `integer`, `date`, oder `injected` aufweisen.

Aufzählungstyp

Verwenden Sie den `enum`-Typ für Partitionsspalten, deren Werte Elemente einer Aufzählungsmenge sind (z. B. Flughafencodes oder AWS-Regionen).

Definieren Sie die Partitionseigenschaften in der Tabelle wie folgt:

Name der Eigenschaft	Beispielwerte	Beschreibung
`projection.`*`columnName`*`.type`	`enum`	Erforderlich Der Projektionstyp, der für die Spalte *`columnName`* verwendet werden soll. Der Wert muss `enum` (ohne Beachtung der Groß- und Kleinschreibung) sein, um die Verwendung des Aufzählungstyps zu signalisieren. Vorangehende und nachfolgende Leerzeichen sind zulässig.
`projection.`*`columnName`*`.values`	`A,B,C,D,E,F,G,Unknown`	Erforderlich Eine durch Kommata getrennte Liste mit

Name der Eigenschaft	Beispielwerte	Beschreibung
		aufgezählten Partitionswerten für die Spalte *columnNam e* . Jeder Leerraum wird als Teil eines Aufzählungswerts betrachtet.

> **Note**
>
> Als bewährte Methode empfehlen wir, die Verwendung von `enum`-basierten Partitionsprojektionen auf einige Dutzend oder weniger zu beschränken. Obwohl es keine spezifische Grenze für `enum`-Projektionen gibt, darf die Gesamtgröße der Metadaten Ihrer Tabelle die AWS Glue-Grenze von etwa 1 MB bei gzip-Komprimierung nicht überschreiten. Beachten Sie, dass dieses Limit für wichtige Teile Ihrer Tabelle wie Spaltennamen, Speicherort, Speicherformat und andere freigegeben wird. Wenn Sie in Ihrer `enum`-Projektion mehr als ein paar Dutzend eindeutige IDs verwenden, sollten Sie einen alternativen Ansatz in Betracht ziehen, z. B. das Bucketing in eine kleinere Anzahl eindeutiger Werte in einem Ersatzfeld. Durch den Handel mit Kardinalität können Sie die Anzahl der eindeutigen Werte in Ihrem `enum`-Feld steuern.

Ganzzahl-Typ

Verwenden Sie den Ganzzahl-Typ für Partitionsspalten, deren mögliche Werte als Ganzzahlen innerhalb eines definierten Bereichs interpretierbar sind. Projizierte Ganzzahl-Spalten sind derzeit auf den Java Signed Long-Bereich (-2^{63} bis $2^{63}-1$ inklusive) begrenzt.

Name der Eigenschaft	Beispielwerte	Beschreibung
`projection.` *columnNam e* `.type`	`integer`	Erforderlich Der Projektio nstyp, der für die Spalte *columnName* verwendet werden soll. Der Wert muss `integer` (ohne Beachtung der Groß- und Kleinschreibung) sein,

Name der Eigenschaft	Beispielwerte	Beschreibung
		um die Verwendung des Ganzzahl-Typs zu signalisieren. Vorangehende und nachfolgende Leerzeichen sind zulässig.

`projection.`*`columnName`*`.range`	`0,10`
	
	`-1,8675309`
	
	`0001,9999`
		Erforderlich Eine durch Kommata getrennte Liste mit zwei Elementen, die die minimalen und maximalen Bereichswerte bereitstellt, die von Abfragen in der Spalte *`columnName`* zurückgegeben werden. Beachten Sie, dass die Werte durch Kommata voneinander getrennt werden müssen, nicht durch einen Bindestrich. Diese Werte sind einschließlich, können negativ sein und können vorangehende Nullen aufweisen. Vorangehende und nachfolgende Leerzeichen sind zulässig.

Name der Eigenschaft	Beispielwerte	Beschreibung

`projection.` *`columnNam e`* `.interval`	`1`
	
	`5`
		Optional. Eine positive Ganzzahl, die das Intervall zwischen aufeinanderfolgend en Partitionswerten für die Spalte *`columnName`* angibt. Beispiel: Ein `range`-Wert von „1,3“ mit dem `interval`-Wert „1“ erzeugt die Werte 1, 2 und 3. Derselbe `range`-Wert mit dem `interval`-Wert „2" erzeugt die Werte 1 und 3, wobei 2 übersprungen wird. Vorangehende und nachfolge nde Leerzeichen sind zulässig. Der Standardwert ist 1.

`projection.` *`columnNam e`* `.digits`	`1`
	
	`5`
		Optional. Eine positive Ganzzahl, die die Anzahl der Ziffern angibt, die in der endgültigen Darstellu ng des Partitionswerts für die Spalte *`columnName`* enthalten sein sollen. Beispiel: Ein `range`-Wert von „1,3“ mit dem `digits`-Wert „1“ erzeugt die Werte 1, 2 und 3. Derselbe `range`-Wert mit dem `digits`-Wert „2" erzeugt die Werte 01, 02 und 03. Vorangehende und nachfolge nde Leerzeichen sind zulässig. Der Standard beinhaltet keine feste Anzahl von Ziffern und keine vorangehenden Nullen.

Datumstyp

Verwenden Sie den Datumstyp für Partitionsspalten, deren Werte innerhalb eines definierten Bereichs als Datumsangaben (mit optionalen Uhrzeiten) interpretierbar sind.

> **Important**
>
> Projizierte Datumsspalten werden in Coordinated Universal Time (UTC) zur Abfrageausführungszeit generiert.

Name der Eigenschaft	Beispielwerte	Beschreibung
`projection.`*`columnName`*`.type`	`date`	Erforderlich Der Projektionstyp, der für die Spalte *`columnName`* verwendet werden soll. Der Wert muss `date` (ohne Beachtung der Groß- und Kleinschreibung) sein, um die Verwendung des Datumstyps zu signalisieren. Vorangehende und nachfolgende Leerzeichen sind zulässig.

`projection.`*`columnName`*`.range`	`201701,201812`
	
	`01-01-2010,12-31-2018`
	
	`NOW-3YEARS,NOW`
	
	`201801,NOW+1MONTH`
		Erforderlich Eine durch Kommata getrennte Liste mit zwei Elementen, die die `range`-Minimal- und Maximalwerte für die Spalte *`columnName`* bereitstellt. Diese Werte sind einschließlich und können jedes Format verwenden, das mit den Java `java.time.*` -Datentypen kompatibel ist. Sowohl der Minimal- als auch der Maximalwert müssen dasselbe Format verwenden. Das in der `.format`-Eigenschaft angegebene Format muss dem Format entsprechen, das für diese Werte verwendet wird.
		
		Diese Spalte kann auch relative Datumszeichenfolgen enthalten, die in diesem Muster für reguläre Ausdrücke formatiert sind:

Name der Eigenschaft	Beispielwerte	Beschreibung
		`\s*NOW\s*(([\+\-])\s*([0-9]+)\s*(YEARS?|MONTHS?|WEEKS?|DAYS?|HOURS?|MINUTES?|SECONDS?)\s*)?` Leerzeichen sind erlaubt, aber in Datumslitералen gelten sie als Teil der Datumszeichenfolgen selbst.

`projection.` *`columnName`* `.format`	`yyyyMM`
	
	`dd-MM-yyyy`
	
	`dd-MM-yyyy-HH-mm-ss`
		Erforderlich Eine Datumsformatzeichenfolge basierend auf dem Java-Datumsformat DateTimeFormatter. Kann ein beliebiger unterstützter `Java.time.*` -Typ sein.

`projection.` *`columnName`* `.interval`	`1`
	
	`5`
		Eine positive Ganzzahl, die das Intervall zwischen aufeinanderfolgenden Partitionswerten für die Spalte *`columnName`* angibt. Beispiel: Ein `range`-Wert von `2017-01,2018-12` mit einem `interval`-Wert von `1` und einem `interval.unit` -Wert von `MONTHS` erzeugt die Werte 2017-01, 2017-02, 2017-03 usw. Derselbe `range`-Wert mit einem `interval`-Wert von 2 und einem `interval.unit` -Wert von `MONTHS` erzeugt die Werte 2017-01, 2017-03, 2017-05 usw. Vorangehende und nachfolgende Leerzeichen sind zulässig.
		
		Wenn die angegebenen Datumsangaben eine Genauigkeit von einem Tag oder einem Monat aufweisen, ist `interval` optional und standardmäßig 1 Tag bzw. 1 Monat. Andernfalls ist `interval` erforderlich.

Name der Eigenschaft	Beispielwerte	Beschreibung

`projection.` *`columnNam e`* `.interval.unit`	`YEARS`
	`MONTHS`
	`WEEKS`
	`DAYS`
	`HOURS`
	`MINUTES`
	`SECONDS`
	`MILLIS`
		Ein Zeiteinheitenwort, das die serialisierte Form einer ChronoUnit darstellt. Mögliche Werte sind `YEARS`, `MONTHS`, `WEEKS`, `DAYS`, `HOURS`, `MINUTES`, `SECONDS` oder `MILLIS`. Bei den Werten wird die Groß- und Kleinschreibung nicht berücksichtigt.
		
		Wenn die angegebenen Datumsangaben eine Genauigkeit von einem Tag oder einem Monat aufweisen, ist `interval.unit` optional und standardmäßig 1 Tag bzw. 1 Monat. Andernfalls ist `interval.unit` erforderlich.

Injizierter Typ

Verwenden Sie den injizierten Typ für Partitionsspalten mit möglichen Werten, die nicht prozedural innerhalb eines logischen Bereichs generiert werden können, aber in der `WHERE`-Klausel einer Abfrage als einzelner Wert bereitgestellt werden.

Es ist wichtig, dabei die folgenden Punkte zu beachten:

- Abfragen zu injizierten Spalten schlagen fehl, wenn nicht für jede injizierte Spalte ein Filterausdruck bereitgestellt wird.
- Abfragen mit mehreren Werten für einen Filterausdruck in einer injizierten Spalte sind nur erfolgreich, wenn die Werte disjunkt sind.
- Nur Spalten von `string`-Typen werden unterstützt.

Name der Eigenschaft	Wert	Beschreibung
`projection.` *`columnNam e`* `.type`	`inje`	Erforderlich Der Projektionstyp, der für die Spalte *`columnName`* verwendet werden soll. Es wird nur der `string`-Typ unterstützt. Der angegebene Wert muss `injected` (ohne Beachtung der Groß- und Kleinschr

Name der Eigenschaft	Wert	Beschreibung
		eibung) sein. Vorangehende und nachfolgende Leerzeichen sind zulässig.

Weitere Informationen finden Sie unter Verwendung des `injected`-Projektionstyps.

Dynamische ID-Partitionierung

Wenn Ihre Daten nach einer Eigenschaft mit hoher Kardinalität partitioniert sind oder die Werte nicht im Voraus bekannt sind, können Sie den `injected`-Projektionstyp verwenden. Beispiele für solche Eigenschaften sind Benutzernamen und IDs von Geräten oder Produkten. Wenn Sie den `injected`-Projektionstyp zum Konfigurieren eines Partitionsschlüssels verwenden, verwendet Athena Werte aus der Abfrage selbst, um die Menge der zu lesenden Partitionen zu berechnen.

Damit Athena eine Abfrage für eine Tabelle ausführen kann, deren Partitionsschlüssel mit dem `injected`-Projektionstyp konfiguriert ist, muss Folgendes zutreffen:

- Ihre Abfrage muss mindestens einen Wert für den Partitionsschlüssel enthalten.
- Bei den Werten muss es sich um Literale oder Ausdrücke handeln, die ohne Lesen von Daten ausgewertet werden können.

Wenn eines dieser Kriterien nicht erfüllt ist, schlägt Ihre Abfrage mit der folgenden Fehlermeldung fehl:

CONSTRAINT_VIOLATION: Die Spalte *`column_name`* der injizierten projizierten Partition darf nur (und mindestens eine) Gleichheitsbedingung in der WHERE-Klausel enthalten!

Verwendung des **`injected`**-Projektionstyps

Stellen Sie sich vor, Sie verfügen über einen Datensatz, der aus Ereignissen von IoT-Geräten besteht und nach Geräte-IDs unterteilt ist. Dieser Datensatz weist folgende Merkmale auf:

- Die Geräte-IDs werden nach dem Zufallsprinzip generiert.
- Neue Geräte werden regelmäßig bereitgestellt.
- Derzeit gibt es Hunderttausende Geräte und in Zukunft werden es Millionen sein.

Dieser Datensatz lässt sich mit herkömmlichen Metastores nur schwer verwalten. Es ist schwierig, die Partitionen zwischen dem Datenspeicher und dem Metastore synchron zu halten. Das Filtern von Partitionen kann während der Abfrageplanung langsam sein. Wenn Sie jedoch eine Tabelle für die Verwendung der Partitionsprojektion konfigurieren und den `injected`-Projektionstyp verwenden, haben Sie zwei Vorteile: Sie müssen keine Partitionen im Metastore verwalten und Ihre Abfragen müssen nicht nach Partitionsmetadaten suchen.

Im folgenden `CREATE TABLE`-Beispiel wird eine Tabelle für den soeben beschriebenen Geräteereignisdatensatz erstellt. Die Tabelle verwendet den Typ der injizierten Projektion.

```
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}"
)
```

Die folgende Beispielabfrage ermittelt die Anzahl der Ereignisse, die innerhalb von 12 Stunden von drei bestimmten Geräten empfangen wurden.

```
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
```

Wenn Sie diese Abfrage ausführen, identifiziert Athena die drei Werte für den `device_id`-Partitionsschlüssel und verwendet sie zur Berechnung der Partitionsspeicherorte. Athena verwendet

den Wert für die `storage.location.template`-Eigenschaft, um die folgenden Standorte zu generieren:

- `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`

Wenn Sie die `storage.location.template`-Eigenschaft in der Konfiguration der Partitionsprojektion weglassen, verwendet Athena eine Partitionierung im Hive-Stil, um Partitionspositionen basierend auf dem Wert in `LOCATION` zu projizieren (z. B. `s3://DOC-EXAMPLE-BUCKET/prefix/device_id=4a770164-0392-4a41-8565-40ed8cec737e`).

Beispiel für Amazon Data Firehose

Wenn Sie Firehose verwenden, um Daten an Amazon S3 zu liefern, schreibt die Standardkonfiguration Objekte mit Schlüsseln, die wie im folgenden Beispiel aussehen:

```
s3://bucket/prefix/yyyy/MM/dd/HH/file.extension
```

Um eine Athena-Tabelle zu erstellen, die die Partitionen automatisch zum Zeitpunkt der Abfrage findet, können Sie die Partitionsprojektion verwenden, anstatt sie dem hinzufügen zu müssen, AWS Glue Data Catalog wenn neue Daten eingehen.

Im folgenden `CREATE TABLE` Beispiel wird die Standardkonfiguration von Firehose verwendet.

```
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}/"
)
```

Die TBLPROPERTIES-Klausel in der CREATE TABLE-Anweisung sagt Athena Folgendes:

- Verwenden Sie Partitionsprojektion beim Abfragen der Tabelle
- Der Partitionsschlüssel `datehour` ist vom Typ `date` (welcher eine optionale Zeit beinhaltet)
- Wie die Daten formatiert werden
- Der Bereich der Datumszeiten. Beachten Sie, dass die Werte durch Kommata voneinander getrennt werden müssen, nicht durch einen Bindestrich.
- Wo Sie die Daten auf Amazon S3 finden.

Wenn Sie die Tabelle abfragen, berechnet Athena die Werte für `datehour` und erstellt mithilfe der Speicherortsvorlage eine Liste von Partitionsspeicherorten.

Verwendung des **`date`**-Typs

Wenn Sie den `date`-Typ für einen projizierten Partitionsschlüssel verwenden, müssen Sie einen Bereich angeben. Da Sie keine Daten für Datumsangaben haben, bevor der Firehose-Bereitstellungs-Stream erstellt wurde, können Sie das Erstellungsdatum als Start verwenden. Und da Sie keine Daten für Datumsangaben in der Zukunft haben, können Sie das spezielle Token `NOW` als Ende verwenden.

Im CREATE TABLE-Beispiel wird als Startdatum der 1. Januar 2021 um Mitternacht UTC angegeben.

> **Note**
>
> Konfigurieren Sie einen Bereich, der Ihren Daten so genau wie möglich entspricht, damit Athena nur nach vorhandenen Partitionen sucht.

Wenn eine Abfrage für die Beispieltabelle ausgeführt wird, verwendet Athena die Bedingungen des `datehour`-Partitionsschlüssels in Kombination mit dem Bereich, um Werte zu generieren. Betrachten Sie folgende Abfrage:

```
SELECT *
```

```
FROM my_ingested_data
WHERE datehour >= '2020/12/15/00'
AND datehour < '2021/02/03/15'
```

Die erste Bedingung in der SELECT-Abfrage verwendet ein Datum, das vor dem Beginn des durch die CREATE TABLE-Anweisung angegebenen Datumsbereichs liegt. Da die Partitionsprojektionskonfiguration keine Partitionen für Daten vor dem 1. Januar 2021 angibt, sucht Athena nur an den folgenden Orten nach Daten und ignoriert die früheren Daten in der Abfrage.

```
s3://bucket/prefix/2021/01/01/00/
s3://bucket/prefix/2021/01/01/01/
s3://bucket/prefix/2021/01/01/02/
...
s3://bucket/prefix/2021/02/03/12/
s3://bucket/prefix/2021/02/03/13/
s3://bucket/prefix/2021/02/03/14/
```

Wenn die Abfrage zu einem Datum und einer Uhrzeit vor dem 3. Februar 2021 um 15:00 Uhr ausgeführt wurde, würde die letzte Partition das aktuelle Datum und die Uhrzeit widerspiegeln, nicht das Datum und die Uhrzeit in der Abfragebedingung.

Wenn Sie die neuesten Daten abfragen möchten, können Sie sich die Tatsache zunutze machen, dass Athena keine zukünftigen Daten generiert und nur ein beginnendes `datehour` angeben, wie im folgenden Beispiel.

```
SELECT *
FROM my_ingested_data
WHERE datehour >= '2021/11/09/00'
```

Auswählen von Partitionsschlüsseln

Sie können angeben, wie die Partitionsprojektion die Partitionspositionen Partitionsschlüsseln zuordnet. Im CREATE TABLE-Beispiel im vorigen Abschnitt wurden Datum und Uhrzeit zu einem Partitionsschlüssel namens datehour zusammengefasst, es sind jedoch auch andere Schemata möglich. Zum Beispiel könnten Sie auch eine Tabelle mit separaten Partitionsschlüsseln für Jahr, Monat, Tag und Stunde konfigurieren.

Die Aufteilung von Daten in Jahr, Monat und Tag bedeutet jedoch, dass der Partitionsprojektionstyp `date` nicht verwendet werden kann. Eine Alternative besteht darin, das Datum von der Stunde zu

trennen, um weiterhin den Partitionsprojektionstyp `date` zu nutzen, Abfragen, die Stundenbereiche angeben, aber leichter lesbar zu machen.

Vor diesem Hintergrund trennt das folgende `CREATE TABLE`-Beispiel das Datum von der Stunde. Da `date` es sich um ein reserviertes Wort in SQL handelt, wird in diesem `day`-Beispiel der Name für den Partitionsschlüssel verwendet, der das Datum darstellt.

```
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}/"
)
```

In der `CREATE TABLE`-Beispielanweisung ist die Stunde ein separater Partitionsschlüssel, der als Ganzzahl konfiguriert ist. Die Konfiguration für den Stundenpartitionsschlüssel gibt den Bereich 0 bis 23 an und dass die Stunde mit zwei Ziffern formatiert werden sollte, wenn Athena die Partitionspositionen generiert.

Eine Abfrage für die `my_ingested_data2`-Tabelle könnte so aussehen:

```
SELECT *
FROM my_ingested_data2
WHERE day = '2021/11/09'
AND hour > 3
```

Partitionsschlüssel-Typen und Partitionsprojektions-Typen

Beachten Sie, dass der `datehour`-Schlüssel im ersten `CREATE TABLE`-Beispiel als `date` in der Partitionsprojektionskonfiguration konfiguriert ist, aber der Typ des Partitionsschlüssels `string` ist. Dies gilt auch für `day` im zweiten Beispiel. Die Typen in der Partitionsprojektionskonfiguration sagen Athena nur, wie die Werte formatiert werden sollen, wenn die Partitionspositionen generiert werden. Die von Ihnen angegebenen Typen ändern den Typ des Partitionsschlüssels nicht – in Abfragen sind `datehour` und `day` vom Typ `string`.

Wenn eine Abfrage eine Bedingung wie `day = '2021/11/09'` enthält, parst Athena die Zeichenfolge auf der rechten Seite des Ausdrucks unter Verwendung des Datumsformats, das in der Partitionsprojektionskonfiguration angegeben ist. Nachdem Athena überprüft hat, dass das Datum innerhalb des konfigurierten Bereichs liegt, wird das Datumsformat erneut als Zeichenfolge in die Speicherortsvorlage eingefügt.

In ähnlicher Weise analysiert Athena für eine Abfragebedingung wie `day > '2021/11/09'` die rechte Seite und generiert eine Liste aller übereinstimmenden Daten innerhalb des konfigurierten Bereichs. Anschließend wird das Datumsformat verwendet, um jedes Datum in die Speicherortvorlage einzufügen, um die Liste der Partitionspositionen zu erstellen.

Das Schreiben der gleichen Bedingung wie `day > '2021-11-09'` oder `day > DATE '2021-11-09'` funktioniert nicht. Im ersten Fall stimmt das Datumsformat nicht überein (beachten Sie die Bindestriche anstelle der Schrägstriche) und im zweiten Fall stimmen die Datentypen nicht überein.

Verwenden von benutzerdefinierten Präfixen und dynamischer Partitionierung

Firehose kann mit benutzerdefinierten Präfixen und dynamischer Partitionierung konfiguriert werden. Mit diesen Features können Sie die Amazon-S3-Schlüssel konfigurieren und Partitionierungsschemata einrichten, die Ihren Anwendungsfall besser unterstützen. Sie können auch die Partitionsprojektion mit diesen Partitionierungsschemas verwenden und entsprechend konfigurieren.

Sie könnten beispielsweise das Feature des benutzerdefinierten Präfixes verwenden, um Amazon-S3-Schlüssel mit ISO-formatierten Daten anstelle des Standard-`yyyy/MM/dd/HH`-Schemas zu erhalten.

Sie können auch benutzerdefinierte Präfixe mit dynamischer Partitionierung kombinieren, um eine Eigenschaft wie `customer_id` aus Firehose-Nachrichten zu extrahieren, wie im folgenden Beispiel.

```
prefix/!{timestamp:yyyy}-!{timestamp:MM}-!{timestamp:dd}/!
{partitionKeyFromQuery:customer_id}/
```

Mit diesem Amazon S3-Präfix würde der Firehose-Bereitstellungs-Stream Objekte in Schlüssel wie schreibens3://bucket/prefix/2021-11-01/customer-1234/file.extension. Für eine Eigenschaft wie customer_id, bei der die Werte möglicherweise nicht im Voraus bekannt sind, können Sie den Partitionsprojektionstyp injected verwenden und eine CREATE TABLE-Anweisung wie die folgende verwenden:

```
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}/"
)
```

Wenn Sie eine Tabelle mit einem Partitionsschlüssel vom Typ injected abfragen, muss Ihre Abfrage einen Wert für diesen Partitionsschlüssel enthalten. Eine Abfrage für die my_ingested_data3-Tabelle könnte so aussehen:

```
SELECT *
FROM my_ingested_data3
WHERE day BETWEEN '2021-11-01' AND '2021-11-30'
AND customer_id = 'customer-1234'
```

ISO-formatierte Daten

Da die Werte für den Partitionsschlüssel `day` ISO-formatiert sind, können Sie anstelle von `STRING` auch den Typ `DATE` für den Partitionsschlüssel Tag verwenden, wie im folgenden Beispiel:

```
PARTITIONED BY (day DATE, customer_id STRING)
```

Wenn Sie eine Abfrage durchführen, können Sie mit dieser Strategie Datumsfunktionen für den Partitionsschlüssel verwenden, ohne sie zu analysieren oder umzuwandeln, wie im folgenden Beispiel:

```
SELECT *
FROM my_ingested_data3
WHERE day > CURRENT_DATE - INTERVAL '7' DAY
AND customer_id = 'customer-1234'
```

Note

Wenn Sie einen Partitionsschlüssel dieses `DATE`-Typs angeben, wird davon ausgegangen, dass Sie das Feature für das benutzerdefinierte Präfix verwendet haben, um Amazon-S3-Schlüssel mit ISO-formatierten Datumsangaben zu erstellen. Wenn Sie das Standard-Firehose-Format von verwendenyyyy/MM/dd/HH, müssen Sie den Partitionsschlüssel als Typ angeben, `string` obwohl die entsprechende Tabelleneigenschaft vom Typ ist`date`, wie im folgenden Beispiel:

```
PARTITIONED BY (
  `mydate` string)
TBLPROPERTIES (
  'projection.enabled'='true',
   ...
  'projection.mydate.type'='date',
  'storage.location.template'='s3://DOC-EXAMPLE-BUCKET/prefix/${mydate}')
```

Erstellen einer Tabelle aus Abfrageergebnissen (CTAS)

Eine `CREATE TABLE AS SELECT`-(CTAS)-Abfrage erstellt eine neue Tabelle in Athena aus den Ergebnissen einer `SELECT`-Anweisung aus einer anderen Abfrage. Athena speichert Datendateien,

die von einer CTAS-Anweisung erstellt wurden, in einem angegebenen Speicherort in Amazon S3. Weitere Informationen zur Syntax finden Sie unter CREATE TABLE AS.

`CREATE TABLE AS` kombiniert eine `CREATE TABLE`-DDL-Anweisung mit einer `SELECT`-DML-Anweisung und enthält daher technisch gesehen sowohl DDL als auch DML. Beachten Sie jedoch, dass CTAS-Abfragen in Athena aus Gründen der Service Quotas als DML behandelt werden. Informationen zu Service Quotas für Athena finden Sie unter Service Quotas.

Verwenden Sie CTAS-Abfragen, um:

- Tabellen aus Abfrageergebnissen in einem Schritt zu erstellen, ohne wiederholt Rohdatensets abzufragen. Dies vereinfacht die Arbeit mit Rohdatensets.
- Transformieren Sie Abfrageergebnisse und migrieren Sie Tabellen in andere Tabellenformate wie Apache Iceberg. Dies verbessert die Abfrageleistung und reduziert die Abfragekosten in Athena. Weitere Informationen finden Sie unter Erstellen von Iceberg-Tabellen.
- Transformieren Sie Abfrageergebnisse in Speicherformate wie Parquet und ORC. Dies verbessert die Abfrageleistung und reduziert die Abfragekosten in Athena. Weitere Informationen finden Sie unter Spaltenbasierte Speicherformate.
- Kopien von vorhandenen Tabellen zu erstellen, die nur die Daten enthalten, die Sie benötigen.

Themen

Überlegungen und Einschränkungen für CTAS-Abfragen

In den folgenden Abschnitten werden Überlegungen und Einschränkungen beschrieben, die Sie bei der Verwendung von `CREATE TABLE AS SELECT` (CTAS)-Abfragen in Athena berücksichtigen sollten.

CTAS-Abfragesyntax

Die CTAS-Abfragesyntax unterscheidet sich von der Syntax `CREATE [EXTERNAL] TABLE` für das Erstellen von Tabellen. Siehe CREATE TABLE AS.

CTAS-Abfragen im Vergleich zu Ansichten

CTAS-Abfragen schreiben neue Daten in einen bestimmten Speicherort in Amazon S3, während Ansichten keine Daten schreiben.

Speicherort der CTAS-Abfrageergebnisse

Wenn Ihre Arbeitsgruppe die clientseitige Einstellung überschreibt, die für den Speicherort der Abfrageergebnisse gilt, erstellt Athena die Tabelle am Speicherort `s3://<workgroup-query-results-location>/tables/<query-id>/`. Den für die Arbeitsgruppe angegebenen Speicherort für Abfrageergebnisse können Sie in den Arbeitsgruppendetails anzeigen.

Wenn Ihre Arbeitsgruppe den Speicherort für Abfrageergebnisse nicht überschreibt, können Sie die Syntax `WITH (external_location ='s3://location/')` in der CTAS-Abfrage verwenden, um den Speicherort für CTAS-Abfrageergebnisse anzugeben.

> **Note**
>
> Die Eigenschaft `external_location` muss einen Speicherort angeben, der leer ist. Eine CTAS-Abfrage überprüft, ob der Pfadspeicherort (Präfix) im Bucket leer ist und überschreibt niemals die Daten, wenn der Speicherort bereits Daten enthält. Um denselben Speicherort erneut zu verwenden, löschen Sie die Daten im Schlüsselpräfixspeicherort im Bucket.

Wenn Sie die Syntax `external_location` auslassen und die Arbeitsgruppeneinstellung nicht verwenden, verwendet Athena die clientseitige Einstellung für den Abfrageergebnisspeicherort und erstellt die Tabelle am Speicherort `s3://<client-query-results-location>/<Unsaved-or-query-name>/<year>/<month/<date>/tables/<query-id>/`.

Suchen verwaister Dateien

Wenn eine `CTAS`- oder `INSERT INTO`-Anweisung fehlschlägt, ist es möglich, dass verwaiste Daten am Datenspeicherort belassen werden. Da Athena keine Daten oder Teildaten aus Ihrem Bucket löscht, können Sie diese Teildaten möglicherweise in nachfolgenden Abfragen lesen. Zur Suche

nach verwaisten Dateien zwecks Überprüfung oder Löschung können Sie die Daten-Manifest-Datei verwenden, die Athena zur Verfügung stellt, um die Liste der zu schreibenden Dateien zu verfolgen. Weitere Informationen finden Sie unter Identifizieren von Abfrageausgabedateien und DataManifestLocation.

ORDER-BY-Klauseln wurden ignoriert

In einer CTAS-Abfrage ignoriert Athena `ORDER BY`-Klauseln im `SELECT`-Teil der Abfrage.

Gemäß der SQL-Spezifikation (ISO 9075 Teil 2) ist die Reihenfolge der Zeilen einer durch einen Abfrageausdruck angegebenen Tabelle nur für den Abfrageausdruck garantiert, der die `ORDER BY`-Klausel unmittelbar enthält. Tabellen in SQL sind ohnehin von Natur aus ungeordnet, und die Implementierung der `ORDER BY` in der Unterabfrage-Klauseln würde sowohl zu einer schlechten Leistung der Abfrage als auch zu keiner geordneten Ausgabe führen. Daher gibt es bei Athena-CTAS-Abfragen keine Garantie dafür, dass die in der `ORDER BY`-Klausel angegebene Reihenfolge beim Schreiben der Daten beibehalten wird.

Formate für das Speichern von Abfrageergebnissen

Die Ergebnisse der CTAS-Abfragen werden standardmäßig in Parquet gespeichert, wenn Sie kein Datenspeicherformat angeben. Sie können CTAS-Ergebnisse in `PARQUET`, `ORC`, `AVRO`, `JSON` und `TEXTFILE` speichern. Mehrzeichen-Trennzeichen werden für das CTAS-`TEXTFILE`-Format nicht unterstützt. CTAS-Abfragen erfordern keine Angabe von a SerDe , um Formattransformationen zu interpretieren. Siehe Example: Writing query results to a different format.

Komprimierungsformate

`GZIP`-Komprimierung wird für CTAS-Abfrageergebnisse in JSON- und TEXTFILE-Formaten verwendet. Für Parquet können Sie `GZIP` oder `SNAPPY` verwenden, der Standardwert ist `GZIP`. Für ORC können Sie `LZ4`, `SNAPPY`, `ZLIB`oder `ZSTD` verwenden, der Standardwert ist `ZLIB`. Beispiele für CTAS, die die Komprimierung angeben, finden Sie unter Example: Specifying data storage and compression formats. Weitere Informationen zur Komprimierung in Athena finden Sie unter Athena-Komprimierungs-Support.

Partitions- und Bucket-Limits

Sie können die Ergebnisdaten einer CTAS-Abfrage partitionieren und in Buckets speichern. Weitere Informationen finden Sie unter Partitionierung und Bucketing in Athena. Beim Erstellen einer partitionierten Tabelle mit CTAS hat Athena ein Limit von 100 Partitionen.

Fügen Sie Partitionierungs- und Bucketing-Prädikate am Ende der `WITH`-Klausel ein, die Eigenschaften der Zieltabelle angibt. Weitere Informationen finden Sie unter Example: Creating bucketed and partitioned tables und Partitionierung und Bucketing in Athena.

Hinweise zum Umgehen der Begrenzung auf 100 Partitionen finden Sie unter Verwenden von CTAS und INSERT INTO zum Umgehen des Limits von 100 Partitionen.

Verschlüsselung

Sie können CTAS-Abfrageergebnisse in Amazon S3 verschlüsseln, so wie Sie auch andere Abfrageergebnisse in Athena verschlüsseln. Weitere Informationen finden Sie unter Verschlüsseln der in Amazon S3 gespeicherten Athena-Abfrageergebnisse.

Erwarteter Bucket-Eigentümer

Für CTAS-Anweisungen gilt die erwartete Bucket-Eigentümereinstellung nicht für den Speicherort der Zieltabelle in Amazon S3. Die erwartete Bucket-Eigentümereinstellung gilt nur für den Amazon-S3-Ausgabespeicherort, den Sie für Athena-Abfrageergebnisse angeben. Weitere Informationen finden Sie unter Angeben eines Speicherorts des Abfrageergebnisses mithilfe der Athena-Konsole.

Datentypen

Spaltendatentypen für eine CTAS-Abfrage sind dieselben wie für die ursprüngliche Abfrage angegeben.

Ausführen von CTAS-Abfragen in der Konsole

In der Athena-Konsole können Sie eine CTAS-Abfrage aus einer anderen Abfrage erstellen.

So erstellen Sie eine CTAS-Abfrage aus einer anderen Abfrage

1. Führen Sie die Abfrage im Abfrage-Editor der Athena-Konsole aus.
2. Wählen Sie unten im Abfrage-Editor die Option Create (Erstellen) aus und wählen Sie dann Table from query (Tabelle aus Abfrage) aus.
3. Füllen Sie im Formular Create table as select (Tabelle als Auswahl erstellen) die Felder wie folgt aus:

    a. Geben Sie im Feld Table name (Tabellenname) den Namen für Ihre neue Tabelle an. Verwenden Sie nur Kleinbuchstaben und Unterstriche, wie z. B. `my_select_query_parquet`.

b. Verwenden Sie für Database configuration (Datenbankkonfiguration) die Optionen, um eine vorhandene Datenbank auszuwählen oder eine Datenbank zu erstellen.

c. (Optional) Führen Sie in Result configuration (Ergebniskonfiguration) für Location of CTAS query results (Speicherort der CTAS-Abfrageergebnisse) einen der folgenden Schritte aus, wenn die Einstellung für den Speicherort Ihrer Arbeitsgruppen-Abfrageergebnisse diese Option nicht überschreibt:

   - Geben Sie den Pfad zu einem vorhandenen S3-Speicherort in das Suchfeld ein oder wählen Sie Browse S3 (S3 durchsuchen) aus, um einen Speicherort aus einer Liste auszuwählen.
   - Wählen Sie View (Anzeigen), um die Seite Buckets der Amazon-S3-Konsole zu öffnen. Hier finden Sie weitere Informationen zu Ihren vorhandenen Buckets und können einen Bucket mit Ihren eigenen Einstellungen auswählen oder erstellen.

   Sie sollten einen leeren Speicherort in Amazon S3 angeben, an dem die Daten ausgegeben werden. Wenn an dem von Ihnen angegebenen Speicherort bereits Daten vorhanden sind, schlägt die Abfrage mit einem Fehler fehl.

   Wenn die Speicherorteinstellung Ihrer Arbeitsgruppen-Abfrageergebnisse diese Standorteinstellung überschreibt, erstellt Athena Ihre Tabelle am Speicherort `s3://workgroup_query_results_location/tables/query_id/`

d. Geben Sie unter Data format (Datenformat) das Format an, in dem Ihre Daten vorliegen.

   - Tabellentyp – Der Standardtabellentyp in Athena ist Apache Hive.
   - Dateiformat – Wählen Sie zwischen Optionen wie CSV, TSV, JSON, Parquet oder ORC. Weitere Informationen über die Formate Parquet und ORC finden Sie unter Spaltenbasierte Speicherformate.
   - Schreibkomprimierung – (Optional) Wählen Sie ein Komprimierungsformat. Athena unterstützt eine Vielzahl von Komprimierungsformate zum Lesen und Schreiben von Daten, einschließlich des Lesens aus einer Tabelle, die mehrere Komprimierungsformate verwendet. Zum Beispiel kann Athena die Daten in einer Tabelle erfolgreich lesen, die das Parquet-Dateiformat verwendet, wenn einige Parquet-Dateien mit Snappy komprimiert werden und andere Parquet-Dateien mit GZIP komprimiert werden. Das gleiche Prinzip gilt für ORC-, Textfile- und JSON-Speicherformate. Weitere Informationen finden Sie unter Athena-Komprimierungs-Support.

- Partitionen – (Optional) Wählen Sie die Spalten aus, die Sie partitionieren möchten. Die Partitionierung Ihrer Daten schränkt die Menge der von jeder Abfrage gescannten Daten ein, wodurch die Leistung verbessert und die Kosten gesenkt werden. Sie können Ihre Daten nach jedem beliebigen Schlüssel partitionieren. Weitere Informationen finden Sie unter Daten in Athena partitionieren.
- Buckets – (Optional) Wählen Sie die Spalten aus, die Sie auslagern möchten. Bucketing ist eine Technik, bei der Daten auf der Grundlage bestimmter Spalten in einer einzigen Partition gruppiert werden. Diese Spalten werden als Bucket-Schlüssel bezeichnet. Durch die Gruppierung zusammengehöriger Daten in einem einzigen Bucket (eine Datei innerhalb einer Partition) reduzieren Sie die Menge der von Athena gescannten Daten erheblich und verbessern so die Abfrageleistung und senken die Kosten. Weitere Informationen finden Sie unter Partitionierung und Bucketing in Athena.

e. Überprüfen Sie für Preview table query (Vorschau der Tabellenabfrage) Ihre Abfrage. Weitere Informationen zur Abfragesyntax finden Sie unter CREATE TABLE AS.

f. Wählen Sie Create table (Tabelle erstellen) aus.

Erstellen Sie eine CTAS-Abfrage mithilfe einer SQL-Vorlage wie folgt

Verwenden Sie die `CREATE TABLE AS SELECT`-Vorlage zum Erstellen einer CTAS-Abfrage im Abfrage-Editor.

1. Wählen Sie in der Athena-Konsole neben Tables and views (Tabellen und Ansichten) Create table (Tabelle erstellen) und dann CREATE TABLE AS SELECT (TABELLE ERSTELLEN WIE AUSGEWÄHLT) aus. Dies befüllt den Abfrage-Editor mit einer CTAS-Abfrage mit Platzhalterwerten.
2. Bearbeiten Sie im Abfrageeditor die Abfrage nach Bedarf. Weitere Informationen zur Abfragesyntax finden Sie unter CREATE TABLE AS.
3. Wählen Sie Run (Ausführen) aus.

Beispiele finden Sie unter Beispiele für CTAS-Abfragen.

Partitionierung und Bucketing in Athena

Partitionierung und Bucketing sind zwei Möglichkeiten, die Datenmenge zu reduzieren, die Athena scannen muss, wenn Sie eine Abfrage ausführen. Partitionierung und Bucketing ergänzen sich

und können zusammen verwendet werden. Durch die Reduzierung der Datenmenge, die gescannt werden muss, wird die Leistung verbessert und die Kosten gesenkt. Allgemeine Hinweise zur Athena-Abfrageleistung finden Sie unter Top 10 der Leistungsoptimierungstipps für Amazon Athena.

Was ist Partitionierung?

Partitionierung bedeutet, Daten auf Amazon S3 auf der Grundlage einer bestimmten Eigenschaft der Daten in Verzeichnissen (oder „Präfixen“) zu organisieren. Solche Eigenschaften werden Partitionsschlüssel genannt. Ein üblicher Partitionsschlüssel ist das Datum oder eine andere Zeiteinheit wie das Jahr oder der Monat. Ein Datensatz kann jedoch nach mehr als einem Schlüssel partitioniert werden. Beispielsweise können Daten über Produktverkäufe nach Datum, Produktkategorie und Markt partitioniert werden.

Entscheiden, wie partitioniert werden soll

Gute Kandidaten für Partitionsschlüssel sind Eigenschaften, die in Abfragen immer oder häufig verwendet werden und eine geringe Kardinalität aufweisen. Es gibt einen Kompromiss zwischen zu vielen Partitionen und zu wenigen. Bei zu vielen Partitionen führt die erhöhte Anzahl von Dateien zu Mehraufwand. Außerdem entsteht durch das Filtern der Partitionen selbst ein gewisser Mehraufwand. Bei zu wenigen Partitionen müssen Abfragen oft mehr Daten scannen.

Eine partitionierte Tabelle erstellen

Wenn ein Datensatz partitioniert ist, können Sie in Athena eine partitionierte Tabelle erstellen. Eine partitionierte Tabelle ist eine Tabelle mit Partitionsschlüsseln. Wenn Sie `CREATE TABLE` verwenden, fügen Sie der Tabelle Partitionen hinzu. Wenn Sie `CREATE TABLE AS` verwenden, werden die Partitionen, die auf Amazon S3 erstellt wurden, automatisch zur Tabelle hinzugefügt.

In einer `CREATE TABLE`-Anweisung geben Sie die Partitionsschlüssel in der `PARTITIONED BY (`*`column_name data_type`*`)`-Klausel an. In einer `CREATE TABLE AS` Anweisung geben Sie die Partitionsschlüssel in einer `WITH (partitioned_by = ARRAY['`*`partition_key`*`'])`-Klausel oder `WITH (partitioning = ARRAY['`*`partition_key`*`'])` für Iceberg-Tabellen an. Aus Leistungsgründen sollten Partitionsschlüssel immer vom Typ `STRING` sein. Weitere Informationen finden Sie unter String als Datentyp für Partitionsschlüssel verwenden.

Weitere Informationen `CREATE TABLE` und Informationen zur `CREATE TABLE AS`-Syntax finden Sie unter CREATE TABLE und CTAS-Tabelleneigenschaften.

Partitionierte Tabellen abfragen

Wenn Sie eine partitionierte Tabelle abfragen, verwendet Athena die Prädikate in der Abfrage, um die Liste der Partitionen zu filtern. Anschließend verwendet es die Speicherorte der passenden Partitionen, um die gefundenen Dateien zu verarbeiten. Athena kann die Menge der gescannten Daten effizient reduzieren, indem es einfach keine Daten in den Partitionen liest, die nicht den Abfrageprädikaten entsprechen.

Beispiele

Angenommen, Sie haben eine Tabelle, die nach `sales_date` und `product_category` partitioniert ist und möchten den Gesamtumsatz einer Woche in einer bestimmten Kategorie ermitteln. Sie fügen Prädikate in die `sales_date`- und `product_category`-Spalten ein, um sicherzustellen, dass Athena nur die minimale Datenmenge scannt, wie im folgenden Beispiel.

```
SELECT SUM(amount) AS total_revenue
FROM sales
WHERE sales_date BETWEEN '2023-02-27' AND '2023-03-05'
AND product_category = 'Toys'
```

Angenommen, Sie haben einen Datensatz, der nach Datum partitioniert ist, aber auch über einen detaillierten Zeitstempel verfügt.

Bei Iceberg-Tabellen können Sie einen Partitionsschlüssel so deklarieren, dass er eine Beziehung zu einer Spalte hat, aber bei Hive-Tabellen hat die Abfrage-Engine keine Kenntnis von den Beziehungen zwischen Spalten und Partitionsschlüsseln. Aus diesem Grund müssen Sie sowohl für die Spalte als auch für den Partitionsschlüssel in Ihrer Abfrage ein Prädikat angeben, um sicherzustellen, dass die Abfrage nicht mehr Daten scannt als nötig.

Nehmen wir beispielsweise an, dass die `sales`-Tabelle im vorherigen Beispiel auch eine `sold_at`-Spalte des `TIMESTAMP`-Datentyps enthält. Wenn Sie den Umsatz nur für einen bestimmten Zeitraum ermitteln möchten, würden Sie die Abfrage wie folgt schreiben:

```
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'
```

Weitere Informationen zu diesem Unterschied zwischen der Abfrage von Hive- und Iceberg-Tabellen finden Sie unter Wie Sie Abfragen für Zeitstempelfelder schreiben, die ebenfalls zeitpartitioniert sind.

Was ist Bucketing?

Bucketing ist eine Möglichkeit, die Inhalte eines Datensatzes in Kategorien zu organisieren, die als Buckets bezeichnet werden.

Die Bedeutung von Bucket und Bucketing unterscheidet sich von Amazon-S3-Buckets und sollte nicht mit diesen verwechselt werden. Beim Daten-Bucketing werden Datensätze, die denselben Wert für eine Eigenschaft haben, in denselben Bucket verschoben. Datensätze werden so gleichmäßig wie möglich auf die Buckets verteilt, sodass jeder Bucket ungefähr die gleiche Datenmenge enthält.

In der Praxis handelt es sich bei den Buckets um Dateien, und eine Hash-Funktion bestimmt, in welchen Bucket ein Datensatz aufgenommen wird. Ein bucketierter Datensatz enthält eine oder mehrere Dateien pro Bucket und Partition. Der Bucket, zu dem eine Datei gehört, ist im Dateinamen kodiert.

Vorteile von Bucketing

Bucketing ist nützlich, wenn eine Datenmenge einer bestimmten Eigenschaft bucketiert ist und Sie Datensätze abrufen möchten, in denen diese Eigenschaft einen bestimmten Wert hat. Da die Daten bucketiert sind, kann Athena anhand des Werts bestimmen, welche Dateien betrachtet werden sollen. Nehmen wir zum Beispiel an, dass ein Datensatz nach `customer_id` bucketiert ist und Sie alle Datensätze für einen bestimmten Kunden suchen möchten. Athena bestimmt den Bucket, der diese Datensätze enthält, und liest nur die Dateien in diesem Bucket.

Gute Kandidaten für das Bucketing sind Spalten mit hoher Kardinalität (d. h. mit vielen unterschiedlichen Werten), die gleichmäßig verteilt sind und die Sie häufig nach bestimmten Werten abfragen.

> **Note**
>
> Athena unterstützt nicht die Verwendung von `INSERT INTO` zum Hinzufügen neuer Datensätze zu bucketierten Tabellen.

Unterstützte Datentypen für das Filtern von Spalten mit Zeiträumen

Sie können Filter für bucketierte Spalten mit bestimmten Datentypen hinzufügen. Athena unterstützt eine solche Filterung nur für bucketierte Spalten mit den folgenden Datentypen:

- BOOLEAN
- BYTE
- DATUM
- DOUBLE
- FLOAT
- INT
- LONG
- SHORT
- STRING
- VARCHAR

Unterstützung für Hive und Spark

Athena-Engine-Version 2 unterstützt Datensätze, die mit dem Hive-Bucket-Algorithmus in Buckets zusammengefasst wurden, und Athena-Engine-Version 3 unterstützt auch den Apache-Spark-Bucketing-Algorithmus. Hive-Bucketing ist die Standardeinstellung. Wenn Ihr Datensatz mithilfe des Spark-Algorithmus in einem Bucket zusammengefasst wird, verwenden Sie die `TBLPROPERTIES`-Klausel, um den `bucketing_format`-Eigenschaftswert auf `spark` festzulegen.

> **Note**
>
> Athena hat ein Limit von 100 Partitionen pro `CREATE TABLE AS SELECT` (CTAS)-Abfrage. Ebenso können Sie einer Zieltabelle mit einer INSERT INTO-Anweisung maximal 100 Partitionen hinzufügen. Dieses Limit von 100 gilt nur, wenn die Tabelle sowohl in Buckets als auch in Partitionen aufgeteilt ist.
> Wenn Sie diese Einschränkung überschreiten, wird möglicherweise die Fehlermeldung HIVE_TOO_MANY_OPEN_PARTITIONS: Exceeded limit of 100 open writers for partitions/buckets angezeigt. Um diese Einschränkung zu umgehen, können Sie eine CTAS-Anweisung und eine Reihe von `INSERT INTO`-Anweisungen verwenden, die jeweils bis zu 100 Partitionen erstellen oder einfügen. Weitere Informationen finden Sie unter Verwenden von CTAS und INSERT INTO zum Umgehen des Limits von 100 Partitionen.

Beispiel für CREATE-TABLE-Bucketing

Verwenden Sie die `CLUSTERED BY (column)`-Klausel, gefolgt von der `INTO N BUCKETS`-Klausel, um eine Tabelle für einen vorhandenen Bucket-Datensatz zu erstellen. Die `INTO N BUCKETS`-Klausel gibt die Anzahl der Buckets an, in die die Daten aufgeteilt werden.

Im folgenden `CREATE TABLE`-Beispiel wird der `sales`-Datensatz mithilfe des Spark-Algorithmus `customer_id` in 8 Buckets aufgeteilt. Die `CREATE TABLE`-Anweisung verwendet die Klauseln `CLUSTERED BY` und `TBLPROPERTIES`, um die Eigenschaften entsprechend festzulegen.

```
CREATE EXTERNAL TABLE sales (...)
...
CLUSTERED BY (`customer_id`) INTO 8 BUCKETS
...
TBLPROPERTIES (
  'bucketing_format' = 'spark'
)
```

Beispiel für CREATE TABLE AS (CTAS)-Bucketing

Um Bucketing mit `CREATE TABLE AS` zu spezifizieren, verwenden Sie die Parameter `bucket_count` und `bucketed_by`, wie im folgenden Beispiel gezeigt.

```
CREATE TABLE sales
WITH (
  ...
  bucketed_by = ARRAY['customer_id'],
  bucket_count = 8
)
AS SELECT ...
```

Beispiel für eine Bucketing-Abfrage

In der folgenden Beispielabfrage wird nach den Namen von Produkten gesucht, die ein bestimmter Kunde im Laufe einer Woche gekauft hat.

```
SELECT DISTINCT product_name
FROM sales
WHERE sales_date BETWEEN '2023-02-27' AND '2023-03-05'
AND customer_id = 'c123'
```

Wenn diese Tabelle von `sales_date` partitioniert ist und von `customer_id` bucketiert ist, kann Athena den Bucket berechnen, in dem sich die Kundendatensätze befinden. Athena liest höchstens eine Datei pro Partition.

Weitere Informationen finden Sie auch unter

Für ein `CREATE TABLE AS`-Beispiel, das sowohl bucketierte als auch partitionierte Tabellen erstellt, siehe das Beispiel: Erstellen von bucketierten Tabellen und partitionierten Tabellen.

Beispiele für CTAS-Abfragen

Verwenden Sie die folgenden Beispiele zum Erstellen von CTAS-Abfragen. Weitere Information zur CTAS-Syntax finden Sie unter CREATE TABLE AS.

In diesem Abschnitt:

Example -Beispiel: Duplizieren einer Tabelle durch Auswahl aller Spalten

Das folgende Beispiel erstellt eine Tabelle durch Kopieren aller Spalten aus einer Tabelle:

```
CREATE TABLE new_table AS
SELECT *
FROM old_table;
```

In der folgenden Variante des gleichen Beispiels enthält Ihre `SELECT`-Anweisung auch eine `WHERE`-Klausel. In diesem Fall wählt die Abfrage nur die Zeilen aus der Tabelle aus, die die `WHERE`-Klausel erfüllen:

```
CREATE TABLE new_table AS
SELECT *
FROM old_table
WHERE condition;
```

Example -Beispiel: Auswählen bestimmter Spalten aus einer oder mehreren Tabellen

Das folgende Beispiel erstellt eine neue Abfrage, die auf einer Reihe von Spalten aus einer anderen Tabelle ausgeführt wird:

```
CREATE TABLE new_table AS
SELECT column_1, column_2, ... column_n
FROM old_table;
```

Diese Variation des gleichen Beispiels erstellt eine neue Tabelle aus bestimmten Spalten aus mehreren Tabellen:

```
CREATE TABLE new_table AS
SELECT column_1, column_2, ... column_n
FROM old_table_1, old_table_2, ... old_table_n;
```

Example -Beispiel: Erstellen einer leeren Kopie einer vorhandenen Tabelle

Das folgende Beispiel verwendet `WITH NO DATA`, um eine neue Tabelle zu erstellen, die leer ist und das gleiche Schema wie die ursprüngliche Tabelle aufweist:

```
CREATE TABLE new_table
AS SELECT *
FROM old_table
WITH NO DATA;
```

Example -Beispiel: Angeben von Datenspeicherungs- und Komprimierungsformaten

Mit CTAS können Sie eine Quelltabelle in einem Speicherformat verwenden, um eine weitere Tabelle in einem anderen Speicherformat zu erstellen.

Verwenden Sie die `format`-Eigenschaft, um `ORC` `PARQUET` ,`AVRO`, `JSON` oder `TEXTFILE` als Speicherformat für die neue Tabelle anzugeben.

Verwenden Sie für die Speicherformate `PARQUET`, `ORC`, `TEXTFILE`, und `JSON` die `write_compression`-Eigenschaft, um das Komprimierungsformat für die Daten der neuen Tabelle

anzugeben. Informationen zu den Komprimierungsformaten, die jedes Dateiformat unterstützt, finden Sie unter Athena-Komprimierungs-Support.

Das folgende Beispiel gibt an, dass die Daten in der Tabelle `new_table` im Parquet-Format gespeichert werden und die Snappy-Komprimierung verwenden. Die Standardkomprimierung für Parquet ist `GZIP`.

```
CREATE TABLE new_table
WITH (
      format = 'Parquet',
      write_compression = 'SNAPPY')
AS SELECT *
FROM old_table;
```

Das folgende Beispiel gibt an, dass die Daten in der Tabelle `new_table` im ORC-Format gespeichert werden und die Snappy-Komprimierung verwenden. Die Standardkomprimierung für ORC ist ZLIB.

```
CREATE TABLE new_table
WITH (format = 'ORC',
      write_compression = 'SNAPPY')
AS SELECT *
FROM old_table ;
```

Das folgende Beispiel gibt an, dass die Daten in der Tabelle `new_table` im Textfile-Format gespeichert werden und die Snappy-Komprimierung verwenden. Die Standardkomprimierung sowohl für das Textfile- als auch für das JSON-Format ist GZIP.

```
CREATE TABLE new_table
WITH (format = 'TEXTFILE',
      write_compression = 'SNAPPY')
AS SELECT *
FROM old_table ;
```

Example -Beispiel: Schreiben von Abfrageergebnissen in ein anderes Format

Die folgende CTAS-Abfrage wählt alle Datensätze aus `old_table` aus, die in CSV oder einem anderen Format gespeichert werden können, und erstellt eine neue Tabelle, wobei die zugrunde liegenden Daten im ORC-Format in Amazon S3 gespeichert sind:

```
CREATE TABLE my_orc_ctas_table
```

```
WITH (
      external_location = 's3://my_athena_results/my_orc_stas_table/',
      format = 'ORC')
AS SELECT *
FROM old_table;
```

Example -Beispiel: Erstellen von nicht partitionierten Tabellen

Die folgenden Beispiele erstellen Tabellen, die nicht partitioniert werden. Die Tabellendaten werden in verschiedenen Formaten gespeichert. Einige dieser Beispiele geben den externen Speicherort an.

Das folgende Beispiel erstellt eine CTAS-Abfrage, die die Ergebnisse als Textdatei speichert:

```
CREATE TABLE ctas_csv_unpartitioned
WITH (
     format = 'TEXTFILE',
     external_location = 's3://my_athena_results/ctas_csv_unpartitioned/')
AS SELECT key1, name1, address1, comment1
FROM table1;
```

Im folgenden Beispiel werden die Ergebnisse in Parquet gespeichert und der standardmäßige Ergebnisspeicherort wird verwendet:

```
CREATE TABLE ctas_parquet_unpartitioned
WITH (format = 'PARQUET')
AS SELECT key1, name1, comment1
FROM table1;
```

In der folgenden Abfrage wird die Tabelle in JSON gespeichert und bestimmte Spalten aus den Ergebnissen der ursprünglichen Tabelle ausgewählt:

```
CREATE TABLE ctas_json_unpartitioned
WITH (
     format = 'JSON',
     external_location = 's3://my_athena_results/ctas_json_unpartitioned/')
AS SELECT key1, name1, address1, comment1
FROM table1;
```

Im folgenden Beispiel lautet das Format ORC:

```
CREATE TABLE ctas_orc_unpartitioned
```

```
WITH (
     format = 'ORC')
AS SELECT key1, name1, comment1
FROM table1;
```

Im folgenden Beispiel lautet das Format Avro:

```
CREATE TABLE ctas_avro_unpartitioned
WITH (
     format = 'AVRO',
     external_location = 's3://my_athena_results/ctas_avro_unpartitioned/')
AS SELECT key1, name1, comment1
FROM table1;
```

Example -Beispiel: Erstellen partitionierter Tabellen

Die folgenden Beispiele zeigen `CREATE TABLE AS SELECT`-Abfragen für partitionierte Tabellen in verschiedenen Datenspeicherformaten unter Verwendung von `partitioned_by` und anderen Eigenschaften in der `WITH`-Klausel. Weitere Informationen zur Syntax finden Sie unter CTAS-Tabelleneigenschaften. Weitere Informationen zur Auswahl der Spalten für die Partitionierung finden Sie unter Partitionierung und Bucketing in Athena.

> **Note**
>
> Auflisten von Partitionsspalten am Ende der Liste der Spalten in der `SELECT`-Anweisung. Sie können in mehrere Spalten partitionieren und über bis zu 100 eindeutige Partitions- und Bucket-Kombinationen verfügen. Sie können beispielsweise 100 Partitionen haben, wenn keine Buckets angegeben sind.

```
CREATE TABLE ctas_csv_partitioned
WITH (
     format = 'TEXTFILE',
     external_location = 's3://my_athena_results/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://my_athena_results/ctas_json_partitioned/',
     partitioned_by = ARRAY['key1'])
AS select name1, address1, comment1, key1
FROM table1;
```

Example -Beispiel: Erstellen von Tabellen mit Bucketing und Partitionierung

Das folgende Beispiel zeigt eine `CREATE TABLE AS SELECT`-Abfrage, die sowohl die Partitionierung als auch das Bucketing zum Speichern von Abfrageergebnissen in Amazon S3 verwendet. Die Tabellenergebnisse werden partitioniert und nach verschiedenen Spalten gruppiert. Athena unterstützt maximal 100 eindeutige Kombinationen aus Bucket und Partition. Wenn Sie beispielsweise eine Tabelle mit fünf Buckets erstellen, werden 20 Partitionen mit je fünf Buckets unterstützt. Weitere Informationen zur Syntax finden Sie unter CTAS-Tabelleneigenschaften.

Weitere Informationen zur Auswahl der Spalten für das Bucketing finden Sie unter Partitionierung und Bucketing in Athena.

```
CREATE TABLE ctas_avro_bucketed
WITH (
      format = 'AVRO',
      external_location = 's3://my_athena_results/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 -Beispiel: Erstellen einer Iceberg-Tabelle mit Parquet-Daten

Im folgenden Beispiel wird eine Iceberg-Tabelle mit Parquet-Datendateien erstellt. Die Dateien werden mithilfe der `dt`-Spalte in `table1` nach Monaten partitioniert. Das Beispiel aktualisiert die Aufbewahrungseigenschaften für die Tabelle, sodass standardmäßig 10 Snapshots in jeder Verzweigung in der Tabelle aufbewahrt werden. Snapshots innerhalb der letzten 7 Tage werden ebenfalls aufbewahrt. Weitere Informationen zu Iceberg-Tabelleneigenschaften in Athena finden in Tabelleneigenschaften.

```
CREATE TABLE ctas_iceberg_parquet
WITH (table_type = 'ICEBERG',
      format = 'PARQUET',
```

```
      location = 's3://my_athena_results/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 -Beispiel: Erstellen einer Iceberg-Tabelle mit Avro-Daten

Im folgenden Beispiel wird eine Iceberg-Tabelle mit Avro-Datendateien mit Partitionierung durch `key1` erstellt.

```
CREATE TABLE ctas_iceberg_avro
WITH ( format = 'AVRO',
       location = 's3://my_athena_results/ctas_iceberg_avro/',
       is_external = false,
       table_type = 'ICEBERG',
       partitioning = ARRAY['key1'])
AS SELECT key1, name1, date FROM table1;
```

Verwenden von CTAS und INSERT INTO für ETL und Datenanalyse

Sie können Create-Table-as-Select-(CTAS)- und INSERT-INTO-Anweisungen in Athena verwenden, um ETL-Daten für die Datenverarbeitung in Amazon S3 zu extrahieren, transformieren und laden. In diesem Thema wird erläutert, wie Sie diese Anweisungen zum Partitionieren und Konvertieren eines Datasets in das spaltenförmige Datenformat verwenden, um es für die Datenanalyse zu optimieren.

CTAS-Anweisungen verwenden standardmäßige SELECT-Abfragen um neue Tabellen zu erstellen. Sie können eine CTAS-Anweisung verwenden, um eine Teilmenge Ihrer Daten für die Analyse zu erstellen. In einer CTAS-Anweisung können Sie die Daten partitionieren, komprimieren und die Daten in ein Spaltenformat wie Apache Parquet oder Apache ORC konvertieren. Wenn Sie die CTAS-Abfrage ausführen, werden die von ihr erstellten Tabellen und Partitionen automatisch dem AWS Glue Data Catalog hinzugefügt. Dadurch sind die neu erstellten Tabellen und Partitionen sofort für nachfolgende Abfragen verfügbar.

INSERT INTO-Anweisungen fügen neue Zeilen basierend auf einer SELECT-Abfrageanweisung, die in einer Quelltabelle ausgeführt wird, in eine Zieltabelle ein. Sie können INSERT INTO-Anweisungen verwenden, um Quelltabellendaten im CSV-Format mit allen Transformationen, die von CTAS unterstützt werden, in Zieltabellendaten zu transformieren und zu laden.

Übersicht

Verwenden Sie in Athena eine CTAS-Anweisung, um eine erste Batch-Konvertierung der Daten durchzuführen. Verwenden Sie dann mehrere INSERT INTO-Anweisungen, um inkrementelle Aktualisierungen der von der CTAS-Anweisung erstellten Tabelle vorzunehmen.

Schritte

- Schritt 1: Erstellen einer Tabelle basierend auf dem ursprünglichen Datensatz
- Schritt 2: Verwenden von CTAS zum Partitionieren, Konvertieren und Komprimieren der Daten
- Schritt 3: Verwenden von INSERT INTO zum Hinzufügen von Daten
- Schritt 4: Messen von Leistungs- und Kostendifferenzen

Schritt 1: Erstellen einer Tabelle basierend auf dem ursprünglichen Datensatz

Das Beispiel in diesem Thema verwendet eine in Amazon S3 lesbare Teilmenge des öffentlich verfügbaren NOAA Global Historical Climatology Network Daily (GHCN-d)-Datensatzes. Die Daten für Amazon S3 haben die folgenden Eigenschaften.

```
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
```

Die Originaldaten werden ohne Partitionen in Amazon S3 gespeichert. Die Daten befinden sich im CSV-Format in Dateien wie den folgenden.

```
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
```

Die Dateigrößen in diesem Beispiel sind relativ klein. Durch die Zusammenführung in größere Dateien können Sie die Gesamtzahl der Dateien reduzieren und so eine bessere Abfrageausführung ermöglichen. Sie können CTAS- und INSERT INTO-Anweisungen verwenden, um die Abfrageleistung zu verbessern.

So erstellen Sie eine Datenbank und eine Tabelle basierend auf dem Beispiel-Dataset

1. Wählen Sie in der Athena-Konsole die USA Ost (Nord-Virginia) AWS-Region. Stellen Sie sicher, dass Sie alle Abfragen in diesem Lernprogramm in `us-east-1` ausführen.
2. Führen Sie im Athena-Abfrage-Editor den Befehl CREATE DATABASE aus, um eine Datenbank zu erstellen.

```
CREATE DATABASE blogdb
```

3. Führen Sie die folgende Anweisung aus, um eine Tabelle zu erstellen.

```
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/'
```

Schritt 2: Verwenden von CTAS zum Partitionieren, Konvertieren und Komprimieren der Daten

Nachdem Sie eine Tabelle erstellt haben, können Sie die Daten mit einer einzelnen CTAS-Anweisung in das Parquet-Format mit Snappy-Komprimierung konvertieren und die Daten nach Jahr partitionieren.

Die Tabelle, die Sie in Schritt 1 erstellt haben, enthält ein `date`-Feld, in dem das Datum als YYYYMMDD formatiert ist (z. B. `20100104`). Da die neue Tabelle nach `year` partitioniert wird, verwendet die Beispielanweisung im folgenden Verfahren die Presto-Funktion `substr("date",1,4)`, um den `year`-Wert aus dem Feld `date` zu extrahieren.

So konvertieren Sie die Daten in das Parquet-Format mit Snappy Komprimierung, partitioniert nach Jahr

- Führen Sie die folgende CTAS-Anweisung aus und ersetzen Sie *your-bucket* durch Ihren Amazon-S3-Bucket-Speicherort.

```
CREATE table new_parquet
WITH (format='PARQUET',
parquet_compression='SNAPPY',
partitioned_by=array['year'],
external_location = 's3://your-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**
>
> In diesem Beispiel enthält die Tabelle, die Sie erstellen, nur die Daten von 2015 bis 2019. In Schritt 3 fügen Sie dieser Tabelle neue Daten hinzu, indem Sie den Befehl INSERT INTO verwenden.

Gehen Sie nach Abschluss der Abfrage folgendermaßen vor, um die Ausgabe an dem Amazon-S3-Speicherort zu überprüfen, den Sie in der CTAS-Anweisung angegeben haben.

So zeigen Sie die Partitionen und Parquet -Dateien an, die von der CTAS-Anweisung erstellt wurden

1. Führen Sie den folgenden AWS CLI-Befehl aus, um die erstellten Partitionen anzuzeigen. Achten Sie darauf, den letzten Schrägstrich (/) einzuschließen.

```
aws s3 ls s3://your-bucket/optimized-data/
```

Die Ausgabe zeigt die Partitionen.

```
      PRE year=2015/
      PRE year=2016/
      PRE year=2017/
      PRE year=2018/
      PRE year=2019/
```

2. Führen Sie den folgenden Befehl aus, um die Parquet-Dateien anzuzeigen. Beachten Sie, dass die Option | head -5, die die Ausgabe auf die ersten fünf Ergebnisse beschränkt, unter Windows nicht verfügbar ist.

```
aws s3 ls s3://your-bucket/optimized-data/ --recursive --human-readable | head -5
```

Die Ausgabe sieht in etwa folgendermaßen aus.

```
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
```

Schritt 3: Verwenden von INSERT INTO zum Hinzufügen von Daten

In Schritt 2 haben Sie CTAS verwendet, um eine Tabelle mit Partitionen für die Jahre 2015 bis 2019 zu erstellen. Der ursprüngliche Datensatz enthält jedoch auch Daten für die Jahre 2010 bis 2014. Nun fügen Sie diese Daten mit einer INSERT INTO-Anweisung hinzu.

So fügen Sie der Tabelle Daten mit einer oder mehreren INSERT INTO-Anweisungen hinzu

1. Führen Sie den folgenden INSERT INTO-Befehl aus und geben Sie die Jahre vor 2015 in der WHERE-Klausel an.

```
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. Führen Sie den `aws s3 ls`-Befehl mit der folgenden Syntax erneut aus.

```
aws s3 ls s3://your-bucket/optimized-data/
```

Die Ausgabe zeigt die neuen Partitionen.

```
      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. Führen Sie den folgenden Befehl aus, um die Verringerung der Größe des Datasets anzuzeigen, die durch Komprimierung und Säulenspeicherung im Parquet-Format erzielt wurde.

```
aws s3 ls s3://your-bucket/optimized-data/ --recursive --human-readable --summarize
```

Die folgenden Ergebnisse zeigen, dass die Größe des Datensatzes nach Parquet mit Snappy-Komprimierung 1,2 GB beträgt.

```
...
```

```
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. Wenn der ursprünglichen Tabelle mehr CSV-Daten hinzugefügt werden, können Sie diese Daten mit INSERT INTO-Anweisungen zur Parquet-Tabelle hinzufügen. Wenn Sie beispielsweise neue Daten für das Jahr 2020 haben, können Sie die folgende INSERT INTO-Anweisung ausführen. Die Anweisung fügt die Daten und die entsprechende Partition zur Tabelle `new_parquet` hinzu.

```
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**
>
> Die Anweisung INSERT INTO unterstützt das Schreiben von maximal 100 Partitionen in die Zieltabelle. Um jedoch mehr als 100 Partitionen hinzuzufügen, können Sie mehrere INSERT INTO-Anweisungen ausführen. Weitere Informationen finden Sie unter Verwenden von CTAS und INSERT INTO zum Umgehen des Limits von 100 Partitionen.

Schritt 4: Messen von Leistungs- und Kostendifferenzen

Nachdem Sie die Daten transformiert haben, können Sie die Leistungssteigerungen und Kosteneinsparungen messen, indem Sie dieselben Abfragen in den neuen und alten Tabellen ausführen und die Ergebnisse vergleichen.

> **Note**
>
> Informationen zu Athena-Kosten pro Abfrage finden Sie unter Preise für Amazon Athena.

So messen Sie Leistungssteigerungen und Kostenunterschiede

1. Führen Sie die folgende Abfrage für die ursprüngliche Tabelle aus. Die Abfrage findet die Anzahl der eindeutigen IDs für jeden Wert des Jahres.

```
SELECT substr("date",1,4) as year,
       COUNT(DISTINCT id)
FROM original_csv
GROUP BY 1 ORDER BY 1 DESC
```

2. Beachten Sie, wie lange Abfrage ausgeführt wurde und die Menge der gescannten Daten.
3. Führen Sie dieselbe Abfrage für die neue Tabelle aus und achten Sie dabei auf die Abfrageausführungszeit und die Menge der gescannten Daten.

```
SELECT year,
  COUNT(DISTINCT id)
FROM new_parquet
GROUP BY 1 ORDER BY 1 DESC
```

4. Vergleichen Sie die Ergebnisse und berechnen Sie die Leistungs- und Kostendifferenz. Die folgenden Beispielergebnisse zeigen, dass die Testabfrage für die neue Tabelle schneller und billiger war als die Abfrage für die alte Tabelle.

Tabelle	Laufzeit	Gescannte Daten
Original	16,88 Sekunden	11,35 GB
Neu	3,79 Sekunden	428,05 MB

5. Führen Sie die folgende Beispielabfrage für die ursprüngliche Tabelle aus. Die Abfrage berechnet die durchschnittliche Höchsttemperatur (Celsius), durchschnittliche Mindesttemperatur (Celsius) und durchschnittliche Niederschlagsmenge (mm) für die Erde im Jahr 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. Beachten Sie, wie lange Abfrage ausgeführt wurde und die Menge der gescannten Daten.
7. Führen Sie dieselbe Abfrage für die neue Tabelle aus und achten Sie dabei auf die Abfrageausführungszeit und die Menge der gescannten Daten.

```
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. Vergleichen Sie die Ergebnisse und berechnen Sie die Leistungs- und Kostendifferenz. Die folgenden Beispielergebnisse zeigen, dass die Testabfrage für die neue Tabelle schneller und billiger war als die Abfrage für die alte Tabelle.

Tabelle	Laufzeit	Gescannte Daten
Original	18,65 Sekunden	11,35 GB
Neu	1,92 Sekunden	68 MB

Übersicht

In diesem Thema wurde gezeigt, wie ETL-Operationen mit CTAS- und INSERT INTO-Anweisungen in Athena ausgeführt werden. Sie haben den ersten Satz von Transformationen mit einer CTAS-Anweisung durchgeführt, die Daten in das Parquet-Format mit Snappy-Komprimierung konvertiert hat. Die CTAS-Anweisung konvertiert auch das Dataset von nicht partitionierten in partitionierte Daten. Dies reduzierte seine Größe und senkte die Kosten für die Ausführung der Abfragen. Wenn neue Daten verfügbar sind, können Sie sie mit einer INSERT INTO-Anweisung in die Tabelle transformieren und laden, die Sie mit der CTAS-Anweisung erstellt haben.

Verwenden von CTAS und INSERT INTO zum Umgehen des Limits von 100 Partitionen

Athena hat ein Limit von 100 Partitionen pro `CREATE TABLE AS SELECT` (CTAS)-Abfrage. Ebenso können Sie einer Zieltabelle mit einer INSERT INTO-Anweisung maximal 100 Partitionen hinzufügen. Dieses Limit gilt nur, wenn die Tabelle sowohl in Buckets als auch in Partitionen aufgeteilt ist.

Wenn Sie diese Einschränkung überschreiten, wird möglicherweise die Fehlermeldung HIVE_TOO_MANY_OPEN_PARTITIONS: Exceeded limit of 100 open writers for partitions/buckets angezeigt. Um diese Einschränkung zu umgehen, können Sie eine CTAS-Anweisung und eine Reihe von `INSERT INTO`-Anweisungen verwenden, die jeweils bis zu 100 Partitionen erstellen oder einfügen.

Das Beispiel in diesem Thema verwendet eine Datenbank mit dem Namen `tpch100`, deren Daten sich im Amazon-S3-Bucket-Speicherort „s3://*<my-tpch-bucket>*/“ befinden.

So erstellen Sie mithilfe von CTAS und INSERT INTO eine Tabelle mit mehr als 100 Partitionen

1. Erstellen Sie mit einer `CREATE EXTERNAL TABLE`-Anweisung eine Tabelle, die für das gewünschte Feld partitioniert ist.

   Die folgende Beispielanweisung partitioniert die Daten durch die Spalte `l_shipdate`. Der Tisch hat 2525 Partitionen.

```
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://<my-tpch-bucket>/lineitem/'
```

2. Führen Sie einen `SHOW PARTITIONS <table_name>`-Befehl wie den folgenden aus, um die Partitionen aufzulisten.

```
SHOW PARTITIONS lineitem_parq_partitioned
```

Es folgen teilweise Beispielergebnisse.

```
/*
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. Führen Sie eine CTAS-Abfrage aus, um eine partitionierte Tabelle zu erstellen.

Im folgenden Beispiel wird eine Tabelle mit dem Namen erstellt `my_lineitem_parq_partitioned` und die `WHERE` -Klausel verwendet, um `DATE` auf früher als `1992-02-01` zu beschränken. Da das Beispiel-Dataset mit Januar 1992 beginnt, werden nur Partitionen für Januar 1992 erstellt.

```
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. Führen Sie den Befehl `SHOW PARTITIONS` aus, um zu überprüfen, ob die Tabelle die gewünschten Partitionen enthält.

```
SHOW PARTITIONS my_lineitem_parq_partitioned;
```

Die Partitionen im Beispiel stammen vom Januar 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. Verwenden Sie eine `INSERT INTO`-Anweisung, um Partitionen zur Tabelle hinzuzufügen.

   Im folgenden Beispiel werden Partitionen für die Datumsangaben des Monats Februar 1992 hinzugefügt.

```
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. Führen Sie `SHOW PARTITIONS` erneut aus.

```
SHOW PARTITIONS my_lineitem_parq_partitioned;
```

   Die Beispieltabelle enthält nun Partitionen vom Januar und Februar 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. Verwenden Sie weiterhin `INSERT INTO`-Anweisungen, die nicht mehr als 100 Partitionen lesen und hinzufügen. Fahren Sie fort, bis Sie die Anzahl der benötigten Partitionen erreicht haben.

> **Important**
>
> Achten Sie beim Festlegen der `WHERE`-Bedingung darauf, dass sich die Abfragen nicht überschneiden. Andernfalls haben einige Partitionen möglicherweise doppelte Daten.

SerDe-Referenz

Athena unterstützt mehrere SerDe-Bibliotheken zur Analyse von Daten aus mit unterschiedlichen Datenformaten wie CSV, JSON, Parquet und ORC. Athena unterstützt keine benutzerdefinierten SerDes.

Themen

Verwenden eines SerDe

Ein SerDe (Serializer/Deserializer) ist eine Methode, mit der Athena mit Daten in verschiedenen Formaten interagiert.

Es ist der von Ihnen angegebene SerDe und nicht die DDL, die zur Definition des Tabellenschemas verwendet wurde. Anders ausgedrückt kann der SerDe die DDL-Konfiguration überschreiben, die Sie in Athena beim Erstellen der Tabelle festgelegt haben.

SO verwenden Sie ein SerDe in Abfragen

Um einen SerDe beim Erstellen einer Tabelle in Athena zu verwenden, nutzen Sie eine der folgenden Methoden:

- Geben Sie `ROW FORMAT DELIMITED` an und verwenden Sie dann DDL-Anweisungen, um Feldtrennzeichen anzugeben, wie im folgenden Beispiel gezeigt. Wenn Sie `ROW FORMAT DELIMITED` verwenden, verwendet Athena standardmäßig LazySimpleSerDe.

  ```
  ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
  ESCAPED BY '\\'
  COLLECTION ITEMS TERMINATED BY '|'
  MAP KEYS TERMINATED BY ':'
  ```

  Beispiele von `ROW FORMAT DELIMITED` finden Sie in den folgenden Themen:

  LazySimpleSerDe für CSV- und TSV-Dateien sowie für benutzerdefinierte, durch Trennzeichen getrennte Dateien

  CloudFront Amazon-Logs abfragen

  Abfragen von Amazon-EMR-Protokollen

  Abfragen von Amazon-VPC-Flow-Protokollen

  Verwenden von CTAS und INSERT INTO für ETL und Datenanalyse

- Verwenden Sie `ROW FORMAT SERDE`, um explizit den Typ von SerDe anzugeben, den Athena beim Lesen und Schreiben von Daten in die Tabelle verwenden soll. Im folgenden Beispiel ist LazySimpleSerDe angegeben. Um die Trennzeichen anzugeben, verwenden Sie `WITH SERDEPROPERTIES`. Die durch `WITH SERDEPROPERTIES` angegebenen Eigenschaften

entsprechen den separaten Anweisungen (wie `FIELDS TERMINATED BY`) im `ROW FORMAT DELIMITED`-Beispiel.

```
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
WITH SERDEPROPERTIES (
'serialization.format' = ',',
'field.delim' = ',',
'collection.delim' = '|',
'mapkey.delim' = ':',
'escape.delim' = '\\'
)
```

Beispiele von `ROW FORMAT SERDE` finden Sie in den folgenden Themen:

Avro SerDe

Grok SerDe

JSON-SerDe-Bibliotheken

OpenCSVSerDe für CSV-Verarbeitung

Regex SerDe

Unterstützte SerDes- und Daten-Formate

Athena unterstützt das Erstellen von Tabellen und das Abfragen von Daten aus Dateien im CSV-, TSV- und JSON-Format sowie im benutzerdefinierten, durch Trennzeichen getrennten Format; aus Daten aus Hadoop-bezogenen Formaten wie ORC, Apache Avro und Parquet, aus Logstash-Protokollen, aus AWS CloudTrail-Protokollen und aus Apache-WebServer-Protokollen.

> **Note**
>
> Die in diesem Abschnitt aufgeführten Formate werden von Athena für das Lesen von Daten verwendet. Weitere Informationen zu Formaten, die Athena zum Schreiben von Daten beim Ausführen von CTAS-Abfragen verwendet, finden Sie unter Erstellen einer Tabelle aus Abfrageergebnissen (CTAS).

Zum Erstellen von Tabellen und zum Abfragen von Daten in diesen Formaten in Athena geben Sie eine Klasse vom Typ „serializer-deserializer“ (SerDe) an, sodass Athena weiß, welches Format verwendet wird und wie die Daten analysiert werden.

Diese Tabelle enthält die Datenformate, die in Athena unterstützt werden, sowie die entsprechenden SerDe-Bibliotheken.

Ein SerDe ist eine benutzerdefinierte Bibliothek, die festlegt, wie der von Athena verwendete Datenkatalog die Daten verarbeitet. Sie geben einen SerDe-Typ an, indem Sie diesen explizit in Athena im `ROW FORMAT`-Teil Ihrer `CREATE TABLE`-Anweisung aufführen. In einigen Fällen können Sie den SerDe-Namen weglassen, da Athena standardmäßig einige SerDe-Typen für bestimmte Arten von Datenformaten verwendet.

Unterstützte Formate für Daten und SerDes

Data format (Datenformat)	Beschreibung	In Athena unterstützte SerDe-Typen
Amazon Ion	Amazon Ion ist ein reich typisiertes, selbstbeschreibend es Datenformat, das eine Obermenge von JSON darstellt, das von Amazon entwickelt und Open Source entwickelt wurde.	Verwenden Sie die Amazon Ion Hive SerDe.
Apache Avro	Ein Format zum Speichern von Daten in Hadoop, bei dem JSON-basierte Schemas zum Erfassen von Werten verwendet werden.	Verwenden Sie Avro SerDe.
Apache Parquet	Ein Format für die spaltenba sierte Speicherung von Daten in Hadoop.	Verwenden Sie die Parquet SerDe und SNAPPY-Ko mprimierung.
Apache WebServer-Protokolle	Ein Format zum Speichern von Protokollen in Apache WebServer.	Verwenden Sie die Grok SerDe oder Regex SerDe.

Data format (Datenformat)	Beschreibung	In Athena unterstützte SerDe-Typen
CloudTrail-Protokolle	Ein Format zum Speichern von Protokollen in CloudTrail.	• Verwenden Sie die Hive JSON SerDe. Weitere Informationen finden Sie unter Abfragen von AWS CloudTrail-Protokollen.
CSV (Comma Separated Values, durch Komma getrennte Werte)	Für Daten im CSV-Format stellt jede Zeile einen Datensatz dar und jeder Datensatz besteht aus mehreren durch Kommata getrennten Feldern.	• Verwenden Sie das LazySimpleSerDe für CSV- und TSV-Dateien sowie für benutzerdefinierte, durch Trennzeichen getrennte Dateien, wenn Ihre Daten keine in Anführungszeichen eingeschlossenen Werte enthalten oder das Format `java.sql.Timestamp` verwendet wird. • Verwenden Sie das OpenCSVSerDe für CSV-Verarbeitung, wenn Ihre Daten Anführungszeichen in Werten enthalten oder das numerische UNIX-Format für `TIMESTAMP` verwendet (z. B. `1564610311`).
Benutzerdefiniert, durch Trennzeichen getrennt	Bei Daten in diesem Format stellt jede Zeile einen Datensatz dar. Die Datensätze sind durch benutzerdefinierte aus einem Zeichen bestehende Trennzeichen getrennt.	Verwenden Sie die LazySimpleSerDe für CSV- und TSV-Dateien sowie für benutzerdefinierte, durch Trennzeichen getrennte Dateien und geben Sie ein einzelnes benutzerdefiniertes Trennzeichen an.

Data format (Datenformat)	Beschreibung	In Athena unterstützte SerDe-Typen
JSON (JavaScript Object Notation)	Bei JSON-Daten stellt jede Zeile einen Datensatz dar und jeder Datensatz besteht aus Attribut/Werte-Paaren und Arrays, die durch Komma getrennt sind.	• Verwenden Sie Hive JSON SerDe. • Verwenden Sie OpenX JSON SerDe.
Logstash-Protokolle	Ein Format zum Speichern von Protokollen in Logstash.	Verwenden Sie Grok SerDe.
ORC (Optimized Row Columnar)	Ein Format für die optimierte spaltenbasierte Speicherung von Hive-Daten.	Verwenden Sie die ORC SerDe und ZLIB-Komp rimierung.
TSV (Tab-Separated Values, tabulatorgetrennte Werte)	Bei Daten im TSV-Forma t stellt jede Zeile einen Datensatz dar und jeder Datensatz besteht aus mehreren durch Tabulatoren getrennten Feldern.	Verwenden Sie die LazySimpl eSerDe für CSV- und TSV-Dateien sowie für benutzerd efinierte, durch Trennzeichen getrennte Dateien und geben Sie das Trennzeichen als `FIELDS TERMINATED BY '\t'` an.

Themen

- Amazon Ion Hive SerDe
- Avro SerDe
- Grok SerDe
- JSON-SerDe-Bibliotheken
- LazySimpleSerDe für CSV- und TSV-Dateien sowie für benutzerdefinierte, durch Trennzeichen getrennte Dateien
- OpenCSVSerDe für CSV-Verarbeitung
- ORC SerDe

- Parquet SerDe
- Regex SerDe

Amazon Ion Hive SerDe

Sie können Amazon Ion Hive SerDe verwenden, um Daten abzufragen, die im Amazon-Ion-Format gespeichert sind. Amazon Ion ist ein reich typisiertes, selbstbeschreibendes Open-Source-Datenformat. Das Amazon-Ion-Format wird von Diensten wie Amazon Quantum Ledger Database (Amazon QLDB) und in der Open-Source-SQL-Abfragesprache PartiQL verwendet.

Amazon Ion verfügt über Binär- und Textformate, die austauschbar sind. Dieses Feature kombiniert die Benutzerfreundlichkeit von Text mit der Effizienz der Binärcodierung.

Um Amazon Ion-Daten von Athena abzufragen, können Sie Amazon Ion Hive SerDe verwenden, das Amazon-Ion-Daten serialisiert und deserialisiert. Die Deserialisierung ermöglicht es Ihnen, Abfragen zu den Amazon-Ion-Daten auszuführen oder sie zu lesen, um sie in ein anderes Format wie Parquet oder ORC zu schreiben. Mit der Serialisierung können Sie Daten im Amazon-Ion-Format generieren, indem Sie `CREATE TABLE AS SELECT` (CTAS)- oder `INSERT INTO`-Abfragen verwenden, um Daten aus vorhandenen Tabellen zu kopieren.

> **Note**
>
> Da Amazon Ion eine Obermenge von JSON ist, können Sie den Amazon Ion Hive SerDe verwenden, um JSON-Datensätze außerhalb von Amazon Ion abzufragen. Im Gegensatz zu anderen JSON-SerDe-Bibliotheken erwartet Amazon Ion SerDe nicht, dass sich jede Datenzeile in einer einzelnen Zeile befindet. Dieses Feature ist nützlich, wenn Sie JSON-Datensätze im „Pretty Print“-Format abfragen oder die Felder in einer Zeile mit Zeilenumbruchzeichen aufteilen möchten.

Weitere Informationen und Beispiele für die Abfrage von Amazon Ion mit Athena finden Sie unter Analysieren von Amazon-Ion-Datensätzen mit Amazon Athena.

SerDe-Name

- com.amazon.ionhiveserde.IonHiveSerDe

Überlegungen und Einschränkungen

- Doppelte Felder – Amazon-Ion-Strukturen sind geordnet und unterstützen duplizierte Felder, während `STRUCT<>` und `MAP<>` von Hive dies nicht tun. Wenn Sie also ein dupliziertes Feld aus einer Amazon-Ion-Struktur deserialisieren, wird ein einzelner Wert nicht deterministisch gewählt und die anderen werden ignoriert.
- Externe Symboltabellen werden nicht unterstützt – Derzeit unterstützt Athena keine externen Symboltabellen oder die folgenden Amazon Ion Hive SerDe Eigenschaften:
  - `ion.catalog.class`
  - `ion.catalog.file`
  - `ion.catalog.url`
  - `ion.symbol_table_imports`
- Dateierweiterungen – Amazon Ion verwendet Dateierweiterungen, um zu ermitteln, welcher Komprimierungs-Codec für die Deserialisierung von Amazon-Ion-Dateien verwendet werden soll. Als solche müssen komprimierte Dateien die Dateierweiterung haben, die dem verwendeten Komprimierungs-Algorithmus entspricht. Wenn beispielsweise ZSTD verwendet wird, sollten entsprechende Dateien die Erweiterung `.zst` haben.
- Homogene Daten – Amazon Ion hat keine Beschränkungen für die Datentypen, die für Werte in bestimmten Feldern verwendet werden können. Beispielsweise können zwei verschiedene Amazon Ion-Dokumente ein Feld mit demselben Namen haben, das verschiedene Datentypen hat. Da Hive jedoch ein Schema verwendet, müssen alle Werte, die Sie in eine einzelne Hive-Spalte extrahieren, denselben Datentyp haben.
- Beschränkungen für Schlüsseltypen zuweisen – Wenn Sie Daten aus einem anderen Format in Amazon Ion serialisieren, stellen Sie sicher, dass der Map-Schlüsseltyp einer von `STRING`, `VARCHAR`, oder `CHAR` ist. Obwohl Sie mit Hive jeden primitiven Datentyp als Zuordnungsschlüssel verwenden können, müssen Amazon-Ion-Symbole vom Typ Zeichenfolge sein.
- Union-Art – Athena unterstützt die Hive-Union-Art derzeit nicht.
- Doppelter Datentyp — Amazon Ion unterstützt den Datentyp `double` derzeit nicht.

Themen

- Erstellen von Amazon-Ion-Tabellen mit CREATE TABLE
- Erstellen von Amazon-Ion-Tabellen mithilfe von CTAS und INSERT INTO
- Verwenden von Amazon Ion SerDe Eigenschaften
- Verwenden von Pfad-Extraktoren

Erstellen von Amazon-Ion-Tabellen mit CREATE TABLE

Um eine Tabelle in Athena aus Daten zu erstellen, die im Amazon-Ion-Format gespeichert sind, können Sie eine der folgenden Techniken in einer CREATE-TABLE-Anweisung verwenden:

- Geben Sie an `STORED AS ION`. Bei dieser Verwendung müssen Amazon Ion Hive SerDe nicht explizit angegeben werden. Diese Wahl ist die einfachere Option.
- Geben Sie die Amazon-Ion-Klassenpfade in den Feldern `ROW FORMAT SERDE`, `INPUTFORMAT` und `OUTPUTFORMAT` an.

Sie können auch `CREATE TABLE AS SELECT` (CTAS)-Anweisungen verwenden, um Amazon-Ion-Tabellen in Athena zu erstellen. Weitere Informationen finden Sie unter Erstellen von Amazon-Ion-Tabellen mithilfe von CTAS und INSERT INTO.

Als ION GESPEICHERT angeben

Die folgende beispielhafte `CREATE TABLE`-Anweisung verwendet `STORED AS ION` vor der `LOCATION`-Klausel, um eine Tabelle basierend auf Flugdaten im Amazon-Ion-Format zu erstellen. Die `LOCATION`-Klausel gibt den Bucket oder Ordner an, in dem sich die Eingabedateien im Ion-Format befinden. Alle Dateien am angegebenen Speicherort werden gescannt.

```
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/'
```

Angeben der Amazon-Ion-Klassenpfade

Anstatt die `STORED AS ION`-Syntax zu verwenden, können Sie die Ion-Klassenpfadwerte für die `ROW FORMAT SERDE`-, `INPUTFORMAT`- und `OUTPUTFORMAT`-Klauseln wie folgt explizit angeben.

Parameter	Ion-Klassenpfad
`ROW FORMAT SERDE`	`'com.amazon.ionhiveserde.IonHiveSerDe'`
`STORED AS INPUTFORMAT`	`'com.amazon.ionhiveserde.formats.IonInputFormat'`
`OUTPUTFORMAT`	`'com.amazon.ionhiveserde.formats.IonOutputFormat'`

Die folgende DDL-Abfrage verwendet diese Technik, um dieselbe externe Tabelle wie im vorherigen Beispiel zu erstellen.

```
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/'
```

Informationen zu den SerDe-Eigenschaften für `CREATE TABLE`-Anweisungen in Athena finden Sie unter Verwenden von Amazon Ion SerDe Eigenschaften.

Erstellen von Amazon-Ion-Tabellen mithilfe von CTAS und INSERT INTO

Sie können die Anweisungen `CREATE TABLE AS SELECT` (CTAS) und `INSERT INTO` verwenden, um Daten aus einer Tabelle in eine neue Tabelle im Amazon-Ion-Format in Athena zu kopieren oder einzufügen.

Geben Sie in einer CTAS-Abfrage `format='ION'` in der `WITH`-Klausel an, wie im folgenden Beispiel.

```
CREATE TABLE new_table
WITH (format='ION')
AS SELECT * from existing_table
```

Standardmäßig serialisiert Athena Amazon Ion-Ergebnisse im Ion-Binärformat, aber Sie können auch das Textformat verwenden. Um das Textformat zu verwenden, geben Sie `ion_encoding = 'TEXT'` in der CTAS-`WITH`-Klausel an, wie im folgenden Beispiel.

```
CREATE TABLE new_table
WITH (format='ION', ion_encoding = 'TEXT')
AS SELECT * from existing_table
```

Weitere Informationen zu Amazon-Ion-spezifischen Eigenschaften in der CTAS-`WITH`-Klausel finden Sie im folgenden Abschnitt.

CTAS-WITH-Klausel von Amazon-Ion-Eigenschaften

In einer CTAS-Abfrage können Sie die `WITH`-Klausel verwenden, um das Amazon-Ion-Format anzugeben und optional die zu verwendende Amazon-Ion-Codierung und/oder den zu verwendenden Schreibkomprimierungs-Algorithmus anzugeben.

format

Sie können das Schlüsselwort `ION` als Formatoption in der `WITH`-Klausel einer CTAS-Abfrage angeben. Wenn Sie dies tun, verwendet die von Ihnen erstellte Tabelle das Format, das Sie für `IonInputFormat` für Lesevorgänge angeben, und sie serialisiert Daten in dem Format, das Sie für `IonOutputFormat` angeben.

Im folgenden Beispiel wird angegeben, dass die CTAS-Abfrage das Amazon-Ion-Format verwendet.

```
WITH (format='ION')
```

ion_encoding

Optional

Standard: `BINARY`

Werte: `BINARY`, `TEXT`

Gibt an, ob Daten im Amazon-Ion-Binärformat oder im Amazon-Ion-Textformat serialisiert werden. Im folgenden Beispiel wird das Amazon-Ion-Textformat angegeben.

```
WITH (format='ION', ion_encoding='TEXT')
```

write_compression

Optional

Standard: `GZIP`

Werte: `GZIP`, `ZSTD`, `BZIP2`, `SNAPPY`, `NONE`

Gibt den Komprimierungs-Algorithmus an, der zum Komprimieren von Ausgabedateien verwendet werden soll.

Das folgende Beispiel gibt an, dass die CTAS-Abfrage ihre Ausgabe im Amazon-Ion-Format unter Verwendung des Zstandard-Komprimierungs-Algorithmus schreibt.

```
WITH (format='ION', write_compression = 'ZSTD')
```

Weitere Informationen zur Verwendung der Komprimierung in Athena finden Sie unter Athena-Komprimierungs-Support.

Weitere CTAS-Eigenschaften in Athena finden Sie unter CTAS-Tabelleneigenschaften.

Verwenden von Amazon Ion SerDe Eigenschaften

Dieses Thema enthält Informationen zu den SerDe-Eigenschaften für `CREATE TABLE`-Anweisungen in Athena. Weitere Informationen und Beispiele zur Verwendung von Amazon-Ion-SerDe-Eigenschaften finden Sie unter SerDe-Eigenschaften in der Dokumentation zu Amazon Ion Hive SerDe auf GitHub.

Angeben von Amazon Ion SerDe Eigenschaften

Um Eigenschaften für Amazon Ion Hive SerDe in Ihrer `CREATE TABLE`-Anweisung anzugeben, verwenden Sie die `WITH SERDEPROPERTIES`-Klausel. Da `WITH SERDEPROPERTIES` ein Unterfeld

der `ROW FORMAT SERDE`-Klausel ist, müssen Sie zuerst `ROW FORMAT SERDE` und den Hive-SerDe-Klassenpfad angeben, wie die folgende Syntax zeigt.

```
...
ROW FORMAT SERDE
 'com.amazon.ionhiveserde.IonHiveSerDe'
WITH SERDEPROPERTIES (
 'property' = 'value',
 'property' = 'value',
...
)
```

Beachten Sie, dass, obwohl die `ROW FORMAT SERDE`-Klausel erforderlich ist, wenn Sie `WITH SERDEPROPERTIES` verwenden möchten, Sie entweder `STORED AS ION` oder die längere `INPUTFORMAT`- und `OUTPUTFORMAT`-Syntax verwenden können, um das Amazon-Ion-Format anzugeben.

Eigenschaften von Amazon Ion SerDe

Im Folgenden finden Sie die Eigenschaften von Amazon Ion SerDe, die in `CREATE TABLE`-Anweisungen in Athena verwendet werden können.

ion.encoding

Optional

Standard: `BINARY`

Werte: `BINARY`, `TEXT`

Diese Eigenschaft gibt an, ob neu hinzugefügte Werte als Amazon-Ion-Binär oder Amazon-Ion-Textformat serialisiert werden.

Das folgende Beispiel für die SerDe-Eigenschaft gibt das Amazon-Ion-Textformat an.

```
'ion.encoding' = 'TEXT'
```

ion.fail_on_overflow

Optional

Standard: `true`

Werte: `true`, `false`

Amazon Ion erlaubt beliebig große numerische Typen, während Hive dies nicht tut. Standardmäßig schlägt SerDe fehl, wenn der Amazon-Ion-Wert nicht in die Hive-Spalte passt, aber Sie können die `fail_on_overflow`-Konfigurationsoption verwenden, um den Wert überlaufen zu lassen, anstatt ihn fehlschlagen zu lassen.

Diese Eigenschaft kann entweder auf Tabellen- oder Spaltenebene festgelegt werden. Um es auf Tabellenebene anzugeben, geben Sie `ion.fail_on_overflow` wie im folgenden Beispiel an. Dies legt das Standardverhalten für alle Spalten fest.

```
'ion.fail_on_overflow' = 'true'
```

Um eine bestimmte Spalte zu steuern, geben Sie den Spaltennamen zwischen `ion` und `fail_on_overflow` an, getrennt durch Punkte, wie im folgenden Beispiel.

```
'ion.<column>.fail_on_overflow' = 'false'
```

ion.path_extractor.case_sensitive

Optional

Standard: `false`

Werte: `true`, `false`

Bestimmt, ob bei Amazon-Ion-Feldnamen die Groß-/Kleinschreibung beachtet werden soll. Bei `false` ignoriert der SerDe die Groß-/Kleinschreibung beim Analysieren von Amazon-Ion-Feldnamen.

Angenommen, Sie haben ein Hive-Tabellenschema, das ein Feld `alias` in Kleinbuchstaben definiert, und ein Amazon Ion-Dokument mit sowohl einem `alias`-Feld als auch einem `ALIAS`-Feld, wie im folgenden Beispiel.

```
-- Hive Table Schema
alias: STRING

-- Amazon Ion Document
{ 'ALIAS': 'value1'}
```

```
{ 'alias': 'value2'}
```

Das folgende Beispiel zeigt SerDe-Eigenschaften und die resultierende extrahierte Tabelle, wenn die Groß-/Kleinschreibung auf `false` eingestellt ist:

```
-- Serde properties
'ion.alias.path_extractor' = '(alias)'
'ion.path_extractor.case_sensitive' = 'false'

--Extracted Table
| alias    |
|----------|
| "value1" |
| "value2" |
```

Das folgende Beispiel zeigt SerDe-Eigenschaften und die resultierende extrahierte Tabelle, wenn die Groß-/Kleinschreibung auf `true` eingestellt ist:

```
-- Serde properties
'ion.alias.path_extractor' = '(alias)'
'ion.path_extractor.case_sensitive' = 'true'

--Extracted Table
| alias    |
|----------|
| "value2" |
```

Im zweiten Fall wird `value1` für das `ALIAS`-Feld ignoriert, wenn die Groß-/Kleinschreibung auf `true` eingestellt ist und der Pfad-Extraktor als `alias` angegeben ist.

ion.*<column>*.path_extractor

Optional

Standard: Nicht angegeben

Werte: Zeichenfolge mit Suchpfad

Erstellt einen Pfad-Extraktor mit dem angegebenen Suchpfad für die angegebene Spalte. Pfad-Extraktoren ordnen Amazon-Ion-Felder Hive-Spalten zu. Wenn keine Pfad-Extraktoren angegeben werden, erstellt Athena dynamisch Pfad-Extraktoren zur Laufzeit basierend auf Spaltennamen.

Im folgenden Beispiel wird der Pfad-Extraktor `example_ion_field` zum `example_hive_column` zugeordnet.

```
'ion.example_hive_column.path_extractor' = '(example_ion_field)'
```

Weitere Informationen zu Pfad-Extraktoren und Suchpfaden finden Sie unter Verwenden von Pfad-Extraktoren.

ion.timestamp.serialization_offset

Optional

Standard: `'Z'`

Werte: `OFFSET`, wobei `OFFSET` als *<signal>*`hh:mm` dargestellt wird. Beispielwerte: `01:00`, `+01:00`, `-09:30`, `Z` (UTC, wie 00:00 Uhr)

Im Gegensatz zu Apache-Hive-Zeitstempeln, die keine integrierte Zeitzone haben und als Offset von der UNIX-Epoche gespeichert werden, haben Amazon-Ion-Zeitstempel einen Offset. Verwenden Sie diese Eigenschaft, um den Offset anzugeben, wenn Sie auf Amazon Ion serialisieren.

Im folgenden Beispiel wird ein Offset von einer Stunde hinzugefügt.

```
'ion.timestamp.serialization_offset' = '+01:00'
```

ion.serialize_null

Optional

Standard: `OMIT`

Werte: `OMIT`, `UNTYPED`, `TYPED`

Der Amazon Ion SerDe kann so konfiguriert werden, dass er Spalten mit Nullwerten serialisiert oder weglässt. Sie können wählen, stark typisierte Nullen auszuschreiben (`TYPED`) oder nicht typisierte Nullen (`UNTYPED`) enthalten. Stark typisierte Nullen werden basierend auf der Standardzuordnung des Typs Amazon Ion zu Hive bestimmt.

Das folgende Beispiel gibt stark typisierte Nullen an.

```
'ion.serialize_null'='TYPED'
```

ion.ignore_malformed

Optional

Standard: `false`

Werte: `true`, `false`

Wenn `true` fehlerhafte Einträge oder die gesamte Datei ignoriert, wenn der SerDe sie nicht lesen kann. Weitere Informationen finden Sie unter Fehlerhafte ignorieren in der Dokumentation auf GitHub.

ion.*<column>*.serialize_as

Optional

Standard: Standardtyp für die Spalte.

Werte: Zeichenfolge mit Amazon-Ion-Typ

Bestimmt den Amazon-Ion-Datentyp, in dem ein Wert serialisiert wird. Da Amazon-Ion- und Hive-Typen nicht immer über eine direkte Zuordnung verfügen, haben einige Hive-Typen mehrere gültige Datentypen für die Serialisierung. Verwenden Sie diese Eigenschaft, um Daten als nicht standardmäßigen Datentyp zu serialisieren. Weitere Informationen zum Typmapping finden Sie auf der Seite Amazon-Ion-Typmapping auf GitHub.

Standardmäßig werden binäre Hive-Spalten als Amazon-Ion-Blobs serialisiert, sie können aber auch als Amazon-Ion-Clob (Character Large Object) serialisiert werden. Im folgenden Beispiel wird die Spalte `example_hive_binary_column` als Clob serialisiert.

```
'ion.example_hive_binary_column.serialize_as' = 'clob'
```

Verwenden von Pfad-Extraktoren

Amazon Ion ist ein Dateiformat im Dokumentstil, aber Apache Hive ist ein flaches Säulenformat. Sie können spezielle Amazon-Ion-SerDe-Eigenschaften namens `path extractors` verwenden, um zwischen den beiden Formaten zuzuordnen. Pfad-Extraktoren flachen das hierarchische Amazon-Ion-Format ab, ordnen Amazon-Ion-Werte Hive-Spalten zu und können zum Umbenennen von Feldern verwendet werden.

Athena kann die Extraktoren für Sie generieren, aber Sie können bei Bedarf auch Ihre eigenen Extraktoren definieren.

Generierte Pfad-Extraktoren

Standardmäßig sucht Athena nach Amazon-Ion-Werten der obersten Ebene, die Hive-Spaltennamen entsprechen, und erstellt zur Laufzeit Pfad-Extraktoren basierend auf diesen übereinstimmenden Werten. Wenn Ihr Amazon-Ion-Datenformat mit dem Hive-Tabellenschema übereinstimmt, generiert Athena die Extraktoren dynamisch für Sie und Sie müssen keine zusätzlichen Pfad-Extraktoren hinzufügen. Diese Standard-Pfad-Extraktoren werden nicht in den Tabellen-Metadaten gespeichert.

Im folgenden Beispiel wird gezeigt, wie Athena Extraktoren basierend auf dem Spaltennamen generiert.

```
-- 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/'
```

Die folgenden Beispiel-Extraktoren werden von Athena generiert. Der erste extrahiert das `identification`-Feld in die `identification`-Spalte und der zweite extrahiert das `alias`-Feld in die `alias`-Spalte.

```
'ion.identification.path_extractor' = '(identification)'
'ion.alias.path_extractor' = '(alias)'
```

Das folgende Beispiel zeigt die extrahierte Tabelle.

```
|                  identification                    |  alias   |
|----------------------------------------------------|----------|
|{["name", "driver_license"],["John Smith", "XXXX"]} | "Johnny" |
```

Angeben eigener Pfad-Extraktoren

Wenn Ihre Amazon-Ion-Felder nicht ordentlich Hive-Spalten zugeordnet werden, können Sie Ihre eigenen Pfad-Extraktoren angeben. Verwenden Sie in der `WITH SERDEPROPERTIES`-Klausel Ihrer `CREATE TABLE`-Anweisung die folgende Syntax.

```
WITH SERDEPROPERTIES (
   "ion.path_extractor.case_sensitive" = "<Boolean>",
   "ion.<column_name>.path_extractor" = "<path_extractor_expression>"
)
```

> **Note**
>
> Standardmäßig wird nicht zwischen Groß- und Kleinschreibung unterschieden. Um diese Einstellung zu überschreiben, setzen Sie die Eigenschaft ion.path_extractor.case_sensitive SerDe auf `true`.

Verwenden von Suchpfaden in Pfad-Extraktoren

Die SerDe Eigenschaftssyntax für den Pfad-Extraktor enthält eine *<path_extractor_expression>*:

```
"ion.<column_name>.path_extractor" = "<path_extractor_expression>"
```

Sie können den *<path_extractor_expression>* verwenden, um einen Suchpfad anzugeben, der das Amazon-Ion-Dokument parst und passende Daten findet. Der Suchpfad ist in Klammern eingeschlossen und kann eine oder mehrere der folgenden durch Leerzeichen getrennten Komponenten enthalten.

- Platzhalter – Entspricht allen Werten.
- Index – Entspricht dem Wert beim angegebenen numerischen Index. Die Indizes sind nullbasiert.
- Text – Entspricht allen Werten, deren Feldnamen übereinstimmen, dem angegebenen Text entsprechen.

- Annotationen – Entspricht Werten, die durch eine umbrochene Pfadkomponente angegeben wurden, für die die Anmerkungen angegeben wurden.

Das folgende Beispiel zeigt ein Amazon-Ion-Dokument und einige Beispiel-Suchpfade.

```
-- 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"
```

Beispiele für Extraktoren

Felder abflachen und umbenennen

Das folgende Beispiel zeigt eine Reihe von Suchpfaden, die Felder abflachen und umbenennen. Im Beispiel werden Suchpfade verwendet, um Folgendes durchzuführen:

- Ordnen Sie die `nickname`-Spalte dem `alias`-Feld zu
- Ordnen Sie die `name`-Spalte dem `name`-Unterfeld zu, das sich in der `identification`-Struktur befindet.

Es folgt das Beispiel für Amazon-Ion-Dokument.

```
-- Example Amazon Ion Document
{
    identification: {
        name: "John Smith",
        driver_license: "XXXX"
    },
```

```
    alias: "Johnny"
}
```

Das Folgende ist die beispielhafte `CREATE TABLE`-Anweisung, die die Pfad-Extraktoren definiert.

```
-- 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/'
```

Das folgende Beispiel zeigt die extrahierten Dateien.

```
-- Extracted Table
| name         |   nickname   |
|--------------|--------------|
| "John Smith" |  "Johnny"    |
```

Weitere Informationen zu Suchpfaden und weitere Beispiele für Suchpfade finden Sie auf der Seite Ion-Java-Pfad-Extraktion auf GitHub.

Extrahieren von Flugdaten in das Textformat

Die folgende `CREATE TABLE`-Beispielabfrage verwendet `WITH SERDEPROPERTIES`, um Pfad-Extraktoren zum Extrahieren von Flugdaten hinzuzufügen und die Ausgabecodierung als Amazon-Ion-Text anzugeben. Im Beispiel wird die `STORED AS ION`-Syntax verwendet.

```
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/'
```

Avro SerDe

SerDe-Name

Avro SerDe

Name der Bibliothek

org.apache.hadoop.hive.serde2.avro.AvroSerDe

Beispiele

Athena unterstützt aus Sicherheitsgründen nicht die Verwendung von `avro.schema.url` zur Angabe des Tabellenschemas. Verwenden Sie `avro.schema.literal`. Sie können Apache `avro-tools-<version>.jar` mit dem Parameter `getschema` verwenden, um ein Schema aus Daten im Avro-Format zu extrahieren. Dadurch erhalten Sie ein Schema, das Sie in Ihrer `WITH SERDEPROPERTIES`-Anweisung nutzen können. Beispiele:

```
java -jar avro-tools-1.8.2.jar getschema my_data.avro
```

Die Datei `avro-tools-<version>.jar` befindet sich im Unterverzeichnis `java` Ihrer installierten Avro-Version. Informationen zum Herunterladen von Avro finden Sie unter Apache Avro Releases. Informationen zum direkten Herunterladen der Apache Avro-Tools finden Sie unter Apache Avro Tools Maven Repository.

Nachdem Sie das Schema abgerufen haben, verwenden Sie eine `CREATE TABLE`-Anweisung, um eine Athena-Tabelle basierend auf den zugrunde liegenden Avro-Daten zu erstellen, die in Amazon S3 gespeichert sind. Um Avro SerDe anzugeben, verwenden Sie `ROW FORMAT SERDE`

`'org.apache.hadoop.hive.serde2.avro.AvroSerDe'`. Wie im folgenden Beispiel gezeigt, müssen Sie das Schema mithilfe der Klausel `WITH SERDEPROPERTIES` angeben, zusätzlich zur Angabe der Spaltennamen und der entsprechenden Datentypen für die Tabelle.

> **Note**
>
> Ersetzen Sie *myregion* in `s3://athena-examples-`*myregion*`/path/to/data/` durch den Regionsbezeichner, in dem Sie Athena ausführen, z. B. `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/';
```

Führen Sie die `MSCK REPAIR TABLE`-Anweisung auf der Tabelle aus, um die Partitionsmetadaten zu aktualisieren.

```
MSCK REPAIR TABLE flights_avro_example;
```

Fragen Sie die am häufigsten genutzten 10 Abflugstädte, gemessen an der Anzahl der Abflüge, ab.

```
SELECT origin, count(*) AS total_departures
FROM flights_avro_example
WHERE year >= '2000'
GROUP BY origin
ORDER BY total_departures DESC
LIMIT 10;
```

> **Note**
>
> Die Flugtabellendaten stammen aus Flügen, die vom US-Verkehrsministerium, Bureau of Transportation Statistics, bereitgestellt werden. Entsättigt vom Original.

Grok SerDe

Beim Logstash Grok SerDe handelt es sich um eine Bibliothek mit einer Reihe spezifischer Muster für die Deserialisierung unstrukturierter Textdaten (in der Regel Protokolle). Jedes Grok-Muster ist ein benannter regulärer Ausdruck. Sie können diese Deserialisierungsmuster identifizieren und bei Bedarf wiederverwenden. Dies macht die Verwendung von Grok einfacher als die Verwendung regulärer Ausdrücke. Grok bietet eine Reihe vordefinierter Muster. Sie können auch benutzerdefinierte Muster erstellen.

Um beim Erstellen einer Tabelle in Athena den Grok SerDe anzugeben, verwenden Sie die Klausel `ROW FORMAT SERDE 'com.amazonaws.glue.serde.GrokSerDe'`, gefolgt von der Klausel `WITH SERDEPROPERTIES`, die die Muster angibt, mit denen eine Übereinstimmung in Ihren Daten erzielt werden soll. Dabei gilt:

- Der `input.format`-Ausdruck definiert die Muster, mit denen in den Daten eine Übereinstimmung erzielt werden soll. Er ist erforderlich.
- Der `input.grokCustomPatterns`-Ausdruck definiert ein benanntes benutzerdefiniertes Muster, das Sie anschließend innerhalb des `input.format`-Ausdrucks verwenden können. Der Schritt ist optional. Um mehrere Mustereinträge im `input.grokCustomPatterns`-Ausdruck einzuschließen, verwenden Sie das Zeilenumbruchszeichen (\n), um diese zu trennen, wie im Folgenden dargestellt: `'input.grokCustomPatterns'='INSIDE_QS ([^\"]*)\nINSIDE_BRACKETS ([^\\]]*)'`.
- Die Klauseln `STORED AS INPUTFORMAT` und `OUTPUTFORMAT` sind erforderlich.

- Die Klausel `LOCATION` gibt einen Amazon-S3-Bucket an, der mehrere Datenobjekte enthalten kann. Alle Datenobjekte im Bucket werden deserialisiert, um die Tabelle zu erstellen.

Beispiele

Diese Beispiele basieren auf der Liste vordefinierter Grok-Muster. Weitere Informationen finden Sie auf der Seite zu vordefinierten Mustern.

Beispiel 1

Dieses Beispiel verwendet Quelldaten von in `s3://mybucket/groksample/` gespeicherten Postfix-E-Mail-Protokolleinträgen.

```
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>
```

Die folgende Anweisung erstellt eine Tabelle namens `mygroktable` in Athena aus den Quelldaten. Dies erfolgt unter Verwendung eines benutzerdefinierten Musters und der von Ihnen angegebenen vordefinierten Muster.

```
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://mybucket/groksample/';
```

Beginnen Sie mit einem einfachen Muster, beispielsweise mit `%{NOTSPACE:column}`, um zuerst die Spalten zuzuordnen, und spezialisieren Sie diese dann bei Bedarf.

Beispiel 2

Im folgenden Beispiel erstellen Sie eine Abfrage für Log4j-Protokolle. Die Einträge der Beispielprotokolle weisen folgendes Format auf:

```
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
```

So fragen Sie diese Protokolldaten ab:

- Fügen Sie das Grok-Muster für jede Spalte zum `input.format` hinzu. Fügen Sie beispielsweise für `timestamp` `%{TIMESTAMP_ISO8601:timestamp}` hinzu. Fügen Sie für `loglevel` `%{LOGLEVEL:loglevel}` hinzu.
- Stellen Sie sicher, dass das Muster in `input.format` exakt mit dem Format des Protokolls übereinstimmt, indem Sie die Bindestriche (`-`) und die Kommata zuweisen, die die Einträge im Protokollformat trennen.

  ```
  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://mybucket/samples/';
```

Beispiel 3

Das folgende Beispiel für die Abfrage von Amazon-S3-Protokollen zeigt den `'input.grokCustomPatterns'`-Ausdruck, der zwei Mustereinträge enthält, die durch das Zeilenumbruchszeichen (\n) getrennt werden. Dies ist im folgenden Ausschnitt der Beispielabfrage dargestellt: `'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://bucket-for-service-logs/s3_access/'
```

JSON-SerDe-Bibliotheken

In Athena können Sie SerDe-Bibliotheken verwenden, um JSON-Daten zu deserialisieren. Die Deserialisierung wandelt die JSON-Daten so um, dass sie in ein anderes Format wie Parquet oder ORC serialisiert (ausgeschrieben) werden können.

- Die native Hive JSON SerDe
- Der OpenX JSON SerDe
- Der Amazon Ion Hive SerDe

> **Note**
>
> Die Hive- und OpenX-Bibliotheken erwarten, dass sich JSON-Daten in einer einzelnen Zeile befinden (nicht formatiert), wobei Datensätze durch ein Zeilenumbruchzeichen getrennt sind. Der Amazon Ion Hive SerDe hat diese Anforderung nicht und kann als Alternative verwendet werden, da das Ion-Datenformat eine Obermenge von JSON ist.

Bibliotheksnamen

Nutzen Sie einen der Folgenden:

org.apache.hive.hcatalog.data.JsonSerDe

org.openx.data.jsonserde.JsonSerDe

com.amazon.ionhiveserde.IonHiveSerDe

Hive JSON SerDe

Der Hive JSON SerDe wird häufig verwendet, um JSON-Daten wie etwa Ereignisse zu verarbeiten. Diese Ereignisse werden als einzeilige Zeichenfolgen aus JSON-codiertem Text dargestellt, die jeweils durch eine neue Zeile voneinander getrennt sind. Der Hive JSON SerDe erlaubt keine doppelten Schlüssel in `map`- oder `struct`-Schlüsselnamen.

> **Note**
>
> Das SerDe erwartet, dass sich jedes JSON-Dokument auf einer einzigen Textzeile befindet, ohne Zeilenabschlusszeichen, die die Felder im Datensatz trennen. Wenn der JSON-Text im hübschen Druckformat vorliegt, erhalten Sie möglicherweise eine Fehlermeldung wie HIVE_CURSOR_ERROR: Zeile ist kein gültiges JSON-Objekt oder HIVE_CURSOR_ERROR: JsonParseException: Unerwartetes Ende der Eingabe: Erwartete Schließmarkierung für OBJEKT, wenn Sie versuchen, die Tabelle abzufragen, nachdem Sie dies erstellt haben. Weitere Informationen finden Sie unter JSON-Datendatei in der OpenX-Serde-Dokumentation auf GitHub.

In der folgenden DDL-Beispielanweisung wird der Hive JSON SerDe verwendet, um eine Tabelle basierend auf Beispiel-Online-Werbedaten zu erstellen. Ersetzen Sie in der `LOCATION`-Klausel *`myregion`* in `s3://`*`myregion`*`.elasticmapreduce/samples/hive-ads/tables/impressions` durch die Kennung der Region, in der Sie Athena ausführen (zum Beispiel `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://myregion.elasticmapreduce/samples/hive-ads/tables/impressions';
```

Angeben von Zeitstempelformaten mit dem Hive JSON SerDe

Um Zeitstempelwerte aus der Zeichenfolge zu analysieren, können Sie das Unterfeld `WITH SERDEPROPERTIES` zur Klausel `ROW FORMAT SERDE` hinzufügen und es verwenden, um den Parameter `timestamp.formats` anzugeben. Geben Sie im Parameter eine durch Kommas getrennte Liste mit einem oder mehreren Zeitstempelmustern an, wie im folgenden Beispiel:

```
...
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")
...
```

Weitere Informationen finden Sie unter Zeitstempel in der Apache-Hive-Dokumentation.

Laden der Tabelle zum Abfragen

Führen Sie nach dem Erstellen der Tabelle MSCK REPAIR TABLE aus, um die Tabelle zu laden und sie von Athena aus abfragen zu können:

```
MSCK REPAIR TABLE impressions
```

Abfragen von CloudTrail-Protokollen

Sie können den Hive-JSON-SerDe verwenden, um CloudTrail-Protokolle abzufragen. Weitere Informationen und `CREATE TABLE`-Beispielanweisungen finden Sie unter Abfragen von AWS CloudTrail-Protokollen.

OpenX JSON SerDe

Wie Hive JSON SerDe können Sie OpenX JSON zur Verarbeitung von JSON-Daten verwenden. Die Daten werden auch als einzeilige Zeichenfolgen aus JSON-codiertem Text dargestellt, die durch eine neue Zeile getrennt sind. Wie der Hive JSON SerDe erlaubt der OpenX JSON SerDe keine doppelten Schlüssel in `map`- oder `struct`-Schlüsselnamen.

> **Note**
>
> Das SerDe erwartet, dass sich jedes JSON-Dokument auf einer einzigen Textzeile befindet, ohne Zeilenabschlusszeichen, die die Felder im Datensatz trennen. Wenn der JSON-Text im hübschen Druckformat vorliegt, erhalten Sie möglicherweise eine Fehlermeldung wie HIVE_CURSOR_ERROR: Zeile ist kein gültiges JSON-Objekt oder HIVE_CURSOR_ERROR: JsonParseException: Unerwartetes Ende der Eingabe: Erwartete Schließmarkierung für OBJEKT, wenn Sie versuchen, die Tabelle abzufragen, nachdem Sie dies erstellt haben. Weitere Informationen finden Sie unter JSON-Datendatei in der OpenX-SerDe-Dokumentation auf GitHub.

Optionale Eigenschaften

Im Gegensatz zu Hive JSON SerDe verfügt OpenX JSON SerDe auch über die folgenden optionalen SerDe-Eigenschaften, die für die Behebung von Daten-Inkonsistenzen nützlich sein können.

ignore.malformed.json

Optional. Bei Festlegung auf `TRUE` können Sie eine fehlerhafte JSON-Syntax überspringen. Der Standardwert ist `FALSE`.

dots.in.keys

Optional. Der Standardwert ist `FALSE`. Bei Festlegung auf `TRUE` kann der SerDe die Punkte in Schlüsselnamen durch Unterstriche ersetzen. Wenn der JSON-Datensatz beispielsweise einen Schlüssel mit dem Namen `"a.b"` enthält, können Sie diese Eigenschaft verwenden, um den Spaltennamen als `"a_b"` in Athena zu definieren. Standardmäßig (ohne diesen SerDe) lässt Athena keine Punkte in Spaltennamen zu.

case.insensitive

Optional. Der Standardwert ist `TRUE`. Bei Festlegung auf `TRUE` wandelt der SerDe alle Großbuchstaben in Kleinbuchstaben um.

Verwenden Sie zur Nutzung von Schlüsselnamen unter Beachtung der Groß-/Kleinschreibung in Ihren Daten `WITH SERDEPROPERTIES ("case.insensitive"= FALSE;)`. Geben Sie dann für jeden Schlüssel, der nicht bereits vollständig aus Kleinbuchstaben besteht, ein Mapping vom Spaltennamen zum Eigenschaftsnamen mit der folgenden Syntax an:

```
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES ("case.insensitive" = "FALSE", "mapping.userid" = "userId")
```

Wenn Sie zwei Schlüssel wie `URL` und `Url` haben die gleich sind, wenn sie aus Kleinbuchstaben bestehen, kann ein Fehler wie der folgende auftreten:

HIVE_CURSOR_ERROR: Zeile ist kein gültiges JSON-Objekt - JSONException: Doppelter Schlüssel „url“

Um dies zu beheben, setzen Sie die `case.insensitive`-Eigenschaft auf `FALSE` und ordnen Sie die Schlüssel verschiedenen Namen zu, wie im folgenden Beispiel gezeigt:

```
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES ("case.insensitive" = "FALSE", "mapping.url1" = "URL",
 "mapping.url2" = "Url")
```

Mapping

Optional. Ordnet Spaltennamen zu JSON-Schlüsseln zu, die nicht mit den Spaltennamen identisch sind. Der `mapping`-Parameter ist nützlich, wenn die JSON-Daten Schlüssel enthalten, die Schlüsselwörter sind. Wenn Sie beispielsweise einen JSON-Schlüssel mit dem Namen `timestamp` haben, verwenden Sie die folgende Syntax, um den Schlüssel einer Spalte mit dem Namen `ts` zuzuordnen:

```
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES ("mapping.ts" = "timestamp")
```

Zuordnung verschachtelter Feldnamen mit Doppelpunkten zu Hive-kompatiblen Namen

Wenn Sie über einen Feldnamen mit Doppelpunkten innerhalb eines `struct` verfügen, können Sie die `mapping`-Eigenschaft verwenden, um das Feld einem Hive-kompatiblen Namen zuzuordnen. Wenn Ihre Spaltentypdefinitionen beispielsweise `my:struct:field:string` enthalten, können Sie die Definition `my_struct_field:string` zuordnen, indem Sie den folgenden Eintrag in `WITH SERDEPROPERTIES` einfügen:

```
("mapping.my_struct_field" = "my:struct:field")
```

Im folgenden Beispiel wird die zugehörige `CREATE TABLE`-Anweisung gezeigt.

```
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")
```

Beispiel: Werbedaten

In der folgenden DDL-Anweisung wird der OpenX JSON SerDe verwendet, um eine Tabelle zu erstellen, die auf denselben Beispiel-Online-Werbedaten basiert, die im Beispiel für Hive JSON SerDe verwendet wurden. Ersetzen Sie in der `LOCATION`-Klausel *`myregion`* durch die Kennung der Region, in der Sie Athena ausführen.

```
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.openx.data.jsonserde.JsonSerDe'
LOCATION 's3://myregion.elasticmapreduce/samples/hive-ads/tables/impressions';
```

Beispiel: Deserialisierung von verschachteltem JSON

Sie können die JSON-SerDes verwenden, um komplexere JSON-codierte Daten zu analysieren. Dies erfordert die Verwendung von `CREATE TABLE`-Anweisungen, die `struct`- und `array`- Elemente verwenden, um verschachtelte Strukturen darzustellen.

Im folgenden Beispiel wird eine Athena-Tabelle aus JSON-Daten mit verschachtelten Strukturen erstellt. Um JSON-kodierte Daten in Athena zu analysieren, stellen Sie sicher, dass sich jedes JSON-Dokument auf einer eigenen Zeile befindet, die durch eine neue Zeile abgetrennt ist.

Bei diesem Beispiel werden JSON-kodierte Daten mit folgender Struktur vorausgesetzt:

```
{
"DocId": "AWS",
"User": {
        "Id": 1234,
        "Username": "bob1234",
        "Name": "Bob",
"ShippingAddress": {
"Address1": "123 Main St.",
"Address2": null,
"City": "Seattle",
"State": "WA"
   },
"Orders": [
   {
     "ItemId": 6789,
     "OrderDate": "11/11/2017"
   },
   {
     "ItemId": 4352,
     "OrderDate": "12/12/2017"
   }
  ]
 }
}
```

Die folgende `CREATE TABLE`-Anweisung verwendet OpenX-JsonSerDe mit den Sammlungsdatentypen `struct` und `array`, um Objektgruppen einzurichten. Jedes JSON-Dokument ist in einer eigenen Zeile aufgeführt und diese sind durch eine neue Zeile voneinander getrennt. Um Fehler zu vermeiden, enthalten die abgefragten Daten keine doppelten Schlüssel in `struct`- oder Map-Schlüsselnamen.

```
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://mybucket/myjsondata/';
```

Weitere Ressourcen

Weitere Informationen zum Arbeiten mit JSON und verschachteltem JSON in Athena finden Sie in den folgenden Ressourcen:

- Erstellen von Tabellen in Amazon Athena aus verschachteltem JSON und Mappings mit JSONSerDe (AWS Big Data Blog)
- Ich erhalte Fehler, wenn ich versuche, JSON-Daten in Amazon Athena zu lesen (AWS Knowledge Center-Artikel)
- hive-json-schema (GitHub) – Tool in Java, das `CREATE TABLE`-Anweisungen aus Beispiel-JSON-Dokumenten generiert. Die generierten `CREATE TABLE` Anweisungen verwenden den OpenX JSON Serde.

LazySimpleSerDe für CSV- und TSV-Dateien sowie für benutzerdefinierte, durch Trennzeichen getrennte Dateien

Diese SerDe-Angabe ist optional. Dies ist der SerDe für Daten in CSV-, TSV- und benutzerdefinierten, durch Trennzeichen getrennten Formaten, die Athena standardmäßig verwendet. Dieser SerDe wird verwendet, wenn Sie keinen SerDe, sondern nur `ROW FORMAT DELIMITED` angeben. Verwenden Sie diesen SerDe, wenn Ihre Daten keine in Anführungszeichen eingeschlossenen Werte enthalten.

Referenzdokumentation zu LazySimpleSerDe finden Sie im Abschnitt Hive SerDe des Apache-Hive-Entwicklerhandbuchs.

Name der Bibliothek

Der Klassebibliotheksname für den LazySimpleSerDe ist `org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe`. Informationen zur LazySimpleSerDe-Klasse finden Sie unter LazySimpleSerDe.java auf GitHub.com.

Ignorieren von Kopfzeilen

Um Header in Ihren Daten beim Definieren einer Tabelle zu ignorieren, können Sie die Tabelleneigenschaft `skip.header.line.count` wie im folgenden Beispiel verwenden.

```
TBLPROPERTIES ("skip.header.line.count"="1")
```

Beispiele finden Sie unter `CREATE TABLE`-Anweisungen in Abfragen von Amazon-VPC-Flow-Protokollen und CloudFront Amazon-Logs abfragen.

CSV-Beispiel

Das folgende Beispiel zeigt, wie Sie mit `LazySimpleSerDe` eine Tabelle in Athena aus CSV-Daten erstellen. Um Dateien mit benutzerdefinierten Trennzeichen mit diesem SerDe zu deserialisieren, folgen Sie dem Muster in den Beispielen, verwenden Sie jedoch die `FIELDS TERMINATED BY`-Klausel, um ein anderes Einzelzeichen-Trennzeichen anzugeben. LazySimpleSerDe unterstützt keine Trennzeichen mit mehreren Zeichen.

> **Note**
>
> Ersetzen Sie *myregion* in `s3://athena-examples-myregion/path/to/data/` durch den Regionsbezeichner, in dem Sie Athena ausführen, z. B. `s3://athena-examples-us-west-1/path/to/data/`.

Verwenden Sie die `CREATE TABLE`-Anweisung zum Erstellen einer Athena-Tabelle basierend auf den in Amazon S3 gespeicherten Daten in CSV.

```
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/';
```

Führen Sie die `MSCK REPAIR TABLE`-Anweisung immer dann aus, wenn eine neue Partition zur Tabelle hinzugefügt wurde, um die Partitionsmetadaten zu aktualisieren:

```
MSCK REPAIR TABLE flight_delays_csv;
```

Fragen Sie die 10 Routen ab, bei denen es zu einer Verzögerung von mehr als 1 Stunde kam:

```
SELECT origin, dest, count(*) as delays
FROM flight_delays_csv
WHERE depdelayminutes > 60
GROUP BY origin, dest
ORDER BY 3 DESC
LIMIT 10;
```

> **Note**
>
> Die Flugtabellendaten stammen aus Flügen, die vom US-Verkehrsministerium, Bureau of Transportation Statistics, bereitgestellt werden. Entsättigt vom Original.

TSV-Beispiel

Um eine Athena-Tabelle aus in Amazon S3 gespeicherten TSV-Daten zu erstellen, verwenden Sie `ROW FORMAT DELIMITED` und spezifizieren Sie `\t` als Tabulator-Feldtrennzeichen, `\n` als Zeilentrennzeichen und `\` als Escape-Zeichen. Im folgenden Beispiel wird diese Syntax dargestellt. An diesem `athena-examples`-Ort sind keine Beispiel-TSV-Flugdaten verfügbar, aber wie bei der CSV-Tabelle würden Sie `MSCK REPAIR TABLE` ausführen, um die Partitions-Metadaten jedes Mal zu aktualisieren, wenn eine neue Partition hinzugefügt wird.

```
...
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
ESCAPED BY '\\'
LINES TERMINATED BY '\n'
...
```

OpenCSVSerDe für CSV-Verarbeitung

Wenn Sie eine Athena-Tabelle für CSV-Daten erstellen, bestimmen Sie die zu verwendende SerDe basierend auf den Wertetypen, die Ihre Daten enthalten:

- Wenn Ihre Daten Werte in doppelten Anführungszeichen (") enthalten, können Sie OpenCSV SerDe verwenden, um die Werte in Athena zu deserialisieren. Wenn Ihre Daten keine Werte in doppelten Anführungszeichen (") enthalten, müssen Sie keine SerDe angeben. In diesem Fall verwendet Athena den Standard `LazySimpleSerDe`. Weitere Informationen finden Sie unter LazySimpleSerDe für CSV- und TSV-Dateien sowie für benutzerdefinierte, durch Trennzeichen getrennte Dateien.
- Wenn Ihre Daten numerische `TIMESTAMP`-UNIX-Werte enthalten (z. B. `1579059880000`), verwenden Sie OpenCSVSerDe. Wenn Ihre Daten das `java.sql.Timestamp`-Format verwenden, verwenden Sie die LazySimpleSerDe.

CSV SerDe (OpenCSVSerDe)

Die OpenCSV SerDe hat die folgenden Eigenschaften für Zeichenfolgendaten:

- Verwendet doppelte Anführungszeichen (") als Standard-Anführungszeichen und ermöglicht es Ihnen, Trennzeichen, Anführungszeichen und Escape-Zeichen wie die Folgenden anzugeben:

```
WITH SERDEPROPERTIES ("separatorChar" = ",", "quoteChar" = "`", "escapeChar" = "\\" )
```

- Es ist kein direktes Escape für `\t` oder `\n` möglich. Um ein Escape dafür durchzuführen, verwenden Sie `"escapeChar" = "\\"`. Sehen Sie sich dazu auch das Beispiel in diesem Thema an.
- Eingebettete Zeilenumbrüche in CSV-Dateien werden nicht unterstützt.

Für andere Datentypen als `STRING` verhält sich OpenCSVSerDe wie folgt:

- Erkennt `BOOLEAN`-, `BIGINT`-, `INT`- und `DOUBLE`-Datentypen.
- Erkennt keine leeren oder Nullwerte in Spalten, die als numerischer Datentyp definiert sind, und belässt sie als `string`. Eine Problemumgehung besteht darin, die Spalte mit den Nullwerten als `string` zu erstellen und dann mit `CAST` das Feld in einer Abfrage in einen numerischen Datentyp zu konvertieren, wobei der Standardwert `0` für Nullwerte bereitgestellt wird. Weitere Informationen finden Sie unter Wenn ich CSV-Daten in Athena abfrage, erhalte ich den Fehler HIVE_BAD_DATA: Fehler beim Parsen des Feldwerts im AWS-Wissenscenter.
- Erkennt für Spalten, die in Ihrer `CREATE TABLE`-Anweisung mit dem Datentyp `timestamp` angegeben sind, `TIMESTAMP`-Daten, wenn sie im numerischen UNIX-Format in Millisekunden angegeben sind, wie z. B. `1579059880000`.

  - Das OpenCSVSerDe unterstützt kein `TIMESTAMP` im JDBC-kompatiblen `java.sql.Timestamp`-Format, wie z. B. `"YYYY-MM-DD HH:MM:SS.fffffffff"` (9 Dezimalstellen).
- Erkennt für Spalten, die in Ihrer `CREATE TABLE`-Anweisung mit dem Datentyp `DATE` angegeben sind, Werte als Datumsangaben, wenn die Werte die Anzahl der Tage darstellen, die seit dem 1. Januar 1970 verstrichen sind. Beispielsweise wird der Wert `18276` in einer Spalte mit dem Datentyp `date` bei Abfrage als `2020-01-15` gerendert. In diesem UNIX-Format gilt jeder Tag als 86.400 Sekunden.
  - Das OpenCSVSerDe unterstützt `DATE` in keinem anderen Format direkt. Um Zeitstempeldaten in anderen Formaten zu verarbeiten, können Sie die Spalte als `string` definieren und dann Zeitkonvertierungsfunktionen verwenden, um die gewünschten Ergebnisse in Ihrer `SELECT`-Abfrage zurückzugeben. Weitere Informationen finden Sie im Artikel Wenn ich eine Tabelle in Amazon Athena abfrage, ist das TIMESTAMP-Ergebnis leer im AWS-Wissenscenter.
- Um weitere Spalten in den gewünschten Typ in einer Tabelle umzuwandeln, können Sie eine Ansicht über die Tabelle erstellen und `CAST` für die Umwandlung in den gewünschten Typ verwenden.

Example Beispiel: Verwenden des im numerischen UNIX-Format angegebenen TIMESTAMP- und DATE-Typs.

Betrachten Sie die folgenden drei Spalten mit durch Kommas getrennten Daten. Die Werte in jeder Spalte sind in doppelten Anführungszeichen eingeschlossen.

```
"unixvalue creationdate 18276 creationdatetime 1579059880000","18276","1579059880000"
```

Über die folgende Anweisung wird eine Tabelle in Athena von dem angegebenen Amazon-S3-Bucket-Speicherort erstellt.

```
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'
```

Anschließen führen Sie die folgende Abfrage aus:

```
SELECT * FROM testtimestamp1
```

Die Abfrage gibt das folgende Ergebnis zurück, das die Datums- und Uhrzeitdaten zeigt:

```
profile_id                                                          creationdate
 creationdatetime
unixvalue creationdate 18276 creationdatetime 1579146280000        2020-01-15
 2020-01-15 03:44:40.000
```

Example Beispiel: Escape von **\t** oder **\n**

Sehen Sie sich die folgenden Testdaten an:

```
" \\t\\t\\n 123 \\t\\t\\n ",abc
" 456 ",xyz
```

Die folgende Anweisung erstellt eine Tabelle in Athena, mit `"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/'
```

Anschließen führen Sie die folgende Abfrage aus:

```
SELECT * FROM test1;
```

Sie gibt das folgende Ergebnis zurück, wobei ein ordnungsgemäßes Escape für `\t` oder `\n` durchgeführt wird:

```
f1            s2
\t\t\n 123 \t\t\n            abc
456                          xyz
```

SerDe-Name

CSV SerDe

Name der Bibliothek

Um dieses SerDe zu verwenden, geben Sie seinen vollqualifizierten Klassennamen nach `ROW FORMAT SERDE` an. Geben Sie außerdem die Trennzeichen innerhalb von `SERDEPROPERTIES` wie folgt an:

```
...
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
  "separatorChar" = ",",
  "quoteChar"     = "`",
  "escapeChar"    = "\\"
)
```

Ignorieren von Kopfzeilen

Um Header in Ihren Daten beim Definieren einer Tabelle zu ignorieren, können Sie die Tabelleneigenschaft `skip.header.line.count` wie im folgenden Beispiel verwenden.

```
TBLPROPERTIES ("skip.header.line.count"="1")
```

Beispiele finden Sie unter `CREATE TABLE`-Anweisungen in Abfragen von Amazon-VPC-Flow-Protokollen und CloudFront Amazon-Logs abfragen.

Beispiel

In diesem Beispiel wird davon ausgegangen, dass Daten in CSV unter `s3://DOC-EXAMPLE-BUCKET/mycsv/` mit folgendem Inhalt gespeichert ist:

```
"a1","a2","a3","a4"
"1","2","abc","def"
"a","a1","abc3","ab4"
```

Verwenden Sie eine `CREATE TABLE`-Anweisung, um eine Athena-Tabelle basierend auf den Daten zu erstellen. Verweisen Sie nach `ROW FORMAT SERDE` auf die Klasse OpenCSVSerDe und geben Sie das Zeichentrennzeichen, das Anführungszeichen und das Escape-Zeichen in `WITH SERDEPROPERTIES` an, wie im folgenden Beispiel.

```
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/';
```

Fragen Sie alle Werte in der Tabelle ab:

```
SELECT * FROM myopencsvtable;
```

Die Abfrage gibt die folgenden Werte zurück:

```
col1     col2    col3    col4
-----------------------------
a1       a2      a3      a4
1        2       abc     def
a        a1      abc3    ab4
```

ORC SerDe

SerDe-Name

OrcSerDe

Name der Bibliothek

Diese Bibliothek verwendet die OrcSerde.java-Klasse für Daten im ORC-Format. Sie übergibt das Objekt aus ORC an den Reader und den Writer.

Beispiele

> **Note**
>
> Ersetzen Sie *myregion* in `s3://athena-examples-myregion/path/to/data/` durch den Regionsbezeichner, in dem Sie Athena ausführen, z. B. `s3://athena-examples-us-west-1/path/to/data/`.

Im folgenden Beispiel wird eine Tabelle für Flugverspätungsdaten in ORC erstellt. Die Tabelle enthält Partitionen:

```
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");
```

Führen Sie die `MSCK REPAIR TABLE`-Anweisung auf der Tabelle aus, um die Partitionsmetadaten zu aktualisieren:

```
MSCK REPAIR TABLE flight_delays_orc;
```

Verwenden Sie diese Abfrage, um die 10 obersten Routen abzurufen, bei denen es zu einer Verzögerung von mehr als 1 Stunde kam:

```
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 Name

ParquetHiveSerDe wird für Daten verwendet, die im Parquet-Format gespeichert sind.

> **Note**
>
> Um Daten in das Parquet-Format zu konvertieren, können Sie CREATE TABLE AS SELECT (CTAS)-Abfragen verwenden. Weitere Informationen finden Sie unter Erstellen einer Tabelle aus Abfrageergebnissen (CTAS), Beispiele für CTAS-Abfragen und Verwenden von CTAS und INSERT INTO für ETL und Datenanalyse.

Name der Bibliothek

Athena verwendet die folgende Klasse, wenn in Parquet gespeicherte Daten deserialisiert werden müssen: `org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe`

Beispiel: Abfragen einer in Parquet gespeicherten Datei

> **Note**
>
> Ersetzen Sie *myregion* in `s3://athena-examples-myregion/path/to/data/` durch den Regionsbezeichner, in dem Sie Athena ausführen, z. B. `s3://athena-examples-us-west-1/path/to/data/`.

Verwenden Sie die folgende `CREATE TABLE` Anweisung, um eine Athena-Tabelle aus den zugrunde liegenden Daten zu erstellen, die im Parquet-Format in Amazon S3 gespeichert sind:

```
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");
```

Führen Sie die `MSCK REPAIR TABLE`-Anweisung auf der Tabelle aus, um die Partitionsmetadaten zu aktualisieren:

```
MSCK REPAIR TABLE flight_delays_pq;
```

Fragen Sie die 10 Routen ab, bei denen es zu einer Verzögerung von mehr als 1 Stunde kam:

```
SELECT origin, dest, count(*) as delays
FROM flight_delays_pq
WHERE depdelayminutes > 60
GROUP BY origin, dest
ORDER BY 3 DESC
LIMIT 10;
```

> **Note**
>
> Die Flugtabellendaten stammen aus Flügen, die vom US-Verkehrsministerium, Bureau of Transportation Statistics, bereitgestellt werden. Entsättigt vom Original.

Parquet-Statistiken ignorieren

Wenn Sie Parquet-Daten lesen, erhalten Sie möglicherweise Fehlermeldungen wie die folgenden:

```
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.
```

Um dieses Problem zu umgehen, verwenden Sie die - CREATE TABLE oder -ALTER TABLE SET TBLPROPERTIESAnweisung, um die Parquet SerDe-`parquet.ignore.statistics`Eigenschaft auf festzulegen`true`, wie in den folgenden Beispielen.

Beispiel für CREATE TABLE

```
...
ROW FORMAT SERDE
'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
WITH SERDEPROPERTIES (
'parquet.ignore.statistics'='true')
```

```
STORED AS PARQUET
...
```

Beispiel für ALTER TABLE

```
ALTER TABLE ... SET TBLPROPERTIES ('parquet.ignore.statistics'='true')
```

Regex SerDe

Die Regex SerDe verwendet einen regulären Ausdruck (Regex), um Daten zu deserialisieren, indem Regex-Gruppen in Tabellenspalten extrahiert werden.

Wenn eine Zeile in den Daten nicht mit dem Regex übereinstimmt, werden alle Spalten in der Zeile als `NULL` zurückgegeben. Wenn eine Zeile mit dem Regex übereinstimmt, aber weniger Gruppen hat als erwartet, sind die fehlenden Gruppen `NULL`. Wenn eine Zeile in den Daten mit dem Regex übereinstimmt, aber mehr Spalten als Gruppen in dem Regex enthält, werden die zusätzlichen Spalten ignoriert.

Weitere Informationen finden Sie unter Class in der Apache Hive-Dokumentation. RegexSerDe

SerDe Name

RegexSerDe

Name der Bibliothek

RegexSerDe

Beispiele

Im folgenden Beispiel wird eine Tabelle aus CloudFront Protokollen mit dem erstellt RegExSerDe. Ersetzen Sie *myregion* in `s3://athena-examples-`*myregion*`/cloudfront/plaintext/` durch den Regionsbezeichner, in dem Sie Athena ausführen, (z. B. `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/';
```

Ausführen von SQL-Abfragen mit Amazon Athena

Sie können SQL-Abfragen mit Amazon Athena für Datenquellen ausführen, die bei AWS Glue Data Catalog registriert sind, und Datenquellen wie Hive-Metastores und Amazon-DocumentDB-Instances, mit denen Sie über das Athena-Federated-Query-Feature eine Verbindung herstellen. Weitere Hinweise zum Arbeiten mit Datenquellen finden Sie unter Herstellen von Verbindungen mit Datenquellen. Wenn Sie eine DDL-Abfrage (Data Definition Language) ausführen, die das Schema ändert, schreibt Athena die Metadaten in den Metastore, der der Datenquelle zugeordnet ist. Darüber hinaus können einige Abfragen, z. B. `CREATE TABLE AS` und `INSERT INTO`, Datensätze in den Datensatz schreiben und etwa einen CSV-Datensatz zu einem Amazon-S3-Speicherort hinzufügen. Beim Ausführen einer Abfrage werden die Ergebnisse einer Abfrage an einem von Ihnen angegebenen Abfrageergebnisspeicherort von Athena gespeichert. Auf diese Weise können Sie den Abfrageverlauf anzeigen und Abfrageergebnissätze herunterladen und anzeigen.

Dieser Abschnitt enthält Anleitungen zum Ausführen von Athena-Abfragen für gängige Datenquellen und Datentypen mit einer Vielzahl von SQL-Anweisungen. Es werden allgemeine Leitlinien für die Arbeit mit gemeinsamen Strukturen und Operatoren gegeben, zum Beispiel für die Arbeit mit Arrays, Verkettung, Filterung, Abflachung und Sortierung. Andere Beispiele sind Abfragen von Daten in Tabellen mit verschachtelten Strukturen und Zuordnungen, Tabellen, die auf JSON-codierten Datensätzen basieren, und Datensätze, die mit AWS-Services Protokollen und Amazon EMR-Protokollen verknüpft sind. AWS CloudTrail Umfassende Abdeckung der SQL-Standardnutzung geht über den Rahmen dieser Dokumentation hinaus. Weitere Informationen zu SQL finden Sie in den Sprachreferenzen Trino und Presto.

Themen

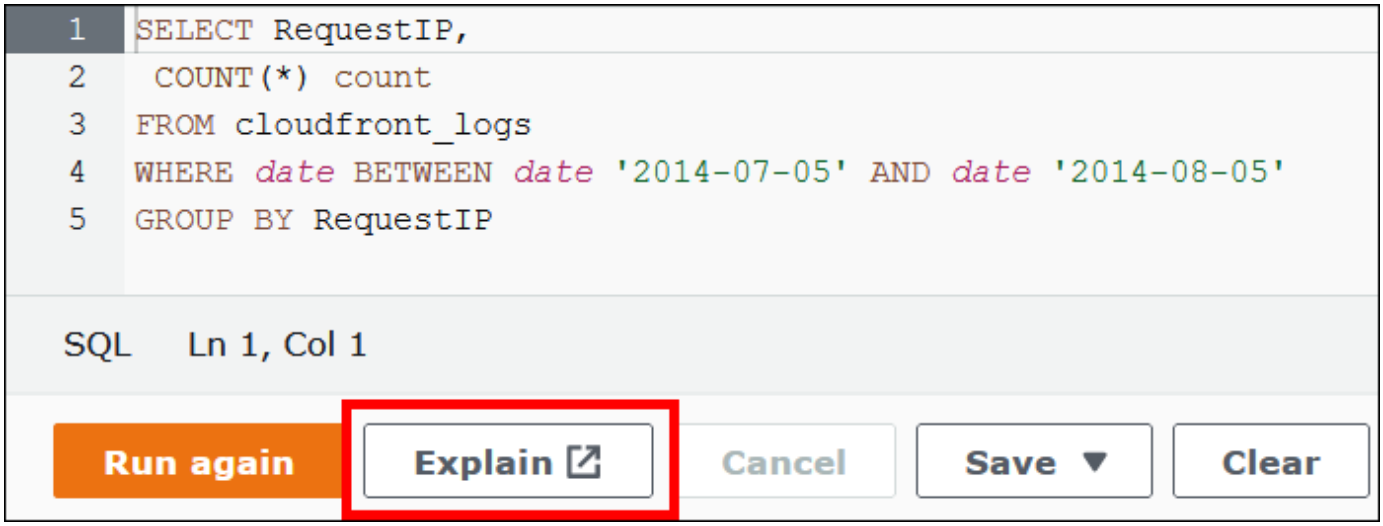
Überlegungen und Einschränkungen finden Sie unter Überlegungen und Einschränkungen für SQL-Abfragen in Amazon Athena.

Anzeigen von Ausführungsplänen für SQL-Abfragen

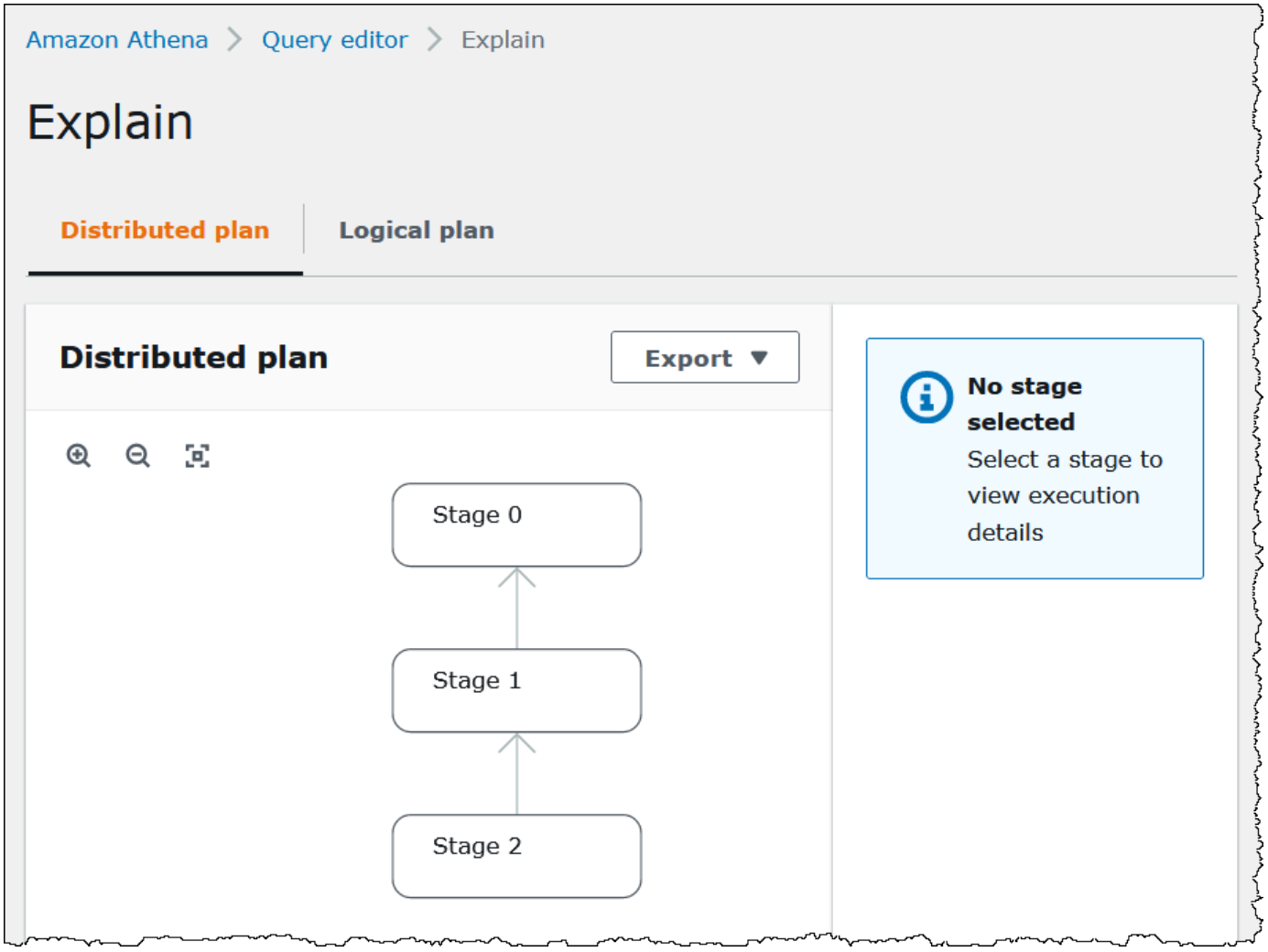
Sie können den Athena-Abfrage-Editor verwenden, um grafisch dargestellt zu bekommen, wie Ihre Abfrage ausgeführt wird. Wenn Sie im Editor eine Abfrage eingeben und die Explain-Option auswählen, verwendet Athena eine EXPLAIN-SQL-Anweisung für Ihre Abfrage, um zwei entsprechende Diagramme zu erstellen: einen verteilten Ausführungsplan und einen logischen Ausführungsplan. Sie können diese Diagramme verwenden, um Ihre Abfragen zu analysieren, Fehler zu beheben und die Effizienz zu verbessern.

So zeigen Sie Ausführungspläne für eine Abfrage an

1. Geben Sie Ihre Abfrage in den Athena-Abfrage-Editor ein und wählen Sie dann Explain (Erklären) aus.



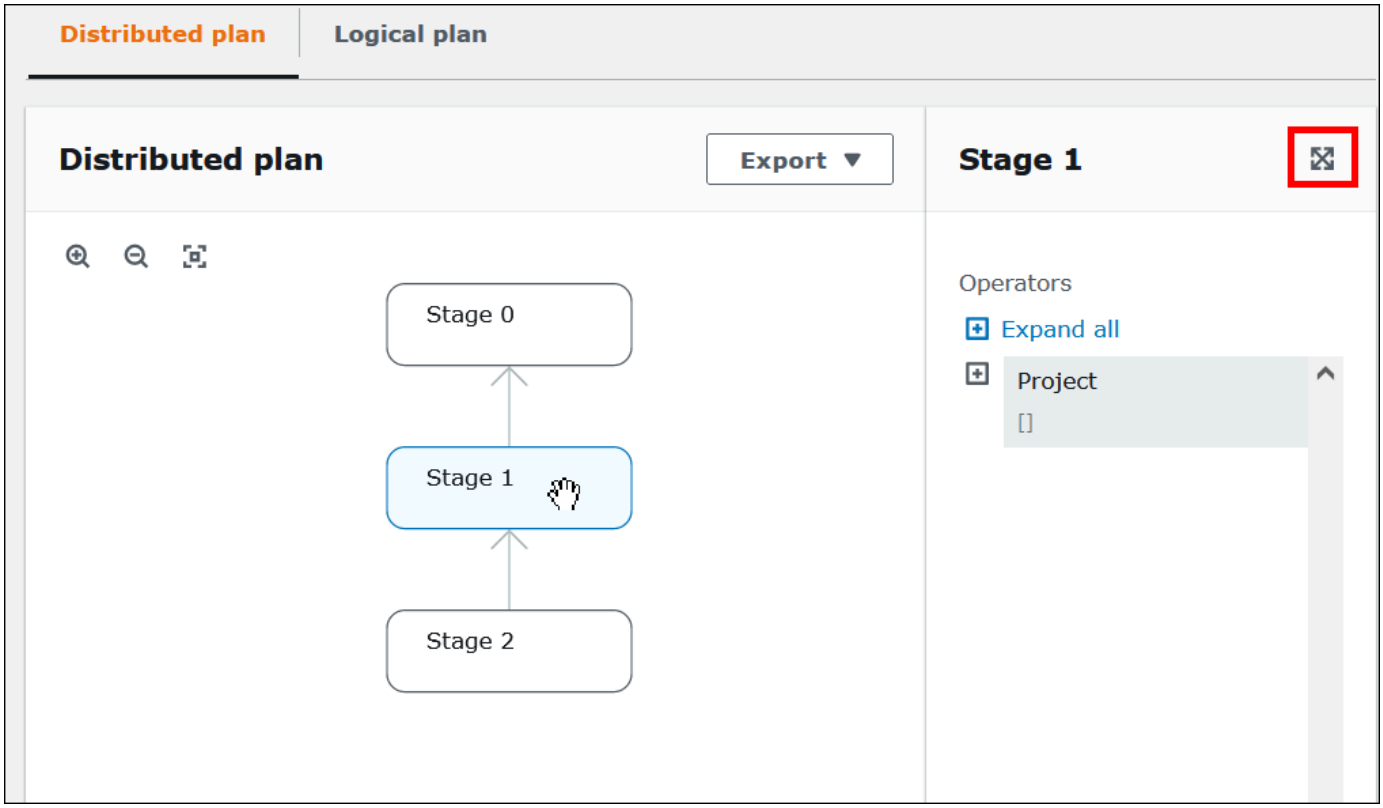
Der Reiter Distributed plan (Verteilter Plan) zeigt Ihnen den Ausführungsplan für Ihre Abfrage in einer verteilten Umgebung an. Ein verteilter Plan hat Verarbeitungsfragmente oder Stufen. Jede Stufe hat eine nullbasierte Indexnummer und wird von einem oder mehreren Knoten verarbeitet. Daten können zwischen Knoten ausgetauscht werden.



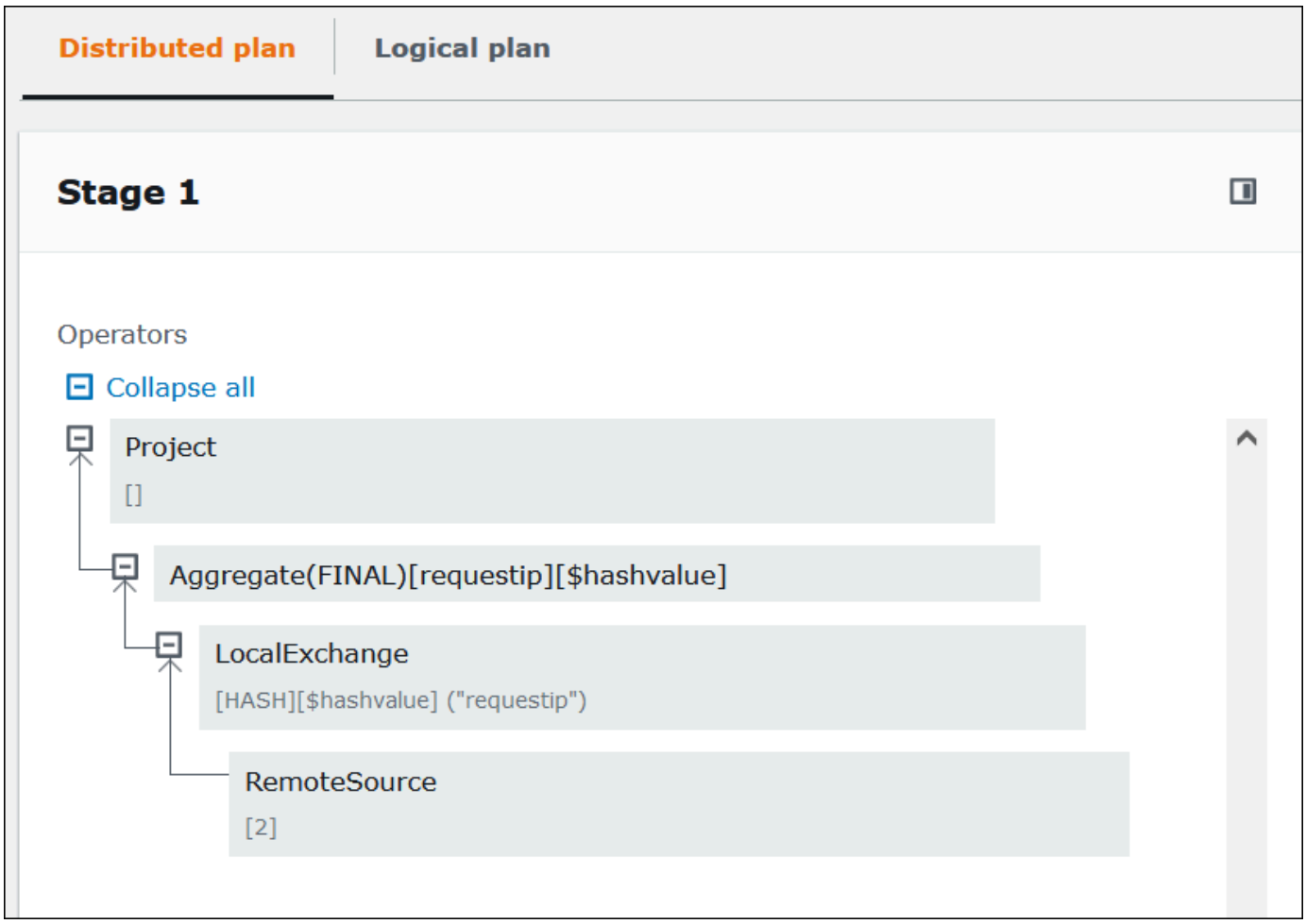
2. Verwenden Sie die folgenden Optionen, um im Diagramm zu navigieren:

   - Zum Vergrößern oder Verkleinern scrollen Sie mit der Maus oder verwenden Sie die Vergrößerungssymbole.
   - Um das Diagramm an den Bildschirm anzupassen, wählen Sie die Option Zoom to fit (Auf passende Größe zoomen) aus.
   - Zum Bewegen des Diagramms ziehen Sie den Mauszeiger.

3. Um Details für eine Stufe anzuzeigen, wählen Sie die Stufe aus.

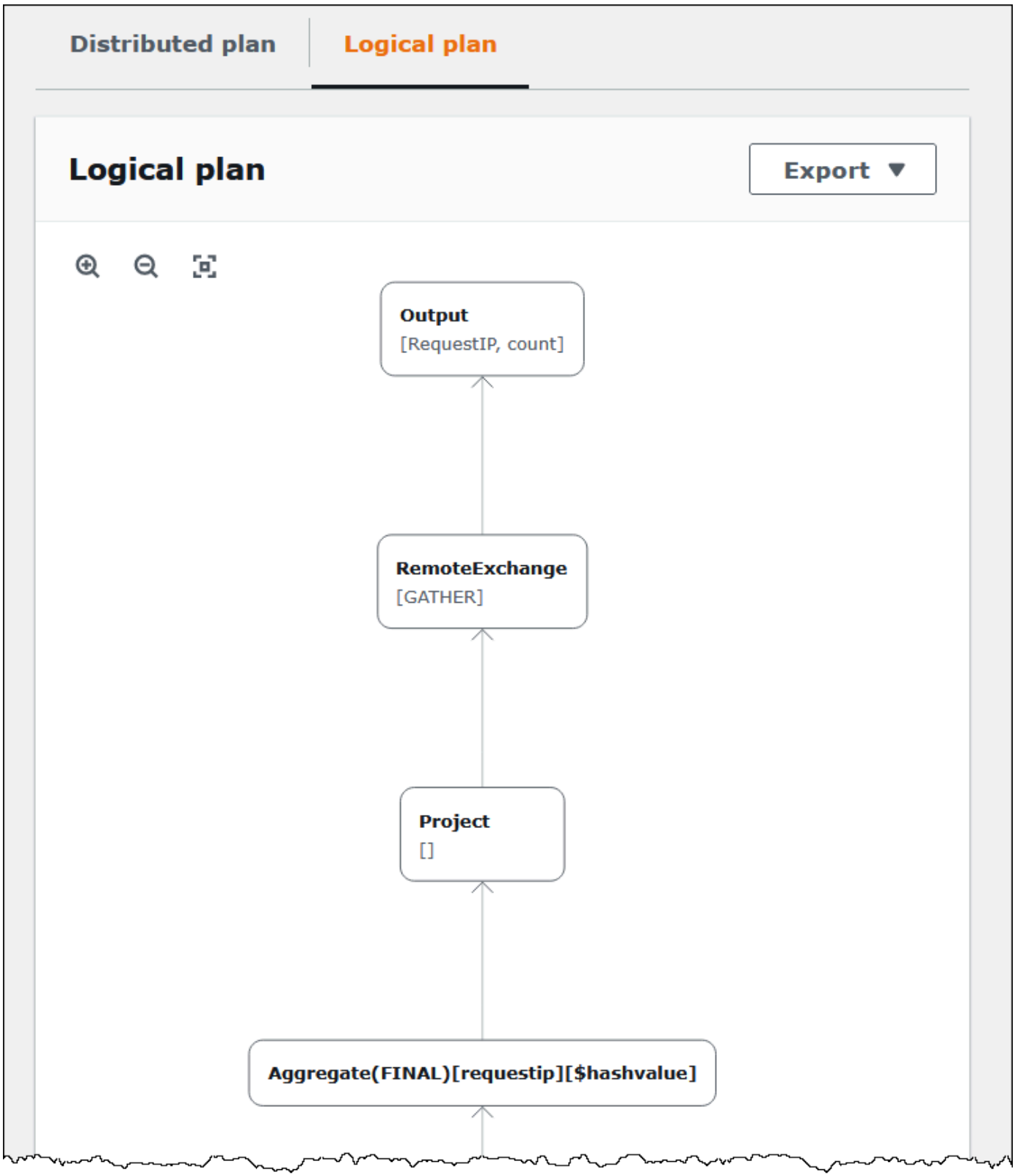


4. Um die Details der Stufe in voller Breite anzuzeigen, wählen Sie das Erweiterungssymbol oben rechts im Detailbereich aus.
5. Um mehr Details anzuzeigen, erweitern Sie ein oder mehrere Elemente im Operator-Baum. Hinweise zu verteilten Planfragmenten finden Sie unter EXPLAIN-Anweisungs-Ausgabetypen.



> **Important**
>
> Derzeit sind einige Partitionsfilter im verschachtelten Operator-Baumdiagramm möglicherweise nicht sichtbar, obwohl Athena sie auf Ihre Abfrage anwendet. Um die Wirkung solcher Filter zu überprüfen, führen Sie EXPLAIN oder EXPLAIN ANALYZE auf Ihre Anfrage aus und sehen Sie sich die Ergebnisse an.

6. Wählen Sie den Reiter Logical plan (Logischer Plan) aus. Das Diagramm zeigt den logischen Plan für die Ausführung Ihrer Abfrage. Informationen über die operativen Bedingungen finden Sie unter Die Ergebnisse der Athena-EXPLAIN-Anweisung verstehen.



7. Um einen Plan als SVG- oder PNG-Bild oder als JSON-Text zu exportieren, wählen Sie Export (Exportieren) aus.

Weitere Informationen finden Sie unter:

Weitere Informationen finden Sie in den folgenden Ressourcen.

Verwenden von EXPLAIN und EXPLAIN ANALYZE in Athena

Die Ergebnisse der Athena-EXPLAIN-Anweisung verstehen

Anzeigen von Statistiken und Ausführungsdetails für abgeschlossene Abfragen

Arbeiten mit Abfrageergebnissen, letzten Abfragen und Ausgabedateien

Amazon Athena speichert automatisch Abfrageergebnisse und Metadateninformationen für jede Abfrage, die an einem Speicherort des Abfrageergebnisses ausgeführt wird, den Sie in Amazon S3 angeben können. Falls erforderlich, können Sie auf die Dateien an diesem Speicherort zugreifen, um mit ihnen zu arbeiten. Sie können Abfrageergebnisdateien auch direkt von der Athena-Konsole herunterladen.

Informationen zum erstmaligen Einrichten eines Amazon S3 Abfrageergebnisses finden Sie unter Angeben eines Speicherorts des Abfrageergebnisses mithilfe der Athena-Konsole.

Ausgabedateien werden automatisch für jede Abfrage gespeichert, die ausgeführt wird. Um auf Abfrageausgabedateien zuzugreifen und diese anzuzeigen, benötigen IAM-Prinzipale (Benutzer und Rollen) die Berechtigung für die Amazon-S3-GetObject-Aktion für den Speicherort des Abfrageergebnisses sowie die Berechtigung für die Athena-GetQueryResults-Aktion. Der Speicherort des Abfrageergebnisses kann verschlüsselt werden. Wenn der Speicherort verschlüsselt ist, müssen Benutzer über die entsprechenden Schlüsselberechtigungen zum Ver- und Entschlüsseln des Speicherorts des Abfrageergebnisses verfügen.

> **Important**
>
> IAM-Prinzipale mit Berechtigung für die `GetObject`-Amazon-S3-Aktion für den Speicherort des Abfrageergebnisses können Abfrageergebnisse auch dann von Amazon S3 abrufen, wenn die Berechtigung für die Athena `GetQueryResults`-Aktion verweigert wird.

Angeben eines Speicherorts des Abfrageergebnisses

Der Speicherort des Abfrageergebnisses, den Athena verwendet, wird durch eine Kombination aus Arbeitsgruppeneinstellungen und clientseitigen Einstellungen bestimmt. Die clientseitigen Einstellungen basieren darauf, wie Sie die Abfrage ausführen.

- Wenn Sie die Abfrage über die Athena-Konsole ausführen, bestimmt der Speicherort des Abfrageergebnisses, der unter Settings (Einstellungen) in der Navigationsleiste eingegeben wurde, die clientseitige Einstellung.
- Wenn Sie die Abfrage mit der Athena-API ausführen, bestimmt der `OutputLocation`-Parameter der StartQueryExecution-Aktion die clientseitige Einstellung.
- Wenn Sie die ODBC- oder JDBC-Treiber zum Ausführen von Abfragen verwenden, bestimmt die in der Verbindungs-URL angegebene `S3OutputLocation`-Eigenschaft die clientseitige Einstellung.

> **Important**
>
> Wenn Sie eine Abfrage mit der API oder mit dem ODBC- oder JDBC-Treiber ausführen, gilt die Konsoleneinstellung nicht.

Jede Arbeitsgruppenkonfiguration verfügt über eine Option Override client-side settings (Clientseitige Einstellungen überschreiben), die aktiviert werden kann. Wenn diese Option aktiviert ist, haben die Arbeitsgruppeneinstellungen Vorrang vor den anwendbaren clientseitigen Einstellungen, wenn ein IAM-Prinzipal, der dieser Arbeitsgruppe zugeordnet ist, die Abfrage ausführt.

Angeben eines Speicherorts des Abfrageergebnisses mithilfe der Athena-Konsole

Bevor Sie eine Abfrage ausführen können, muss in Amazon S3 ein Speicherort für das Abfrageergebnis angegeben werden, oder Sie müssen eine Arbeitsgruppe verwenden, für die ein Bucket angegeben wurde und deren Konfiguration die Client-Einstellungen überschreibt.

So geben Sie einen clientseitigen Speicherort für das Abfrageergebnis mit der Athena-Konsole an

1. Wechseln Sie zu der Arbeitsgruppe, für die Sie einen Speicherort für Abfrageergebnisse angeben möchten. Der Name der Standardarbeitsgruppe ist primär.
2. Wählen Sie in der Navigationsleiste Settings (Einstellungen) aus.
3. Wählen Sie auf der Navigationsleiste Manage (Verwalten).
4. Führen Sie für Manage settings (Einstellungen verwalten) einen der folgenden Schritte aus:
   - Geben Sie im Feld Speicherort der Abfrageergebnisse den Pfad zu dem Bucket ein, den Sie in Amazon S3 für Ihre Abfrageergebnisse erstellt haben. Stellen Sie dem Pfad einen Präfix mit `s3://` aus.

- Wählen Sie das Symbol S3 durchsuchen, wählen Sie den Amazon-S3-Bucket aus, den Sie in Ihrer aktuellen Region erstellt haben und wählen Sie dann Auswählen.

> **Note**
>
> Wenn Sie eine Arbeitsgruppe verwenden, in der ein Speicherort für das Abfrageergebnis für alle Benutzer der Arbeitsgruppe angegeben wird, steht die Option zum Ändern des Speicherorts für das Abfrageergebnis nicht zur Verfügung.

5. (Optional) Wählen Sie View lifecycle configuration (Lebenszykluskonfiguration anzeigen) aus, um die Amazon S3 lifecycle rules (Amazon-S3-Lebenszyklusregeln) für Ihren Abfrageergebnis-Bucket anzuzeigen und zu konfigurieren. Die Amazon-S3-Lebenszyklusregeln, die Sie erstellen, können entweder Ablaufregeln oder Übergangsregeln sein. Ablaufregeln löschen Abfrageergebnisse automatisch nach einer bestimmten Zeit. Übergangsregeln verschieben sie auf eine andere Amazon-S3-Speicherebene. Weitere Informationen finden Sie unter Festlegen der Lebenszykluskonfiguration für einen Bucket im Benutzerhandbuch zu Amazon Simple Storage Service.
6. (Optional) Geben Sie für Expected bucket owner (Erwarteter Bucket-Eigentümer) die ID des AWS-Konto ein, von dem Sie erwarten, dass es der Eigentümer des Ausgabespeicherort-Buckets ist. Dies ist eine zusätzliche Sicherheitsmaßnahme. Wenn die Konto-ID des Bucket-Eigentümers nicht mit der ID übereinstimmt, die Sie hier angeben, schlagen Versuche, in den Bucket auszugeben, fehl. Ausführliche Informationen finden Sie unter Überprüfen der Bucket-Eigentümerschaft mit Bucket-Eigentümer-Bedingung im Amazon-S3-Benutzerhandbuch.

> **Note**
>
> Die erwartete Bucket-Eigentümereinstellung gilt nur für den Amazon-S3-Ausgabespeicherort, den Sie für Athena-Abfrageergebnisse angeben. Sie gilt nicht für andere Amazon-S3-Speicherorte wie Datenquellenspeicherorte in externen Amazon-S3-Buckets, `CTAS`- und `INSERT INTO`-Speicherorte der Zieltabelle, `UNLOAD`-Speicherorte der Anweisungsangaben, Vorgänge zum Verschütten von Buckets für Verbundabfragen oder `SELECT`-Abfragen, die für eine Tabelle in einem anderen Konto ausgeführt werden.

7. (Optional) Wählen Sie Encrypt query results (Abfrageergebnisse verschlüsseln) aus, wenn Sie die in Amazon S3 gespeicherten Abfrageergebnisse verschlüsseln möchten. Weitere Informationen zur Verschlüsselung in Athena finden Sie unter Verschlüsselung im Ruhezustand.

8. (Optional) Wählen Sie Assign bucket owner full control over query results (Bucket-Eigetümer die volle Kontrolle über Abfrage zuweisen) aus, um dem Bucket-Eigentümer die volle Kontrolle über Abfrageergebnisse zu gewähren, wenn für den Abfrageergebnis-Bucket ACLs aktiviert sind. Wenn der Standort Ihres Abfrageergebnisses beispielsweise einem anderen Konto gehört, können Sie dem anderen Konto das Eigentum und die volle Kontrolle über Ihre Abfrageergebnisse gewähren. Weitere Informationen finden Sie unter Steuern des Eigentums an Objekten und Deaktivieren von ACLs für Ihren Bucket im Amazon-S3-Benutzerhandbuch.
9. Wählen Sie Save (Speichern).

Zuvor erstellte Standardspeicherorte

Wenn Sie früher eine Abfrage in Athena ausgeführt haben, ohne einen Wert für den Speicherort für das Abfrageergebnis anzugeben und wenn die Speicherort-Einstellung für das Abfrageergebnis nicht von einer Arbeitsgruppe überschrieben wurde, wurde früher von Athena ein Standardspeicherort angelegt. Der Standardspeicherort war `aws-athena-query-results-`*`MyAcctID`*`-`*`MyRegion`*, wobei *`MyAcctID`* die Amazon-Web-Services-Konto-ID des IAM-Prinzipals war, der die Abfrage ausgeführt hat, und *`MyRegion`*die Region war, in der die Abfrage ausgeführt wurde (z. B. `us-west-1`).

Bevor Sie nun eine Athena-Abfrage in einer Region ausführen können, in der Ihr Konto bisher nicht Athena verwendete, müssen Sie einen Speicherort für das Abfrageergebnis angeben oder eine Arbeitsgruppe verwenden, die die Speicherort-Einstellung für das Abfrageergebnis überschreibt. Auch wenn Athena keinen Standardspeicherort für das Abfrageergebnis mehr für Sie erstellt, bleiben zuvor angelegte Standard-`aws-athena-query-results-`*`MyAcctID`*`-`*`MyRegion`*-Speicherorte gültig und Sie können sie weiterhin verwenden.

Angeben eines Speicherorts des Abfrageergebnisses mithilfe einer Arbeitsgruppe

Sie geben den Speicherort des Abfrageergebnisses in einer Arbeitsgruppenkonfiguration mithilfe der AWS Management Console, der AWS CLI oder der Athena-API an.

Wenn Sie AWS CLI verwenden, geben Sie den Speicherort des Abfrageergebnisses mithilfe des `OutputLocation`-Parameters der Option `--configuration` an, wenn Sie den Befehl `aws athena create-work-group` oder `aws athena update-work-group` ausführen.

So geben Sie den Speicherort des Abfrageergebnisses für eine Arbeitsgruppe mithilfe der Athena-Konsole an

1. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.

2. Wählen Sie im Navigationsbereich die Option Workgroups (Arbeitsgruppen) aus.
3. Wählen Sie in der Liste der Arbeitsgruppen den Link der Arbeitsgruppe aus, die Sie bearbeiten möchten.
4. Wählen Sie Edit (Bearbeiten).
5. Führen Sie für Speicherort und Verschlüsselung des Abfrageergebnisses einen der folgenden Schritte aus:

   - Geben Sie im Feld Speicherort des Abfrageergebnisses den Pfad zu einem Bucket in Amazon S3 für Ihre Abfrageergebnisse ein. Stellen Sie dem Pfad einen Präfix mit `s3://` aus.
   - Wählen Sie das Symbol S3 durchsuchen, wählen Sie den Amazon-S3-Bucket aus, den Sie in Ihrer aktuellen Region verwenden möchten und wählen Sie dann Auswählen.

6. (Optional) Geben Sie für Expected bucket owner (Erwarteter Bucket-Eigentümer) die ID des AWS-Konto ein, von dem Sie erwarten, dass es der Eigentümer des Ausgabespeicherort-Buckets ist. Dies ist eine zusätzliche Sicherheitsmaßnahme. Wenn die Konto-ID des Bucket-Eigentümers nicht mit der ID übereinstimmt, die Sie hier angeben, schlagen Versuche, in den Bucket auszugeben, fehl. Ausführliche Informationen finden Sie unter Überprüfen der Bucket-Eigentümerschaft mit Bucket-Eigentümer-Bedingung im Amazon-S3-Benutzerhandbuch.

> **Note**
>
> Die erwartete Bucket-Eigentümereinstellung gilt nur für den Amazon-S3-Ausgabespeicherort, den Sie für Athena-Abfrageergebnisse angeben. Sie gilt nicht für andere Amazon-S3-Speicherorte wie Datenquellenspeicherorte in externen Amazon-S3-Buckets, `CTAS`- und `INSERT INTO`-Speicherorte der Zieltabelle, `UNLOAD`-Speicherorte der Anweisungsangaben, Vorgänge zum Verschütten von Buckets für Verbundabfragen oder `SELECT`-Abfragen, die für eine Tabelle in einem anderen Konto ausgeführt werden.

7. (Optional) Wählen Sie Encrypt query results (Abfrageergebnisse verschlüsseln) aus, wenn Sie die in Amazon S3 gespeicherten Abfrageergebnisse verschlüsseln möchten. Weitere Informationen zur Verschlüsselung in Athena finden Sie unter Verschlüsselung im Ruhezustand.
8. (Optional) Wählen Sie Assign bucket owner full control over query results (Bucket-Eigetümer die volle Kontrolle über Abfrage zuweisen) aus, um dem Bucket-Eigentümer die volle Kontrolle über Abfrageergebnisse zu gewähren, wenn für den Abfrageergebnis-Bucket ACLs aktiviert sind. Wenn der Standort Ihres Abfrageergebnisses beispielsweise einem anderen Konto gehört, können Sie dem anderen Konto das Eigentum und die volle Kontrolle über Ihre Abfrageergebnisse gewähren.

   Wenn die Einstellung zur S3-Objekteigentümerschaft des Bucket Bucket owner preferred (Bucket-Eigentümer bevorzugt) lautet, besitzt der Bucket-Eigentümer auch alle Abfrageergebnisobjekte, die aus dieser Arbeitsgruppe geschrieben wurden. Wenn beispielsweise die Arbeitsgruppe eines externen Kontos diese Option aktiviert und den Speicherort des Abfrageergebnisses auf den Amazon-S3-Bucket Ihres Kontos festlegt, der eine Einstellung zur S3-Objekteigentümerschaft von Bucket owner preferred (Bucket-Eigentümer bevorzugt) hat, sind Sie Eigetümer der Abfrageergebnisse der externen Arbeitsgruppe und haben die volle Kontrolle über sie.

   Wenn Sie diese Option auswählen, wenn die Einstellung zur S3-Objekteigentümerschaft des Bucket Bucket owner enforced (Bucket-Eigentümer durchgesetzt) ist, hat dies keine Auswirkungen. Weitere Informationen finden Sie unter Steuern des Eigentums an Objekten und Deaktivieren von ACLs für Ihren Bucket im Amazon-S3-Benutzerhandbuch.
9. Wenn Sie möchten, dass alle Benutzer der Arbeitsgruppe den von Ihnen angegebenen Speicherort für Abfrageergebnisse verwenden, scrollen Sie nach unten zum Abschnitt Settings (Einstellungen) und wählen Sie Override client-side settings (Clientseitige Einstellungen überschreiben).

10. Wählen Sie Save Changes (Änderungen speichern).

Herunterladen von Abfrageergebnisdateien mithilfe der Athena-Konsole

Sie können die CSV-Datei mit den Abfrageergebnissen direkt nach dem Ausführen einer Abfrage aus dem Abfragebereich herunterladen. Sie können auch Abfrageergebnisse aus den kürzlichen Abfragen von der Registerkarte Kürzliche Abfragen herunterladen.

> **Note**
>
> Athena-Abfrageergebnisdateien sind Dateien, die Informationen enthalten, die von den einzelnen Benutzern konfiguriert werden können. Einige Programme, die diese Daten lesen und analysieren, können möglicherweise einige der Daten als Befehle interpretieren (CSV-Injektion). Wenn Sie CSV-Daten von Abfrageergebnissen in eine Kalkulationstabelle importieren, warnt Sie dieses Programm möglicherweise vor Sicherheitsrisiken. Um Ihr System sicher zu halten, sollten Sie immer Links oder Makros aus heruntergeladenen Abfrageergebnissen deaktivieren.

So führen Sie eine Abfrage aus und laden die Abfrageergebnisse herunter

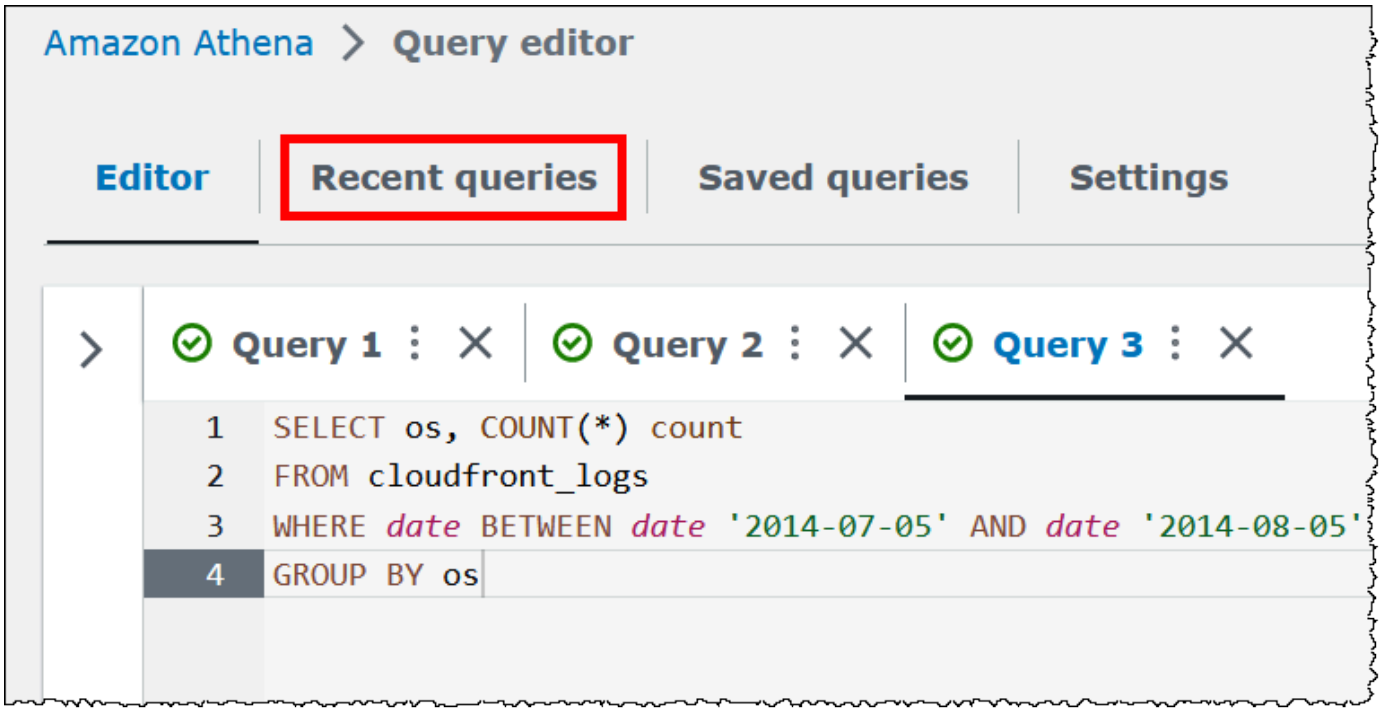
1. Geben Sie Ihre Abfrage in den Abfrage-Editor ein und wählen Sie dann Run (Ausführen) aus.

   Wenn die Ausführung der Abfrage abgeschlossen ist, zeigt der Bereich Results (Ergebnisse) die Abfrageergebnisse an.

2. Um eine CSV-Datei mit den Abfrageergebnissen herunterzuladen, wählen Sie Herunterladen von Ergebnissen über dem Bereich „Abfrageergebnisse“. Je nach Browser- und Browserkonfiguration müssen Sie möglicherweise den Download bestätigen.

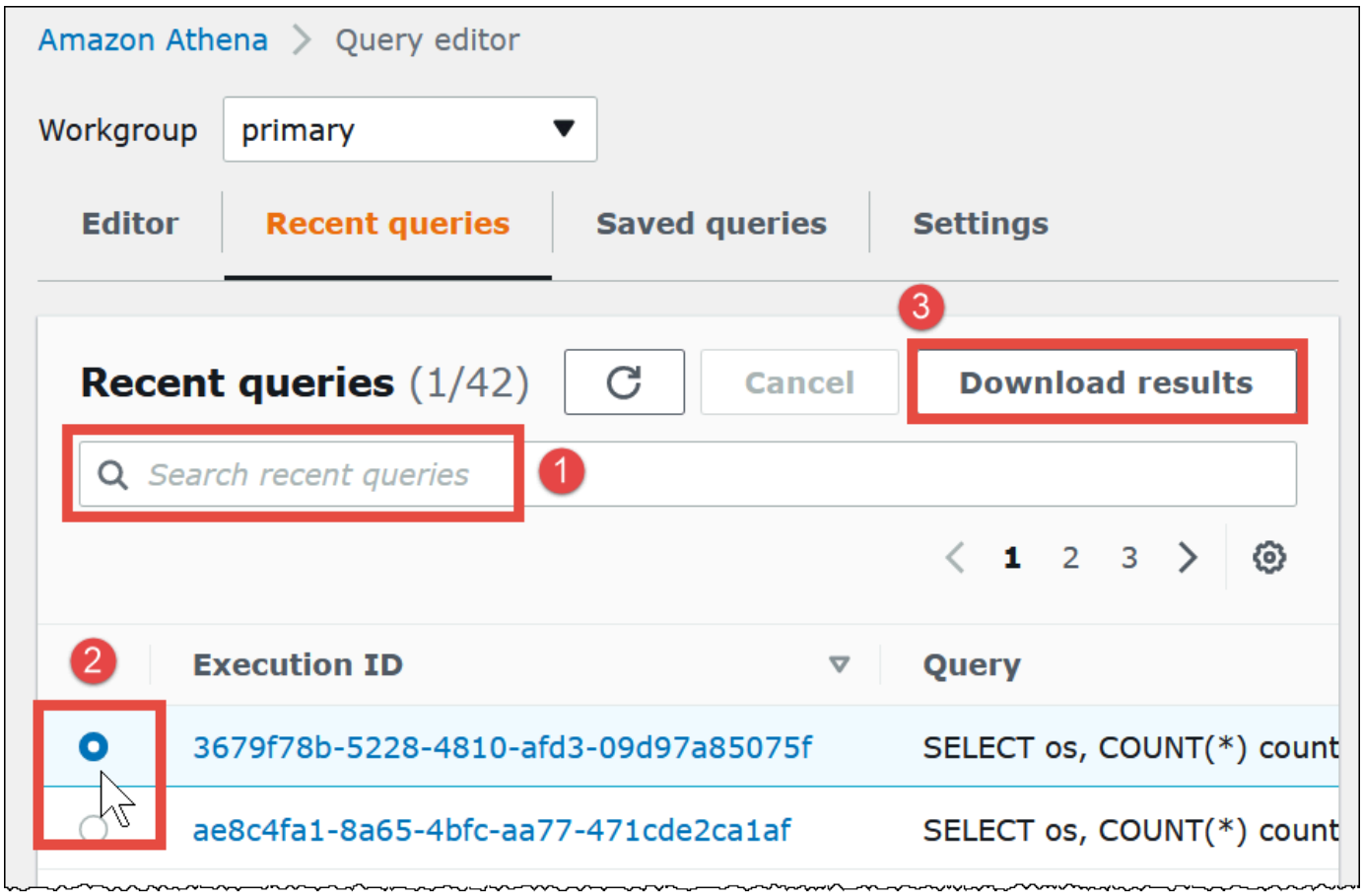
So laden Sie eine Abfrageergebnisdatei für eine frühere Abfrage herunter

1. Wählen Sie Recent queries (Kürzliche Abfragen).



2. Verwenden Sie das Suchfeld, um die Abfrage zu finden, wählen Sie die Abfrage aus und dann Herunterladen von Ergebnissen.

> **Note**
>
> Sie können die Option Download results (Ergebnisse herunterladen) nicht verwenden, um Abfrageergebnisse abzurufen, die manuell gelöscht wurden, oder um Abfrageergebnisse abzurufen, die gelöscht oder durch Amazon-S3-Lebenszyklusregeln an einen anderen Speicherort verschoben wurden.

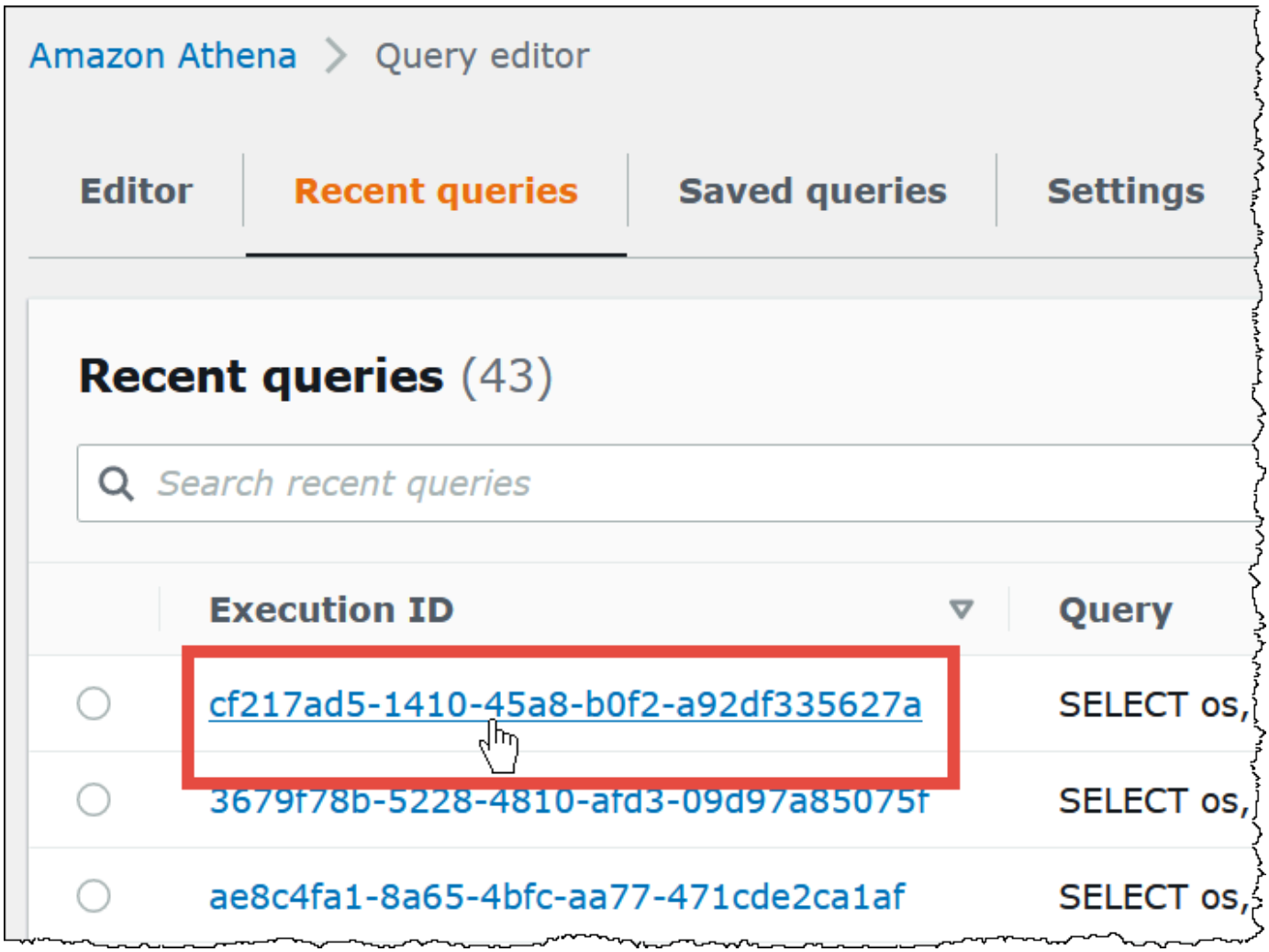


Anzeige kürzlicher Abfragen

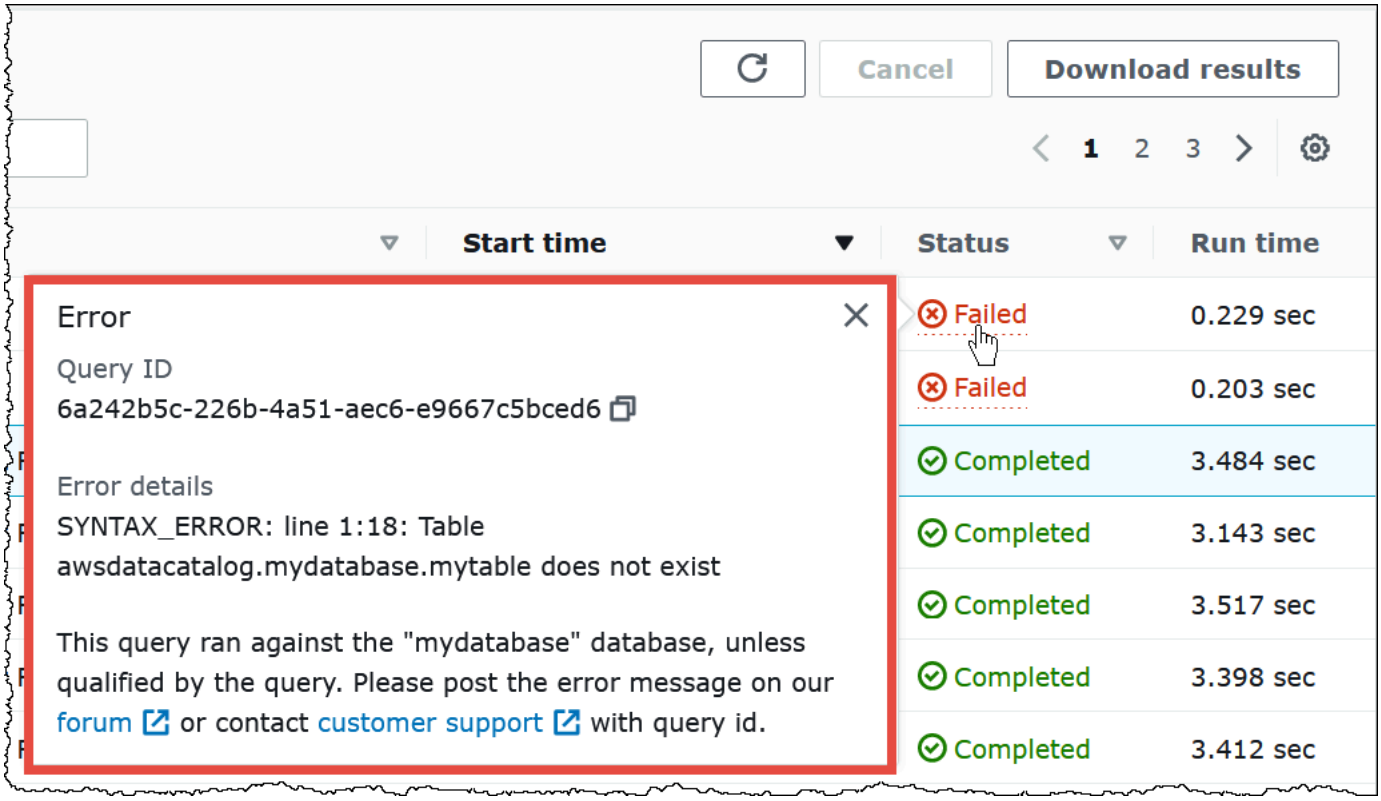
Sie können die Athena-Konsole verwenden, um zu sehen, welche Abfragen erfolgreich waren oder fehlgeschlagen sind und Fehlerdetails für die fehlgeschlagenen Abfragen anzeigen. Athena hält den Abfrageverlauf 45 Tage lang vor.

Anzeigen von aktuellen Abfragen in der Athena-Konsole

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wählen Sie Recent queries (Kürzliche Abfragen). Auf der Registerkarte Kürzliche Abfragen werden Informationen zu jeder ausgeführten Abfrage angezeigt.
3. Um eine Abfrageanweisung im Abfrage-Editor zu öffnen, wählen Sie die Ausführungs-ID der Abfrage aus.



4. Um die Details zu einer fehlgeschlagenen Abfrage anzuzeigen, wählen Sie den Link Fehlgeschlagen für die Abfrage.

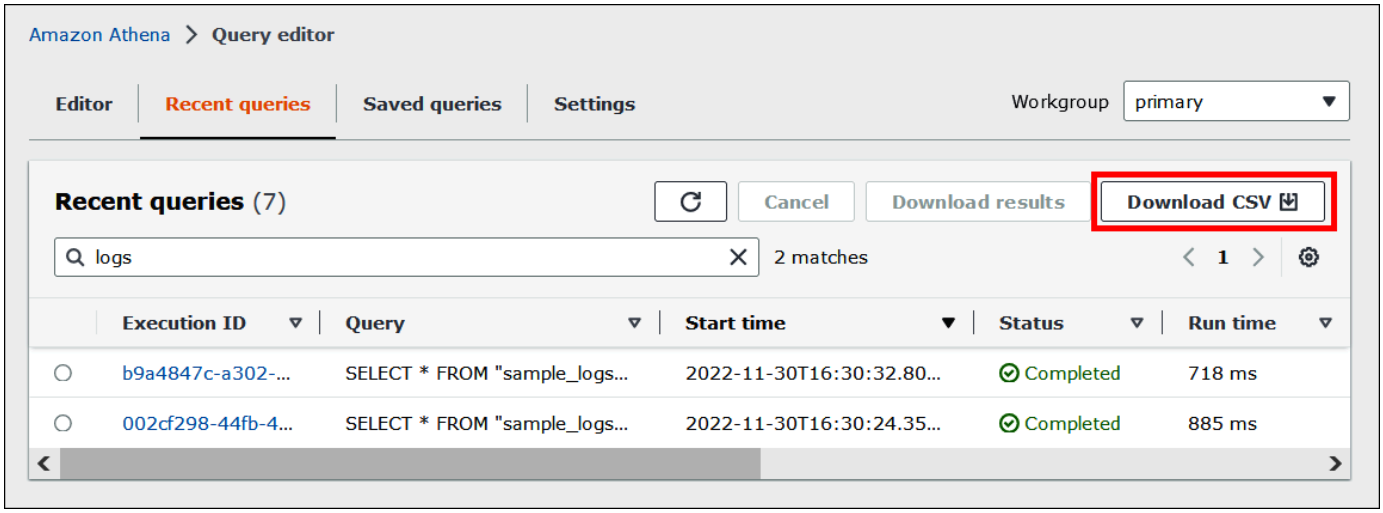


Herunterladen mehrerer aktueller Abfragen in eine CSV-Datei

Sie können die Registerkarte Recent queries (Neueste Abfragen) der Athena-Konsole verwenden, um eine oder mehrere aktuelle Abfragen in eine CSV-Datei zu exportieren, um sie im Tabellenformat anzuzeigen. Die heruntergeladene Datei enthält nicht die Abfrageergebnisse, sondern die SQL-Abfragezeichenfolge selbst und andere Informationen zur Abfrage. Zu den exportierten Feldern gehören die Ausführungs-ID, der Inhalt der Abfragezeichenfolge, die Startzeit der Abfrage, der Status, die Laufzeit, die Menge der gescannten Dateien, die verwendete Version der Abfrage-Engine und die Verschlüsselungsmethode. Sie können maximal 500 kürzlich durchgeführte Abfragen oder maximal 500 gefilterte Abfragen anhand von Kriterien exportieren, die Sie in das Suchfeld eingeben.

So exportieren Sie eine oder mehrere aktuelle Abfragen in eine CSV-Datei

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wählen Sie Recent queries (Kürzliche Abfragen).
3. (Optional) Verwenden Sie das Suchfeld, um nach den neuesten Abfragen zu filtern, die Sie herunterladen möchten.
4. Wählen Sie CSV herunterladen aus.



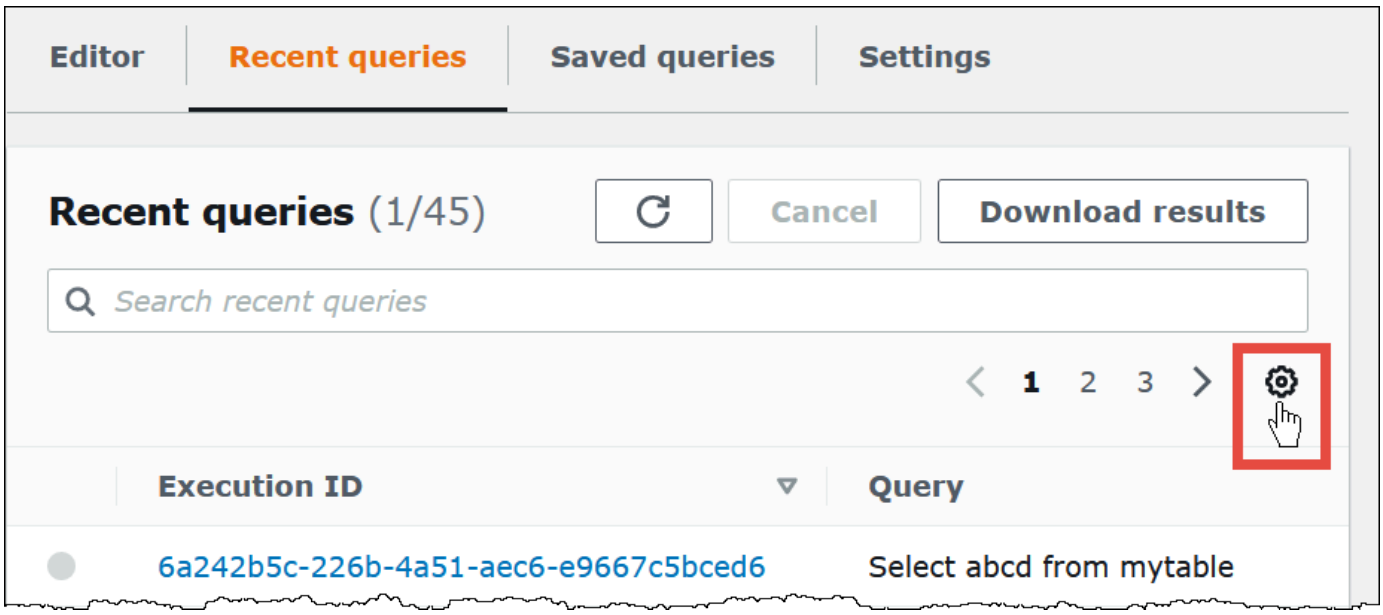
5. Wählen Sie bei der Aufforderung zum Speichern der Datei Save (Speichern). Der Standarddateiname ist `Recent Queries`, gefolgt von einem Zeitstempel (z. B. `Recent Queries 2022-12-05T16 04 27.352-08 00.csv`)

Konfigurieren der Anzeigeoptionen für aktuelle Abfragen

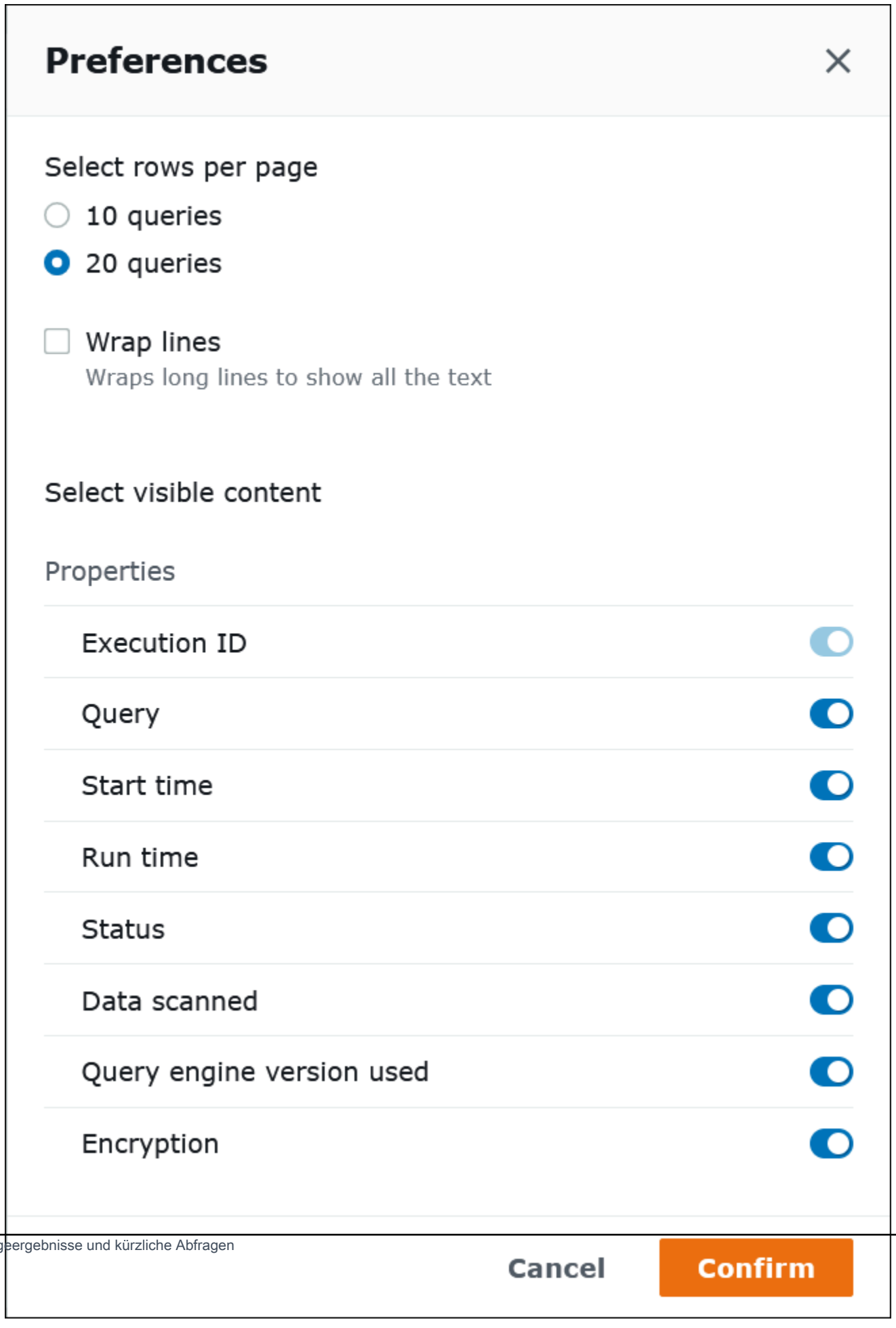
Sie können Optionen für die Registerkarte Recent queries (Aktuelle Abfragen) konfigurieren, z. B. anzuzeigende Spalten und Textumbruch.

So konfigurieren Sie Optionen für die Registerkarte Recent queries (Aktuelle Abfragen)

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wählen Sie Recent queries (Kürzliche Abfragen).
3. Wählen Sie die Optionsschaltfläche (Zahnradsymbol).



4. In den Präferenzen wählen Sie die Anzahl der Zeilen pro Seite, das Zeilenumbruchverhalten und die anzuzeigenden Spalten aus.


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. Wählen Sie Confirm (Bestätigen).

Abfrageverlauf länger als 45 Tage beibehalten

Wenn Sie den Abfrageverlauf länger als 45 Tage beibehalten möchten, können Sie den Abfrageverlauf abrufen und in einem Datenspeicher wie z. B. Amazon S3 speichern. Um diesen Prozess zu automatisieren, können Sie Athena- und Amazon-S3-API-Aktionen und -CLI-Befehle verwenden. Im folgenden Verfahren werden diese Schritte zusammengefasst.

So rufen Sie den Abfrageverlauf programmgesteuert ab und speichern ihn

1. Verwenden Sie die Athena-API-Aktion ListQueryExecutions oder den CLI-Befehl list-query-executions, um die Abfrage-IDs abzurufen.
2. Verwenden Sie die Athena-API-Aktion GetQueryExecution oder den CLI-Befehl get-query-execution, um Informationen zu jeder Abfrage basierend auf ihrer ID abzurufen.
3. Verwenden Sie die Amazon-S3-API-Aktion PutObject oder den CLI-Befehl put-object, um die Informationen in Amazon S3 zu speichern.

Suchen von Abfrageausgabedateien in Amazon S3

Abfrageausgabedateien werden in Unterordnern in Amazon S3 im folgenden Pfadmuster gespeichert, es sei denn, die Abfrage erfolgt in einer Arbeitsgruppe, deren Konfiguration clientseitige Einstellungen außer Kraft setzt. Wenn die Arbeitsgruppenkonfiguration clientseitige Einstellungen außer Kraft setzt, verwendet die Abfrage den von der Arbeitsgruppe angegebenen Ergebnispfad.

```
QueryResultsLocationInS3/[QueryName|Unsaved/yyyy/mm/dd/]
```

- *QueryResultsLocationInS3* ist der Speicherort des Abfrageergebnisses, der entweder durch Arbeitsgruppeneinstellungen oder clientseitige Einstellungen angegeben wird. Weitere Informationen finden Sie unter the section called “Angeben eines Speicherorts des Abfrageergebnisses” an späterer Stelle in diesem Dokument.
- Die folgenden Unterordner werden nur für Abfragen erstellt, die von der Konsole ausgeführt werden, deren Ergebnispfad nicht durch die Arbeitsgruppenkonfiguration außer Kraft gesetzt wurde. Abfragen, die über die AWS CLI oder mithilfe der Athena-API ausgeführt werden, werden direkt in *QueryResultsLocationInS3* gespeichert.
  - *QueryName* ist der Name der Abfrage, deren Ergebnisse gespeichert wurden. Wenn die Abfrage ausgeführt, aber nicht gespeichert wurde, wird `Unsaved` verwendet.

- *yyyy/mm/dd (jjjj/mm/tt)* ist das Datum, an dem die Abfrage ausgeführt wurde.

Dateien, die einer `CREATE TABLE AS SELECT`-Abfrage zugeordnet sind, werden in einem `tables`-Unterordner des obigen Musters gespeichert.

Identifizieren von Abfrageausgabedateien

Dateien werden basierend auf dem Namen der Abfrage, der Abfrage-ID und dem Datum, an dem die Abfrage ausgeführt wurde, am Speicherort des Abfrageergebnisses in Amazon S3 gespeichert. Die Dateien für jede Abfrage werden mit der *QueryID* benannt. Dabei handelt es sich um eine eindeutige Kennung, die Athena jeder Abfrage zuweist, wenn sie ausgeführt wird.

Die folgenden Dateitypen werden gespeichert:

Dateityp	Dateibenennungsmuster	Beschreibung
Abfrageergebnisdateien	*QueryID*`.csv` *QueryID*`.txt`	DML-Abfrageergebnisdateien werden im CSV-Format (durch Komma getrennte Werte) gespeichert. DDL-Abfrageergebnisse werden als reine Textdateien gespeichert. Sie können Ergebnisdateien über die Konsole im Bereich Results (Ergebnisse) herunterl aden, wenn Sie die Konsole verwenden oder über den Verlauf der Abfrage. Weitere Informationen finden Sie unter Herunterladen von Abfrageer gebnisdateien mithilfe der Athena-Konsole.
Abfragemetadatendateien	*QueryID*`.csv.metadata`	DML- und DDL-Abfra gemetadateien werden

Dateityp	Dateibenennungsmuster	Beschreibung
	QueryID.txt.metadata	im Binärformat gespeicherte und sind nicht lesbar. Die Dateierweiterung entspricht der zugehörigen Abfrageergebnisdatei. Athena verwendet die Metadaten beim Lesen von Abfrageergebnissen mithilfe der `GetQueryResults` -Aktion Obwohl diese Dateien gelöscht werden können, empfehlen wir dies nicht, da wichtige Informationen über die Abfrage verloren gehen.
Daten-Manifest-Dateien	*QueryID*-manifest.csv	Daten-Manifest-Dateien werden generiert, um Dateien nachzuverfolgen, die Athena in Amazon-S3-Datenquellenspeicherorten erstellt, wenn eine INSERT INTO-Abfrage ausgeführt wird. Wenn eine Abfrage fehlschlägt, verfolgt das Manifest auch Dateien, die die Abfrage schreiben wollte. Das Manifest ist nützlich für die Identifizierung verwaister Dateien, die aus einer fehlgeschlagenen Abfrage resultieren.

Verwenden der AWS CLI zum Identifizieren des Speicherorts und der Dateien der Abfrageausgabe

Um das AWS CLI zum Identifizieren des Abfrageausgabespeicherorts und der Ergebnisdateien zu verwenden, führen Sie den `aws athena get-query-execution`-Befehl wie im folgenden Beispiel aus. Ersetzen Sie *abc1234d-5efg-67hi-jklm-89n0op12qr34* durch die Abfrage-ID.

```
aws athena get-query-execution --query-execution-id abc1234d-5efg-67hi-
jklm-89n0op12qr34
```

Daraufhin erhalten Sie ein Ergebnis, das dem hier dargestellten entspricht. Beschreibungen der einzelnen Ausgabeparameter finden Sie unter get-query-execution in der AWS CLI-Befehlsreferenz.

```
{
    "QueryExecution": {
        "Status": {
            "SubmissionDateTime": 1565649050.175,
            "State": "SUCCEEDED",
            "CompletionDateTime": 1565649056.6229999
        },
        "Statistics": {
            "DataScannedInBytes": 5944497,
            "DataManifestLocation": "s3://aws-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://aws-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"
    }
}
```

Wiederverwenden von Abfrageergebnissen

Wenn Sie eine Abfrage in Athena erneut ausführen, können Sie optional auswählen, ob das zuletzt gespeicherte Abfrageergebnis wiederverwendet werden soll. Diese Option kann die Leistung erhöhen und die Kosten in Bezug auf die Anzahl der gescannten Byte reduzieren. Die Wiederverwendung von Abfrageergebnissen ist beispielsweise dann sinnvoll, wenn Sie wissen, dass sich die Ergebnisse innerhalb eines bestimmten Zeitrahmens nicht ändern werden. Sie können ein Höchstalter für die Wiederverwendung von Abfrageergebnissen festlegen. Athena verwendet das gespeicherte Ergebnis, solange es nicht älter als das von Ihnen angegebene Alter ist. Weitere Informationen finden Sie unter Kosten reduzieren und die Abfrageleistung verbessern mit Amazon Athena im AWS -Big-Data-Blog.

> **Note**
>
> Das Feature zur Wiederverwendung von Abfrageergebnissen erfordert Athena-Engine-Version 3. Informationen zum Ändern von Engine-Versionen finden Sie unter Ändern von Athena-Engine-Versionen.

Schlüsselfeatures

- Die Wiederverwendung von Abfrageergebnissen ist ein Opt-In-Feature pro Abfrage. Sie können die Wiederverwendung von Abfrageergebnissen für jede einzelne Abfrage aktivieren.
- Das Höchstalter für Abfrageergebnissen kann in Minuten, Stunden oder Tagen angegeben werden. Das Höchstalter, das angegeben werden kann, entspricht 7 Tagen, unabhängig von der verwendeten Zeiteinheit. Der -Standardwert beträgt 60 Minuten.
- Wenn Sie die Wiederverwendung von Ergebnissen für eine Abfrage aktivieren, sucht Athena nach einer vorherigen Ausführung der Abfrage innerhalb derselben Arbeitsgruppe. Wenn Athena entsprechende gespeicherte Abfrageergebnisse findet, führt es die Abfrage nicht erneut aus, sondern verweist auf den vorherigen Ergebnisspeicherort oder ruft Daten von dort ab.

- Für jede Abfrage, die die Option zur Wiederverwendung von Ergebnissen aktiviert, verwendet Athena das zuletzt im Arbeitsgruppenordner gespeicherte Abfrageergebnis nur dann wieder, wenn alle folgenden Bedingungen erfüllt sind:
  - Die Abfragezeichenfolge ist eine genaue Übereinstimmung.
  - Die Datenbank und der Katalogname stimmen überein.

- Das bisherige Ergebnis ist nicht älter als das angegebene Höchstalter, bzw. nicht älter als 60 Minuten, wenn kein Höchstalter angegeben wurde.
- Athena verwendet nur eine Ausführung wieder, die genau dieselbe Ergebniskonfiguration wie die aktuelle Ausführung aufweist.
- Sie haben Zugriff auf alle Tabellen, auf die in der Abfrageergebnissen verwiesen wird.
- Sie haben Zugriff auf den Speicherort der S3-Datei, in dem das vorherige Ergebnis gespeichert ist.

Wenn eine dieser Bedingungen nicht erfüllt ist, führt Athena die Abfrage aus, ohne die zwischengespeicherten Ergebnisse zu verwenden.

Überlegungen und Einschränkungen

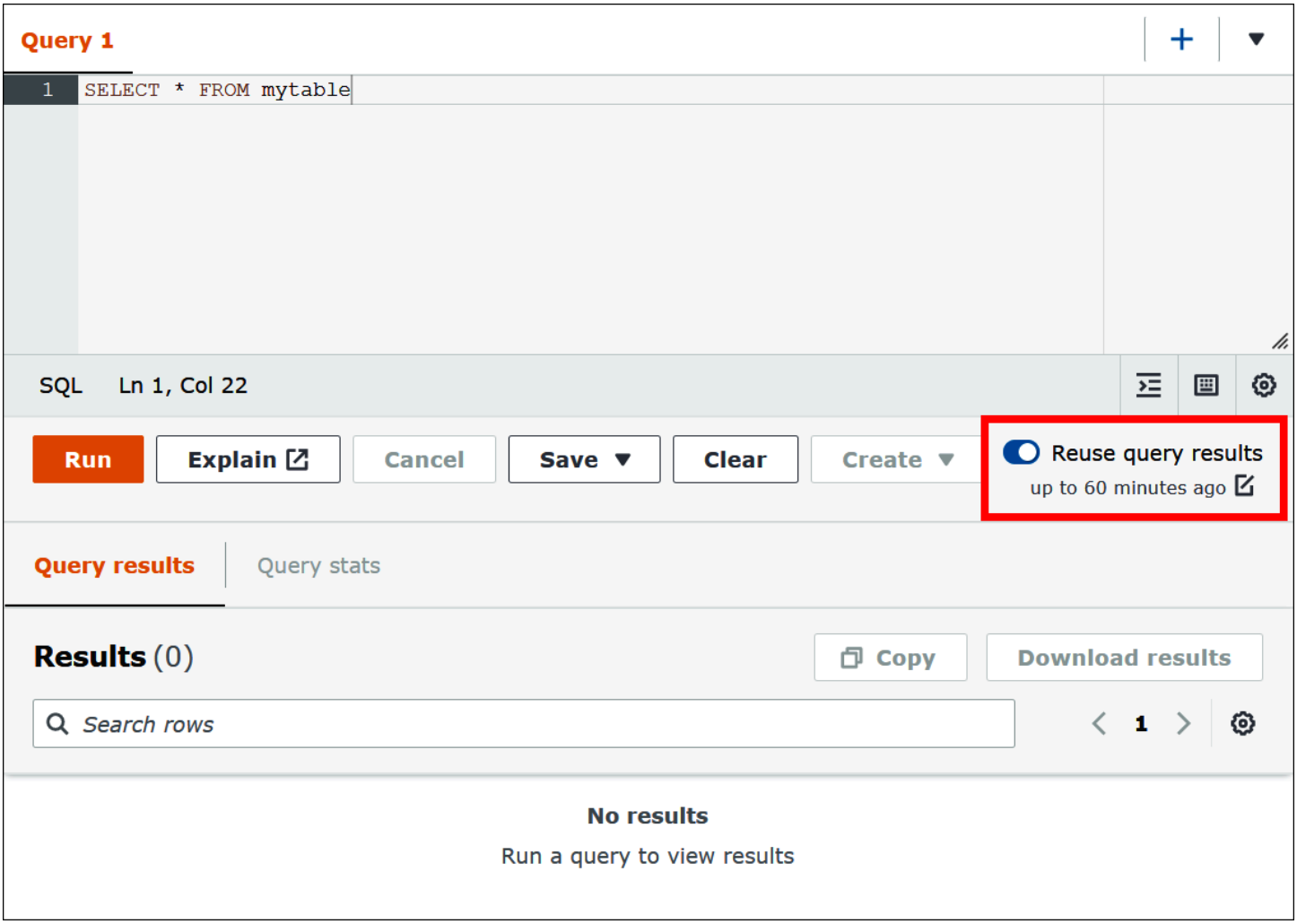
Beachten Sie bei der Verwendung des Wiederverwendungsfeatures für Abfrageergebnisse die folgenden Punkte:

- Athena verwendet Abfrageergebnisse nur innerhalb derselben Arbeitsgruppe wieder.
- Das Feature zur Wiederverwendung von Abfrageergebnissen berücksichtigt Arbeitsgruppenkonfigurationen. Wenn Sie die Ergebniskonfiguration für eine Abfrage überschreiben, ist das Feature deaktiviert.
- Apache Hive-, Apache Hudi-, Apache Iceberg- und Delta Lake-Tabellen der Linux Foundation, bei AWS Glue denen Sie registriert sind, werden unterstützt. Externe Hive-Metastores werden nicht unterstützt.
- Abfragen, die auf Verbundkataloge oder einen externen Hive-Metastore verweisen, werden nicht unterstützt.
- Die Wiederverwendung von Abfrageergebnissen wird für in Lake Formation registrierte Tabellen nicht unterstützt.
- Die Wiederverwendung von Abfrageergebnissen wird nicht unterstützt, wenn der Amazon-S3-Speicherort der Tabellenquelle als Datenstandort in Lake Formation registriert ist.
- Tabellen mit Zeilen- und Spaltenberechtigungen werden nicht unterstützt.
- Tabellen mit detaillierter Zugriffskontrolle (z. B. Spalten- oder Zeilenfilterung) werden nicht unterstützt.
- Abfragen, die auf eine Tabelle verweisen, die nicht unterstützt wird, sind nicht für die Wiederverwendung von Abfrageergebnissen geeignet.

- AAthena erfordert, dass Sie über Amazon-S3-Leseberechtigungen verfügen, damit die zuvor generierte Ausgabedatei wiederverwendet werden kann.
- Die Funktion zur Wiederverwendung von Abfrageergebnissen geht davon aus, dass der Inhalt des vorherigen Ergebnisses nicht geändert wurde. Athena überprüft die Integrität eines früheren Ergebnisses nicht, bevor es verwendet wird.
- Wenn die Abfrageergebnisse der vorherigen Ausführung gelöscht oder an einen anderen Ort in Amazon S3 verschoben wurden, werden die Abfrageergebnisse bei der nachfolgenden Ausführung derselben Abfrage nicht wiederverwendet.
- Potenziell veraltete Ergebnisse können zurückgegeben werden. Athena sucht erst nach Änderungen an den Quelldaten, wenn das von Ihnen angegebene maximale Wiederverwendungsalter erreicht ist.
- Wenn mehrere Ergebnisse zur Wiederverwendung verfügbar sind, verwendet Athena das neueste Ergebnis.
- Abfragen, die nicht deterministische Operatoren oder Funktionen wie `rand()` oder `shuffle()` verwenden, verwenden keine zwischengespeicherten Ergebnisse. Beispielsweise ist `LIMIT` ohne `ORDER BY` nicht deterministisch und wird nicht zwischengespeichert, aber `LIMIT` mit `ORDER BY` ist deterministisch und wird zwischengespeichert.
- Die Wiederverwendung von Abfrageergebnissen wird in der Athena-Konsole, in der Athena-API und im JDBC-Treiber unterstützt. Derzeit ist die ODBC-Treiberunterstützung für die Wiederverwendung von Abfrageergebnissen nur für Windows verfügbar.
- Um das Feature zur Wiederverwendung von Abfrageergebnissen mit JDBC zu verwenden, ist mindestens die Treiberversion 2.0.34.1000 erforderlich. Für ODBC ist die mindestens erforderliche Treiberversion 1.1.19.1002. Informationen zum Treiber-Download finden Sie unter Herstellen einer Verbindung mit Amazon Athena mit ODBC- und JDBC-Treibern.
- Die Wiederverwendung von Abfrageergebnissen wird für Abfragen, die mehr als einen Datenkatalog verwenden, nicht unterstützt.
- Die Wiederverwendung von Abfrageergebnissen wird für Abfragen, die mehr als 20 Tabellen enthalten, nicht unterstützt.

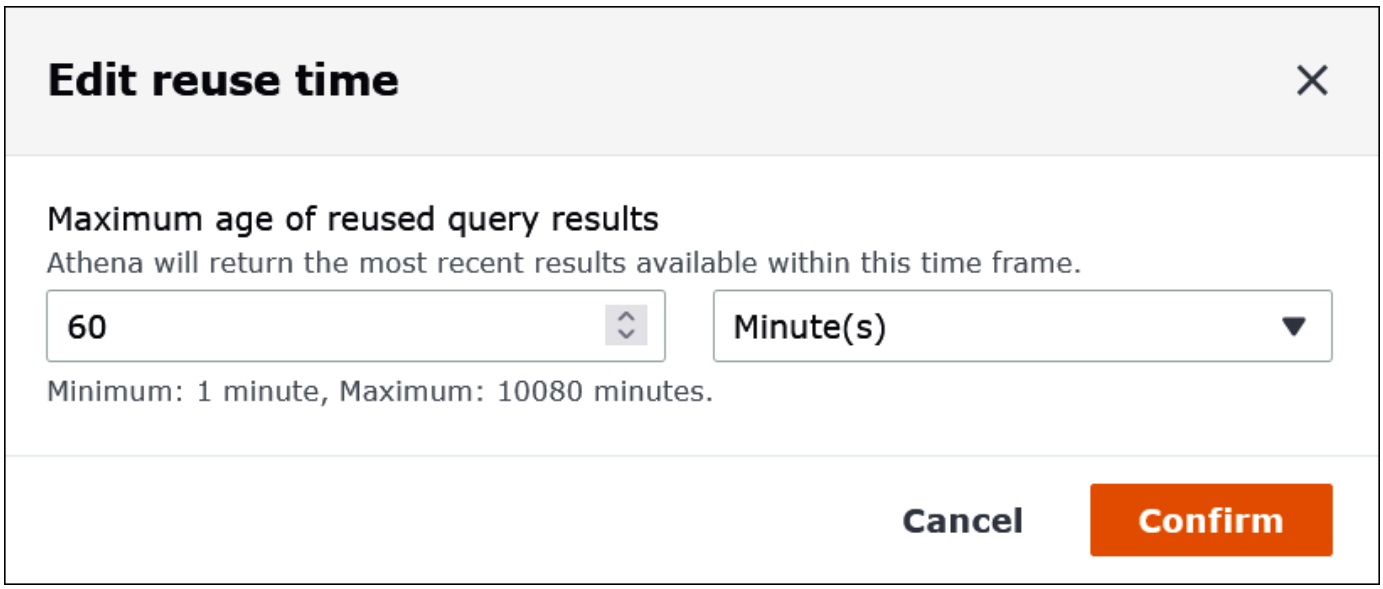
Wiederverwendung von Abfrageergebnissen in der Athena-Konsole

Um das Feature zu verwenden, aktivieren Sie die Option Reuse query results (Abfrageergebnisse wiederverwenden) im Athena-Abfrage-Editor.

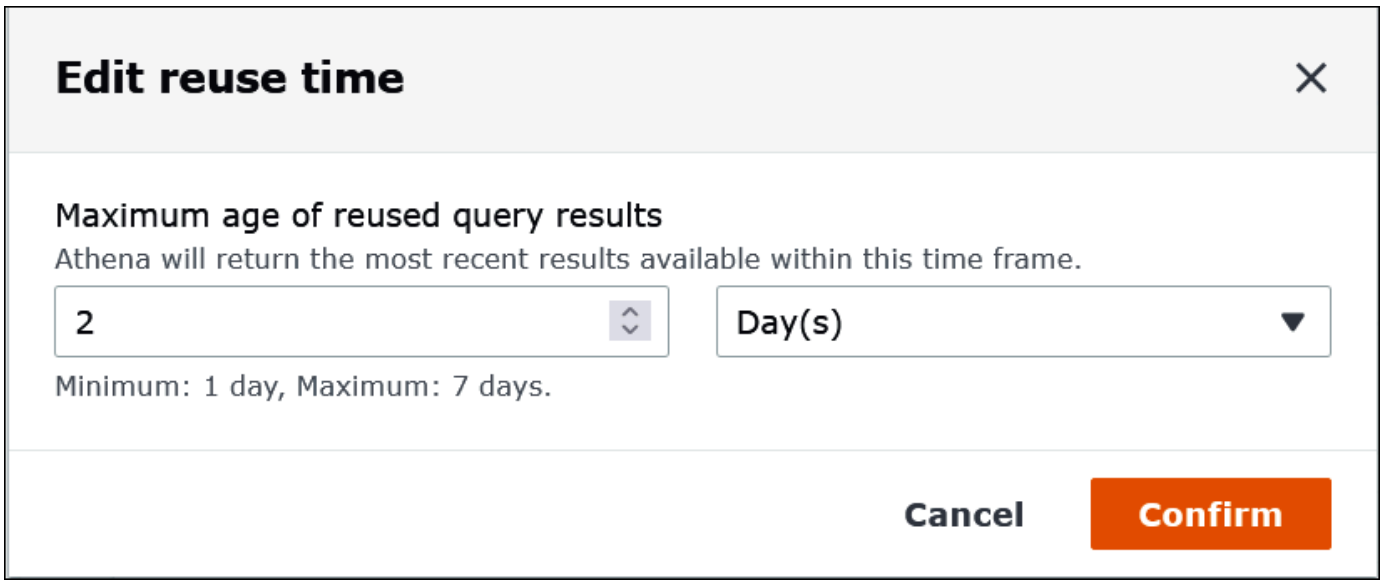


So konfigurieren Sie das Feature zur Wiederverwendung von Abfrageergebnissen

1. Wählen Sie im Athena-Abfrage-Editor unter der Option Reuse query results (Abfrageergebnisse wiederverwenden) das Bearbeitungssymbol neben up to 60 minutes ago (bis zu 60 Minuten vorher) aus.
2. Wählen Sie im Dialogfeld Edit reuse time (Wiederverwendungszeit bearbeiten) aus dem Feld auf der rechten Seite eine Zeiteinheit (Minuten, Stunden oder Tage) aus.
3. Geben Sie im Feld auf der linken Seite die Anzahl der Zeiteinheiten ein, die Sie festlegen möchten, oder wählen Sie diese aus. Die maximale Zeit, die Sie eingeben können, entspricht sieben Tagen, unabhängig von der gewählten Zeiteinheit.

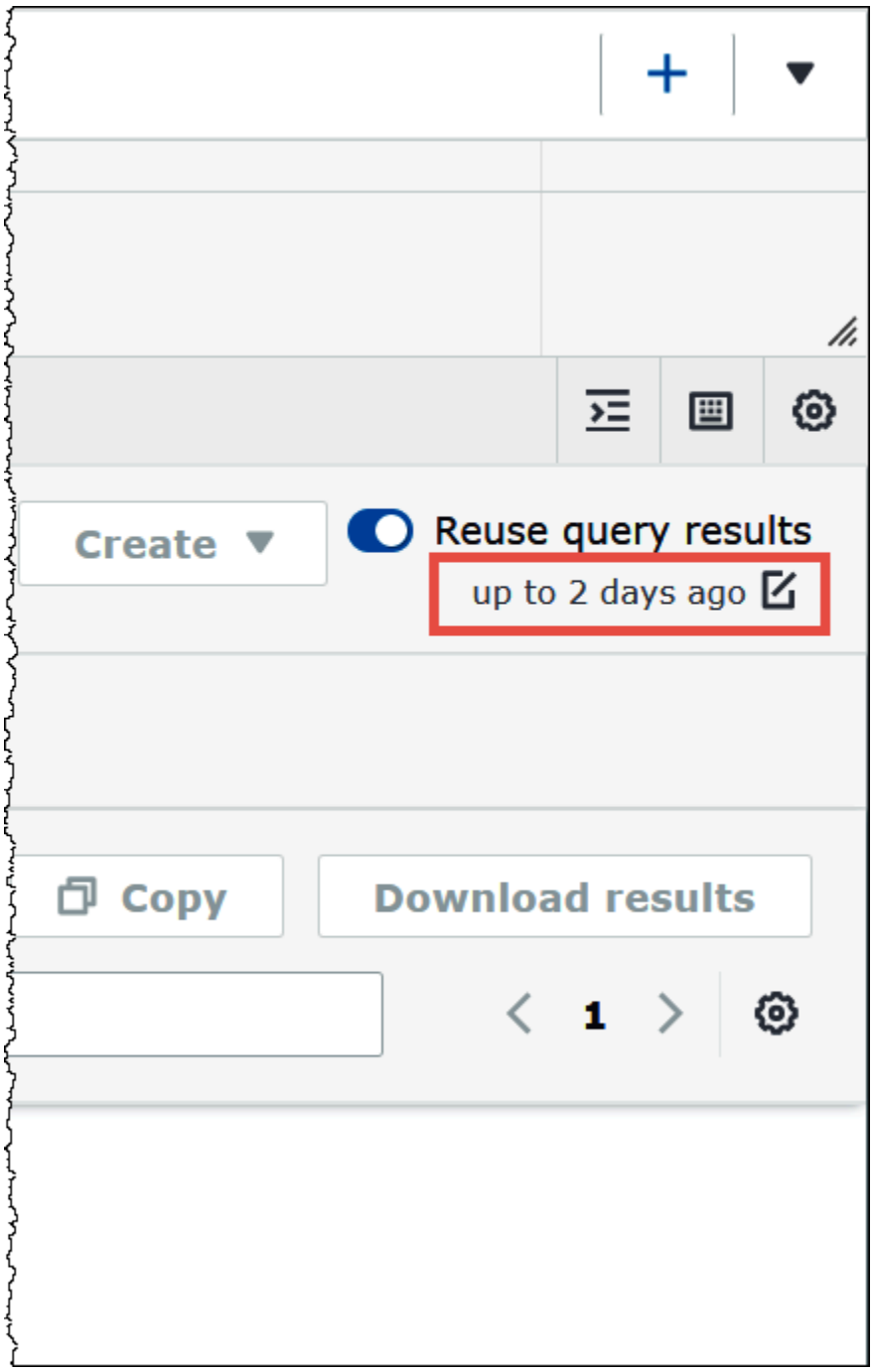


Das folgende Beispiel gibt eine maximale Wiederverwendungsdauer von zwei Tagen an.



4. Wählen Sie Bestätigen aus.

   Ein Banner bestätigt Ihre Konfigurationsänderung, und die Option Reuse query results (Abfrageergebnisse wiederverwenden) zeigt Ihre neue Einstellung an.

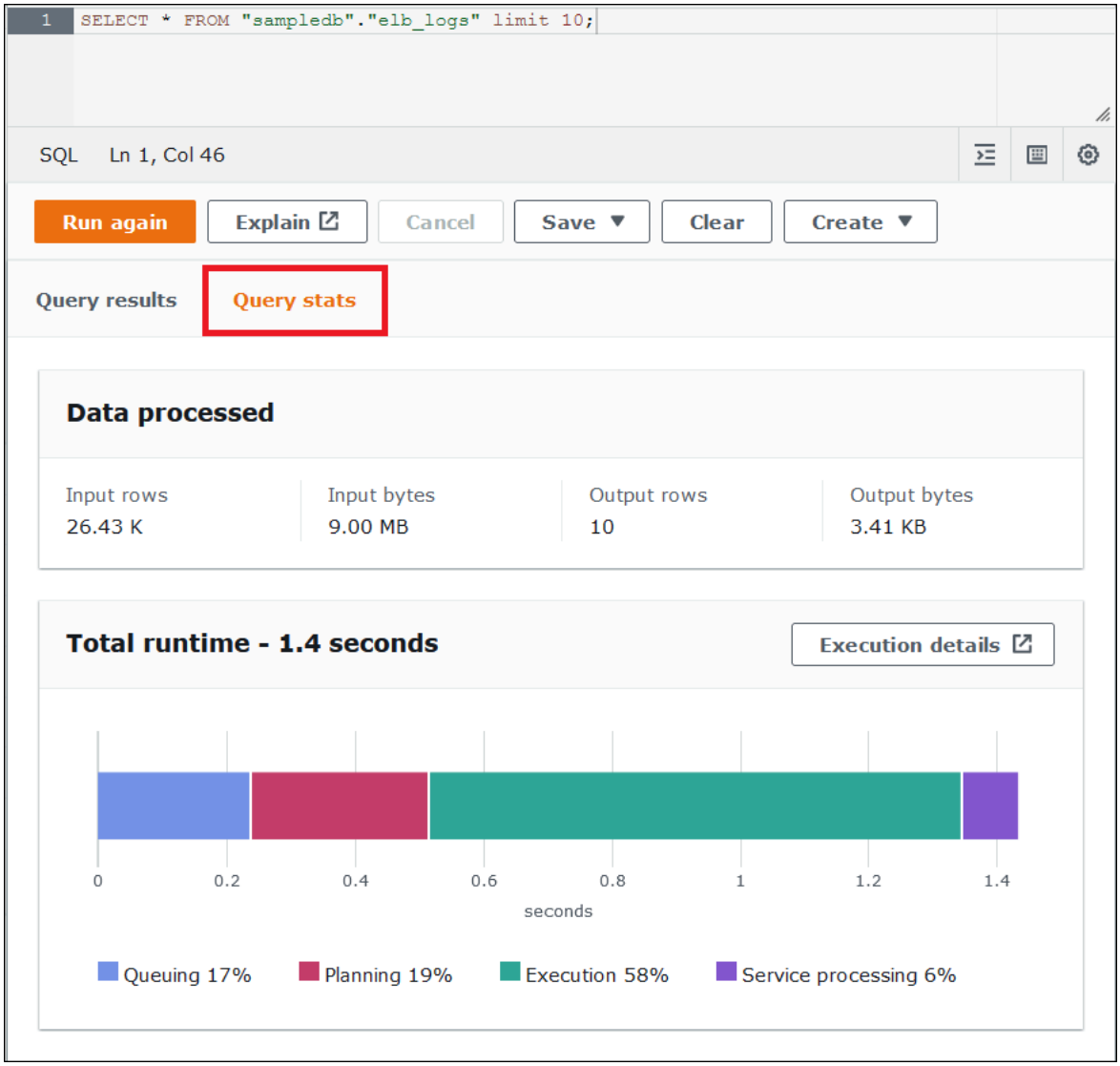


Anzeigen von Statistiken und Ausführungsdetails für abgeschlossene Abfragen

Nachdem Sie eine Abfrage ausgeführt haben, können Sie Statistiken über die verarbeiteten Eingabe- und Ausgabedaten abrufen, eine grafische Darstellung der für jede Phase der Abfrage benötigten Zeit anzeigen und interaktiv Ausführungsdetails untersuchen.

So zeigen Sie Abfragestatistiken für eine abgeschlossene Abfrage an

1. Nachdem Sie eine Abfrage im Athena-Abfrage-Editor ausgeführt haben, wählen Sie den Reiter Query stats (Abfragestatistiken) aus.



Die Registerkarte Query stats (Abfragestatistiken) enthält die folgenden Informationen:

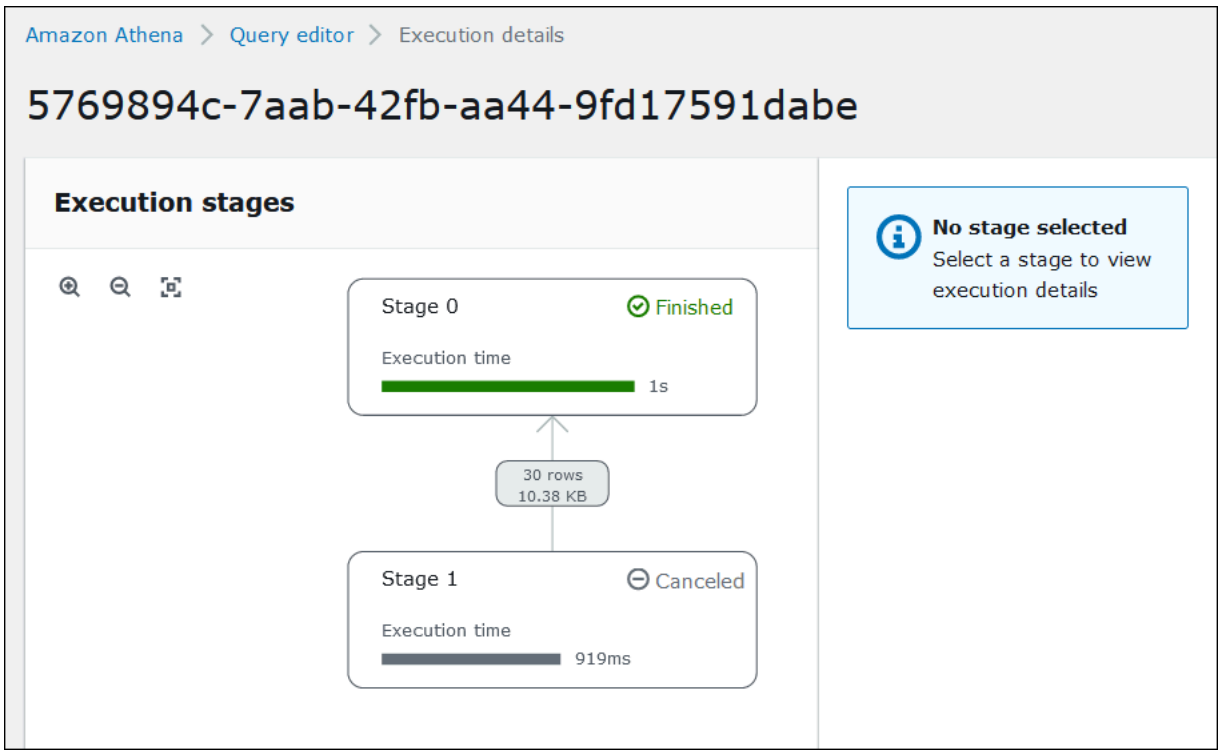
- Data processed (verarbeitete Daten) – Zeigt Ihnen die Anzahl der verarbeiteten Eingabezeilen und Bytes sowie die Anzahl der ausgegebenen Zeilen und Bytes an.
- The Total runtime (Gesamtlaufzeit) – Zeigt die Gesamtzeit an, die die Abfrage zur Ausführung benötigt hat, und stellt grafisch dar, wie viel dieser Zeit für Warteschlangen, Planung, Ausführung und Serviceverarbeitung aufgewendet wurde.

> **Note**
>
> Informationen zur Anzahl der Eingabe- und Ausgabezeilen auf Phasenebene und zur Datengröße werden nicht angezeigt, wenn für eine Abfrage in Lake Formation Filter auf Zeilenebene definiert sind.

2. Um interaktiv Informationen zur Ausführung der Abfrage zu untersuchen, wählen Sie Execution details (Ausführungsdetails) aus.

Die Seite Execution details (Ausführungsdetails) zeigt die Ausführungs-ID für die Abfrage und ein Diagramm der nullbasierten Stufen in der Abfrage. Die Stufen sind von Anfang bis Ende von unten nach oben angeordnet. Die Bezeichnung jeder Stufe zeigt die Zeit an, die die Stufe für die Ausführung benötigt hat.

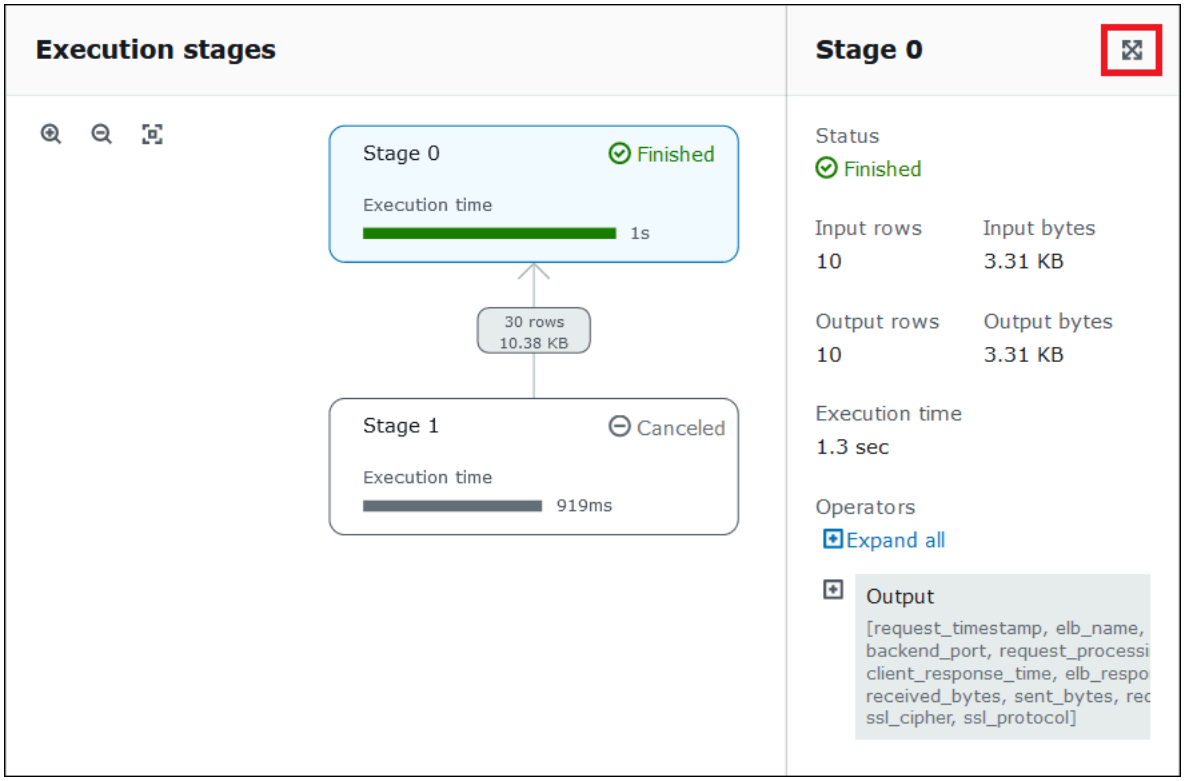
> **Note**
>
> Die Gesamtlaufzeit und die Ausführungszeit einer Abfrage unterscheiden sich häufig erheblich. Beispielsweise kann eine Abfrage mit einer Gesamtlaufzeit in Minuten die Ausführungszeit für eine Phase in Stunden anzeigen. Da es sich bei einer Phase um eine logische Recheneinheit handelt, die parallel für viele Aufgaben ausgeführt wird, entspricht die Ausführungszeit einer Phase der Gesamtausführungszeit aller ihrer Aufgaben. Trotz dieser Diskrepanz kann die Ausführungszeit einer Phase als relativer Indikator dafür nützlich sein, welche Phase in einer Abfrage am rechenintensivsten war.



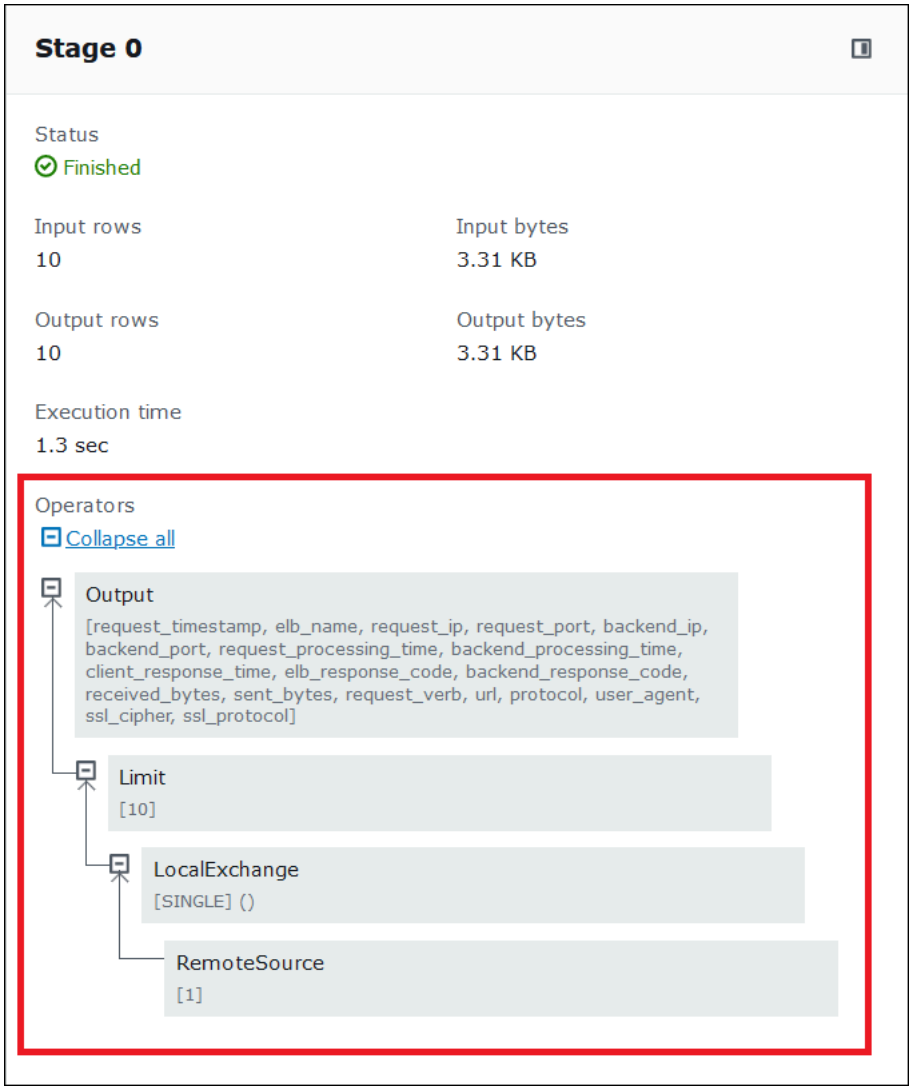
Verwenden Sie die folgenden Optionen, um im Diagramm zu navigieren:

- Zum Vergrößern oder Verkleinern scrollen Sie mit der Maus oder verwenden Sie die Vergrößerungssymbole.
- Um das Diagramm an den Bildschirm anzupassen, wählen Sie die Option Zoom to fit (Auf passende Größe zoomen) aus.
- Zum Bewegen des Diagramms ziehen Sie den Mauszeiger.

3. Um mehr Details zu einer Stufe einzusehen, wählen Sie die Stufe aus. Der Stufendetailbereich auf der rechten Seite zeigt die Anzahl der Ein- und Ausgaben von Zeilen und Bytes sowie einen Operator-Baum an.



4. Um die Details der Stufe in voller Breite anzuzeigen, wählen Sie das Erweiterungssymbol oben rechts im Detailbereich aus.
5. Um Informationen über die Teile der Stufe zu erhalten, erweitern Sie ein oder mehrere Elemente im Operator-Baum.



Weitere Informationen zu Ausführungsdetails finden Sie unter Die Ergebnisse der Athena-EXPLAIN-Anweisung verstehen.

Weitere Informationen finden Sie auch unter

Weitere Informationen finden Sie in den folgenden Ressourcen.

Anzeigen von Ausführungsplänen für SQL-Abfragen

Verwenden von EXPLAIN und EXPLAIN ANALYZE in Athena

Arbeiten mit Ansichten

Eine Ansicht in Amazon Athena ist eine logische, keine physische Tabelle. Die Abfrage, die eine Ansicht definiert, wird jedes Mal ausgeführt, wenn in einer Abfrage auf die Ansicht verwiesen wird.

Sie können eine Ansicht aus einer `SELECT`-Abfrage erstellen und dann in zukünftigen Abfragen auf diese Ansicht verweisen. Weitere Informationen finden Sie unter CREATE VIEW.

Themen

Einsetzen von Ansichten – wann sinnvoll?

Sie können Ansichten für folgende Zwecke erstellen:

- Abfrage einer Teilmenge von Daten. Sie können beispielsweise eine Ansicht mit einer Teilmenge der Spalten aus der ursprünglichen Tabelle erstellen. Dadurch werden Abfragen von Daten vereinfacht.
- Kombinieren mehrerer Tabellen in einer Abfrage. Wenn Sie mehrere Tabellen haben und sie mit `UNION ALL` kombinieren möchten, können Sie mit diesem Ausdruck eine Ansicht erstellen, um Abfragen für die kombinierten Tabellen zu vereinfachen.
- Blenden Sie die Komplexität der vorhandenen Basis-Abfragen aus und vereinfachen Sie die Abfragen, die die Benutzer ausführen. Basis-Abfragen sind häufig Joins zwischen Tabellen, Ausdrücke in der Spaltenliste und andere SQL-Syntax, die schwer zu verstehen und zu debuggen ist. Sie können eine Ansicht erstellen, mit der die Komplexität ausgeblendet wird und die Abfragen vereinfacht werden.
- Experimentieren mit Optimierungstechniken und erstellen optimierter Abfragen. Wenn Sie beispielsweise eine Kombination von `WHERE`-Bedingungen, `JOIN`-Reihenfolge oder andere

Ausdrücke finden, die eine optimale Leistung zeigen, erstellen Sie eine Ansicht mit diesen Klauseln und Ausdrücken. Anwendungen können anschließend relativ einfache Abfragen für diese Ansicht ausführen. Wenn Sie zu einem späteren Zeitpunkt eine bessere Möglichkeit finden, die ursprüngliche Abfrage zu optimieren, wenn Sie die Ansicht neu erstellen, profitieren alle Anwendungen sofort von den Vorteilen der optimierten Basis-Abfrage.

- Ausblenden zugrunde liegender Tabellen- und Spaltennamen und Minimieren von Wartungsproblemen, wenn sich diese Namen ändern. In diesem Fall erstellen Sie die Ansicht mit den neuen Namen. Alle Abfragen, die die Ansicht anstelle der zugrunde liegenden Tabellen verwenden, werden weiterhin ohne Änderungen ausgeführt.

Unterstützte Aktionen für Ansichten in Athena

Athena unterstützt die folgenden Aktionen für Ansichten. Sie können diese Befehle im Abfrage-Editor ausführen.

Statement	Beschreibung
CREATE VIEW	Erstellt eine neue Ansicht aus einer angegebenen `SELECT`-Abfrage. Weitere Informationen finden Sie unter Erstellen von Ansichten. Mit der optionalen `OR REPLACE`-Klausel können Sie die vorhandene Ansicht aktualisieren, indem Sie sie ersetzen.
DESCRIBE VIEW	Zeigt die Liste der Spalten für die benannte Ansicht an. Auf diese Weise können Sie die Attribute einer komplexen Ansicht untersuchen.
DROP VIEW	Löscht eine vorhandene Ansicht. Mit der optionalen `IF EXISTS`-Klausel wird der Fehler unterdrückt, falls die Ansicht nicht existiert.
SHOW CREATE VIEW	Zeigt die SQL-Anweisung an, die die angegebene Ansicht erstellt.
SHOW VIEWS	Listet die Ansichten in der angegebenen Datenbank oder in der aktuellen Datenbank auf, wenn Sie den Datenbanknamen weglassen . Verwenden Sie die optionale `LIKE`-Klausel mit einem regulären Ausdruck, um die Liste der Ansichtsnamen einzugrenzen. Im linken Bereich der Konsole sehen Sie die Liste der Ansichten ebenfalls.
SHOW COLUMNS	Führt die Spalten im Schema für eine Ansicht auf.

Überlegungen für Ansichten

Die folgenden Überlegungen beziehen sich auf das Erstellen und Verwenden von Ansichten in Athena:

- In Athena können Sie eine Vorschau der Ansichten anzeigen und mit ihnen arbeiten, die in der Athena-Konsole erstellt wurden, in der AWS Glue Data Catalog, falls Sie darauf migriert haben, oder mit Presto, das auf dem Amazon EMR-Cluster ausgeführt wird, der mit demselben Katalog verbunden ist. Sie können keine Vorschau für Ansichten anzeigen oder diese zu Athena hinzufügen, wenn sie auf andere Weise erstellt wurden.
- Wenn Sie Ansichten über den AWS Glue Datenkatalog erstellen, müssen Sie den `PartitionKeys` Parameter angeben und seinen Wert wie folgt auf eine leere Liste setzen:. `"PartitionKeys":[]` Andernfalls schlägt die Ansichtsabfrage in Athena fehl. Im folgenden Beispiel ist eine Ansicht dargestellt, die aus dem Datenkatalog mit `"PartitionKeys":[]` erstellt wurde:

```
aws glue create-table
--database-name mydb
--table-input '{
"Name":"test",
  "TableType": "EXTERNAL_TABLE",
  "Owner": "hadoop",
  "StorageDescriptor":{
     "Columns":[{
           "Name":"a","Type":"string"},{"Name":"b","Type":"string"}],
   "Location":"s3://xxxxx/Oct2018/25Oct2018/",
   "InputFormat":"org.apache.hadoop.mapred.TextInputFormat",
   "OutputFormat": "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat",
   "SerdeInfo":{"SerializationLibrary":"org.apache.hadoop.hive.serde2.OpenCSVSerde",
   "Parameters":{"separatorChar": "|", "serialization.format":
 "1"}}},"PartitionKeys":[]}'
```

- Wenn Sie Athena-Ansichten im Datenkatalog erstellt haben, behandelt Ansichten als Tabellen. Sie können die feinkörnige Zugriffskontrolle auf Tabellenebene im Datenkatalog verwenden, um den Zugriff auf diese Ansichten einzuschränken.
- Athena verhindert, dass Sie rekursive Ansichten ausführen, und zeigt in solchen Fällen eine Fehlermeldung an. Eine rekursive Ansicht ist eine Abfrage, die auf sich selbst verweist.
- Athena zeigt eine Fehlermeldung an, wenn veraltete Ansichten erkannt werden. Eine veraltete Ansicht wird gemeldet, wenn eines der folgenden Ereignisse auftritt:

  - Die Ansicht verweist auf Tabellen oder Datenbanken, die nicht vorhanden sind.
  - Eine Schema- oder Metadatenänderung wird in einer referenzierten Tabelle vorgenommen.
  - Eine referenzierte Tabelle wird gelöscht und mit einem anderen Schema oder einer anderen Konfiguration neu erstellt.
- Sie können verschachtelte Ansichten erstellen und ausführen, solange die Abfrage hinter der verschachtelten Ansicht gültig ist und die Tabellen und Datenbanken vorhanden ist.

Einschränkungen für Ansichten

- Athena-Ansichtsnamen dürfen keine Sonderzeichen enthalten außer Unterstriche (`_`). Weitere Informationen finden Sie unter Namen für Tabellen, Datenbanken und Spalten.
- Vermeiden Sie reservierte Schlüsselwörter für die Benennung von Ansichten. Wenn Sie reservierte Schlüsselwörter verwenden, umschließen Sie sie in Ihren Abfragen in Ansichten mit doppelten Anführungszeichen. Siehe Reservierte Schlüsselwörter.
- Sie können in Athena erstellte Ansichten nicht mit externen Hive-Metaspeichern oder UDFs verwenden. Weitere Informationen zum Arbeiten mit Ansichten, die extern in Hive erstellt wurden, finden Sie unter Arbeiten mit Hive-Ansichten.
- Sie können keine Ansichten mit raumbezogenen Funktionen verwenden.
- Sie können keine Ansichten zum Verwalten der Zugriffskontrolle auf Daten in Amazon S3 verwenden. Um eine Ansicht abzufragen, benötigen Sie Berechtigungen für den Zugriff auf die in Amazon S3 gespeicherten Daten. Weitere Informationen finden Sie unter Zugriff auf Amazon S3.
- Die kontenübergreifende Abfrage von Ansichten wird zwar sowohl in der Athena-Engine-Version 2 als auch in der Athena-Engine-Version 3 unterstützt, Sie können jedoch keine Ansicht erstellen, die eine kontoübergreifende AWS Glue Data Catalog enthält. Informationen zum kontenübergreifenden Datenkatalog-Zugriff finden Sie unter Kontoübergreifender Zugriff auf AWS Glue -Datenkataloge.
- Die versteckten Metadatenspalten `$bucket`, `$file_modified_time`, `$file_size` und `$partition` werden für Ansichten in Athena nicht unterstützt. Informationen zur Verwendung der `$path`-Metadatenspalte in Athena finden Sie unter Abrufen der Dateispeicherorte für Quelldaten in Amazon S3 .

Arbeiten mit Ansichten in der Konsole

In der Athena-Konsole stehen Ihnen folgende Funktionen zur Verfügung:

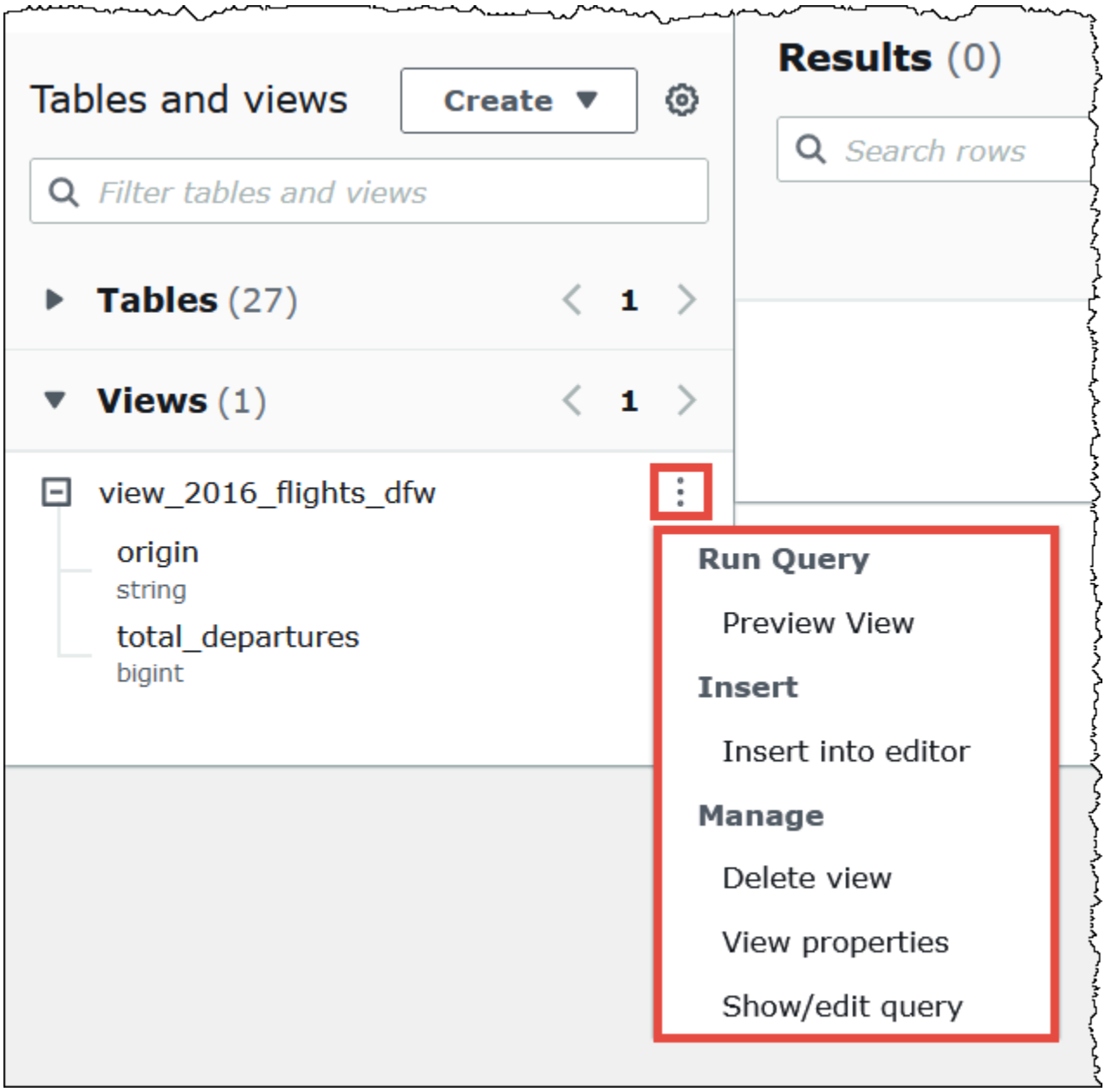
- Zugriff auf alle Ansichten über den linken Bereich, in dem Tabellen aufgelistet sind.

- Filtern von Ansichten.
- Ansichten in der Vorschau anzeigen, ihre Eigenschaften anzeigen, Ansichten bearbeiten oder löschen.

So zeigen Sie die Aktionen für eine Ansicht an

Eine Ansicht wird nur dann in der Konsole sichtbar, wenn Sie sie bereits erstellt haben.

1. Wählen Sie in der Athena-Konsole Views (Ansichten), und wählen Sie dann eine Ansicht aus, um sie zu erweitern und die Spalten in der Ansicht anzuzeigen.
2. Wählen Sie die drei vertikalen Punkte neben der Ansicht aus, um eine Liste von Aktionen für die Ansicht anzuzeigen.



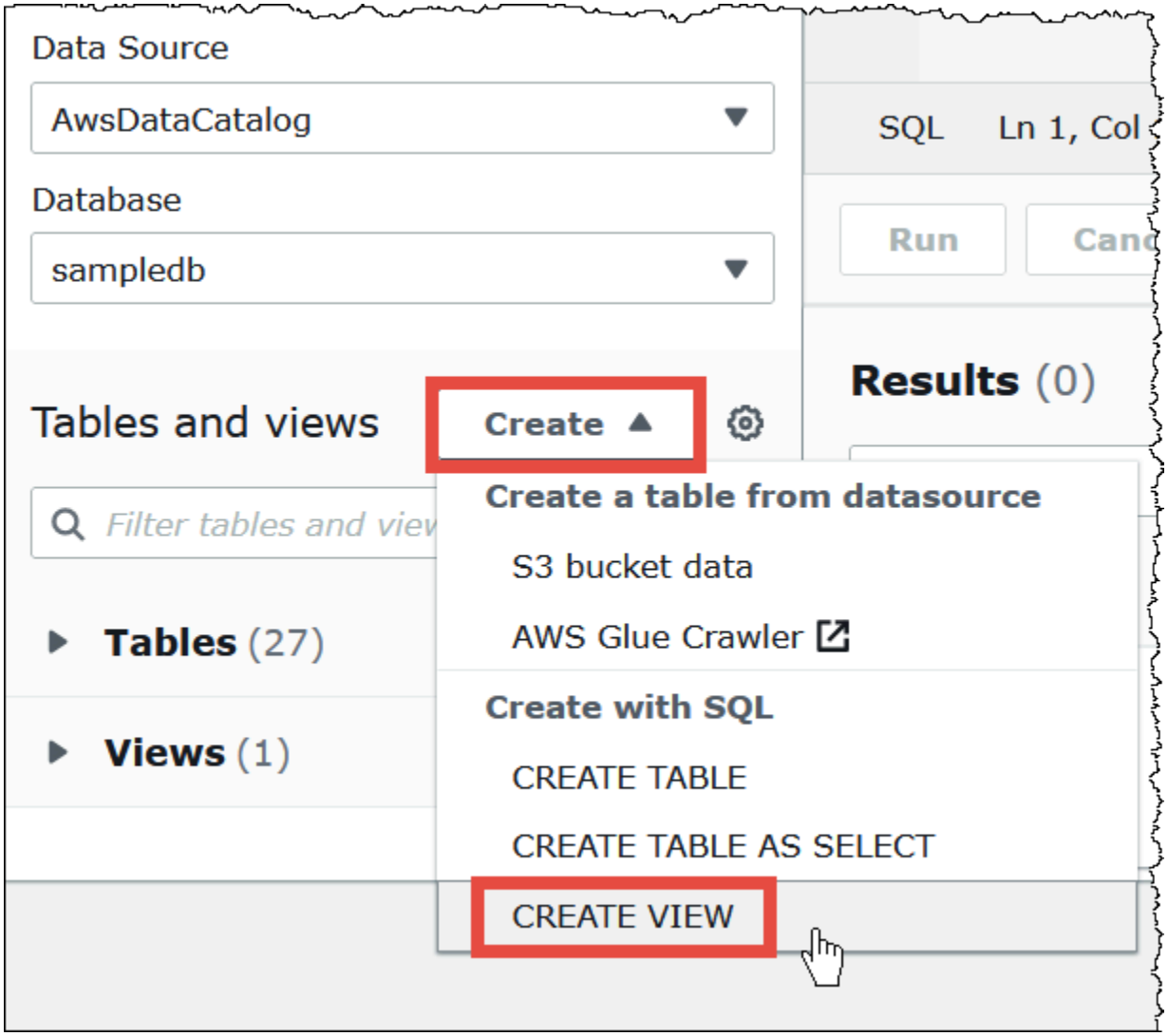
3. Wählen Sie Aktionen aus, um eine Vorschau der Ansicht anzuzeigen, den Ansichtsnamen in den Abfrage-Editor einzufügen, die Ansicht zu löschen, die Eigenschaften der Ansicht anzuzeigen oder die Ansicht im Abfrage-Editor anzuzeigen und zu bearbeiten.

Erstellen von Ansichten

Sie können eine Ansicht in der Athena-Konsole erstellen, indem Sie eine Vorlage verwenden oder eine vorhandene Abfrage ausführen.

So erstellen Sie eine Ansicht mithilfe einer Vorlage

1. Wählen Sie in der Athena-Konsole neben Tables and views (Tabellen und Ansichten) Create (Erstellen) und dann Create view (Ansicht erstellen) aus.



   Diese Aktion platziert eine bearbeitbare Ansichtsvorlage in den Abfrage-Editor.

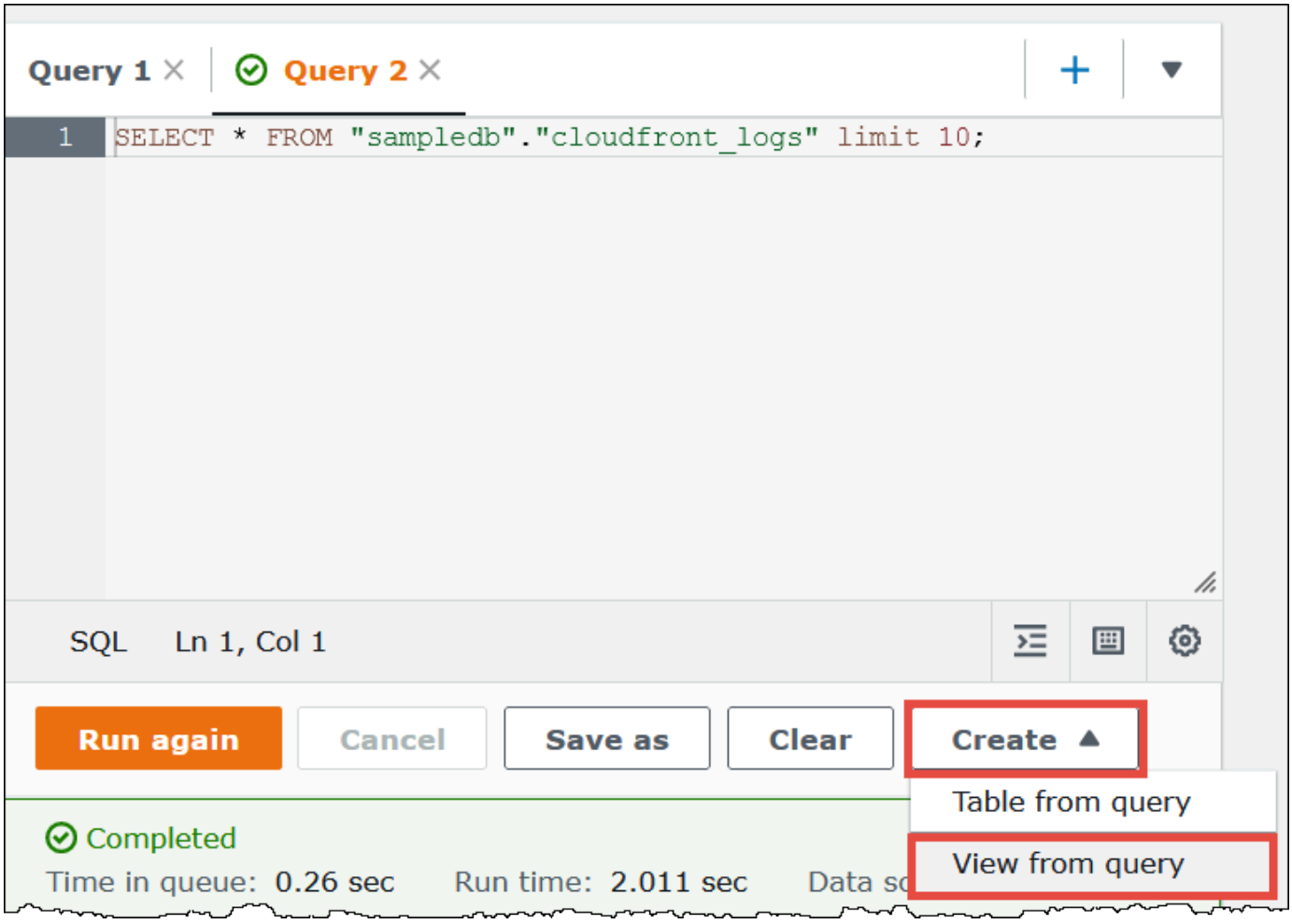
2. Bearbeiten Sie die Anzeigevorlage entsprechend Ihren Anforderungen. Wenn Sie einen Namen für die Ansicht in die Anweisung eingeben, denken Sie daran, dass Ansichtsnamen keine anderen Sonderzeichen als den Unterstrich (_)enthalten können. Siehe Namen für Tabellen, Datenbanken und Spalten. Vermeiden Sie Reservierte Schlüsselwörter für die Benennung von Ansichten.

   Weitere Informationen zum Erstellen von Ansichten finden Sie unter CREATE VIEW und Beispiele für Ansichten.

3. Wählen Sie Run (Ausführen), um die Ansicht zu erstellen. Die Ansicht wird in der Liste der Ansichten in der Athena-Konsole angezeigt.

So erstellen Sie eine Ansicht aus einer vorhandenen Abfrage

1. Verwenden Sie den Athena-Abfrage-Editor, um eine vorhandene Abfrage auszuführen.
2. Wählen Sie im Fenster des Abfrage-Editors Create (Erstellen) und dann View from query (Aus Abfrage anzeigen) aus.



3. Geben Sie im Dialogfeld Create View (Ansicht erstellen) einen Namen für die Ansicht ein und wählen Sie Create (Erstellen). Ansichtsnamen dürfen keine Sonderzeichen enthalten außer Unterstriche (_). Siehe Namen für Tabellen, Datenbanken und Spalten. Vermeiden Sie Reservierte Schlüsselwörter für die Benennung von Ansichten.

   Athena fügt die Ansicht zur Liste der Ansichten in der Konsole hinzu und zeigt die`CREATE VIEW`-Anweisung für die Ansicht im Abfrage-Editor.

Hinweise

- Wenn Sie eine Tabelle löschen, auf der eine Tabelle basiert, und anschließend versuchen, die Ansicht auszuführen, wird in Athena eine Fehlermeldung angezeigt.
- Sie können eine verschachtelte Ansicht erstellen. Dabei handelt es sich um eine Ansicht auf eine vorhandene Ansicht. Athena verhindert, dass Sie eine rekursive Ansicht ausführen, die auf sich selbst verweist.

Beispiele für Ansichten

Verwenden Sie zum Anzeigen der Syntax für die Ansichtsabfrage SHOW CREATE VIEW.

Example Beispiel 1

Stellen Sie sich folgende zwei Tabellen vor: eine Tabelle `employees` mit zwei Spalten, `id` und `name`, und eine Tabelle `salaries` mit zwei Spalten, `id` und `salary`.

In diesem Beispiel erstellen wir eine Ansicht mit dem Namen `name_salary` als `SELECT`-Abfrage, die eine Liste mit IDs abruft, die Gehältern aus den Tabellen `employees` und `salaries` zugeordnet sind:

```
CREATE VIEW name_salary AS
SELECT
 employees.name,
 salaries.salary
FROM employees, salaries
WHERE employees.id = salaries.id
```

Example Beispiel 2

Im folgenden Beispiel erstellen wir eine Ansicht mit dem Namen `view1` , mit der Sie die komplexere Abfrage-Syntax ausblenden.

Diese Ansicht läuft auf zwei Tabellen, `table1` und `table2`, wobei jede Tabelle eine andere `SELECT`-Abfrage ist. Die Ansicht wählt Spalten aus `table1` und fügt die Ergebnisse mit `table2` zusammen. Der Join basiert auf der Spalte `a`, die in beiden Tabellen vorhanden ist.

```
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;
```

Weitere Informationen zu Verbundabfrageansichten finden Sie unter Verbundansichten abfragen.

AWS Glue Data Catalog Ansichten verwenden

Bei diesem Feature handelt es sich um eine Vorabversion, die Änderungen unterliegt. Weitere Informationen dazu finden Sie in den Abschnitten „Betas“ und „Vorschauen“ im Dokument AWS - Servicebedingungen.

Verwenden Sie AWS Glue Data Catalog Ansichten, wenn Sie eine einzige gemeinsame Ansicht für Amazon Athena und Amazon Redshift AWS-Services wünschen. In Data-Catalog-Ansichten werden die Zugriffsberechtigungen durch den Benutzer definiert, der die Ansicht erstellt hat, und nicht durch den Benutzer, der die Ansicht abfragt. Diese Methode zum Gewähren von Berechtigungen wird als Definer-Semantik bezeichnet.

Die folgenden Anwendungsfälle veranschaulichen, wie Sie Data-Catalog-Ansichten verwenden können.

- Bessere Zugriffskontrolle – Sie erstellen eine Ansicht, die den Datenzugriff basierend auf der vom Benutzer benötigten Berechtigungsebene einschränkt. Mithilfe von Data-Catalog-Ansichten können Sie beispielsweise verhindern, dass Mitarbeiter, die nicht in der Personalabteilung arbeiten, personenbezogene Daten sehen.
- Vollständige Datensätze sicherstellen – Durch die Anwendung bestimmter Filter auf Ihre Data-Catalog-Ansicht stellen Sie sicher, dass die Datensätze in einer Data-Catalog-Ansicht immer vollständig sind.

- Verbesserte Sicherheit – In Data-Catalog-Ansichten muss die Abfragedefinition, mit der die Ansicht erstellt wird, intakt sein, damit die Ansicht erstellt werden kann. Dadurch sind Data-Catalog-Ansichten weniger anfällig für SQL-Befehle von böswilligen Akteuren.
- Zugriff auf zugrunde liegende Tabellen verhindern – Definer-Semantik ermöglicht es Benutzern, auf eine Ansicht zuzugreifen, ohne ihnen die zugrunde liegende Tabelle zur Verfügung zu stellen. Zugriff auf die Tabellen benötigt nur der Benutzer, der die Ansicht definiert.

Definitionen der Data-Catalog-Ansicht werden im AWS Glue Data Catalog gespeichert. Dies bedeutet, dass Sie AWS Lake Formation verwenden können, um den Zugriff durch Ressourcengewährung, Spaltengewährung oder Tag-basierte Zugriffskontrollen zu gewähren. Weitere Informationen zum Gewähren und Widerrufen des Zugriffs in Lake Formation finden Sie unter Gewähren und Widerrufen von Berechtigungen für Data-Catalog-Ressourcen im AWS Lake Formation -Entwicklerhandbuch.

Berechtigungen

Für Data-Catalog-Ansichten sind drei Rollen erforderlich: `Lake Formation Admin`, `Definer` und `Invoker`.

- **`Lake Formation Admin`** – Hat Zugriff auf die Konfiguration aller Lake-Formation-Berechtigungen.
- **`Definer`** – Erstellt die Data-Catalog-Ansicht. Die `Definer`-Rolle muss über vollständige `SELECT`-Berechtigungen für alle zugrunde liegenden Tabellen verfügen, auf die die Ansichtsdefinition verweist.
- **`Invoker`** – Kann die Data-Catalog-Ansicht abfragen oder deren Metadaten prüfen.

Die Vertrauensbeziehungen der `Definer` Rolle müssen den Dienstleitern AWS Glue und Lake Formation die `sts:AssumeRole` Aktion ermöglichen, wie im folgenden Beispiel gezeigt.

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

```
                    "lakeformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Für den Athena-Zugriff sind außerdem IAM-Berechtigungen erforderlich. Weitere Informationen finden Sie unter AWS verwaltete Richtlinien für Amazon Athena.

Einschränkungen

- Data-Catalog-Ansichten können nicht auf andere Ansichten verweisen.
- Sie können in der Ansichtsdefinition auf bis zu 10 Tabellen verweisen.
- Zugrundeliegende Tabellen müssen bei Lake Formation registriert sein.
- Beim `DEFINER`-Prinzipal kann es sich nur um eine IAM-Rolle handeln.
- Die `DEFINER`-Rolle muss über vollständige `SELECT`-Berechtigungen (gewährbar) für die zugrunde liegenden Tabellen verfügen.
- `UNPROTECTED`-Data-Catalog-Ansichten werden nicht unterstützt.
- Benutzerdefinierte Funktionen (UDFs) werden in der Ansichtsdefinition nicht unterstützt.
- Verbunddatenquellen von Athena können nicht in Data-Catalog-Ansichten verwendet werden.
- Data-Catalog-Ansichten werden für externe Hive-Metastores nicht unterstützt.
- Athena zeigt eine Fehlermeldung an, wenn veraltete Ansichten erkannt werden. Eine veraltete Ansicht wird gemeldet, wenn eines der folgenden Ereignisse auftritt:
  - Die Ansicht verweist auf Tabellen oder Datenbanken, die nicht vorhanden sind.
  - Eine Schema- oder Metadatenänderung wird in einer referenzierten Tabelle vorgenommen.
  - Eine referenzierte Tabelle wird gelöscht und mit einem anderen Schema oder einer anderen Konfiguration neu erstellt.

Erstellen einer Data-Catalog-Ansicht

Die folgende Beispielsyntax veranschaulicht, wie ein Benutzer der `Definer`-Rolle die Data-Catalog-Ansicht `orders_by_date` erstellt. Im Beispiel wird davon ausgegangen, dass die `Definer`-Rolle über vollständige `SELECT`-Berechtigungen für die `orders`-Tabelle in der `default`-Datenbank verfügt.

```
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
```

Abfrage einer Data-Catalog-Ansicht

Nachdem die Ansicht erstellt wurde, kann der `Lake Formation Admin` die Berechtigungen `SELECT` für die Data-Catalog-Ansicht den `Invoker`-Auftraggebern gewähren. Die `Invoker`-Prinzipale können dann die Ansicht abfragen, ohne Zugriff auf die zugrunde liegenden Basistabellen zu haben, auf die in der Ansicht verwiesen wird. Im Folgenden finden Sie ein Beispiel für eine `Invoker`-Abfrage.

```
SELECT * from orders_by_date where price > 5000
```

Aktualisieren einer Data-Catalog-Ansicht

Der `Lake Formation Admin` oder der `Definer` kann die `ALTER VIEW UPDATE DIALECT`-Syntax verwenden, um die Ansichtsdefinition zu aktualisieren. Im folgenden Beispiel wird die Ansichtsdefinition so geändert, dass Spalten aus der `returns`-Tabelle statt aus der `orders`-Tabelle ausgewählt werden.

```
ALTER VIEW orders_by_date UPDATE DIALECT
AS
SELECT return_date, sum(totalprice) AS price
FROM returns
WHERE order_city = 'SEATTLE'
GROUP BY orderdate
```

Weitere Informationen zur Syntax zum Erstellen und Verwalten von Data-Catalog-Ansichten finden Sie unter Syntax der Glue-Data-Catalog-Ansicht.

Syntax der Glue-Data-Catalog-Ansicht

Bei diesem Feature handelt es sich um eine Vorabversion, die Änderungen unterliegt. Weitere Informationen dazu finden Sie in den Abschnitten „Betas“ und „Vorschauen“ im Dokument AWS - Servicebedingungen.

In diesem Abschnitt werden die DDL-Befehle (Data Definition Language) zum Erstellen und Verwalten AWS Glue Data Catalog von Ansichten beschrieben.

ANSICHTSDIALEKT ÄNDERN

Sie können Data-Catalog-Ansichten aktualisieren, indem Sie entweder einen Engine-Dialekt hinzufügen oder einen vorhandenen Engine-Dialekt aktualisieren oder löschen. Nur der `Lake FormationAdmin` und der `Definer` (der Benutzer, der die Ansicht erstellt hat) verfügen über die Berechtigung, die `ALTER VIEW DIALECT`-Anweisung in einer Data-Catalog-Ansicht zu verwenden.

Syntax

```
ALTER VIEW name [ FORCE ] [ ADD|UPDATE ] DIALECT AS query
```

```
ALTER VIEW name [ DROP ] DIALECT
```

FORCE

Das `FORCE`-Schlüsselwort führt dazu, dass widersprüchliche Engine-Dialektinformationen in einer Ansicht mit der neuen Definition überschrieben werden. Das `FORCE`-Schlüsselwort ist hilfreich, wenn eine Aktualisierung einer Data-Catalog-Ansicht zu widersprüchlichen Ansichtsdefinitionen in den vorhandenen Engine-Dialekten führt. Angenommen, eine Data-Catalog-Ansicht verfügt sowohl über den Athena- als auch den Amazon-Redshift-Dialekt und die Aktualisierung führt zu einem Konflikt mit Amazon Redshift in der Ansichtsdefinition. In diesem Fall können Sie das `FORCE`-Schlüsselwort verwenden, damit die Aktualisierung abgeschlossen werden kann und der Amazon-Redshift-Dialekt als veraltet markiert wird. Wenn als veraltet markierte Engines die Ansicht abfragen, schlägt die Abfrage fehl. Die Engines lösen eine Ausnahme aus, um veraltete Ergebnisse zu verhindern. Um dies zu beheben, aktualisieren Sie die veralteten Dialekte in der Ansicht.

ADD

Fügt der Data-Catalog-Ansicht einen neuen Engine-Dialekt hinzu. Die angegebene Engine darf nicht bereits in der Data-Catalog-Ansicht vorhanden sein.

UPDATE

Aktualisiert einen Engine-Dialekt, der bereits in der Data-Catalog-Ansicht vorhanden ist.

DROP

Löscht einen vorhandenen Engine-Dialekt aus einer Data-Catalog-Ansicht. Nachdem Sie eine Engine aus einer Data-Catalog-Ansicht gelöscht haben, kann die Data-Catalog-Ansicht nicht mehr von der gelöschten Engine abgefragt werden. Andere Engine-Dialekte in der Ansicht können die Ansicht weiterhin abfragen.

DIALEKT ALS

Führt eine Engine-spezifische SQL-Abfrage ein.

Beispiele

```
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
```

ERSTELLEN EINER GESCHÜTZTEN ANSICHT MIT MEHREREN DIALEKTEN

Erstellt eine Datenkatalogansicht in der AWS Glue Data Catalog. Eine Data-Catalog-Ansicht ist ein einzelnes Ansichtsschema, das nahtlos mit Athena und anderen SQL-Engines wie Amazon Redshift und Amazon EMR funktioniert.

Syntax

```
CREATE [ OR REPLACE ] PROTECTED MULTI DIALECT VIEW view_name
 [ SECURITY DEFINER ]
AS query
```

GESCHÜTZT

Erforderliches Schlüsselwort. Gibt an, dass die Ansicht vor Datenverlusten geschützt ist. Data-Catalog-Ansichten können nur als `PROTECTED`-Ansicht erstellt werden.

MULTI-DIALEKT

Gibt an, dass die Ansicht die SQL-Dialekte unterschiedlicher Abfrage-Engines unterstützt und daher von diesen Engines gelesen werden kann.

SICHERHEITS-DEFINER

Gibt an, dass die Definer-Semantik für diese Ansicht in Kraft ist. Definer-Semantik bedeutet, dass die effektiven Leseberechtigungen für die zugrunde liegenden Tabellen dem Prinzipal oder der Rolle gehören, der die Ansicht definiert hat, und nicht dem Prinzipal, der den tatsächlichen Lesevorgang ausführt.

OR REPLACE

Eine Data-Catalog-Ansicht kann nicht ersetzt werden, wenn in der Ansicht SQL-Dialekte von anderen Engines vorhanden sind. Wenn die aufrufende Engine über den einzigen in der Ansicht vorhandenen SQL-Dialekt verfügt, kann die Ansicht ersetzt werden.

Beispiel

Im folgenden Beispiel wird die Data-Catalog-Ansicht `orders_by_date` basierend auf einer Abfrage der `orders`-Tabelle erstellt.

```
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

Zeigt die Spaltenliste für die angegebene Data-Catalog-Ansicht an. Die `DESCRIBE`-Anweisung ähnelt der `DESCRIBE`-Anweisung für Athena-Ansichten. Im Gegensatz zu Athena-Ansichten wird die Ausgabe der Anweisung durch die Zugriffskontrolle von Lake Formation gesteuert. Die Ausgabe dieser Abfrage umfasst daher nicht alle Spalten der Ansicht, sondern nur die Spalten, auf die der Aufrufer Zugriff hat.

Syntax

```
DESCRIBE [db_name.]view_name
```

Beispiele

```
DESCRIBE orders
```

DROP VIEW

Löscht eine Data-Catalog-Ansicht nur, wenn der Dialekt der aufrufenden Engine in der Data-Catalog-Ansicht vorhanden ist. Wenn ein Benutzer beispielsweise `DROP VIEW` von Athena aus aufruft, wird die Ansicht nur gelöscht, wenn der Dialekt von Athena in der Ansicht vorhanden ist. Andernfalls schlägt die Operation fehl. Nur der Lake-Formation-Administrator und der Ansichts-Definer verfügen über die Berechtigung, die `DROP VIEW`-Anweisung für eine Data-Catalog-Ansicht zu verwenden.

Syntax

```
DROP VIEW [ IF EXISTS ] view_name
```

Beispiele

```
DROP VIEW orders_by_date
```

```
DROP FORCE VIEW IF EXISTS orders_by_date
```

Mit der optionalen `IF EXISTS`-Klausel wird der Fehler unterdrückt, falls die Ansicht nicht existiert.

SHOW_COLUMNS

Zeigt nur die Spaltennamen für eine einzelne angegebene Data-Catalog-Ansicht an. Die `SHOW COLUMNS`-Anweisung ähnelt der `SHOW COLUMNS`-Anweisung für Athena-Ansichten. Im Gegensatz zu

Athena-Ansichten wird die Ausgabe der Anweisung durch die Zugriffskontrolle von Lake Formation gesteuert. Die Ausgabe dieser Abfrage umfasst daher nicht alle Spalten der Ansicht, sondern nur die Spalten, auf die der Aufrufer Zugriff hat.

Syntax

```
SHOW COLUMNS {FROM|IN} database_name.view_name
```

```
SHOW COLUMNS {FROM|IN} view_name [{FROM|IN} database_name]
```

SHOW CREATE VIEW

Zeigt die SQL-Syntax an, mit der die Data-Catalog-Ansicht erstellt wurde. Die zurückgegebene SQL zeigt die in Athena verwendete Syntax zum Erstellen einer Ansicht. Nur der Lake-Formation-Administrator und die Prinzipale des Ansichts-Definer sind berechtigt, `SHOW CREATE VIEW` in einer Data-Catalog-Ansicht aufzurufen.

Syntax

```
SHOW CREATE VIEW view_name
```

Beispiele

```
SHOW CREATE VIEW orders_by_date
```

SHOW VIEWS

Listet die Namen aller Ansichten in der Datenbank auf. Alle Data-Catalog-Ansichten in der Datenbank, die über den Athena-Engine-SQL-Dialekt verfügen, werden aufgelistet. Andere Data-Catalog-Ansichten, die nicht über den Athena-Engine-Dialekt verfügen, werden herausgefiltert.

Syntax

```
SHOW VIEWS [IN database_name] [LIKE 'regular_expression']
```

Beispiele

```
SHOW VIEWS
```

```
SHOW VIEWS IN marketing_analytics LIKE 'orders*'
```

Verwenden von gespeicherten Abfragen

Sie können die Athena-Konsole verwenden, um die im Abfrage-Editor erstellten Abfragen zu speichern, zu bearbeiten, auszuführen, umzubenennen und zu löschen.

Überlegungen und Einschränkungen

- Sie können den Namen, die Beschreibung und den Abfragetext gespeicherter Abfragen aktualisieren.
- Sie können die Abfragen nur in Ihrem eigenen Konto aktualisieren.
- Sie können die Arbeitsgruppe oder Datenbank, zu der die Abfrage gehört, nicht ändern.
- Athena speichert keinen Verlauf der Abfrageänderungen. Wenn Sie eine bestimmte Version einer Abfrage behalten möchten, speichern Sie sie unter einem anderen Namen.

Arbeiten mit gespeicherten Abfragen in der Athena-Konsole

So speichern Sie eine Abfrage und geben ihr einen Namen

1. Geben Sie im Athena-Abfrage-Editor eine Abfrage ein oder führen Sie eine Abfrage aus.
2. Wählen Sie über dem Fenster des Abfrage-Editors auf der Registerkarte für die Abfrage die drei vertikalen Punkte aus, und wählen Sie dann Save as (Speichern unter) aus.
3. Geben Sie im Dialogfeld Save query (Abfrage speichern) einen Namen für die Abfrage und eine optionale Beschreibung ein. Sie können das erweiterbare Fenster Vorschau der SQL-Abfrage verwenden, um den Inhalt der Abfrage zu überprüfen, bevor Sie sie speichern.
4. Wählen Sie Save query (Abfrage speichern).

   Im Abfrage-Editor zeigt die Registerkarte für die Abfrage den von Ihnen angegebenen Namen an.

So führen Sie eine gespeicherte Abfrage aus

1. Wählen Sie in der Athena-Konsole die Registerkarte Saved queries (Gespeicherte Abfragen) aus.

2. Wählen Sie in der Liste Saved queries (Gespeicherte Abfragen) die ID der Abfrage aus, die Sie ausführen möchten.

   Der Abfrage-Editor zeigt die von Ihnen gewählte Abfrage an.
3. Wählen Sie Run (Ausführen) aus.

So bearbeiten Sie eine gespeicherte Abfrage

1. Wählen Sie in der Athena-Konsole die Registerkarte Saved queries (Gespeicherte Abfragen) aus.
2. Wählen Sie in der Liste Saved queries (Gespeicherte Abfragen) die ID der Abfrage aus, die Sie bearbeiten möchten.
3. Bearbeiten Sie die Abfrage im Abfrage-Editor.
4. Führen Sie einen der folgenden Schritte aus:
   - Wählen Sie dann Run (Ausführen) aus, um die Abfrage auszuführen.
   - Um die Abfrage zu speichern, wählen Sie die drei vertikalen Punkte auf der Registerkarte für die Abfrage aus, und wählen Sie dann Save (Speichern) aus.
   - Um die Abfrage unter einem anderen Namen zu speichern, wählen Sie die drei vertikalen Punkte auf der Registerkarte für die Abfrage aus, und wählen Sie dann Save as (Speichern unter) aus.

So benennen Sie eine bereits im Abfrage-Editor angezeigte gespeicherte Abfrage um oder löschen sie

1. Wählen Sie die drei vertikalen Punkte auf der Registerkarte für die Abfrage aus, und wählen Sie dann Rename (Umbenennen) oder Delete (Löschen) aus.
2. Folgen Sie den Eingabeaufforderungen, um die Abfrage umzubenennen oder zu löschen.

So benennen Sie eine gespeicherte Abfrage um, die nicht im Abfrage-Editor angezeigt wird

1. Wählen Sie in der Athena-Konsole die Registerkarte Saved queries (Gespeicherte Abfragen) aus.
2. Aktivieren Sie das Kontrollkästchen für die Abfrage, die Sie umbenennen möchten.
3. Wählen Sie Rename (Umbenennen) aus.

4. Bearbeiten Sie im Dialogfeld Rename query (Abfrage umbenennen) den Abfragenamen und die Abfragebeschreibung. Sie können das erweiterbare Fenster Vorschau der SQL-Abfrage verwenden, um den Inhalt der Abfrage zu überprüfen, bevor Sie sie umbenennen.
5. Klicken Sie auf Rename query (Abfrage umbenennen).

   Die umbenannte Abfrage wird in der Liste Saved queries (Gespeicherte Abfragen) angezeigt.

So löschen Sie eine gespeicherte Abfrage, die nicht im Abfrage-Editor angezeigt wird

1. Wählen Sie in der Athena-Konsole die Registerkarte Saved queries (Gespeicherte Abfragen) aus.
2. Aktivieren Sie ein oder mehrere Kontrollkästchen der Abfragen, die Sie löschen möchten.
3. Wählen Sie Delete (Löschen).
4. Wählen Sie bei der Bestätigungsaufforderung Delete (Löschen) aus.

   Eine oder mehr Abfragen wird/werden aus der Liste Saved queries (Gespeicherte Abfragen) entfernt.

Verwenden der Athena-API zum Aktualisieren gespeicherter Abfragen

Weitere Informationen zur Verwendung der Athena-API zum Aktualisieren einer gespeicherten Abfrage finden Sie in der UpdateNamedQuery-Aktion in der Athena-API-Referenz.

Verwenden von parametrisierten Abfragen

Sie können parametrisierte Athena-Abfragen verwenden, um dieselbe Abfrage mit unterschiedlichen Parameterwerten zur Ausführungszeit erneut auszuführen und SQL-Injection-Angriffe zu verhindern. In Athena können parametrisierte Abfragen in beliebigen DML-Abfragen oder vorbereiteten SQL-Anweisungen die Form von Ausführungsparametern annehmen.

- Abfragen mit Ausführungsparametern können in einem einzigen Schritt durchgeführt werden und sind nicht arbeitsgruppenspezifisch. Sie platzieren Fragezeichen in jeder DML-Abfrage für die Werte, die Sie parametrisieren möchten. Wenn Sie die Abfrage ausführen, deklarieren Sie die Werte der Ausführungsparameter sequenziell. Die Deklaration von Parametern und die Zuweisung von Werten für die Parameter können in derselben Abfrage erfolgen, jedoch entkoppelt. Im Gegensatz zu vorbereiteten Anweisungen können Sie die Arbeitsgruppe auswählen, wenn Sie eine Abfrage mit Ausführungsparametern senden.

- Vorbereitete Anweisungen erfordern zwei separate SQL-Anweisungen: `PREPARE` und `EXECUTE`. Zunächst definieren Sie die Parameter in der `PREPARE`-Anweisung. Dann führen Sie eine `EXECUTE`-Anweisung aus, die Werte für die von Ihnen definierten Parameter bereitstellt. Vorbereitete Anweisungen sind arbeitsgruppenspezifisch. Sie können sie nicht außerhalb des Kontextes der Arbeitsgruppe, zu der sie gehören, ausführen.

Überlegungen und Einschränkungen

- Parametrisierte Abfragen werden in Athena-Engine-Version 2 und höheren Versionen unterstützt. Weitere Informationen über Athena-Engine-Versionen finden Sie unter Athena-Engine-Versionierung.
- Derzeit werden parametrisierte Abfragen nur für `SELECT`-, `INSERT INTO`-, `CTAS`- und `UNLOAD`-Anweisungen.
- In parametrisierten Abfragen sind Parameter positionell und werden mit `?` angegeben. Parametern werden Werte nach ihrer Reihenfolge in der Abfrage zugewiesen. Benannte Parameter werden nicht unterstützt.
- Derzeit können `?`-Parameter nur in der `WHERE`-Klausel platziert werden. Syntax wie `SELECT ? FROM table` wird nicht unterstützt.
- Fragezeichen-Parameter können nicht in doppelte oder einfache Anführungszeichen gesetzt werden (d. h. `'?'` und `"?"` sind keine gültige Syntax).
- Damit SQL-Ausführungsparameter als Zeichenfolgen behandelt werden, müssen sie in einfache Anführungszeichen und nicht in doppelte Anführungszeichen eingeschlossen werden.
- Bei Bedarf können Sie die `CAST`-Funktion verwenden, wenn Sie einen Wert für einen parametrisierten Begriff eingeben. Wenn Sie beispielsweise über eine Spalte des `date`-Typs verfügen, den Sie in einer Abfrage parametrisiert haben, und Sie das Datum `2014-07-05` abfragen möchten, wird `CAST('2014-07-05' AS DATE)` bei Eingabe des Parameterwerts das Ergebnis zurückgegeben.
- Vorbereitete Anweisungen sind arbeitsgruppenspezifisch und die Namen vorbereiteter Anweisungen müssen innerhalb der Arbeitsgruppe eindeutig sein.
- IAM-Berechtigungen für vorbereitete Anweisungen sind erforderlich. Weitere Informationen finden Sie unter Zugriff auf vorbereitete Anweisungen zulassen.
- Abfragen mit Ausführungsparametern in der Athena-Konsole sind auf maximal 25 Fragezeichen beschränkt.

Abfragen mithilfe von Ausführungsparametern

Sie können Fragezeichen-Platzhalter in jeder DML-Abfrage verwenden, um eine parametrisierte Abfrage zu erstellen, ohne zuerst eine vorbereitete Anweisung zu erstellen. Um diese Abfragen auszuführen, können Sie die Athena-Konsole oder die AWS CLI oder das AWS SDK verwenden und die Variablen im `execution-parameters`-Argument deklarieren.

Ausführen von Abfragen mit Ausführungsparametern in der Athena-Konsole

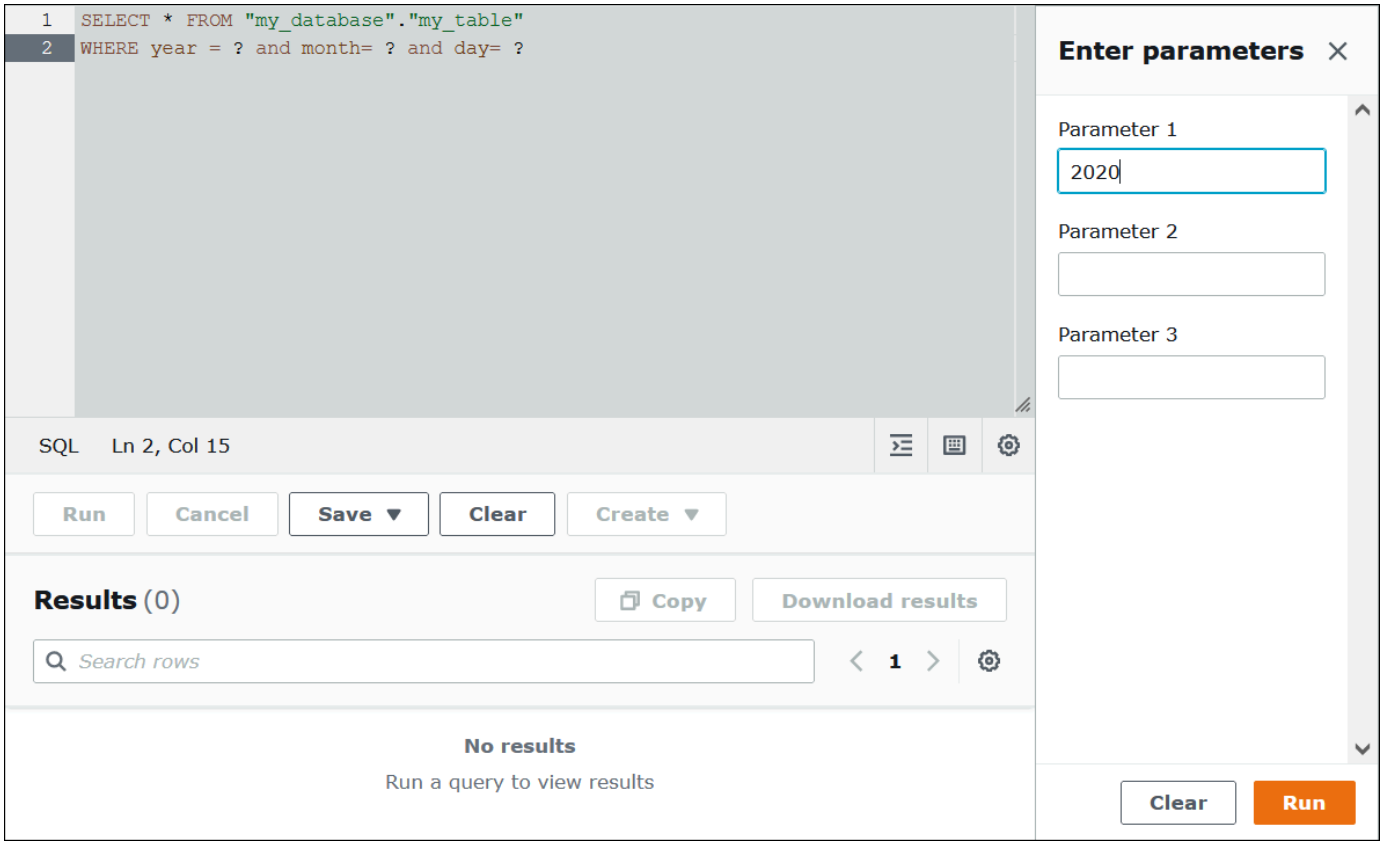
Wenn Sie eine parametrisierte Abfrage ausführen, die Ausführungsparameter (Fragezeichen) in der Athena-Konsole enthält, werden Sie aufgefordert, die Werte in der Reihenfolge einzugeben, in der die Fragezeichen in der Abfrage auftreten.

So führen Sie eine Abfrage mit Ausführungsparametern aus

1. Geben Sie im Athena-Editor eine Abfrage mit Fragezeichen-Platzhaltern ein, wie im folgenden Beispiel gezeigt.

   ```
   SELECT * FROM "my_database"."my_table"
   WHERE year = ? and month= ? and day= ?
   ```

2. Wählen Sie Run (Ausführen) aus.
3. In dem Dialogfeld Enter parameters (Parameter eingeben) geben Sie für jedes der Fragezeichen in der Abfrage einen Wert in der richtigen Reihenfolge ein.



4. Wenn Sie alle Parameter eingegeben haben, wählen Sie Run (Ausführen) aus. Der Editor zeigt die Abfrageergebnisse für die von Ihnen eingegebenen Parameterwerte an.

Jetzt können Sie einen der folgenden Schritte ausführen:

- Geben Sie verschiedene Parameterwerte für dieselbe Abfrage ein und wählen Sie anschließend Run again (Erneut ausführen) aus.
- Um alle Werte, die Sie eingegeben haben, gleichzeitig zu löschen, wählen Sie Clear (Löschen) aus.
- Um die Abfrage direkt zu bearbeiten (z. B. um Fragezeichen hinzuzufügen oder zu entfernen), schließen Sie zunächst das Dialogfeld Enter parameters (Parameter eingeben).
- Um die parametrisierte Abfrage für eine spätere Verwendung zu speichern, wählen Sie Save (Speichern) oder Save as (Speichern als) aus und geben Sie der Abfrage dann einen Namen. Weitere Informationen zur Verwendung von gespeicherten Abfragen finden Sie unter Verwenden von gespeicherten Abfragen.

Als Annehmlichkeit merkt sich das Dialogfeld Enter parameters (Parameter eingeben) die Werte, die Sie zuvor für die Abfrage eingegeben haben, solange Sie dieselbe Registerkarte im Abfrage-Editor verwenden.

Ausführen von Abfragen mit Ausführungsparametern in der AWS CLI

Um Abfragen mit Ausführungsparametern mithilfe der AWS CLI auszuführen, verwenden Sie den `start-query-execution`-Befehl und stellen Sie eine parametrisierte Abfrage im `query-string`-Argument bereit. Geben Sie daraufhin im `execution-parameters`-Argument die Werte für die Ausführungsparameter an. Das folgende Beispiel illustriert diese Technik.

```
aws athena start-query-execution
--query-string "SELECT * FROM table WHERE x = ? AND y = ?"
--query-execution-context "Database"="default"
--result-configuration "OutputLocation"="s3://..."
--execution-parameters "1" "2"
```

Abfragen mit vorbereiteten Anweisungen

Sie können eine vorbereitete Anweisung für die wiederholte Ausführung derselben Abfrage mit unterschiedlichen Abfrageparametern verwenden. Eine vorbereitete Anweisung enthält Parameterplatzhalter, deren Werte zur Ausführungszeit angegeben werden.

> **Note**
>
> Die maximale Anzahl vorbereiteter Anweisungen in einer Arbeitsgruppe beträgt 1 000.

SQL-Anweisungen

Sie können die SQL-Anweisungen `PREPARE`, `EXECUTE` und `DEALLOCATE PREPARE` verwenden, um parametrisierte Abfragen im Abfrage-Editor der Athena-Konsole auszuführen.

- Um Parameter anzugeben, bei denen Sie normalerweise Literalwerte verwenden würden, verwenden Sie Fragezeichen in der `PREPARE`-Anweisung.
- Um die Parameter beim Ausführen der Abfrage durch Werte zu ersetzen, verwenden Sie die `USING`-Klausel in der `EXECUTE`-Anweisung.
- Um eine vorbereitete Anweisung aus den vorbereiteten Anweisungen in einer Arbeitsgruppe zu entfernen, verwenden Sie die `DEALLOCATE PREPARE`-Anweisung.

Die folgenden Abschnitte enthalten zusätzliche Details zu jeder dieser Aussagen.

PREPARE

Bereitet eine Anweisung vor, die zu einem späteren Zeitpunkt ausgeführt werden soll. Vorbereitete Anweisungen werden in der aktuellen Arbeitsgruppe mit dem von Ihnen angegebenen Namen gespeichert. Die Anweisung kann Parameter anstelle von Literalen enthalten, die beim Ausführen der Abfrage ersetzt werden sollen. Parameter, die durch Werte ersetzt werden sollen, werden durch Fragezeichen gekennzeichnet.

Syntax

```
PREPARE statement_name FROM statement
```

In der Tabelle unten werden diese Parameter beschrieben.

Parameter	Beschreibung
statement_name	Der Name der zu erstellenden Anweisung. Der Name muss innerhalb der Arbeitsgruppe eindeutig sein.
statement	SELECT-, CTAS- oder INSERT INTO-Abfrage.

PREPARE-Beispiele

Die folgenden Beispiele zeigen die Verwendung der PREPARE-Anweisung. Fragezeichen kennzeichnen die Werte, die von der EXECUTE-Anweisung beim Ausführen der Abfrage geliefert werden sollen.

```
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://my_output_bucket/'
WITH (format='PARQUET')
```

EXECUTE

Führt eine vorbereitete Anweisung aus. Werte für Parameter werden in der `USING`-Klausel angegeben.

Syntax

```
EXECUTE statement_name [USING value1 [ ,value2, ... ] ]
```

`statement_name` ist der Name der vorbereiteten Anweisung. *`value1`* und *`value2`* sind die Werte, die für die Parameter in der Anweisung anzugeben sind.

EXECUTE-Beispiele

Im folgenden Beispiel wird die vorbereitete Anweisung `my_select1` ausgeführt, die keine Parameter enthält.

```
EXECUTE my_select1
```

Im folgenden Beispiel wird die vorbereitete Anweisung `my_select2` ausgeführt, die einen einzelnen Parameter enthält.

```
EXECUTE my_select2 USING 2012
```

Im folgenden Beispiel wird die vorbereitete Anweisung `my_select3` ausgeführt, die über zwei Parameter verfügt.

```
EXECUTE my_select3 USING 346078, 12
```

Das folgende Beispiel liefert einen Zeichenfolgenwert für einen Parameter in der vorbereiteten Anweisung `my_insert`.

```
EXECUTE my_insert USING 'usa'
```

Das folgende Beispiel liefert einen Zahlenwert für den `productid`-Parameter in der vorbereiteten Anweisung `my_unload`.

```
EXECUTE my_unload USING 12
```

DEALLOCATE PREPARE

Entfernt die vorbereitete Anweisung mit dem angegebenen Namen aus der Liste der vorbereiteten Anweisungen in der aktuellen Arbeitsgruppe.

Syntax

```
DEALLOCATE PREPARE statement_name
```

`statement_name` ist der Name der vorbereiteten Anweisung, die entfernt werden soll.

Beispiel

Im folgenden Beispiel wird die vorbereitete Anweisung `my_select1` aus der aktuellen Arbeitsgruppe entfernt.

```
DEALLOCATE PREPARE my_select1
```

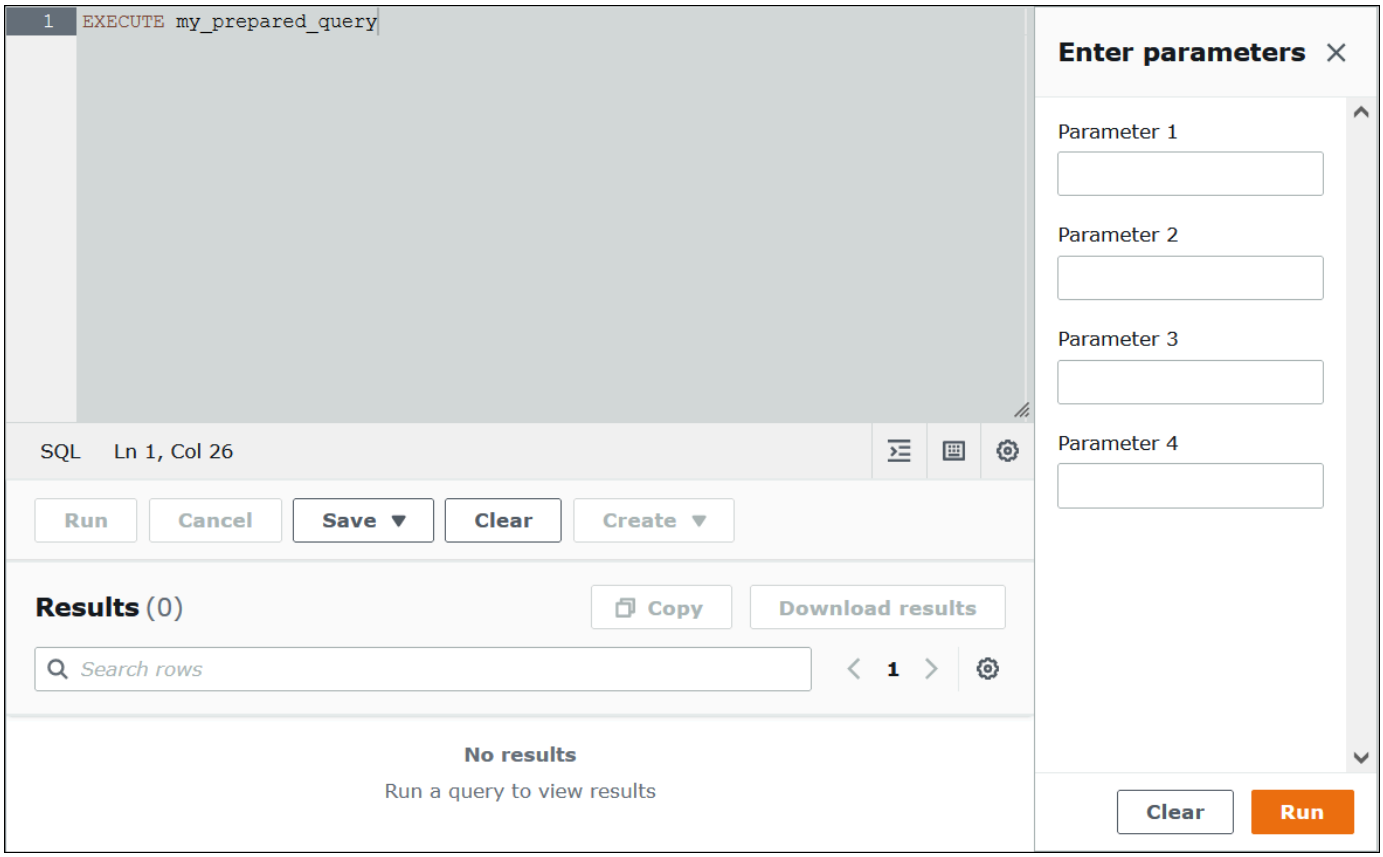
Vorbereitete Anweisungen ohne die USING-Klausel in der Athena-Konsole ausführen

Wenn Sie eine vorhandene vorbereitete Anweisung mit der Syntax `EXECUTE` *`prepared_statement`* im Abfrage-Editor ausführen, öffnet Athena das Dialogfeld Enter parameters (Parameter eingeben), sodass Sie die Werte eingeben können, die normalerweise in die `USING`-Klausel der `EXECUTE ... USING`-Anweisung eingegeben werden.

So führen Sie eine vorbereitete Anweisung mit dem Dialogfeld Enter parameters (Parameter eingeben) aus

1. Im Abfrage-Editor verwenden Sie anstelle der Syntax `EXECUTE prepared_statement USING` *`value1`*`,` *`Value2`* `...` die Syntax `EXECUTE` *`prepared_statement`*.

2. Wählen Sie Run (Ausführen) aus. Das Dialogfeld Enter parameters (Parameter eingeben) wird angezeigt.



3. Geben Sie die Werte der Reihe nach in das Dialogfeld Execution parameters (Ausführungsparameter) ein. Da der Originaltext der Abfrage nicht sichtbar ist, müssen Sie sich an die Bedeutung der einzelnen Positionsparameter erinnern oder die vorbereitete Anweisung als Referenz zur Verfügung haben.
4. Wählen Sie Run (Ausführen) aus.

Erstellen von vorbereiteten Anweisungen mit der AWS CLI

Um eine vorbereitete Anweisung mit der AWS CLI zu erstellen, können Sie einen der folgenden `athena`-Befehle verwenden:

- Verwenden Sie den `create-prepared-statement`-Befehl und geben Sie eine Abfrageanweisung mit Ausführungsparametern an.
- Verwenden Sie den `start-query-execution`-Befehl und geben Sie eine Abfragezeichenfolge an, die die PREPARE-Syntax verwendet.

Verwenden von create-prepared-statement

Definieren Sie in einem `create-prepared-statement`-Befehl den Abfragetext im `query-statement`-Argument, wie im folgenden Beispiel.

```
aws athena create-prepared-statement
--statement-name PreparedStatement1
--query-statement "SELECT * FROM table WHERE x = ?"
--work-group athena-engine-v2
```

Verwenden von start-query-execution und der PREPARE-Syntax

Verwenden Sie den `start-query-execution`-Befehl. Setzen Sie die `PREPARE`-Anweisung in das `query-string`-Argument, wie im folgenden Beispiel gezeigt:

```
aws athena start-query-execution
--query-string "PREPARE PreparedStatement1 FROM SELECT * FROM table WHERE x = ?"
--query-execution-context '{"Database": "default"}'
--result-configuration '{"OutputLocation": "s3://..."}'
```

Ausführen von vorbereiteten Anweisungen mit der AWS CLI

Um eine vorbereitete Anweisung mit der AWS CLI auszuführen, können Sie mithilfe einer der folgenden Methoden Werte für die Parameter bereitstellen:

- Verwenden Sie das `execution-parameters`-Argument.
- Verwenden Sie die `EXECUTE ... USING`-SQL-Syntax im `query-string`-Argument.

Verwenden des Arguments execution-parameters

Bei diesem Ansatz verwenden Sie den `start-query-execution`-Befehl und geben den Namen einer vorhandenen vorbereiteten Anweisung im `query-string`-Argument ein. Geben Sie daraufhin im `execution-parameters`-Argument die Werte für die Ausführungsparameter an. Die folgende Beispielrichtlinie zeigt diese Methode.

```
aws athena start-query-execution
--query-string "Execute PreparedStatement1"
--query-execution-context "Database"="default"
--result-configuration "OutputLocation"="s3://..."
```

```
--execution-parameters "1" "2"
```

Verwenden von EXECUTE ... VERWENDEN der SQL-Syntax

Um eine vorhandene vorbereitete Anweisung mithilfe der `EXECUTE ... USING`-Syntax auszuführen, verwenden Sie den `start-query-execution`-Befehl und platzieren sowohl den Namen der vorbereiteten Anweisung als auch die Parameterwerte in das `query-string`-Argument, wie im folgenden Beispiel:

```
aws athena start-query-execution
--query-string "EXECUTE PreparedStatement1 USING 1"
--query-execution-context '{"Database": "default"}'
--result-configuration '{"OutputLocation": "s3://..."}'
```

Vorbereitete Anweisungen auflisten

Um die vorbereiteten Anweisungen für eine bestimmte Arbeitsgruppe aufzulisten, können Sie den AWS CLI-Befehl list-prepared-statements von Athena oder die API-Aktion ListPreparedStatements von Athena verwenden. Der Parameter `--work-group` muss angegeben werden.

```
aws athena list-prepared-statements --work-group primary
```

Weitere Informationen finden Sie auch unter

Lesen Sie dazu die folgenden verwandten Beiträge im AWS Big Data Blog.

- Die Wiederverwendbarkeit und Sicherheit mithilfe von in Amazon Athena parametrisierten Abfragen verbessern
- In Amazon Athena parametrisierte Abfragen verwenden, um Daten als Service bereitzustellen

Verwenden des kostenbasierten Optimierers

Sie können das CBO-Feature (Kostenbasierter Optimierer) in Athena SQL verwenden, um Ihre Abfragen zu optimieren. Sie können optional anfordern, dass Athena Statistiken auf Tabellen- oder Spaltenebene für eine Ihrer Tabellen in AWS Glue sammelt. Wenn alle Tabellen in Ihrer Abfrage Statistiken enthalten, verwendet Athena die Statistiken, um einen Ausführungsplan zu erstellen, den es für den leistungsfähigsten hält. Der Abfrageoptimierer berechnet alternative Pläne auf der Grundlage eines statistischen Modells und wählt dann den Plan aus, mit dem die Abfrage wahrscheinlich am schnellsten ausgeführt werden kann.

Statistiken zu AWS Glue Tabellen werden im gesammelt und gespeichert und Athena zur Verfügung gestellt, um die AWS Glue Data Catalog Planung und Ausführung von Abfragen zu verbessern. Bei diesen Statistiken handelt es sich um Statistiken auf Spaltenebene, z. B. die Anzahl der eindeutigen Werte, die Anzahl der Nullwerte, Maximal- und Minimalwerte für Dateitypen wie Parquet, ORC, JSON, ION, CSV und XML. Amazon Athena verwendet diese Statistiken, um Abfragen zu optimieren, indem die restriktivsten Filter so früh wie möglich bei der Abfrageverarbeitung angewendet werden. Diese Filterung begrenzt die Speichernutzung und die Anzahl der Datensätze, die gelesen werden müssen, um die Abfrageergebnisse zu liefern.

In Verbindung mit CBO verwendet Athena ein Feature, das als regelbasierter Optimierer (RBO) bezeichnet wird. RBO wendet mechanisch Regeln an, von denen erwartet wird, dass sie die Abfrageleistung verbessern. RBO ist im Allgemeinen vorteilhaft, da seine Transformationen darauf abzielen, den Abfrageplan zu vereinfachen. Da RBO jedoch keine Kostenberechnungen oder Planvergleiche durchführt, erschweren kompliziertere Abfragen es RBO, einen optimalen Plan zu erstellen.

Aus diesem Grund verwendet Athena sowohl RBO als auch CBO, um Ihre Abfragen zu optimieren. Nachdem Athena Möglichkeiten zur Verbesserung der Abfrageausführung identifiziert hat, erstellt es einen optimalen Plan. Informationen zu Ausführungsplandetails finden Sie unter Anzeigen von Ausführungsplänen für SQL-Abfragen. Eine ausführliche Erläuterung der Funktionsweise von CBO finden Sie im Blogbeitrag zur kostenbasierten Optimierung von AWS Big Data.

Um Statistiken für AWS Glue Katalogtabellen zu generieren, können Sie die Athena-Konsole, die AWS Glue Konsole oder AWS Glue APIs verwenden. Da Athena in AWS Glue Catalog integriert ist, erhalten Sie automatisch die entsprechenden Verbesserungen der Abfrageleistung, wenn Sie Abfragen von Amazon Athena ausführen.

Überlegungen und Einschränkungen

- Tabellentypen – Derzeit unterstützt das CBO-Feature in Athena nur Hive-Tabellen, die sich in AWS Glue Data Catalog befinden.
- Athena for Spark – Das CBO-Feature ist in Athena for Spark nicht verfügbar.
- Preise – Informationen zur Preisgestaltung finden Sie auf der AWS Glue -Preisseite.

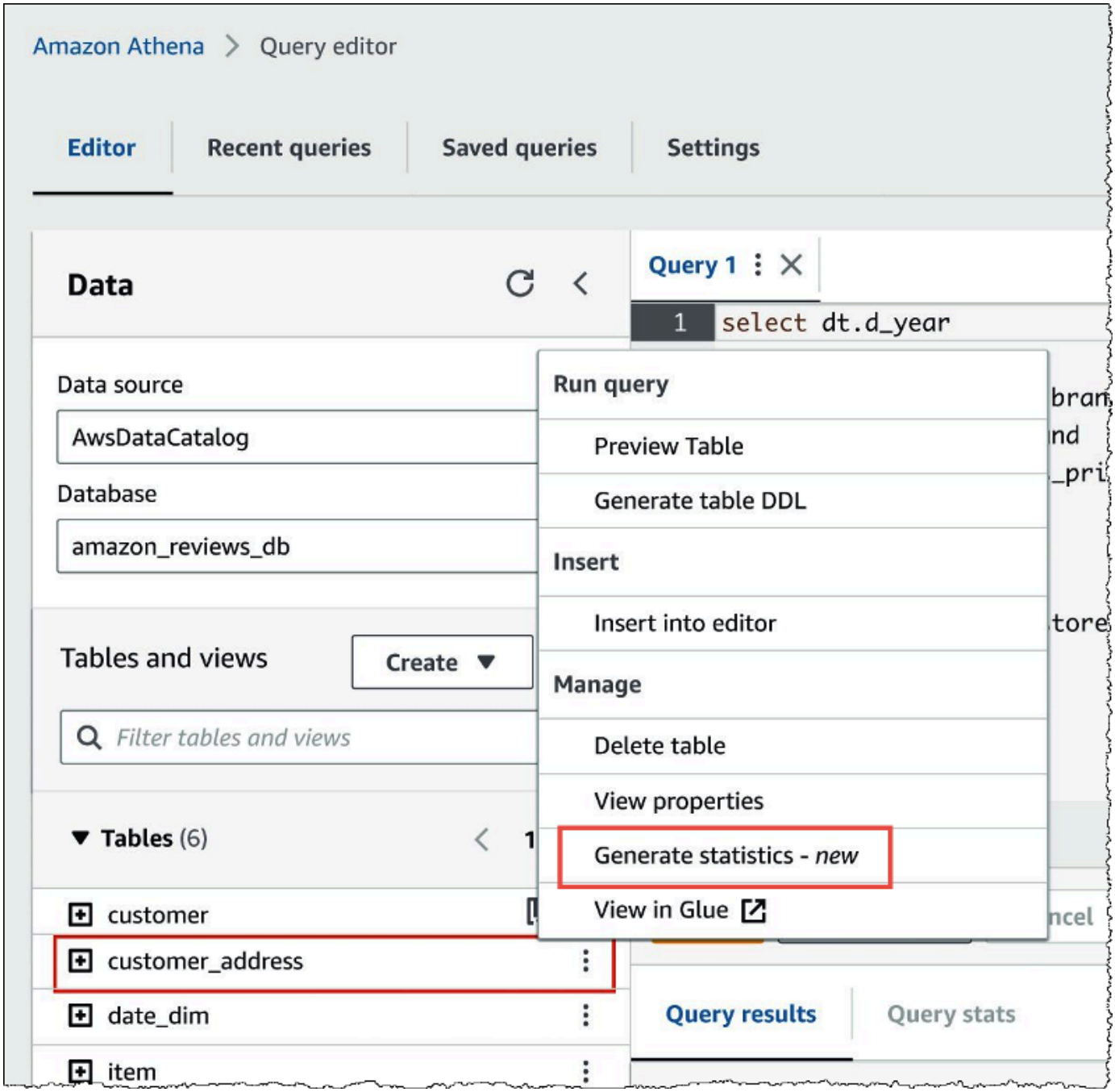
Generieren von Tabellenstatistiken mit der Athena-Konsole

In diesem Abschnitt wird beschrieben, wie Sie mit der Athena-Konsole Statistiken auf Tabellen- oder Spaltenebene für eine Tabelle in AWS Glue generieren. Informationen AWS Glue zur

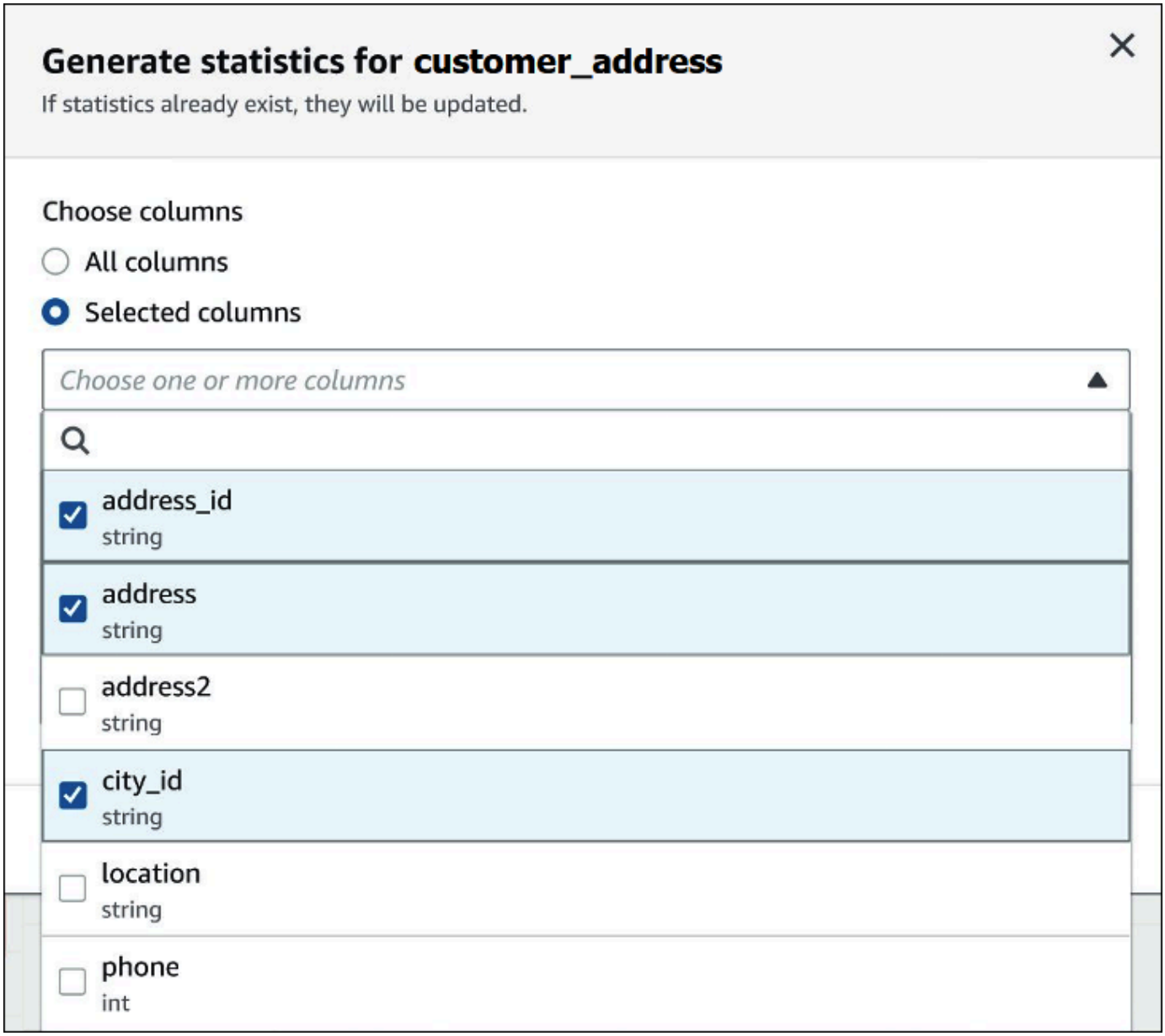
Generierung von Tabellenstatistiken finden Sie unter Arbeiten mit Spaltenstatistiken im AWS Glue Entwicklerhandbuch.

So generieren Sie Tabellenstatistiken mit der Athena-Konsole

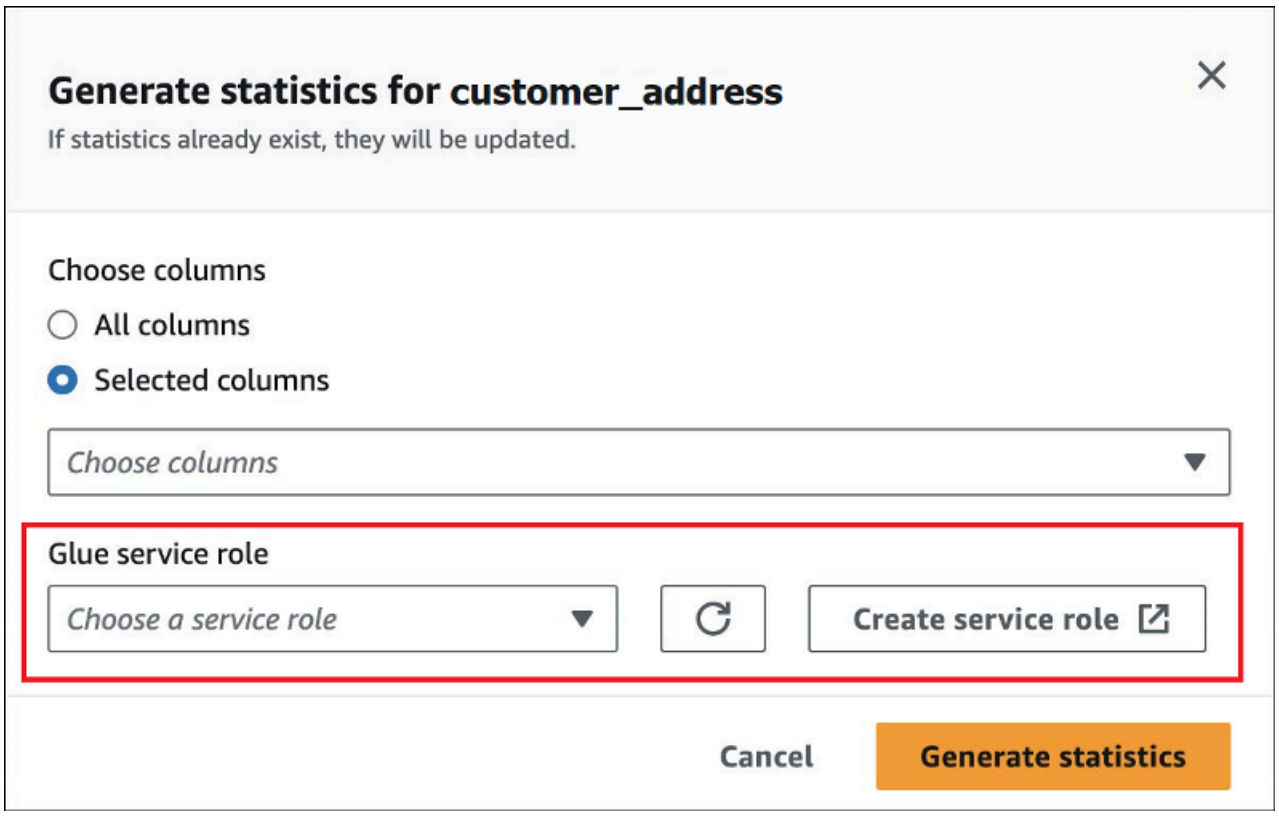
1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wählen Sie in der Tabellenliste des Athena-Abfrageeditors die drei vertikalen Punkte für die gewünschte Tabelle aus, und wählen Sie dann Statistik generieren aus.



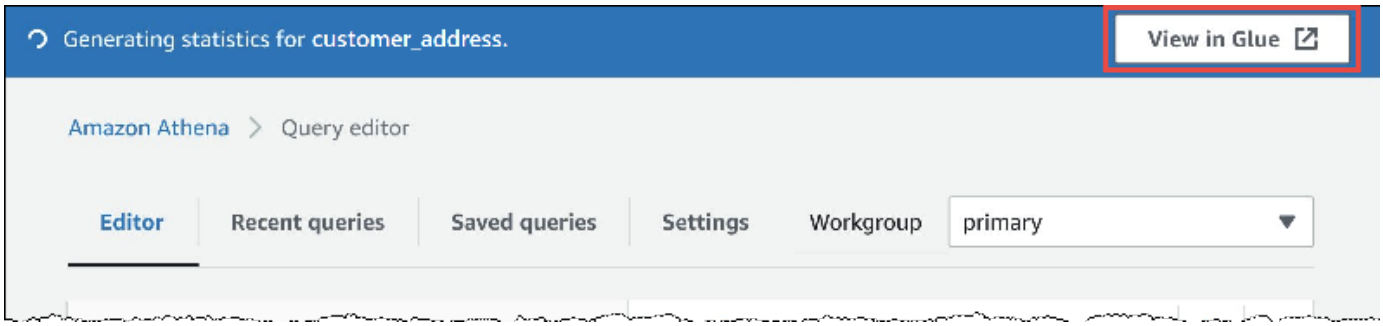
3. Wählen Sie im Dialogfeld Statistik generieren die Option Alle Spalten, um Statistiken für alle Spalten in der Tabelle zu generieren, oder wählen Sie Ausgewählte Spalten, um bestimmte Spalten auszuwählen. Der Standardwert ist Alle Spalten.



4. Erstellen Sie für die AWS Glue Servicerolle eine vorhandene Servicerolle oder wählen Sie eine aus, um die Erlaubnis zum Generieren von Statistiken AWS Glue zu erteilen. Die AWS Glue -Servicerolle erfordert außerdem `S3:GetObject`-Berechtigungen für den Amazon-S3-Bucket, der die Daten der Tabelle enthält.



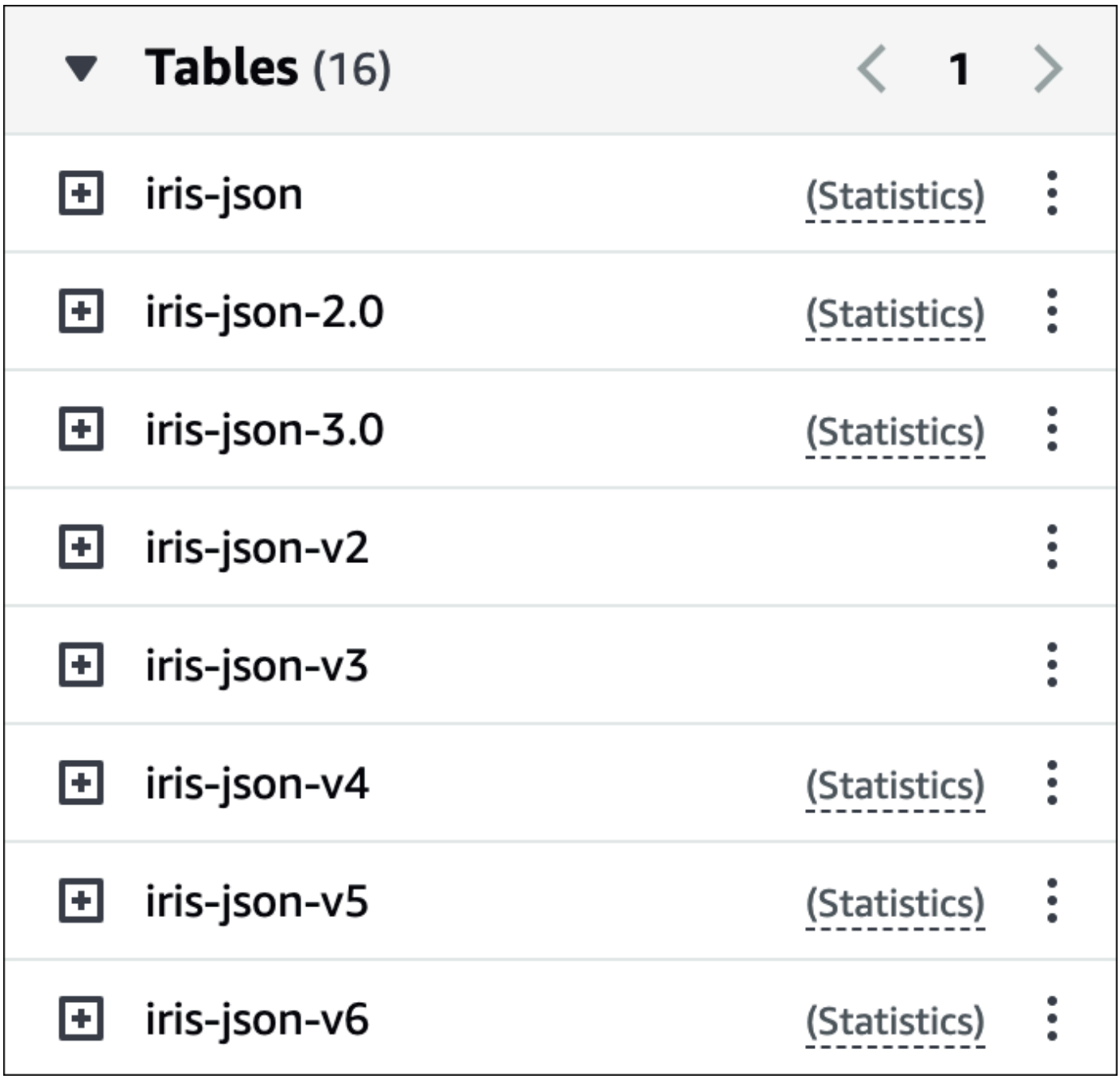
5. Wählen Sie Statistik generieren. In der Benachrichtigung Statistiken für ***Tabellenname*** werden generiert wird der Status der Aufgabe angezeigt.



6. Um Details in der AWS Glue Konsole anzuzeigen, wählen Sie In Glue anzeigen.

   Informationen zum Anzeigen von Statistiken in der AWS Glue Konsole finden Sie unter Spaltenstatistiken anzeigen im AWS Glue Entwicklerhandbuch.

7. Nach der Generierung der Statistiken wird in den Tabellen und Spalten mit Statistiken das Wort Statistik in Klammern angezeigt, wie in der folgenden Abbildung.



Wenn Sie jetzt Ihre Abfragen ausführen, führt Athena eine kostenbasierte Optimierung der Tabellen und Spalten durch, für die Statistiken generiert wurden.

Weitere Informationen finden Sie auch unter

Weitere Informationen finden Sie in der folgenden Ressource.

Abfragen von Daten der S3 Express One Zone

Die Speicherklasse der Amazon S3 Express One Zone ist eine hochleistungsfähige Amazon-S3-Speicherklasse, die Reaktionszeiten im einstelligen Millisekundenbereich bietet. Daher ist dies für Anwendungen nützlich, die häufig mit Hunderttausenden Anfragen pro Sekunde auf Daten zugreifen.

S3 Express One Zone repliziert und speichert Daten innerhalb derselben Availability Zone, um Geschwindigkeit und Kosten zu optimieren. Dies unterscheidet sich von den regionalen Speicherklassen von Amazon S3, die Daten automatisch über mindestens drei AWS Availability Zones innerhalb einer AWS-Region replizieren.

Weitere Informationen finden Sie unter Was ist S3 Express One Zone? im Amazon-S3-Benutzerhandbuch.

Voraussetzungen

Stellen Sie sicher, dass die folgenden Bedingungen erfüllt sind, bevor Sie beginnen:

- Athena-Engine-Version 3 – Um S3 Express One Zone mit Athena SQL zu verwenden, muss Ihre Arbeitsgruppe für die Verwendung der Athena-Engine-Version 3, konfiguriert sein.
- Berechtigungen für S3 Express One Zone – Wenn S3 Express One Zone eine Aktion wie `GET`, `LIST` oder `PUT` für ein Amazon-S3-Objekt aufruft, ruft die Speicherklasse `CreateSession` in Ihrem Namen auf. Aus diesem Grund muss Ihre IAM-Richtlinie die `s3express:CreateSession`-Aktion zulassen, wodurch Athena den entsprechenden API-Vorgang aufrufen kann.

Überlegungen und Einschränkungen

Berücksichtigen Sie bei der Abfrage von S3 Express One Zone mit Athena die folgenden Punkte.

- Buckets der S3 Express One Zone unterstützen nur `SSE_S3`-Verschlüsselung. Athena-Abfrageergebnisse werden mithilfe der `SSE_S3`-Verschlüsselung geschrieben, unabhängig von der Option, die Sie in den Arbeitsgruppeneinstellungen zum Verschlüsseln der Abfrageergebnisse auswählen. Diese Einschränkung gilt für alle Szenarien, in denen Athena Daten in Buckets der S3 Express One Zone schreibt, einschließlich `CREATE TABLE AS`- (CTAS) und `INSERT INTO`-Anweisungen.
- Der AWS Glue-Crawler wird nicht für die Erstellung von Tabellen für Daten der S3 Express One Zone unterstützt.

- Die `MSCK REPAIR TABLE`-Anweisung wird nicht unterstützt. Als Problemumgehung verwenden Sie ALTER TABLE ADD PARTITION.
- Die folgenden Datei- und Tabellenformate werden nicht oder nur eingeschränkt unterstützt. Wenn Formate nicht aufgeführt sind, aber für Athena unterstützt werden (z. B. Parquet, ORC und JSON), werden sie auch für die Verwendung mit Speicher der S3 Express One Zone unterstützt.

Datei- oder Tabellenformat	Einschränkung
Apache Avro	Nicht unterstützt
CloudTrail-Protokolle	Nicht unterstützt
Apache Hudi	Nicht unterstützt
Amazon Ion	Nicht unterstützt
Logstash-Protokolle	Nicht unterstützt
Apache WebServer-Protokolle	Nicht unterstützt
Delta Lake	DDL wird nicht unterstützt. Informationen zum Erstellen einer Delta-Lake-Tabelle mithilfe eines Dummy-Schemas finden Sie unter Synchronisieren von Delta-Lake-Metadaten. `SELECT`-Abfragen für die Tabelle werden unterstützt.

Erste Schritte

Das Abfragen von Daten der S3 Express One Zone mit Athena ist unkompliziert. Führen Sie zunächst die folgenden Schritte aus.

So verwenden Sie Athena SQL zum Abfragen von Daten der S3 Express One Zone

1. Übertragen Sie Ihre Daten auf Speicher der S3 Express One Zone. Weitere Informationen finden Sie unter Festlegen der Speicherklasse eines Objekts im Amazon-S3-Benutzerhandbuch.

2. Verwenden Sie eine CREATE TABLE-Anweisung in Athena, um Ihre Daten in AWS Glue Data Catalog zu katalogisieren. Informationen zum Erstellen von Tabellen in Athena finden Sie unter Erstellen von Tabellen in Athena und der CREATE TABLE-Anweisung.
3. (Optional) Konfigurieren Sie den Speicherort der Abfrageergebnisse Ihrer Athena-Arbeitsgruppe für die Verwendung eines Verzeichnis-Buckets von Amazon S3. Amazon-S3-Verzeichnis-Buckets sind leistungsfähiger als allgemeine Buckets und wurden für Workloads oder leistungskritische Anwendungen entwickelt, die eine konstante Latenzzeit im einstelligen Millisekundenbereich erfordern. Weitere Informationen finden Sie unter Übersicht über Verzeichnis-Buckets im Amazon-S3-Benutzerhandbuch.

Wiederhergestellte Amazon-S3-Glacier-Objekte abfragen

Sie können Athena verwenden, um wiederhergestellte Objekte aus den Amazon-S3-Speicherklassen S3 Glacier Flexible Retrieval (früher Glacier) und S3 Glacier Deep Archive abzufragen. Sie müssen diese Funktion für jede Tabelle aktivieren. Wenn Sie das Feature nicht für eine Tabelle aktivieren, bevor Sie eine Abfrage ausführen, überspringt Athena während der Abfrageausführung alle Objekte der Tabelle von S3 Glacier Flexible Retrieval und S3 Glacier Deep Archive.

Überlegungen und Einschränkungen

- Die Abfrage wiederhergestellter Amazon-S3-Glacier-Objekte wird nur auf der Athena-Engine-Version 3 unterstützt.
- Das Feature wird nur für Apache-Hive-Tabellen unterstützt.
- Sie müssen Ihre Objekte wiederherstellen, bevor Sie Ihre Daten abfragen; Athena stellt keine Objekte für Sie wieder her.

Konfiguration einer Tabelle für die Verwendung wiederhergestellter Objekte

Um Ihre Athena-Tabelle so zu konfigurieren, dass sie wiederhergestellte Objekte in Ihre Abfragen einbezieht, müssen Sie ihre `read_restored_glacier_objects`-Tabelleneigenschaft auf `true` setzen. Dazu können Sie den Athena-Abfrage-Editor oder die AWS Glue Konsole verwenden. Sie können aber auch die AWS Glue -CLI, die AWS Glue -API oder das AWS Glue -SDK. verwenden.

Arbeiten mit dem Athena-Abfrage-Editor

In Athena können Sie den ALTER TABLE SET TBLPROPERTIES Befehl verwenden, um die Tabelleneigenschaft festzulegen, wie im folgenden Beispiel.

```
ALTER TABLE table_name SET TBLPROPERTIES ('read_restored_glacier_objects' = 'true')
```

Verwenden der AWS Glue -Konsole

Führen Sie in der - AWS Glue Konsole die folgenden Schritte aus, um die `read_restored_glacier_objects` Tabelleneigenschaft hinzuzufügen.

So konfigurieren Sie Tabelleneigenschaften in der AWS Glue Konsole

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die - AWS Glue Konsole unter https://console.aws.amazon.com/glue/.
2. Führen Sie eine der folgenden Aktionen aus:
   - Wählen Sie Zu Data Catalog gehen.
   - Wählen Sie im Navigationsbereich die Option Data-Catalog-Tabellen.
3. Wählen Sie auf der Seite Tabellen in der Liste der Tabellen den Link zu der Tabelle, die Sie bearbeiten möchten.
4. Wählen Sie Actions (Aktionen) und Edit table (Tabelle bearbeiten) aus.
5. Fügen Sie auf der Seite Tabelle bearbeiten im Abschnitt Tabelleneigenschaften für jede partitionierte Spalte das folgende Schlüssel-Wert-Paar hinzu.
   - Fügen Sie für Key (Schlüssel) `read_restored_glacier_objects` hinzu.
   - Geben Sie für Wert `true` ein.
6. Wählen Sie Save aus.

Verwenden der AWS CLI

In der können Sie den Befehl AWS Glue update-table und sein `--table-input` Argument verwenden AWS CLI, um die Tabelle neu zu definieren und dabei die `-read_restored_glacier_objects`Eigenschaft hinzuzufügen. Verwenden Sie im `--table-input`-Argument die `Parameters`-Struktur, um die `read_restored_glacier_objects`-Eigenschaft und den Wert von `true` anzugeben. Beachten Sie, dass das Argument für `--table-input` keine Leerzeichen enthalten darf und umgekehrte Schrägstriche verwenden muss, um die doppelten Anführungszeichen zu maskieren. Ersetzen Sie im folgenden Beispiel *`my_database`* und *`my_table`* durch den Namen Ihrer Datenbank und Tabelle.

```
aws glue update-table \
```

```
    --database-name my_database \
    --table-input={\"Name\":\"my_table\",\"Parameters\":{\"read_restored_glacier_objects
\":\"true\"}}
```

> **Important**
>
> Der AWS Glue `update-table` Befehl funktioniert im Überschreibmodus, was bedeutet, dass er die vorhandene Tabellendefinition durch die neue Definition ersetzt, die durch den `table-input` Parameter angegeben wird. Stellen Sie daher sicher, dass Sie beim `table-input` Hinzufügen der `read_restored_glacier_objects` Eigenschaft auch alle Felder angeben, die sich in Ihrer Tabelle befinden sollen.

Verarbeiten von Schema-Updates

In diesem Abschnitt finden Sie Anleitungen zum Umgang mit Schema-Updates für verschiedene Datenformate. Athena ist eine Schema-on-Read-Abfrage-Engine. Dies bedeutet, dass beim Anlegen einer Tabelle in Athena Schemata beim Lesen der Daten angewendet werden. Dabei werden die zugrunde liegenden Daten nicht geändert oder überschrieben.

Wenn Sie Änderungen in Tabellenschemata vorweg nehmen, sollten Sie diese in einem für Ihre Bedürfnisse geeigneten Datenformat anlegen. Ihr Ziel ist es, vorhandene Athena-Abfragen bei sich weiterentwickelnden Schemata wiederzuverwenden und beim Abfragen von Tabellen mit Partitionen Fehler aufgrund von Schemakonflikten zu vermeiden.

Um diese Ziele zu erreichen, wählen Sie das Datenformat basierend auf der Tabelle im folgenden Thema.

Themen

Zusammenfassung: Updates und Datenformate in Athena

Die folgende Tabelle enthält eine Übersicht über die Datenspeicherungsformate und ihre unterstützten Schema-Manipulationen. Verwenden Sie diese Tabelle, um das Format zu wählen, mit

dem Sie weiterhin Athena-Abfragen verwenden können, auch wenn sich Ihre Schemata im Lauf der Zeit ändern.

Beachten Sie, dass Parquet und ORC in dieser Tabelle Spaltenformate mit unterschiedlichen Standard-Spaltenzugriffsmethoden sind. Standardmäßig greift Parquet dem Namen nach auf Spalten zu, ORC dem Index nach (Ordnungswert). Aus diesem Grund stellt Athena eine SerDe-Eigenschaft bereit, die definiert wird, wenn eine Tabelle erstellt wird, um die Standard-Spaltenzugriffsmethode umzustellen. Dies ermöglicht eine größere Flexibilität bei der Schema-Entwicklung.

Für Parquet kann die `parquet.column.index.access`-Eigenschaft auf `true` gesetzt werden, sodass die Spaltenzugriffsmethode die Ordinalzahl für die Spalte verwendet. Wenn Sie diese Eigenschaft auf `false` setzen, ändert sich die Spaltenzugriffsmethode auf die Verwendung des Spaltennamens. Analog dazu verwenden Sie für ORC die `orc.column.index.access` -Eigenschaft, um die Spaltenzugriffsmethode festzulegen. Weitere Informationen finden Sie unter Index-Zugriff in ORC und Parquet.

Mit CSV und TSV können Sie alle Schema-Manipulationen durchführen, mit Ausnahme der Neuanordnung von Spalten oder des Hinzufügens von Spalten am Anfang der Tabelle. Wenn Ihre Schemaentwicklung z. B. nur das Umbenennen von Spalten, nicht aber deren Entfernung erfordert, können Sie Ihre Tabellen in CSV oder TSV erstellen. Wenn Sie Spalten entfernen möchten, verwenden Sie nicht CSV oder TSV, sondern eines der anderen unterstützten Formate, vorzugsweise ein Spaltenformat wie Parquet oder ORC.

Schema-Updates und Datenformate in Athena

Erwarteter Typ des Schema-Updates	Übersicht	CSV (mit und ohne Header) und TSV	JSO	AVI	PARQUE Lesen nach Namen (Standard)	PARQUE Lesen nach Index	ORC: Lesen nach Index (Standard)	ORC: Lesen nach Namen
Spalten umbenennen	Speichern Sie Ihre Daten in CSV und TSV oder in ORC und Parquet, wenn sie	Y	N	N	N	Y	Y	N

Erwarteter Typ des Schema-Updates	Übersicht	CSV (mit und ohne Header) und TSV	JSO	AVF	PARQUE Lesen nach Namen (Standard)	PARQUE Lesen nach Index	ORC: Lesen nach Index (Standard)	ORC: Lesen nach Namen
	nach dem Index gelesen werden.							
Hinzufügen von Spalten am Anfang oder in der Mitte der Tabelle	Speichern Sie Ihre Daten in JSON, AVRO oder in Parquet und ORC, wenn sie nach dem Namen gelesen werden. Verwenden Sie nicht CSV und TSV.	N	Y	Y	Y	N	N	Y
Hinzufügen von Spalten am Ende der Tabelle	Speichern Sie Ihre Daten im CSV-, TSV-, JSON-, AVRO-, ORC- oder Parquet-Format.	Y	Y	Y	Y	Y	Y	Y

Erwarteter Typ des Schema-Updates	Übersicht	CSV (mit und ohne Header) und TSV	JSO	AVF	PARQUE Lesen nach Namen (Standard)	PARQUE Lesen nach Index	ORC: Lesen nach Index (Standard)	ORC: Lesen nach Namen
Spalten entfernen	Speichern Sie Ihre Daten in JSON, AVRO oder in Parquet und ORC, wenn sie nach dem Namen gelesen werden. Verwenden Sie nicht CSV und TSV.	N	Y	Y	Y	N	N	Y
Neuanordnen von Spalten	Speichern Sie Ihre Daten in AVRO, JSON oder ORC und Parquet, wenn sie nach dem Namen gelesen werden.	N	Y	Y	Y	N	N	Y

Erwarteter Typ des Schema-Updates	Übersicht	CSV (mit und ohne Header) und TSV	JSO	AVF	PARQUE Lesen nach Namen (Standard)	PARQUE Lesen nach Index	ORC: Lesen nach Index (Standard)	ORC: Lesen nach Namen
Ändern des Datentyps einer Spalte	Speichern Sie Ihre Daten in einem beliebigen Format, testen Sie Ihre Abfrage jedoch in Athena, um sicherzustellen, dass die Datentypen kompatibel sind. Für Parquet und ORC ist eine Änderung des Datentyps nur für partitionierte Tabellen möglich.	Y	Y	Y	Y	Y	Y	Y

Index-Zugriff in ORC und Parquet

PARQUET und ORC sind spaltenbasierte Dateiformaten, die nach Index gelesen werden können, oder nach Namen. Wenn Sie Ihre Daten in einem dieser Formate speichern, können Sie alle Operationen an Schemata durchführen und Athena-Abfragen ohne Schema-Fehler ausführen.

- Athena liest ORC standardmäßig nach dem Index, wie in `SERDEPROPERTIES ( 'orc.column.index.access'='true')` definiert. Weitere Informationen finden Sie unter ORC: Lesen nach Index.

- Athena liest Parquet standardmäßig nach dem Namen, wie in `SERDEPROPERTIES ( 'parquet.column.index.access'='false')` definiert. Weitere Informationen finden Sie unter Parquet: Lesen nach Namen.

Da dies die Standardoptionen sind, ist die Angabe dieser SerDe-Eigenschaften in Ihren `CREATE TABLE`-Abfragen optional, weil sie implizit verwendet werden. Wenn sie verwendet werden, gestatten sie Ihnen die Ausführung einiger Schema-Update-Operationen, während sie andere solcher Operationen verhindern. Um diese Operationen zu aktivieren, führen Sie eine weitere `CREATE TABLE`-Anfrage aus und ändern die SerDe-Einstellungen.

> **Note**
>
> Die SerDe-Eigenschaften werden nicht automatisch für jede Partition weitergegeben. Verwenden Sie `ALTER TABLE ADD PARTITION`-Anweisungen zum Festlegen der SerDe-Eigenschaften für jede Partition. Um diesen Prozess zu automatisieren, schreiben Sie ein Skript, das `ALTER TABLE ADD PARTITION`-Anweisungen ausführt.

In den folgenden Abschnitten werden diese Fälle ausführlich beschrieben.

ORC: Lesen nach Index

Eine Tabelle wird in ORC standardmäßig nach dem Index gelesen. Dies wird durch die folgende Syntax definiert:

```
WITH SERDEPROPERTIES (
  'orc.column.index.access'='true')
```

Lesen nach Index erlaubt, die Spalten umzubenennen. Sie verlieren damit jedoch die Möglichkeit, Spalten zu entfernen oder in der Mitte der Tabelle hinzuzufügen.

Damit ORC nach dem Namen liest, sodass Sie Spalten in der Mitte der Tabelle hinzufügen oder entfernen können, setzhen Sie die SerDe-Eigenschaft `orc.column.index.access` in der `CREATE TABLE`-Anweisung auf `false`. In dieser Konfiguration verlieren Sie die Möglichkeit, Spalten umzubenennen.

> **Note**
>
> Wenn in Athena-Engine-Version 2 ORC-Tabellen so eingestellt sind, dass sie nach Namen lesen, erfordert Athena, dass alle Spaltennamen in den ORC-Dateien in Kleinbuchstaben sind. Da Apache Spark beim Generieren von ORC-Dateien keine Feldnamen in Kleinbuchstaben enthält, kann Athena die so generierten Daten möglicherweise nicht lesen. Die Abhilfe besteht darin, die Spalten in Kleinbuchstaben umzubenennen oder Athena-Engine-Version 3 zu verwenden.

Das folgende Beispiel zeigt, wie Sie die ORC abändern, um nach Namen zu lesen:

```
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://schema_updates/orders_orc/';
```

Parquet: Lesen nach Namen

Eine Tabelle wird in Parquet standardmäßig nach dem Namen gelesen. Dies wird durch die folgende Syntax definiert:

```
WITH SERDEPROPERTIES (
  'parquet.column.index.access'='false')
```

Lesen nach Namen ermöglicht Ihnen das Hinzufügen von Spalten in der Mitte der Tabelle und das Entfernen von Spalten. Aber dann verlieren Sie die Möglichkeit, Spalten umzubenennen.

Um Parquet umzustellen, sodass es nach dem Index liest und Sie Spalten umbenennen können, müssen Sie eine Tabelle erstellen, deren SerDe-Eigenschaft `parquet.column.index.access` auf `true` gesetzt ist.

Arten von Updates

In diesem Thema werden einige der Änderungen beschrieben, die Sie in `CREATE TABLE`-Anweisungen am Schema vornehmen können, ohne Ihre Daten tatsächlich zu ändern. Wir überprüfen jede Art Schema-Update und geben an, welche Datenformate sie in Athena unterstützen. Um ein Schema zu aktualisieren, können Sie in einigen Fällen einen `ALTER TABLE`-Befehl verwenden, in anderen Fällen ändern Sie jedoch nicht wirklich eine vorhandene Tabelle. Stattdessen erstellen Sie eine Tabelle mit einem neuen Namen, der das Schema ändert, das Sie in Ihrer ursprünglichen `CREATE TABLE`-Anweisung verwendet haben.

- Hinzufügen von Spalten am Anfang oder in der Mitte der Tabelle
- Hinzufügen von Spalten am Ende der Tabelle
- Spalten entfernen
- Spalten umbenennen
- Reihenfolge von Spalten ändern
- Ändern des Datentyps einer Spalte

Je nachdem, wie sich Ihre Schemata voraussichtlich weiterentwickeln, verwenden Sie ein kompatibles Datenformat, um weiterhin Athena-Abfragen verwenden zu können.

Untersuchen wir als Erstes eine Anwendung, in der Bestellinformationen aus einer `orders`-Tabelle gelesen werden, die in zwei Formaten existiert: CSV und Parquet.

Mit dem folgenden Beispiel wird eine Tabelle in Parquet erstellt:

```
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/';
```

Mit dem folgenden Beispiel wird die gleiche Tabelle in CSV erstellt:

```
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/';
```

In den folgenden Abschnitten sehen wir uns an, wie sich Aktualisierungen für diese Tabellen auf Athena-Abfragen auswirken.

Hinzufügen von Spalten am Anfang oder in der Mitte der Tabelle

Das Hinzufügen von Spalten ist eine der am häufigsten vorkommenden Schemaänderungen. Beispielsweise können Sie eine neue Spalte hinzufügen, um die Tabelle um neue Daten zu erweitern. Alternativ können Sie eine neue Spalte hinzufügen, wenn sich die Quelle für eine vorhandene Spalte geändert hat, und die vorherige Version dieser Spalte beibehalten. So haben Sie Zeit, Anwendungen anzupassen, die von dieser Spalte abhängig sind.

Um Spalten am Anfang oder in der Mitte der Tabelle hinzuzufügen und Abfragen für bestehende Tabellen durchzuführen, verwenden Sie AVRO, JSON und Parquet und ORC, wenn deren SerDe Eigenschaft so gesetzt ist, dass sie dem Namen nach lesen. Weitere Informationen finden Sie unter Index-Zugriff in ORC und Parquet.

Fügen Sie keine Spalten am Anfang oder in der Mitte der Tabelle in CSV und TSV hinzu, da diese Formate von der Reihenfolge abhängen. Das Hinzufügen einer Spalte führt in diesen Fällen zu Schemakonflikten, falls sich das Schema der Partitionen ändert.

Im folgenden Beispiel wird eine neue Tabelle erstellt, die eine `o_comment`-Spalte in der Mitte einer auf JSON-Daten basierenden Tabelle hinzufügt.

```
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/';
```

Hinzufügen von Spalten am Ende der Tabelle

Wenn Sie Tabellen in einem der von Athena unterstützten Formate wie Parquet, ORC, Avro, JSON, CSV und TSV erstellen, können Sie mit der `ALTER TABLE ADD COLUMNS`-Anweisung Spalten nach vorhandenen Spalten, aber vor Partitionsspalten hinzufügen.

Im folgenden Beispiel wird eine `comment`-Spalte am Ende der `orders_parquet`-Tabelle vor allen Partitionsspalten hinzugefügt:

```
ALTER TABLE orders_parquet ADD COLUMNS (comment string)
```

> **Note**
>
> Um im Athena-Abfrage-Editor nach der Ausführung von `ALTER TABLE ADD COLUMNS` eine neue Tabellenspalte anzuzeigen, aktualisieren Sie die Tabellenliste im Editor manuell und erweitern die Tabelle anschließend erneut.

Spalten entfernen

Möglicherweise müssen Sie Spalten aus Tabellen entfernen, wenn sie keine Daten mehr enthalten, oder zum Einschränken des Zugriffs auf die darin enthaltenen Daten.

- Sie können Spalten aus Tabellen in JSON, Avro und Parquet und ORC entfernen, wenn diese dem Namen nach lesen. Weitere Informationen finden Sie unter Index-Zugriff in ORC und Parquet.

- Wir raten davon ab, Spalten aus Tabellen in CSV und TSV zu entfernen, wenn Sie die Tabellen, die Sie bereits in Athena angelegt haben, beibehalten möchten. Das Entfernen einer Spalte stört das Schema und erfordert, dass Sie die Tabelle ohne die entfernte Spalte neu anlegen.

In diesem Beispiel entfernen Sie eine Spalte `` `totalprice` `` aus einer Tabelle in Parquet und führen eine Abfrage aus. In Athena wird Parquet standardmäßig dem Namen nach gelesen. Deshalb lassen wir die Konfiguration SERDEPROPERTIES weg, die das Lesen dem Namen nach angibt. Beachten Sie, dass die folgende Abfrage erfolgreich ist, obwohl Sie das Schema geändert haben:

```
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/';
```

Spalten umbenennen

Möglicherweise möchten Sie in Ihren Tabellen Spalten umbenennen, um die Schreibweise zu korrigieren, Spalten mit besser verständlichen Namen zu versehen oder eine vorhandene Spalte erneut nutzen, um eine Änderung der Reihenfolge für die Spalten zu vermeiden.

Sie können Spalten umbenennen, wenn Sie Ihre Daten in CSV und TSV speichern, oder in Parquet und ORC, wenn diese so konfiguriert sind, dass sie dem Index nach lesen. Weitere Informationen finden Sie unter Index-Zugriff in ORC und Parquet.

Athena liest Daten in CSV und TSV in der Reihenfolge der Spalten in das Schema und gibt sie in der gleichen Reihenfolge aus. Hierbei werden keine Spaltennamen für das Mapping von Daten in einer Spalte verwendet. Aus diesem Grund können Sie keine Spalten in CSV oder TSV umbenennen, ohne Athena-Abfragen zu beschädigen.

Eine Strategie zum Umbenennen von Spalten besteht darin, eine neue Tabelle zu erstellen, die auf denselben zugrunde liegenden Daten basiert, jedoch unter Verwendung neuer Spaltennamen. Im folgenden Beispiel wird eine neue `orders_parquet`-Tabelle

`orders_parquet_column_renamed`erstellt. Im Beispiel wird der Spaltenname `` `o_totalprice` `` zu `` `o_total_price` `` und führt dann eine Abfrage in Athena aus:

```
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/';
```

Im Fall der Parquet-Tabelle wird die folgende Abfrage erfolgreich ausgeführt, aber die umbenannte Spalte zeigt keine Daten an, da auf die Spalte nicht über den Index, sondern über den Namen zugegriffen wurde (ein Standard in Parquet):

```
SELECT *
FROM orders_parquet_column_renamed;
```

Eine Abfrage mit einer Tabelle in CSV sieht ähnlich aus:

```
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/';
```

Bei der CSV-Tabelle wird die folgende Abfrage ausgeführt und für alle Spalten erscheinen Daten, auch für die umbenannte Spalte:

```
SELECT *
FROM orders_csv_column_renamed;
```

Reihenfolge von Spalten ändern

Sie können Spalten nur für Tabellen mit Daten in Formaten neu anordnen, die dem Namen nach lesen, wie JSON oder Parquet, die standardmäßig dem Namen nach lesen. Sie können ORC bei Bedarf auch nach Namen lesen lassen. Weitere Informationen finden Sie unter Index-Zugriff in ORC und Parquet.

Das folgende Beispiel erstellt eine neue Tabelle mit den Spalten in einer anderen Reihenfolge:

```
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/';
```

Ändern des Datentyps einer Spalte

Möglicherweise möchten Sie einen anderen Spaltentyp verwenden, wenn der vorhandene Typ nicht mehr die erforderliche Menge an Informationen aufnehmen kann. Beispielsweise könnten die Werte einer ID-Spalte die Größe des `INT`-Datentyps überschreiten und die Verwendung des `BIGINT`-Datentyps erfordern.

Berücksichtigen Sie bei der Verwendung eines anderen Datentyps für eine Spalte die folgenden Punkte:

- In den meisten Fällen können Sie den Datentyp einer Spalte nicht direkt ändern. Stattdessen erstellen Sie die Athena-Tabelle neu und definieren die Spalte mit dem neuen Datentyp.

- Nur bestimmte Datentypen können als andere Datentypen gelesen werden. In der Tabelle in diesem Abschnitt finden Sie die Datentypen, die so behandelt werden können.
- Für Daten in Parquet und ORC können Sie keinen anderen Datentyp für eine Spalte verwenden, wenn die Tabelle nicht partitioniert ist.
- In partitionierten Tabellen in Parquet und ORC kann sich der Spaltentyp einer Partition vom Spaltentyp einer anderen Partition unterscheiden und Athena wendet `CAST` auf den gewünschten Typ an, wenn dies möglich ist. Weitere Informationen finden Sie unter Vermeiden von Schemakonflikt-Fehlern für Tabellen mit Partitionen.
- Für Tabellen, die nur mit LazySimpleSerde erstellt wurden, ist es möglich, die `ALTER TABLE REPLACE COLUMNS`-Anweisung zu verwenden, um vorhandene Spalten durch einen anderen Datentyp zu ersetzen, aber alle vorhandenen Spalten, die Sie behalten möchten, müssen ebenfalls in der Anweisung neu definiert werden, sonst werden sie gelöscht. Weitere Informationen finden Sie unter ALTER TABLE REPLACE COLUMNS.
- Nur für Apache-Iceberg-Tabellen können Sie die ALTER TABLE CHANGE COLUMN-Anweisung verwenden, um den Datentyp einer Spalte zu ändern. `ALTER TABLE REPLACE COLUMNS` wird für Iceberg-Tabellen nicht unterstützt. Weitere Informationen finden Sie unter Iceberg-Tabellenschema weiterentwickeln.

> **Important**
>
> Wir raten Ihnen dringend, Ihre Abfragen zu testen und zu überprüfen, bevor Sie Datentyp-Änderungen vornehmen. Wenn Athena den Zieldatentyp nicht verwenden kann, schlägt die `CREATE TABLE`-Abfrage möglicherweise fehl.

In der folgenden Tabelle sind Datentypen aufgeführt, die wie andere Datentypen behandelt werden:

Kompatible Datentypen

Original-Datentyp	Verfügbare Ziel-Datentypen
`STRING`	`BYTE`, `TINYINT`, `SMALLINT`, `INT`, `BIGINT`
`BYTE`	`TINYINT`, `SMALLINT`, `INT`, `BIGINT`
`TINYINT`	`SMALLINT`, `INT`, `BIGINT`

Original-Datentyp	Verfügbare Ziel-Datentypen
SMALLINT	INT, BIGINT
INT	BIGINT
FLOAT	DOUBLE

Im folgenden Beispiel wird die CREATE TABLE-Anweisung für die orders_json-Originaltabelle verwendet, um eine neue Tabelle mit dem Namen orders_json_bigint zu erstellen. Die neue Tabelle verwendet BIGINT statt INT als Datentyp für die `o_shippriority`-Spalte.

```
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';
```

Die folgende Abfrage wird erfolgreich ausgeführt, ähnlich wie die SELECT-Originalabfrage vor dem Ändern des Datentyps:

```
Select * from orders_json
LIMIT 10;
```

Updates in Tabellen mit Partitionen

In Athena müssen eine Tabelle und ihre Partitionen die gleichen Datenformate verwenden, ihre Schemata können aber unterschiedlich sein. Wenn Sie eine neue Partition erstellen, erbt die Partition in der Regel das Schema der Tabelle. Im Laufe der Zeit können sich die Schemas ggf. unterscheiden. Gründe sind u. a.:

- Wenn sich das Schema der Tabelle ändert, werden die Schemas für Partitionen nicht aktualisiert, um weiterhin dem Schema der Tabelle zu entsprechen.
- Mit dem AWS Glue Crawler können Sie Daten in Partitionen mit abweichenden Schemata finden. Das bedeutet: Wenn Sie eine Tabelle in Athena mit AWS Glue erstellen, können sich die Schemata für die Tabelle und ihre Partitionen unterscheiden, nachdem der Crawler mit der Verarbeitung fertig ist.
- Wenn Sie Partitionen direkt mit einer AWS-API hinzufügen.

Athena verarbeitet Tabellen mit Partitionen erfolgreich, wenn sie die folgenden Beschränkungen erfüllen. Wenn diese Beschränkungen nicht erfüllt sind, gibt Athena einen HIVE_PARTITION_SCHEMA_MISMATCH-Fehler aus.

- Das Schema jeder Partition ist kompatibel mit dem Tabellenschema.
- Das Datenformat der Tabelle ermöglicht die Art der Aktualisierung, die Sie ausführen möchten: Spalten hinzufügen, löschen, ihre Reihenfolge ändern oder den Datentyp einer Spalte ändern.

  Beispiel: In den Formaten CSV und TSV können Sie Spalten umbenennen, neue Spalten am Ende der Tabelle hinzufügen und den Datentyp einer Spalte ändern, falls die Typen kompatibel sind. Sie können aber keine Spalten entfernen. Bei anderen Formaten können Sie Spalten hinzufügen oder entfernen oder den Datentyp in einen anderen ändern, wenn die Typen kompatibel sind. Weitere Informationen finden Sie unter Zusammenfassung: Updates und Datenformate in Athena.

Vermeiden von Schemakonflikt-Fehlern für Tabellen mit Partitionen

Zu Beginn der Abfrageausführung überprüft Athena das Schema einer Tabelle, indem überprüft wird, dass der Datentyp jeder Spalte in der Tabelle und der Partition kompatibel ist.

- Für Parquet- und ORC-Datenspeichertypen stützt sich Athena auf die Spaltennamen und verwendet sie für die auf Spaltennamen basierte Überprüfung des Schemas. Dadurch werden `HIVE_PARTITION_SCHEMA_MISMATCH`-Fehler für Tabellen mit Partitionen in Parquet und ORC verhindert. (Dies gilt für ORC, wenn die SerDe-Eigenschaft für den Zugriff auf den Index nach Name festgelegt ist: `orc.column.index.access=FALSE`. Parquet liest den Index standardmäßig nach Namen).
- Für CSV, JSON und Avro verwendet Athena eine Index-basierte Schemaverifizierung. Das bedeutet: Wenn bei Ihnen ein Schemakonflikt-Fehler auftritt, sollten Sie die Partition löschen, die einen Schemakonflikt auslöst, und sie erneut erstellen. So kann Athena sie ohne Fehler abfragen.

Athena vergleicht das Schema der Tabelle mit den Partitionsschemata. Wenn Sie eine Tabelle in CSV, JSON und AVRO in Athena mit AWS Glue Crawler erstellen, können sich die Schemata für die Tabelle und ihre Partitionen unterscheiden, nachdem der Crawler mit der Verarbeitung fertig ist. Tritt ein Konflikt zwischen dem Tabellen- und dem Partitionsschema auf, scheitern Ihre Abfragen in Athena wegen eines Fehlers beim Überprüfen des Schemas, ähnlich dem Folgenden: „‚crawler_test.click_avro' ist als Typ ‚string' deklariert, in der Partition ‚partition_0=2017-01-17' ist die Spalte ‚col68' aber mit dem Typ ‚double' deklariert."

Eine typische Behelfslösung für diesen Fehler besteht darin, dass die Partition entfernt wird, die den Fehler auslöst, und Sie sie dann erneut erstellen. Weitere Informationen finden Sie unter ALTER TABLE DROP PARTITION und ALTER TABLE ADD PARTITION.

Abfragen von Arrays

Sie können mit Amazon Athena Arrays erstellen, sie verketten, in andere Datentypen umwandeln und anschließend filtern, reduzieren und sortieren.

Themen

- Erstellen von Arrays
- Verketten von Zeichenfolgen und Arrays
- Konvertieren von Array-Datentypen
- Suchen nach Längen
- Zugreifen auf Array-Elemente
- Reduzieren von verschachtelten Arrays
- Erstellen von Arrays aus Unterabfragen
- Filtern von Arrays
- Sortieren von Arrays
- Verwenden von Aggregations-Funktionen mit Arrays
- Konvertieren von Arrays in Zeichenfolgen
- Verwenden von Arrays zum Erstellen von Karten
- Abfragen von Arrays mit komplexen Typen und verschachtelten Strukturen

Erstellen von Arrays

Zum Erstellen eines Arrayliterals in Athena verwenden Sie das Schlüsselwort ARRAY, gefolgt von Klammern [ ], und schließen die durch Kommata getrennten Array-Elemente ein.

Beispiele

Diese Abfrage erstellt ein Array mit vier Elementen.

```
SELECT ARRAY [1,2,3,4] AS items
```

Folgendes wird zurückgegeben:

```
+-----------+
| items     |
+-----------+
| [1,2,3,4] |
+-----------+
```

Diese Abfrage erstellt zwei Arrays.

```
SELECT ARRAY[ ARRAY[1,2], ARRAY[3,4] ] AS items
```

Folgendes wird zurückgegeben:

```
+--------------------+
| items              |
+--------------------+
| [[1, 2], [3, 4]]   |
+--------------------+
```

Zum Erstellen eines Arrays aus ausgewählten Spalten kompatibler Typen nutzen Sie eine Abfrage, wie im folgenden Beispiel gezeigt:

```
WITH
dataset AS (
  SELECT 1 AS x, 2 AS y, 3 AS z
)
SELECT ARRAY [x,y,z] AS items FROM dataset
```

Diese Abfrage gibt Folgendes zurück:

```
+-----------+
| items     |
+-----------+
| [1,2,3]   |
+-----------+
```

Im folgenden Beispiel werden zwei Arrays ausgewählt und in Form einer Willkommensnachricht zurückgegeben.

```
WITH
dataset AS (
  SELECT
    ARRAY ['hello', 'amazon', 'athena'] AS words,
    ARRAY ['hi', 'alexa'] AS alexa
)
SELECT ARRAY[words, alexa] AS welcome_msg
FROM dataset
```

Diese Abfrage gibt Folgendes zurück:

```
+----------------------------------------+
| welcome_msg                            |
+----------------------------------------+
| [[hello, amazon, athena], [hi, alexa]] |
+----------------------------------------+
```

Zum Erstellen eines Arrays aus Schlüssel/Wert-Paaren nutzen Sie den MAP-Operator. Dieser verwendet ein Schlüssel-Array gefolgt von einem Werte-Array, wie im folgenden Beispiel gezeigt:

```
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
```

Diese Abfrage gibt Folgendes zurück:

```
+-------------------------------------------------------------------------------------------------------
+
| people
               |
```

```
+-----------------------------------------------------------------------------------------------
+
| [{last=Smith, first=Bob, age=40}, {last=Doe, first=Jane, age=30}, {last=Smith,
 first=Billy, age=8}] |
+-----------------------------------------------------------------------------------------------
+
```

Verketten von Zeichenfolgen und Arrays

Verketten von Zeichenfolgen

Um zwei Zeichenfolgen zu verketten, können Sie doppelte Pipe-**| |**-Operator verwenden, wie im folgenden Beispiel gezeigt.

```
SELECT 'This' || ' is' || ' a' || ' test.' AS Concatenated_String
```

Diese Abfrage gibt Folgendes zurück:

#	Concatenated_String
1	This is a test.

Sie können die `concat()`-Funktion verwenden, um dasselbe Ergebnis zu erzielen.

```
SELECT concat('This', ' is', ' a', ' test.') AS Concatenated_String
```

Diese Abfrage gibt Folgendes zurück:

#	Concatenated_String
1	This is a test.

Sie können die `concat_ws()`-Funktion verwenden, um Zeichenketten mit dem im ersten Argument angegebenen Trennzeichen zu verketten.

```
SELECT concat_ws(' ', 'This', 'is', 'a', 'test.') as Concatenated_String
```

Diese Abfrage gibt Folgendes zurück:

#	Concatenated_String
1	This is a test.

Um zwei Spalten des Datentyps „Zeichenfolge“ mit einem Punkt zu verketten, verweisen Sie mit doppelten Anführungszeichen auf die beiden Spalten und schließen Sie den Punkt als hartcodierte Zeichenfolge in einfache Anführungszeichen ein. Wenn eine Spalte nicht vom Datentyp „Zeichenfolge“ ist, können Sie `CAST("column_name" as VARCHAR)` verwenden, um die Spalte zuerst umzuwandeln.

```
SELECT "col1" || '.' || "col2" as Concatenated_String
FROM my_table
```

Diese Abfrage gibt Folgendes zurück:

#	Concatenated_String
1	*col1_string_value* .*col2_string_value*

Verketten von Arrays

Sie können die gleichen Techniken verwenden, um Arrays zu verketten.

Verwenden Sie zum Verketten von mehreren Arrays den doppelten Pipe-Operator `||`.

```
SELECT ARRAY [4,5] || ARRAY[ ARRAY[1,2], ARRAY[3,4] ] AS items
```

Diese Abfrage gibt Folgendes zurück:

#	Elemente
1	[[4, 5], [1, 2], [3, 4]]

Um mehrere Arrays zu einem einzigen Array zu kombinieren, verwenden Sie den Doppelpipe-Operator oder die `concat()`-Funktion.

```
WITH
dataset AS (
  SELECT
    ARRAY ['Hello', 'Amazon', 'Athena'] AS words,
    ARRAY ['Hi', 'Alexa'] AS alexa
)
SELECT concat(words, alexa) AS welcome_msg
FROM dataset
```

Diese Abfrage gibt Folgendes zurück:

#	welcome_msg
1	[Hello, Amazon, Athena, Hi, Alexa]

Weitere Informationen zum `concat()` anderer Zeichenfolgenfunktionen finden Sie unter Zeichenfolgenfunktionen und -operatoren in der Trino-Dokumentation.

Konvertieren von Array-Datentypen

Um Daten in einem Array in einen unterstützten Datentyp zu konvertieren, verwenden Sie den `CAST`-Operator als `CAST(value AS type)`. Athena unterstützt alle nativen Presto-Datentypen.

```
SELECT
   ARRAY [CAST(4 AS VARCHAR), CAST(5 AS VARCHAR)]
AS items
```

Diese Abfrage gibt Folgendes zurück:

```
+-------+
| items |
+-------+
| [4,5] |
+-------+
```

Erstellen Sie zwei Arrays mit Schlüssel-Wert-Paar-Elementen, konvertieren Sie sie zu JSON und verketten Sie sie wie in diesem Beispiel:

```
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
```

Diese Abfrage gibt Folgendes zurück:

```
+-------------------------------------------------+
| items                                           |
+-------------------------------------------------+
| [{"a1":1,"a2":2,"a3":3}, {"b1":4,"b2":5,"b3":6}] |
+-------------------------------------------------+
```

Suchen nach Längen

Die Funktion `cardinality` gibt die Länge eines Arrays wie in diesem Beispiel zurück:

```
SELECT cardinality(ARRAY[1,2,3,4]) AS item_count
```

Diese Abfrage gibt Folgendes zurück:

```
+------------+
| item_count |
+------------+
| 4          |
+------------+
```

Zugreifen auf Array-Elemente

Verwenden Sie für den Zugriff auf Array-Elemente den `[]`-Operator, wobei 1 das erste Elemente bezeichnet, 2 das zweite Elemente usw., wie im folgenden Beispiel:

```
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
```

Diese Abfrage gibt Folgendes zurück:

```
+------------------------+
```

```
| item                    |
+-------------------------+
| {"a1":1,"a2":2,"a3":3} |
+-------------------------+
```

Verwenden Sie für den Zugriff auf die Array-Elemente an einer bestimmten Position (die Indexposition) die Funktion `element_at()` und geben Sie den Namen des Array und die Indexposition an:

- Wenn der Index größer als 0 ist, gibt `element_at()` das angegebene Element zurück und zählt das Array dabei von Anfang bis Ende durch. Die Funktion verhält sich dabei wie der `[]`-Operator.
- Wenn der Index kleiner als 0 ist, gibt `element_at()` das angegebene Element zurück und zählt das Array dabei von Ende bis Anfang durch.

In der folgenden Abfrage wird ein Array `words` erstellt und das erste Element `hello` als `first_word`, das zweite Element `amazon` (vom Ende des Arrays aus gezählt) als `middle_word` und das dritte Element `athena` als `last_word` ausgewählt.

```
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
```

Diese Abfrage gibt Folgendes zurück:

```
+----------------------------------------+
| first_word  | middle_word | last_word  |
+----------------------------------------+
| hello       | amazon      | athena     |
+----------------------------------------+
```

Reduzieren von verschachtelten Arrays

Beim Arbeiten mit verschachtelten Arrays müssen Sie verschachtelte Array-Elemente oft zu einem einzelnen Array erweitern oder das Array in mehrere Zeilen erweitern.

Beispiele

Verwenden Sie die Funktion `flatten`, um die Elemente eines verschachtelten Arrays zu einem einzelnen Array mit Werten zu reduzieren. Diese Abfrage gibt eine Zeile für jedes Element im Array zurück.

```
SELECT flatten(ARRAY[ ARRAY[1,2], ARRAY[3,4] ]) AS items
```

Diese Abfrage gibt Folgendes zurück:

```
+-----------+
| items     |
+-----------+
| [1,2,3,4] |
+-----------+
```

Um ein Array in mehrere Zeilen zu reduzieren, verwenden Sie `CROSS JOIN` in Verbindung mit dem `UNNEST`-Operator wie in diesem Beispiel:

```
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)
```

Diese Abfrage gibt Folgendes zurück:

```
+----------------------+
| department  | names  |
+----------------------+
| engineering | Sharon |
+----------------------|
| engineering | John   |
+----------------------|
| engineering | Bob    |
+----------------------|
| engineering | Sally  |
+----------------------+
```

Um ein Array mit Schlüssel-Wert-Paaren zu reduzieren, verteilen Sie die ausgewählten Schlüssel wie in diesem Beispiel in Spalten:

```
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)
```

Diese Abfrage gibt Folgendes zurück:

```
+---------------------------------------+
| first_name | last_name | department  |
+---------------------------------------+
| Bob        | Smith     | engineering |
| Jane       | Doe       | engineering |
| Billy      | Smith     | engineering |
+---------------------------------------+
```

Wählen Sie aus einer Liste mit Mitarbeitern den Mitarbeiter mit den höchsten kombinierten Werten aus. `UNNEST` kann in der Klausel `FROM` ohne vorangestelltes `CROSS JOIN`, da es der Standard-JOIN-Operator ist und somit vorausgesetzt wird.

```
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
```

Diese Abfrage gibt Folgendes zurück:

```
+---------------------------------+
| name | department | total_score |
+---------------------------------+
| Amy  | devops     | 54          |
+---------------------------------+
```

Wählen Sie aus einer Liste mit Mitarbeitern den Mitarbeiter mit den höchsten einzelnen Werten aus.

```
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
```

Diese Abfrage gibt Folgendes zurück:

```
+--------------+
| name | score |
+--------------+
| Amy  | 15    |
+--------------+
```

Überlegungen und Einschränkungen

Wenn UNNEST für ein oder mehrere Arrays in der Abfrage verwendet wird und eines der Arrays NULL ist, gibt die Abfrage keine Zeilen zurück. Wenn UNNEST in ein Array verwendet wird, das eine leere Zeichenfolge ist, wird die leere Zeichenfolge zurückgegeben.

In der folgenden Abfrage werden beispielsweise keine Zeilen zurückgegeben, da das zweite Array null ist.

```
SELECT
    col1,
    col2
FROM UNNEST (ARRAY ['apples','oranges','lemons']) AS t(col1)
CROSS JOIN UNNEST (ARRAY []) AS t(col2)
```

Im nächsten Beispiel wird das zweite Array so geändert, dass es eine leere Zeichenfolge enthält. Für jede Zeile gibt die Abfrage den Wert in `col1` und eine leere Zeichenfolge für den Wert in `col2` zurück. Die leere Zeichenfolge im zweiten Array ist erforderlich, damit die Werte im ersten Array zurückgegeben werden.

```
SELECT
    col1,
    col2
FROM UNNEST (ARRAY ['apples','oranges','lemons']) AS t(col1)
CROSS JOIN UNNEST (ARRAY ['']) AS t(col2)
```

Erstellen von Arrays aus Unterabfragen

Erstellen Sie ein Array aus einer Sammlung von Zeilen.

```
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)
```

Diese Abfrage gibt Folgendes zurück:

```
+-----------------+
| array_items     |
+-----------------+
| [1, 2, 3, 4, 5] |
+-----------------+
```

Nutzen Sie das Schlüsselwort `distinct`, um ein Array, bestehend aus eindeutigen Werten, aus einer Reihe von Zeilen zu erstellen.

```
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)
```

Diese Abfrage gibt das folgende Ergebnis zurück. Beachten Sie, dass die Reihenfolge nicht garantiert ist.

```
+-----------------+
| array_items     |
+-----------------+
| [1, 2, 3, 4, 5] |
+-----------------+
```

Weitere Informationen zur Verwendung der `array_agg`-Funktion siehe Aggregate functions (Aggregationsfunktionen) in der Trino-Dokumentation.

Filtern von Arrays

Erstellen Sie ein Array aus einer Reihe von Zeilen, die den Filterkriterien entsprechen.

```
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
```

Diese Abfrage gibt Folgendes zurück:

```
+-------------+
| array_items |
+-------------+
| [4, 5]      |
+-------------+
```

Filtern Sie ein Array basierend darauf, ob Elemente einen bestimmten Wert (z. B. 2) enthalten, wie in diesem Beispiel:

```
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)
```

Diese Abfrage gibt Folgendes zurück:

```
+--------------+
| array_items  |
```

```
+--------------+
| [1, 2, 3, 4] |
+--------------+
```

Die **`filter`**-Funktion

```
filter(ARRAY [list_of_values], boolean_function)
```

Sie können die `filter`-Funktion auf einem ARRAY-Ausdruck verwenden, um ein neues Array zu erstellen, das die Teilmenge der Elemente in *`list_of_values`* (Werteliste) ist, für die *`boolean_function`* (Boolesche Funktion) „true“ (wahr) ist. Die Funktion `filter` kann nützlich sein, wenn Sie die Funktion *`UNNEST`* nicht verwenden können.

Im folgenden Beispiel wird im Array `[1,0,5,-1]` nach Werten größer als Null gefiltert.

```
SELECT filter(ARRAY [1,0,5,-1], x -> x>0)
```

Ergebnisse

```
[1,5]
```

Im folgenden Beispiel wird im Array `[-1, NULL, 10, NULL]` nach Werten ungleich Null gefiltert.

```
SELECT filter(ARRAY [-1, NULL, 10, NULL], q -> q IS NOT NULL)
```

Ergebnisse

```
[-1,10]
```

Sortieren von Arrays

Um ein sortiertes Array eindeutiger Werte aus einer Reihe von Zeilen zu erstellen, können Sie wie im folgenden Beispiel die Funktion array_sort verwenden.

```
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)
```

Diese Abfrage gibt Folgendes zurück:

```
+--------------------+
| array_items        |
+--------------------+
| [1, 2, 3, 4, 5, 6] |
+--------------------+
```

Informationen zum Erweitern eines Arrays in mehrere Zeilen finden Sie unter Reduzieren von verschachtelten Arrays.

Verwenden von Aggregations-Funktionen mit Arrays

- Fügen Sie mit SUM Werte zu einem Array hinzu, wie im folgenden Beispiel dargestellt.
- Aggregieren Sie mit `array_agg` mehrere Zeilen in einem Array. Weitere Informationen finden Sie unter Erstellen von Arrays aus Unterabfragen.

> **Note**
>
> ORDER BY wird für Aggregationsfunktionen unterstützt, die in Athena Engine Version 2 beginnen.

```
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;
```

In der letzten SELECT-Anweisung können Sie statt `sum( )` und UNNEST zu verwenden, die Verarbeitungszeit und die Datenübertragung, wie im folgenden Beispiel, mit `reduce( )` verkürzen.

```
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;
```

Jede Abfrage gibt die folgenden Ergebnisse zurück. Die Reihenfolge der zurückgegebenen Ergebnisse ist nicht gewährleistet.

```
+----------------------+
| array_items  | total |
+----------------------+
| [1, 2, 3, 4] | 10    |
| [5, 6, 7, 8] | 26    |
| [9, 0]       | 9     |
+----------------------+
```

Konvertieren von Arrays in Zeichenfolgen

Zum Konvertieren eines Arrays in eine einzelne Zeichenfolge verwenden Sie die `array_join`-Funktion. Im folgenden eigenständigen Beispiel wird eine Tabelle namens `dataset` erstellt, die ein aliasiertes Array namens `words` enthält. Die Abfrage verwendet `array_join`, um die Array-Elemente in `words` zu verbinden, sie durch Leerzeichen zu trennen und die resultierende Zeichenfolge in einer Alias-Spalte namens `welcome_msg` zurückzugeben.

```
WITH
dataset AS (
  SELECT ARRAY ['hello', 'amazon', 'athena'] AS words
)
SELECT array_join(words, ' ') AS welcome_msg
FROM dataset
```

Diese Abfrage gibt Folgendes zurück:

```
+---------------------+
| welcome_msg         |
+---------------------+
| hello amazon athena |
+---------------------+
```

Verwenden von Arrays zum Erstellen von Karten

Zuordnungen sind Schlüssel-Wert-Paare, die aus einem in Athena verfügbaren Datentyp bestehen. Verwenden Sie zum Erstellen von Zuordnungen den `MAP`-Operator und übergeben Sie zwei Arrays: Das erste Array enthält die Spaltennamen (Schlüssel) und das zweite Array enthält die Werte. Alle Werte in den Arrays müssen denselben Typ haben. Wenn bestimmte Zuordnungswerte im Array einen anderen Typ benötigen, können Sie sie später umwandeln.

Beispiele

In diesem Beispiel wird ein Benutzer aus einem Dataset ausgewählt. Dem verwendeten `MAP`-Operator werden zwei Arrays übergeben. Das erste Array enthält Werte für die Spaltennamen wie "first", "last" und "age". Das zweite Array enthält die Werte für die einzelnen Spalten, z. B. "Bob", "Smith", "35".

```
WITH dataset AS (
  SELECT MAP(
    ARRAY['first', 'last', 'age'],
    ARRAY['Bob', 'Smith', '35']
  ) AS user
)
SELECT user FROM dataset
```

Diese Abfrage gibt Folgendes zurück:

```
+--------------------------------+
| user                           |
+--------------------------------+
| {last=Smith, first=Bob, age=35} |
+--------------------------------+
```

Sie können Map-Werte abrufen, indem Sie den Feldnamen gefolgt von `[key_name]` wie in diesem Beispiel auswählen:

```
WITH dataset AS (
 SELECT MAP(
   ARRAY['first', 'last', 'age'],
   ARRAY['Bob', 'Smith', '35']
 ) AS user
)
SELECT user['first'] AS first_name FROM dataset
```

Diese Abfrage gibt Folgendes zurück:

```
+------------+
| first_name |
+------------+
| Bob        |
+------------+
```

Abfragen von Arrays mit komplexen Typen und verschachtelten Strukturen

Quelldaten enthalten oft Arrays mit komplexen Datentypen und verschachtelten Strukturen. In den Beispielen in diesem Abschnitt wird gezeigt, wie Sie mit Athena-Abfragen den Datentyp eines Elements ändern, Elemente in einem Array finden und Schlüsselwörter finden können.

- Erstellen einer `ROW`
- Ändern von Feldnamen in Arrays mit `CAST`
- Filtern von Arrays mit der `.` Notation
- Filtern von Arrays mit verschachtelten Werten
- Filtern von Arrays mit `UNNEST`
- Suchen nach Schlüsselwörtern in Arrays mithilfe von `regexp_like`

Erstellen einer `ROW`

> **Note**
>
> In den Beispielen in diesem Abschnitt wird `ROW` verwendet, um Beispieldaten zu erstellen. Wenn Sie Tabellen mit Athena abfragen, müssen Sie keine `ROW`-Datentypen erstellen, da sie bereits aus Ihrer Datenquelle erstellt wurden. Wenn Sie `CREATE_TABLE` verwenden, definiert Athena eine `STRUCT` darin, füllt sie mit Daten und erstellt für jede Zeile in einem Dataset den `ROW`-Datentyp für Sie. Der zugrunde liegende `ROW`-Datentyp besteht aus benannten Feldern beliebiger unterstützter SQL-Datentypen.

```
WITH dataset AS (
 SELECT
   ROW('Bob', 38) AS users
 )
SELECT * FROM dataset
```

Diese Abfrage gibt Folgendes zurück:

```
+-------------------------+
| users                   |
+-------------------------+
| {field0=Bob, field1=38} |
+-------------------------+
```

Ändern von Feldnamen in Arrays mit `CAST`

Verwenden Sie zum Ändern des Feldnamens in einem Array, das `ROW`-Werte enthält, eine `CAST`-Anweisung für die `ROW`-Deklaration:

```
WITH dataset AS (
  SELECT
    CAST(
      ROW('Bob', 38) AS ROW(name VARCHAR, age INTEGER)
    ) AS users
)
SELECT * FROM dataset
```

Diese Abfrage gibt Folgendes zurück:

```
+--------------------+
| users              |
+--------------------+
| {NAME=Bob, AGE=38} |
+--------------------+
```

> **Note**
>
> Im obigen Beispiel deklarieren Sie `name` als `VARCHAR`, da dies der Datentyp in Presto ist. Wenn Sie dieses `STRUCT` innerhalb einer `CREATE TABLE`-Anweisung deklarieren, verwenden Sie den Datentyp `String`, da Hive diesen Datentyp als `String` definiert.

Filtern von Arrays mit der `.` Notation

Wählen Sie im folgenden Beispiel das Feld `accountId` aus der Spalte `userIdentity` einer AWS CloudTrail-Protokolltabelle mit der Punkt-`.`-Notation aus. Weitere Informationen finden Sie unter Abfragen von AWS CloudTrail-Protokollen.

```
SELECT
  CAST(useridentity.accountid AS bigint) as newid
FROM cloudtrail_logs
LIMIT 2;
```

Diese Abfrage gibt Folgendes zurück:

```
+--------------+
| newid        |
+--------------+
| 112233445566 |
+--------------+
| 998877665544 |
+--------------+
```

Um ein Array mit Werten abzufragen, senden Sie diese Abfrage:

```
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
```

Sie erhalten das folgende Ergebnis:

```
+-----------------------------------------------------------------+
| users                                                           |
+-----------------------------------------------------------------+
| [{NAME=Bob, AGE=38}, {NAME=Alice, AGE=35}, {NAME=Jane, AGE=27}] |
+-----------------------------------------------------------------+
```

Filtern von Arrays mit verschachtelten Werten

Große Arrays enthalten oft verschachtelte Strukturen und Sie müssen die darin enthaltenen Werte filtern oder suchen können.

Um ein Dataset für ein Array mit Werten zu definieren, das einen verschachtelten `BOOLEAN`-Wert enthält, senden Sie die folgende Abfrage:

```
WITH dataset AS (
  SELECT
    CAST(
      ROW('aws.amazon.com', ROW(true)) AS ROW(hostname VARCHAR, flaggedActivity
 ROW(isNew BOOLEAN))
    ) AS sites
)
SELECT * FROM dataset
```

Sie erhalten das folgende Ergebnis:

```
+-------------------------------------------------------+
| sites                                                 |
+-------------------------------------------------------+
| {HOSTNAME=aws.amazon.com, FLAGGEDACTIVITY={ISNEW=true}}  |
+-------------------------------------------------------+
```

Verwenden Sie nun zum Filtern von und Zugreifen auf den `BOOLEAN`-Wert dieses Elements die Punkt-`.`-Notation.

```
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
```

Mit dieser Abfrage werden die verschachtelten Felder ausgewählt und folgendes Ergebnis zurückgegeben:

```
+------------------------+
| hostname       | isnew |
+------------------------+
| aws.amazon.com | true  |
+------------------------+
```

Filtern von Arrays mit **`UNNEST`**

Um ein Array mit einer verschachtelten Struktur nach einem untergeordneten Element zu filtern, verwenden Sie eine Abfrage mit einem UNNEST-Operator. Weitere Informationen zu UNNEST finden Sie unter Reduzieren von verschachtelten Arrays.

Diese Abfrage sucht beispielsweise Hostnamen von Websites im Datensatz.

```
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
```

Folgendes wird zurückgegeben:

```
+------------------------+
| hostname       | isnew |
+------------------------+
| aws.amazon.com | true  |
+------------------------+
```

Suchen nach Schlüsselwörtern in Arrays mithilfe von **`regexp_like`**

Die folgenden Beispiele veranschaulichen, wie Sie mit der Funktion regexp_like ein Dataset nach einem Schlüsselwort innerhalb eines Elements in einem Array durchsuchen. Sie übernimmt als Eingabe ein auszuwertendes reguläres Ausdrucksmuster oder eine Liste von Begriffen, die durch einen senkrechten Strich (|) voneinander getrennt sind, und bestimmt, ob diese Eingabe in der angegebenen Zeichenfolge enthalten ist.

Das reguläre Ausdrucksmuster muss innerhalb der Zeichenfolge enthalten sein, braucht aber nicht genau mit ihr übereinzustimmen. Wenn es mit der gesamten Zeichenfolge übereinstimmen soll, setzen Sie das Muster zwischen ein ^ am Anfang und ein $ am Ende, z. B. `'^pattern$'`.

Stellen Sie sich ein Array von Websites mit deren Hostname und einem `flaggedActivity`-Element vor. Dieses Element enthält ein `ARRAY` mit mehreren `MAP`-Elementen, die jeweils beliebte Schlüsselwörter und deren Beliebtheitswert enthalten. Angenommen, Sie möchten ein bestimmtes Schlüsselwort innerhalb einer `MAP` in diesem Array finden.

Um dieses Dataset nach Websites mit einem bestimmten Schlüsselwort zu durchsuchen, verwenden Sie `regexp_like` anstelle des ähnlichen SQL-`LIKE`-Operators, da die Suche nach einer großen Anzahl von Schlüsselwörtern mit `regexp_like` effizienter ist.

Example Beispiel 1: Verwenden von **`regexp_like`**

Die Abfrage in diesem Beispiel sucht mit der Funktion `regexp_like` nach den Begriffen `'politics|bigdata'` in den Werten innerhalb des Arrays:

```
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)
```

Diese Abfrage gibt zwei Websites zurück:

```
+----------------+
```

```
| hostname       |
+----------------+
| aws.amazon.com |
+----------------+
| news.cnn.com   |
+----------------+
```

Example Beispiel 2: Verwenden von **`regexp_like`**

Bei der Abfrage im folgenden Beispiel werden die Gesamtbeliebtheitswerte für die Websites, die mit Ihren Suchbegriffen übereinstimmen, mit der Funktion `regexp_like` addiert und dann von hoch nach niedrig sortiert.

```
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
```

Diese Abfrage gibt zwei Websites zurück:

```
+-----------------------------------+
| hostname       | terms    | total  |
+----------------+------------------+
| news.cnn.com   | politics |  241   |
+----------------+------------------+
| aws.amazon.com | bigdata |  10    |
+----------------+------------------+
```

Abfragen von koordinatenbasierten Daten

Koordinatenbasierte Daten enthalten Kennungen, die eine geografische Position für ein Objekt angeben. Zu den Beispielen für diese Art von Daten zählen Wetterberichte, Wegbeschreibungen, Tweets mit geografischen Positionen, Filialstandorte und Flugrouten. Koordinatenbasierte Daten spielen eine wichtige Rolle bei Geschäftsanalysen, der Berichterstattung und bei Vorhersagen.

Geografische Kennungen, wie z. B. Längen- und Breitengrade, ermöglichen das Umwandeln von Postanschriften in geografische Koordinaten.

Themen

- Was ist eine koordinatenbasierte Abfrage?
- Eingabedatenformate und Geometrie-Datentypen
- Unterstützte koordinatenbasierte Funktionen

- Beispiele: Koordinatenbasierte Abfragen

Was ist eine koordinatenbasierte Abfrage?

Koordinatenbasierte Abfragen sind spezielle SQL-Abfragen, die in Athena unterstützt werden. Sie unterscheiden sich von nicht koordinatenbezogenen SQL-Abfragen wie folgt:

- Verwendung der folgenden spezialisierten Geometrie-Datentypen: `pointline`, `multiline`, `polygon` und `multipolygon`.
- Angeben von Beziehungen zwischen Geometrie-Datentypen wie `distance`, `equals`, `crosses`, `touches`, `overlaps`, `disjoint` und anderen.

Mit koordinatenbasierten Abfragen in Athena können Sie diese und ähnliche Operationen ausführen:

- Ermitteln Sie den Abstand zwischen zwei Punkten.
- Überprüfen Sie, ob ein Bereich (Polygon) ein weiteres enthält.
- Feststellen, ob eine Linie eine andere oder ein Polygon berührt oder schneidet.

Um beispielsweise einen Geometriedatentyp `point` aus Werten des Typs `double` für die geografischen Koordinaten des Mount Rainier in Athena zu erhalten, verwenden Sie die Geodatenfunktion `ST_Point (longitude, latitude)` wie im folgenden Beispiel.

```
ST_Point(-121.7602, 46.8527)
```

Eingabedatenformate und Geometrie-Datentypen

Zum Verwenden von koordinatenbasierten Funktionen in Athena geben Sie die Daten im WKT-Format ein oder nutzen Sie den Hive JSON SerDe. Sie können auch die in Athena unterstützten Geometrie-Datentypen nutzen.

Eingabedatenformate

Zur Verarbeitung von koordinatenbasierten Abfragen unterstützt Athena Eingabedaten in diesen Datenformaten:

- WKT (Well-known Text). In Athena wird WKT als `varchar(x)`- oder `string`-Datentyp dargestellt.

- JSON-kodierte koordinatenbasierte Daten. Zum Analysieren von JSON-Dateien mit koordinatenbasierten Daten und zum Erstellen von Tabellen für diese verwendet Athena den Hive JSON SerDe. Weitere Informationen zur Verwendung dieses SerDe in Athena finden Sie unter JSON-SerDe-Bibliotheken.

Geometrie-Datentypen

Zur Verarbeitung von koordinatenbasierten Abfragen unterstützt Athena diese speziellen Geometrie-Datentypen:

- `point`
- `line`
- `polygon`
- `multiline`
- `multipolygon`

Unterstützte koordinatenbasierte Funktionen

Die koordinatenbasierten Funktionen, die in Athena verfügbar sind, hängen von der verwendeten Engine-Version ab.

- Weitere Informationen zu den Geodatenfunktionen in Athena-Engine-Version 3 finden Sie unter Geodatenfunktionen in der Trino-Dokumentation.
- Eine Liste der geänderten Funktionsnamen und neuen Funktionen ab Athena-Engine-Version 2 finden Sie unter Namensänderungen und neue Funktionen der Geodaten-Funktion in Athena-Engine-Version 2.

Weitere Informationen zum Athena-Engine-Versioning finden Sie unter Athena-Engine-Versionierung.

Themen

- Geodatenfunktionen in Athena-Engine-Version 3
- Geodaten-Funktionen in Athena-Engine-Version 2

Geodatenfunktionen in Athena-Engine-Version 3

Weitere Informationen zu den Geodatenfunktionen in Athena-Engine-Version 3 finden Sie unter Geodatenfunktionen in der Trino-Dokumentation.

Geodaten-Funktionen in Athena-Engine-Version 2

In diesem Thema werden die ESRI-Geodatenfunktionen aufgeführt, die ab der Athena-Engine-Version 2 unterstützt werden. Weitere Informationen über Athena-Engine-Versionen finden Sie unter Athena-Engine-Versionierung.

Änderungen in Athena-Engine-Version 2.

- Die Eingabe- und Ausgabetypen für einige Funktionen haben sich geändert. Vor allem wird der Typ `VARBINARY` nicht mehr direkt für die Eingabe unterstützt. Weitere Informationen finden Sie unter Änderungen an Geodatenfunktionen.
- Die Namen einiger räumlicher Funktionen haben sich geändert. Weitere Informationen finden Sie unter Änderungen des Geodaten-Funktionennamens in Athena-Engine-Version 2.
- Es wurden neue Funktionen hinzugefügt. Weitere Informationen finden Sie unter Neue Geodaten-Funktionen in Athena-Engine-Version 2.

Athena unterstützt die folgenden Arten von koordinatenbasierten Funktionen:

- Konstruktor-Funktionen
- Koordinatenbasierte Verhältnisfunktionen
- Operationsfunktionen
- Zugriffsfunktionen
- Aggregationsfunktionen
- Bing-Kachelfunktionen

Konstruktor-Funktionen

Mithilfe von Konstruktorfunktionen erhalten Sie binäre Werte der Geometriedatentypen `point`, `line` oder `polygon`. Sie können diese Funktionen auch verwenden, um binäre Daten in Text umzuwandeln und binäre Werte für Geometriedaten zu erhalten, die als Well-Known Text (WKT) ausgedrückt werden.

ST_AsBinary(geometry)

Gibt einen varbinären Datentyp zurück, der die WKB-Darstellung der angegebenen Geometrie enthält. Beispiel:

```
SELECT ST_AsBinary(ST_Point(-158.54, 61.56))
```

ST_AsText(geometry)

Konvertiert die einzelnen angegebenen Geometrie-Datentypen in Text. Gibt einen Wert mit dem Varchar-Datentyp zurück, der eine WKT-Darstellung des Geometrie-Datentyps ist. Beispiel:

```
SELECT ST_AsText(ST_Point(-158.54, 61.56))
```

ST_GeomAsLegacyBinary(geometry)

Gibt eine Legacy-Varbinary aus der angegebenen Geometrie zurück. Beispiel:

```
SELECT ST_GeomAsLegacyBinary(ST_Point(-158.54, 61.56)
```

ST_GeometryFromText(varchar)

Konvertiert Text im WKT-Format in einen Geometriedatentyp. Gibt einen Wert mit dem Geometrie-Datentyp zurück. Beispiel:

```
SELECT ST_GeometryFromText(ST_AsText(ST_Point(1, 2)))
```

ST_GeomFromBinary(varbinary)

Gibt ein Geometrietypobjekt aus einer WKB-Darstellung zurück. Beispiel:

```
SELECT ST_GeomFromBinary(ST_AsBinary(ST_Point(-158.54, 61.56)))
```

ST_GeomFromLegacyBinary(varbinary)

Gibt ein Geometrietypobjekt aus einem Legacy-Varbinary-Typ zurück. Beispiel:

```
SELECT ST_GeomFromLegacyBinary(ST_GeomAsLegacyBinary(ST_Point(-158.54, 61.56)))
```

ST_LineFromText(varchar)

Gibt einen Wert im Geometriedatentyp `line` zurück. Beispiel:

```
SELECT ST_Line('linestring(1 1, 2 2, 3 3)')
```

ST_LineString(array(point))

Gibt einen `LineString`-Geometrietyp zurück, der aus einem Array von Punktgeometrietypen gebildet wird. Wenn das angegebene Array weniger als zwei nicht leere Punkte enthält, wird eine leere `LineString` zurückgegeben. Löst eine Ausnahme aus, wenn ein Element im Array null, leer oder dasselbe wie das vorherige ist. Die zurückgegebene Geometrie ist möglicherweise nicht einfach. Abhängig von der angegebenen Eingabe kann sich die zurückgegebene Geometrie selbst schneiden oder doppelte Scheitelpunkte enthalten. Beispiel:

```
SELECT ST_LineString(ARRAY[ST_Point(-158.54, 61.56), ST_Point(-158.55, 61.56)])
```

ST_MultiPoint(array(point))

Gibt ein `MultiPoint`-Geometrieobjekt zurück, das aus den angegebenen Punkten gebildet wird. Gibt null zurück, wenn das angegebene Array leer ist. Löst eine Ausnahme aus, wenn ein Element im Array null oder leer ist. Die zurückgegebene Geometrie ist möglicherweise nicht einfach und kann doppelte Punkte enthalten, wenn das angegebene Array Duplikate aufweist. Beispiel:

```
SELECT ST_MultiPoint(ARRAY[ST_Point(-158.54, 61.56), ST_Point(-158.55, 61.56)])
```

ST_Point(double, double)

Gibt ein Objekt vom Geometrietyp `point` zurück. Als Eingabedatenwerte für diese Funktion verwenden Sie geometrische Werte, z. B. Werte im kartesischen Universal Transverse Mercator (UTM)-Koordinatensystem oder geografische Karteneinheiten (Längen- und Breitengrad) in Dezimalgraden. Die Längen- und Breitengradwerte verwenden das World Geodetic System, auch als WGS 1984 oder EPSG:4326 bezeichnet. WGS 1984 ist das vom Global Positioning System (GPS) verwendete Koordinatensystem.

In der folgenden Notation werden die Kartenkoordinaten beispielsweise in Längen- und Breitengraden angegeben. Der Wert `.072284`, bei dem es sich um die Pufferdistanz handelt, wird in Winkeleinheiten als Dezimalgrad angegeben:

```
SELECT ST_Buffer(ST_Point(-74.006801, 40.705220), .072284)
```

Syntax:

```
SELECT ST_Point(longitude, latitude) FROM earthquakes LIMIT 1
```

Im nächsten Beispiel werden bestimmte Längen- und Breitenkoordinaten verwendet:

```
SELECT ST_Point(-158.54, 61.56)
FROM earthquakes
LIMIT 1
```

Im nächsten Beispiel werden bestimmte Längen- und Breitenkoordinaten verwendet:

```
SELECT ST_Point(-74.006801, 40.705220)
```

Im folgenden Beispiel wird die `ST_AsText`-Funktion verwendet, um die Geometrie von WKT zu erhalten:

```
SELECT ST_AsText(ST_Point(-74.006801, 40.705220)) AS WKT
```

`ST_Polygon(varchar)`

Unter Verwendung der angegebenen Ordinatenreihenfolge im Uhrzeigersinn von links nach rechts wird ein Geometriedatentyp `polygon` zurückgegeben. Ab Athena-Engine-Version 2 werden nur Polygone als Eingaben akzeptiert. Beispiel:

```
SELECT ST_Polygon('polygon ((1 1, 1 4, 4 4, 4 1))')
```

`to_geometry(sphericalGeography)`

Gibt ein Geometrieobjekt aus dem angegebenen sphärischen Geographieobjekt zurück. Beispiel:

```
SELECT to_geometry(to_spherical_geography(ST_Point(-158.54, 61.56)))
```

`to_spherical_geography(geometry)`

Gibt ein sphärisches Geographieobjekt aus der angegebenen Geometrie zurück. Verwenden Sie diese Funktion, um ein Geometrieobjekt in ein kugelförmiges Geographieobjekt auf der Kugel des

Radius der Erde zu konvertieren. Diese Funktion kann nur für POINT-, MULTIPOINT-, LINESTRING-, MULTILINESTRING-, POLYGON- und MULTIPOLYGON- Geometrien verwendet werden, die im 2D-Raum oder einer GEOMETRYCOLLECTION solcher Geometrien definiert sind. Für jeden Punkt der angegebenen Geometrie überprüft die Funktion, ob `point.x` innerhalb von `[-180.0, 180.0]` und `point.y` innerhalb von `[-90.0, 90.0]` liegt. Die Funktion verwendet diese Punkte als Längen- und Breitengrad, um die Form des `sphericalGeography`-Ergebnisses zu konstruieren.

Beispiel:

```
SELECT to_spherical_geography(ST_Point(-158.54, 61.56))
```

Koordinatenbasierte Verhältnisfunktionen

Die folgenden Funktionen drücken Beziehungen zwischen zwei verschiedenen Geometrien aus, die Sie als Eingabe angeben und Ergebnisse vom Typ `boolean` zurückgeben. Es ist wichtig, in welcher Reihenfolge Sie die Geometrien angeben: Der erste Geometriewert wird als linke Geometrie bezeichnet und der zweite Geometriewert als rechte Geometrie.

Diese Funktionen geben Folgendes zurück:

- TRUE, wenn und nur wenn das von der Funktion beschriebene Verhältnis erfüllt ist.
- FALSE, wenn und nur wenn das von der Funktion beschriebene Verhältnis nicht erfüllt ist.

ST_Contains(geometry, geometry)

Gibt TRUE zurück, wenn und nur wenn die linke Geometrie die rechte Geometrie enthält. Beispiele:

```
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)

Gibt TRUE zurück, wenn und nur wenn die linke Geometrie die rechte Geometrie überschreitet. Beispiel:

```
SELECT ST_Crosses(ST_Line('linestring(1 1, 2 2 )'), ST_Line('linestring(0 1, 2 2)'))
```

ST_Disjoint(geometry, geometry)

Gibt TRUE zurück, wenn und nur wenn die Schnittmenge der linken und rechten Geometrie leer ist. Beispiel:

```
SELECT ST_Disjoint(ST_Line('linestring(0 0, 0 1)'), ST_Line('linestring(1 1, 1 0)'))
```

ST_Equals(geometry, geometry)

Gibt TRUE zurück, wenn und nur wenn die linke Geometrie die rechte Geometrie übereinstimmen. Beispiel:

```
SELECT ST_Equals(ST_Line('linestring( 0 0, 1 1)'), ST_Line('linestring(1 3, 2 2)'))
```

ST_Intersects(geometry, geometry)

Gibt TRUE zurück, wenn und nur wenn die linke Geometrie die rechte Geometrie schneidet. Beispiel:

```
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)

Gibt TRUE zurück, wenn und nur wenn die linke Geometrie sich mit der rechten Geometrie überschneidet. Beispiel:

```
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)

Gibt TRUE zurück, wenn und nur wenn die linke Geometrie das angegebene DE-9IM (Dimensionally Extended nine-Intersection Model)-Verhältnis zu der rechten Geometrie hat. Die dritte (`varchar`) Eingabe übernimmt die Beziehung. Beispiel:

```
SELECT ST_Relate(ST_Line('linestring(0 0, 3 3)'), ST_Line('linestring(1 1, 4 4)'),
 'T********')
```

ST_Touches(geometry, geometry)

Gibt TRUE zurück, wenn und nur wenn die linke Geometrie die rechte Geometrie berührt.

Beispiel:

```
SELECT ST_Touches(ST_Point(8, 8), ST_Polygon('polygon((1  1, 1  4, 4  4, 4 1))'))
```

ST_Within(geometry, geometry)

Gibt TRUE zurück, wenn und nur wenn die linke Geometrie sich innerhalb der rechten Geometrie befindet.

Beispiel:

```
SELECT ST_Within(ST_Point(8, 8), ST_Polygon('polygon((1  1, 1  4, 4  4, 4 1))'))
```

Operationsfunktionen

Verwenden Sie Operationsfunktionen, um Operationen auf den Werten des Geometrie-Datentyps auszuführen. Sie können beispielsweise die Grenzen eines einzelnen Geometrie-Datentyps, Schnittpunkte zwischen zwei Geometrie-Datentypen, Unterschiede zwischen linker und rechter Geometrie (sofern beide Geometrien denselben Datentyp aufweisen) oder einen externen Puffer oder Ring um einen bestimmten Geometrie-Datentyp abrufen.

geometry_union(array(geometry))

Gibt eine Geometrie zurück, die die Punktsatzunion der angegebenen Geometrien darstellt. Beispiel:

```
SELECT geometry_union(ARRAY[ST_Point(-158.54, 61.56), ST_Point(-158.55, 61.56)])
```

ST_Boundary(geometry)

Nimmt als Eingabe einen der Geometriedatentypen und gibt den `boundary`-Geometriedatentyp zurück.

Beispiele:

```
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)

Nimmt als Eingabe einen der geometrischen Datentypen, wie z. B. Punkt, Linie, Polygon, Mehrfachzeile oder Mehrfachpolygon, und als Typ `double` einen Abstand an. Gibt den um den angegebenen Abstand (oder Radius) gepufferten Geometriedatentyp zurück. Beispiel:

```
SELECT ST_Buffer(ST_Point(1, 2), 2.0)
```

Im folgenden Beispiel werden die Kartenkoordinaten in Längen- und Breitengraden angegeben. Der Wert `.072284`, bei dem es sich um die Pufferdistanz handelt, wird in Winkeleinheiten als Dezimalgrad angegeben:

```
SELECT ST_Buffer(ST_Point(-74.006801, 40.705220), .072284)
```

ST_Difference(geometry, geometry)

Gibt eine Geometrie der Differenz zwischen der linken und der rechten Geometrie zurück. Beispiel:

```
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)

Nimmt als Eingabe die Geometriedatentypen `line`, `polygon`, `multiline` und `multipolygon`. Unterstützt den Geometriedatentyp `point` nicht. Gibt die Hülle als Geometrie zurück, wobei eine Hülle ein Rechteck um den angegebenen Geometriedatentyp ist. Beispiele:

```
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)

Gibt ein Array von zwei Punkten zurück, die die untere linke und obere rechte Ecke des umgrenzenden rechteckigen Polygons einer Geometrie darstellen. Gibt null zurück, wenn die angegebene Geometrie leer ist. Beispiel:

```
SELECT ST_EnvelopeAsPts(ST_Point(-158.54, 61.56))
```

ST_ExteriorRing(geometry)

Gibt die Geometrie des äußeren Rings des Eingabetyps `polygon` zurück. Ab Athena-Engine-Version 2 sind Polygone die einzigen Geometrien, die als Eingaben akzeptiert werden. Beispiele:

```
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)

Gibt die Geometrie des Schnittpunkts der linken und rechten Geometrie zurück. Beispiele:

```
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)

Gibt die Geometrie der geometrisch symmetrischen Differenz zwischen der linken Geometrie und der rechten Geometrie zurück. Beispiel:

```
SELECT ST_AsText(ST_SymDifference(ST_Line('linestring(0 2, 2 2)'),
 ST_Line('linestring(1 2, 3 2)')))
```

ST_Union(geometry, geometry)

Gibt einen Geometriedatentyp zurück, der die Punktsatzunion der angegebenen Geometrien darstellt. Beispiel:

```
SELECT ST_Union(ST_Point(-158.54, 61.56),ST_LineString(array[ST_Point(1,2),
 ST_Point(3,4)]))
```

Zugriffsfunktionen

Mithilfe von Zugriffsfunktionen können Sie Werte der Typen `varchar`, `bigint` und `double` von unterschiedlichen `geometry`-Datentypen abrufen, wobei `geometry` einer der von Athena unterstützten Geometrie-Datentypen ist: `point`, `line`, `polygon`, `multiline` und `multipolygon`. Sie können beispielsweise einen Bereich eines `polygon`-Geometrie-Datentyps, die maximalen und minimalen X- und Y-Werte für einen angegebenen Geometrie-Datentyp, die Länge einer `line` oder die Anzahl der Punkte in einem angegebenen Geometrie-Datentyp abrufen.

`geometry_invalid_reason(geometry)`

Gibt in einem varchar-Datentyp den Grund zurück, warum die angegebene Geometrie ungültig oder nicht einfach ist. Wenn die angegebene Geometrie weder gültig noch einfach ist, wird der Grund zurückgegeben, warum sie ungültig ist. Wenn die angegebene Geometrie gültig und einfach ist, wird null zurückgegeben. Beispiel:

```
SELECT geometry_invalid_reason(ST_Point(-158.54, 61.56))
```

`great_circle_distance(latitude1, longitude1, latitude2, longitude2)`

Gibt als Double die Großkreisentfernung zwischen zwei Punkten auf der Erdoberfläche in Kilometern zurück. Beispiel:

```
SELECT great_circle_distance(36.12, -86.67, 33.94, -118.40)
```

`line_locate_point(lineString, point)`

Gibt ein Double zwischen 0 und 1 zurück, das die Position des nächsten Punkts auf der angegebenen Linienfolge zum angegebenen Punkt als Bruchteil der gesamten 2d-Linienlänge darstellt.

Gibt null zurück, wenn die angegebene Zeilenzeichenfolge oder der angegebene Punkt leer oder null ist. Beispiel:

```
SELECT line_locate_point(ST_GeometryFromText('LINESTRING (0 0, 0 1)'), ST_Point(0,
 0.2))
```

simplify_geometry(geometry, double)

Verwendet den Ramer-Douglas-Peucker-Algorithmus, um einen Geometrie-Datentyp zurückzugeben, der eine vereinfachte Version der angegebenen Geometrie ist. Vermeidet das Erstellen abgeleiteter Geometrien (insbesondere Polygone), die ungültig sind. Beispiel:

```
SELECT simplify_geometry(ST_GeometryFromText('POLYGON ((1 0, 2 1, 3 1, 3 1, 4 1, 1
 0))'), 1.5)
```

ST_Area(geometry)

Übernimmt als Eingabe einen Geometrie-Datentyp und gibt einen Bereich vom Typ `double` zurück. Beispiel:

```
SELECT ST_Area(ST_Polygon('polygon((1 1, 4 1, 4 4, 1 4))'))
```

ST_Centroid(geometry)

Nimmt als Eingabe einen Geometriedatentyp `polygon` und gibt einen `point`-Geometriedatentyp zurück, der die Mitte der Polygonhülle ist. Beispiele:

```
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)

Gibt einen Geometriedatentyp zurück, der die kleinste konvexe Geometrie ist, die alle Geometrien in der angegebenen Eingabe umschließt. Beispiel:

```
SELECT ST_ConvexHull(ST_Point(-158.54, 61.56))
```

ST_CoordDim(geometry)

Übernimmt als Eingabe einen der unterstützten Geometrie-Datentypen und gibt die Anzahl der Koordinatenkomponenten mit dem Typ `tinyint` zurück. Beispiel:

```
SELECT ST_CoordDim(ST_Point(1.5,2.5))
```

ST_Dimension(geometry)

Übernimmt als Eingabe einen der unterstützten Geometrie-Datentypen und gibt die räumliche Dimension einer Geometrie mit dem Typ `tinyint` zurück. Beispiel:

```
SELECT ST_Dimension(ST_Polygon('polygon((1 1, 4 1, 4 4, 1 4))'))
```

ST_Distance(geometry, geometry)

Gibt basierend auf der räumlichen Referenz ein Double zurück, das den zweidimensionalen minimalen kartesischen Abstand zwischen zwei Geometrien in projizierten Einheiten enthält. Ab Athena-Engine-Version 2 wird null zurückgegeben, wenn eine der Eingaben eine leere Geometrie ist. Beispiel:

```
SELECT ST_Distance(ST_Point(0.0,0.0), ST_Point(3.0,4.0))
```

ST_Distance(sphericalGeography, sphericalGeography)

Gibt als Double den Abstand zwischen zwei sphärischen Geographiepunkten in Metern zurück. Beispiel:

```
SELECT ST_Distance(to_spherical_geography(ST_Point(61.56,
 -86.67)),to_spherical_geography(ST_Point(61.56, -86.68)))
```

ST_EndPoint(geometry)

Gibt den letzten Punkt eines `line`-Geometriedatentyps in einem `point`-Geometriedatentyp zurück. Beispiel:

```
SELECT ST_EndPoint(ST_Line('linestring(0 2, 2 2)'))
```

ST_Geometries(geometry)

Gibt ein Array mit Geometrien in der angegebenen Sammlung zurück. Wenn es sich bei der angegebenen Geometrie nicht um eine Multi-Geometrie handelt, wird ein Array mit einem Element zurückgegeben. Gibt null zurück, wenn die angegebene Geometrie leer ist.

Bei einem gegebenen `MultiLineString`-Objekt erstellt `ST_Geometries` beispielsweise ein Array von `LineString`-Objekten. Bei einem `GeometryCollection`-Objekt gibt `ST_Geometries` ein nicht abgeflachtes Array seiner Bestandteile zurück. Beispiel:

```
SELECT ST_Geometries(GEOMETRYCOLLECTION(MULTIPOINT(0 0, 1 1),
 GEOMETRYCOLLECTION(MULTILINESTRING((2 2, 3 3)))))
```

Ergebnis:

```
array[MULTIPOINT(0 0, 1 1),GEOMETRYCOLLECTION(MULTILINESTRING((2 2, 3 3)))]
```

ST_GeometryN(geometry, index)

Gibt als Geometriedatentyp das Geometrieelement an einem angegebenen ganzzahligen Index zurück. Indizes beginnen bei 1. Wenn die angegebene Geometrie eine Sammlung von Geometrien ist (z. B. ein `GEOMETRYCOLLECTION`- oder `MULTI*`-Objekt), wird die Geometrie am angegebenen Index zurückgegeben. Wenn der angegebene Index kleiner als 1 oder größer als die Gesamtzahl der Elemente in der Auflistung ist, wird null zurückgegeben. Um die Gesamtanzahl der Elemente zu ermitteln, verwenden Sie `ST_NumGeometries`. Singuläre Geometrien (z. B. `POINT`, `LINESTRING`, oder `POLYGON`), werden als Sammlungen eines Elements behandelt. Leere Geometrien werden als leere Sammlungen behandelt. Beispiel:

```
SELECT ST_GeometryN(ST_Point(-158.54, 61.56),1)
```

ST_GeometryType(geometry)

Gibt den Typ der Geometrie als varchar zurück. Beispiel:

```
SELECT ST_GeometryType(ST_Point(-158.54, 61.56))
```

ST_InteriorRingN(geometry, index)

Gibt das innere Ringelement am angegebenen Index zurück (Indizes beginnen bei 1). Wenn der angegebene Index kleiner als 1 oder größer als die Gesamtanzahl der inneren Ringe in der angegebenen Geometrie ist, wird null zurückgegeben. Löst einen Fehler aus, wenn die angegebene Geometrie kein Polygon ist. Um die Gesamtanzahl der Elemente zu ermitteln, verwenden Sie `ST_NumInteriorRing`. Beispiel:

```
SELECT ST_InteriorRingN(st_polygon('polygon ((0 0, 1 0, 1 1, 0 1, 0 0))'),1)
```

ST_InteriorRings(geometry)

Gibt ein Geometrie-Array aller inneren Ringe zurück, die in der angegebenen Geometrie gefunden werden, oder ein leeres Array, wenn das Polygon keine inneren Ringe hat. Gibt null zurück, wenn die angegebene Geometrie leer ist. Wenn die angegebene Geometrie kein Polygon ist, wird ein Fehler ausgelöst. Beispiel:

```
SELECT ST_InteriorRings(st_polygon('polygon ((0 0, 1 0, 1 1, 0 1, 0 0))'))
```

ST_IsClosed(geometry)

Als Eingabe werden nur die Geometriedatentypen `line` und `multiline` verwendet. Gibt `TRUE` (Typ `boolean`) zurück, wenn und nur wenn die Linie geschlossen ist. Beispiel:

```
SELECT ST_IsClosed(ST_Line('linestring(0 2, 2 2)'))
```

ST_IsEmpty(geometry)

Als Eingabe werden nur die Geometriedatentypen `line` und `multiline` verwendet. Gibt `TRUE` (Typ `boolean`) zurück, wenn und nur wenn die angegebene Geometrie leer ist, d. h., wenn die `line`-Anfangs- und Endwerte übereinstimmen. Beispiel:

```
SELECT ST_IsEmpty(ST_Point(1.5, 2.5))
```

ST_IsRing(geometry)

Gibt `TRUE` (Typ `boolean`) zurück, wenn und nur wenn der `line`-Typ geschlossen und einfach ist. Beispiel:

```
SELECT ST_IsRing(ST_Line('linestring(0 2, 2 2)'))
```

ST_IsSimple(geometry)

Gibt true zurück, wenn die angegebene Geometrie keine anomalen geometrischen Punkte aufweist (z. B. Selbstschnittstellen oder Selbsttangentialität). Um festzustellen, warum die Geometrie nicht einfach ist, verwenden Sie `geometry_invalid_reason()`. Beispiel:

```
SELECT ST_IsSimple(ST_LineString(array[ST_Point(1,2), ST_Point(3,4)]))
```

ST_IsValid(geometry)

Gibt true zurück, wenn und nur wenn die angegebene Geometrie gut geformt ist. Um festzustellen, warum die Geometrie nicht gut geformt ist, verwenden Sie geometry_invalid_reason(). Beispiel:

```
SELECT ST_IsValid(ST_Point(61.56, -86.68))
```

ST_Length(geometry)

Gibt die Länge von `line` mit dem Typ `double` zurück. Beispiel:

```
SELECT ST_Length(ST_Line('linestring(0 2, 2 2)'))
```

ST_NumGeometries(geometry)

Gibt als Ganzzahl die Anzahl der Geometrien in der Sammlung zurück. Wenn die Geometrie eine Sammlung von Geometrien ist (z. B. ein `GEOMETRYCOLLECTION`- oder `MULTI*`-Objekt), wird die Anzahl der Geometrien zurückgegeben. Einzelne Geometrien geben 1 zurück; leere Geometrien geben 0 zurück. Eine leere Geometrie in einem `GEOMETRYCOLLECTION`-Objekt zählt als eine Geometrie. Das folgende Beispiel wird beispielsweise zu 1 ausgewertet:

```
ST_NumGeometries(ST_GeometryFromText('GEOMETRYCOLLECTION(MULTIPOINT EMPTY)'))
```

ST_NumInteriorRing(geometry)

Gibt die Anzahl an inneren Ringen in der `polygon`-Geometrie mit dem Typ `bigint` zurück. Beispiel:

```
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)

Gibt die Anzahl an Punkten in der Geometrie mit dem Typ `bigint` zurück. Beispiel:

```
SELECT ST_NumPoints(ST_Point(1.5, 2.5))
```

ST_PointN(lineString, index)

Gibt als Punktgeometrie-Datentyp den Scheitelpunkt der angegebenen Linienzeichenfolge am angegebenen ganzzahligen Index zurück. Indizes beginnen bei 1. Wenn der angegebene Index

kleiner als 1 oder größer als die Gesamtzahl der Elemente in der Auflistung ist, gibt null zurück. Um die Gesamtanzahl der Elemente zu ermitteln, verwenden Sie ST_NumPoints. Beispiel:

```
SELECT ST_PointN(ST_LineString(array[ST_Point(1,2), ST_Point(3,4)]),1)
```

ST_Points(geometry)

Gibt ein Array von Punkten aus dem angegebenen Linien-Zeichenfolgen-Geometrieobjekt zurück. Beispiel:

```
SELECT ST_Points(ST_LineString(array[ST_Point(1,2), ST_Point(3,4)]))
```

ST_StartPoint(geometry)

Gibt den ersten Punkt eines `line`-Geometriedatentyps in einem `point`-Geometriedatentyp zurück. Beispiel:

```
SELECT ST_StartPoint(ST_Line('linestring(0 2, 2 2)'))
```

ST_X(point)

Gibt die X-Koordinate eines Punkts mit dem Typ `double` zurück. Beispiel:

```
SELECT ST_X(ST_Point(1.5, 2.5))
```

ST_XMax(geometry)

Gibt die maximale X-Koordinate einer Geometrie mit dem Typ `double` zurück. Beispiel:

```
SELECT ST_XMax(ST_Line('linestring(0 2, 2 2)'))
```

ST_XMin(geometry)

Gibt die minimale X-Koordinate einer Geometrie mit dem Typ `double` zurück. Beispiel:

```
SELECT ST_XMin(ST_Line('linestring(0 2, 2 2)'))
```

ST_Y(point)

Gibt die Y-Koordinate eines Punkts mit dem Typ `double` zurück. Beispiel:

```
SELECT ST_Y(ST_Point(1.5, 2.5))
```

ST_YMax(geometry)

Gibt die maximale Y-Koordinate einer Geometrie mit dem Typ `double` zurück. Beispiel:

```
SELECT ST_YMax(ST_Line('linestring(0 2, 2 2)'))
```

ST_YMin(geometry)

Gibt die minimale Y-Koordinate einer Geometrie mit dem Typ `double` zurück. Beispiel:

```
SELECT ST_YMin(ST_Line('linestring(0 2, 2 2)'))
```

Aggregationsfunktionen

convex_hull_agg(geometry)

Gibt die minimale konvexe Geometrie zurück, die alle Geometrien umschließt, die als Eingabe übergeben wurden.

geometry_union_agg(geometry)

Gibt eine Geometrie zurück, die die Punktsatzvereinigung aller Geometrien darstellt, die als Eingabe übergeben werden.

Bing-Kachelfunktionen

Die folgenden Funktionen konvertieren zwischen Geometrien und Kacheln im Microsoft-Bing-Maps-Kachelsystem.

bing_tile(x, y, zoom_level)

Gibt ein Bing-Kachelobjekt aus ganzzahligen Koordinaten `x` und `y` und der angegebenen Zoomstufe zurück. Die Zoomstufe muss eine Ganzzahl zwischen 1 und 23 sein. Beispiel:

```
SELECT bing_tile(10, 20, 12)
```

bing_tile(quadKey)

Gibt ein Bing-Kachelobjekt aus einem Quadkey zurück. Beispiel:

```
SELECT bing_tile(bing_tile_quadkey(bing_tile(10, 20, 12)))
```

bing_tile_at(latitude, longitude, zoom_level)

Gibt ein Bing-Kachelobjekt mit dem angegebenen Breiten-, Längen- und Zoomgrad zurück. Der Breitengrad muss zwischen -85.05112878 und 85.05112878 liegen. Der Längengrad muss zwischen -180 und 180 liegen. Die `latitude`- und `longitude`-Werte müssen `double` und `zoom_level` eine ganze Zahl sein. Beispiel:

```
SELECT bing_tile_at(37.431944, -122.166111, 12)
```

bing_tiles_around(latitude, longitude, zoom_level)

Gibt ein Array von Bing-Kacheln zurück, die den angegebenen Breiten- und Längengradpunkt auf der angegebenen Zoomstufe umgeben. Beispiel:

```
SELECT bing_tiles_around(47.265511, -122.465691, 14)
```

bing_tiles_around(latitude, longitude, zoom_level, radius_in_km)

Gibt auf der angegebenen Zoomstufe ein Array von Bing-Kacheln zurück. Das Array enthält den minimalen Satz von Bing-Kacheln, die einen Kreis mit dem angegebenen Radius in Kilometern um den angegebenen Breiten- und Längengrad abdecken. Die `latitude`-, `longitude`- und `radius_in_km`-Werte sind `double`; die Zoomstufe ist eine `integer`. Beispiel:

```
SELECT bing_tiles_around(37.8475, 112.596667, 10, .5)
```

bing_tile_coordinates(tile)

Gibt die x- und y-Koordinaten der angegebenen Bing-Kachel zurück. Beispiel:

```
SELECT bing_tile_coordinates(bing_tile_at(37.431944, -122.166111, 12))
```

bing_tile_polygon(tile)

Gibt die Polygondarstellung der angegebenen Bing-Kachel zurück. Beispiel:

```
SELECT bing_tile_polygon(bing_tile_at(47.265511, -122.465691, 4))
```

`bing_tile_quadkey(tile)`

Gibt den Quadkey der angegebenen Bing-Kachel zurück. Beispiel:

```
SELECT bing_tile_quadkey(bing_tile(52, 143, 10))
```

`bing_tile_zoom_level(tile)`

Gibt die Zoomstufe der angegebenen Bing-Kachel als Ganzzahl zurück. Beispiel:

```
SELECT bing_tile_zoom_level(bing_tile(52, 143, 10))
```

`geometry_to_bing_tiles(geometry, zoom_level)`

Gibt den Mindestsatz von Bing-Kacheln zurück, der die angegebene Geometrie auf der angegebenen Zoomstufe vollständig abdeckt. Zoomstufen von 1 bis 23 werden unterstützt. Beispiel:

```
SELECT geometry_to_bing_tiles(ST_Point(61.56, 58.54), 10)
```

Namensänderungen und neue Funktionen der Geodaten-Funktion in Athena-Engine-Version 2

In diesem Abschnitt werden Änderungen an Geodatenfunktionen und Geodatenfunktionen aufgeführt, die in der Athena-Engine-Version 2 neu sind. Informationen zu weiteren Änderungen ab Athena-Engine-Version 2 finden Sie unter Athena-Engine-Version 2.

Weitere Informationen zum Athena-Engine-Versioning finden Sie unter Athena-Engine-Versionierung.

Änderungen des Geodaten-Funktionennamens in Athena-Engine-Version 2

Die Namen der folgenden Funktionen haben sich geändert. In einigen Fällen haben sich auch die Eingabe- und Ausgabetypen geändert. Weitere Informationen finden Sie unter den entsprechenden Links.

Vorheriger Funktionsname	Funktionsname ab Athena-Engine-Version 2
st_coordinate_dimension	ST_CoordDim
st_end_point	ST_EndPoint

Vorheriger Funktionsname	Funktionsname ab Athena-Engine-Version 2
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
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

Neue Geodaten-Funktionen in Athena-Engine-Version 2

Die folgenden Geodatenfunktionen sind neu ab Athena-Engine-Version 2. Weitere Informationen finden Sie unter den entsprechenden Links.

Konstruktor-Funktionen

- ST_AsBinary
- ST_GeomAsLegacyBinary
- ST_GeomFromBinary
- ST_GeomFromLegacyBinary
- ST_LineString

- ST_MultiPoint
- to_geometry
- to_spherical_geography

Operationsfunktionen

- geometry_union
- ST_EnvelopeAsPts
- ST_Union

Zugriffsfunktionen

- geometry_invalid_reason
- great_circle_distance
- line_locate_point
- simplify_geometry
- ST_ConvexHull
- st_Distance (sphärische Geographie)
- ST_Geometries
- ST_GeometryN
- ST_GeometryType
- ST_InteriorRingN
- ST_InteriorRings
- ST_IsSimple
- ST_IsValid
- ST_NumGeometries
- ST_PointN
- ST_Points

Aggregationsfunktionen

- convex_hull_agg

- geometry_union_agg

Bing-Kachelfunktionen

- bing_tile
- bing_tile (quadkey)
- bing_tile_at
- bing_tiles_around
- bing_tiles_around (radius)
- bing_tile_coordinates
- bing_tile_polygon
- bing_tile_quadkey
- bing_tile_zoom_level
- geometry_to_bing_tiles

Beispiele: Koordinatenbasierte Abfragen

Die Beispiele in diesem Thema erstellen zwei Tabellen aus Beispieldaten, die in verfügbar sind, GitHub und fragen die Tabellen basierend auf den Daten ab. Die Beispieldaten, die nur zu Illustrationszwecken dienen und nicht garantiert korrekt sind, befinden sich in den folgenden Dateien:

- **`earthquakes.csv`** – Führt Erdbeben auf, die sich in Kalifornien ereignet haben. In der `earthquakes`-Beispieltabelle werden Felder aus diesen Daten verwendet.
- **`california-counties.json`** – Führt Daten auf Landkreisebene für den Bundesstaat Kalifornien im ESRI-gemäßen GeoJSON-Format auf. Die Daten beinhalten zahlreiche Felder wie `AREA`, `PERIMETER`, `STATE`, `COUNTY` und `NAME`, aber die `counties`-Beispieltabelle verwendet nur zwei: `Name` (Zeichenfolge) und `BoundaryShape` (binär).

> **Note**
>
> Athena verwendet die `com.esri.json.hadoop.EnclosedEsriJsonInputFormat`, um die JSON-Daten in das koordinatenbasierte Binärformat zu konvertieren.

Mit dem folgenden Code-Beispiel wird eine Tabelle namens `earthquakes` erstellt:

```
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/';
```

Mit dem folgenden Code-Beispiel wird eine Tabelle namens `counties` erstellt:

```
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/';
```

Die folgende Beispielabfrage verwendet die Funktion `CROSS JOIN` für die `counties`- und `earthquake`-Tabellen. Im Beispiel wird `ST_CONTAINS` verwendet, um nach Landkreisen abzufragen, deren Grenzen Erdbebenstandorte enthalten, die mit `ST_POINT` angegeben sind. Die Abfrage gruppiert solche Landkreise nach Namen, ordnet sie nach Anzahl und gibt sie in absteigender Reihenfolge zurück.

> **Note**
>
> Ab der Athena-Engine-Version 2 unterstützen Funktionen wie `ST_CONTAINS` nicht mehr den `VARBINARY`-Typ als Eingabe. Aus diesem Grund verwendet das Beispiel die `ST_GeomFromLegacyBinary(varbinary)`-Funktion, um den `boundaryshape`-Binärwert

in eine Geometrie umzuwandeln. Weitere Informationen finden Sie unter Änderungen an Geodatenfunktionen in der Athena-Engine-Version 2-Referenz.

```
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
```

Diese Abfrage gibt Folgendes zurück:

```
+------------------------+
| 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   |
+------------------------+
```

Weitere Ressourcen

Weitere Beispiele für koordinatenbasierte Abfragen finden Sie in den folgenden Blogbeiträgen:

- Erweitern Sie Geodatenabfragen in Amazon Athena mit UDFs und AWS Lambda
- Visualisieren Sie die globalen Telefoniedaten von über 200 Jahren mit Amazon Athena und Amazon QuickSight.
- Abfragen OpenStreetMap mit Amazon Athena

Abfragen von &JSON

Mit Amazon Athena können Sie JSON-kodierte Werte analysieren, Daten aus JSON extrahieren, nach Werten suchen und die Länge und Größe von JSON-Arrays bestimmen.

Themen

- Bewährte Methoden zum Lesen von JSON-Daten
- Extrahieren von Daten aus JSON
- Suchen nach Werten in JSON-Arrays
- Abrufen der Länge und Größe von JSON-Arrays
- Fehlerbehebung für JSON-Abfragen

Bewährte Methoden zum Lesen von JSON-Daten

JavaScript Object Notation (JSON) ist eine gängige Methode zur Kodierung von Datenstrukturen als Text. Viele Anwendungen und Tools geben Daten im JSON-Format aus.

In Amazon Athena können Sie Tabellen aus externen Daten erstellen und die JSON-kodierten Daten darin einschließen. Verwenden Sie für solche Arten von Quelldaten Athena zusammen mit JSON-SerDe-Bibliotheken.

Nutzen Sie die folgenden Tipps, um JSON-kodierte Daten zu lesen:

- Wählen Sie den richtigen SerDe aus: einen nativen JSON SerDe, `org.apache.hive.hcatalog.data.JsonSerDe` oder einen OpenX SerDe, `org.openx.data.jsonserde.JsonSerDe`. Weitere Informationen finden Sie unter JSON-SerDe-Bibliotheken.
- Stellen Sie sicher, dass jeder JSON-codierte Datensatz in einer separaten Zeile dargestellt wird, nicht hübsch gedruckt.

> **Note**
>
> Das SerDe erwartet, dass sich jedes JSON-Dokument auf einer einzigen Textzeile befindet, ohne Zeilenabschlusszeichen, die die Felder im Datensatz trennen. Wenn der JSON-Text im hübschen Druckformat vorliegt, erhalten Sie möglicherweise eine Fehlermeldung wie HIVE_CURSOR_ERROR: Zeile ist kein gültiges JSON-Objekt oder HIVE_CURSOR_ERROR: JsonParseException: Unerwartetes Ende der Eingabe: Erwartete Schließmarkierung für OBJEKT, wenn Sie versuchen, die Tabelle abzufragen, nachdem Sie dies erstellt haben. Weitere Informationen finden Sie unter JSON-Datendateiin der OpenX-Serde-Dokumentation auf GitHub.

- Generieren Sie die JSON-kodierten Daten in Spalten (ohne Berücksichtigung von Groß-/Kleinschreibung).
- Binden Sie eine Option zum Ignorieren fehlerhafter Datensätze (wie in diesem Beispiel) ein.

```
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://bucket/path/';
```

- Konvertieren Sie Felder der Quelldaten, die ein unbestimmtes Schema haben, in JSON-Zeichenfolgen in Athena.

Wenn Athena Tabellen basierend auf JSON-Daten erstellt, werden die Daten basierend auf vorhandenen und vordefinierten Schemata analysiert. Es verfügen jedoch möglicherweise nicht alle Ihre Daten über ein vordefiniertes Schema. Um die Schemaverwaltung in solchen Fällen zu

vereinfachen, ist es oft hilfreich, Felder in Quelldaten, die ein unbestimmtes Schema haben, zu JSON-Zeichenfolgen in Athena zu konvertieren und dann JSON-SerDe-Bibliotheken zu verwenden.

Nehmen wir als Beispiel eine IoT-Anwendung, die Ereignisse mit gängigen Feldern von verschiedenen Sensoren veröffentlicht. In einem dieser Felder muss eine benutzerdefinierte Nutzlast gespeichert werden, die für den Sensor, der das Ereignis sendet, eindeutig ist. Da Sie in diesem Fall das Schema nicht kennen, empfehlen wir, die Information als JSON-kodierte Zeichenfolge zu speichern. Konvertieren Sie hierfür die Daten in Ihrer Athena-Tabelle in das JSON-Format, wie im folgenden Beispiel dargestellt. Sie können JSON-kodierte Daten auch in Athena-Datentypen konvertieren.

- Konvertieren von Athena-Datentypen nach JSON
- Konvertieren von JSON-Datentypen nach Athena

Konvertieren von Athena-Datentypen nach JSON

Verwenden Sie `CAST`, um Athena-Datentypen in JSON zu konvertieren.

```
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
```

Diese Abfrage gibt Folgendes zurück:

```
+-----------------------------------------+
| hello_msg      | some_int | some_map      |
+-----------------------------------------+
| "HELLO ATHENA" | 12345    | {"a":1,"b":2} |
+-----------------------------------------+
```

Konvertieren von JSON-Datentypen nach Athena

Verwenden Sie `CAST`, um JSON-Daten in Athena-Datentypen zu konvertieren.

> **Note**
>
> Um in diesem Beispiel Zeichenfolgen als JSON-kodiert zu kennzeichnen, beginnen Sie mit dem `JSON`-Schlüsselwort und verwenden Sie einfache Anführungszeichen wie `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
```

Diese Abfrage gibt Folgendes zurück:

```
+-------------------------------------+
| hello_msg    | some_int | some_map  |
+-------------------------------------+
| HELLO ATHENA | 12345    | {a:1,b:2} |
+-------------------------------------+
```

Extrahieren von Daten aus JSON

Möglicherweise verfügen Sie über Quelldaten mit JSON-kodierten Zeichenfolgen, die Sie nicht in eine Tabelle in Athena deserialisieren möchten. In diesem Fall können Sie mit den JSON-Funktionen in Presto dennoch SQL-Operationen für diese Daten ausführen.

Betrachten Sie diese JSON-Zeichenfolge als Beispiel-Dataset.

```
{"name": "Susan Smith",
"org": "engineering",
"projects":
    [
     {"name":"project1", "completed":false},
     {"name":"project2", "completed":true}
    ]
}
```

Beispiele: Extrahieren von Eigenschaften

Um die Eigenschaften `name` und `projects` aus der JSON-Zeichenfolge zu extrahieren, verwenden Sie die Funktion `json_extract` wie im folgenden Beispiel. Die Funktion `json_extract` übernimmt die Spalte mit der JSON-Zeichenfolge und durchsucht sie mit einem `JSONPath`-artigen Ausdruck in der Punkt-`.`-Notation.

> **Note**
>
> `JSONPath` führt einen einfachen Strukturdurchlauf aus. Dabei wird das Stammverzeichnis des JSON-Dokuments mit $ bezeichnet, gefolgt von einem Punkt und einem direkt unterhalb des Stammverzeichnisses verschachtelten Elements wie `$.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
```

Der zurückgegebene Wert ist eine JSON-kodierte Zeichenfolge und kein nativer Athena-Datentyp.

```
+-------------------------------------------------------------------------------------------------
+
| name           | projects
         |
+-------------------------------------------------------------------------------------------------
+
| "Susan Smith"  | [{"name":"project1","completed":false},
{"name":"project2","completed":true}] |
+-------------------------------------------------------------------------------------------------
+
```

Verwenden Sie zum Extrahieren von skalaren Werten aus der JSON-Zeichenfolge die Funktion `json_extract_scalar`. Sie ist `json_extract` ähnlich, gibt jedoch nur skalare Werte (boolesche Werte, Zahlen oder Zeichenfolgen) zurück.

> **Note**
>
> Verwenden Sie die Funktion `json_extract_scalar` nicht für Arrays, Zuordnungen oder Strukturen.

```
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
```

Diese Abfrage gibt Folgendes zurück:

```
+---------------------------+
| name           | projects |
+---------------------------+
| Susan Smith    |          |
+---------------------------+
```

Um das erste Element der Eigenschaft `projects` in dem Beispiel-Array abzurufen, verwenden Sie die Funktion `json_array_get` und geben Sie die Indexposition an.

```
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
```

Es wird der Wert an der angegebenen Indexposition in dem JSON-kodierten Array zurückgegeben.

```
+---------------------------------------+
| item                                  |
+---------------------------------------+
| {"name":"project1","completed":false} |
+---------------------------------------+
```

Um einen Athena-Zeichenfolgetyp zurückzugeben, verwenden Sie den `[]`-Operator in einem `JSONPath`-Ausdruck und verwenden Sie anschließend die Funktion `json_extract_scalar`. Mehr über `[]` erfahren Sie unter Zugreifen auf Array-Elemente.

```
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
```

Sie erhalten das folgende Ergebnis:

```
+--------------+
| project_name |
+--------------+
| project1     |
+--------------+
```

Suchen nach Werten in JSON-Arrays

Um zu ermitteln, ob ein bestimmter Wert in einem JSON-kodierten Array vorhanden ist, nutzen Sie die `json_array_contains`-Funktion.

Die folgende Abfrage führt die Namen der Benutzer auf, die an "project2" teilnehmen.

```
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')
```

Diese Abfrage gibt eine Liste von Benutzern zurück.

```
+-------------+
| user        |
+-------------+
| Susan Smith |
+-------------+
| Jane Smith  |
+-------------+
```

Die folgende Beispielabfragespiel listet die Namen von Benutzern auf, die Projekte abgeschlossen haben, einschließlich der Anzahl abgeschlossener Projekte. Folgende Aktionen werden durchgeführt:

- Verwendet zum Zwecke der Übersicht `SELECT`-Anweisungen.
- Extrahiert das Projekt-Array.
- Wandelt das Array mittels `CAST` in einen nativen Array von Schlüssel/Wert-Paaren um.
- Extrahiert jedes einzelne Array-Element mit dem `UNNEST`-Operator.
- Filtert abgerufene Werte nach abgeschlossenen Projekten und zählt diese.

> **Note**
>
> Wenn Sie `CAST` zum `MAP` verwenden, können Sie das Schlüsselelement als `VARCHAR` (native Zeichenfolge in Presto) angeben, behalten aber den Wert als JSON bei, da die Werte in `MAP` unterschiedliche Typen aufweisen: Das erste Schlüssel/Wert-Paar ist ein Zeichenfolgenwert und das zweite ein boolescher.

```
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
```

Diese Abfrage gibt das folgende Ergebnis zurück:

```
+----------------------------------+
| name        | completed_projects |
+----------------------------------+
| Susan Smith | 2                  |
+----------------------------------+
| Jane Smith  | 1                  |
+----------------------------------+
```

Abrufen der Länge und Größe von JSON-Arrays

Beispiel: **`json_array_length`**

Um die Länge eines JSON-kodierten Arrays abzurufen, verwenden Sie die Funktion `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
```

Diese Abfrage gibt das folgende Ergebnis zurück:

```
+---------------------+
| name        | count |
+---------------------+
| Susan Smith | 2     |
+---------------------+
| Bob Smith   | 1     |
+---------------------+
| Jane Smith  | 1     |
+---------------------+
```

Beispiel: `json_size`

Um die Größe eines JSON-kodierten Arrays oder Objekts abzurufen, verwenden Sie die Funktion `json_size` und geben die Spalte an, die die JSON-Zeichenfolge und den `JSONPath`-Ausdruck zu dem Array oder Objekt enthält.

```
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
```

Diese Abfrage gibt das folgende Ergebnis zurück:

```
+---------------------+
| name        | count |
+---------------------+
| Susan Smith | 2     |
+---------------------+
| Bob Smith   | 1     |
+---------------------+
| Jane Smith  | 1     |
+---------------------+
```

Fehlerbehebung für JSON-Abfragen

Hilfe zur Fehlerbehebung für JSON-bezogene Abfragen finden Sie in JSON-bezogene Fehler oder in den folgenden Ressourcen:

- Ich erhalte beim Lesen von JSON-Daten in Amazon Athena Fehlermeldungen.
- Wie behebe ich „HIVE_CURSOR_ERROR: Zeile ist kein gültiges JSON-Objekt - JSONException: Duplizierter Schlüssel“ beim Lesen von Dateien aus AWS Config in Athena?

- Die Abfrage SELECT COUNT in Amazon Athena gibt einen einzigen Datensatz zurück, obwohl die JSON-Eingabedatei mehrere Datensätze enthält.
- Wie kann ich die Amazon S3-Quelldatei für eine Zeile in einer Athena-Tabelle anzeigen?

Weitere Informationen finden Sie auch unter Überlegungen und Einschränkungen für SQL-Abfragen in Amazon Athena.

Verwendung von Machine Learning (ML) mit Amazon Athena

Mit Machine Learning (ML) mit Amazon Athena können Sie Athena verwenden, um SQL-Anweisungen zu schreiben, die Machine-Learning (ML)-Inferenz mit Amazon SageMaker ausführen. Dieses Feature vereinfacht den Zugriff auf ML-Modelle zum Zweck von Datenanalysen. Daher müssen keine komplexen Programmiermethoden verwendet werden, um Inferenzen auszuführen.

Um ML mit Athena zu verwenden, definieren Sie eine ML mit Athena Funktion mit der `USING EXTERNAL FUNCTION`-Klausel. Die Funktion zeigt auf den SageMaker-Modellendpunkt, den Sie verwenden sollten, und gibt die Variablennamen und Datentypen an, die an das Modell übergeben werden sollen. Die folgenden Klauseln in der Abfrage verweisen auf die Funktion, um Werte an das Modell zu übergeben. Das Modell führt Inferenzen basierend auf den von der Abfrage übergebenen Werten aus und gibt anschließend Inferenzergebnisse zurück. Weitere Informationen zu SageMaker und zur Funktionsweise von SageMaker-Endpunkten finden Sie im Entwicklerhandbuch von Amazon SageMaker.

Ein Beispiel, das ML mit Athena- und SageMaker-Inferenz verwendet, um einen anomalen Wert in einem Resultset zu erkennen, finden Sie im AWS-Big-Data-Blog-Artikel Erkennen anomaler Werte durch Aufrufen der Inferenzfunktion von Amazon Athena Machine Learning.

Überlegungen und Einschränkungen

- Verfügbare Regionen – Das Athena-ML-Feature wird in der AWS-Regionen angeboten, in der die Athena-Engine-Version 2 oder höher unterstützt wird.
- SageMaker-Modell-Endpunkt muss **`text/csv`** akzeptieren und zurückgeben – Weitere Informationen zu Datenformaten finden Sie unter Häufig verwendete Datenformate für Inferenzen im Amazon-SageMaker-Entwicklerhandbuch.
- Athena sendet keine CSV-Header – Wenn Ihr SageMaker-Endpunkt `text/csv` ist, sollte Ihr Eingabe-Handler nicht davon ausgehen, dass die erste Zeile der Eingabe ein CSV-Header ist. Da Athena keine CSV-Header sendet, enthält die an Athena zurückgegebene Ausgabe eine Zeile weniger als von Athena erwartet, und verursacht einen Fehler.

- SageMaker-Endpunktskalierung – Der SageMaker-Modellendpunkt, auf den verwiesen wird, muss ausreichend skaliert sein, um Athena-Aufrufe an den Endpunkt verarbeiten zu können. Weitere Informationen finden Sie unter SageMaker-Modelle automatisch skalieren im Amazon-SageMaker-Entwicklerhandbuch und CreateEndpointConfig in der Amazon-SageMaker-API-Referenz.
- IAM-Berechtigungen – Um Abfragen auszuführen, die eine ML-Athena-Funktion angeben, muss der IAM-Prinzipal, der die Abfrage ausführt, die Aktion `sagemaker:InvokeEndpoint` für den SageMaker-Modellendpunkt ausführen können, auf den verwiesen wird. Weitere Informationen finden Sie unter Erlauben des Zugriffs für ML mit Athena.
- ML-Athena-Funktionen dürfen in **`GROUP BY`**-Klauseln nicht direkt verwendet werden

ML mit Athena-Syntax

Die Klausel `USING EXTERNAL FUNCTION` gibt eine ML-Athena-Funktion oder mehrere Funktionen an, auf die durch eine nachfolgende `SELECT`-Anweisung in der Abfrage verwiesen werden kann. Sie definieren den Funktionsnamen, die Variablennamen und die Datentypen für die Variablen und Rückgabewerte.

Syntax

Die folgende Syntax zeigt ein e`USING EXTERNAL FUNCTION`-Klausel, die eine ML-Athena-Funktion angibt.

```
USING EXTERNAL FUNCTION ml_function_name (variable1 data_type[, variable2 data_type]
[,...])
RETURNS data_type
SAGEMAKER 'sagemaker_endpoint'
SELECT ml_function_name()
```

Parameter

USING EXTERNAL FUNCTION ***`ml_function_name`*** (***`variable1 data_type`***[, ***`variable2 data_type`***][,...])

> *`ml_function_name`* definiert den Namen der Funktion. Dieser kann in nachfolgenden Abfrageklauseln verwendet werden. Jede *`variable_data_type`* gibt eine benannte Variable und ihren entsprechenden Datentyp an, den das SageMaker-Modell als Eingabe akzeptiert. Der angegebene Datentyp muss ein unterstützter Athena-Datentyp sein.

RETURNS ***data_type***

data_type gibt den SQL-Datentyp an, den *ml_function_name* als Ausgabe aus dem SageMaker-Modell an die Abfrage zurückgibt.

SAGEMAKER '***sagemaker_endpoint***'

sagemaker_endpoint gibt den Endpunkt des SageMaker-Modells an.

SELECT [...] ***ml_function_name***(***expression***) [...]

Die SELECT-Abfrage, die Werte an Funktionsvariablen und das SageMaker-Modell übergibt, um ein Ergebnis zurückzugeben. *ml_function_name* gibt die zuvor in der Abfrage definierte Funktion an, gefolgt von einem *Ausdruck*, der ausgewertet wird, um Werte zu übergeben. Die übergebenen und zurückgegebenen Werte müssen mit den Datentypen übereinstimmen, die in der Klausel `USING EXTERNAL FUNCTION` für die Funktion angegeben sind.

Beispiel

Das folgende Beispiel zeigt eine Abfrage mittels ML mit 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;
```

Beispiele für die Verwendung von Kunden

Die folgenden Videos, die die Vorschauversion von Machine Learning (ML) mit Amazon Athena verwenden, zeigen Möglichkeiten, wie Sie SageMaker mit Athena verwenden können.

Vorhersage der Kundenabwanderung

Das folgende Video zeigt, wie Athena mit den Fähigkeiten des Machine Learnings von Amazon SageMaker kombiniert wird, um die Kundenabwanderung vorherzusagen.

Vorhersagen der Kundenabwanderung mit Amazon Athena und Amazon SageMaker

Erkennen von Bot-Netzen

Das folgende Video zeigt, wie ein Unternehmen Amazon Athena und Amazon SageMaker nutzt, um Bot-Netze zu erkennen.

Erkennen von Bot-Netzen mit Amazon Athena und Amazon SageMaker

Abfragen mit benutzerdefinierten Funktionen (User Defined Functions, UDFs)

Mit benutzerdefinierten Funktionen (UDF) in Amazon Athena können Sie eigene Funktionen zum Verarbeiten von Datensätzen oder Datensatzgruppen erstellen. Eine UDF akzeptiert Parameter, führt Arbeit aus und gibt dann ein Ergebnis zurück.

Um eine UDF in Athena zu verwenden, schreiben Sie eine `USING EXTERNAL FUNCTION`-Klausel vor eine `SELECT`-Anweisung in einer SQL-Abfrage. Die `SELECT`-Anweisung verweist auf die UDF und definiert die Variablen, die beim Ausführen der Abfrage an die UDF übergeben werden. Die SQL-Abfrage ruft eine Lambda-Funktion mit der Java-Laufzeitumgebung auf, wenn sie die UDF aufruft. UDFs werden innerhalb der Lambda-Funktion als Methoden in einem Java-Bereitstellungspaket definiert. Mehrere UDFs können im gleichen Java-Bereitstellungspaket für eine Lambda-Funktion definiert werden. Sie geben auch den Namen der Lambda-Funktion in der `USING EXTERNAL FUNCTION`-Klausel an.

Sie haben zwei Optionen zum Bereitstellen einer Lambda-Funktion für Athena-UDFs. Sie können die Funktion direkt mit Lambda bereitstellen, oder Sie können die AWS Serverless Application Repository verwenden. Um vorhandene Lambda-Funktionen für UDFs zu finden, können Sie das öffentliche AWS Serverless Application Repository oder das private Repository durchsuchen und dann auf Lambda bereitstellen. Sie können auch Java-Quellcode erstellen oder ändern, ihn in eine JAR-Datei verpacken und mit Lambda oder dem AWS Serverless Application Repository bereitstellen. Beispiele für Java-Quellcode und -Pakete für den Einstieg finden Sie unter Erstellen und Bereitstellen einer UDF mit Lambda. Weitere Informationen zu Lambda finden Sie im AWS Lambda-Entwicklerhandbuch. Weitere Informationen über AWS Serverless Application Repository finden Sie im AWS Serverless Application Repository-Entwicklerleitfaden.

Ein Beispiel, das UDFs mit Athena zum Übersetzen und Analysieren von Text verwendet, finden Sie im AWS-Machine-Learning-Blog-Artikel Übersetzen und analysieren Sie Text mit SQL-Funktionen mit Amazon Athena, Amazon Translate und Amazon Comprehend, oder sehen Sie sich video an.

Ein Beispiel für die Verwendung von UDFs zur Erweiterung von Geodatenabfragen in Amazon Athena finden Sie unter Erweitern von Geodatenabfragen in Amazon Athena mit UDFs und AWS Lambda im AWS-Big-Data-Blog.

Überlegungen und Einschränkungen

- Integrierte Athena Funktionen – Integrierte Funktionen in Athena sind so konzipiert, dass sie sehr leistungsfähig sind. Wir empfehlen, wenn möglich integrierte Funktionen anstelle von UDFs zu verwenden. Weitere Hinweise zu integrierten Funktionen finden Sie unter Funktionen in Amazon Athena.
- Nur skalare UDFs – Athena unterstützt nur skalare UDFs, die jeweils eine Zeile verarbeiten und einen einzelnen Spaltenwert zurückgeben. Athena übergibt einen Batch von Zeilen, möglicherweise parallel, an die UDF, wenn Lambda aufgerufen wird. Bedenken Sie beim Entwerfen von UDFs und Abfragen die möglichen Auswirkungen dieser Verarbeitung auf den Netzwerkverkehr.
- UDF-Handler-Funktionen verwenden ein abgekürztes Format — Verwenden Sie das abgekürzte Format (kein Vollformat) für Ihre UDF-Funktionen (z. B. `package.Class` anstelle von`package.Class::method`).
- UDF-Methoden müssen in Kleinbuchstaben sein — UDF-Methoden müssen in Kleinbuchstaben vorliegen; Kamelschreibung ist nicht zulässig.
- UDF-Methoden benötigen Parameter – UDF-Methoden müssen mindestens einen Eingabeparameter haben. Der Versuch, eine ohne Eingabeparameter definierte UDF aufzurufen, führt zu einer Laufzeitausnahme. UDFs sollen Funktionen für Datensätze ausführen, aber eine UDF ohne Argumente nimmt keine Daten auf, sodass eine Ausnahme auftritt.
- Java-Laufzeitunterstützung – Derzeit unterstützen Athena-UDFs die Java 8 und Java 11 Laufzeiten für Lambda. Weitere Informationen finden Sie unter entwickeln von Lambda-Funktionen mit Java im AWS Lambda-Entwicklerhandbuch.
- IAM-Berechtigungen – Um UDF-Abfrageanweisungen in Athena auszuführen und zu erstellen, muss der IAM-Prinzipal, der die Abfrage ausführt, zusätzlich zu den Athena-Funktionen auch Aktionen ausführen dürfen. Weitere Informationen finden Sie unter Beispiel für IAM-Berechtigungsrichtlinien zum Zulassen von benutzerdefinierten Amazon-Athena-Funktionen (UDF).
- Lambda-Kontingente – Lambda Kontingente gelten für UDFs. Weitere Informationen finden Sie unter Lambda quotas (Lambda-Kontingente) im AWS Lambda-Entwicklerhandbuch.
- Filterung auf Zeilenebene – Die Filterung auf Zeilenebene von Lake Formation wird für UDFs nicht unterstützt.

- Ansichten – Sie können keine Ansichten mit UDFs verwenden.
- Bekannte Probleme – Die aktuelle Liste bekannter Probleme finden Sie unter Einschränkungen und Probleme im Abschnitt awslabs/aws-athena-query-federation von GitHub.

Videos

Sehen Sie sich die folgenden Videos an, um mehr über die Verwendung von UDFs in Athena zu erfahren.

Einführung von benutzerdefinierten Funktionen (User Defined Functions, UDFs) in Amazon Athena

Das folgende Video zeigt, wie Sie UDFs in Amazon Athena verwenden können, um vertrauliche Informationen zu überarbeiten.

> **Note**
>
> Die Syntax in diesem Video ist Vorabversion, aber die Konzepte sind die gleichen. Verwenden Sie Athena ohne die `AmazonAthenaPreviewFunctionality`-Arbeitsgruppe.

Einführung von benutzerdefinierten Funktionen (User Defined Functions, UDFs) in Amazon Athena

Video: Übersetzen, analysieren und redigieren Sie Textfelder mithilfe von SQL-Abfragen in Amazon Athena

Das folgende Video zeigt, wie Sie UDFs in Amazon Athena zusammen mit anderen AWS-Services verwenden können, um Text zu übersetzen und zu analysieren.

> **Note**
>
> Die Syntax in diesem Video ist Vorabversion, aber die Konzepte sind die gleichen. Die korrekte Syntax finden Sie im zugehörigen Blogbeitrag Übersetzen, redigieren und analysieren Sie Text mithilfe von SQL-Funktionen mit Amazon Athena, Amazon Translate und Amazon Comprehend im AWS-Machine-Learning-Blog.

Übersetzen, analysieren und redigieren Sie Textfelder mithilfe von SQL-Abfragen in Amazon Athena

UDF-Abfragesyntax

Die `USING EXTERNAL FUNCTION`-Klausel gibt eine UDF oder mehrere UDFs an, auf die durch eine nachfolgende `SELECT`-Anweisung in der Abfrage verwiesen werden kann. Sie benötigen den Methodennamen für die UDF und den Namen der Lambda-Funktion, die die UDF hostet. Anstelle des Lambda-Funktionsnamens können Sie den Lambda-ARN verwenden. In kontenübergreifenden Szenarien ist der Lambda-ARN erforderlich.

Syntax

```
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)] [...]
```

Parameter

USING EXTERNAL FUNCTION ***UDF_name***(***variable1 data_type***[, ***variable2 data_type***][,...])

UDF_name gibt den Namen der UDF an, der einer Java-Methode innerhalb der referenzierten Lambda-Funktion entsprechen muss. Jede *variable_data_type* gibt eine benannte Variable und ihren entsprechenden Datentyp an, den UDF als Eingabe akzeptiert. Der *data_type* muss einer der in der folgenden Tabelle aufgeführten unterstützten Athena-Datentypen sein und dem entsprechenden Java-Datentyp zugeordnet sein.

Athena-Datentyp	Java-Datentyp
TIMESTAMP	java.time.LocalDateTime (UTC)
DATUM	java.time.LocalDate (UTC)
TINYINT	java.lang.Byte
SMALLINT	java.lang.Short
REAL	java.lang.Float

Athena-Datentyp	Java-Datentyp
DOUBLE	java.lang.Double
DECIMALE (siehe RETURNS-Hinweis)	java.math.BigDecimal
BIGINT	java.lang.Long
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 ***data_type***

`data_type` gibt den SQL-Datentyp an, den die UDF als Ausgabe zurückgibt. Athena Datentypen, die in der obigen Tabelle aufgeführt sind, werden unterstützt. Verwenden Sie für den Datentyp `DECIMAL` die Syntax `RETURNS DECIMAL(`*`precision`*`, `*`scale`*`)`, bei der *Präzision* und *Skalierung* Ganzzahlen sind.

LAMBDA '***lambda_funktion***'

my_lambda_function gibt den Namen der Lambda-Funktion an, die bei Ausführung der UDF aufgerufen werden soll.

SELECT [...] ***UDF_name***(***expression***) [...]

Die `SELECT`-Abfrage, die Werte an die UDF übergibt und ein Ergebnis zurückgibt. *UDF_name* gibt die zu verwendende UDF an, gefolgt von einem *Ausdruck*, der ausgewertet wird, um Werte zu übergeben. Werte, die übergeben und zurückgegeben werden, müssen mit den entsprechenden

Datentypen übereinstimmen, die für die UDF in der `USING EXTERNAL FUNCTION`-Klausel angegeben sind.

Beispiele

Beispielabfragen, die auf dem Code AthenaUDFHandler.java auf GitHub basieren, finden Sie auf der GitHub-Seite Amazon-Athena-UDF-Connector.

Erstellen und Bereitstellen einer UDF mit Lambda

Um eine eigene UDF zu erstellen, erstellen Sie eine neue Java-Klasse, indem Sie die `UserDefinedFunctionHandler`-Klasse erweitern. Der Quellcode für das UserDefinedFunctionHandler.java im SDK ist auf GitHub im Repository awslabs/aws-athena-query-federation/athena-federation-sdk verfügbar, zusammen mit Beispiel-UDF-Implementierungen, die Sie untersuchen und ändern können, um eine eigene UDF zu erstellen.

Die Schritte in diesem Abschnitt veranschaulichen das Schreiben und Erstellen einer benutzerdefinierten UDF-JAR-Datei mit Apache Maven über die Befehlszeile und eine Bereitstellung.

Schritte zum Erstellen einer eigenen UDF für die Athena Verwendung von Maven

- Klonen des SDK und Vorbereitung der Entwicklungsumgebung
- Erstellen Ihres Maven-Projekts
- Hinzufügen von Abhängigkeiten und Plugins zu Ihrem Maven-Projekt
- Schreiben von Java-Code für die UDFs
- Erstellen der JAR-Datei
- Bereitstellen des JAR zu AWS Lambda

Klonen des SDK und Vorbereitung der Entwicklungsumgebung

Bevor Sie beginnen, stellen Sie mithilfe von `sudo yum install git -y` sicher, dass git auf Ihrem System installiert ist.

So installieren Sie das AWS-Query Federation SDK

- Geben Sie Folgendes in der Befehlszeile ein, um das SDK-Repository zu klonen. Dieses Repository enthält den SDK, Beispiele und eine Suite von Datenquellen-Connectors. Weitere

Hinweise zu Datenquellen-Connectors finden Sie unter Nutzung von Amazon-Athena-Verbundabfrage.

```
git clone https://github.com/awslabs/aws-athena-query-federation.git
```

So installieren Sie die Voraussetzungen für dieses Verfahren:

Wenn Sie an einer Entwicklungsmaschine arbeiten, auf der bereits Apache Maven, die AWS CLI und das AWS Serverless Application Model-Build-Tool installiert sind, können Sie diesen Schritt überspringen.

1. Führen Sie im `aws-athena-query-federation`-Stammverzeichnis des Verzeichnisses, das Sie beim Klonen erstellt haben, das Skript prepare_dev_env.sh aus, das die Entwicklungsumgebung vorbereitet.
2. Aktualisieren Sie die Shell, um neue Variablen zu erzeugen, die durch den Installationsprozess erstellt wurden, oder starten Sie die Terminalsitzung neu.

```
source ~/.profile
```

> **Important**
>
> Wenn Sie diesen Schritt überspringen, erhalten Sie später Fehler dazu, dass das AWS CLI- oder AWS SAM-Build-Tool Ihre Lambda-Funktion nicht veröffentlichen kann.

Erstellen Ihres Maven-Projekts

Führen Sie den folgenden Befehl aus, um Ihr Maven-Projekt zu erstellen. Ersetzen Sie *`groupId`* durch die eindeutige ID Ihrer Organisation und *`my-athena-udf`* durch den Namen Ihrer Anwendung. Weitere Informationen finden Sie unter Wie erstelle ich mein erstes Maven-Projekt? in der Apache Maven Dokumentation.

```
mvn -B archetype:generate \
-DarchetypeGroupId=org.apache.maven.archetypes \
-DgroupId=groupId \
-DartifactId=my-athena-udfs
```

Hinzufügen von Abhängigkeiten und Plugins zu Ihrem Maven-Projekt

Fügen Sie die folgenden Konfigurationen zu Ihrer Maven-`pom.xml`-Projektdatei hinzu. Ein Beispiel finden Sie in der Datei pom.xml in GitHub.

```
<properties>
    <aws-athena-federation-sdk.version>2021.6.1</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>
```

Schreiben von Java-Code für die UDFs

Erstellen Sie eine neue Klasse, indem Sie UserDefinedFunctionHandler.java erweitern. Schreiben Sie Ihre UDFs in der Klasse.

Im folgenden Beispiel werden zwei Java-Methoden für UDFs, `compress()` und `decompress()`, innerhalb der Klasse `MyUserDefinedFunctions` erstellt.

```
*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);
    }
}
```

Erstellen der JAR-Datei

Führen Sie `mvn clean install` aus um Ihr Projekt zu erstellen. Nachdem es erfolgreich erstellt wurde, wird eine JAR-Datei im `target`-Ordner Ihres Projekts mit dem Namen *`artifactId`*`-`*`version`*`.jar` erstellt, wobei *`artifactId`* der Name ist, den Sie im Maven-Projekt angegeben haben, z. B. `my-athena-udfs`.

Bereitstellen des JAR zu AWS Lambda

Sie haben zwei Möglichkeiten, Ihren Code in Lambda bereitzustellen:

- Bereitstellen mit AWS Serverless Application Repository (empfohlen)
- Erstellen einer Lambda-Funktion aus der JAR-Datei

Option 1: Bereitstellen zu AWS Serverless Application Repository

Wenn Sie Ihre JAR-Datei zu AWS Serverless Application Repository bereitstellen, erstellen Sie eine AWS SAM-YAML-Vorlagendatei, die die Architektur Ihrer Anwendung darstellt. Anschließend geben Sie diese YAML-Datei und einen Amazon-S3-Bucket an, in den Artefakte für Ihre Anwendung hochgeladen und für AWS Serverless Application Repository verfügbar gemacht werden.

Im folgenden Verfahren wird das Skript publish.sh verwendet, das sich im `athena-query-federation/tools`-Verzeichnis des Athena-Query-Federation-SDKs befindet, das Sie zuvor geklont haben.

Weitere Informationen und Anforderungen finden Sie unter Veröffentlichen von Anwendungen im AWS Serverless Application Repository-Entwicklerhandbuch, unter AWS SAM-Vorlagenkonzepte im AWS Serverless Application Model-Entwicklerhandbuch und in Veröffentlichen von Serverless-Anwendungen mit der AWS SAM-CLI.

Das folgende Beispiel veranschaulicht Parameter in einer YAML-Datei. Fügen Sie Ihrer YAML-Datei ähnliche Parameter hinzu und speichern Sie diese in Ihrem Projektverzeichnis. Ein vollständiges Beispiel finden Sie unter athena-udf.yaml in 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
```

Kopieren Sie das `publish.sh`-Skript in das Projektverzeichnis, in dem Sie Ihre YAML-Datei gespeichert haben, und führen Sie den folgenden Befehl aus:

```
./publish.sh MyS3Location MyYamlFile
```

Zum Beispiel, wenn Ihr Bucket-Speicherort `s3://mybucket/mysarapps/athenaudf` ist und Ihre YAML-Datei unter `my-athena-udfs.yaml` gespeichert wurde:

```
./publish.sh mybucket/mysarapps/athenaudf my-athena-udfs
```

So erstellen Sie eine Lambda-Funktion:

1. Öffnen Sie die Lambda-Konsole unter https://console.aws.amazon.com/lambda/, wählen Sie Funktion erstellen und danach Serverless-App-Repository durchsuchen aus
2. Wählen Sie Private applications (Private Anwendungen) aus, suchen Sie Ihre Anwendung in der Liste oder mit Schlüsselwörtern und wählen Sie sie aus.
3. Überprüfen und geben Sie Anwendungsdetails an, und wählen Sie dann Deploy (Bereitstellen).

   Sie können nun die Methodennamen verwenden, die in Ihrer Lambda-Funktion-JAR-Datei als UDFs in Athena definiert sind.

Option 2: Direktes Erstellen einer Lambda-Funktion

Sie können eine Lambda-Funktion auch direkt über die Konsole oder AWS CLI erstellen. Das folgende Beispiel veranschaulicht die Verwendung des Lambda-CLI-Befehls `create-function`.

```
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
```

Abfragen über Regionen hinweg

Athena unterstützt die Möglichkeit, Amazon-S3-Daten in einem AWS-Region abzufragen, das sich von der Region unterscheidet, in der Sie Athena verwenden. Das Abfragen über Regionen hinweg kann eine Option sein, wenn das Verschieben der Daten nicht praktikabel oder zulässig ist oder wenn Sie Daten über mehrere Regionen hinweg abfragen möchten. Selbst wenn Athena in einer bestimmten Region nicht verfügbar ist, können Daten aus dieser Region aus einer anderen Region abgefragt werden, in der Athena verfügbar ist.

Um Daten in einer Region abzufragen, muss Ihr Konto in dieser Region aktiviert sein, auch wenn die Amazon-S3-Daten nicht zu Ihrem Konto gehören. Für einige Regionen wie USA Ost (Ohio) wird Ihr Zugriff auf die Region automatisch aktiviert, wenn Ihr Konto erstellt wird. Andere Regionen erfordern, dass Ihr Konto für die Region „angemeldet“ ist, bevor Sie es verwenden können. Eine Liste der Regionen, für die eine Anmeldung erforderlich ist, finden Sie unter Verfügbare Regionen im Amazon-EC2-Benutzerhandbuch für Linux-Instances. Spezifische Anweisungen zum Opt-in für eine Region finden Sie unter Verwalten von AWS-Regionen in der Allgemeine Amazon Web Services-Referenz.

Überlegungen und Einschränkungen

- Datenzugriffs-Berechtigungen – Um Amazon-S3-Daten von Athena über Regionen hinweg erfolgreich abzufragen, muss Ihr Konto über Berechtigungen zum Lesen der Daten verfügen. Wenn die Daten, die Sie abfragen möchten, zu einem anderen Konto gehören, muss Ihnen das andere Konto Zugriff auf den Amazon-S3-Standort gewähren, der die Daten enthält.
- Gebühren für die Datenübertragung – Für regionsübergreifende Anfragen können Datenübertragungs-Gebühren für Amazon S3 anfallen. Das Ausführen einer Abfrage kann dazu führen, dass mehr Daten übertragen werden als die Größe des Datensatzes. Wir empfehlen Ihnen, Ihre Abfragen zunächst an einer Teilmenge von Daten zu testen und die Kosten in AWS Cost Explorer zu überprüfen.
- AWS Glue – Sie können AWS Glue über Regionen hinweg benutzen. Für regionsübergreifenden AWS Glue-Datenverkehr können zusätzliche Gebühren anfallen. Weitere Informationen finden Sie im AWS Big Data Blog unter Konto- und regionenübergreifende AWS Glue-Verbindungen erstellen.
- Amazon-S3-Verschlüsselungsoptionen – Die SSE-S3- und SSE-KMS-Verschlüsselungsoptionen werden für Abfragen über Regionen hinweg unterstützt; CSE-KMS nicht. Weitere Informationen finden Sie unter Unterstützte Verschlüsselungsoptionen der Amazon S3.

- Verbundabfragen – Die Verwendung von Verbundabfragen über mehrere AWS-Regionen wird nicht unterstützt.

Sofern die oben genannten Bedingungen erfüllt sind, können Sie eine Athena-Tabelle erstellen, die auf den von Ihnen angegebenen `LOCATION`-Wert verweist, und die Daten transparent abfragen. Es ist keine spezielle Syntax erforderlich. Weitere Informationen zum Erstellen von Athena-Tabellen finden Sie unter Erstellen von Tabellen in Athena.

Abfragen von AWS Glue Data Catalog

Da AWS Glue Data Catalog von vielen AWS-Services als zentrales Metadaten-Repository verwendet wird, möchten Sie möglicherweise Datenkatalog-Metadaten abfragen. Dazu können Sie SQL-Abfragen in Athena verwenden. Sie können Athena verwenden, um AWS Glue-Katalog-Metadaten wie Datenbanken, Tabellen, Partitionen und Spalten abzufragen.

Um AWS Glue-Katalog-Metadaten abzurufen, fragen Sie die `information_schema`-Datenbank im Athena-Backend ab. In den Beispielabfragen in diesem Thema wird gezeigt, wie Sie Athena verwenden, um AWS Glue-Katalog-Metadaten für häufige Anwendungsfälle abzufragen.

Themen

Überlegungen und Einschränkungen

- Anstatt die `information_schema`-Datenbank abzufragen, ist es möglich, einzelne Apache-Hive-DDL-Befehle zu verwenden, um Metadateninformationen für bestimmte Datenbanken, Tabellen, Ansichten, Partitionen und Spalten aus Athena zu extrahieren. Die Ausgabe erfolgt jedoch in einem nicht tabellarischen Format.

- Abfragen von `information_schema` sind am leistungsstärksten, wenn Sie eine kleine bis mittelgroße Menge an AWS Glue-Metadaten haben. Wenn Sie eine große Menge an Metadaten haben, können Fehler auftreten.
- Sie können mit `CREATE VIEW` keine Ansicht in der `information_schema`-Datenbank erstellen.

Auflisten von Datenbanken und Durchsuchen einer angegebenen Datenbank

Die Beispiele in diesem Abschnitt zeigen, wie die Datenbanken in Metadaten nach Schema-Namen aufgelistet werden.

Example – Auflisten von Datenbanken

Über die folgende Beispielabfrage werden die Datenbanken aus der `information_schema.schemata` Tabelle aufgelistet.

```
SELECT schema_name
FROM   information_schema.schemata
LIMIT  10;
```

In der folgenden Tabelle werden Beispielergebnisse angezeigt.

6	alb-databas1
7	alb_original_cust
8	alblogsdatabase
9	athena_db_test
10	athena_ddl_db

Example – Durchsuchen einer angegebenen Datenbank

In der folgenden Beispielabfrage ist `rdspostgresql` eine Beispieldatenbank.

```
SELECT schema_name
FROM   information_schema.schemata
WHERE  schema_name = 'rdspostgresql'
```

In der folgenden Tabelle werden Beispielergebnisse angezeigt.

	schema_name
1	rdspostgresql

Auflisten von Tabellen in einer angegebenen Datenbank und Suche nach einer Tabelle anhand des Namens

Um Metadaten für Tabellen aufzulisten, kann die Abfrage anhand des Tabellenschemas oder Tabellennamens erfolgen.

Example – Auflisten von Tabellen nach Schema

Über die folgende Abfrage werden Tabellen aufgelistet, die das `rdspostgresql`-Tabellenschema verwenden.

```
SELECT table_schema,
       table_name,
       table_type
FROM   information_schema.tables
WHERE  table_schema = 'rdspostgresql'
```

In der folgenden Tabelle wird ein Beispielergebnis gezeigt.

	table_schema	table_name	table_type
1	rdspostgresql	rdspostgresqldb1_public_account	BASE TABLE

Example – Suche nach einer Tabelle anhand des Namens

Über die folgende Abfrage werden Metadaten-Informationen für die Tabelle `athena1` abgerufen.

```
SELECT table_schema,
       table_name,
       table_type
```

```
FROM   information_schema.tables
WHERE  table_name = 'athena1'
```

In der folgenden Tabelle wird ein Beispielergebnis gezeigt.

	table_schema	table_name	table_type
1	default	athena1	BASE TABLE

Auflisten von Partitionen für eine bestimmte Tabelle

Sie können `SHOW PARTITIONS` *`table_name`* verwenden, um die Partitionen für eine angegebene Tabelle aufzulisten, wie im folgenden Beispiel.

```
SHOW PARTITIONS cloudtrail_logs_test2
```

Sie können auch eine `$partitions`-Metadatenabfrage verwenden, um die Partitionsnummern und -werte für eine bestimmte Tabelle aufzulisten.

Example – Abfragen der Partitionen für eine Tabelle mit der $partitions-Syntax

Über die folgende Beispielabfrage werden die Partitionen für die Tabelle `cloudtrail_logs_test2` mit der Syntax `$partitions` aufgelistet.

```
SELECT * FROM default."cloudtrail_logs_test2$partitions" ORDER BY partition_number
```

In der folgenden Tabelle werden Beispielergebnisse angezeigt.

	table_catalog	table_schema	table_name	Jahr	Monat	Tag
1	awsdatacatalog	Standard	cloudtrail_logs_test2	2020	08	10
2	awsdatacatalog	Standard	cloudtrail_logs_test2	2020	08	11

	table_catalog	table_schema	table_name	Jahr	Monat	Tag
3	awsdatacatalog	Standard	cloudtrail_logs_test2	2020	08	12

Auflisten aller Spalten für alle Tabellen

Sie können alle Spalten für alle Tabellen in `AwsDataCatalog` oder für alle Tabellen in einer bestimmten Datenbank in `AwsDataCatalog` auflisten.

- Um alle Spalten für alle Datenbanken in `AwsDataCatalog` aufzulisten, verwenden Sie die Abfrage `SELECT * FROM information_schema.columns`.
- Um die Ergebnisse auf eine bestimmte Datenbank zu beschränken, verwenden Sie `table_schema='database_name'` in der `WHERE`-Klausel.

Example – Auflisten aller Spalten für alle Tabellen in einer bestimmten Datenbank

Über die folgende Beispielabfrage werden alle Spalten für alle Tabellen in der Datenbank `webdata` aufgelistet.

```
SELECT * FROM information_schema.columns WHERE table_schema = 'webdata'
```

Auflistung der Spalten, die bestimmte Tabellen gemeinsam haben

Sie können die Spalten auflisten, die bestimmte Tabellen in einer Datenbank gemeinsam haben.

- Verwenden der Syntax `SELECT column_name FROM information_schema.columns`.
- Verwenden Sie für die `WHERE`-Klausel die Syntax `WHERE table_name IN ('table1', 'table2')`.

Example – Listet gemeinsame Spalten für zwei Tabellen in derselben Datenbank auf

Die folgende Beispielabfrage listet die Spalten auf, die die Tabellen `table1` und ich gemeinsam `table2` haben.

```
SELECT column_name
```

```
FROM information_schema.columns
WHERE table_name IN ('table1', 'table2')
GROUP BY column_name
HAVING COUNT(*) > 1;
```

Auflisten oder Durchsuchen von Spalten für eine angegebene Tabelle oder Ansicht

Sie können alle Spalten für eine Tabelle und alle Spalten für eine Ansicht auflisten oder anhand des Namens nach einer Spalte in einer angegebenen Datenbank und Tabelle suchen.

Um die Spalten aufzulisten, verwenden Sie eine `SELECT *`-Abfrage. Geben Sie in der `FROM`-Klausel `information_schema.columns` an. Verwenden Sie in der `WHERE`-Klausel `table_schema='database_name'`, um die Datenbank anzugeben und `table_name = 'table_name'`, um die Tabelle oder Ansicht anzugeben, die die aufzulistenden Spalten enthält.

Example – Auflisten aller Spalten für eine angegebene Tabelle

Über die folgende Beispielabfrage werden alle Spalten für die Tabelle auf `rdspostgresqldb1_public_account` aufgelistet.

```
SELECT *
FROM   information_schema.columns
WHERE  table_schema = 'rdspostgresql'
       AND table_name = 'rdspostgresqldb1_public_account'
```

In der folgenden Tabelle werden Beispielergebnisse angezeigt.

	table_cat alog	table_scl ema	table_nam e	column_ me	ordinal_p osition	column_d fault	is_nulla le	data_t	Komr	extra_inf o
1	awsdataca talog	rdspostg esql	rdspostgr esqldb1_p ublic_acc ount	passwor	1		JA	varcha		
2	awsdataca talog	rdspostg esql	rdspostgr esqldb1_p ublic_acc ount	user_id	2		JA	Ganzz		

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_t	Komm	extra_info
3	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	created_on	3		JA	Zeitstel		
4	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	last_login	4		JA	Zeitstel		
5	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	email	5		JA	varcha		
6	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	usernam	6		JA	varcha		

Example – Auflisten der Spalten für eine angegebene Ansicht

Über die folgende Beispielabfrage werden alle Spalten in der `default`-Datenbank für die Ansicht auf `arrayview` aufgelistet.

```
SELECT *
FROM   information_schema.columns
WHERE  table_schema = 'default'
       AND table_name = 'arrayview'
```

In der folgenden Tabelle werden Beispielergebnisse angezeigt.

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_type	Komm	extra_info
1	awsdataca talog	Standarc	arrayvie	searchdate	1		JA	varchar		
2	awsdataca talog	Standarc	arrayvie	sid	2		JA	varchar		
3	awsdataca talog	Standarc	arrayvie	btid	3		JA	varchar		
4	awsdataca talog	Standarc	arrayvie	p	4		JA	varchar		
5	awsdataca talog	Standarc	arrayvie	infantprice	5		JA	varchar		
6	awsdataca talog	Standarc	arrayvie	sump	6		JA	varchar		
7	awsdataca talog	Standarc	arrayvie	journeymaparray	7		JA	array(varchar)		

Example – Suche nach einer Spalte anhand des Namens in einer angegebenen Datenbank und Tabelle

Über die folgende Beispielabfrage wird nach Metadaten für die `sid`-Spalte in der `arrayview`-Ansicht der `default`-Datenbank gesucht.

```
SELECT *
FROM   information_schema.columns
WHERE  table_schema = 'default'
       AND table_name = 'arrayview'
       AND column_name='sid'
```

In der folgenden Tabelle wird ein Beispielergebnis gezeigt.

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_t	Komm	extra_info
1	awsdatacatalog	Standard	arrayview	sid	2		JA	varcha		

Abfragen von AWS-Service-Protokollen

Dieser Abschnitt enthält mehrere Verfahren zum Verwenden von Amazon Athena für die Abfrage beliebter Datensätze wie AWS CloudTrail-Protokolle, Amazon-CloudFront-Protokolle, Classic-Load-Balancer-Protokolle, Application-Load-Balancer-Protokolle, Amazon-VPC-Flow-Protokolle und Network-Load-Balancer-Protokolle.

Für die Aufgaben in diesem Abschnitt wird die Athena-Konsole eingesetzt. Sie können jedoch auch andere Tools wie den Athena-JDBC-Treiber, die AWS CLI, oder die Amazon-Athena-API-Referenz verwenden.

Informationen zur Verwendung von AWS CloudFormation zum automatischen Erstellen von AWS-Service-Protokolltabellen, -partitionen und Beispielabfragen in Athena finden Sie unter Automatisieren der Erstellung von AWS-Service-Protokolltabellen und deren Abfrage mit Amazon Athena im AWS-Big-Data-Blog. Informationen zur Verwendung einer Python-Bibliothek für AWS Glue zum Erstellen eines gemeinsamen Frameworks für die Verarbeitung von AWS-Service-Protokollen und deren Abfrage in Athena finden Sie unter Einfache Abfrage von AWS-Service-Protokollen mit Amazon Athena.

Die Themen in diesem Abschnitt setzen voraus, dass Sie die entsprechenden Berechtigungen für den Zugriff auf Athena und den Amazon-S3-Bucket konfiguriert haben, in dem sich die abzufragenden Daten befinden sollten. Weitere Informationen finden Sie unter Einrichtung und Erste Schritte.

Themen

- Abfragen von Application-Load-Balancer-Protokollen
- Abfragen von Classic Load Balancer-Protokollen
- CloudFront Amazon-Logs abfragen
- Abfragen von AWS CloudTrail-Protokollen
- Abfragen von Amazon-EMR-Protokollen
- Abfragen von AWS Global Accelerator-Flow-Protokollen

- Abfragen von Amazon-GuardDuty-Ergebnissen
- Abfragen von AWS Network Firewall-Protokollen
- Abfragen von Network-Load-Balancer-Protokollen
- Amazon-Route-53-Resolver-Abfrageprotokolle abfragen
- Abfragen von Amazon-SES-Ereignisprotokollen
- Abfragen von Amazon-VPC-Flow-Protokollen
- Abfragen von AWS WAF Protokollen

Abfragen von Application-Load-Balancer-Protokollen

Ein Application Load Balancer ist eine Lastverteilungsoption für Elastic Load Balancing, die eine Verteilung des Datenverkehrs in einer Microservices-Bereitstellung mit Containern ermöglicht. Durch das Abfragen von Application Load Balancer-Protokollen können Sie die Datenverkehrsquelle, die Latenz und die Bytes anzeigen, die zwischen Elastic Load Balancing-Instances und Backend-Anwendungen übermittelt werden. Weitere Informationen finden Sie unter Zugriffsprotokolle für Ihren Application Load Balancer und Verbindungsprotokolle für Ihren Application Load Balancer im Benutzerhandbuch für Application Load Balancer.

Themen

- Voraussetzungen
- Die Tabelle für ALB-Zugriffsprotokolle erstellen
- Erstellen der Tabelle für ALB-Zugriffsprotokolle in Athena mithilfe der Partitionsprojektion
- Beispielabfragen für ALB-Zugriffsprotokolle
- Die Tabelle für ALB-Verbindungsprotokolle wird erstellt
- Erstellen der Tabelle für ALB-Verbindungsprotokolle in Athena mithilfe der Partitionsprojektion
- Beispielabfragen für ALB-Verbindungsprotokolle
- Weitere Informationen finden Sie auch unter

Voraussetzungen

- Aktivieren Sie die Zugriffs - oder Verbindungsprotokollierung, damit die Application Load Balancer Balancer-Protokolle in Ihrem Amazon S3 S3-Bucket gespeichert werden können.
- Eine Datenbank, die die Tabelle enthält, die Sie für Athena erstellen werden. Um eine Datenbank zu erstellen, können Sie Athena oder die AWS Glue Konsole verwenden. Weitere Informationen

finden Sie unter Erstellen von Datenbanken in Athena in diesem Leitfaden oder unter Arbeiten mit Datenbanken in der AWS -Glue-Konsole im Entwicklerhandbuch für AWS Glue .

Die Tabelle für ALB-Zugriffsprotokolle erstellen

1. Kopieren Sie die folgende `CREATE TABLE`-Anweisung und fügen Sie sie in den Abfrage-Editor der Athena-Konsole ein. Weitere Informationen zu den ersten Schritten mit der Athena-Konsole finden Sie unter Erste Schritte. Ersetzen Sie die Werte in `LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/<ACCOUNT-NUMBER>/elasticloadbalancing/<REGION>/'` mit denen, die Ihrem Amazon-S3-Bucket-Standort entsprechen. Informationen zu den einzelnen Feldern finden Sie unter Zugriffsprotokoll-Einträge im Benutzerhandbuch für Application Load Balancer.

> **Note**
>
> Die folgende `CREATE TABLE`-Anweisung enthält die kürzlich hinzugefügten`classification`- und `classification_reason`-Spalten. Um eine Tabelle für Application Load Balancer-Zugriffsprotokolle zu erstellen, die diese Einträge nicht enthalten, entfernen Sie diese beiden Spalten aus der `CREATE TABLE`-Anweisung und ändern Sie die Regex entsprechend.

```
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
            )
            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>/'
```

2. Führen Sie die Abfrage in der Athena-Konsole aus. Nach Beendigung der Abfrage registriert Athena die `alb_access_logs`-Tabelle, sodass Sie die Daten zum Ausgeben von Abfragen nutzen können.

Erstellen der Tabelle für ALB-Zugriffsprotokolle in Athena mithilfe der Partitionsprojektion

Da ALB-Zugriffsprotokolle eine bekannte Struktur haben, deren Partitionsschema Sie im Voraus angeben können, können Sie die Abfragelaufzeit reduzieren und die Partitionsverwaltung automatisieren, indem Sie die Athena-Partitionsprojektionsfunktion verwenden. Partitionsprojektion fügt automatisch neue Partitionen hinzu, wenn neue Daten hinzugefügt werden. Dadurch entfällt die Notwendigkeit, Partitionen manuell mithilfe von `ALTER TABLE ADD PARTITION` hinzuzufügen.

Die folgende `CREATE TABLE` Beispielanweisung verwendet automatisch die Partitionsprojektion für ALB-Zugriffsprotokolle von einem bestimmten Datum bis heute für eine einzelne Region. AWS Die Anweisung basiert auf dem Beispiel im vorherigen Abschnitt, fügt jedoch `PARTITIONED BY`- und `TBLPROPERTIES`-Klauseln hinzu, um die Partitionsprojektion zu ermöglichen. Ersetzen Sie in den `storage.location.template` Klauseln `LOCATION` und die Platzhalter durch Werte, die den Amazon S3 S3-Bucket-Speicherort Ihrer ALB-Zugriffsprotokolle identifizieren. Ersetzen Sie für `projection.day.range` den *`01.01.2022`* durch das Startdatum, das Sie verwenden möchten. Nach dem erfolgreichen Ausführen der Abfrage können Sie die Tabelle abfragen. Sie müssen `ALTER TABLE ADD PARTITION` nicht ausführen, um die Partitionen zu laden.

```
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
            )
            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}"
            )
```

Weitere Informationen zur Partitionsprojektion finden Sie unter Partitionsprojektion mit Amazon Athena.

Beispielabfragen für ALB-Zugriffsprotokolle

Bei der folgenden Abfrage wird die Anzahl der HTTP GET-Anfragen gezählt und nach Client-IP-Adressen gruppiert, die vom Load Balancer empfangen wurden:

```
SELECT COUNT(request_verb) AS
 count,
 request_verb,
 client_ip
FROM alb_logs
```

```
GROUP BY request_verb, client_ip
LIMIT 100;
```

Eine weitere Abfrage zeigt die URLs, die von Safari-Browser-Benutzern besucht wurden:

```
SELECT request_url
FROM alb_logs
WHERE user_agent LIKE '%Safari%'
LIMIT 10;
```

Die folgende Abfrage zeigt Datensätze mit ELB-Statuscodewerten größer oder gleich 500.

```
SELECT * FROM alb_logs
WHERE elb_status_code >= 500
```

Im folgenden Beispiel wird gezeigt, wie Sie die Protokolle nach `datetime` durchsuchen:

```
SELECT client_ip, sum(received_bytes)
FROM alb_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;
```

Die folgende Abfrage fragt die Tabelle ab, die die Partitionsprojektion für alle ALB-Protokolle vom angegebenen Tag verwendet.

```
SELECT *
FROM alb_logs
WHERE day = '2022/02/12'
```

Die Tabelle für ALB-Verbindungsprotokolle wird erstellt

1. Kopieren Sie die folgende `CREATE TABLE`-Anweisung und fügen Sie sie in den Abfrage-Editor der Athena-Konsole ein. Weitere Informationen zu den ersten Schritten mit der Athena-Konsole finden Sie unter Erste Schritte. Ersetzen Sie die Werte in `LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/<ACCOUNT-NUMBER>/elasticloadbalancing/<REGION>/'` mit denen, die Ihrem Amazon-S3-Bucket-Standort entsprechen. Informationen zu den einzelnen Feldern finden Sie unter Verbindungsprotokolleinträge im Benutzerhandbuch für Application Load Balancers.

```
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
         )
         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>/'
```

2. Führen Sie die Abfrage in der Athena-Konsole aus. Nach Beendigung der Abfrage registriert Athena die `alb_connection_logs`-Tabelle, sodass Sie die Daten zum Ausgeben von Abfragen nutzen können.

Erstellen der Tabelle für ALB-Verbindungsprotokolle in Athena mithilfe der Partitionsprojektion

Da ALB-Verbindungsprotokolle eine bekannte Struktur haben, deren Partitionsschema Sie im Voraus angeben können, können Sie die Abfragelaufzeit reduzieren und die Partitionsverwaltung automatisieren, indem Sie die Athena-Partitionsprojektionsfunktion verwenden. Partitionsprojektion fügt automatisch neue Partitionen hinzu, wenn neue Daten hinzugefügt werden. Dadurch entfällt die Notwendigkeit, Partitionen manuell mithilfe von `ALTER TABLE ADD PARTITION` hinzuzufügen.

Die folgende `CREATE TABLE` Beispielanweisung verwendet automatisch die Partitionsprojektion für ALB-Verbindungsprotokolle von einem bestimmten Datum bis heute für eine einzelne Region. AWS Die Anweisung basiert auf dem Beispiel im vorherigen Abschnitt, fügt jedoch `PARTITIONED BY`- und `TBLPROPERTIES`-Klauseln hinzu, um die Partitionsprojektion zu ermöglichen. Ersetzen Sie in den `storage.location.template` Klauseln `LOCATION` und die Platzhalter durch Werte, die den Amazon S3 S3-Bucket-Speicherort Ihrer ALB-Verbindungsprotokolle identifizieren. Ersetzen Sie

für `projection.day.range` *2023/01/01* durch das Startdatum, das Sie verwenden möchten. Nach dem erfolgreichen Ausführen der Abfrage können Sie die Tabelle abfragen. Sie müssen `ALTER TABLE ADD PARTITION` nicht ausführen, um die Partitionen zu laden.

```
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
         )
            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}"
            )
```

Weitere Informationen zur Partitionsprojektion finden Sie unter Partitionsprojektion mit Amazon Athena.

Beispielabfragen für ALB-Verbindungsprotokolle

Bei der folgenden Abfrage werden Vorkommen gezählt, bei denen der Wert für nicht `tls_verify_status` angegeben wurde`'Success'`, gruppiert nach Client-IP-Adresse:

```
SELECT DISTINCT client_ip, count() AS count FROM alb_connection_logs
WHERE tls_verify_status != 'Success'
GROUP BY client_ip
ORDER BY count() DESC;
```

Die folgende Abfrage sucht nach Ereignissen, bei denen der Wert für im angegebenen Zeitraum mehr als 2 Sekunden `tls_handshake_latency` betrug:

```
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);
```

Weitere Informationen finden Sie auch unter

- Wie analysiere ich meine Zugriffsprotokolle von Application Load Balancer mit Amazon Athena im AWS Knowledge Center.
- Informationen zu den HTTP-Statuscodes in Elastic Load Balancing finden Sie unter Fehlerbehebung bei Ihren Application Load Balancers im Benutzerhandbuch für Application Load Balancer.
- Katalogisieren und analysieren Sie Application Load Balancer Balancer-Protokolle effizienter mit AWS Glue benutzerdefinierten Klassifikatoren und Amazon Athena im AWS Big Data-Blog.

Abfragen von Classic Load Balancer-Protokollen

Mithilfe von Classic-Load-Balancer-Protokollen können Sie Datenverkehrsmuster von und zu Elastic-Load-Balancing-Instances und Backend-Anwendungen analysieren und nachvollziehen. Dabei werden Ihnen die Datenverkehrsquelle, die Latenz und die übertragenen Bytes angezeigt.

Bevor Sie die Elastic-Load-Balancing-Protokolle analysieren, konfigurieren Sie sie für die Speicherung im Amazon-S3-Ziel-Bucket. Weitere Informationen finden Sie unter Aktivieren der Zugriffsprotokolle für Ihren Classic Load Balancer.

- Erstellen der Tabelle für Elastic-Load-Balancing-Protokolle
- Elastic-Load-Balancing-Beispielabfragen

So erstellen Sie die Tabelle für Elastic-Load-Balancing-Protokolle

1. Kopieren Sie die folgende DDL-Anweisung in die Athena-Konsole. Überprüfen Sie die Syntaxder Elastic-Load-Balancing Protokolldatensätze. Möglicherweise müssen Sie die folgende Abfrage aktualisieren, so dass sie die Spalten und die Regex-Syntax für die neueste Version des Datensatzes enthält.

```
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://your_log_bucket/prefix/AWSLogs/AWS_account_ID/
elasticloadbalancing/';
```

2. Ändern Sie den `LOCATION`-Amazon-S3-Bucket, um das Ziel Ihrer Elastic-Load-Balancing-Protokolle anzugeben.
3. Führen Sie die Abfrage in der Athena-Konsole aus. Nach Beendigung der Abfrage wird die Tabelle `elb_logs` von Athena registriert, sodass Sie die darin enthaltenen Daten für Abfragen nutzen können. Weitere Informationen finden Sie unter Elastic-Load-Balancing-Beispielabfragen.

Elastic-Load-Balancing-Beispielabfragen

Verwenden Sie eine ähnliche Abfrage wie im folgenden Beispiel. Die Abfrage listet die Backend-Anwendungsserver auf, die den Fehlercode `4XX` oder `5XX` zurückgegeben haben. Mit dem Operator `LIMIT` lässt sich die Anzahl der im gleichen Schritt abzufragenden Protokolle einschränken.

```
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;
```

Mit einer weiteren Abfrage fassen Sie die Reaktionszeit aller Transaktionen gruppiert nach Backend-IP-Adresse und Elastic-Load-Balancing-Instance-Namen zusammen.

```
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;
```

Weitere Informationen finden Sie unter Analyzing Data in S3 using Athena.

CloudFront Amazon-Logs abfragen

Sie können Amazon CloudFront CDN so konfigurieren, dass Zugriffsprotokolle für die Webdistribution nach Amazon Simple Storage Service exportiert werden. Verwenden Sie diese Protokolle, um das Surfverhalten der Benutzer auf Ihren Websites zu untersuchen, die von CloudFront bereitgestellt werden.

Bevor Sie mit der Abfrage der Protokolle beginnen, aktivieren Sie das Zugriffsprotokoll für Webverteilungen auf Ihrer bevorzugten CloudFront Distribution. Weitere Informationen finden Sie unter Zugriffsprotokolle im Amazon CloudFront Developer Guide. Notieren Sie sich den Amazon S3 S3-Bucket, in dem Sie diese Protokolle speichern.

- Eine Tabelle für CloudFront Standardprotokolle erstellen
- Erstellen einer Tabelle für CloudFront Echtzeitprotokolle
- Beispielabfragen für CloudFront Standardprotokolle

Eine Tabelle für CloudFront Standardprotokolle erstellen

> **Note**
>
> Dieses Verfahren funktioniert für die Anmeldung über den Zugriff auf die Webdistribution CloudFront. Es kann nicht für Streaming-Protokolle von RTMP-Verteilungen eingesetzt werden.

Um eine Tabelle für CloudFront Standard-Protokolldateifelder zu erstellen

1. Kopieren Sie die folgende beispielhafte DDL-Anweisung und fügen Sie sie in den Abfrageeditor der Athena-Konsole ein. Die Beispielanweisung verwendet die Protokolldateifelder, die im Abschnitt Standard-Protokolldateifelder des Amazon CloudFront Developer Guide dokumentiert sind. Ändern Sie `LOCATION` in den Amazon-S3-Bucket, in dem die Protokolle gespeichert werden. Weitere Informationen zur Verwendung des visuellen Abfrage-Editors finden Sie unter Erste Schritte.

   Diese Abfrage spezifiziert `ROW FORMAT DELIMITED` und gibt `FIELDS TERMINATED BY '\t'` an, dass die Felder durch Tabulatorzeichen getrennt sind. Denn `ROW FORMAT DELIMITED` Athena verwendet LazySimpleSerDestandardmäßig das. Die Spalte `date` ist mit Escape-Zeichen

– einfachen umgekehrten Anführungszeichen (`) – versehen, da es sich um ein reserviertes Wort in Athena handelt. Weitere Informationen finden Sie unter Reservierte Schlüsselwörter.

```
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. Führen Sie die Abfrage in der Athena-Konsole aus. Nach Beendigung der Abfrage registriert Athena die `cloudfront_standard_logs`-Tabelle, sodass Sie die Daten zum Ausgeben von Abfragen nutzen können.

Erstellen einer Tabelle für CloudFront Echtzeitprotokolle

Um eine Tabelle für CloudFront Echtzeit-Protokolldateifelder zu erstellen

1. Kopieren Sie die folgende beispielhafte DDL-Anweisung und fügen Sie sie in den Abfrageeditor der Athena-Konsole ein. Die Beispielanweisung verwendet die Protokolldateifelder, die im Abschnitt Echtzeitprotokolle des Amazon CloudFront Developer Guide dokumentiert sind. Ändern Sie `LOCATION` in den Amazon-S3-Bucket, in dem die Protokolle gespeichert werden. Weitere Informationen zur Verwendung des visuellen Abfrage-Editors finden Sie unter Erste Schritte.

   Diese Abfrage spezifiziert `ROW FORMAT DELIMITED` und gibt `FIELDS TERMINATED BY '\t'` an, dass die Felder durch Tabulatorzeichen getrennt sind. Denn `ROW FORMAT DELIMITED` Athena verwendet LazySimpleSerDestandardmäßig das. Die Spalte `timestamp` ist mit Escape-Zeichen – einfachen umgekehrten Anführungszeichen (`) – versehen, da es sich um ein reserviertes Wort in Athena handelt. Weitere Informationen finden Sie unter Reservierte Schlüsselwörter.

   Das folgende Beispiel enthält alle verfügbaren Felder. Sie können Felder, die Sie nicht benötigen, auskommentieren oder entfernen.

```
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. Führen Sie die Abfrage in der Athena-Konsole aus. Nach Beendigung der Abfrage registriert Athena die `cloudfront_real_time_logs`-Tabelle, sodass Sie die Daten zum Ausgeben von Abfragen nutzen können.

Beispielabfragen für CloudFront Standardprotokolle

Die folgende Abfrage summiert die Anzahl der Byte, die CloudFront zwischen dem 9. Juni und dem 11. Juni 2018 bereitgestellt wurden. Setzen Sie den Spaltennamen "date" in doppelte Anführungszeichen, da es sich um ein reserviertes Wort handelt.

```
SELECT SUM(bytes) AS total_bytes
FROM cloudfront_standard_logs
WHERE "date" BETWEEN DATE '2018-06-09' AND DATE '2018-06-11'
LIMIT 100;
```

Um doppelte Zeilen (z. B. doppelte Leerzeilen) aus den Abfrageergebnissen zu entfernen, können Sie, wie im folgenden Beispiel die `SELECT DISTINCT`-Anweisung verwenden.

```
SELECT DISTINCT *
FROM cloudfront_standard_logs
LIMIT 10;
```

Weitere Ressourcen

Weitere Informationen zur Verwendung von Athena zum Abfragen von CloudFront Protokollen finden Sie in den folgenden Beiträgen aus dem AWS Big-Data-Blog.

Einfache Abfrage von AWS-Service Protokollen mit Amazon Athena (29. Mai 2019).

Analysieren Sie Ihre CloudFront Amazon-Zugriffsprotokolle in großem Umfang (21. Dezember 2018).

Erstellen Sie eine serverlose Architektur zur Analyse von CloudFront Amazon-Zugriffsprotokollen mithilfe von AWS Lambda Amazon Athena und Amazon Managed Service für Apache Flink (26. Mai 2017).

Abfragen von AWS CloudTrail-Protokollen

AWS CloudTrail ist ein Service zur Erfassung von AWS-API-Aufrufen und Ereignissen für Amazon-Web-Services-Konten.

CloudTrail -Protokolle enthalten Details zu allen API-Aufrufen an Ihr AWS-Services, einschließlich der console. CloudTrail generates verschlüsselte Protokolldateien und speichert sie in Amazon S3. Weitere Informationen finden Sie im AWS CloudTrail-Benutzerhandbuch.

> **Note**
>
> Wenn Sie SQL-Abfragen für CloudTrail Ereignisinformationen über Konten, Regionen und Datumsangaben hinweg durchführen möchten, sollten Sie CloudTrail Lake verwenden. CloudTrail Lake ist eine AWS Alternative zum Erstellen von Trails, die Informationen aus einem Unternehmen in einem einzigen, durchsuchbaren Ereignisdatenspeicher zusammenfassen. Anstatt Amazon-S3-Bucket-Speicher zu verwenden, speichert es Ereignisse in einem Data Lake, was umfangreichere und schnellere Abfragen ermöglicht. Sie können damit SQL-Abfragen erstellen, die Ereignisse in Organisationen, Regionen und innerhalb benutzerdefinierter Zeitbereiche suchen. Da Sie Lake-Abfragen innerhalb der CloudTrail Konsole selbst ausführen CloudTrail, erfordert CloudTrail Lake keine Athena. Weitere Informationen finden Sie in der CloudTrail Lake-Dokumentation.

Die Verwendung von Athena mit - CloudTrail Protokollen ist eine leistungsstarke Möglichkeit, Ihre Analyse von AWS-Service Aktivitäten zu verbessern. Beispielsweise können Sie mithilfe von Abfragen Trends ermitteln und Vorgänge nach Attributen (z. B. Quell-IP-Adresse oder Benutzer) trennen.

Eine gängige Anwendung besteht darin, - CloudTrail Protokolle zu verwenden, um betriebliche Aktivitäten auf Sicherheit und Compliance zu analysieren. Informationen zu einem detaillierten Beispiel finden Sie im AWS Big-Data-Blogbeitrag unter Analysieren von Sicherheit, Compliance und Betriebsaktivitäten mit AWS CloudTrail und Amazon Athena.

Mithilfe von Athena können Sie diese Protokolldateien direkt aus Amazon S3 abrufen, wenn Sie die `LOCATION` der Protokolldateien angeben. Dafür stehen Ihnen zwei Optionen zur Verfügung:

- Durch das Erstellen von Tabellen für CloudTrail Protokolldateien direkt von der CloudTrail Konsole aus.
- Durch manuelles Erstellen von Tabellen für CloudTrail Protokolldateien in der Athena-Konsole.

Themen

- Erstellen der Tabelle für CloudTrail -Protokolle in Athena mithilfe der Partitionsprojektion
- Abfragen von verschachtelten Feldern
- Beispielabfrage
- Tipps zum Abfragen von CloudTrail Protokollen

Grundlegendes zu CloudTrail Protokollen und Athena-Tabellen

Bevor Sie mit dem Erstellen von Tabellen beginnen, sollten Sie etwas mehr über CloudTrail und darüber erfahren, wie Daten gespeichert werden. Auf diese Weise können Sie die benötigten Tabellen erstellen, unabhängig davon, ob Sie sie über die CloudTrail Konsole oder von Athena erstellen.

CloudTrail speichert Protokolle als JSON-Textdateien im komprimierten gzip-Format (*.json.gzip). Der Speicherort der Protokolldateien hängt von Ihrer Einrichtung der Trails und der zu protokollierenden AWS-Region (oder Regionen) und weiteren Faktoren ab.

Weitere Informationen zum Speicherort der Protokolldateien, der JSON-Struktur und den Datensatzinhalten finden Sie unter folgenden Themen im AWS CloudTrail-Benutzerhandbuch:

- Suchen Ihrer CloudTrail Protokolldateien
- CloudTrail Beispiele für Protokolldateien
- CloudTrail -Inhalte aufzeichnen
- CloudTrail -Ereignisreferenz

Um Protokolle zu sammeln und in Amazon S3 zu speichern, aktivieren CloudTrail Sie über die AWS Management Console. Weitere Informationen finden Sie unter Erstellen eines Trails im Benutzerhandbuch für AWS CloudTrail.

Notieren Sie den Amazon-S3-Ziel-Bucket, in dem die Protokolle gespeichert werden. Ersetzen Sie die `-LOCATIONKlausel` durch den Pfad zum CloudTrail Protokollspeicherort und den Satz von Objekten, mit denen Sie arbeiten möchten. Im Beispiel wird der Wert `LOCATION` für Protokolle eines bestimmten Kontos verwendet, aber Sie können den Grad an Spezifität angeben, der für Ihre Anwendung erforderlich ist.

Beispiel:

- Für die Datenanalyse von mehreren Konten können Sie den Bezeichner `LOCATION` zurücksetzen und dann mit `LOCATION 's3://MyLogFiles/AWSLogs/'` alle `AWSLogs` angeben.

- Zur Analyse der Daten von einem bestimmten Datum, Konto und Region verwenden Sie `LOCATION 's3://MyLogFiles/123456789012/CloudTrail/us-east-1/2016/03/14/'`.

Die oberste Ebene der Objekthierarchie bietet bei Abfragen mit Athena die größte Flexibilität.

Verwenden der CloudTrail Konsole zum Erstellen einer Athena-Tabelle für CloudTrail Protokolle

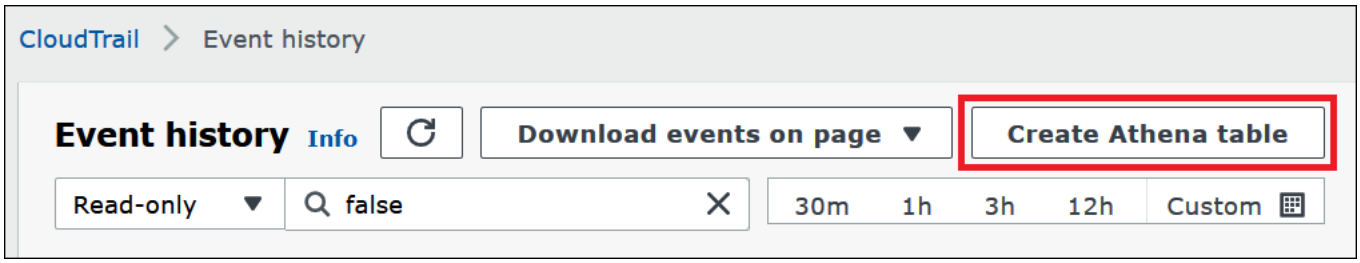
Sie können eine nicht partitionierte Athena-Tabelle zum Abfragen von CloudTrail Protokollen direkt von der CloudTrail Konsole aus erstellen. Das Erstellen einer Athena-Tabelle über die CloudTrail Konsole erfordert, dass Sie mit einer Rolle angemeldet sind, die über ausreichende Berechtigungen zum Erstellen von Tabellen in Athena verfügt.

> **Note**
>
> Sie können die CloudTrail Konsole nicht verwenden, um eine Athena-Tabelle für Organisations-Trail-Protokolle zu erstellen. Erstellen Sie stattdessen die Tabelle manuell mit der Athena-Konsole, damit Sie den richtigen Speicherort angeben können. Weitere Informationen zu Organisation-Trails finden Sie unter Erstellen eines Trails für eine Organisation im AWS CloudTrail-Benutzerhandbuch.

- Informationen zum Festlegen von Athena-Berechtigungen finden Sie unter Einrichtung.
- Weitere Informationen zum Erstellen einer Tabelle mit Partitionen finden Sie unter Erstellen einer Tabelle für CloudTrail Protokolle in Athena mithilfe der manuellen Partitionierung.

So erstellen Sie eine Athena-Tabelle für einen CloudTrail Trail mithilfe der CloudTrail Konsole

1. Öffnen Sie die - CloudTrail Konsole unter https://console.aws.amazon.com/cloudtrail/.
2. Wählen Sie im Navigationsbereich Ereignisverlauf aus.
3. Wählen Sie Create Athena table (Erstellen der Athena-Tabelle).



4. Verwenden Sie für den Speicherort den Abwärtspfeil, um den Amazon-S3-Bucket auszuwählen, in dem Protokolldateien für den abzufragenden Trail gespeichert werden.

   > **Note**
   >
   > Um den Namen des Buckets zu finden, der einem Trail zugeordnet ist, wählen Sie im CloudTrail Navigationsbereich Trails und zeigen Sie die S3-Bucket-Spalte des Trails an. Um die Position des Buckets in Amazon S3 anzuzeigen, wählen Sie den Link für den Bucket in der S3-Bucket-Spalte aus. Dadurch wird die Amazon S3-Konsole zum CloudTrail Bucket-Speicherort geöffnet.

5. Wählen Sie Create table (Tabelle erstellen) aus. Die Tabelle wird mit einem Standardnamen erstellt, in dem der Name des Amazon-S3-Buckets enthalten ist.

Erstellen einer Tabelle für CloudTrail Protokolle in Athena mithilfe der manuellen Partitionierung

Sie können Tabellen für CloudTrail Protokolldateien manuell in der Athena-Konsole erstellen und dann Abfragen in Athena ausführen.

So erstellen Sie eine Athena-Tabelle für einen CloudTrail Trail mit der Athena-Konsole

1. Kopieren Sie die folgende DDL-Anweisung und fügen Sie sie in den Abfrage-Editor der Athena-Konsole ein.

```
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: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://CloudTrail_bucket_name/AWSLogs/Account_ID/CloudTrail/';
```

> **Note**
>
> Wir empfehlen die Verwendung der im Beispiel `org.apache.hive.hcatalog.data.JsonSerDe` gezeigten . Obwohl vorhanden `com.amazon.emr.hive.serde.CloudTrailSerde` ist, werden derzeit einige der neueren CloudTrail Felder nicht verarbeitet.

2. (Optional) Entfernen Sie alle für Ihre Tabelle nicht benötigten Felder. Wenn Sie nur eine bestimmte Gruppe von Spalten lesen müssen, kann Ihre Tabellendefinition die anderen Spalten ausschließen.
3. Ändern Sie `s3://CloudTrail_bucket_name/AWSLogs/Account_ID/CloudTrail/` so, dass auf den Amazon-S3-Bucket gezeigt wird, der Ihre Protokolldaten enthält.
4. Überprüfen Sie, ob die Felder korrekt aufgeführt werden. Weitere Informationen zur vollständigen Liste der Felder in einem CloudTrail Datensatz finden Sie unter CloudTrail Datensatzinhalt .

   Das folgende Beispiel verwendet die Hive JSON SerDe. In diesem Beispiel werden die Felder `additionaleventdata`, `requestparameters` und `responseelements` als Typ `STRING` in der Abfrage aufgelistet, stellen jedoch den Datentyp `STRUCT` dar, der in JSON verwendet wird. Um aus diesen Feldern Daten zu extrahieren, müssen Sie daher `JSON_EXTRACT`-Funktionen verwenden. Weitere Informationen finden Sie unter the section called “Extrahieren von Daten aus JSON”. Zur Leistungsverbesserung werden die Daten in diesem Beispiel nach AWS-Region, Jahr, Monat und Tag partitioniert.
5. Führen Sie die `CREATE TABLE`-Anweisung in der Athena-Konsole aus.
6. Verwenden Sie den Befehl ALTER TABLE ADD PARTITION, um die Partitionen zu laden, sodass Sie sie abfragen können, wie im folgenden Beispiel.

```
ALTER TABLE table_name ADD
   PARTITION (region='us-east-1',
              year='2019',
              month='02',
              day='01')
   LOCATION 's3://cloudtrail_bucket_name/AWSLogs/Account_ID/CloudTrail/us-
east-1/2019/02/01/'
```

Erstellen einer Tabelle für einen organisationsweiten Trail mit manueller Partitionierung

Um eine Tabelle für organisationsweite CloudTrail Protokolldateien in Athena zu erstellen, führen Sie die Schritte unter ausErstellen einer Tabelle für CloudTrail Protokolle in Athena mithilfe der manuellen Partitionierung, nehmen jedoch die im folgenden Verfahren genannten Änderungen vor.

So erstellen Sie eine Athena-Tabelle für organisationsweite CloudTrail Protokolle

1. Ändern Sie in der `CREATE TABLE`-Anweisung die `LOCATION`-Klausel so, dass sie die Organisations-ID enthält, wie im folgenden Beispiel:

   ```
   LOCATION 's3://cloudtrail_bucket_name/AWSLogs/organization_id/Account_ID/
   CloudTrail/'
   ```

2. In der `PARTITIONED BY`-Klausel fügen Sie einen Eintrag für die Konto-ID als Zeichenfolge hinzu, wie im folgenden Beispiel veranschaulicht:

   ```
   PARTITIONED BY (account string, region string, year string, month string, day
    string)
   ```

   Das folgende Beispiel zeigt das Ergebnis beider Aktionen:

   ```
   ...

   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://cloudtrail_bucket_name/AWSLogs/organization_id/Account_ID/
   CloudTrail/'
   ```

3. Fügen Sie in der `ALTER TABLE`-Anweisung die Konto–ID in die `ADD PARTITION`-Klausel ein, wie im folgenden Beispiel veranschaulicht:

   ```
   ALTER TABLE table_name ADD
   PARTITION (account='111122223333',
   region='us-east-1',
   year='2022',
   month='08',
   day='08')
   ```

4. Fügen Sie in der `ALTER TABLE`-Anweisung die Organisations-ID, die Konto-ID und die hinzuzufügende Partition in die `LOCATION`-Klausel ein, wie im folgenden Beispiel veranschaulicht:

```
LOCATION 's3://cloudtrail_bucket_name/AWSLogs/organization_id/Account_ID/
CloudTrail/us-east-1/2022/08/08/'
```

Die folgende `ALTER TABLE`-Beispielanweisung zeigt das Ergebnis beider Aktionen:

```
ALTER TABLE table_name ADD
PARTITION (account='111122223333',
region='us-east-1',
year='2022',
month='08',
day='08')
LOCATION 's3://cloudtrail_bucket_name/AWSLogs/organization_id/111122223333/
CloudTrail/us-east-1/2022/08/08/'
```

Erstellen der Tabelle für CloudTrail -Protokolle in Athena mithilfe der Partitionsprojektion

Da CloudTrail Protokolle über eine bekannte Struktur verfügen, deren Partitionsschema Sie im Voraus angeben können, können Sie die Abfragelaufzeit reduzieren und die Partitionsverwaltung mithilfe der Athena-Partitionsprojektion automatisieren. Partitionsprojektion fügt automatisch neue Partitionen hinzu, wenn neue Daten hinzugefügt werden. Dadurch entfällt die Notwendigkeit, Partitionen manuell mithilfe von `ALTER TABLE ADD PARTITION` hinzuzufügen.

Die folgende `CREATE TABLE` Beispielanweisung verwendet automatisch die Partitionsprojektion für CloudTrail Protokolle von einem bestimmten Datum bis zum aktuellen für ein einzelnes AWS-Region. Ersetzen Sie in den `LOCATION`- und `storage.location.template`-Klauseln die Platzhalter *`Bucket`*, *`Konto-ID`* und *`aws-region`* durch entsprechend identische Werte. Ersetzen Sie für `projection.timestamp.range` den *`01.01.2020`* durch das Startdatum, das Sie verwenden möchten. Nach dem erfolgreichen Ausführen der Abfrage können Sie die Tabelle abfragen. Sie müssen `ALTER TABLE ADD PARTITION` nicht ausführen, um die Partitionen zu laden.

```
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: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://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://bucket/AWSLogs/account-id/CloudTrail/aws-region/
${timestamp}')
```

Weitere Informationen zur Partitionsprojektion finden Sie unter Partitionsprojektion mit Amazon Athena.

Abfragen von verschachtelten Feldern

Da die `userIdentity`- und `resources`-Felder verschachtelte Datentypen sind, erfordert ihre Abfrage eine besondere Behandlung.

Das `userIdentity`-Objekt besteht aus verschachtelten `STRUCT`-Typen. Diese können mithilfe eines Punkts abgefragt werden, um die Felder wie im folgenden Beispiel zu trennen:

```
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
```

Das `resources`-Feld ist ein Array von `STRUCT`-Objekten. Verwenden Sie für diese Arrays `CROSS JOIN UNNEST`, um die Verschachtelung des Arrays aufzuheben, damit Sie seine Objekte abfragen können.

Im folgenden Beispiel werden alle Zeilen zurückgegeben, in denen der Ressourcen-ARN auf `example/datafile.txt` endet. Zur besseren Lesbarkeit entfernt die Ersetzungsfunktion die anfängliche `arn:aws:s3:::`-Teilzeichenfolge aus dem 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
```

Es folgen Beispiele der Ereignisse für `DeleteBucket`-Ereignisse. Die Abfrage extrahiert den Namen des Buckets und die Konto-ID, zu der der Bucket gehört, aus dem `resources`-Objekt.

```
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
```

Weitere Informationen zum Aufheben der Verschachtelung finden Sie unter Filtern von Arrays.

Beispielabfrage

Das folgende Beispiel zeigt einen Teil einer Abfrage, die alle anonymen (unsignierten) Anforderungen aus der Tabelle zurückgibt, die für CloudTrail Ereignisprotokolle erstellt wurde. Diese Abfrage wählt die Anforderungen aus, in denen `useridentity.accountid` anonym ist und `useridentity.arn` nicht angegeben wird:

```
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 ]%';
```

Weitere Informationen finden Sie im AWS-Big-Data-Blogbeitrag unter Analysieren Sie Sicherheit, Compliance und Betriebsaktivitäten mit AWS CloudTrail und Amazon Athena.

Tipps zum Abfragen von CloudTrail Protokollen

Verwenden Sie die folgenden Tipps, um die CloudTrail Protokolldaten zu untersuchen:

- Bevor Sie Abfragen für Protokolle ausführen, sollten Sie sicherstellen, dass Ihre Protokolltabelle ebenso aussieht wie die unter the section called “Erstellen einer Tabelle für CloudTrail Protokolle in Athena mithilfe der manuellen Partitionierung”. Falls dies nicht die erste Tabelle ist, löschen Sie die vorhandene Tabelle mit folgendem Befehl: `DROP TABLE cloudtrail_logs`.
- Nachdem Sie die vorhandene Tabelle gelöscht haben, erstellen Sie diese neu. Weitere Informationen finden Sie unter Erstellen einer Tabelle für CloudTrail Protokolle in Athena mithilfe der manuellen Partitionierung.

  Überprüfen Sie, ob die Felder in der Athena-Abfrage korrekt aufgeführt werden. Informationen zur vollständigen Liste der Felder in einem CloudTrail Datensatz finden Sie unter CloudTrail Datensatzinhalt .

  Falls die Abfrage Felder in JSON-Formaten enthält (z. B. `STRUCT`), extrahieren Sie die Daten aus JSON. Weitere Informationen finden Sie unter Extrahieren von Daten aus JSON.

  Einige Vorschläge für die Ausgabe von Abfragen für Ihre CloudTrail Tabelle:

- Sehen Sie sich zunächst an, welche -Benutzer welche API-Operationen über welche Quell-IP-Adressen aufgerufen haben.
- Verwenden Sie die folgende SQL-Basisabfrage als Vorlage. Fügen Sie die Abfrage in die Athena-Konsole ein und führen Sie sie aus.

```
SELECT
 useridentity.arn,
```

```
 eventname,
 sourceipaddress,
 eventtime
FROM cloudtrail_logs
LIMIT 100;
```

- Ändern Sie die Abfrage, um Ihre Daten weiter zu untersuchen.
- Um die Leistung zu verbessern, binden Sie die Klausel `LIMIT` ein, damit ein bestimmtes Subset von Zeilen zurückgegeben wird.

Abfragen von Amazon-EMR-Protokollen

Amazon EMR und Big-Data-Anwendungen, die auf von Amazon EMR erzeugten Protokolldateien ausgeführt werden. Protokolldateien werden auf den Master-Knoten geschrieben, und Sie können Amazon EMR zudem so konfigurieren, dass Protokolldateien automatisch in Amazon S3 archiviert werden. Sie können mit Amazon Athena diese Protokolle abfragen, um Ereignisse und Trends für Anwendungen und Cluster zu identifizieren. Weitere Informationen zu den Arten von Protokolldateien in Amazon EMR und deren Speicherung in Amazon S3 finden Sie unter Anzeigen von Protokolldateien im Amazon-EMR-Verwaltungshandbuch.

Erstellen und Abfragen einer Basistabelle basierend auf Amazon-EMR-Protokolldateien

Im folgenden Beispiel wird die Basistabelle `myemrlogs` erstellt, die auf Protokolldateien basiert, die in `s3://aws-logs-`*`123456789012`*`-`*`us-west-2`*`/elasticmapreduce/`*`j-2ABCDE34F5GH6`*`/elasticmapreduce/` gespeichert sind. Der in den folgenden Beispielen verwendete Amazon-S3-Speicherort spiegelt das Muster des Standardprotokollspeicherorts für einen EMR-Cluster wider, der durch das Amazon-Web-Services-Konto *`123456789012`* in der Region *`us-west-2`* erstellt wurde. Wenn Sie einen benutzerdefinierten Speicherort verwenden, lautet das Muster s3://*`PathToEMRLogs`*/*`ClusterID`*.

Hinweise zum Erstellen einer partitionierten Tabelle zur potenziellen Verbesserung der Abfrageleistung und zur Verringerung der Datenübertragung finden Sie unter Erstellen und Abfragen einer partitionierten Tabelle basierend auf Amazon-EMR-Protokollen.

```
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'
```

Die folgenden Beispielabfragen können für die im vorherigen Beispiel erstellte `myemrlogs`-Tabelle ausgeführt werden.

Example – Abfrage von step-Protokollen nach Vorkommen von ERROR, WARN, INFO, EXCEPTION, FATAL oder 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 – Abfrage eines bestimmten Instance-Protokolls, i-00b3c0a839ece0a9c, nach ERROR, WARN, INFO, EXCEPTION, FATAL oder 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 – Abfrage von presto-Anwendungsprotokollen nach ERROR, WARN, INFO, EXCEPTION, FATAL oder 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 – Abfrage von Namenode-Anwendungsprotokollen nach ERROR, WARN, INFO, EXCEPTION, FATAL oder 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 – Abfrage aller Protokolle nach Datum und Stunde für ERROR, WARN, INFO, EXCEPTION, FATAL oder 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;
```

Erstellen und Abfragen einer partitionierten Tabelle basierend auf Amazon-EMR-Protokollen

Diese Beispiele verwenden denselben Protokollspeicherort zum Erstellen einer Athena-Tabelle, die Tabelle wird aber partitioniert, und dann wird für jeden Protokollspeicherort eine Partition erstellt. Weitere Informationen finden Sie unter Daten in Athena partitionieren.

Die folgende Abfrage erstellt die partitionierte Tabelle mit dem Namen `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'
```

Die folgenden Abfrageanweisungen erstellen dann Tabellenpartitionen basierend auf Unterverzeichnissen für verschiedene Protokolltypen, die Amazon EMR in Amazon S3 erstellt:

```
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/'
```

Nachdem Sie die Partitionen erstellt haben, können Sie eine `SHOW PARTITIONS`-Abfrage in der Tabelle ausführen, um Folgendes zu bestätigen:

```
SHOW PARTITIONS mypartitionedemrlogs;
```

In den folgenden Beispielen werden Abfragen für bestimmte Protokolleinträge veranschaulicht, die die anhand der obigen Beispiele erstellten Tabelle und Partitionen verwenden.

Example – Abfrage der Protokolle von Anwendung application_1561661818238_0002 in der Container-Partition nach ERROR oder 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 – Abfrage der hadoop-Mapreduce-Partition nach Auftrag job_1561661818238_0004 und fehlgeschlagene Reduzierungen

```
SELECT data,
        "$PATH"
FROM "default"."mypartitionedemrlogs"
WHERE logtype='hadoop-mapreduce'
        AND regexp_like(data,'job_1561661818238_0004|Failed Reduces') limit 100;
```

Example – Abfrage von Hive-Protokollen in der Knotenpartition nach Abfrage-ID 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 – Abfrage von resourcemanager-Protokollen in der Knotenpartition für die Anwendung 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
```

Abfragen von AWS Global Accelerator-Flow-Protokollen

Sie können mit AWS Global Accelerator Accelerators erstellen, die den Netzwerkverkehr über das globale AWS-Netzwerk an optimale Endpunkte leiten. Weitere Informationen zu Global Accelerator finden Sie unter Was ist AWS Global Accelerator?

Mit Global-Accelerator-Flow-Protokollen können Sie Informationen zum Datenverkehr über die IP-Adresse zu und von Netzwerkschnittstellen in Ihren Accelerators erfassen. Flow-Protokolldaten

werden in Amazon S3 veröffentlicht, wo Sie Ihre Daten abrufen und anzeigen können. Weitere Informationen finden Sie unter Flow-Protokolle in AWS Global Accelerator.

Sie können Athena verwenden, um Ihre Global-Accelerator-Flow-Protokolle abzufragen, indem Sie eine Tabelle erstellen, die ihren Speicherort in Amazon S3 angibt.

So erstellen Sie die Tabelle für Global-Accelerator-Flow-Protokolle

1. Kopieren Sie die folgende DDL-Anweisung in die Athena-Konsole. Diese Abfrage gibt ROW FORMAT DELIMITED an und lässt die Angabe eines SerDe weg, was bedeutet, dass die Abfrage `LazySimpleSerDe` verwendet. Bei dieser Abfrage werden Felder durch ein Leerzeichen beendet.

```
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://your_log_bucket/prefix/AWSLogs/account_id/globalaccelerator/region/'
TBLPROPERTIES ("skip.header.line.count"="1");
```

2. Ändern Sie den `LOCATION`-Wert so, dass auf den Amazon-S3-Bucket verwiesen wird, der Ihre Protokolldaten enthält.

```
's3://your_log_bucket/prefix/AWSLogs/account_id/globalaccelerator/region_code/'
```

3. Führen Sie die Abfrage in der Athena-Konsole aus. Nach Beendigung der Abfrage wird die Tabelle `aga_flow_logs` von Athena registriert, so dass Sie die darin enthaltenen Daten für Abfragen nutzen können.
4. Erstellen Sie Partitionen zum Lesen der Daten, wie in der folgenden Beispielabfrage. Diese Abfrage erstellt eine einfache Partition für ein angegebenes Datum. Ersetzen Sie die Platzhalter für Datum und Standort.

```
ALTER TABLE aga_flow_logs
ADD PARTITION (dt='YYYY-MM-dd')
LOCATION 's3://your_log_bucket/prefix/AWSLogs/account_id/
globalaccelerator/region_code/YYYY/MM/dd';
```

Beispiel-Abfragen für AWS Global Accelerator-Flow-Protokolle

Example – Liste der Anforderungen, die eine bestimmte Edge-Position durchlaufen

In der folgenden Beispielabfrage werden Anforderungen aufgeführt, die durch die LHR-Edgeposition übergeben wurden. Mit dem Operator `LIMIT` lässt sich die Anzahl der im gleichen Schritt abzufragenden Protokolle einschränken.

```
SELECT
  clientip,
  agaregion,
  protocol,
  action
FROM
  aga_flow_logs
WHERE
  agaregion LIKE 'LHR%'
LIMIT
  100;
```

Example – Liste der Endpunkt-IP-Adressen, die die meisten HTTPS-Anforderungen empfangen

Verwenden Sie die folgende Abfrage, um zu sehen, welche Endpunkt-IP-Adressen die höchste Anzahl von HTTPS-Anforderungen empfangen. Diese Abfrage zählt die Anzahl der vom HTTPS-Port 443 empfangenen Pakete, gruppiert sie nach IP-Zieladresse und gibt die Top 10 IP-Adressen aus.

```
SELECT
  SUM(numpackets) AS packetcount,
  endpointip
FROM
  aga_flow_logs
WHERE
  endpointport = 443
GROUP BY
  endpointip
ORDER BY
  packetcount DESC
LIMIT
  10;
```

Abfragen von Amazon-GuardDuty-Ergebnissen

Amazon GuardDuty ist ein Sicherheitsüberwachungsservice, mit dem Sie unerwartete und potenziell unbefugte oder bösartige Aktivitäten in Ihrer AWS-Umgebung erkennen können. Bei Erkennung von unerwarteten und potenziell bösartigen Aktivitäten generiert GuardDuty Sicherheitsergebnisse, die Sie zur Speicherung und Analyse in Amazon S3 exportieren können. Nachdem Sie Ihre Ergebnisse in Amazon S3 exportiert haben, können Sie diese mit Athena abfragen. Dieser Artikel zeigt, wie Sie eine Tabelle in Athena für Ihre GuardDuty-Ergebnisse erstellen und diese abfragen.

Weitere Informationen über Amazon GuardDuty finden Sie im Amazon-GuardDuty-Benutzerhandbuch.

Voraussetzungen

- Aktivieren Sie die GuardDuty-Funktion zum Exportieren von Ergebnissen in Amazon S3. Schritte dazu finden Sie unter Ergebnisse exportieren im Amazon-GuardDuty-Benutzerhandbuch.

Erstellen einer Tabelle in Athena für GuardDuty-Ergebnisse

Um Ihre GuardDuty-Ergebnisse aus Athena abzufragen, müssen Sie eine Tabelle dafür erstellen.

So erstellen Sie eine Tabelle in Athena für GuardDuty-Ergebnisse

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Fügen Sie die folgende DDL-Anweisung in die Athena-Konsole ein. Ändern Sie die Werte in `LOCATION 's3://findings-bucket-name/AWSLogs/account-id/GuardDuty/'`, um in Amazon S3 auf Ihre GuardDuty-Ergebnisse zu zeigen.

```
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://findings-bucket-name/AWSLogs/account-id/GuardDuty/'
 TBLPROPERTIES ('has_encrypted_data'='true')
```

> **Note**
>
> Das SerDe erwartet, dass sich jedes JSON-Dokument auf einer einzigen Textzeile befindet, ohne Zeilenabschlusszeichen, die die Felder im Datensatz trennen. Wenn der JSON-Text im hübschen Druckformat vorliegt, erhalten Sie möglicherweise eine Fehlermeldung wie HIVE_CURSOR_ERROR: Zeile ist kein gültiges JSON-Objekt oder HIVE_CURSOR_ERROR: JsonParseException: Unerwartetes Ende der Eingabe: Erwartete Schließmarkierung für OBJEKT, wenn Sie versuchen, die Tabelle abzufragen, nachdem Sie dies erstellt haben. Weitere Informationen finden Sie unter JSON-Datendatei in der OpenX-SerDe-Dokumentation auf GitHub.

3. Führen Sie die Abfrage in der Athena-Konsole aus, um die `gd_logs`-Tabelle zu registrieren. Wenn die Abfrage abgeschlossen ist, können Sie die Ergebnisse aus Athena abfragen.

Beispielabfragen

In den folgenden Beispielen wird gezeigt, wie GuardDuty-Ergebnisse aus Athena abgefragt werden.

Example – DNS-Datenexfiltration

Die folgende Abfrage gibt Informationen zu Amazon-EC2-Instances zurück, die Daten möglicherweise über DNS-Abfragen exfiltrieren.

```
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 – Unberechtigter IAM-Benutzerzugriff

Die folgende Abfrage gibt alle `UnauthorizedAccess:IAMUser`-Suchtypen für einen IAM-Prinzipal aus allen Regionen zurück.

```
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;
```

Tipps zum Abfragen von GuardDuty Ergebnissen

Beachten Sie beim Erstellen der Abfrage die folgenden Punkte.

- Um Daten aus verschachtelten JSON-Feldern zu extrahieren, verwenden Sie die `json_extract`- oder `json_extract_scalar`-Funktionen von Presto. Weitere Informationen finden Sie unter Extrahieren von Daten aus JSON.
- Stellen Sie sicher, dass alle Zeichen in den JSON-Feldern Kleinbuchstaben sind.
- Hinweise zum Herunterladen von Abfrageergebnissen finden Sie unter Herunterladen von Abfrageergebnisdateien mithilfe der Athena-Konsole.

Abfragen von AWS Network Firewall-Protokollen

AWS Network Firewall ist ein verwalteter Service, mit dem Sie wesentliche Netzwerkschutzmaßnahmen für Ihre Amazon-Virtual-Private-Cloud-Instances bereitstellen können. AWS Network Firewall arbeitet mit AWS Firewall Manager zusammen, sodass Sie Richtlinien basierend auf AWS Network Firewall-Regeln erstellen und diese Richtlinien dann zentral auf Ihre VPCs und Konten anwenden können. Mehr über AWS Network Firewall erfahren Sie unter AWS Network Firewall.

Sie können die AWS Network Firewall-Protokollierung für Datenverkehr konfigurieren, den Sie an die zustandsbehaftete Regel-Engine Ihrer Firewall weiterleiten. Mit der Protokollierung erhalten Sie detaillierte Informationen zum Netzwerkverkehr, einschließlich der Zeit, zu der die Stateful-Engine ein Paket empfangen hat, detaillierte Informationen über das Paket und alle Stateful-Regelaktionen, die gegen das Paket ausgeführt werden. Die Protokolle werden in dem von Ihnen konfigurierten Protokollziel veröffentlicht, wo Sie sie abrufen und anzeigen können. Weitere Informationen finden Sie unter Protokollieren des Netzwerkverkehrs von AWS Network Firewall im AWS Network Firewall-Entwicklerhandbuch.

So erstellen Sie die Tabelle für Network-Firewall-Protokolle

1. Kopieren Sie die folgende DDL-Anweisung und fügen Sie sie in den Athena-Abfrage-Editor ein: Möglicherweise müssen Sie die Anweisung so aktualisieren, dass sie die Spalten für die neueste

Version der Protokolle enthält. Weitere Informationen finden Sie unter Inhalt eines Firewall-Protokolls im AWS Network Firewall-Entwicklerhandbuch.

```
CREATE EXTERNAL TABLE anf_logs(
  firewall_name string,
  availability_zone string,
  event_timestamp bigint,
  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
    >
  >
)
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://bucket_name/AWSLogs/'
```

2. Ändern Sie `LOCATION` für den Amazon-S3-Bucket und geben Sie den Zielspeicherort der Network-Firewall-Protokolle an.
3. Führen Sie die Abfrage in der Athena-Konsole aus. Nach Beendigung der Abfrage wird die Tabelle `anf_logs` von Athena registriert, sodass Sie die darin enthaltenen Daten für Abfragen nutzen können.

Beispielabfrage für Network Firewall

Die folgende Abfrage zeigt die Anzahl der von AWS Network Firewall ausgewerteten Anforderungen mit eindeutigen Quell- und Ziel-IP-Paaren.

```
SELECT COUNT(*) AS count,
       event.src_ip,
       event.src_port,
       event.dest_ip,
       event.dest_port
FROM anf_logs
GROUP BY event.src_ip,
         event.src_port,
         event.dest_ip,
         event.dest_port
ORDER BY COUNT DESC
LIMIT 100
```

Abfragen von Network-Load-Balancer-Protokollen

Verwenden Sie Athena, um Protokolle aus dem Network Load Balancer zu analysieren und zu verarbeiten. Diese Protokolle erhalten detaillierte Informationen über die Transport-Layer-Security(TLS)-Anfragen, die an Network Load Balancer gesendet werden. Sie können diese Zugriffsprotokolle für die Analyse von Datenverkehrsmustern und zur Problembehebung verwenden.

Bevor Sie die Network-Load-Balancer-Zugriffsprotokolle analysieren, aktivieren und konfigurieren Sie sie für die Speicherung im Amazon-S3-Ziel-Bucket. Weitere Informationen und Informationen zu den einzelnen Network-Load-Balancer-Zugriffsprotokolleinträgen finden Sie unter Zugriffsprotokolle für Ihren Network Load Balancer.

- Erstellen der Tabelle für die Protokolle des Network Load Balancers
- Beispielabfragen für Network Load Balancer

Erstellen der Tabelle für die Protokolle des Network Load Balancers

1. Kopieren Sie die folgende DDL-Anweisung in die Athena-Konsole. Überprüfen Sie die Syntax der Protokolleinträge des Network Load Balancer. Möglicherweise müssen Sie die folgende Abfrage aktualisieren, so dass sie die Spalten und die Regex-Syntax für die neueste Version des Datensatzes enthält.

```
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://your_log_bucket/prefix/AWSLogs/AWS_account_ID/
elasticloadbalancing/region';
```

2. Ändern Sie den `LOCATION`-Amazon-S3-Bucket, um das Ziel Ihrer Network-Load-Balancer-Protokolle anzugeben.
3. Führen Sie die Abfrage in der Athena-Konsole aus. Nach Beendigung der Abfrage wird die Tabelle `nlb_tls_logs` von Athena registriert, sodass Sie die darin enthaltenen Daten für Abfragen nutzen können.

Beispielabfragen für Network Load Balancer

Wenn Sie wissen möchten, wie oft ein Zertifikat verwendet wird, verwenden Sie eine Abfrage ähnlich wie in diesem Beispiel:

```
SELECT count(*) AS
         ct,
         cert_arn
FROM "nlb_tls_logs"
GROUP BY  cert_arn;
```

Die folgende Abfrage zeigt, wie viele Benutzer eine TLS-Version vor 1.3 verwenden:

```
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;
```

Verwenden Sie die folgende Abfrage, um Verbindungen zu identifizieren, die eine lange TLS-Handshake-Zeit benötigen:

```
SELECT *
FROM "nlb_tls_logs"
ORDER BY  tls_handshake_time_ms DESC
LIMIT 10;
```

Verwenden Sie die folgende Abfrage, um zu ermitteln und zu zählen, welche TLS-Protokollversionen und Verschlüsselungssammlungen in den letzten 30 Tagen ausgehandelt wurden.

```
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;
```

Amazon-Route-53-Resolver-Abfrageprotokolle abfragen

Sie können Athena-Tabellen für Ihre Amazon-Route-53-Resolver-Abfrageprotokolle erstellen und von Athena abfragen.

Die Route-53-Resolver-Abfrageprotokollierung dient zum Protokollieren von DNS-Abfragen, die von Ressourcen innerhalb einer VPC, On-Premises-Ressourcen, die eingehende Resolver-Endpunkte verwenden, Abfragen, die einen ausgehenden Resolver-Endpunkt für die rekursive DNS-Auflösung verwenden, und Abfragen, die Route-53-Resolver-DNS-Firewallregeln verwenden um eine Domainliste zu blockieren, zuzulassen oder zu überwachen. Weitere Informationen zur Resolver-Abfrageprotokollierung finden Sie unter Resolver-Abfrageprotokollierung im Entwicklerhandbuch für Amazon Route 53. Informationen zu den einzelnen Feldern in den Protokollen finden Sie unter Werte, die in Resolver-Abfrageprotokollen angezeigt werden im Entwicklerhandbuch zu Amazon Route 53.

Erstellen der Tabelle für Resolver-Abfrageprotokolle

Sie können den Abfrage-Editor in der Athena-Konsole verwenden, um eine Tabelle für Ihre Route-53-Resolver-Abfrageprotokolle zu erstellen und abzufragen.

So erstellen und fragen Sie eine Athena-Tabelle für Route-53-Resolver-Abfrageprotokolle ab

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Geben Sie im Athena-Abrage-Editor folgende `CREATE TABLE`-Anweisung ein. Ersetzen Sie die `LOCATION`-Klauselwerte durch diejenigen, die dem Speicherort Ihrer Resolver-Protokolle in Amazon S3 entsprechen.

```
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}/'
```

Da Resolver-Abfrageprotokolldaten im JSON-Format vorliegen, verwendet die CREATE TABLE-Anweisung eine JSON-SerDe-Bibliothek, um die Daten zu analysieren.

> **Note**
>
> Das SerDe erwartet, dass sich jedes JSON-Dokument auf einer einzigen Textzeile befindet, ohne Zeilenabschlusszeichen, die die Felder im Datensatz trennen. Wenn der JSON-Text im hübschen Druckformat vorliegt, erhalten Sie möglicherweise eine Fehlermeldung wie HIVE_CURSOR_ERROR: Zeile ist kein gültiges JSON-Objekt oder HIVE_CURSOR_ERROR: JsonParseException: Unerwartetes Ende der Eingabe: Erwartete Schließmarkierung für OBJEKT, wenn Sie versuchen, die Tabelle abzufragen, nachdem Sie dies erstellt haben. Weitere Informationen finden Sie unter JSON-Datendatei in der OpenX-SerDe-Dokumentation auf GitHub.

3. Wählen Sie Abfrage ausführen. Die Anweisung erstellt eine Athena-Tabelle mit dem Namen `r53_rlogs`, deren Spalten jedes der Felder in Ihren Resolver-Protokolldaten darstellen.
4. Führen Sie im Abfrage-Editor der Athena-Konsole die folgende Abfrage aus, um zu überprüfen, ob Ihre Tabelle erstellt wurde.

```
SELECT * FROM "r53_rlogs" LIMIT 10
```

Beispiele für Partitionierung

Das folgende Beispiel zeigt eine `CREATE TABLE` Anweisung für Resolver-Abfrageprotokolle, die Partitionsprojektion verwendet und nach VPC und Datum partitioniert ist. Weitere Informationen zur Partitionsprojektion finden Sie unter Partitionsprojektion mit 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}/'
)
```

Beispielabfragen

Die folgenden Beispiele zeigen einige Abfragen, die Sie von Athena in Ihren Resolver-Abfrageprotokollen ausführen können.

Beispiel 1 – Abfragen von Protokollen in absteigender Reihenfolge query_timestamp

Die folgende Abfrage zeigt Protokollergebnisse in absteigender `query_timestamp`-Reihenfolge an.

```
SELECT * FROM "r53_rlogs"
ORDER BY query_timestamp DESC
```

Beispiel 2 – Abfrage von Protokollen innerhalb der angegebenen Start- und Endzeiten

Die folgende Abfrage fragt Protokolle zwischen Mitternacht und 8 Uhr am 24. September 2020 ab. Ersetzen Sie die Start- und Endzeiten nach Ihren eigenen Anforderungen.

```
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
```

Beispiel 3 – Abfrageprotokolle basierend auf einem angegebenen DNS-Abfragenamenmuster

Die folgende Abfrage wählt Datensätze aus, deren Abfragename die Zeichenfolge „example.com“ enthält.

```
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
```

Beispiel 4 – Abfrageprotokollanforderungen ohne Antwort

Die folgende Abfrage wählt Protokolleinträge aus, in denen die Anforderung keine Antwort erhalten hat.

```
SELECT query_timestamp, srcids.instance, srcaddr, srcport, query_name, rcode, answers
FROM "r53_rlogs"
WHERE cardinality(answers) = 0
```

Beispiel 5 – Abfragen von Protokollen mit einer bestimmten Antwort

Die folgende Abfrage zeigt Protokolle, in denen der `answer.Rdata`-Wert die angegebene IP-Adresse hat.

```
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';
```

Abfragen von Amazon-SES-Ereignisprotokollen

Sie können Amazon Athena zum Abfragen von Ereignisprotokollen des Amazon Simple Email Service (Amazon SES) verwenden.

Amazon SES ist eine E-Mail-Plattform, die eine praktische und kostengünstige Möglichkeit bietet, E-Mails mit Ihren eigenen E-Mail-Adressen und Domänen zu versenden und zu empfangen. Sie können Ihre Amazon-SES-Versandaktivität auf granularer Ebene mithilfe von Ereignissen, Metriken und Statistiken überwachen.

Basierend auf den Eigenschaften, die Sie definieren, können Sie Amazon SES-Ereignisse in Amazon CloudWatch, Amazon Data Firehose oder Amazon Simple Notification Service veröffentlichen. Nachdem die Informationen in Amazon S3 gespeichert wurden, können Sie sie bei Amazon S3 von Amazon Athena abfragen.

Weitere Informationen zur Analyse von Amazon SES-E-Mail-Ereignissen mit Firehose, Amazon Athena und Amazon QuickSightfinden Sie unter Analysieren von Amazon SES-Ereignisdaten mit AWS Analytics Services im AWS Messaging and Targeting Blog.

Abfragen von Amazon-VPC-Flow-Protokollen

Amazon-Virtual-Private-Cloud-Flow-Protokolle erfassen Informationen zum IP-Datenverkehr zu und von Netzwerkschnittstellen in einer VPC. Verwenden Sie die Protokolle zur Untersuchung von Netzwerkdatenverkehrsmustern und zur Identifizierung von Bedrohungen und Risiken in Ihrem VPC-Netzwerk.

Sie haben zwei Möglichkeiten, Ihre Amazon-VPC-Flow-Protokolle abzufragen:

- Amazon-VPC-Konsole – Verwenden Sie die Athena-Integrationsfunktion in der Amazon-VPC-Konsole, um eine - AWS CloudFormation Vorlage zu generieren, die eine Athena-Datenbank-, Arbeitsgruppen- und Flow-Protokolltabelle mit Partitionierung für Sie erstellt. Die Vorlage erstellt auch eine Reihe von Vordefinierte Flow-Protokoll-Abfragen, die Sie benutzen können, um Einblicke in den Verkehr zu erhalten, der durch Ihre VPC fließt.

  Informationen zu diesem Ansatz finden Sie unter Abfrage-Flow-Protokolle mit Amazon Athena im Amazon-VPC-Benutzerhandbuch.
- Amazon-Athena-Konsole – Erstellen Sie Ihre Tabellen und Abfragen direkt in der Athena-Konsole. Weitere Informationen finden Sie auf dieser Seite.

Erstellen und Abfragen von Tabellen für benutzerdefinierte VPC-Flow-Protokolle

Bevor Sie die Protokolle in Athena abfragen, aktivieren Sie VPC-Flow-Protokolle und konfigurieren Sie diese so, dass sie in Ihrem Amazon-S3-Bucket gespeichert werden. Nach dem Erstellen der Protokolle führen Sie diese einige Minuten lang aus, um Daten zu erfassen. Die Protokolle werden in einem GZIP-Komprimierungsformat erstellt, das Sie über Athena direkt abfragen können.

Wenn Sie ein VPC-Flow-Protokoll erstellen, können Sie ein benutzerdefiniertes Format verwenden, wenn Sie die Felder, die im Flow-Protokoll zurückgegeben werden sollen, und die Reihenfolge, in der die Felder erscheinen, angeben möchten. Weitere Informationen zu Flow-Protokolle-Datensätzen finden Sie unter Flow-Protokolle-Datensätze im Amazon-VPC-Benutzerhandbuch.

Häufige Überlegungen

Beachten Sie beim Erstellen von Tabellen in Athena für Amazon-VPC-Flow-Protokolle die folgenden Punkte:

- Standardmäßig greift Parquet in Athena auf Spalten nach Namen zu. Weitere Informationen finden Sie unter Verarbeiten von Schema-Updates.

- Verwenden Sie die Namen in den Flow-Protokoll-Datensätzen für die Spaltennamen in Athena. Die Namen der Spalten im Athena-Schema sollten genau mit den Feldnamen in den Amazon-VPC-Flow-Protokollen übereinstimmen, mit den folgenden Unterschieden:
  - Ersetzen Sie die Bindestriche in den Namen des Amazon-VPC-Protokollfelds durch Unterstriche in den Spaltennamen in Athena. Für Datenbank-, Tabellen- und Spaltennamen in Athena dürfen nur Kleinbuchstaben, Zahlen und Unterstriche verwendet werden. Weitere Informationen finden Sie unter Datenbank-, Tabellen- und Spaltennamen.
  - Maskieren Sie die Namen der Flow-Protokoll-Datensätze, die in Athena reservierte Schlüsselwörter sind, indem Sie sie in Backticks einschließen.
- VPC-Flow-Protokolle sind AWS-Konto spezifisch. Wenn Sie Ihre Protokolldateien in Amazon S3 veröffentlichen, enthält der Pfad, den Amazon VPC in Amazon S3 erstellt, die ID des AWS-Konto , das für die Erstellung des Flow-Protokolls verwendet wurde. Weitere Informationen finden Sie unter Veröffentlichen von Flow-Protokollen in Amazon S3 im Amazon-VPC-Benutzerhandbuch.

CREATE-TABLE-Anweisung für Amazon-VPC-Flow-Protokolle

Mit dem folgenden Verfahren wird eine Amazon-VPC-Tabelle für Amazon-VPC-Flow-Protokolle erstellt. Wenn Sie ein Flow-Protokoll mit einem benutzerdefinierten Format erstellen, erstellen Sie eine Tabelle mit Feldern, die mit den Feldern übereinstimmen, die Sie bei der Erstellung des Flow-Protokolls in der gleichen Reihenfolge angegeben haben, in der Sie sie angegeben haben.

So erstellen Sie eine Athena-Tabelle für Amazon-VPC-Flow-Protokolle

1. Geben Sie eine DDL-Anweisung wie die folgende in den Abfrageeditor der Athena-Konsole ein, und befolgen Sie die Richtlinien in diesem Häufige Überlegungen-Abschnitt. Die Beispielanweisung erstellt eine Tabelle mit den Spalten für die Amazon-VPC-Flow-Protokolle der Versionen 2 bis 5, wie in Flow-Protokoll-Datensätze dokumentiert. Wenn Sie einen anderen Spaltensatz oder eine andere Spaltenreihenfolge verwenden, ändern Sie die Anweisung entsprechend.

```
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");
```

Beachten Sie folgende Punkte:

- Die Abfrage gibt an `ROW FORMAT DELIMITED` und lässt die Angabe eines aus SerDe. Das heißt, die Abfrage verwendet LazySimpleSerDe für CSV- und TSV-Dateien sowie für benutzerdefinierte, durch Trennzeichen getrennte Dateien. Bei dieser Abfrage werden Felder durch ein Leerzeichen beendet.
- Die `PARTITIONED BY`-Klausel verwendet den `date`-Typ. Dies ermöglicht es, in Abfragen mathematische Operatoren für eine Auswahl vor oder nach einem bestimmten Datum zu verwenden.

> **Note**
>
> Da `date` ein reserviertes Schlüsselwort in DDL-Anweisungen ist, wird es mit Back-Tick-Zeichen maskiert. Weitere Informationen finden Sie unter Reservierte Schlüsselwörter.

- Für ein VPC-Flow-Protokoll mit einem anderen benutzerdefinierten Format ändern Sie die Felder so, dass sie den Feldern entsprechen, die Sie beim Erstellen des Flow-Protokolls angegeben haben.

2. Ändern Sie `LOCATION 's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/{account_id}/vpcflowlogs/{region_code}/'` so, dass auf den Amazon-S3-Bucket gezeigt wird, der Ihre Protokolldaten enthält.
3. Führen Sie die Abfrage in der Athena-Konsole aus. Nach Beendigung der Abfrage registriert Athena die `vpc_flow_logs`-Tabelle, sodass Sie die Daten zum Ausgeben von Abfragen nutzen können.
4. Erstellen Sie Partitionen, um die Daten lesen zu können, wie in der folgenden Beispielabfrage. Diese Abfrage erstellt eine einfache Partition für ein angegebenes Datum. Ersetzen Sie die Platzhalter für Datum und Standort nach Bedarf.

> **Note**
>
> Diese Abfrage erstellt für einen von Ihnen angegebenen Zeitraum lediglich eine einzige Partition. Um den Prozess zu automatisieren, verwenden Sie ein Skript, das diese Abfrage ausführt und auf diese Weise Partitionen für `year/month/day` erstellt, oder verwenden Sie eine `CREATE TABLE`-Anweisung, die die partition projection (Partitionsprojektion) angibt.

```
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';
```

Beispielabfragen für die Tabelle vpc_flow_logs

Verwenden Sie den Abfrage-Editor in der Athena-Konsole, um SQL-Anweisungen für die von Ihnen erstellte Tabelle auszuführen. Sie können die Abfragen speichern, frühere Abfragen anzeigen oder Abfrageergebnisse im CSV-Format herunterladen. Ersetzen Sie in den folgenden Beispielen `vpc_flow_logs` mit dem Namen Ihrer Tabelle. Ändern Sie die Spaltenwerte und andere Variablen entsprechend Ihren Anforderungen.

Die folgende Beispielabfrage listet maximal 100 Flow-Protokolle für das angegebene Datum auf.

```
SELECT *
FROM vpc_flow_logs
WHERE date = DATE('2020-05-04')
LIMIT 100;
```

Mit der folgenden Abfrage werden alle abgelehnten TCP-Verbindungen aufgelistet. Die neu erstellte Datumspartitionsspalte `date` wird verwendet, um daraus den Wochentag zu extrahieren, an dem diese Ereignisse aufgetreten sind.

```
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;
```

Nutzen Sie die folgende Abfrage, um herauszufinden, welcher Ihrer Server die meisten HTTPS-Anforderungen empfängt. Bei der Abfrage werden die vom HTTPS-Port 443 empfangenen Pakete gezählt. Sie werden nach IP-Adressen gruppiert und es werden die ersten 10 der letzten Woche zurückgegeben.

```
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;
```

Erstellen von Tabellen für Flow-Protokolle im Apache-Parquet-Format

Das folgende Verfahren erstellt eine Amazon-VPC-Tabelle für Amazon-VPC-Flow-Protokolle im Apache-Parquet-Format.

So erstellen Sie eine Athena-Tabelle für Amazon-VPC-Flow-Protokolle im Parquet-Format

1. Geben Sie eine DDL-Anweisung wie die folgende in den Abfrageeditor der Athena-Konsole ein und befolgen Sie die Richtlinien in diesem Häufige Überlegungen-Abschnitt. Die Beispielanweisung erstellt eine Tabelle mit den Spalten für Amazon-VPC-Flow-Protokolle Version 2 bis 5, wie in Flow-Protokoll-Datensätze im Parquet-Format dokumentiert, Hive stündlich partitioniert. Wenn Sie keine stündlichen Partitionen haben, entfernen Sie `hour` aus der `PARTITIONED BY`-Klausel.

```
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. Ändern Sie Beispiel-`LOCATION 's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/'` so, dass auf den Amazon-S3-Pfad gezeigt wird, der Ihre Protokolldaten enthält.
3. Führen Sie die Abfrage in der Athena-Konsole aus.
4. Wenn Ihre Daten im HIVE-kompatiblen Format vorliegen, führen Sie den folgenden Befehl in der Athena-Konsole aus, um die Hive-Partitionen im Metastore zu aktualisieren und zu laden. Nachdem die Abfrage abgeschlossen ist, können Sie die Daten in der `vpc_flow_logs_parquet`-Tabelle abfragen.

   ```
   MSCK REPAIR TABLE vpc_flow_logs_parquet
   ```

   Wenn Sie keine Hive-kompatiblen Daten verwenden, führen Sie ALTER TABLE ADD PARTITION aus, um die Partitionen zu laden.

Weitere Informationen zur Verwendung von Athena zum Abfragen von Amazon-VPC-Flow-Protokollen im Parquet-Format finden Sie im Beitrag Optimieren der Leistung und Reduzieren der Kosten für Netzwerkanalysen mit VPC-Flow-Protokollen im Apache-Parquet-Format im AWS Big Data Blog.

Erstellen und Abfragen einer Tabelle für Amazon-VPC-Flow-Protokolle mit Partitionsprojektion

Verwenden Sie eine `CREATE TABLE`-Anweisung wie die folgende, um eine Tabelle zu erstellen, die Tabelle zu partitionieren und die Partitionen mithilfe der partition projection (Partitionsprojektion) automatisch aufzufüllen. Ersetzt den Tabellennamen `test_table_vpclogs` im Beispiel mit dem Namen Ihrer Tabelle. Bearbeiten Sie die `LOCATION`-Klausel, um den Amazon-S3-Bucket anzugeben, der Ihre Amazon-VPC-Protokolldaten enthält.

Folgende `CREATE TABLE`-Anweisung gilt für VPC-Flow-Protokolle, die im Partitionierungsformat im Nicht-Hive-Stil geliefert werden. Das Beispiel ermöglicht die Aggregation mehrerer Konten. Wenn Sie VPC-Ablaufprotokolle von mehreren Konten in einem Amazon-S3-Bucket zentralisieren, muss die Konto-ID im Amazon-S3-Pfad angegeben werden.

```
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" = "$START_RANGE,NOW",
"projection.day.format" = "yyyy/MM/dd",
"storage.location.template" = "s3://$LOCATION_OF_LOGS/AWSLogs/${accid}/vpcflowlogs/
${region}/${day}"
)
```

Beispielabfragen für test_table_vpclogs

Die folgenden Beispielabfragen fragen `test_table_vpclogs` ab, das von der vorherigen `CREATE TABLE`-Anweisung erstellt wurde. Ersetzen Sie `test_table_vpclogs` in den Abfragen mit dem Namen Ihrer eigenen Tabelle. Ändern Sie die Spaltenwerte und andere Variablen entsprechend Ihren Anforderungen.

Um die ersten 100 Zugriffsprotokolleinträge für einen bestimmten Zeitraum in chronologischer Reihenfolge zurückzugeben, führen Sie eine Abfrage wie die folgende aus.

```
SELECT *
FROM test_table_vpclogs
WHERE day >= '2021/02/01' AND day < '2021/02/28'
ORDER BY day ASC
LIMIT 100
```

Um anzuzeigen, welcher Server die zehn wichtigsten HTTP-Pakete für einen bestimmten Zeitraum erhält, führen Sie eine Abfrage wie die folgende aus. Die Abfrage zählt die Anzahl der auf HTTPS-Port 443 empfangenen Pakete, gruppiert sie nach Ziel-IP-Adresse und gibt die Top-10-Einträge der Vorwoche zurück.

```
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
```

Um die Protokolle zurückzugeben, die während eines bestimmten Zeitraums erstellt wurden, führen Sie eine Abfrage wie die folgende aus.

```
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'
```

Um die Zugriffsprotokolle für eine Quell-IP-Adresse zwischen bestimmten Zeiträumen zurückzugeben, führen Sie eine Abfrage wie die folgende aus.

```
SELECT *
FROM test_table_vpclogs
WHERE srcaddr = '10.117.1.22'
  AND day >= '2021/02/01'
  AND day < '2021/02/28'
```

Führen Sie eine Abfrage wie die folgende aus, um abgelehnte TCP-Verbindungen aufzulisten.

```
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
```

Führen Sie eine Abfrage wie die folgende aus, um die Zugriffsprotokolle für den IP-Adressbereich zurückzugeben, der mit `10.117` beginnt.

```
SELECT *
FROM test_table_vpclogs
WHERE split_part(srcaddr,'.', 1)='10'
  AND split_part(srcaddr,'.', 2) ='117'
```

Um die Zugriffsprotokolle für eine Ziel-IP-Adresse in einem bestimmten Zeitraum zurückzugeben, führen Sie eine Abfrage wie die folgende aus.

```
SELECT *
FROM test_table_vpclogs
WHERE dstaddr = '10.0.1.14'
  AND day >= '2021/01/01'
  AND day < '2021/01/31'
```

Tabellen für Flow-Protokolle im Apache-Parquet-Format mit Partitionsprojektion erstellen

Die folgende CREATE-TABLE-Anweisung der Partitionsprojektion für VPC-Flow-Logs ist im Apache-Parquet-Format, nicht Hive-kompatibel und nach Stunde und Datum statt nach Tag partitioniert. Ersetzt den Tabellennamen `test_table_vpclogs_parquet` im Beispiel mit dem Namen Ihrer Tabelle. Bearbeiten Sie die `LOCATION`-Klausel, um den Amazon-S3-Bucket anzugeben, der Ihre Amazon-VPC-Protokolldaten enthält.

```
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}"
```

```
)
```

Weitere Ressourcen

Weitere Informationen zum Abfragen von VPC-Ablaufprotokollen mithilfe von Athena finden Sie in den folgenden Beiträgen aus dem AWS -Big-Data-Blog:

- Analysieren von VPC-Flow-Protokollen mit point-and-click Amazon Athena-Integration
- Analysieren von VPC-Flow-Protokollen mit Amazon Athena und Amazon QuickSight
- Die Leistung optimieren und die Kosten für Netzwerkanalysen mit VPC-Ablaufprotokollen im Apache-Parquet-Format senken

Abfragen von AWS WAF Protokollen

AWS WAF ist eine Firewall für Webanwendungen, mit der Sie die HTTP- und HTTPS-Anforderungen überwachen und steuern können, die Ihre geschützten Webanwendungen von Clients erhalten. Sie definieren, wie die Webanforderungen verarbeitet werden, indem Sie Regeln in einer - AWS WAF Web-Zugriffskontrollliste (ACL) konfigurieren. Anschließend schützen Sie eine Webanwendung, indem Sie ihr eine Web-ACL zuordnen. Beispiele für Webanwendungsressourcen, mit denen Sie schützen können, AWS WAF sind Amazon- CloudFront Verteilungen, Amazon API Gateway-REST-APIs und Application Load Balancer. Weitere Informationen zu finden Sie AWS WAFunter AWS WAF im AWS WAF Entwicklerhandbuch für .

AWS WAF -Protokolle enthalten Informationen über den Datenverkehr, der von Ihrer Web-ACL analysiert wird, z. B. den Zeitpunkt, zu dem die Anforderung von Ihrer AWS Ressource AWS WAF empfangen hat, detaillierte Informationen über die Anforderung und die Aktion für die Regel, der die einzelnen Anforderungen entsprachen.

Sie können eine AWS WAF -Web-ACL so konfigurieren, dass Protokolle an einem von mehreren Zielen veröffentlicht werden, wo Sie sie abfragen und anzeigen können. Weitere Informationen zur Konfiguration der Web-ACL-Protokollierung und zum Inhalt der AWS WAF Protokolle finden Sie unter Protokollieren von AWS WAF Web-ACL-Datenverkehr im AWS WAF Entwicklerhandbuch für .

Ein Beispiel dafür, wie AWS WAF Protokolle in einem zentralen Data-Lake-Repository aggregiert und mit Athena abgefragt werden, finden Sie im AWS -Big-Data-Blogbeitrag Analysieren von AWS WAF Protokollen mit OpenSearch Service, Amazon Athena und Amazon QuickSight.

In diesem Thema finden Sie zwei `CREATE TABLE`-Beispielanweisungen: eine, die Partitionierung verwendet, und eine, die keine Partitionierung verwendet.

> **Note**
>
> Die `CREATE TABLE`-Anweisungen in diesem Thema können sowohl für Version 1 als auch für Version 2 der AWS WAF -Protokolle verwendet werden. In Version 1 enthält das `webaclid`-Feld eine ID. In Version 2 enthält das `webaclid`-Feld einen vollständigen ARN. Die `CREATE TABLE`-Anweisungen behandeln diesen Inhalt agnostisch, indem sie den `string`-Datentyp verwenden.

Themen

Erstellen einer Tabelle für AWS WAF -S3-Protokolle in Athena mithilfe der Partitionsprojektion

Da AWS WAF Protokolle über eine bekannte Struktur verfügen, deren Partitionsschema Sie im Voraus angeben können, können Sie die Abfragelaufzeit reduzieren und die Partitionsverwaltung mithilfe der Athena-Partitionsprojektion automatisieren. Partitionsprojektion fügt automatisch neue Partitionen hinzu, wenn neue Daten hinzugefügt werden. Dadurch entfällt die Notwendigkeit, Partitionen manuell mithilfe von `ALTER TABLE ADD PARTITION` hinzuzufügen.

Die folgende `CREATE TABLE` Beispielanweisung verwendet automatisch die Partitionsprojektion für AWS WAF Protokolle von einem bestimmten Datum bis zum aktuellen für vier verschiedene AWS Regionen. Die `PARTITION BY`-Klausel in diesem Beispiel partitioniert nach Region und Datum, aber Sie können diesen Vorgang entsprechend Ihren Anforderungen ändern. Ändern Sie die Felder nach Bedarf, damit diese mit Ihrer Protokollausgabe übereinstimmen. Ersetzen Sie in den `storage.location.template` Klauseln `LOCATION` und die Platzhalter *`bucket`* und *`accountID`* durch Werte, die den Amazon S3-Bucket-Speicherort Ihrer AWS WAF Protokolle identifizieren. Ersetzen Sie für `projection.day.range` den *`01.01.2021`* durch das Startdatum, das Sie verwenden möchten. Nach dem erfolgreichen Ausführen der Abfrage können Sie die Tabelle abfragen. Sie müssen `ALTER TABLE ADD PARTITION` nicht ausführen, um die Partitionen zu laden.

```
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
)
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**
>
> Das Format des Pfads in der -LOCATIONKlausel im Beispiel ist Standard, kann aber je nach der von Ihnen implementierten AWS WAF Konfiguration variieren. Der folgende AWS WAF Beispielprotokollpfad gilt beispielsweise für eine - CloudFront Verteilung:

```
s3://DOC-EXAMPLE-BUCKET/AWSLogs/12345678910/WAFLogs/cloudfront/
cloudfronyt/2022/08/08/17/55/
```

Wenn beim Erstellen oder Abfragen Ihrer AWS WAF Protokolltabelle Probleme auftreten, bestätigen Sie den Speicherort Ihrer Protokolldaten oder wenden Sie sich an AWS Support.

Weitere Informationen zur Partitionsprojektion finden Sie unter Partitionsprojektion mit Amazon Athena.

Erstellen einer Tabelle für AWS WAF Protokolle ohne Partitionierung

In diesem Abschnitt wird beschrieben, wie Sie eine Tabelle für AWS WAF Protokolle ohne Partitionierung oder Partitionsprojektion erstellen.

Note

Aus Leistungs- und Kostengründen empfehlen wir nicht, nicht partitioniertes Schema für Abfragen zu verwenden. Weitere Informationen finden Sie unter Top 10 Tipps zur Leistungsoptimierung für Amazon Athena im - AWS Big-Data-Blog.

So erstellen Sie die AWS WAF Tabelle

1. Kopieren Sie die folgende DDL-Anweisung in die Athena-Konsole. Ändern Sie die Felder nach Bedarf, damit diese mit Ihrer Protokollausgabe übereinstimmen. Ändern Sie den `LOCATION` des Amazon-S3-Bucket, in dem die Protokolle gespeichert werden.

   Diese Abfrage verwendet die OpenX JSON SerDe.

   Note

   erwartet, SerDe dass sich jedes JSON-Dokument in einer einzigen Textzeile befindet, ohne Zeilenabschlusszeichen, die die Felder im Datensatz trennen. Wenn der JSON-Text ein hübsches Druckformat aufweist, erhalten Sie möglicherweise eine Fehlermeldung wie HIVE_CURSOR_ERROR: Zeile ist kein gültiges JSON-Objekt oder HIVE_CURSOR_ERROR: JsonParseException: Unerwartet end-of-input: erwartete Schließmarkierung für OBJECT, wenn Sie versuchen, die Tabelle abzufragen, nachdem

Sie sie erstellt haben. Weitere Informationen finden Sie unter JSON-Datendateien in der OpenX SerDe -Dokumentation auf 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
)
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. Führen Sie die `CREATE EXTERNAL TABLE`-Anweisung im Abfrage-Editor der Athena-Konsole aus. Dadurch wird die `waf_logs`-Tabelle registriert und die darin enthaltenen Daten für Abfragen von Athena verfügbar gemacht.

Beispielabfragen für AWS WAF Protokolle

Viele der folgenden Beispielabfragen fragen die zuvor in diesem Dokument erstellte Partitionsprojektionstabelle ab. Ändern Sie den Tabellennamen, die Spaltenwerte und andere Variablen in den Beispielen entsprechend Ihren Anforderungen. Um die Leistung Ihrer Abfragen zu verbessern und Kosten zu senken, fügen Sie die Partitionsspalte in der Filterbedingung hinzu.

- 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

Arbeiten mit Datum und Uhrzeit

- 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

Arbeiten mit blockierten Anforderungen und Adressen

- 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 – Zählt die Anzahl der Referrer, die einen bestimmten Begriff enthalten

Die folgende Abfrage zählt die Anzahl der Referrer, die den Begriff „amazon“ für den angegebenen Datumsbereich enthalten.

```
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 – Zählt alle übereinstimmenden IP-Adressen in den letzten 10 Tagen, die mit ausgeschlossenen Regeln übereinstimmen

Mit der folgenden Abfrage wird die Anzahl der Male in den letzten 10 Tagen gezählt, die die IP-Adresse der ausgeschlossenen Regel in der Regelgruppe entspricht.

```
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 – Gruppiert alle gezählten verwalteten Regeln nach der Anzahl der Treffer

Wenn Sie Regelgruppenregelaktionen vor dem 27. Oktober 2022 auf Count in Ihrer Web-ACL-Konfiguration festlegen, hat Ihre Überschreibungen in der Web-ACL-JSON als AWS WAF gespeichert`excludedRules`. Jetzt befindet sich die JSON-Einstellung zum Überschreiben einer Regel zum Zählen in den `ruleAction0verrides`-Einstellungen. Weitere Informationen finden Sie unter Aktionsüberschreibungen in Regelgruppen im AWS WAF -Entwicklerhandbuch. Um verwaltete Regeln im Zähl-Modus aus der neuen Protokollstruktur zu extrahieren, fragen Sie den Wert `nonTerminatingMatchingRules` im Abschnitt `ruleGroupList` statt im Feld `excludedRules` ab, wie im folgenden Beispiel.

```
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 – Gruppiert alle gezählten benutzerdefinierten Regeln nach der Anzahl der Treffer

Die folgende Abfrage gruppiert alle gezählten benutzerdefinierten Regeln nach der Anzahl der Übereinstimmungen.

```
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
```

Informationen zu den Protokollspeicherorten für benutzerdefinierte Regeln und verwaltete Regelgruppen finden Sie unter Überwachung und Optimierung im AWS WAF -Entwicklerhandbuch.

Arbeiten mit Datum und Uhrzeit

Example – Gibt das Zeitstempelfeld im menschenlesbaren ISO-8601-Format zurück

Die folgende Abfrage verwendet die `from_unixtime`- und `to_iso8601`-Funktionen, um das `timestamp`-Feld im menschenlesbaren ISO 8601-Format zurückzugeben (z. B. `2019-12-13T23:40:12.000Z` statt `1576280412771`). Die Abfrage gibt auch den HTTP-Quellnamen, die Quell-ID und die Anforderung zurück.

```
SELECT to_iso8601(from_unixtime(timestamp / 1000)) as time_ISO_8601,
       httpsourcename,
       httpsourceid,
       httprequest
FROM waf_logs
LIMIT 10;
```

Example – Gibt Datensätze der letzten 24 Stunden zurück

Die folgende Abfrage verwendet einen Filter in der WHERE-Klausel, um den HTTP-Quellnamen, die HTTP-Quell-ID und die HTTP-Anforderungsfelder für Datensätze der letzten 24 Stunden zurückzugeben.

```
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 – Gibt Datensätze für einen angegebenen Datumsbereich und eine IP-Adresse zurück

Die folgende Abfrage listet die Datensätze in einem angegebenen Datumsbereich für eine angegebene Client-IP-Adresse auf.

```
SELECT *
FROM waf_logs
WHERE httprequest.clientip='53.21.198.66' AND "date" >= '2021/03/01' AND "date" <
 '2021/03/31'
```

Example – Zählt Sie für einen angegebenen Datumsbereich die Anzahl der IP-Adressen in 5-Minuten-Intervallen

Die folgende Abfrage zählt für einen bestimmten Datumsbereich die Anzahl der IP-Adressen in fünf Minuten Intervallen.

```
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 – Zählt die Anzahl der X-Forwarded-For-IP in den letzten zehn Tagen

Die folgende Abfrage filtert die Anforderungsheader und zählt die Anzahl der X-Forwarded-For-IPs in den letzten zehn Tagen.

```
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
```

Weitere Informationen zu Datums- und Uhrzeitfunktionen finden Sie unter Datums- und Uhrzeitfunktionen und Operatoren in der Trino-Dokumentation.

Arbeiten mit blockierten Anforderungen und Adressen

Example – Extrahiert die Top-100-IP-Adressen, die von einem angegebenen Regeltyp blockiert werden

Die folgende Abfrage extrahiert und zählt die 100 häufigsten IP-Adressen, die während des angegebenen Datumsbereichs durch die `RATE_BASED`-Beendigungsregel blockiert wurden.

```
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 – Zählt, wie oft eine Anfrage aus einem bestimmten Land blockiert wurde

Die folgende Abfrage zählt, wie oft die Anforderung von einer IP-Adresse angekommen ist, die zu Irland (IE) gehört und von der `RATE_BASED`-Beendigungsregel gesperrt wurde.

```
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 – Zählt, wie oft eine Anforderung blockiert wurde, gruppiert nach bestimmten Attributen

Die folgende Abfrage zählt, wie oft die Anforderung blockiert wurde, wobei die Ergebnisse nach WebACL , RuleIdClientIP und HTTP-Anforderungs-URI gruppiert sind.

```
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 – Zählt, wie oft eine bestimmte beendende Regel-ID abgeglichen wurde

Die folgende Abfrage zählt, wie oft eine bestimmte Beendigungsregel-ID vorkam (`WHERE terminatingruleid='e9dd190d-7a43-4c06-bcea-409613d9506e'`). Die Abfrage gruppiert dann die Ergebnisse nach WebACL, Action, ClientIP und HTTP-Anforderungs-URI.

```
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 – Abrufen der Top-100-IP-Adressen, die während eines angegebenen Datumsbereichs blockiert wurden

Die folgende Abfrage extrahiert die Top-100-IP-Adressen, die für einen angegebenen Datumsbereich gesperrt wurden. Die Abfrage listet auch auf, wie oft die IP-Adressen gesperrt wurden.

```
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
```

Informationen zum Abfragen von Amazon-S3-Protokollen finden Sie in den folgenden Themen:

- Wie analysiere ich meine Amazon-S3-Serverzugriffsprotokolle mit Athena? im AWS-Wissenscenter
- Abfragen von Amazon-S3-Zugriffsprotokollen für Anfragen mit Amazon Athena im Benutzerhandbuch für Amazon Simple Storage Service.
- Verwenden von AWS CloudTrail zum Identifizieren von Amazon-S3-Anforderungen im Benutzerhandbuch für Amazon Simple Storage Service.

Abfragen von Webserverprotokollen in Amazon S3

Sie können Athena verwenden, um Webserverprotokolle abzufragen, die in Amazon S3 gespeichert sind. In den Themen in diesem Abschnitt wird gezeigt, wie Tabellen in Athena erstellt werden, in denen Webserverprotokolle in einer Vielzahl von Formaten abgefragt werden.

Themen

- Abfragen von Apache-Protokollen in Amazon S3 abfragen
- Abfragen von Internetinformationsserver-Protokollen (IIS), die in Amazon S3 gespeichert sind

Abfragen von Apache-Protokollen in Amazon S3 abfragen

Sie können Amazon Athena verwenden, um Protokolldateien von Apache HTTP Server abzufragen, die in Ihrem Amazon-S3-Konto gespeichert sind. In diesem Thema erfahren Sie, wie Sie Tabellenschemas erstellen, um Apache-Zugriffsprotokoll-Dateien im allgemeinen Protokollformat abzufragen.

Zu den Feldern im allgemeinen Protokollformat gehören die Client-IP-Adresse, die Client-ID, die Benutzer-ID, der empfangene Zeitstempel der Anforderung, der Text der Clientanforderung, der Serverstatuscode und die Größe des an den Client zurückgegebenen Objekts.

Die folgenden Beispieldaten zeigen das allgemeine Apache-Protokollformat.

```
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
```

Erstellen einer Tabelle in Athena für Apache-Protokolle

Bevor Sie Apache-Protokolle abfragen können, die in Amazon S3 gespeichert sind, müssen Sie ein Tabellenschema für Athena erstellen, in dem Sie die Protokolldaten lesen. Um eine Athena-Tabelle für Apache-Protokolle zu erstellen, können Sie die Grok SerDe verwenden. Weitere Informationen zur Verwendung des Grok SerDe finden Sie unter Angepasste Grok-Classifier schreiben im AWS Glue-Entwicklerhandbuch.

So erstellen Sie eine Tabelle in Athena für Apache-Webserver-Protokolle

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Kopieren Sie die folgende DDL-Anweisung und fügen Sie sie in den Abfrage-Editor der Athena-Konsole ein: Ändern Sie die Werte in `LOCATION 's3://bucket-name/apache-log-folder/'`, um auf Ihre Apache-Protokolle in Amazon S3 zu verweisen.

```
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://bucket-name/apache-log-folder/';
```

3. Führen Sie die Abfrage in der Athena-Konsole aus, um die `apache_logs`-Tabelle zu registrieren. Wenn die Abfrage abgeschlossen ist, können Sie die Protokolle aus Athena abfragen.

Beispiel „Abfragen für Apache-Protokolle auswählen“

Example – Filterung nach 404-Fehlern

Die folgende Beispielabfrage wählt die Empfangszeit der Anforderung, den Text der Clientanforderung und den Serverstatuscode aus der `apache_logs`-Tabelle aus. Die `WHERE`-Klausel filtert nach HTTP-Statuscode `404` (Seite nicht gefunden)

```
SELECT request_received_time, client_request, server_status
FROM apache_logs
WHERE server_status = '404'
```

Das folgende Image zeigt die Ergebnisse der Abfrage im Athena-Abfrage-Editor.

Results

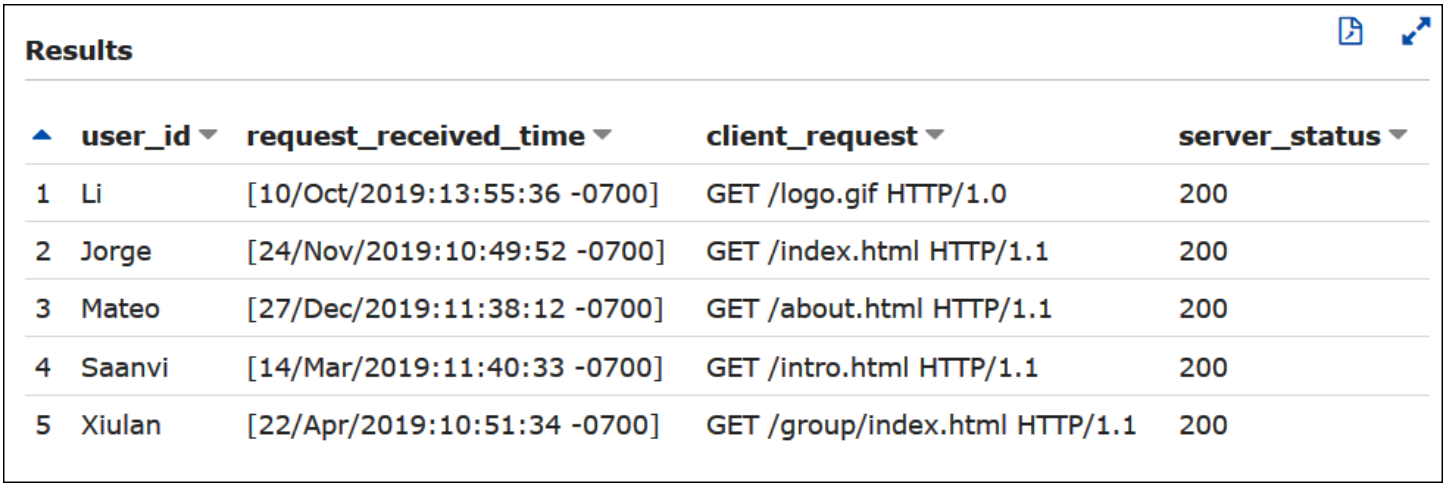
	request_received_time	client_request	server_status
1	[11/Jan/2020:11:40:11 -0700]	GET /image.png HTTP/1.1	404
2	[15/Feb/2019:10:12:22 -0700]	GET /favicon.ico HTTP/1.1	404

Example — Filterung nach erfolgreichen Anfragen

Die folgende Beispielabfrage wählt die Benutzer-ID, die Empfangszeit der Anforderung, den Text der Clientanforderung und den Serverstatuscode aus der `apache_logs`-Tabelle aus. Die `WHERE`-Klausel filtert nach HTTP-Statuscode `200` (erfolgreich).

```
SELECT user_id, request_received_time, client_request, server_status
FROM apache_logs
WHERE server_status = '200'
```

Das folgende Image zeigt die Ergebnisse der Abfrage im Athena-Abfrage-Editor.



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 – Filterung nach Zeitstempel

Im folgenden Beispiel werden Datensätze abgefragt, deren Empfangszeit der Anfrage länger als der angegebene Zeitstempel ist.

```
SELECT * FROM apache_logs WHERE request_received_time > 10/Oct/2023:00:00:00
```

Abfragen von Internetinformationsserver-Protokollen (IIS), die in Amazon S3 gespeichert sind

Sie können Amazon Athena verwenden, um die Webserverprotokolle von Microsoft Internet Information Services (IIS) abzufragen, die in Ihrem Amazon-S3-Konto gespeichert sind. Obwohl IIS eine Vielzahl von Protokolldateiformaten verwendet, zeigt Ihnen dieses Thema, wie Sie Tabellenschemas erstellen, um W3C-erweiterte Protokolle und Protokolle im IIS-Protokolldateiformat von Athena abzufragen.

Da die W3C-erweiterten und IIS-Protokolldateiformate einzelne Zeichentrennzeichen (Leerzeichen bzw. Kommas) verwenden und keine Werte in Anführungszeichen eingeschlossen sind, können Sie mit LazySimpleSerDe Athena-Tabellen dafür erstellen.

Erweitertes W3C-Format für Protokolldateien

Das erweiterte W3C-Protokolldateidatenformat hat durch Leerzeichen getrennte Felder. Die Felder, die in erweiterten W3C-Protokollen erscheinen, werden von einem Webserver-Administrator bestimmt, der auswählt, welche Protokollfelder eingeschlossen werden sollen. Die folgenden Beispielprotokolldaten enthalten die Felder `date, time`, `c-ip`, `s-ip`, `cs-method`, `cs-uri-stem`, `sc-status`, `sc-bytes`, `cs-bytes`, `time-taken` und `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
```

Erstellen einer Tabelle in Athena für erweiterte W3C-Protokolle

Bevor Sie Ihre erweiterten W3C-Protokolle abfragen können, müssen Sie ein Tabellenschema erstellen, damit Athena die Protokolldaten lesen kann.

So erstellen Sie eine Tabelle in Athena für erweiterte W3C-Protokolle

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Fügen Sie eine DDL-Anweisung wie folgt in die Athena-Konsole ein und beachten Sie folgende Punkte:

    a. Fügen Sie die Spalten im Beispiel hinzu, oder entfernen Sie sie, damit sie den Feldern in den Protokollen entsprechen, die Sie abfragen möchten.

b. Spaltennamen im erweiterten W3C-Format für Protokolldateien enthalten Bindestriche (-). In Übereinstimmung mit den Athena-Namenskonventionen werden sie jedoch in der `CREATE TABLE`-Beispielanweisung durch Unterstriche (_) ersetzt.

c. Um das Leerzeichen anzugeben, verwenden Sie `FIELDS TERMINATED BY ' '`.

d. Ändern Sie die Werte in `LOCATION 's3://bucket-name/w3c-log-folder/'` so, dass sie auf Ihre erweiterten W3C-Protokolle in Amazon S3 verweisen.

```
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://bucket-name/w3c-log-folder/'
```

3. Führen Sie die Abfrage in der Athena-Konsole aus, um die `iis_w3c_logs`-Tabelle zu registrieren. Wenn die Abfrage abgeschlossen ist, können Sie die Protokolle aus Athena abfragen.

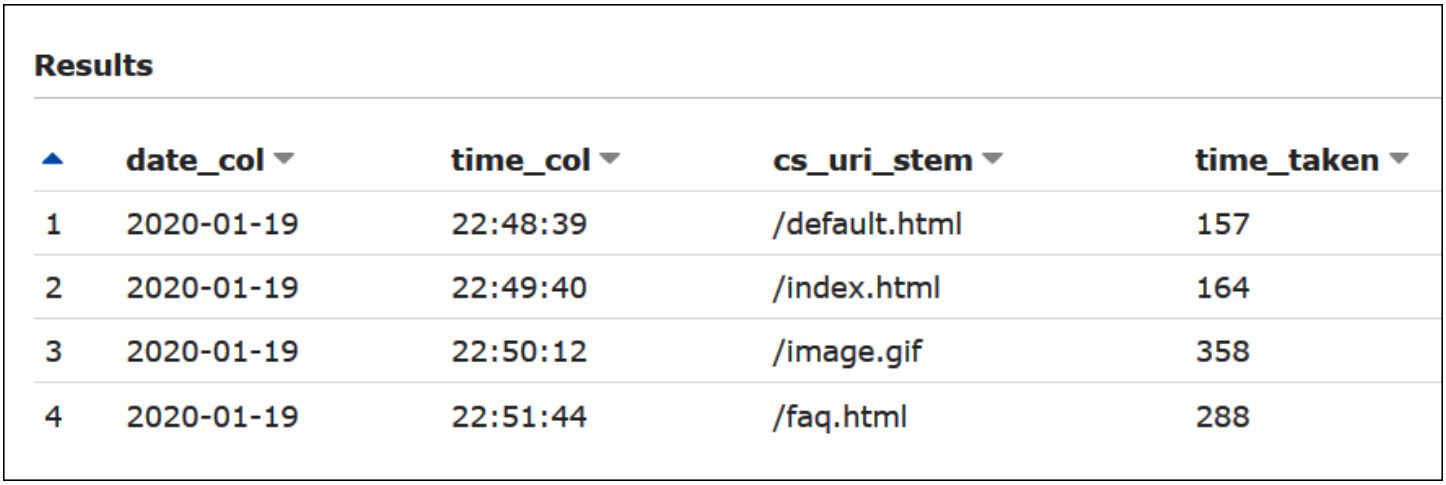
Beispiel für eine erweiterte W3C-Protokoll-Auswahlabfrage

Die folgende Beispielabfrage wählt Datum, Uhrzeit, Anforderungsziel und die für die Anforderung benötigte Zeit aus der Tabelle `iis_w3c_logs` aus. Die `WHERE`-Klausel filtert nach Fällen, in denen die HTTP-Methode `GET` und der HTTP-Statuscode `200` (erfolgreich) ist.

```
SELECT date_col, time_col, cs_uri_stem, time_taken
```

```
FROM iis_w3c_logs
WHERE cs_method = 'GET' AND sc_status = '200'
```

Das folgende Image zeigt die Ergebnisse der Abfrage im Athena-Abfrage-Editor.



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

Kombinieren der Datums- und Zeitfelder

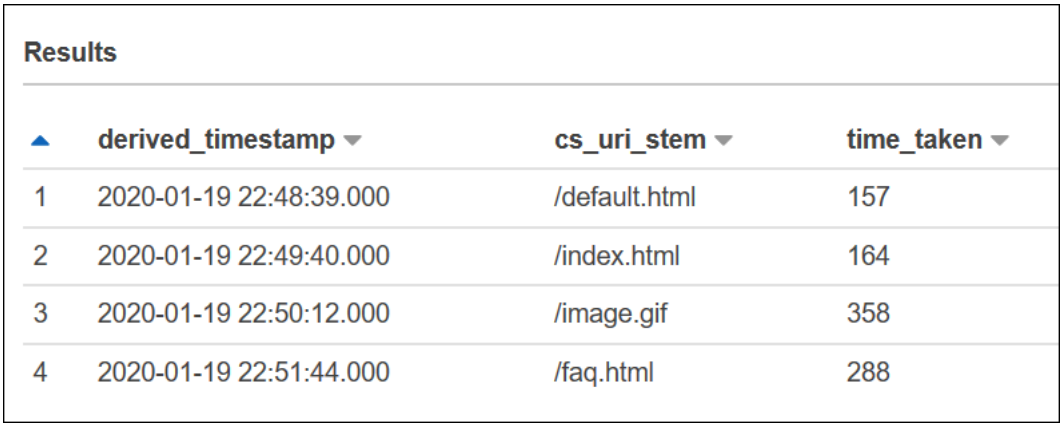
Die durch Leerzeichen getrennten `date`- und `time`-Felder sind separate Einträge in den Protokollquelldaten, aber Sie können sie bei Bedarf zu einem Zeitstempel kombinieren. Verwenden Sie die Funktionen concat() und date_parse() in einer SELECT- oder CREATE TABLE AS SELECT-Abfrage, um die Datums- und Uhrzeitspalten zu verketten und in das Zeitstempelformat zu konvertieren. Im folgenden Beispiel wird eine CTAS-Abfrage verwendet, um eine neue Tabelle mit einer `derived_timestamp`-Spalte zu erstellen.

```
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
```

Nachdem die Tabelle erstellt wurde, können Sie die neue Zeitstempelspalte direkt abfragen, wie im folgenden Beispiel.

```
SELECT derived_timestamp, cs_uri_stem, time_taken
FROM iis_w3c_logs_w_timestamp
WHERE cs_method = 'GET' AND sc_status = '200'
```

Das folgende Image zeigt die Ergebnisse der Abfrage.



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

IIS-Protokolldateiformat

Im Gegensatz zum erweiterten W3C-Format verfügt das IIS-Protokolldateiformat über einen festen Satz von Feldern und enthält ein Komma als Trennzeichen. Der LazySimpleSerDe behandelt das Komma als Trennzeichen und das Leerzeichen nach dem Komma als Anfang des nächsten Feldes.

Das folgende Beispiel zeigt Beispieldaten im IIS-Protokolldateiformat.

```
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, -,
```

Erstellen einer Tabelle in Athena für IIS-Protokolldateien

Um die Protokolle des IIS-Protokolldateiformats in Amazon S3 abzufragen, erstellen Sie zunächst ein Tabellenschema, damit Athena die Protokolldaten lesen kann.

So erstellen Sie eine Tabelle in Athena für Protokolle im IIS-Protokolldateiformat

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.

2. Fügen Sie die folgende DDL-Anweisung in die Athena-Konsole ein und beachten Sie dabei die folgenden Punkte:

   a. Verwenden Sie `FIELDS TERMINATED BY ','`, um das Kommatrennzeichen anzugeben.

   b. Ändern Sie die Werte in LOCATION 's3://*bucket-name*/*iis-log-file-folder*/' so, dass sie auf Ihre Protokolldateien im IIS-Protokollformat in Amazon S3 verweisen.

```
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://bucket-name/iis-log-file-folder/'
```

3. Führen Sie die Abfrage in der Athena-Konsole aus, um die `iis_format_logs`-Tabelle zu registrieren. Wenn die Abfrage abgeschlossen ist, können Sie die Protokolle aus Athena abfragen.

Beispiel für IIS-Protokollformat-Auswahlabfrage

Die folgende Beispielabfrage wählt das Anforderungsdatum, die Anforderungszeit, das Anforderungsziel und die benötigte Zeit in Millisekunden aus der Tabelle `iis_format_logs` aus. Die `WHERE`-Klausel filtert nach Fällen, in denen der Anforderungstyp `GET` und der HTTP-Statuscode `200` (erfolgreich) ist. Beachten Sie in der Abfrage, dass die führenden Leerzeichen in `' GET'` und `' 200'` erforderlich sind, damit die Abfrage erfolgreich ist.

```
SELECT request_date, request_time, target_of_operation, time_taken_millisec
FROM iis_format_logs
WHERE request_type = ' GET' AND service_status_code = ' 200'
```

Das folgende Image zeigt die Ergebnisse der Abfrage der Beispieldaten.

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

NCSA-Protokolldateiformat

IIS verwendet auch das NCSA-Protokollierungsformat, das eine feste Anzahl von Feldern im ASCII-Textformat aufweist, die durch Leerzeichen getrennt sind. Die Struktur ähnelt dem allgemeinen Protokollformat, das für Apache-Zugriffsprotokolle verwendet wird. Felder im allgemeinen NCSA-Protokolldatenformat umfassen die Client-IP-Adresse, die Client-ID (wird normalerweise nicht verwendet), die Domäne\Benutzer-ID, den Zeitstempel der empfangenen Anforderung, den Text der Client-Anfrage, den Serverstatuscode und die Größe des an den Client zurückgegebenen Objekts .

Das folgende Beispiel zeigt Daten im allgemeinen NCSA-Protokollformat, wie für IIS dokumentiert.

```
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
```

Erstellen einer Tabelle in Athena für IIS-NCSA-Protokolle

Für Ihre `CREATE TABLE`-Anweisung können Sie das Grok SerDe- und ein grok-Muster ähnlich dem für Apache-Webserver-Protokolle verwenden. Im Gegensatz zu Apache-Protokollen verwendet das grok-Muster `%{DATA:user_id}` für das dritte Feld anstelle von `%{USERNAME:user_id}`, um das Vorhandensein des umgekehrten Schrägstrichs in `domain\user_id` zu berücksichtigen. Weitere Informationen zur Verwendung des Grok SerDe finden Sie unter Angepasste Grok-Classifier schreiben im AWS Glue-Entwicklerhandbuch.

So erstellen Sie eine Tabelle in Athena für IIS-NCSA-Webserverprotokolle

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Kopieren Sie die folgende DDL-Anweisung und fügen Sie sie in den Abfrage-Editor der Athena-Konsole ein: Ändern Sie die Werte in `LOCATION 's3://bucket-name/iis-ncsa-logs/'`, um auf Ihre IIS-NCSA-Protokolle in Amazon S3 zu verweisen.

```
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://bucket-name/iis-ncsa-logs/';
```

3. Führen Sie die Abfrage in der Athena-Konsole aus, um die `iis_ncsa_logs`-Tabelle zu registrieren. Wenn die Abfrage abgeschlossen ist, können Sie die Protokolle aus Athena abfragen.

Beispielauswahlabfragen für IIS-NCSA-Protokolle auswählen

Example – Filterung nach 404-Fehlern

Die folgende Beispielabfrage wählt die Empfangszeit der Anforderung, den Text der Clientanforderung und den Serverstatuscode aus der `iis_ncsa_logs`-Tabelle aus. Die `WHERE`-Klausel filtert nach HTTP-Statuscode `404` (Seite nicht gefunden)

```
SELECT request_received_time, client_request, server_status
FROM iis_ncsa_logs
WHERE server_status = '404'
```

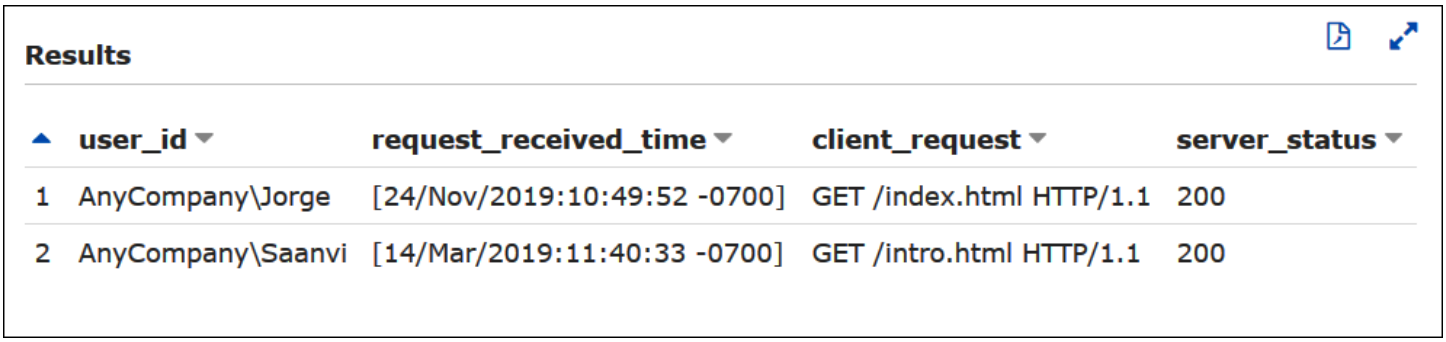
Das folgende Image zeigt die Ergebnisse der Abfrage im Athena-Abfrage-Editor.

Example – Filtern nach erfolgreichen Anforderungen aus einer bestimmten Domäne

Die folgende Beispielabfrage wählt die Benutzer-ID, die Empfangszeit der Anforderung, den Text der Clientanforderung und den Serverstatuscode aus der `iis_ncsa_logs`-Tabelle aus. Die `WHERE`-Klausel filtert nach Anforderungen mit dem HTTP-Statuscode `200` (erfolgreich) von Benutzern in der `AnyCompany`-Domäne.

```
SELECT user_id, request_received_time, client_request, server_status
FROM iis_ncsa_logs
WHERE server_status = '200' AND user_id LIKE 'AnyCompany%'
```

Das folgende Image zeigt die Ergebnisse der Abfrage im Athena-Abfrage-Editor.



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

Athena-ACID-Transaktionen verwenden

Der Begriff „ACID-Transaktionen“ bezieht sich auf eine Reihe von Eigenschaften (Atomizität, Konsistenz, Isolation und Dauerhaftigkeit), die die Datenintegrität bei Datenbanktransaktionen gewährleisten. ACID-Transaktionen ermöglichen es mehreren Benutzern, Amazon-S3-Objekte gleichzeitig und zuverlässig atomar hinzuzufügen und zu löschen, während alle vorhandenen Abfragen isoliert werden, indem die Lesekonsistenz für Abfragen gegen den Data Lake beibehalten wird. Athena-ACID-Transaktionen fügen der Athena-SQL-Datenmanipulationssprache (DML) Unterstützung für Einfügungs-, Lösch-, Aktualisierungs- und Zeitreiseoperationen mit einer einzigen Tabelle hinzu. Sie und mehrere gleichzeitige Benutzer können Athena-ACID-Transaktionen verwenden, um zuverlässige Änderungen an Amazon-S3-Daten auf Zeilenebene vorzunehmen. Athena-Transaktionen verwalten automatisch Sperren-Semantik und -koordination und erfordern keine benutzerdefinierte Datensatzsperrlösung.

Athena-ACID-Transaktionen und vertraute SQL-Syntax vereinfachen Aktualisierungen Ihrer Geschäfts- und regulatorischen Daten. Um beispielsweise auf eine Datenlöschanforderung zu antworten, können Sie einen `DELETE`-SQL-Vorgang ausführen. Um manuelle Datensatzkorrekturen vorzunehmen, können Sie eine einzelne `UPDATE`-Anweisung verwenden. Um kürzlich gelöschte Daten wiederherzustellen, können Sie Zeitreiseabfragen mit einer `SELECT`-Anweisung verwenden.

Da sie auf freigegebenen Tabellenformaten basieren, sind Athena-ACID-Transaktionen mit anderen Services und Engines wie Amazon EMR und Apache Spark kompatibel, die auch gemeinsame Tabellenformate unterstützen.

Athena-Transaktionen sind über die Athena-Konsole, API-Operationen sowie ODBC- und JDBC-Treiber verfügbar.

Themen

Delta-Lake-Tabellen von Linux Foundation abfragen

Linux Foundation Delta Lake ist ein Tabellenformat für Big-Data-Analytik. Sie können Amazon Athena verwenden, um direkt in Amazon S3 gespeicherte Delta-Lake-Tabellen zu lesen, ohne Manifest-Dateien generieren oder die `MSCK REPAIR`-Anweisung ausführen zu müssen.

Das Delta-Lake-Format speichert die Mindest- und Höchstwerte pro Spalte jeder Datendatei. Die Athena-Implementierung verwendet diese Informationen, um das Überspringen von Dateien bei Prädikaten zu ermöglichen, um unerwünschte Dateien aus der Berücksichtigung zu entfernen.

Überlegungen und Einschränkungen

Die Delta-Lake-Unterstützung in Athena weist die folgenden Einschränkungen auf:

- Nur Tabellen mit AWS Glue-Katalog – Die native Delta-Lake-Unterstützung wird nur durch Tabellen unterstützt, die mit AWS Glue registriert sind. Wenn Sie über eine Delta-Lake-Tabelle verfügen, die bei einem anderen Metastore registriert ist, können Sie diese trotzdem behalten und als Ihren primären Metastore behandeln. Da Delta Lake-Metadaten im Dateisystem (z. B. in Amazon S3) und nicht im Metaspeicher gespeichert sind, benötigt Athena nur die Speicherort-Eigenschaft in AWS Glue, um aus Ihren Delta-Lake-Tabellen zu lesen.
- V3 engine only (Nur V3-Engine) – Delta-Lake-Abfragen werden nur auf der Athena-Engine-Version 3 unterstützt. Sie müssen sicherstellen, dass die von Ihnen erstellte Arbeitsgruppe für die Verwendung der Athena-Engine-Version 3 konfiguriert ist.
- No time travel support (Keine Unterstützung für Zeitreisen) – Anfragen, die die Zeitreisefunktionen von Delta Lake nutzen, werden nicht unterstützt.
- Read only (Schreibgeschützt) – Schreiben von DML-Anweisungen wie `UPDATE`, `INSERT` oder `DELETE` werden nicht unterstützt.
- Lake-Formation-Unterstützung – Die Lake-Formation-Integration ist für Delta-Lake-Tabellen erhältlich, deren Schema mit AWS Glue synchron ist. Weitere Informationen finden Sie unter AWS

Lake Formation mit Amazon Athena verwenden und Einrichten von Berechtigungen für eine Delta-Lake-Tabelle im AWS Lake Formation-Entwicklerhandbuch.

- Limited DDL support (Eingeschränkte DDL-Unterstützung) – Die folgenden DDL-Anweisungen werden unterstützt: `CREATE EXTERNAL TABLE`, `SHOW COLUMNS`, `SHOW TBLPROPERTIES`, `SHOW PARTITIONS`, `SHOW CREATE TABLE` und `DESCRIBE`. Informationen zur Verwendung der `CREATE EXTERNAL TABLE`-Anweisung finden Sie im Erste Schritte-Abschnitt.
- Überspringen von S3-Glacier-Objekten wird nicht unterstützt – Wenn sich Objekte in der Delta-Lake-Tabelle der Linux Foundation in einer Amazon-S3-Glacier-Speicherklasse befinden, hat das Setzen der Tabelleneigenschaft `read_restored_glacier_objects` auf `false` keine Auswirkung.

  Angenommen, Sie führen den folgenden Befehl aus:

  ```
  ALTER TABLE table_name SET TBLPROPERTIES ('read_restored_glacier_objects' = 'false')
  ```

  Bei Iceberg- und Delta-Lake-Tabellen erzeugt der Befehl den Fehler Unsupported table property key: read_restored_glacier_objects. Bei Hudi-Tabellen erzeugt der `ALTER TABLE`-Befehl keinen Fehler, aber Amazon-S3-Glacier-Objekte werden immer noch nicht übersprungen. Beim Ausführen von `SELECT`-Abfragen nach dem `ALTER TABLE`-Befehl werden weiterhin alle Objekte zurückgegeben.

Unterstützte nicht partitionierte Spalten-Datentypen

Für nicht partitionierte Spalten werden alle von Athena unterstützten Datentypen außer `CHAR` unterstützt (`CHAR` wird im Delta-Lake-Protokoll selbst nicht unterstützt). Die unterstützten Datentypen beinhalten:

```
boolean
tinyint
smallint
integer
bigint
double
float
decimal
varchar
string
binary
```

```
date
timestamp
array
map
struct
```

Unterstützte partitionierte Spalten-Datentypen

Für Partitionsspalten unterstützt Athena Tabellen mit den folgenden Datentypen:

```
boolean
integer
smallint
tinyint
bigint
decimal
float
double
date
timestamp
varchar
```

Weitere Informationen zu den Datentypen in Athena finden Sie unter Datentypen in Amazon Athena.

Erste Schritte

Um abfragbar zu sein, muss Ihre Delta-Lake-Tabelle in AWS Glue vorhanden sein. Wenn sich Ihre Tabelle in Amazon S3 befindet, aber nicht in AWS Glue, führen Sie eine `CREATE EXTERNAL TABLE`-Anweisung mit der folgenden Syntax aus. Wenn Ihre Tabelle bereits in AWS Glue vorhanden ist (z. B. weil Sie Apache Spark oder eine andere Engine mit AWS Glue verwenden), können Sie diesen Schritt überspringen.

```
CREATE EXTERNAL TABLE
  [db_name.]table_name
  LOCATION 's3://DOC-EXAMPLE-BUCKET/your-folder/'
  TBLPROPERTIES ('table_type' = 'DELTA')
```

Notieren Sie sich die Auswirkungen von Spaltendefinitionen, SerDe Bibliotheken und anderen Tabelleneigenschaften. Im Gegensatz zu herkömmlichen Hive-Tabellen werden die Metadaten der Delta Lake-Tabelle aus dem Delta-Lake-Transaktionsprotokoll abgeleitet und direkt mit AWS Glue synchronisiert.

> **Note**
>
> Für Delta-Lake-Tabellen sind `CREATE TABLE`-Aussagen, die mehr als die Eigenschaft `LOCATION` und `table_type` enthalten, nicht zulässig.

Lesen von Delta Lake-Tabellen

Verwenden Sie die `SELECT`-Standard-SQL-Syntax, um eine Delta-Lake-Tabelle abzufragen:

```
[ 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 ] ]
```

Weitere Informationen zur `SELECT`-Syntax finden Sie unter SELECT in der Athena-Dokumentation.

Das Delta-Lake-Format speichert die Mindest- und Höchstwerte pro Spalte jeder Datendatei. Athena verwendet diese Informationen, um das Überspringen von Dateien bei Prädikaten zu ermöglichen, um unnötige Dateien aus der Berücksichtigung zu entfernen.

Synchronisieren von Delta-Lake-Metadaten

Athena synchronisiert Tabellenmetadaten, einschließlich Schema, Partitionsspalten und Tabelleneigenschaften, mit AWS Glue, wenn Sie Athena zum Erstellen Ihrer Delta-Lake-Tabelle verwenden. Im Laufe der Zeit können diese Metadaten deren Synchronisierung mit den zugrunde liegenden Tabellenmetadaten im Transaktionsprotokoll verlieren. Um Ihre Tabelle auf dem neuesten Stand zu halten, können Sie eine der folgenden Optionen auswählen:

- Verwenden Sie den AWS Glue-Crawler für Delta-Lake-Tabellen. Weitere Informationen finden Sie unter Einführung der nativen Delta-Lake-Tabellenunterstützung mit AWS Glue-Crawlern im AWS-Big-Data-Blog und Planen eines AWS Glue-Crawlers im AWS Glue-Entwicklerhandbuch.
- Die Tabelle fallen lassen und in Athena neu erstellen.
- Verwenden Sie das SDK, die CLI oder die AWS Glue-Konsole, um das Schema in manuell in AWS Glue zu aktualisieren.

Beachten Sie, dass für die folgenden Features Ihr AWS Glue-Schema immer dasselbe Schema wie das Transaktionsprotokoll haben muss:

- Lake Formation
- Ansichten
- Zeilen- und Spaltenfilter

Wenn Ihr Workflow keine dieser Funktionen erfordert und Sie diese Kompatibilität nicht aufrechterhalten möchten, können Sie `CREATE TABLE` DDL in Athena verwenden und dann den Amazon S3-Pfad als SerDe Parameter in hinzufügenAWS Glue.

Zur Erstellung einer Delta-Lake-Tabelle mit den Athena- und AWS Glue-Konsolen

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Verwenden Sie im Athena-Abfrage-Editor die folgende DDL, um Ihre Delta-Lake-Tabelle zu erstellen. Beachten Sie, dass bei Verwendung dieser Methode der Wert für `TBLPROPERTIES` `'spark.sql.sources.provider' = 'delta'` sein muss und nicht `'table_type' = 'delta'`.

   Beachten Sie, dass dasselbe Schema (mit einer einzigen Spalte namens `col` vom Typ `array<string>`) eingefügt wird, wenn Sie Apache Spark (Athena für Apache Spark) oder die meisten anderen Engines verwenden, um Ihre Tabelle zu erstellen.

   ```
   CREATE EXTERNAL TABLE
      [db_name.]table_name(col array<string>)
      LOCATION 's3://DOC-EXAMPLE-BUCKET/your-folder/'
      TBLPROPERTIES ('spark.sql.sources.provider' = 'delta')
   ```
3. Öffnen Sie die AWS Glue-Konsole unter https://console.aws.amazon.com/glue/.
4. Wählen Sie im Navigationsbereich unter Data Catalog die Option Tabellen.
5. Wählen Sie in der Tabellenliste den Link für Ihre Tabelle aus.
6. Wählen Sie auf der Seite für die Tabelle Aktionen und Tabelle bearbeiten aus.
7. Fügen Sie im Abschnitt Serde-Parameter den Schlüssel **`path`** mit dem Wert **`s3://DOC-EXAMPLE-BUCKET/your-folder/`** hinzu.
8. Wählen Sie Speichern.

Weitere Informationen finden Sie auch unter

Eine Erläuterung der Verwendung von Delta Lake-Tabellen mit AWS Glue und deren Abfrage mit Athena finden Sie unter Verarbeiten Sie UPSERT-Datenoperationen mit der Open-Source-Software Delta Lake und AWS Glue im AWS-Big-Data-Blog.

Verwenden von Athena zum Abfragen von Apache-Hudi-Datensätzen

Apache Hudi ist ein Open-Source-Datenmanagement-Framework, das die inkrementelle Datenverarbeitung vereinfacht. Einfüge-, Aktualisierungs-, Upsert- und Löschaktionen auf Datensatzebene werden viel granularer verarbeitet, wodurch der Overhead reduziert wird. `Upsert` bezieht sich auf die Möglichkeit, Datensätze in einen vorhandenen Datensatz einzufügen, wenn sie noch nicht vorhanden sind, oder sie zu aktualisieren, wenn dies der Fall ist.

Hudi verarbeitet Dateneinfügungs- und Aktualisierungsereignisse, ohne viele kleine Dateien zu erstellen, die Leistungsprobleme bei der Analyse verursachen können. Apache Hudi verfolgt automatisch Änderungen und führt Dateien zusammen, damit sie die optimale Größe behalten. Dadurch entfällt die Notwendigkeit, benutzerdefinierte Lösungen zu erstellen, die viele kleine Dateien überwachen und in weniger große Dateien umschreiben.

Hudi-Datensätze eignen sich für die folgenden Anwendungsfälle:

- Einhaltung von Datenschutzbestimmungen wie der Datenschutz-Grundverordnung (GDPR) und dem California Consumer Privacy Act (CCPA), die das Recht der Menschen auf Entfernung personenbezogener Daten oder Änderung der Verwendung ihrer Daten durchsetzen.
- Arbeiten mit Streaming-Daten von Sensoren und anderen IoT-Geräten (Internet of Things), die bestimmte Dateneinfüge- und Aktualisierungsereignisse erfordern.
- Implementieren eines Change-Data-Capture (CDC)-Systems.

Von Hudi verwaltete Datensätze werden mithilfe von offenen Speicherformaten in Amazon S3 gespeichert. Derzeit kann Athena komprimierte Hudi-Datensätze lesen, aber keine Hudi-Daten schreiben. Athena unterstützt bis zu Hudi-Version 0.8.0 mit Athena-Engine-Version 2 und Hudi-Version 0.14.0 mit Athena-Engine-Version 3. Änderungen sind vorbehalten. Athena kann die Lesekompatibilität mit Tabellen, die mit späteren Versionen von Hudi erstellt wurden, nicht garantieren. Weitere Informationen zum Athena-Engine-Versioning finden Sie unter Athena-Engine-Versionierung. Weitere Informationen zu Hudi-Features und Versionsverwaltung finden Sie in der Hudi-Dokumentation auf der Apache-Website.

Hudi-Datensatz-Tabellen-Typen

Ein Hudi-Datensatz kann einen der folgenden Typen haben:

- Copy on Write (CoW, Beim Schreiben kopieren) – Daten werden in einem spaltenbasierten Format (Parquet) gespeichert, und jedes Update erstellt während eines Schreibvorgangs eine neue Version von Dateien.
- Merge on Read (MoR, Beim Lesen zusammenführen) – Daten werden mit einer Kombination aus spalten- (Parquet) und zeilenbasierten (Avro) Formaten gespeichert. Updates werden in zeilenbasierten `delta`-Dateien protokolliert und nach Bedarf komprimiert, um neue Versionen der Spaltendateien zu erstellen.

Bei CoW-Datasets wird jedes Mal, wenn ein Datensatz aktualisiert wird, die Datei, die den Datensatz enthält, mit den aktualisierten Werten neu geschrieben. Bei einem MoR-Datensatz schreibt Hudi jedes Mal, wenn es eine Aktualisierung gibt, nur die Zeile für den geänderten Datensatz. MoR eignet sich besser für schreib- oder änderungsintensive Workloads mit weniger Lesevorgängen. CoW eignet sich besser für leseintensive Workloads für Daten, die sich seltener ändern.

Hudi bietet drei Abfragetypen für den Zugriff auf die Daten:

- Snapshots – Abfragen, die den neuesten Snapshot der Tabelle ab einer bestimmten Commit- oder Komprimierungsaktion anzeigen. Bei MoR-Tabellen stellen Snapshot-Abfragen den neuesten Status der Tabelle dar, indem die Basis- und Deltadateien des letzten Datei-Slices zum Zeitpunkt der Abfrage zusammengeführt werden.
- Inkrementelle Abfragen – Abfragen sehen nur neue Daten, die seit einem bestimmten Commit/Komprimierung in die Tabelle geschrieben wurden. Dies bietet effektiv Änderungsströme, um inkrementelle Data-Pipelines zu ermöglichen.
- Lesen von optimierten Abfragen – Bei MoR-Tabellen sehen Abfragen die neuesten komprimierten Daten. Bei CoW-Tabellen sehen Abfragen die neuesten festgeschriebenen Daten.

Die folgende Tabelle zeigt die möglichen Hudi-Abfragetypen für jeden Tabellentyp.

Tabellentyp	Mögliche Hudi-Abfragetypen
Kopieren Sie beim Schreiben	Snapshot, inkrementell
Beim Lesen zusammenführen	Snapshot, inkrementell, leseoptimiert

Derzeit unterstützt Athena-Snapshot-Abfragen und lese-optimierte Abfragen, aber keine inkrementellen Abfragen. In MoR-Tabellen werden alle Daten, die leseoptimierten Abfragen ausgesetzt sind, komprimiert. Dies bietet eine gute Leistung, enthält jedoch nicht die neuesten Delta-Commits. Snapshot-Abfragen enthalten die aktuellsten Daten, verursachen jedoch einen gewissen Rechenaufwand, wodurch diese Abfragen weniger leistungsfähig sind.

Weitere Informationen zu den Kompromissen zwischen Tabellen- und Abfragetypen finden Sie unter Tabellen- und Abfragetypen in der Apache-Hudi-Dokumentation.

Hudi Terminologieänderung: Ansichten sind jetzt Abfragen

Ab Version 0.5.1 hat Apache Hudi einige seiner Terminologie geändert. Was früher Ansichten waren, werden in späteren Versionen Abfragen genannt. In der folgenden Tabelle werden die Änderungen zwischen dem alten und dem neuen Term zusammengefasst.

Alter Begriff	Neuer Begriff
CoW: Lese-optimierte Ansicht MoR: Echtzeitansicht	Snapshotabfragen
Inkrementelle Ansicht	Inkrementelle Abfrage
MoR-Lese-optimierte Ansicht	Leseoptimierte Abfrage

Tabellen aus Bootstrap-Operation

Ab Apache Hudi Version 0.6.0 bietet das Bootstrap-Feature eine bessere Leistung mit vorhandenen Parquet-Datensätzen. Anstatt den Datensatz neu zu schreiben, kann ein Bootstrap-Vorgang nur Metadaten generieren und den Datensatz an Ort und Stelle belassen.

Sie können Athena verwenden, um Tabellen aus einem Bootstrap-Vorgang abzufragen, genau wie andere Tabellen, die auf Daten in Amazon S3 basieren. Geben Sie in Ihrer `CREATE TABLE`-Anweisung den Hudi-Tabellenpfad in Ihrer `LOCATION`-Klausel an.

Weitere Informationen zum Erstellen von Hudi-Tabellen mithilfe der Bootstrap-Operation in Amazon EMR finden Sie im Artikel Neue Funktionen von Apache Hudi, die in Amazon EMR verfügbar sind, im Big Data-Blog. AWS

Liste der Hudi-Metadaten

Apache Hudi verfügt über eine Metadatentabelle, die Indizierungs-Feature für eine verbesserte Leistung enthält, wie z. B. das Auflisten von Dateien, das Überspringen von Daten mithilfe von Spaltenstatistiken und einen auf Bloomfiltern basierenden Index.

Von diesen Features unterstützt Athena derzeit nur den Dateiauflistungsindex. Der Dateiauflistungsindex eliminiert Dateisystemaufrufe wie „Dateien auflisten“, indem er die Informationen aus einem Index abruft, der die Zuordnung von Partitionen zu Dateien verwaltet. Dadurch entfällt die Notwendigkeit, jede einzelne Partition unter dem Tabellenpfad rekursiv aufzulisten, um einen Überblick über das Dateisystem zu erhalten. Wenn Sie mit großen Datensätzen arbeiten, reduziert diese Indizierung die Latenz, die sonst beim Abrufen der Dateiliste bei Schreib- und Abfragen auftreten würde, drastisch. Außerdem werden Engpässe wie die Drosselung von Anforderungslimits Amazon-S3-`LIST`-Aufrufen vermieden.

> **Note**
>
> Athena unterstützt derzeit weder das Überspringen von Daten noch die Bloom-Filter-Indizierung.

Die Hudi-Metadatentabelle aktivieren

Die auf Metadatentabellen basierende Dateiauflistung ist standardmäßig deaktiviert. Um die Hudi-Metadatentabelle und die zugehörige Dateiauflistungsfunktion zu aktivieren, setzen Sie die `hudi.metadata-listing-enabled`-Tabelleneigenschaft auf `TRUE`.

Beispiel

Im folgenden `ALTER TABLE SET TBLPROPERTIES`-Beispiel wird die Metadatentabelle in der `partition_cow`-Beispieltabelle aktiviert.

```
ALTER TABLE partition_cow SET TBLPROPERTIES('hudi.metadata-listing-enabled'='TRUE')
```

Überlegungen und Einschränkungen

- Athena unterstützt keine inkrementellen Abfragen.
- Athena unterstützt CTAS oder INSERT INTO auf Hudi-Daten nicht. Wenn Sie Athena-Unterstützung beim Schreiben von Hudi-Datensätzen wünschen, senden Sie Feedback an `<athena-feedback@amazon.com>`.

  Weitere Informationen zum Schreiben von Hudi-Daten finden Sie in den folgenden Ressourcen:

  - Arbeiten mit einem Hudi-Datensatz im Amazon-EMR-Versionshandbuch.
  - Schreiben von Daten in der Apache-Hudi-Dokumentation.
- Die Verwendung von MSCK REPAIR TABLE auf Hudi-Tabellen in Athena wird nicht unterstützt. Wenn Sie eine Hudi-Tabelle laden müssen, die nicht in erstellt wurde, verwenden Sie. AWS GlueALTER TABLE ADD PARTITION
- Überspringen von S3-Glacier-Objekten wird nicht unterstützt – Wenn sich Objekte in der Apache–Hudi-Tabelle in einer Amazon-S3-Glacier-Speicherklasse befinden, hat das Setzen der Tabelleneigenschaft `read_restored_glacier_objects` auf `false` keine Auswirkung.

  Angenommen, Sie führen den folgenden Befehl aus:

  ```
  ALTER TABLE table_name SET TBLPROPERTIES ('read_restored_glacier_objects' = 'false')
  ```

  Bei Iceberg- und Delta-Lake-Tabellen erzeugt der Befehl den Fehler Unsupported table property key: read_restored_glacier_objects. Bei Hudi-Tabellen erzeugt der `ALTER TABLE`-Befehl keinen Fehler, aber Amazon-S3-Glacier-Objekte werden immer noch nicht übersprungen. Beim Ausführen von `SELECT`-Abfragen nach dem `ALTER TABLE`-Befehl werden weiterhin alle Objekte zurückgegeben.

Video

Das folgende Video zeigt, wie Sie mit Amazon Athena ein leseoptimierter Apache-Hudi-Datensatz in Ihrem Amazon-S3-basierten Data Lake abfragen können.

Abfragen von Apache-Hudi-Datensätzen mit Amazon Athena

Erstellen von Hudi-Tabellen

Dieser Abschnitt enthält Beispiele für CREATE-TABLE-Anweisungen in Athena für partitionierte und nicht partitionierte Tabellen von Hudi-Daten.

Wenn Sie bereits Hudi-Tabellen erstellt haben AWS Glue, können Sie diese direkt in Athena abfragen. Wenn Sie in Athena partitionierte Hudi-Tabellen erstellen, müssen Sie `ALTER TABLE ADD PARTITION` ausführen, um die Hudi-Daten zu laden, bevor Sie sie abfragen können.

Copy on Write (CoW), Tabellenbeispiele erstellen

Nicht partitionierte CoW-Tabelle

Im folgenden Beispiel wird eine unpartitionierte CoW-Tabelle in Athena erstellt.

```
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://bucket/folder/non_partition_cow/'
```

Partitionierte CoW-Tabelle

Im folgenden Beispiel wird eine partitionierte CoW-Tabelle in Athena erstellt.

```
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://bucket/folder/partition_cow/'
```

Im folgenden `ALTER TABLE ADD PARTITION`-Beispiel werden der `partition_cow`-Beispieltabelle zwei Partitionen hinzugefügt.

```
ALTER TABLE partition_cow ADD
  PARTITION (event_type = 'one') LOCATION 's3://bucket/folder/partition_cow/one/'
  PARTITION (event_type = 'two') LOCATION 's3://bucket/folder/partition_cow/two/'
```

„Beim Lesen zusammenführen (MoR)“ Tabellenbeispiele erstellen

Hudi erstellt zwei Tabellen im Metastore für MoR: eine Tabelle für Snapshot-Abfragen und eine Tabelle für leseoptimierte Abfragen. Beide Tabellen können abgefragt werden. In Hudi-Versionen vor 0.5.1 hatte die Tabelle für leseoptimierte Abfragen den Namen, den Sie beim Erstellen der Tabelle angegeben haben. Ab Hudi-Version 0.5.1 wird dem Tabellennamen standardmäßig ein `_ro` angehängt. Der Name der Tabelle für Snapshot-Abfragen ist der Name, den Sie mit angehängtem `_rt` angegeben haben.

Nicht partitionierte Merge-on-Read (MoR)-Tabelle

Im folgenden Beispiel wird eine nicht partitionierte MoR-Tabelle in Athena für leseoptimierte Abfragen erstellt. Beachten Sie, dass leseoptimierte Abfragen das Eingabeformat `HoodieParquetInputFormat` verwenden.

```
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://bucket/folder/nonpartition_mor/'
```

Im folgenden Beispiel wird eine nicht partitionierte MoR-Tabelle in Athena für Snapshot-Abfragen erstellt. Verwenden Sie für Snapshot-Abfragen das Eingabeformat `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://bucket/folder/nonpartition_mor/'
```

Partitionierte Merge-on-Read-(MoR)-Tabelle

Im folgenden Beispiel wird eine partitionierte MoR-Tabelle in Athena für leseoptimierte Abfragen erstellt.

```
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://bucket/folder/partition_mor/'
```

Im folgenden `ALTER TABLE ADD PARTITION`-Beispiel werden der `partition_mor`-Beispieltabelle zwei Partitionen hinzugefügt.

```
ALTER TABLE partition_mor ADD
  PARTITION (event_type = 'one') LOCATION 's3://bucket/folder/partition_mor/one/'
  PARTITION (event_type = 'two') LOCATION 's3://bucket/folder/partition_mor/two/'
```

Im folgenden Beispiel wird eine partitionierte MoR-Tabelle in Athena für Snapshot-Abfragen erstellt.

```
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://bucket/folder/partition_mor/'
```

Ebenso werden im folgenden `ALTER TABLE ADD PARTITION`-Beispiel der `partition_mor_rt`-Beispieltabelle zwei Partitionen hinzugefügt.

```
ALTER TABLE partition_mor_rt ADD
  PARTITION (event_type = 'one') LOCATION 's3://bucket/folder/partition_mor/one/'
  PARTITION (event_type = 'two') LOCATION 's3://bucket/folder/partition_mor/two/'
```

Weitere Informationen finden Sie auch unter

- Informationen zur Verwendung von AWS Glue benutzerdefinierten Konnektoren und AWS Glue 2.0-Jobs zum Erstellen einer Apache Hudi-Tabelle, die Sie mit Athena abfragen können, finden Sie unter Schreiben in Apache Hudi-Tabellen mithilfe eines AWS Glue benutzerdefinierten Konnektors im AWS Big Data-Blog.
- Einen Artikel über die Verwendung von Apache Hudi und Amazon Athena zur Erstellung eines Datenverarbeitungs-Frameworks für einen Data Lake finden Sie unter Simplify Operational Data Processing in Data Lakes using AWS Glue und Apache Hudi im AWS Big Data-Blog. AWS Glue

Apache-Iceberg-Tabellen verwenden

Athena unterstützt Lese-, Zeitreise-, Schreib- und DDL-Abfragen für Apache-Iceberg-Tabellen, die das Apache-Parquet-Format für Daten und den - AWS Glue Katalog für ihren Metastore verwenden.

Apache Iceberg ist ein offenes Tabellenformat für sehr große analytische Datensätze. Iceberg verwaltet große Sammlungen von Dateien als Tabellen und unterstützt moderne analytische Data-Lake-Operationen wie Einfüge-, Aktualisierungs-, Löschen- und Zeitreiseabfragen auf Datensatzebene. Die Iceberg-Spezifikation ermöglicht eine nahtlose Tabellenentwicklung wie Schema- und Partitionsentwicklung, und ihr Design ist für die Verwendung auf Amazon S3 optimiert. Iceberg trägt auch dazu bei, die Datenkorrektheit in gleichzeitigen Schreibszenarien zu gewährleisten.

Weitere Informationen zu Apache Iceberg finden Sie unter http://iceberg.apache.org/.

Überlegungen und Einschränkungen

Die Athena-Unterstützung für Iceberg-Tabellen weist die folgenden Einschränkungen auf:

- Nur Tabellen mit AWS Glue Katalog – Nur Iceberg-Tabellen, die anhand des AWS Glue Katalogs basierend auf Spezifikationen erstellt wurden, die von der Open-Source-Glue-Katalogimplementierung definiert wurden, werden von Athena unterstützt.
- Unterstützung für Tabellensperren AWS Glue nur durch – Im Gegensatz zur Open-Source-Glue-Katalogimplementierung, die benutzerdefinierte Plug-In-Sperren unterstützt, unterstützt Athena nur AWS Glue optimistische Sperren. Die Verwendung von Athena zum Ändern einer Iceberg-Tabelle mit einer anderen Sperren-Implementierung führt zu potenziellen Datenverlust und bricht Transaktionen ab.
- Unterstützte Dateiformate – Die Unterstützung des Iceberg-Dateiformats in Athena hängt von der Athena-Engine-Version ab, wie in der folgenden Tabelle dargestellt.

Athena-Engine-Version	Parquet	ORC	Avro
2	Ja	Nein	Nein
3	Ja	Ja	Ja

- Iceberg-v2-Tabellen – Athena erstellt und arbeitet nur auf Iceberg-v2-Tabellen. Den Unterschied zwischen v1- und v2-Tabellen finden Sie unter Formatversionsänderungen in der Apache-Iceberg-Dokumentation.

- Anzeige von Zeittypen ohne Zeitzone – Die Zeit und der Zeitstempel ohne Zeitzonentypen werden in UTC angezeigt. Wenn die Zeitzone in einem Filterausdruck für eine Zeitspalte nicht angegeben ist, wird UTC verwendet.
- Zeitstempelbezogene Datengenauigkeit – Obwohl Iceberg Mikrosekunden-Genauigkeit für den Zeitstempel-Datentyp unterstützt, unterstützt Athena nur Millisekunden-Genauigkeit für Zeitstempel sowohl beim Lesen als auch beim Schreiben. Für Daten in zeitbezogenen Spalten, die bei manuellen Verdichtungsvorgängen neu geschrieben werden, behält Athena nur die Millisekundengenauigkeit bei.
- Nicht unterstützte Operationen – Die folgenden Athena-Operationen werden für Iceberg-Tabellen nicht unterstützt.
  - ALTER TABLE SET LOCATION
- Ansichten – Verwenden Sie `CREATE VIEW` zum Erstellen von Athena-Ansichten, wie unter Arbeiten mit Ansichten beschrieben. Wenn Sie daran interessiert sind, die Iceberg-Ansichtsspezifikation zum Erstellen von Ansichten zu verwenden, wenden Sie sich an athena-feedback@amazon.com.
- TTF-Verwaltungsbefehle werden in nicht unterstützt AWS Lake Formation – Obwohl Sie Lake Formation verwenden können, um Lesezugriffsberechtigungen für TransactionTable Formate (TTFs) wie Apache Iceberg, Apache Hudi und Linux Foundation Delta Lake zu verwalten, können Sie Lake Formation nicht verwenden`MERGE`, um Berechtigungen für Operationen wie `VACUUM`, `UPDATE` oder `OPTIMIZE` mit diesen Tabellenformaten zu verwalten. Weitere Informationen zur Integration von Lake Formation mit Athena finden Sie unter Verwenden von AWS Lake Formation mit Amazon Athena im AWS Lake Formation -Entwicklerhandbuch.
- Partitionierung nach verschachtelten Feldern – Die Partitionierung nach verschachtelten Feldern wird nicht unterstützt. Wenn Sie dies versuchen, wird die Meldung NOT_SUPPORTED: Die Unterteilung nach verschachtelten Feldern wird nicht unterstützt: *`column_name.nested_field_name`* erstellt.
- Überspringen von S3-Glacier-Objekten wird nicht unterstützt – Wenn sich Objekte in der Apache-Iceberg-Tabelle in einer Amazon-S3-Glacier-Speicherklasse befinden, hat das Setzen der Tabelleneigenschaft `read_restored_glacier_objects` auf `false` keine Auswirkung.

  Angenommen, Sie führen den folgenden Befehl aus:

```
ALTER TABLE table_name SET TBLPROPERTIES ('read_restored_glacier_objects' = 'false')
```

Bei Iceberg- und Delta-Lake-Tabellen erzeugt der Befehl den Fehler Unsupported table property key: read_restored_glacier_objects. Bei Hudi-Tabellen erzeugt der `ALTER TABLE`-Befehl keinen Fehler, aber Amazon-S3-Glacier-Objekte werden immer noch nicht übersprungen. Beim Ausführen von `SELECT`-Abfragen nach dem `ALTER TABLE`-Befehl werden weiterhin alle Objekte zurückgegeben.

Wenn Sie möchten, dass Athena ein bestimmtes Feature unterstützt, senden Sie Feedback an athena-feedback@amazon.com.

Themen

Erstellen von Iceberg-Tabellen

Um eine Iceberg-Tabelle für die Verwendung in Athena zu erstellen, können Sie eine -`CREATE TABLE`Anweisung verwenden, wie auf dieser Seite dokumentiert, oder Sie können einen AWS Glue -Crawler verwenden.

CREATE-TABLE-Anweisung verwenden

Athena erstellt Iceberg-v2-Tabellen. Den Unterschied zwischen v1- und v2-Tabellen finden Sie unter Formatversionsänderungen in der Apache-Iceberg-Dokumentation.

Athena `CREATE TABLE` erstellt eine Iceberg-Tabelle ohne Daten. Sie können eine Tabelle direkt aus externen Systemen wie Apache Spark abfragen, wenn die Tabelle den Open-Source-Glue-Katalog von Iceberg verwendet. Sie müssen keine externe Tabelle erstellen.

> **Warning**
>
> Das Ausführen von `CREATE EXTERNAL TABLE` führt zu der Fehlermeldung Externes Schlüsselwort wird für den Tabellentyp ICEBERG nicht unterstützt.

Um eine Iceberg-Tabelle aus Athena zu erstellen, legen Sie die `'table_type'`-Tabellen-Eigenschaft in der `'ICEBERG'`-Klausel auf `TBLPROPERTIES`, wie in der folgenden Syntaxzusammenfassung.

```
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] )
```

Informationen zu den Datentypen, die Sie in Iceberg-Tabellen abfragen können, finden Sie unter Unterstützte Datentypen für Iceberg-Tabellen in Athena.

Partitionierung

Um Iceberg-Tabellen mit Partitionen zu erstellen, verwenden Sie die `PARTITIONED BY`-Syntax. Spalten, die für die Partitionierung verwendet werden, müssen zuerst in den Spaltendeklarationen angegeben werden. Innerhalb der `PARTITIONED BY`-Klausel darf der Spaltentyp nicht enthalten sein. Sie können auch die Partitionstransformationen in der `CREATE TABLE`-Syntax definieren. Um mehrere Spalten für die Partitionierung anzugeben, trennen Sie die Spalten durch das Komma (`,`), wie im folgenden Beispiel.

```
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' )
```

Die folgende Tabelle zeigt die verfügbaren Partitionstransformationsfunktionen.

Funktion	Beschreibung	Unterstützte Typen
`year(ts)`	Partition nach Jahr	`date`, `timestamp`

Funktion	Beschreibung	Unterstützte Typen
`month(ts)`	Partition nach Monat	`date`, `timestamp`
`day(ts)`	Partition nach Tag	`date`, `timestamp`
`hour(ts)`	Partition nach Stunde	`timestamp`
`bucket(N, col)`	Partitionieren nach Hash-Werte-Mod *N*-Buckets. Dies ist das gleiche Konzept wie Hash-Bucketing für Hive-Tabellen.	`int`, `long`, `decimal`, `date`, `timestamp` , `string`, `binary`
`truncate( L, col)`	Partitionieren nach Wert gekürzt auf *L*	`int`, `long`, `decimal`, `string`

Athena unterstützt Icebergs versteckte Partitionierung. Weitere Informationen finden Sie unter Icebergs versteckte Partitionierung in der Apache-Iceberg-Dokumentation.

Tabelleneigenschaften

In diesem Abschnitt werden Tabelleneigenschaften beschrieben, die als Schlüssel-Wert-Paare in der `TBLPROPERTIES`-Klausel der Erklärung `CREATE TABLE` angegeben werden. Athena erlaubt nur eine vordefinierte Liste von Schlüssel-Wert-Paaren in den Tabelleneigenschaften zum Erstellen oder Ändern von Iceberg-Tabellen. In den folgenden Tabellen finden Sie die Tabelleneigenschaften, die Sie angeben können. Weitere Hinweise zu den Verdichtungsoptionen finden Sie unter Optimieren von Iceberg-Tabellen in diesem Dokument. Wenn Sie möchten, dass Athena eine bestimmte Open-Source-Tabellen-Konfigurationseigenschaft unterstützt, senden Sie Feedback an athena-feedback@amazon.com.

Format

Beschreibung	Dateidaten-Format

Zulässige Eigenschaftswerte	Die unterstützten Dateiformat- und Komprimierungskombinationen variieren je nach Athena-Engine-Version. Weitere Informationen finden Sie unter Unterstützung der Komprimierung von Iceberg-Tabellen nach Dateiformaten.
Standardwert	parquet

write_compression

Beschreibung	Dateikomprimierungscodec
Zulässige Eigenschaftswerte	Die unterstützten Dateiformat- und Komprimierungskombinationen variieren je nach Athena-Engine-Version. Weitere Informationen finden Sie unter Unterstützung der Komprimierung von Iceberg-Tabellen nach Dateiformaten.
Standardwert	Die standardmäßige Schreibkomprimierung variiert je nach Athena-Engine-Version. Weitere Informationen finden Sie unter Unterstützung der Komprimierung von Iceberg-Tabellen nach Dateiformaten.

optimize_rewrite_data_file_threshold

Beschreibung	Spezifische Konfiguration zur Datenoptimierung. Wenn es weniger Datendateien gibt, die eine Optimierung erfordern als der angegebene Schwellenwert, werden die Dateien nicht neu geschrieben. Dies ermöglicht die Ansammlung von mehr Datendateien, um Dateien näher an der Zielgröße zu erzeugen und unnötige Berechnungen zur Kosteneinsparung zu überspringen.
Zulässige Eigenschaftswerte	Eine positive Zahl. Muss kleiner als 50 sein.
Standardwert	5

optimize_rewrite_delete_file_threshold

Beschreibung	Spezifische Konfiguration zur Datenoptimierung. Wenn weniger Löschdateien mit einer Datendatei verknüpft sind als der Schwellenwert, wird die Datendatei nicht neu geschrieben. Dies ermöglicht die Anhäufung von mehr Löschdateien für jede Datendatei zur Kostenein sparung.
Zulässige Eigenschaftswerte	Eine positive Zahl. Muss kleiner als 50 sein.
Standardwert	2

vacuum_min_snapshots_to_keep

Beschreibung	Mindestanzahl von Snapshots, die im Hauptzweig einer Tabelle beibehalten werden sollen. Dieser Wert hat Vorrang vor der `vacuum_max_snapshot_age_seconds` -Eigenschaft. Wenn die Mindestanzahl der verbleibenden Snapshots älter als das von `vacuum_max_snapshot_age_seconds` angegebene Alter ist, werden die Snapshots beibehalten und der Wert von `vacuum_max_snapshot_age_seconds` wird ignoriert.
Zulässige Eigenschaftswerte	Eine positive Zahl.
Standardwert	1

vacuum_max_snapshot_age_seconds

Beschreibung	Höchstalter der Snapshots, die auf dem Hauptzweig beibehalten werden sollen. Dieser Wert wird ignoriert, wenn die verbleibende Mindestanzahl an Snapshots, die von `vacuum_min_snapshots_to_keep` angegeben wurden, älter als das angegebene Alter ist.

Zulässige Eigenschaftswerte	Eine positive Zahl.
Standardwert	432 000 Sekunden (5 Tage)

vacuum_max_metadata_files_to_keep

Beschreibung	Die maximale Anzahl früherer Metadatendateien, die im Hauptzweig der Tabelle aufbewahrt werden sollen.
Zulässige Eigenschaftswerte	Eine positive Zahl.
Standardwert	100

Beispiel einer CREATE-TABLE-Anweisung

Im folgenden Beispiel wird eine Iceberg-Tabelle mit drei Spalten erstellt.

```
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)

Informationen zum Erstellen einer Iceberg-Tabelle mithilfe der `CREATE TABLE AS`-Anweisung finden Sie unter CREATE TABLE AS, wobei dem Abschnitt CTAS-Tabelleneigenschaften besondere Aufmerksamkeit geschenkt werden sollte.

Einen AWS Glue -Crawler verwenden

Sie können einen AWS Glue -Crawler verwenden, um Ihre Iceberg-Tabellen automatisch in der zu registrieren AWS Glue Data Catalog. Wenn Sie aus einem anderen Iceberg-Katalog migrieren möchten, können Sie einen AWS Glue -Crawler erstellen und planen und die Amazon S3-Pfade angeben, in denen sich die Iceberg-Tabellen befinden. Sie können die maximale Tiefe der Amazon-S3-Pfade angeben, die der AWS Glue -Crawler durchqueren kann. Nachdem Sie einen AWS Glue -Crawler geplant haben, extrahiert der Crawler Schemainformationen und aktualisiert die bei jeder Ausführung AWS Glue Data Catalog mit den Schemaänderungen. Der AWS Glue Crawler unterstützt die Schemazusammenführung zwischen Snapshots und aktualisiert den neuesten Speicherort der Metadatendatei in der AWS Glue Data Catalog. Weitere Informationen finden Sie unter Data Catalog und Crawler in AWS Glue.

Verwalten von Iceberg-Tabellen

Athena unterstützt die folgenden DDL-Operationen der Tabelle für Iceberg-Tabellen.

ALTER TABLE RENAME

Umbenennen einer Tabelle.

Da die Tabellenmetadaten einer Iceberg-Tabelle in Amazon S3 gespeichert sind, können Sie den Datenbank- und Tabellennamen einer von Iceberg verwalteten Tabelle aktualisieren, ohne die zugrunde liegenden Tabelleninformationen zu beeinträchtigen.

Syntax

```
ALTER TABLE [db_name.]table_name RENAME TO [new_db_name.]new_table_name
```

Beispiel

```
ALTER TABLE my_db.my_table RENAME TO my_db2.my_table2
```

FESTGELEGTE TABELLENEIGENSCHAFTEN ÄNDERN

Fügt Eigenschaften zu einer Iceberg-Tabelle hinzu und legt deren zugewiesene Werte fest.

In Übereinstimmung mit Iceberg-Spezifikationen werden Tabelleneigenschaften in der Metadatendatei der Iceberg-Tabelle gespeichert und nicht in AWS Glue. Athena akzeptiert keine benutzerdefinierten Tabelleneigenschaften. Weitere Informationen finden Sie im

Tabelleneigenschaften-Abschnitt für zulässige Schlüssel-Wert-Paare. Wenn Sie möchten, dass Athena eine bestimmte Open-Source-Tabellen-Konfigurationseigenschaft unterstützt, senden Sie Feedback an athena-feedback@amazon.com.

Syntax

```
ALTER TABLE [db_name.]table_name SET TBLPROPERTIES ('property_name' =
 'property_value' [ , ... ])
```

Beispiel

```
ALTER TABLE iceberg_table SET TBLPROPERTIES (
  'format'='parquet',
  'write_compression'='snappy',
  'optimize_rewrite_delete_file_threshold'='10'
)
```

NICHT FESTGELEGTE TABELLENEIGENSCHAFTEN ÄNDERN

Lässt bestehende Eigenschaften von einer Iceberg-Tabelle fallen.

Syntax

```
ALTER TABLE [db_name.]table_name UNSET TBLPROPERTIES ('property_name' [ , ... ])
```

Beispiel

```
ALTER TABLE iceberg_table UNSET TBLPROPERTIES ('write_compression')
```

DESCRIBE TABLE

Beschreibt Tabelleninformationen.

Syntax

```
DESCRIBE [FORMATTED] [db_name.]table_name
```

Wenn die Option FORMATTED angegeben ist, zeigt die Ausgabe zusätzliche Informationen wie Tabellenspeicherort und Eigenschaften an.

Beispiel

```
DESCRIBE iceberg_table
```

DROP TABLE

Lässt einen Iceberg-Tabelle fallen.

> **Warning**
>
> Da Iceberg-Tabellen in Athena als verwaltete Tabellen gelten, werden durch das Löschen einer Iceberg-Tabelle auch alle Daten in der Tabelle entfernt.

Syntax

```
DROP TABLE [IF EXISTS] [db_name.]table_name
```

Beispiel

```
DROP TABLE iceberg_table
```

SHOW CREATE TABLE

Zeigt eine `CREATE TABLE`-DDL-Anweisung an, mit der die Iceberg-Tabelle in Athena neu erstellt werden kann. Wenn Athena die Tabellenstruktur nicht reproduzieren kann (z. B. weil benutzerdefinierte Tabelleneigenschaften in der Tabelle angegeben sind), wird ein UNSUPPORTED-Fehler ausgegeben.

Syntax

```
SHOW CREATE TABLE [db_name.]table_name
```

Beispiel

```
SHOW CREATE TABLE iceberg_table
```

ANZEIGEN DER TABELLENEINGENSCHAFTEN

Zeigt eine oder mehrere Tabelleneigenschaften einer Iceberg-Tabelle an. Es werden nur Athena-unterstützte Tabelleneigenschaften angezeigt.

Syntax

```
SHOW TBLPROPERTIES [db_name.]table_name [('property_name')]
```

Beispiel

```
SHOW TBLPROPERTIES iceberg_table
```

Iceberg-Tabellen-Metadaten abfragen

In einer SELECT-Abfrage können Sie die folgenden Eigenschaften hinter *table_name verwenden, um Iceberg-Tabellenmetadaten* abzufragen:

- $files – Zeigt die aktuellen Datendateien einer Tabelle an.
- $manifests – Zeigt die aktuellen Dateimanifeste einer Tabelle an.
- $history – Zeigt den Verlauf einer Tabelle an.
- $partitions – Zeigt die aktuellen Partitionen einer Tabelle an.
- $snapshots – Zeigt die Snapshots einer Tabelle an.
- $refs – Zeigt die Verweise einer Tabelle an.

Syntax

Die folgende Anweisung listet die Dateien für eine Iceberg-Tabelle auf.

```
SELECT * FROM "dbname"."tablename$files"
```

Die folgende Anweisung listet das Manifest für eine Iceberg-Tabelle auf.

```
SELECT * FROM "dbname"."tablename$manifests"
```

Die folgende Anweisung zeigt die Historie einer Iceberg-Tabelle.

```
SELECT * FROM "dbname"."tablename$history"
```

Im folgenden Beispiel werden die Partitionen für eine Iceberg-Tabelle gezeigt.

```
SELECT * FROM "dbname"."tablename$partitions"
```

Im folgenden Beispiel werden die Snapshots für eine Iceberg-Tabelle gelistet.

```
SELECT * FROM "dbname"."tablename$snapshots"
```

Im folgenden Beispiel werden die Referenzen für eine Iceberg-Tabelle gezeigt.

```
SELECT * FROM "dbname"."tablename$refs"
```

Iceberg-Tabellenschema weiterentwickeln

Iceberg-Schemaaktualisierungen sind reine Metadaten-Änderungen. Bei einer Schemaaktualisierung werden keine Datendateien geändert.

Das Iceberg-Format unterstützt die folgenden Änderungen an der Schemaentwicklung:

- Einfügen – Fügt einer Tabelle oder einer verschachtelten `struct` eine neue Spalte hinzu.
- Entfernen – Entfernt eine vorhandene Spalte aus einer Tabelle oder einem verschachtelten `struct`.
- Umbenennen – Benennt eine vorhandene Spalte oder ein vorhandenes Feld in einer verschachtelten `struct`.
- Neuanordnen – Ändert die Reihenfolge der Spalten.
- Promotionstyp – Erweitert den Typ einer Spalte, ein `struct`-Feld, eine `map`-Schlüssel, einen `map`-Wert, oder ein `list`-Element. Derzeit werden die folgenden Fälle für Iceberg-Tabellen unterstützt:
  - Ganzzahl bis große Ganzzahl
  - FLOAT, DOUBLE
  - Erhöhung der Genauigkeit eines Dezimaltyps

TABELLE ÄNDERN SPALTEN HINZUFÜGEN

Fügt einer vorhandenen Iceberg-Tabelle eine oder mehrere Spalten hinzu.

Syntax

```
ALTER TABLE [db_name.]table_name ADD COLUMNS (col_name data_type [,...])
```

Beispiele

Im folgenden Beispiel wird eine `comment`-Spalte des Typs `string` einer Iceberg-Tabelle hinzugefügt.

```
ALTER TABLE iceberg_table ADD COLUMNS (comment string)
```

Im folgenden Beispiel wird eine `point`-Spalte des Typs `struct` einer Iceberg-Tabelle hinzugefügt.

```
ALTER TABLE iceberg_table
ADD COLUMNS (point struct<x: double, y: double>)
```

Im folgenden Beispiel wird eine `points`-Spalte, die ein Array von Strukturen zu einer Iceberg-Tabelle ist, hinzugefügt.

```
ALTER TABLE iceberg_table
ADD COLUMNS (points array<struct<x: double, y: double>>)
```

ALTER TABLE DROP COLUMN

Lässt eine Spalte von einer vorhandenen Iceberg-Tabelle fallen.

Syntax

```
ALTER TABLE [db_name.]table_name DROP COLUMN col_name
```

Beispiel

```
ALTER TABLE iceberg_table DROP COLUMN userid
```

ALTER TABLE CHANGE COLUMN

Ändert den Namen, den Typ, die Reihenfolge oder den Kommentar einer Spalte.

> **Note**
>
> `ALTER TABLE REPLACE COLUMNS` wird nicht unterstützt. Da `REPLACE COLUMNS` alle Spalten entfernt und dann neue hinzufügt, wird es für Iceberg nicht unterstützt. `CHANGE COLUMN` ist die bevorzugte Syntax für die Schemaentwicklung.

Syntax

```
ALTER TABLE [db_name.]table_name
  CHANGE [COLUMN] col_old_name col_new_name column_type
  [COMMENT col_comment] [FIRST|AFTER column_name]
```

Beispiel

```
ALTER TABLE iceberg_table CHANGE comment blog_comment string AFTER id
```

SHOW_COLUMNS

Zeigt die Spalten in einer Tabelle an.

Syntax

```
SHOW COLUMNS (FROM|IN) [db_name.]table_name
```

Beispiel

```
SHOW COLUMNS FROM iceberg_table
```

Iceberg-Tabellen abfragen und Zeitreisen durchführen

Um einen Iceberg-Datensatz abzufragen, verwenden Sie eine `SELECT`-Standardanweisung wie die folgende. Abfragen folgen der Spezifikation des Apache-Iceberg-Formats v2 und führen sowohl merge-on-read Positions- als auch Gleichheitslöschungen durch.

```
SELECT * FROM [db_name.]table_name [WHERE predicate]
```

Um die Abfragezeiten zu optimieren, werden alle Prädikate dorthin verschoben, wo sich die Daten befinden.

Zeitreisen- und Versionsreiseabfragen

Jede Apache-Iceberg-Tabelle verwaltet ein versioniertes Manifest der darin enthaltenen Amazon-S3-Objekte. Frühere Versionen des Manifests können für Zeitreise- und Versionsreiseabfragen verwendet werden.

Zeitreiseabfragen in Athena fragen Amazon S3 nach historischen Daten aus einem konsistenten Snapshot ab einem bestimmten Datum und einer bestimmten Uhrzeit ab. Versionsreiseabfragen in Athena fragen Amazon S3 nach historischen Daten ab einer angegebenen Snapshot-ID ab.

Zeitreiseabfragen

Um eine Zeitreiseabfrage auszuführen, verwenden Sie `FOR TIMESTAMP AS OF` *`timestamp`* nach dem Tabellennamen in der `SELECT`-Anweisung, wie im folgenden Beispiel.

```
SELECT * FROM iceberg_table FOR TIMESTAMP AS OF timestamp
```

Die für Reisen festzulegende Systemzeit ist entweder ein Zeitstempel oder ein Zeitstempel mit einer Zeitzone. Wenn nicht angegeben, betrachtet Athena den Wert als Zeitstempel in UTC-Zeit.

In den folgenden Beispielabfragen für Zeitreisen werden CloudTrail Daten für das angegebene Datum und die angegebene Uhrzeit ausgewählt.

```
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)
```

Versionsreiseabfragen

Um eine Versionsreiseabfrage durchzuführen (d. h. einen konsistenten Snapshot ab einer angegebenen Version anzuzeigen), verwenden Sie `FOR VERSION AS OF` *`version`* nach dem Tabellennamen in der `SELECT`-Anweisung, wie im folgenden Beispiel.

```
SELECT * FROM [db_name.]table_name FOR VERSION AS OF version
```

Der *`Versions`*parameter ist die `bigint`-Snapshot-ID, die einer Iceberg-Tabellenversion zugeordnet ist.

Die folgende Beispielversionsreiseabfrage wählt Daten für die angegebene Version aus.

```
SELECT * FROM iceberg_table FOR VERSION AS OF 949530903748831860
```

> **Note**
>
> Die `FOR SYSTEM_TIME AS OF`- und `FOR SYSTEM_VERSION AS OF`-Klauseln in Athena-Engine-Version 2 wurden durch die `FOR TIMESTAMP AS OF`- und `FOR VERSION AS OF`-Klauseln in Athena-Engine-Version 3 ersetzt.

Abrufen der Snapshot-ID

Sie können die von Iceberg bereitgestellte Java-SnapshotUtilKlasse verwenden, um die Iceberg-Snapshot-ID abzurufen, wie im folgenden Beispiel.

```
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://my-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);
```

Zeit- und Versionsreisen kombinieren

Sie können Zeitreisen- und Versionsreisesyntax in derselben Abfrage verwenden, um verschiedene Timing- und Versionsbedingungen anzugeben, wie im folgenden Beispiel.

```
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)
```

Erstellen und Abfragen von Ansichten mit Iceberg-Tabellen

Verwenden Sie zum Erstellen und Abfragen von Athena-Ansichten für Iceberg-Tabellen `CREATE VIEW`-Ansichten wie in Arbeiten mit Ansichten beschrieben.

Beispiel:

```
CREATE VIEW view1 AS SELECT * FROM iceberg_table
```

```
SELECT * FROM view1
```

Wenn Sie daran interessiert sind, die Iceberg-Ansicht-Spezifikation zum Erstellen von Ansichten zu verwenden, wenden Sie sich an athena-feedback@amazon.com.

Arbeiten mit der differenzierte Zugriffskontrolle von Lake Formation

Athena-Engine-Version 3 unterstützt die differenzierte Zugriffskontrolle von Lake Formation mit Iceberg-Tabellen, einschließlich der Sicherheitskontrolle auf Spalten- und Zeilenebene. Diese Zugriffskontrolle funktioniert mit Zeitreiseabfragen und mit Tabellen, die eine Schemaentwicklung durchgeführt haben. Weitere Informationen finden Sie unter Differenzierte Zugriffskontrolle von Lake Formation und Athena-Arbeitsgruppen.

Wenn Sie Ihre Iceberg-Tabelle außerhalb von Athena erstellt haben, verwenden Sie Apache Iceberg SDK Version 0.13.0 oder höher, damit Ihre Iceberg-Tabellenspalteninformationen im AWS Glue Data Catalogausgefüllt werden. Wenn Ihre Iceberg-Tabelle keine Spalteninformationen in enthält AWS Glue, können Sie die Athena-FESTGELEGTE TABELLENEIGENSCHAFTEN ÄNDERNAnweisung oder das neueste Iceberg-SDK verwenden, um die Tabelle zu reparieren und die Spalteninformationen in zu aktualisieren AWS Glue.

Aktualisieren von Iceberg-Datentabellen

Iceberg-Tabellendaten können direkt auf Athena mit `INSERT`, `UPDATE`, und `DELETE`-Abfragen verwaltet werden. Jede Datenverwaltungstransaktion erzeugt einen neuen Snapshot, der über Zeitreisen abgefragt werden kann. Die `UPDATE`- und `DELETE`-Anweisungen folgen der Spezifikation

zum Löschen von Positionen auf Zeilenebene im Iceberg-Format v2 und erzwingen die Snapshot-Isolation.

Verwenden Sie die folgenden Befehle, um Datenverwaltungsvorgänge für Iceberg-Tabellen durchzuführen.

INSERT INTO

Fügt Daten in eine Iceberg-Tabelle ein. Athena Iceberg `INSERT INTO` wird nach der gescannten Datenmenge genauso berechnet wie aktuelle `INSERT INTO`-Abfragen für externe Hive-Tabellen. Um Daten in eine Iceberg-Tabelle einzufügen, verwenden Sie die folgende Syntax, wobei *`Abfragen`* `VALUES (val1, val2, ...)` oder `SELECT (col1, col2, …) FROM [`*`db_name`*`.]`*`table_name`* `WHERE` *`predicate`* sein kann. Informationen zur SQL-Syntax und semantischen Details finden Sie unter INSERT INTO.

```
INSERT INTO [db_name.]table_name [(col1, col2, …)] query
```

Die folgenden Beispiele fügen Werte in die Tabelle `iceberg_table` ein.

```
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` schreibt Iceberg-Positionslöschdateien in eine Tabelle. Dies wird als merge-on-read Löschen bezeichnet. Im Gegensatz zu einem copy-on-write Löschvorgang ist ein merge-on-read Löschvorgang effizienter, da Dateidaten nicht neu geschrieben werden. Wenn Athena Iceberg-Daten liest, führt sie die Iceberg-Position-Löschdateien mit Datendateien zusammen, um die neueste Ansicht einer Tabelle zu erzeugen. Um diese Positionslöschdateien zu entfernen, können Sie die Komprimierungs-Aktion REWRITE DATA ausführen. `DELETE`-Vorgänge werden nach der gescannten Datenmenge abgerechnet. Weitere Informationen zur Syntax finden Sie unter DELETE.

Im folgenden Beispiel werden Zeilen aus `iceberg_table` gelöscht, die `c3` als Wert für `category` haben.

```
DELETE FROM iceberg_table WHERE category='c3'
```

UPDATE

Athena Iceberg `UPDATE` schreibt Iceberg-Positions-Löschdateien und neu aktualisierte Zeilen als Datendateien in derselben Transaktion. `UPDATE` kann man sich als Kombination von `INSERT INTO` und `DELETE` vorstellen. `UPDATE`-Vorgänge werden nach der gescannten Datenmenge abgerechnet. Weitere Informationen zur Syntax finden Sie unter UPDATE.

Das folgende Beispiel aktualisiert die angegebenen Werte in der Tabelle `iceberg_table`.

```
UPDATE iceberg_table SET category='c2' WHERE category='c1'
```

MERGE INTO

Aktualisiert, löscht oder fügt bedingt Zeilen in eine Iceberg-Tabelle ein. Eine einzige Anweisung kann Aktionen zum Aktualisieren, Löschen und Einfügen kombinieren. Weitere Informationen zur Syntax finden Sie unter MERGE INTO.

> **Note**
>
> `MERGE INTO` ist transaktionsbasiert und wird nur für Apache-Iceberg-Tabellen in Athena-Engine-Version 3 unterstützt.

Das folgende Beispiel löscht alle Kunden aus der Tabelle `t`, die sich in der Quelltabelle `s` befinden.

```
MERGE INTO accounts t USING monthly_accounts_update s
ON t.customer = s.customer
WHEN MATCHED
THEN DELETE
```

Im folgenden Beispiel wird die Zieltabelle `t` mit Informationen aus Quelltabelle `s` aktualisiert. Für Kundenzeilen in Tabelle `t`, die über übereinstimmende Kundenzeilen in Tabelle `s` verfügen, erhöht das Beispiel die Einkäufe in Tabelle t. Wenn Tabelle `t` keine Übereinstimmung mit einer Kundenzeile in der Tabelle `s` hat, fügt das Beispiel die Kundenzeile aus Tabelle `s` in Tabelle `t` ein.

```
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)
```

Im folgenden Beispiel wird die Zieltabelle `t` bedingt mit Informationen aus der Quelltabelle `s` aktualisiert. Das Beispiel löscht alle übereinstimmenden Zielzeilen, deren Quelladresse Centreville ist. Für alle weiteren übereinstimmenden Zeilen fügt das Beispiel die Quellkäufe hinzu und legt die Zieladresse auf die Quelladresse fest. Wenn es in der Zieltabelle keine Übereinstimmung gibt, fügt das Beispiel die Zeile aus der Quelltabelle ein.

```
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)
```

Optimieren von Iceberg-Tabellen

Wenn sich Daten in einer Iceberg-Tabelle ansammeln, werden Abfragen aufgrund der erhöhten Bearbeitungszeit, die zum Öffnen von Dateien erforderlich ist, allmählich weniger effizient. Zusätzliche Rechenkosten fallen an, wenn die Tabelle Dateien löschen enthält. In Iceberg speichern Löschdateien Löschvorgänge auf Zeilenebene, und die Engine muss die gelöschten Zeilen auf Abfrageergebnisse anwenden.

Um die Leistung von Abfragen auf Iceberg-Tabellen zu optimieren, unterstützt Athena die manuelle Verdichtung als Tabellenwartungsbefehl. Verdichtungen optimieren das strukturelle Layout der Tabelle, ohne den Tabelleninhalt zu ändern.

OPTIMIZE

Die `OPTIMIZE` *`table`* `REWRITE DATA`-Verdichtungsaktion schreibt Datendateien basierend auf ihrer Größe und Anzahl der zugehörigen Löschdateien in ein optimierteres Layout um. Einzelheiten zur Syntax und Tabelleneigenschaften finden Sie unter OPTIMIZE.

Beispiel

Im folgenden Beispiel werden Löschdateien in Datendateien zusammengeführt und Dateien in der Nähe der Zieldateigröße erstellt, bei denen der Wert von `category` `c1` ist.

```
OPTIMIZE iceberg_table REWRITE DATA USING BIN_PACK
  WHERE category = 'c1'
```

VACUUM

`VACUUM` führt den Snapshot-Ablauf und das Entfernen verwaister Dateien durch. Diese Aktionen reduzieren die Größe der Metadaten und entfernen Dateien, die sich nicht im aktuellen Tabellenstatus befinden und zudem älter als die für die Tabelle angegebene Aufbewahrungsfrist sind. Einzelheiten zur Syntax finden Sie unter VACUUM.

Beispiel

Im folgenden Beispiel wird eine Tabelleneigenschaft verwendet, um die Tabelle `iceberg_table` so zu konfigurieren, dass die Daten der letzten drei Tage beibehalten werden. Anschließend verwendet es `VACUUM`, um die alten Snapshots ablaufen zu lassen und die verwaisten Dateien aus der Tabelle zu entfernen.

```
ALTER TABLE iceberg_table SET TBLPROPERTIES (
  'vacuum_max_snapshot_age_seconds'='259200'
)

VACUUM iceberg_table
```

Unterstützte Datentypen für Iceberg-Tabellen in Athena

Athena kann Iceberg-Tabellen abfragen, die die folgenden Datentypen enthalten:

```
binary
boolean
date
decimal
double
float
int
list
long
map
string
struct
timestamp without time zone
```

Weitere Hinweise zu Iceberg-Tabellentypen finden Sie auf der Schemas-Seite für Iceberg in der Apache-Dokumentation.

Die folgende Tabelle zeigt das Verhältnis zwischen Athena-Datentypen und Datentypen von Iceberg-Tabellen.

Iceberg-Typ	Athena-Typ	Hinweise
`boolean`	`boolean`	
-	`tinyint`	Wird für Iceberg-Tabellen in Athena nicht unterstützt.
-	`smallint`	Wird für Iceberg-Tabellen in Athena nicht unterstützt.
`int`	`int`	In Athena-DML-Anweisungen ist dieser Typ `INTEGER`.
`long`	`bigint`	
`double`	`double`	
`float`	`float`	
`decimal(P, S)`	`decimal(P, S)`	`P` ist Präzision, `S` ist Skalieren.
-	`char`	Wird für Iceberg-Tabellen in Athena nicht unterstützt.
`string`	`string`	In Athena-DML-Anweisungen ist dieser Typ `VARCHAR`.
`binary`	`binary`	
`date`	`date`	
`time`	-	Für Athena-Iceberg-DDL-Anweisungen wie `CREATE TABLE` wird nur Iceberg-Zeitstempel (ohne Zeitzone) unterstützt, aber alle Zeitstempeltypen können über Athena abgefragt werden.
`timestamp`	`timestamp`	
`timestamptz`	`timestamptz`	

Iceberg-Typ	Athena-Typ	Hinweise
`list<E>`	`array`	
`map<K,V>`	`map`	
`struct<...>`	`struct`	
`fixed(L)`	-	Der `fixed(L)`-Typ wird derzeit in Athena nicht unterstützt.

Weitere Informationen zu Datentypen in Athena finden Sie unter Datentypen in Amazon Athena.

Andere Athena-Operationen an Iceberg-Tabellen

Operationen auf Datenbankebene

Wenn Sie DROP DATABASE mit der Option `CASCADE` verwenden, werden auch alle Iceberg-Tabellendaten entfernt. Die folgenden DDL-Operationen haben keine Auswirkungen auf Iceberg-Tabellen.

- CREATE DATABASE
- ALTER DATABASE SET DBPROPERTIES
- SHOW DATABASES
- SHOW TABLES
- SHOW VIEWS

Partition bezogene Operationen

Da Iceberg-Tabellen versteckte Partitionierung verwenden, müssen Sie nicht direkt mit physischen Partitionen arbeiten. Infolgedessen unterstützen Iceberg-Tabellen in Athena die folgenden partitionsbezogenen DDL-Operationen nicht:

- SHOW PARTITIONS
- ALTER TABLE ADD PARTITION
- ALTER TABLE DROP PARTITION

- ALTER TABLE RENAME PARTITION

Wenn Sie die Evolution der Iceberg-Partition in Athena sehen möchten, senden Sie Feedback an athena-feedback@amazon.com.

Iceberg-Tabellen entladen

Iceberg-Tabellen können in Dateien in einem Ordner auf Amazon S3 entladen werden. Weitere Informationen finden Sie unter UNLOAD.

MSCK REPAIR

Da Iceberg-Tabellen Informationen zum Tabellenlayout verfolgen, ist das Ausführen von MSCK REPAIR TABLE wie bei Hive-Tabellen nicht erforderlich und wird nicht unterstützt.

Weitere Ressourcen

Ausführliche Artikel zur Verwendung von Athena mit Apache-Iceberg-Tabellen finden Sie in den folgenden Beiträgen im AWS -Big-Data-Blog.

- Beschleunigen Sie das Datenwissenschaft-Feature-Engineering auf transaktionalen Data Lakes mithilfe von Amazon Athena mit Apache Iceberg
- Erstellen Sie einen Apache-Iceberg-Data Lake mit Amazon Athena , Amazon EMR und AWS Glue
- Führen Sie Upserts in einem Data Lake mit Amazon Athena und Apache Iceberg durch
- Erstellen Sie einen transaktionalen Data Lake mit Apache Iceberg und AWS Gluekontoübergreifende Datenfreigaben mit AWS Lake Formation und Amazon Athena
- Verwenden Sie Apache Iceberg in einem Data Lake, um die inkrementelle Datenverarbeitung zu unterstützen
- Erstellen Sie einen Data Lake für Apache Iceberg in Echtzeit, der an die DSGVO angepasst ist
- Automatisieren Sie die Replikation relationaler Quellen in einen transaktionalen Data Lake mit Apache Iceberg und AWS Glue
- Interagieren Sie mit Apache-Iceberg-Tabellen mithilfe von Amazon Athena und kontoübergreifenden, detaillierten Berechtigungen mithilfe von AWS Lake Formation
- Erstellen Sie mit Apache Iceberg, Amazon EMR Serverless und Amazon Athena Serverless einen Serverless-Transaktions-Data-Lake

Sicherheit von Amazon Athena

Die Sicherheit in der Cloud hat bei AWS höchste Priorität. Als AWS-Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die zur Erfüllung der Anforderungen von Organisationen entwickelt wurden, für die Sicherheit eine kritische Bedeutung hat.

Sicherheit gilt zwischen AWS und Ihnen eine geteilte Verantwortung. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud selbst – AWS ist dafür verantwortlich, die Infrastruktur zu schützen, mit der AWS-Services in der AWS Cloud ausgeführt werden. AWS stellt Ihnen außerdem Services bereit, die Sie sicher nutzen können. Die Wirksamkeit unserer Sicherheitsfunktionen wird regelmäßig von externen Prüfern im Rahmen des AWS-Compliance-Programms getestet und überprüft. Weitere Informationen zu den Compliance-Programmen für Athena finden Sie unter Durch das Compliance-Programm abgedeckte AWS-Services.
- Sicherheit in der Cloud – Ihr Verantwortungsumfang wird durch den AWS-Service bestimmt, den Sie verwenden. In Ihre Verantwortung fallen außerdem weitere Faktoren, wie z. B. die Vertraulichkeit der Daten, die Anforderungen Ihrer Organisation sowie geltende Gesetze und Vorschriften.

Diese Dokumentation beschreibt, wie Sie das Modell der übergreifenden Verantwortlichkeit bei der Verwendung von Amazon Athena anwenden können. Die folgenden Themen veranschaulichen, wie Sie Athena zur Erfüllung Ihrer Sicherheits- und Compliance-Ziele konfigurieren können. Sie erfahren außerdem, wie Sie andere AWS-Services verwenden können, die Ihnen beim Überwachen und Sichern Ihrer Athena-Ressourcen helfen.

Themen

Datenschutz in Athena

Das AWS Modell der geteilten VerantwortungModell gilt für den Datenschutz in Amazon Athena . Wie in diesem Modell beschrieben, AWS ist für den Schutz der globalen Infrastruktur verantwortlich, die alle ausführt AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS -Modell der geteilten Verantwortung und in der DSGVO im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir Ihnen, -Anmeldeinformationen zu schützen AWS-Konto und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einzurichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor Authentifizierung (MFA).
- Verwenden Sie SSL/TLS für die Kommunikation mit - AWS Ressourcen. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API- und Benutzeraktivitätsprotokollierung mit ein AWS CloudTrail.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen Standardsicherheitskontrollen in AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff auf AWS über eine Befehlszeilenschnittstelle oder eine API FIPS-140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-2.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit Athena oder anderen AWS-Services über die Konsole, API AWS CLIoder AWS SDKs arbeiten. Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Als zusätzliche Sicherheitsmaßnahme können Sie den globalen Bedingungskontextschlüssel aws:CalledVia verwenden, um Anfragen auf die von Athena gestellten zu beschränken. Weitere Informationen finden Sie unter Athena mit CalledVia-Kontextschlüsseln verwenden.

Schutz mehrerer Datentypen

Es sind mehrere Datentypen involviert, wenn Sie Datenbanken und Tabellen mit Athena erstellen. Zu diesen Datentypen gehören Quelldaten, die in Amazon S3 gespeichert sind, Metadaten für Datenbanken und Tabellen, die Sie erstellen, wenn Sie Abfragen ausführen, oder der AWS Glue Crawler, um Daten, Abfrageergebnisdaten und den Abfrageverlauf zu ermitteln. In diesem Abschnitt werden alle diese Arten von Daten besprochen und ihre Verarbeitung erläutert.

- Quelldaten – Sie speichern die Daten für Datenbanken und Tabellen in Amazon S3 und Athena verändert diese nicht. Weitere Informationen finden Sie unter Datenschutz in Amazon S3 im Benutzerhandbuch für Amazon Simple Storage Service. Sie steuern den Zugriff auf Ihre Quelldaten und ihre Verschlüsselung in Amazon S3. Sie können Athena zum Erstellen von Tabellen basierend auf verschlüsselten Datensätzen in Amazon S3 verwenden.
- Datenbank- und Tabellenmetadaten (Schema) – Athena verwendet schema-on-read Technologie, was bedeutet, dass Ihre Tabellendefinitionen auf Ihre Daten in Amazon S3 angewendet werden, wenn Athena Abfragen ausführt. Alle von Ihnen definierten Schemata werden automatisch gespeichert, es sei denn, Sie löschen sie ausdrücklich. In Athena können Sie die Datenkatalog-Metadaten mit DDL-Anweisungen ändern. Sie können auch Tabellendefinitionen und Schemata löschen, ohne dass dies Auswirkungen auf die zugrunde liegenden Daten hat, die in Amazon S3 gespeichert sind. Die Metadaten für Datenbanken und Tabellen, die Sie in Athena verwenden, werden im AWS Glue Data Cataloggespeichert.

  Mit AWS Identity and Access Management (IAM) können Sie differenzierte Zugriffsrichtlinien für Datenbanken und Tabellen definieren, die in der registriert AWS Glue Data Catalog sind. Sie können auch Metadaten im AWS Glue Data Catalogverschlüsseln. Wenn Sie die Metadaten verschlüsseln, verwenden Sie Berechtigungen für den Zugriff auf verschlüsselte Metadaten.
- Abfrageergebnisse und Abfrageverläufe, einschließlich gespeicherter Abfragen – Abfrageergebnisse werden an einem Speicherort in Amazon S3 gespeichert, den Sie global oder für jede Arbeitsgruppe wählen können. Wenn Sie dies nicht angeben, verwendet Athena in jedem Fall den Standard-Speicherort. Sie kontrollieren den Zugriff auf Amazon-S3-Buckets zum Speichern von Abfrageergebnisse und gespeicherten Abfragen. Darüber hinaus können Sie die Abfrageergebnisse verschlüsseln, die Sie in Amazon S3 speichern. Benutzer müssen über die entsprechenden Berechtigungen zum Zugriff auf Amazon-S3-Speicherorte und zum Entschlüsseln

von Dateien verfügen. Weitere Informationen finden Sie unter Verschlüsseln der in Amazon S3 gespeicherten Athena-Abfrageergebnisse in diesem Dokument.

Athena hält den Abfrageverlauf 45 Tage vor. Sie können den Abfrageverlauf mit Athena-APIs , in der Konsole und mit anzeigen AWS CLI. APIs Um die Abfragen für länger als 45 Tage lang vorzuhalten, müssen Sie sie speichern. Verwenden Sie zum Schutz des Zugriffs auf gespeicherte Abfragen use workgroups (Arbeitsgruppen verwenden) in Athena, die den Zugriff auf gespeicherte Abfragen nur auf dazu berechtigte Benutzer einschränken.

Themen

Verschlüsselung im Ruhezustand

Sie können in Amazon Athena Abfragen zu verschlüsselten Daten in Amazon S3 in derselben Region und in einer begrenzten Anzahl von Regionen ausführen. Sie können die Abfrageergebnisse auch in Amazon S3 und die Daten im AWS Glue Data Catalog verschlüsseln.

Sie können die folgenden Ressourcen in Athena verschlüsseln:

- Die Ergebnisse aller Abfragen in Amazon S3, die von Athena in einem als Amazon-S3-Ergebnisort bezeichneten Ort gespeichert werden. Die in Amazon S3 gespeicherten Abfrageergebnisse lassen sich verschlüsseln, und zwar unabhängig davon, ob der zugrunde liegende Datensatz in Amazon S3 verschlüsselt oder unverschlüsselt ist. Weitere Informationen finden Sie unter Verschlüsseln der in Amazon S3 gespeicherten Athena-Abfrageergebnisse.
- Die Daten im AWS Glue Data Catalog. Weitere Informationen finden Sie unter Berechtigungen für verschlüsselte Metadaten im AWS Glue -Datenkatalog.

> **Note**
>
> Sie müssen die Abfrage von verschlüsselten Datensätzen in Amazon S3 sowie die Optionen zum Verschlüsseln der Abfrageergebnisse in Athena unterschiedlich einrichten. Jede Option wird separat konfiguriert und aktiviert. Sie können jeweils verschiedene

Verschlüsselungsmethoden oder Schlüssel verwenden. Das heißt, das Lesen von verschlüsselten Daten in Amazon S3 führt nicht automatisch zur Verschlüsselung der Athena-Abfrageergebnisse in Amazon S3. Dies gilt auch umgekehrt. Mit der Verschlüsselung der Athena-Abfrageergebnisse in Amazon S3 wird nicht automatisch auch der zugrunde liegende Datensatz in Amazon S3 verschlüsselt.

Themen

Unterstützte Verschlüsselungsoptionen der Amazon S3

Athena unterstützt die folgenden Verschlüsselungsoptionen für Datasets und Abfrageergebnisse in Amazon S3.

Verschlüsselungstyp	Beschreibung	Regionsübergreifende Unterstützung
SSE-S3 (SSE-S3)	Serverseitige Verschlüsselung (SSE) mit einem von Amazon S3 verwalteten Schlüssel.	Ja
SSE-KMS	Serverseitige Verschlüsselung (SSE) mit einem vom AWS Key Management Service Kunden verwalteten Schlüssel. **Note** Bei diesem Verschlüsselungstyp müssen Sie beim Erstellen einer Tabelle in Athena nicht angeben, dass Daten verschlüsselt sind.	Ja
CSE-KMS	Clientseitige Verschlüsselung (CSE) mit einem vom AWS KMS Kunden verwalteten Schlüssel. In Athena erfordert	Nein

Verschlüsselungstyp	Beschreibung	Regionsübergreifende Unterstützung
	diese Option, dass Sie eine `CREATE TABLE`-Anweisung mit einer `TBLPROPERTIES` -Klausel verwenden, die `'has_encrypted_data'='true'` angibt. Weitere Informationen finden Sie unter Erstellen von Tabellen basierend auf verschlüsselten Datensätzen in Amazon S3.	

Weitere Informationen zur AWS KMS Verschlüsselung mit Amazon S3 finden Sie unter Was ist AWS Key Management Service und wie Amazon Simple Storage Service (Amazon S3) nutzt AWS KMS im AWS Key Management Service -Entwicklerhandbuch. Weitere Informationen zur Verwendung von SSE-KMS oder CSE-KMS mit Athena finden Sie unter Launch: Amazon Athena fügt Unterstützung für das Abfragen verschlüsselter Daten aus dem AWS -Big-Data-Blog hinzu.

Nicht unterstützte Optionen

Die folgenden Verschlüsselungsoptionen werden nicht unterstützt:

- SSE mit vom Kunden bereitgestellten Schlüsseln (SSE-C)
- Clientseitige Verschlüsselung mit einem clientseitigen verwalteten Schlüssel.
- Asymmetrische Schlüssel.

Einen Vergleich der Verschlüsselungsoptionen von Amazon S3 finden Sie unter Daten durch Verschlüsselung schützen im Benutzerhandbuch für Amazon Simple Storage Service.

Tools für die clientseitige Verschlüsselung

Beachten Sie für die clientseitige Verschlüsselung, dass zwei Tools verfügbar sind:

- Amazon-S3-Verschlüsselungs-Client – Dies verschlüsselt Daten nur für Amazon S3 und wird von Athena unterstützt.
- AWS Encryption SDK – Das SDK kann verwendet werden, um Daten überall in zu verschlüsseln, wird AWS aber nicht direkt von Athena unterstützt.

Diese Tools sind nicht kompatibel und Daten, die mit einem Tool verschlüsselt wurden, können nicht vom anderen entschlüsselt werden. Athena unterstützt den Amazon-S3-Verschlüsselungs-Client nur

direkt. Wenn Sie das SDK verwenden, um Ihre Daten zu verschlüsseln, können Sie Abfragen von Athena ausführen, aber die Daten werden als verschlüsselter Text zurückgegeben.

Wenn Sie Athena verwenden möchten, um Daten abzufragen, die mit dem AWS -Verschlüsselungs-SDK verschlüsselt wurden, müssen Sie Ihre Daten herunterladen und entschlüsseln und sie dann erneut mit dem Amazon-S3-Verschlüsselungs-Client verschlüsseln.

Berechtigungen für verschlüsselte Daten in Amazon S3

Abhängig von der Art der in Amazon S3 verwendeten Verschlüsselung müssen Sie möglicherweise Berechtigungen, auch „Allow“-Aktionen (Zulassen) genannt, zu Ihren in Athena verwendeten Richtlinien hinzufügen:

- SSE-S3 – Wenn Sie für die Verschlüsselung SSE-S3 verwenden, benötigen Athena-Benutzer keine zusätzlichen Berechtigungen in ihren Richtlinien. Es reicht, wenn die entsprechenden Amazon-S3-Berechtigungen für den jeweiligen Amazon-S3-Speicherort und für Athena-Aktionen vorhanden sind. Weitere Informationen zu Richtlinien, die entsprechende Athena- und Amazon-S3-Berechtigungen erlauben, finden Sie unter AWS verwaltete Richtlinien für Amazon Athena und Zugriff auf Amazon S3.
- AWS KMS – Wenn Sie AWS KMS für die Verschlüsselung verwenden, müssen Athena-Benutzer zusätzlich zu den Athena- und Amazon S3-Berechtigungen bestimmte AWS KMS Aktionen ausführen dürfen. Sie lassen diese Aktionen zu, indem Sie die Schlüsselrichtlinie für die vom AWS KMS Kunden verwalteten CMKs bearbeiten, die zum Verschlüsseln von Daten in Amazon S3 verwendet werden. Um Schlüsselbenutzer zu den entsprechenden AWS KMS Schlüsselrichtlinien hinzuzufügen, können Sie die - AWS KMS Konsole unter https://console.aws.amazon.com/kms verwenden. Informationen zum Hinzufügen eines Benutzers zu einer AWS KMS Schlüsselrichtlinie finden Sie unter Erlaubt Schlüsselbenutzern die Verwendung des CMK im AWS Key Management Service Entwicklerhandbuch für .

> **Note**
>
> Erfahrene Schlüsselrichtlinien-Administratoren können die Schlüsselrichtlinien anpassen. `kms:Decrypt` ist die minimal erlaubte Aktion für einen Athena-Benutzer, um mit einem verschlüsselten Datensatz arbeiten zu können. Zur Nutzung von verschlüsselten Abfrageergebnissen sind Berechtigungen für die Aktionen `kms:GenerateDataKey` und `kms:Decrypt` notwendig.

Wenn Sie Athena verwenden, um Datensätze in Amazon S3 mit einer großen Anzahl von Objekten abzufragen, die mit verschlüsselt sind AWS KMS, AWS KMS können Drosselungsabfrageergebnisse auftreten. Bei einer hohen Zahl kleiner Objekte ist dies noch wahrscheinlicher. Wiederholte Anforderungen werden von Athena unterbunden, dennoch kann ein Drosselungsfehler auftreten. Wenn Sie mit einer großen Anzahl verschlüsselter Objekte arbeiten und dieses Problem auftritt, besteht eine Möglichkeit darin, Amazon-S3-Bucket-Schlüssel zu aktivieren, um die Anzahl der Aufrufe an KMS zu reduzieren. Weitere Informationen finden Sie unter Reduzieren der Kosten für SSE-KMS mit Amazon-S3-Bucket-Schlüsseln im Benutzerhandbuch von Amazon Simple Storage Service. Eine weitere Möglichkeit besteht darin, Ihre Servicekontingente für AWS KMSzu erhöhen. Weitere Informationen finden Sie unter Kontingente im AWS Key Management Service -Entwicklerhandbuch.

Informationen zur Fehlerbehebung bei Berechtigungen bei der Verwendung von Amazon S3 mit Athena finden Sie im Abschnitt Berechtigungen des Themas Athena-Fehlerbehebung.

Berechtigungen für verschlüsselte Metadaten im AWS Glue -Datenkatalog

Wenn Sie Metadaten in der verschlüsseln AWS Glue Data Catalog, müssen Sie den Richtlinien`"kms:GenerateDataKey"`, die Sie für den Zugriff auf Athena verwenden, die `"kms:Encrypt"` Aktionen `"kms:Decrypt"`, und hinzufügen. Weitere Informationen finden Sie unter Zugriff auf verschlüsselte Metadaten von Athena im AWS Glue Data Catalog.

Verschlüsseln der in Amazon S3 gespeicherten Athena-Abfrageergebnisse

Sie richten die Verschlüsselung der Abfrageergebnisse mithilfe der Athena-Konsole oder mithilfe von JDBC oder ODBC ein. Arbeitsgruppen ermöglichen Ihnen die Durchsetzung der Verschlüsselung der Abfrageergebnisse.

In der Konsole können Sie die Einstellung für die Verschlüsselung von Abfrageergebnissen auf zwei Arten konfigurieren:

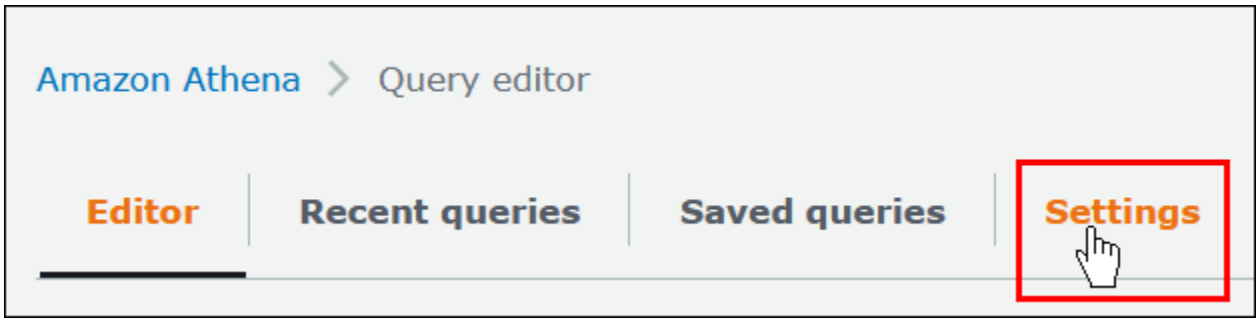
- Clientseitige Einstellungen – Wenn Sie Settings (Einstellungen) in der Konsole oder den API-Vorgängen verwenden, um anzugeben, dass Sie Abfrageergebnisse verschlüsseln möchten, wird dies als Verwendung clientseitiger Einstellungen bezeichnet. Zu den clientseitigen Einstellungen gehören der Speicherort der Abfrageergebnisse und die Verschlüsselung. Wenn Sie diese angeben, werden sie verwendet, solange sie nicht von den Arbeitsgruppeneinstellungen überschrieben werden.

- Arbeitsgruppeneinstellungen – Wenn Sie eine Arbeitsgruppe erstellen oder bearbeiten und das Feld Clientseitige Einstellungen überschreiben auswählen, verwenden alle Abfragen, die in dieser Arbeitsgruppe ausgeführt werden, die Einstellungen zur Arbeitsgruppenverschlüsselung und zum Speicherort der Abfrageergebnisse. Weitere Informationen finden Sie unter Arbeitsgruppen-Einstellungen überschreiben clientseitige Einstellungen.

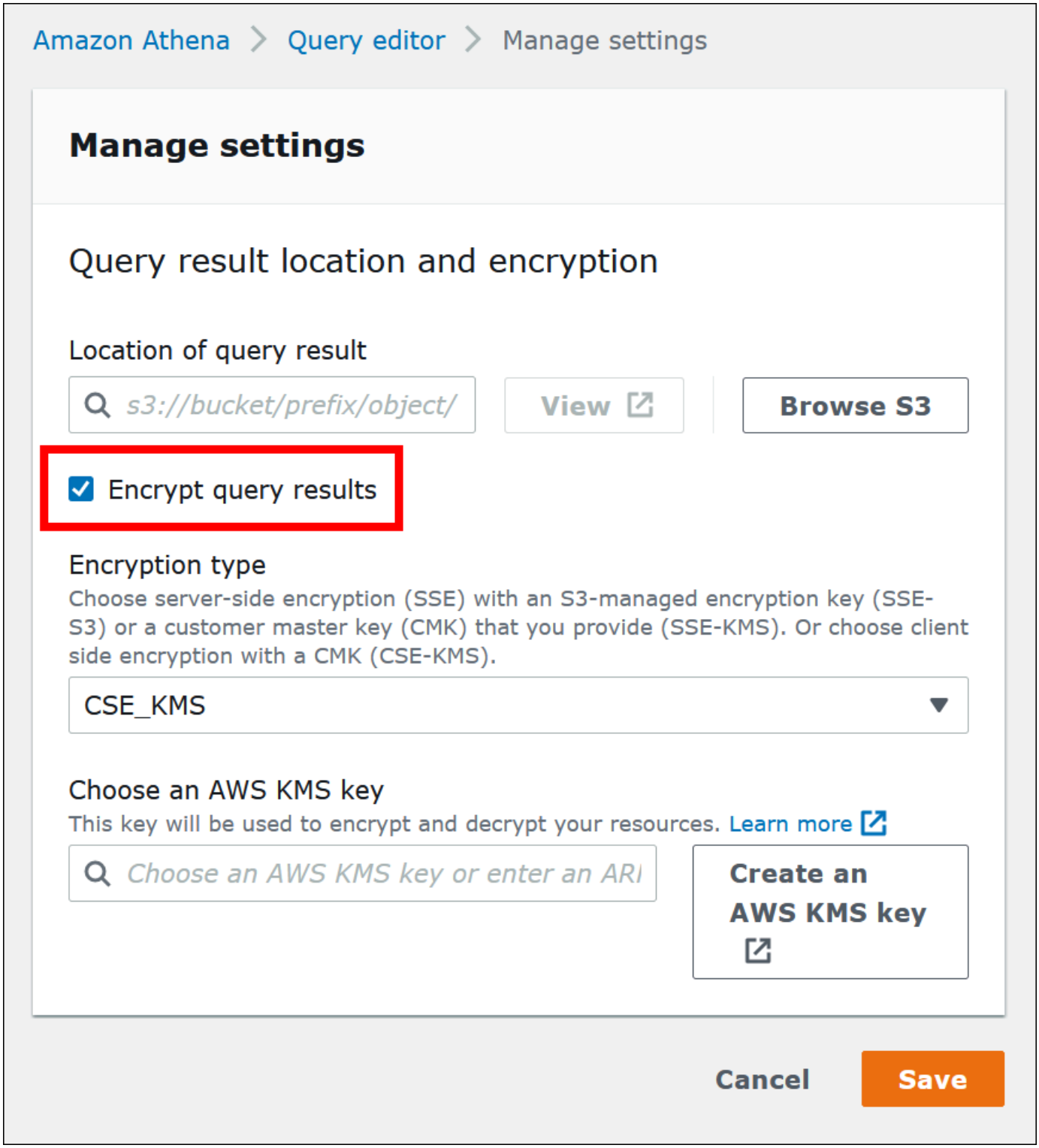
So verschlüsseln Sie die in Amazon S3 gespeicherten Abfrageergebnisse mit der Konsole

> **Important**
>
> Wenn für Ihre Arbeitsgruppe das Feld Override client-side settings (Clientseitige Einstellungen überschreiben) ausgewählt ist, verwenden alle Abfragen in der Arbeitsgruppe die Arbeitsgruppeneinstellungen. Die Verschlüsselungskonfiguration und der Speicherort der Abfrageergebnisse, die auf der Registerkarte Einstellungen in der Athena-Konsole, nach API-Vorgängen und nach JDBC- und ODBC-Treibern angegeben sind, werden nicht verwendet. Weitere Informationen finden Sie unter Arbeitsgruppen-Einstellungen überschreiben clientseitige Einstellungen.

1. Wählen Sie in der Athena-Konsole Settings (Einstellungen).



2. Wählen Sie Manage (Verwalten).
3. Geben Sie für Speicherort des Abfrageergebnisses einen Amazon-S3-Pfad ein oder wählen Sie diesen. Dies ist der Amazon-S3-Speicherort, an dem Abfrageergebnisse gespeichert werden.
4. Wählen Sie Encrypt query results aus.



5. Wählen Sie im Feld Encryption type den Eintrag CSE-KMS, SSE-KMS oder SSE-S3 aus. Von diesen drei bietet CSE-KMS die höchste Verschlüsselungsstufe und SSE-S3 die niedrigste.
6. Wenn Sie SSE-KMS oder CSE-KMS ausgewählt haben, geben Sie einen - AWS KMS Schlüssel an.

- Wählen Sie unter AWS KMS Schlüssel auswählen den Alias aus oder geben Sie einen - AWS KMS Schlüssel-ARN ein, wenn Ihr Konto Zugriff auf einen vorhandenen AWS KMS kundenverwalteten Schlüssel (CMK) hat.
- Wenn Ihr Konto keinen Zugriff auf einen vorhandenen kundenverwalteten Schlüssel (CMK) hat, wählen Sie Erstellen eines - AWS KMS Schlüssels und öffnen Sie dann die AWS KMS -Konsole. Weitere Informationen finden Sie unter Erstellen von Schlüsseln im AWS Key Management Service -Entwicklerhandbuch.

> **Note**
>
> Athena unterstützt nur symmetrische Schlüssel zum Lesen und Schreiben von Daten.

7. Kehren Sie zur Athena-Konsole zurück und wählen Sie den Schlüssel aus, den Sie mit Alias oder ARN erstellt haben.
8. Wählen Sie Save (Speichern) aus.

Verschlüsseln von Athena-Abfrageergebnissen bei Verwendung von JDBC oder ODBC

Wenn Sie Verbindungen mithilfe der JDBC- oder ODBC-Treiber herstellen, konfigurieren Sie Treiberoptionen zur Angabe der zu verwendenden Verschlüsselungsart und des Speicherorts des Amazon-S3-Staging-Verzeichnisses. Informationen zum Konfigurieren des JDBC- oder ODBC-Treibers zum Verschlüsseln der Abfrageergebnisse mit einem der von Athena unterstützten Verschlüsselungsprotokolle finden Sie unter Herstellen einer Verbindung mit Amazon Athena mit ODBC- und JDBC-Treibern.

Erstellen von Tabellen basierend auf verschlüsselten Datensätzen in Amazon S3

Wenn Sie eine Tabelle erstellen, zeigen Sie Athena an, dass ein Datensatz in Amazon S3 verschlüsselt ist. Bei Verwendung von SSE-KMS ist dies nicht erforderlich. Sowohl für SSE-S3 als auch für die AWS KMS Verschlüsselung bestimmt Athena, wie der Datensatz entschlüsselt und die Tabelle erstellt wird, sodass Sie keine Schlüsselinformationen angeben müssen.

Benutzer, die Abfragen ausführen möchten. einschließlich des Benutzers, der die Tabelle erstellt, müssen die zuvor in diesem Thema beschriebenen Berechtigungen haben.

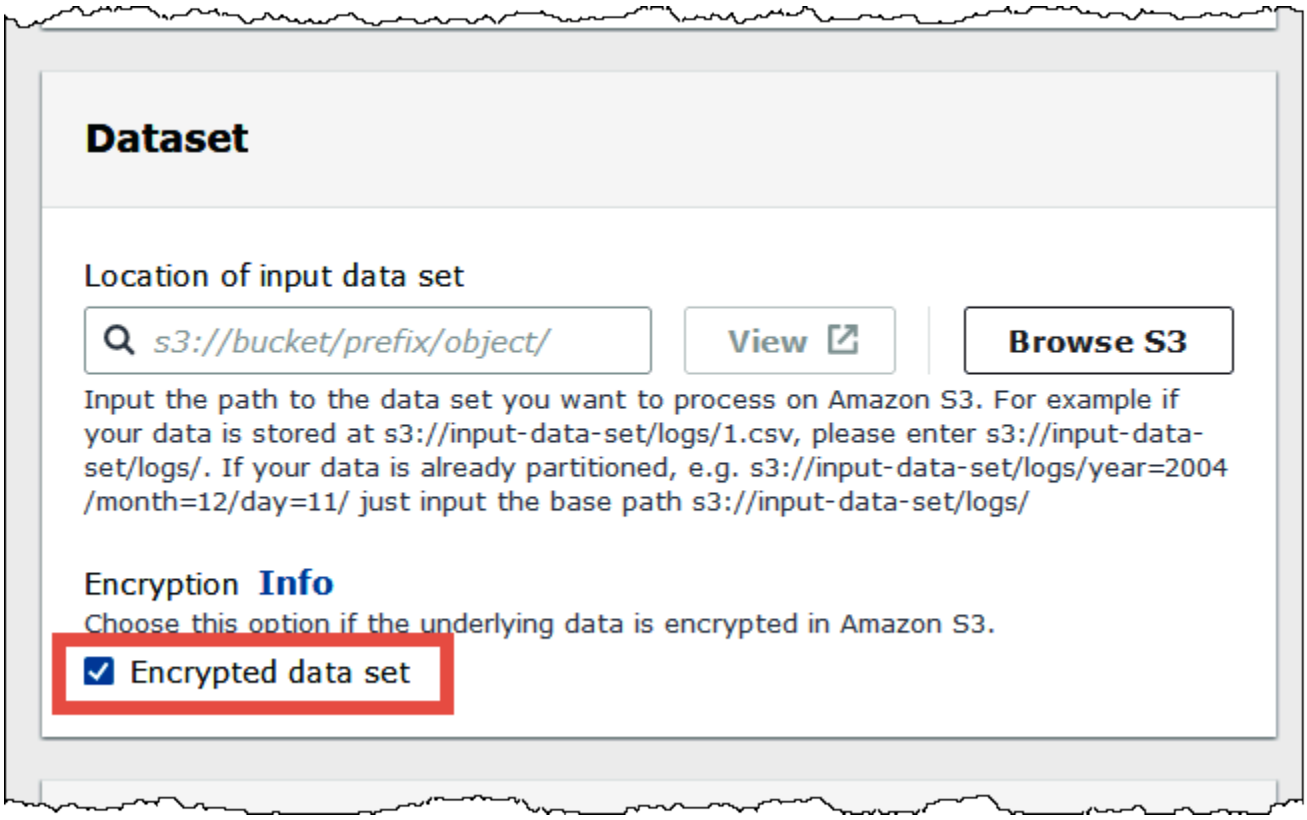
> **Important**
>
> Wenn Sie Amazon EMR zusammen mit EMRFS zum Hochladen verschlüsselter Parquet-Dateien verwenden, müssen Sie mehrteilige Uploads deaktivieren, indem Sie für `false` `fs.s3n.multipart.uploads.enabled` festlegen. Wenn Sie dies nicht tun, kann Athena die Länge der Parquet-Datei nicht ermitteln und der Fehler HIVE_CANNOT_OPEN_SPLIT tritt auf. Weitere Informationen finden Sie unter Konfigurieren von mehrteiligen Uploads für Amazon S3 im Verwaltungshandbuch für Amazon EMR.

Führen Sie einen der folgenden Schritte aus, um anzugeben, dass ein Datensatz in Amazon S3 verschlüsselt ist. Bei Verwendung von SSE-KMS ist dieser Schritt nicht erforderlich.

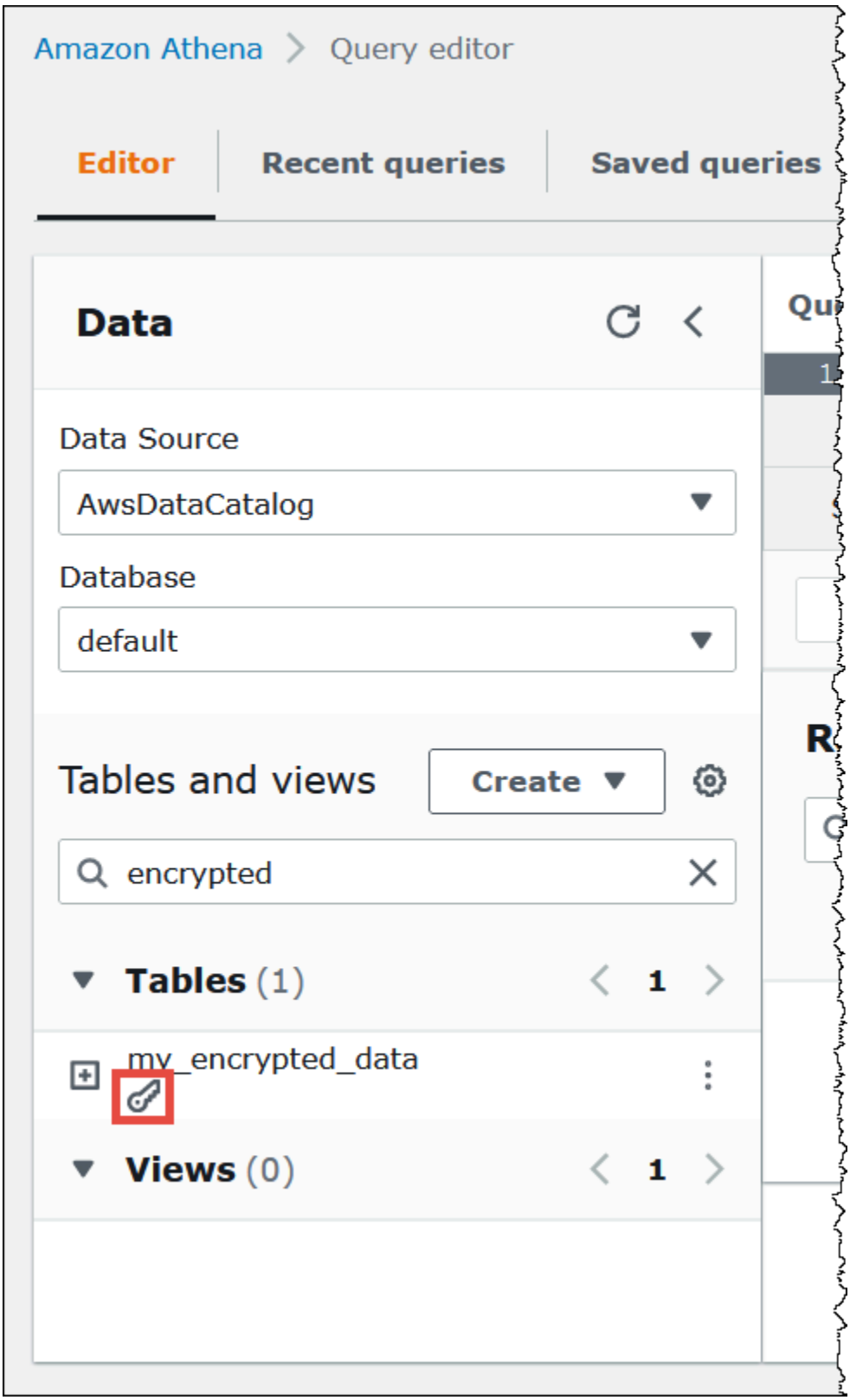
- Verwenden Sie in einer CREATE-TABLE-Anweisung eine `TBLPROPERTIES`-Klausel, die `'has_encrypted_data'='true'` angibt, wie im folgenden Beispiel.

```
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://bucket/folder_with_my_encrypted_data/'
TBLPROPERTIES (
   'has_encrypted_data'='true')
```

- Verwenden Sie den JDBC-Treiber und legen Sie den `TBLPROPERTIES`-Wert wie im vorherigen Beispiel gezeigt fest, wenn Sie `statement.executeQuery()` verwenden, um die Anweisung CREATE TABLE auszuführen.
- Wenn Sie die Athena-Konsole zum Erstellen einer Tabelle mit einem Formular verwenden und einen Tabellenspeicherort angeben, wählen Sie die Option Verschlüsselter Datensatz aus.



In der Tabellenliste der Athena-Konsole zeigen verschlüsselte Tabellen ein schlüsselförmiges Symbol an.


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

Verschlüsselung während der Übertragung

Zusätzlich zum Verschlüsseln von Data-at-Rest in Amazon S3 verwendet Amazon Athena die Transport-Layer-Security (TLS)-Verschlüsselung für Daten während der Übertragung zwischen Athena und Amazon S3 und zwischen Athena und Kundenanwendungen, die darauf zugreifen.

Erlauben Sie nur verschlüsselte Verbindungen über HTTPS (TLS) unter Verwendung der `aws:SecureTransport condition` auf Amazon-S3-Bucket-IAM-Richtlinien.

Abfrageergebnisse, die zu JDBC- oder ODBC-Clients gestreamt werden, werden mit TLS verschlüsselt. Weitere Informationen zu den neuesten Versionen der JDBC- und ODBC-Treiber und deren Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC und Herstellen einer Verbindung mit Amazon Athena mit ODBC.

Für Athena-Verbunddatenquellenkonnektoren hängt die Unterstützung der Verschlüsselung bei der Übertragung mit TLS vom jeweiligen Konnektor ab. Informationen finden Sie in der Dokumentation zu den einzelnen Datenquellen-Konnektoren.

Schlüsselverwaltung

Amazon Athena unterstützt AWS Key Management Service (AWS KMS), um Datensätze in Amazon S3 und Athena-Abfrageergebnissen zu verschlüsseln. AWS KMS verwendet vom Kunden verwaltete Schlüssel (CMKs), um Ihre Amazon S3-Objekte zu verschlüsseln, und stützt sich auf Envelope-Verschlüsselung.

In können AWS KMSSie die folgenden Aktionen ausführen:

- Erstellen von Schlüsseln
- Import Ihres eigenen Schlüsselmaterials für neue CMKs

> **Note**
>
> Athena unterstützt nur symmetrische Schlüssel zum Lesen und Schreiben von Daten.

Weitere Informationen finden Sie unter Was ist AWS Key Management Service im AWS Key Management Service -Entwicklerhandbuch und Wie Amazon Simple Storage Service AWS KMSverwendet. Um die Schlüssel in Ihrem Konto anzuzeigen, die für Sie AWS erstellt und verwaltet, wählen Sie im Navigationsbereich AWS verwaltete Schlüssel aus.

Wenn Sie mit SSE-KMS verschlüsselte Objekte hochladen oder darauf zugreifen, verwenden Sie AWS Signature Version 4 für zusätzliche Sicherheit. Weitere Informationen finden Sie unter Angeben der Signaturversion in der Anforderungsauthentifizierung im Benutzerhandbuch für Amazon Simple Storage Service.

Wenn Ihre Athena-Workloads eine große Datenmenge verschlüsseln, können Sie Amazon-S3-Bucket-Schlüssel verwenden, um die Kosten zu senken. Weitere Informationen finden Sie unter Reduzieren der Kosten für SSE-KMS mit Amazon-S3-Bucket-Schlüsseln im Benutzerhandbuch von Amazon Simple Storage Service.

Richtlinie für den Datenverkehr zwischen Netzwerken

Der Datenverkehr wird sowohl zwischen Athena und On-Premises-Anwendungen sowie zwischen Athena und Amazon S3 geschützt. Der Datenverkehr zwischen Athena und anderen -Services wie AWS Glue und AWS Key Management Serviceverwendet standardmäßig HTTPS.

- Für Datenverkehr zwischen Athena und On-Premises-Clients und Anwendungen werden Abfrageergebnisse, die zu JDBC- oder ODBC-Clients gestreamt werden, mit Transport Layer Security (TLS) verschlüsselt.

  Sie können eine der Konnektivitätsoptionen zwischen Ihrem privaten Netzwerk und AWSverwenden:

  - Eine Site-to-Site-VPN- AWS VPN Verbindung. Weitere Informationen finden Sie unter Was ist Site-to-Site VPN AWS VPN im AWS Site-to-Site VPN -Benutzerhandbuch.
  - Eine - AWS Direct Connect Verbindung. Weitere Informationen finden Sie unter Was ist AWS Direct Connect? im AWS Direct Connect -Benutzerhandbuch.
- Für Datenverkehr zwischen Athena- und Amazon-S3-Buckets verschlüsselt Transport Layer Security (TLS) die Objekte bei der Übertragung zwischen Athena und Amazon S3 sowie zwischen Athena und Kundenanwendungen, die darauf zugreifen; Sie sollten nur verschlüsselte Verbindungen über HTTPS (TLS) unter Verwendung von `aws:SecureTransport condition` auf Amazon-S3-Bucket-IAM-Richtlinien zulassen. Obwohl Athena derzeit den öffentlichen Endpunkt verwendet, um auf Daten in Amazon-S3-Buckets zuzugreifen, bedeutet dies nicht, dass die Daten das öffentliche Internet durchqueren. Der gesamte Datenverkehr zwischen Athena und Amazon S3 wird über das AWS Netzwerk geleitet und mit TLS verschlüsselt.
- Compliance-Programme – Amazon Athena erfüllt mehrere AWS Compliance-Programme, darunter SOC, PCI, FedRAMP und andere. Weitere Informationen finden Sie unterAWS-Services im Rahmen des Compliance-Programms.

Identity and Access Management in Athena

Amazon Athena verwendet AWS Identity and Access Management-(IAM)-Richtlinien, um den Zugriff auf Athena-Vorgänge einzuschränken. Eine vollständige Liste der Berechtigungen für Athena finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Athena in der Service-Autorisierungs-Referenz.

Wenn Sie IAM-Richtlinien verwenden, stellen Sie sicher, dass Sie die bewährten Methoden von IAM befolgen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Zu den Berechtigungen, die zum Ausführen von Athena-Abfragen erforderlich sind:

- Amazon-S3-Standorte, an denen die zugrunde liegenden Daten für die Abfrage gespeichert sind. Weitere Informationen finden Sie unter Identity and Access Management in Amazon S3 im Benutzerhandbuch von Amazon Simple Notification Service.
- Metadaten und Ressourcen, die Sie im AWS Glue Data Catalog speichern, wie z. B. Datenbanken und Tabellen, einschließlich zusätzlicher Aktionen für verschlüsselte Metadaten. Weitere Informationen finden Sie unter Einrichten von IAM-Berechtigungen für AWS Glue und Einrichten der Verschlüsselung in AWS Glue im Entwicklerhandbuch für AWS Glue.
- Athena-API-Aktionen Eine vollständige Liste mit API-Aktionen in Athena finden Sie unter Aktionen in der Amazon-Athena-API-Referenz.

Die folgenden Themen stellen weitere Informationen zu Berechtigungen für bestimmte Bereiche von Athena zur Verfügung.

Themen

- Zugriff auf vorbereitete Anweisungen zulassen
- Athena mit CalledVia-Kontextschlüsseln verwenden
- Zugriff auf einen Athena-Daten-Connector für externen Hive-Metastore zulassen
- Gewährung von Lambda-Funktionszugriff auf externe Hive-Metastores
- Beispiel für IAM-Berechtigungsrichtlinien zum Zulassen von Athena Federated Query
- Beispiel für IAM-Berechtigungsrichtlinien zum Zulassen von benutzerdefinierten Amazon-Athena-Funktionen (UDF)
- Erlauben des Zugriffs für ML mit Athena
- Aktivieren des föderierten Zugriffs auf die Athena-API

AWS verwaltete Richtlinien für Amazon Athena

Eine von AWS verwaltete Richtlinie ist eine eigenständige Richtlinie, die von AWS erstellt und verwaltet wird. Von AWS verwaltete Richtlinien stellen Berechtigungen für viele häufige Anwendungsfälle bereit, damit Sie beginnen können, Benutzern, Gruppen und Rollen Berechtigungen zuzuweisen.

Beachten Sie, dass AWS-verwaltete Richtlinien möglicherweise nicht die geringsten Berechtigungen für Ihre spezifischen Anwendungsfälle gewähren, da sie für alle AWS-Kunden verfügbar sind. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie kundenverwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind.

Die Berechtigungen, die in den von AWS verwalteten Richtlinien definiert sind, können nicht geändert werden. Wenn AWS Berechtigungen aktualisiert, die in einer von AWS verwalteten Richtlinie definiert werden, wirkt sich das Update auf alle Prinzipalidentitäten (Benutzer, Gruppen und Rollen) aus, denen die Richtlinie zugeordnet ist. AWS aktualisiert am wahrscheinlichsten eine von AWS verwaltete Richtlinie, wenn ein neuer AWS-Service gestartet wird oder neue API-Operationen für bestehende Dienste verfügbar werden.

Weitere Informationen finden Sie unter Von AWS verwaltete Richtlinien im IAM-Benutzerhandbuch.

Überlegungen bei der Verwendung verwalteter Richtlinien mit Athena

Verwaltete Richtlinien sind einfach zu nutzen und werden automatisch mit den erforderlichen Aktionen aktualisiert, wenn sich der Service weiterentwickelt. Beachten Sie bei der Verwendung von verwalteten Richtlinien mit Athena die folgenden Punkte:

- Um Amazon-Athena-Service-Aktionen für sich oder andere Benutzer von AWS Identity and Access Management (IAM) zuzulassen oder zu verweigern, hängen Sie Prinzipalen wie Benutzern oder Gruppen identitätsbasierte Richtlinien an.
- Jede identitätsbasierte Richtlinie besteht aus Anweisungen, die die zugelassenen oder nicht zugelassenen Aktionen definieren. Weitere Informationen und Schritt-für-Schritt-Anweisungen für das Zuweisen einer Richtlinie zu einem Benutzer finden Sie auf der Seite zum Anfügen verwalteter Richtlinien im IAM-Benutzerhandbuch. Eine Liste der Aktionen finden Sie in der Amazon Athena-API-Referenz.
- Vom Kunden verwaltete und eingebundene identitätsbasierte Richtlinien ermöglichen Ihnen, detailliertere Athena-Aktionen innerhalb einer Richtlinie zur Optimierung der Zugriffsregelung anzugeben. Wir empfehlen, dass Sie die `AmazonAthenaFullAccess`-Richtlinie als Ausgangspunkt verwenden und dann bestimmte, in der Amazon Athena-API-Referenz definierte Aktionen zulassen oder verweigern. Weitere Informationen zu Inline-Richtlinien finden Sie unter Verwaltete Richtlinien und Inline-Richtlinien im IAM-Benutzerhandbuch.
- Wenn Sie außerdem Prinzipale verwalten, die Verbindungen per JDBC herstellen, müssen Sie die JDBC-Treiber-Anmeldeinformationen für Ihre Anwendung bereitstellen. Weitere Informationen finden Sie unter Zugriff über JDBC- und ODBC-Verbindungen.
- Wenn Sie den AWS Glue-Datenkatalog verschlüsselt haben, müssen Sie zusätzliche Aktionen in den identitätsbasierten IAM-Richtlinien für Athena angeben. Weitere Informationen finden Sie unter Zugriff auf verschlüsselte Metadaten von Athena im AWS Glue Data Catalog.
- Wenn Sie Arbeitsgruppen erstellen und verwenden, müssen Sie sicherstellen, dass Ihre Richtlinien einen entsprechenden Zugriff auf Arbeitsgruppenaktionen enthalten. Detaillierte Informationen hierzu finden Sie unter the section called “ IAM-Richtlinien für den Zugriff auf Arbeitsgruppen” und the section called “Beispiel-Arbeitsgruppenrichtlinien”.

AWS Verwaltete Richtlinie: AmazonAthenaFullAccess

Die verwaltete Richtlinie `AmazonAthenaFullAccess` gewährt vollständigen Zugriff auf Athena.

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center:

  Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter Erstellen eines Berechtigungssatzes im AWS IAM Identity Center-Benutzerhandbuch.
- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch.

- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen IAM-Benutzer im IAM-Benutzerhandbuch.
  - (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter Hinzufügen von Berechtigungen zu einem Benutzer (Konsole) im IAM-Benutzerhandbuch.

Berechtigungsgruppierungen

Die `AmazonAthenaFullAccess`-Richtlinie wird in die folgenden Gruppen von Berechtigungen gruppiert.

- **`athena`** – Ermöglicht Prinzipalen Zugriff auf Athena-Ressourcen.
- **`glue`** – Ermöglicht Prinzipalen Zugriff auf AWS Glue-Datenbanken, Tabellen und Partitionen. Dies ist erforderlich, damit der Prinzipal die AWS Glue Data Catalog mit Athena verwenden kann.
- **`s3`** – Ermöglicht dem Prinzipal, Abfrageergebnisse aus Amazon S3 zu schreiben und zu lesen, öffentlich verfügbare Athena Datenbeispiele zu lesen, die sich in Amazon S3 befinden und Buckets aufzulisten. Dies ist erforderlich, damit der Prinzipal Athena verwenden kann, um mit Amazon S3 zu arbeiten.
- **`sns`** – Ermöglicht es Prinzipalen, Amazon-SNS-Themen aufzulisten und Themenattribute abzurufen. Dies ermöglicht es Prinzipalen, Amazon-SNS-Themen mit Athena für Überwachungs- und Warnzwecke zu verwenden.
- **`cloudwatch`** – Ermöglicht es Prinzipalen, CloudWatch-Alarme zu erstellen, zu lesen und zu löschen. Weitere Informationen finden Sie unter Steuern von Kosten und Überwachen von Abfragen mit CloudWatch Metriken und Ereignissen.
- **`lakeformation`** – Ermöglicht es Prinzipalen, temporäre Anmeldeinformationen für den Zugriff auf Daten an einem Data-Lake-Standort anzufordern, der bei Lake Formation registriert ist. Weitere Informationen finden Sie unter Zugriffskontrolle im Benutzerhandbuch von AWS Lake Formation.
- **`datazone`** – Ermöglicht es Prinzipalen, Amazon DataZone-Projekte, -Domänen und -Umgebungen aufzulisten. Informationen zur Verwendung von DataZone in Athena finden Sie unter Verwenden von Amazon DataZone in Athena.

- **`pricing`** – Bietet Zugriff auf AWS Billing and Cost Management. Weitere Informationen finden Sie unter GetProducts in der AWS Billing and Cost Management-API-Referenz.

```
{
    "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"
```

```
            ],
            "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 Verwaltete Richtlinie: AWSQuicksightAthenaAccess

`AWSQuicksightAthenaAccess` gewährt Zugriff auf Aktionen, den Amazon QuickSight für die Integration mit Athena benötigt. Sie können die `AWSQuicksightAthenaAccess`-Richtlinie an Ihre IAM-Identitäten anfügen. Fügen Sie diese Richtlinie nur zu Prinzipalen hinzu, die Amazon QuickSight mit Athena verwenden. Diese Richtlinie enthält einige Aktionen für Athena, die entweder veraltet und nicht in der aktuellen öffentlichen API eingebunden sind oder die ausschließlich mit dem JDBC- und ODBC-Treiber verwendet werden.

Berechtigungsgruppierungen

Die `AWSQuicksightAthenaAccess`-Richtlinie wird in die folgenden Gruppen von Berechtigungen gruppiert.

- **`athena`** – Ermöglicht dem Prinzipal, Abfragen auf Athena-Ressourcen auszuführen.
- **`glue`**— Ermöglicht Prinzipalen Zugriff auf AWS Glue-Datenbanken, Tabellen und Partitionen. Dies ist erforderlich, damit der Prinzipal die AWS Glue Data Catalog mit Athena benutzen kann.
- **`s3`** – Erlaubt dem Prinzipal, Abfrageergebnisse aus Amazon S3 zu schreiben und zu lesen.
- **`lakeformation`** – Ermöglicht es Prinzipalen, temporäre Anmeldeinformationen für den Zugriff auf Daten an einem Data-Lake-Standort anzufordern, der bei Lake Formation registriert ist. Weitere Informationen finden Sie unter Zugriffskontrolle im Benutzerhandbuch von 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-Updates für AWS-verwaltete Richtlinien

Anzeigen von Details zu Aktualisierungen für AWS-verwaltete Richtlinien für Athena, seit dieser Service mit der Verfolgung dieser Änderungen begonnen hat.

Änderung	Beschreibung	Datum
AmazonAthenaFullAccess – Aktualisierung der bestehenden Richtlinie	Die Berechtigungen `datazone:ListDomains` , `datazone:ListProjects` und `datazone:ListAccountEnvironments` und wurden hinzugefügt, damit Athena-Benutzer mit Amazon DataZone-Domains, -Projekten und -Umgebungen arbeiten können. Weitere Informationen finden Sie unter Verwenden von Amazon DataZone in Athena.	3. Januar 2024
AmazonAthenaFullAccess – Aktualisierung der bestehenden Richtlinie	Die Berechtigungen `glue:StartColumnStatisticsTaskRun` , `glue:GetColumnStatisticsTaskRun` und `glue:GetColumnStatisticsTaskRuns` und wurden hinzugefügt, um Athena das Recht zu geben, AWS Glue aufzurufen, um Statistiken für die kostenbasierte Optimierungsfunktion abzurufen. Weitere Informationen finden Sie unter Verwenden des kostenbasierten Optimierers.	3. Januar 2024
AmazonAthenaFullAccess – Aktualisierung der bestehenden Richtlinie	Athena hat `pricing:GetProducts` hinzugefügt, um Zugriff auf AWS Billing and Cost Management zu gewähren. Weitere Informationen finden Sie unter GetProducts in der AWS	25. Januar 2023

Änderung	Beschreibung	Datum
	Billing and Cost Management-API-Referenz.	
AmazonAthenaFullAccess – Aktualisierung der bestehenden Richtlinie	Athena hat `cloudwatch:GetMetricData` hinzugefügt, um CloudWatch-Metrikwerte abzurufen. Weitere Informationen finden Sie in der API-Referenz für Amazon CloudWatch unter GetMetricData.	14. November 2022
AmazonAthenaFullAccess und AWSQuicksightAthenaAccess – Aktualisierungen vorhandener Richtlinien	Athena hat `s3:PutBucketPublicAccessBlock` hinzugefügt, um die Blockierung des öffentlichen Zugriffs auf die von Athena erstellten Buckets zu ermöglichen.	7. Juli 2021
Athena hat damit begonnen, Änderungen zu verfolgen	Athena hat mit der Verfolgung von Änderungen für seine AWS-verwaltete Richtlinien begonnen.	7. Juli 2021

Zugriff über JDBC- und ODBC-Verbindungen

Stellen Sie die JDBC- oder OCBC-Treiber-Anmeldeinformationen für Ihre Anwendung bereit, um Zugriff auf AWS-Services und -Ressourcen wie Athena und Amazon-S3-Buckets zu erhalten. Wenn Sie den JDBC- oder ODBC-Treiber verwenden, stellen Sie sicher, dass die IAM-Berechtigungs-Richtlinie alle der unter AWS Verwaltete Richtlinie: AWSQuicksightAthenaAccess aufgeführten Aktionen enthält.

Wenn Sie IAM-Richtlinien verwenden, stellen Sie sicher, dass Sie die bewährten IAM-Methoden befolgen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Authentifizierungsmethoden

Die Athena-JDBC- und ODBC-Treiber unterstützen die SAML-2.0-basierte Authentifizierung, einschließlich der folgenden Identitätsanbieter:

- Active Directory Federation Services (AD FS)
- Azure Active Directory (AD)
- Okta
- PingFederate

Weitere Informationen finden Sie in den Installations- und Konfigurationshandbüchern für die jeweiligen Treiber, die im PDF-Format von den Seiten der JDBC- und ODBC-Treiber heruntergeladen werden können. Weitere Informationen finden Sie unter:

- Aktivieren des föderierten Zugriffs auf die Athena-API
- Verwenden von Lake Formation und den Athena-JDBC- und ODBC-Treibern für den Verbundzugriff auf Athena
- Konfigurieren von Single Sign-On mit ODBC, SAML 2.0 und dem Okta-Identitätsanbieter

Weitere Informationen zu den neuesten Versionen der JDBC- und ODBC-Treiber und deren Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC und Herstellen einer Verbindung mit Amazon Athena mit ODBC.

Zugriff auf Amazon S3

Sie können den Zugriff auf Amazon-S3-Standorte mithilfe von identitätsbasierten Richtlinien, Bucket-Ressourcenrichtlinien, Zugriffspunktrichtlinien oder einer beliebigen Kombination der oben genannten gewähren. Wenn Akteure mit Athena interagieren, werden ihre Berechtigungen durch Athena geleitet, um zu bestimmen, worauf Athena zugreifen kann. Das bedeutet, dass Benutzer die Berechtigung zum Zugriff auf Amazon-S3-Buckets haben müssen, um diese mit Athena abfragen zu können.

Wenn Sie IAM-Richtlinien verwenden, stellen Sie sicher, dass Sie die bewährten IAM-Methoden befolgen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Wenn Sie `aws:SourceIp` in Ihren Richtlinien konfigurieren, greift Athena mit der von Ihnen angegebenen IP-Adresse auf den Amazon-S3-Bucket zu. Sie können den Zugriff auf

Amazon-S3-Ressourcen nicht basierend auf den `aws:SourceVpc`- oder `aws:SourceVpce`-Bedingungsschlüsseln einschränken oder zulassen.

> **Note**
>
> Athena-Arbeitsgruppen, die die IAM Identity-Center-Authentifizierung verwenden, müssen die S3-Zugriffsberechtigungen so konfigurieren, dass sie die Verwendung der Weitergabe vertrauenswürdiger Identitäten ermöglichen. Weitere Informationen finden Sie unter S3-Zugriffsberechtigungen und Verzeichnisidentitäten im Benutzerhandbuch für Amazon Simple Storage Service.

Amazon-S3-Zugriffspunkte und Zugriffspunkt-Aliasse

Wenn Sie ein freigegebener Datensatz in einem Amazon-S3-Bucket haben, kann es schwierig sein, eine einzelne Bucket-Richtlinie zu verwalten, die den Zugriff für Hunderte von Anwendungsfällen verwaltet.

Amazon-S3-Bucket-Zugriffspunkte helfen bei der Lösung dieses Problems. Ein Bucket kann über mehrere Zugriffspunkte verfügen, von denen jeder eine Richtlinie verfügt, die den Zugriff auf den Bucket auf andere Weise steuert.

Für jeden von Ihnen erstellten Zugriffspunkt generiert Amazon S3 einen Alias, der den Zugriffspunkt darstellt. Da der Alias im Amazon-S3-Bucket-Namensformat vorliegt, können Sie den Alias in der `LOCATION`-Klausel Ihrer Anweisungen in `CREATE TABLE`-Athena verwenden. Der Zugriff von Athena auf den Bucket wird dann durch die Richtlinie für den Zugriffspunkt gesteuert, den der Alias darstellt.

Weitere Informationen finden Sie unter Tabellenspeicherort in Amazon S3 und Verwenden von Zugriffspunkten im Amazon-S3-Benutzerhandbuch.

Verwenden von CalledVia-Kontextschlüsseln

Für zusätzliche Sicherheit können Sie den globalen Bedingungskontextschlüssel aws:CalledVia verwenden. Der Schlüssel `aws:CalledVia` enthält eine geordnete Liste aller Services in der Kette, die im Auftrag des Auftraggebers Anforderungen ausgegeben haben. Durch Angeben des Prinzipalnamens des Athena-Dienstes `athena.amazonaws.com` für den `aws:CalledVia`-Kontextschlüssel können Sie Anfragen auf die von Athena gestellten beschränken. Weitere Informationen finden Sie unter Athena mit CalledVia-Kontextschlüsseln verwenden.

Weitere Ressourcen

Ausführliche Informationen und Beispiele zum Gewähren des Amazon-S3-Zugriffs finden Sie in den folgenden Ressourcen:

- Beispielexemplarische Vorgehensweisen: Verwalten des Zugriffs im Amazon-S3-Benutzerhandbuch.
- Wie kann ich kontoübergreifenden Zugriff auf Objekte bereitstellen, die sich in Amazon-S3-Buckets befinden? im AWS-Wissenscenter
- Kontoübergreifender Zugriff auf Amazon-S3-Buckets in Athena.

Kontoübergreifender Zugriff auf Amazon-S3-Buckets in Athena

Ein gängiges Amazon-Athena-Szenario ist das Gewähren des Zugriffs für Benutzer in einem Konto, das sich von dem des Bucket-Eigentümers unterscheidet, damit diese Abfragen ausführen können. Verwenden Sie in diesem Fall eine Bucket-Richtlinie zum Gewähren des Zugriffs.

> **Note**
>
> Weitere Informationen zum kontoübergreifenden Zugriff auf AWS Glue-Datenkataloge von Athena, siehe Kontoübergreifender Zugriff auf AWS Glue -Datenkataloge.

Die folgende Beispiel-Bucket-Richtlinie, die vom Bucket-Eigentümer erstellt und auf den Bucket `s3://DOC-EXAMPLE-BUCKET` angewendet wurde, gewährt allen Benutzern im Konto `123456789123` Zugriff. Hierbei handelt es sich um ein anderes Konto.

```
{
   "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/*"
         ]
      }
   ]
}
```

Um einem bestimmten Benutzer in einem Konto Zugriff zu gewähren, ersetzen Sie den Schlüssel `Principal` durch einen Schlüssel, der anstelle von `root` den Benutzer angibt. Verwenden Sie für das Benutzerprofil `Dave` beispielsweise `arn:aws:iam::123456789123:user/Dave`.

Kontoübergreifender Zugriff auf einen Bucket, der mit einem benutzerdefinierten AWS KMS-Schlüssel verschlüsselt ist

Wenn Sie über einen Amazon-S3-Bucket verfügen, der mit einem benutzerdefinierten AWS Key Management Service-(AWS KMS)-Schlüssel verschlüsselt ist, müssen Sie möglicherweise Benutzern von einem anderen Amazon-Web-Services-Konto aus Zugriff darauf gewähren.

Um einem Benutzer in Konto B Zugriff auf einen AWS KMS-verschlüsselten Bucket in Konto A gewähren zu können, sind folgende Berechtigungen erforderlich:

- Die Bucket-Richtlinie in Konto A muss Zugriff auf die von Konto B übernommene Rolle gewähren.
- Die AWS KMS-Schlüsselrichtlinie in Konto A muss Zugriff auf die Rolle gewähren, die der Benutzer in Konto B übernimmt.
- Die AWS Identity and Access Management (IAM)-Rolle, die von Konto B übernommen wird, muss sowohl Zugriff auf den Bucket als auch auf den Schlüssel in Konto A gewähren.

In den folgenden Verfahren wird beschrieben, wie jede dieser Berechtigungen erteilt wird.

So gewähren Sie dem Benutzer in Konto B Zugriff auf den Bucket in Konto A

- Überprüfen Sie in Konto A die S3-Bucket-Richtlinie und bestätigen Sie, dass eine Anweisung vorhanden ist, die den Zugriff von der Konto-ID von Konto B ermöglicht.

Die folgende Bucket-Richtlinie ermöglicht beispielsweise `s3:GetObject` den auf die Konto-ID `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"
        ]
      }
    }
  ]
}
```

So gewähren Sie dem Benutzer in Konto B Zugriff über die AWS KMS-Schlüsselrichtlinie in Konto A

1. Gewähren Sie in der AWS KMS-Schlüsselrichtlinie für Konto A der von Konto B übernommenen Rolle Berechtigungen für die folgenden Aktionen:

   - `kms:Encrypt`
   - `kms:Decrypt`
   - `kms:ReEncrypt*`
   - `kms:GenerateDataKey*`
   - `kms:DescribeKey`

   Im folgenden Beispiel wird nur einer IAM-Rolle der Schlüsselzugriff gewährt.

```
{
    "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. Überprüfen Sie unter Konto A die Schlüsselrichtlinie mithilfe der AWS Management Console-Richtlinienansicht.
3. Überprüfen Sie in der Schlüsselrichtlinie, ob in der folgenden Anweisung Konto B als Prinzipal aufgeführt wird.

```
"Sid": "Allow use of the key"
```

4. Wenn die `"Sid": "Allow use of the key"`-Anweisung nicht vorhanden ist, führen Sie die folgenden Schritte aus:

   a. Wechseln Sie, um die Schlüsselrichtlinie mithilfe der Standardansicht der Konsole anzuzeigen.

   b. Fügen Sie die Konto-ID von Konto B als externes Konto mit Zugriff auf den Schlüssel hinzu.

So gewähren Sie von der von Konto B übernommenen IAM-Rolle aus Zugriff auf den Bucket und den Schlüssel in Konto A

1. Öffnen Sie zuerst die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Öffnen Sie die IAM-Rolle, die dem Benutzer in Konto B zugeordnet ist.
3. Überprüfen Sie die Liste der Berechtigungsrichtlinien, die auf eine IAM-Rolle angewendet werden.

4. Stellen Sie sicher, dass eine Richtlinie angewendet wird, die Zugriff auf den Bucket gewährt.

   Die folgende Beispielanweisung gewährt der IAM-Rolle Zugriff auf die Operationen `s3:GetObject` und `s3:PutObject` im Bucket *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. Stellen Sie sicher, dass eine Richtlinie angewendet wird, die Zugriff auf den Schlüssel gewährt.

   > **Note**
   >
   > Wenn die von Konto B übernommene IAM-Rolle bereits über Administratorzugriff verfügt, müssen Sie keinen Zugriff auf den Schlüssel aus den IAM-Richtlinien des Benutzers gewähren.

   Die folgende Beispielanweisung gewährt der IAM-Rolle Zugriff auf die Verwendung des Schlüssels `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"
    }
  ]
}
```

Kontenübergreifender Zugriff auf Bucket-Objekte

Objekte, die von einem anderen Konto (Konto C) als dem Konto in Besitz des Buckets (Konto A) hochgeladen werden, erfordern möglicherweise explizite ACLs auf Objektebene, die Lesezugriff auf das abfragende Konto (Konto B) gewähren. Um diese Anforderung zu vermeiden, sollte Konto C eine Rolle in Konto A übernehmen, bevor Objekte in den Bucket von Konto A platziert werden. Weitere Informationen finden Sie unter How can I provide cross-account access to objects that are in Amazon S3 buckets?.

Differenzierter Zugriff auf Datenbanken und Tabellen in AWS Glue Data Catalog

Wenn Sie AWS Glue Data Catalog mit Amazon Athena verwenden, können Sie Richtlinien auf Ressourcenebene für die Datenkatalog-Objekte Datenbank und Tabelle definieren, die in Athena verwendet werden.

> **Note**
>
> Der Begriff „feinkörnige Zugriffskontrolle“ bezieht sich hier auf die Sicherheit auf Datenbank- und Tabellenebene. Hinweise zur Sicherheit auf Spalten-, Zeilen- und Zellen-Ebene finden Sie unter Data filtering and cell-level security in Lake Formation (Daten-Filterung und Sicherheit auf Zellen-Ebene in Lake Formation).

Berechtigungen auf Ressourcenebene werden in identitätsbasierten IAM-Richtlinien definiert.

> **Important**
>
> In diesem Abschnitt werden Berechtigungen auf Ressourcenebene in identitätsbasierten IAM-Richtlinien besprochen. Diese unterscheiden sich von ressourcenbasierten Richtlinien.

Weitere Informationen zu den Unterschieden finden Sie unter Identitätsbasierte Richtlinien und ressourcenbasierte Richtlinien im IAM-Benutzerhandbuch.

Beachten Sie für diese Aufgaben folgende Themen:

Zur Ausführung dieser Aufgabe	Beachten Sie folgendes Thema
Erstellen einer IAM-Richtlinie, die differenzierten Zugriff auf Ressourcen definiert	Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.
Informationen zu den in AWS Glue verwendeten identitätsbasierten Richtlinien (IAM-Richtlinien)	Identitätsbasierte Richtlinien (IAM-Richtlinien) im AWS Glue-Entwicklerhandbuch.

In diesem Abschnitt

Einschränkungen

Berücksichtigen Sie die folgenden Einschränkungen bei Verwendung der differenzierten Zugriffskontrolle mit AWS Glue Data Catalog und Athena:

- Für IAM Identity Center aktivierte Athena-Arbeitsgruppen muss Lake Formation für die Verwendung von IAM-Identity-Center-Identitäten konfiguriert sein. Weitere Informationen finden Sie unter Integration von IAM Identity Center im AWS Lake Formation-Entwicklerhandbuch.

- Sie können den Zugriff nur auf Datenbanken und Tabellen beschränken. Differenzierte Zugriffskontrollen werden auf Tabellenebene angewendet und Sie können den Zugriff auf einzelne Partitionen in einer Tabelle nicht einschränken. Weitere Informationen finden Sie unter Tabellenpartitionen und -versionen in AWS Glue.
- AWS Glue Data Catalog enthält folgende Ressourcen: `CATALOG`, `DATABASE`, `TABLE` und `FUNCTION`.

  > **Note**
  >
  > Ressourcen in dieser Liste, die zwischen Athena und AWS Glue Data Catalog gleich sind, sind für jedes Konto `TABLE`, `DATABASE` und `CATALOG`. `Function` ist spezifisch für AWS Glue. Zum Löschen von Aktionen in Athena müssen Sie Berechtigungen für AWS Glue-Aktionen einschließen. Siehe Beispiele für differenzierte Berechtigungen für Tabellen und Datenbanken.

  Die Hierarchie lautet wie folgt: `CATALOG` ist ein Vorgänger von allen `DATABASES` in jedem Konto und jede `DATABASE` ist ein Vorgänger für alle ihre `TABLES` und `FUNCTIONS`. Beispiel: Für eine Tabelle mit dem Namen `table_test` einer Datenbank `db` im Katalog in Ihrem Konto sind ihre Vorgänger `db` und der Katalog in Ihrem Konto. Für die `db`-Datenbank ist ihr Vorgänger der Katalog in Ihrem Konto und ihre untergeordneten Elemente sind Tabellen und Funktionen. Weitere Informationen über die hierarchische Struktur von Ressourcen finden Sie unter Liste der ARNs im Datenkatalog im AWS Glue Entwicklerhandbuch.
- Für jede Nicht-Löschaktion in Athena auf einer Ressource, z. B. `CREATE DATABASE`, `CREATE TABLE`, `SHOW DATABASE`, `SHOW TABLE` oder `ALTER TABLE`, benötigen Sie Berechtigungen zum Aufrufen dieser Aktion auf dieser Ressource (Tabelle oder Datenbank) und alle Vorgänger dieser Ressource in Datenkatalog. Beispielsweise sind die Vorgänger einer Tabelle die Datenbank, zu der sie gehört, und der Katalog für das Konto. Für eine Datenbank ist ihr Vorgänger der Katalog für das Konto. Siehe Beispiele für differenzierte Berechtigungen für Tabellen und Datenbanken.
- Für eine Löschaktion in Athena, wie z. B. `DROP DATABASE` oder `DROP TABLE`, müssen Sie auch Berechtigungen zum Aufrufen der Löschaktion auf allen Vorgängern und untergeordneten Elemente der Ressource im Datenkatalog besitzen. Wenn Sie beispielsweise eine Datenbank löschen, benötigen Sie Berechtigungen für die Datenbank, den Katalog, der ihr untergeordnetes Element ist, sowie alle Tabellen und benutzerdefinierten Funktionen, die untergeordnete Elemente von ihnen sind. Eine Tabelle hat keine untergeordneten Elemente. Zur Ausführung von `DROP TABLE` müssen Sie Berechtigungen für diese Aktion auf der Tabelle der Datenbank, zu dem sie

gehört, und den Katalog haben. Siehe Beispiele für differenzierte Berechtigungen für Tabellen und Datenbanken.

AWS Glue-Zugriff auf Ihren Katalog und Ihre Datenbank per AWS-Region

Damit Athena mit dem AWS Glue arbeiten kann, ist eine AWS-Region-Richtlinie erforderlich, die Zugriff auf Ihre Datenbank und auf die AWS Glue Data Catalog in Ihrem Konto per gewährt. Um Datenbanken zu erstellen, ist ebenfalls eine `CreateDatabase`-Genehmigung erforderlich. Ersetzen Sie in der folgenden Beispielrichtlinie die AWS-Region, AWS-Konto-ID und den Datenbanknamen durch Ihre eigenen.

```
{
   "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"
   ]
}
```

Tabellenpartitionen und -versionen in AWS Glue

Tabellen können in AWS Glue Partitionen und Versionen haben. Tabellenversionen und -partitionen gelten nicht als unabhängige Ressourcen in AWS Glue. Zugriff auf die Tabellenversionen und -partitionen wird durch Erteilen des Zugriffs auf die Tabelle und Vorgängerressourcen für die Tabelle gewährt.

Für die Zwecke der differenzierten Zugriffskontrolle gelten die folgenden Zugriffsberechtigungen:

- Differenzierte Zugriffskontrollen gelten auf Tabellenebene. Sie können den Zugriff nur auf Datenbanken und Tabellen beschränken. Wenn Sie zum Beispiel den Zugriff auf eine partitionierte Tabelle gewähren, gilt dies für alle Partitionen in der Tabelle. Sie können den Zugriff nicht auf einzelne Partitionen innerhalb einer Tabelle beschränken.

> **Important**
>
> Um Aktionen in AWS Glue auf Partitionen auszuführen, sind Berechtigungen für Partitionsaktionen auf Katalog-, Datenbank- und Tabellenebene erforderlich. Der Zugriff auf Partitionen innerhalb einer Tabelle ist nicht ausreichend. Um z. B. `GetPartitions` auf Tabelle `myTable` in der Datenbank `myDB` auszuführen, müssen Sie `glue:GetPartitions` die Berechtigungen für den Katalog, die `myDB`-Datenbank und die `myTable`-Ressourcen erteilen.

- Differenzierte Zugriffskontrollen gelten nicht für Tabellenversionen. Wie bei Partitionen wird der Zugriff auf frühere Versionen einer Tabelle durch den Zugriff auf die Tabellenversions-APIs in AWS Glue in der Tabelle und für die Tabellenvorfahren gewährt.

Weitere Informationen zu Berechtigungen für AWS Glue-Aktionen finden Sie unter AWS Glue-API-Berechtigungen: Referenz für Aktionen und Ressourcen im AWS Glue Entwicklerhandbuch.

Beispiele für differenzierte Berechtigungen für Tabellen und Datenbanken

In der folgenden Tabelle sind Beispiele für identitätsbasierte IAM-Richtlinien aufgeführt, die einen differenzierten Zugriff auf Datenbanken und Tabellen in Athena ermöglichen. Wir empfehlen Ihnen, mit diesen Beispielen zu beginnen und diese Ihren Anforderungen entsprechend anzupassen, um bestimmte Aktionen für einzelne Datenbanken und Tabellen zuzulassen oder zu verweigern.

Zu diesen Beispielen gehört der Zugriff auf Datenbanken und Kataloge, sodass Athena und AWS Glue zusammenarbeiten können. Nehmen Sie bei mehreren AWS-Regionen diese Richtlinie für jede Ihrer Datenbanken und ihrer Kataloge in einer Zeile für jede Region auf.

Ersetzen Sie in den Beispielen außerdem die Namen der `example_db`-Datenbank und `test`-Tabelle durch die Namen Ihrer Datenbanken und Tabellen.

DDL-Anweisung	Beispiel für eine IAM-Zugriffsrichtlinie, die Zugriff auf die Ressource erteilt

ALTER DATABASE	Hiermit können Sie die Eigenschaften für die `example_db` -Datenbank ändern.
	
	`{`
	`"Effect": "Allow",`

DDL-Anweisung	Beispiel für eine IAM-Zugriffsrichtlinie, die Zugriff auf die Ressource erteilt
	"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	Hiermit können Sie die Datenbank mit dem Namen `example_db` erstellen.
	{
	"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* "
	]
	}

DDL-Anweisung	Beispiel für eine IAM-Zugriffsrichtlinie, die Zugriff auf die Ressource erteilt

CREATE TABLE	Hiermit können Sie eine Tabelle mit dem Namen `test` in der `example_db` -Datenbank erstellen.
	
	{
	"Sid": "DatabasePermissions",
	"Effect": "Allow",
	"Action": [
	"glue:GetDatabase",
	"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_db* /*test*"
	]
	}

DDL-Anweisung	Beispiel für eine IAM-Zugriffsrichtlinie, die Zugriff auf die Ressource erteilt

DROP DATABASE	Hiermit können Sie die `example_db` -Datenbank mit allen darin enthaltenen Tabellen löschen.
	
	`{`
	`"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 /*"`
	`]`
	`}`

DDL-Anweisung	Beispiel für eine IAM-Zugriffsrichtlinie, die Zugriff auf die Ressource erteilt

DROP TABLE	Hiermit können Sie eine partitionierte Tabelle mit dem Namen `test` in der `example_db` -Datenbank löschen. Falls Ihre Tabelle keine Partitionen hat, nehmen Sie keine Aktionen für Partitionen auf.
	
	`{`
	`"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"`
	`]`
	`}`

DDL-Anweisung	Beispiel für eine IAM-Zugriffsrichtlinie, die Zugriff auf die Ressource erteilt
MSCK REPAIR TABLE	Ermöglicht das Aktualisieren von Katalog-Metadaten, nachdem Sie Hive-kompatible Partitionen zur Tabelle mit dem Namen `test` in die `example_db` -Datenbank hinzugefügt haben. `{ "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	Hiermit können Sie alle Datenbanken in AWS Glue Data Catalog auflisten. `{ "Effect": "Allow", "Action": [ "glue:GetDatabase", "glue:GetDatabases" ], "Resource": [ "arn:aws:glue: us-east-1 :123456789012 :catalog", "arn:aws:glue: us-east-1 :123456789012 :database/*" ] }`

DDL-Anweisung	Beispiel für eine IAM-Zugriffsrichtlinie, die Zugriff auf die Ressource erteilt
SHOW TABLES	Hiermit können Sie alle Tabellen in der `example_db` -Datenbank auflisten. `{ "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 /*" ] }`

Kontoübergreifender Zugriff auf AWS Glue -Datenkataloge

Sie können die kontoübergreifende AWS Glue Katalogfunktion von Athena verwenden, um einen Katalog von einem AWS Glue anderen Konto als Ihrem eigenen aus zu registrieren. Nachdem Sie die erforderlichen IAM-Berechtigungen für den Katalog konfiguriert AWS Glue und ihn als DataCatalogAthena-Ressource registriert haben, können Sie mit Athena kontoübergreifende Abfragen ausführen. Informationen zur Verwendung der Athena-Konsole zum Registrieren eines Katalogs von einem anderen Konto finden Sie unter Registrieren eines AWS Glue Data Catalog von einem anderen Konto.

Weitere Informationen zum kontoübergreifenden Zugriff finden Sie unter Gewähren von kontenübergreifendem Zugriff im AWS Glue Entwicklerhandbuch.AWS Glue

Bevor Sie beginnen

Da dieses Feature vorhandene Athena-`DataCatalog`-Ressourcen-APIs und -Funktionen verwendet, um den kontoübergreifenden Zugriff zu ermöglichen, empfehlen wir Ihnen, die folgenden Ressourcen zu lesen, bevor Sie beginnen:

- Herstellen von Verbindungen mit Datenquellen- Enthält Themen zur Verwendung von Athena mit AWS Glue, Hive oder Lambda-Datenkatalogquellen.
- Datenkatalog-Beispielrichtlinien – Zeigt, wie Richtlinien geschrieben werden, die den Zugriff auf Datenkataloge steuern.
- Verwendung der AWS CLI mit Hive-Metastores— Zeigt, wie Metastore AWS CLI mit Hive verwendet werden, enthält jedoch Anwendungsfälle, die auf andere Datenquellen anwendbar sind.

Überlegungen und Einschränkungen

Derzeit gelten für den kontoübergreifenden Zugriff auf den AWS Glue Athena-Katalog die folgenden Einschränkungen:

- Die Funktion ist nur verfügbar, AWS-Regionen wenn die Athena-Engine-Version 2 oder höher unterstützt wird. Weitere Informationen über Athena-Engine-Versionen finden Sie unter Athena-Engine-Versionierung. Informationen zum Aktualisieren der Engine-Version für eine Arbeitsgruppe finden Sie unter Ändern von Athena-Engine-Versionen.
- Wenn Sie ein anderes Konto AWS Glue Data Catalog in Ihrem Konto registrieren, erstellen Sie eine regionale `DataCatalog` Ressource, die nur mit den Daten des anderen Kontos in dieser bestimmten Region verknüpft ist.
- Derzeit werden `CREATE VIEW`-Anweisungen, die einen kontoübergreifenden AWS Glue -Katalog enthalten, nicht unterstützt.
- Kataloge, die mit AWS verwalteten Schlüsseln verschlüsselt wurden, können nicht kontenübergreifend abgefragt werden. Verwenden Sie für Kataloge, die Sie kontenübergreifend abfragen möchten, stattdessen vom Kunden verwaltete Schlüssel (). `KMS_CMK` Informationen zu den Unterschieden zwischen vom Kunden verwalteten Schlüsseln und AWS verwalteten Schlüsseln finden Sie im AWS Key Management Service Entwicklerhandbuch unter AWS Kundenschlüssel und Schlüssel.

Erste Schritte

Im folgenden Szenario möchte das Konto „Kreditnehmer“ (666666666666) eine `SELECT` Abfrage ausführen, die sich auf den AWS Glue Katalog bezieht, der zum Konto „Besitzer“ (999999999999) gehört, wie im folgenden Beispiel:

```
SELECT * FROM ownerCatalog.tpch1000.customer
```

Im folgenden Verfahren wird in den Schritten 1a und 1b gezeigt, wie dem Kreditnehmerkonto sowohl vom Kreditnehmer als auch vom Eigentümer aus Zugriff auf die Ressourcen des Eigentümerkontos gewährt wird. AWS Glue Das Beispiel gewährt Zugriff auf die Datenbank `tpch1000` und die Tabelle `customer`. Ändern Sie diese Beispielnamen entsprechend Ihren Anforderungen.

Schritt 1a: Erstellen Sie eine Kreditnehmerrolle mit einer Richtlinie für den Zugriff auf die Ressourcen des Eigentümers AWS Glue

Um eine Rolle für ein Kreditnehmerkonto mit einer Richtlinie für den Zugriff auf die AWS Glue Ressourcen des Eigentümerkontos zu erstellen, können Sie die AWS Identity and Access Management (IAM-) Konsole oder die IAM-API verwenden. Die folgenden Verfahren verwenden die IAM-Konsole.

Um eine Kreditnehmerrolle und eine Richtlinie für den Zugriff auf die Ressourcen des Eigentümerkontos zu erstellen AWS Glue

1. Melden Sie sich über das Kreditnehmerkonto bei der IAM-Konsole unter https://console.aws.amazon.com/iam/ an.
2. Erweitern Sie im Navigationsbereich die Option Zugriffsverwaltung, und wählen Sie dann Richtlinien aus.
3. Wählen Sie Richtlinie erstellen aus.
4. Wählen Sie für Richtlinien-Editor die Option JSON aus.
5. Geben Sie im Richtlinien-Editor die folgende Richtlinie ein und ändern Sie sie dann entsprechend Ihren Anforderungen:

```
{
    "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. Wählen Sie Weiter aus.
7. Geben Sie auf der Seite Überprüfen und erstellen in das Feld Richtlinienname einen Namen für die Richtlinie ein (z. B.**`CrossGluePolicyForBorrowerRole`**).
8. Wählen Sie Richtlinie erstellen aus.
9. Wählen Sie im Navigationsbereich Rollen aus.
10. Wählen Sie Rolle erstellen aus.
11. Wählen Sie auf der Seite Vertrauenswürdige Entität auswählen AWS-Kontodie Option und anschließend Weiter aus.
12. Geben Sie auf der Seite „Berechtigungen hinzufügen“ den Namen der Richtlinie, die Sie erstellt haben, in das Suchfeld ein (z. B.**`CrossGluePolicyForBorrowerRole`**).
13. Aktivieren Sie das Kontrollkästchen neben dem Richtliniennamen und wählen Sie dann Weiter aus.
14. Geben Sie auf der Seite Name, review, and create (Benennen, überprüfen und erstellen) für Role name (Rollenname) einen Namen für die Rolle ein (z. B. **`CrossGlueBorrowerRole`**).
15. Wählen Sie Rolle erstellen aus.

Schritt 1b: Erstellen Sie eine Eigentümerrichtlinie, um dem Kreditnehmer AWS Glue Zugriff zu gewähren

Um vom Besitzerkonto (999999999999) aus AWS Glue Zugriff auf die Rolle des Kreditnehmers zu gewähren, können Sie die Konsole oder den API-Vorgang verwenden. AWS Glue AWS Glue PutResourcePolicy Das folgende Verfahren verwendet die Konsole. AWS Glue

Um dem Eigentümer AWS Glue Zugriff auf das Kreditnehmerkonto zu gewähren

1. Melden Sie sich über das Besitzerkonto bei der AWS Glue Konsole unter https://console.aws.amazon.com/glue/ an.
2. Erweitern Sie im Navigationsbereich den Eintrag Data Catalog und wählen Sie dann Katalogeinstellungen aus.
3. Geben Sie im Feld Berechtigungen eine Richtlinie wie die folgende ein. Geben Sie als *`Rollenname`* die Rolle ein, die der Kreditnehmer in Schritt 1a erstellt hat (z. B.). **`CrossGlueBorrowerRole`** Wenn Sie den Berechtigungsumfang erweitern möchten, können Sie das Platzhalterzeichen `*` sowohl für den Datenbank- als auch für den Tabellenressourcentyp verwenden.

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

Wenn Sie fertig sind, empfehlen wir Ihnen, mithilfe der AWS Glue API einige kontoübergreifende Testaufrufe durchzuführen, um zu überprüfen, ob die Berechtigungen erwartungsgemäß konfiguriert sind.

Schritt 2: Der Kreditnehmer registriert das Konto AWS Glue Data Catalog , das zum Eigentümer gehört

Das folgende Verfahren zeigt Ihnen, wie Sie die Athena-Konsole verwenden, um den AWS Glue Data Catalog im Amazon-Web-Services-Besitzerkonto als Datenquelle zu konfigurieren. Informationen zur Verwendung von API-Vorgängen anstelle der Konsole zum Registrieren des Katalogs finden Sie unter Verwenden der API zum Registrieren eines Athena-Datenkatalogs, der zum Eigentümerkonto gehört.

Um ein Konto zu registrieren, das zu einem anderen Konto AWS Glue Data Catalog gehört

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.

3. Erweitern Sie Administration und wählen Sie dann Datenquellen aus.
4. Wählen Sie oben rechts Create data source (Datenquelle erstellen) aus.
5. Wählen Sie auf der Seite Datenquelle auswählen für Datenquellen die Option S3 - AWS Glue Data Catalog und wählen Sie dann Weiter aus.
6. Wählen Sie auf der Seite Enter data source details (Details zur Datenquelle eingeben) im Abschnitt AWS Glue Data Catalog für Choose an AWS Glue Data Catalog (auswählen) die Option AWS Glue Data Catalog in another account (in einem anderen Konto) aus.
7. Geben Sie für Data source details (Datenquellen-Details) die folgenden Informationen ein:
   - Data source name (Datenquellenname) – Geben Sie den Namen ein, den Sie in Ihren SQL-Abfragen verwenden möchten, um auf den Datenkatalog im anderen Konto zu verweisen.
   - Beschreibung – (Optional) Geben Sie eine Beschreibung des Datenkatalogs im anderen Konto ein.
   - Katalog-ID – Geben Sie die 12-stellige Amazon-Web-Services-Konto-ID des Kontos ein, zu dem der Datenkatalog gehört. Die Amazon-Web-Services-Konto-ID ist die Katalog-ID.
8. (Optional) Erweitern Sie Tags und geben Sie Schlüssel-Wert-Paare ein, die Sie mit der Datenquelle verknüpfen möchten. Weitere Informationen zu Tags erhalten Sie unter Markieren von Athena-Ressourcen.
9. Wählen Sie Weiter aus.
10. Überprüfen Sie auf der Seite Review and create (Überprüfen und erstellen) die von Ihnen bereitgestellten Informationen, und wählen Sie dann Create data source (Datenquelle erstellen)

aus. Die Seite Data source details (Datenquellen-Details) listet die Datenbanken und Tags für den von Ihnen registrierten Datenkatalog auf.

11. Wählen Sie Data sources (Datenquellen) aus. Der von Ihnen registrierte Datenkatalog wird in der Spalte Data source name (Datenquellen-Name) aufgeführt.
12. Um Informationen zum Datenkatalog anzuzeigen oder zu bearbeiten, wählen Sie den Katalog und dann Actions (Aktionen), Edit (Bearbeiten) aus.
13. Um den neuen Datenkatalog zu löschen, wählen Sie den Katalog und dann Actions (Aktionen), Delete (Löschen) aus.

Schritt 3: Der Empfänger sendet eine Abfrage

Der Kreditnehmer sendet mithilfe des Katalogs eine Abfrage, die auf den Katalog verweist. Datenbank. Tabellensyntax, wie im folgenden Beispiel:

```
SELECT * FROM ownerCatalog.tpch1000.customer
```

Anstatt die vollqualifizierte Syntax zu verwenden, kann der Kreditnehmer den Katalog auch kontextbezogen angeben, indem er ihn über die weitergibt. QueryExecutionContext

Zusätzliche Amazon-S3-Berechtigungen

- Wenn das Kreditnehmerkonto eine Athena-Abfrage verwendet, um neue Daten in eine Tabelle im Besitzerkonto zu schreiben, hat der Eigentümer nicht automatisch Zugriff auf diese Daten in Amazon S3, obwohl die Tabelle im Konto des Eigentümers vorhanden ist. Dies liegt daran, dass der Kreditnehmer der Objekteigentümer der Informationen in Amazon S3 ist, sofern nicht anders konfiguriert. Um dem Eigentümer Zugriff auf die Daten zu gewähren, müssen Sie in einem zusätzlichen Schritt die Berechtigungen für die Objekte entsprechend festlegen.
- Bestimmte kontoübergreifende DDL-Vorgänge wie MSCK REPAIR TABLE erfordern Amazon-S3-Berechtigungen. Wenn das Kreditnehmerkonto beispielsweise eine kontoübergreifende `MSCK REPAIR` Operation mit einer Tabelle im Besitzerkonto durchführt, deren Daten sich in einem S3-Bucket für das Besitzerkonto befinden, muss dieser Bucket der Rolle, die der Kreditnehmer übernommen hat, Berechtigungen gewähren, damit die Abfrage erfolgreich ist.

Informationen zum Erteilen von Bucket-Berechtigungen finden Sie unter Wie lege ich ACL-Bucket-Berechtigungen fest? im Benutzerhandbuch für Amazon Simple Storage Service.

Dynamisches Verwenden eines Katalogs

In einigen Fällen möchten Sie möglicherweise schnell einen kontoübergreifenden AWS Glue -Katalog testen, ohne ihn registrieren zu müssen. Sie können kontoübergreifende Abfragen dynamisch ausführen, ohne das Ressourcenobjekt `DataCatalog` zu erstellen, wenn die erforderlichen IAM- und Amazon-S3-Berechtigungen wie zuvor in diesem Dokument beschrieben korrekt konfiguriert sind.

Verwenden Sie die Syntax im folgenden Beispiel, um explizit auf einen Katalog ohne Registrierung zu verweisen:

```
SELECT * FROM "glue:arn:aws:glue:us-east-1:999999999999:catalog".tpch1000.customer
```

Verwenden Sie das Format „`glue:<arn>`“, wobei *`<arn>`* der AWS Glue Data Catalog -ARN ist, den Sie verwenden möchten. In diesem Beispiel verwendet Athena diese Syntax, um dynamisch auf den AWS Glue Datenkatalog des Kontos 999999999999 zu verweisen, als ob Sie ein separates Objekt dafür erstellt hätten. `DataCatalog`

Hinweise zur Verwendung dynamischer Kataloge

Beachten Sie beim Verwenden von dynamischen Katalogen die folgenden Punkte.

- Die Verwendung eines dynamischen Katalogs erfordert die IAM-Berechtigungen, die Sie normalerweise für Athena-Datenkatalog-API-Vorgänge verwenden. Der Hauptunterschied besteht darin, dass der Name der Datenkatalog-Ressource der `glue:*`-Namenskonvention folgt.
- Der Katalog-ARN muss zu derselben Region gehören, in der die Abfrage ausgeführt wird.
- Wenn Sie einen dynamischen Katalog in einer DML-Abfrage oder -Ansicht verwenden, umgeben Sie ihn mit Escapezeichen in doppelten Anführungszeichen (\"). Wenn Sie einen dynamischen Katalog in einer DDL-Abfrage verwenden, umgeben Sie ihn mit Backtick-Zeichen (`).

Verwenden der API zum Registrieren eines Athena-Datenkatalogs, der zum Eigentümerkonto gehört

Anstatt die Athena-Konsole wie in Schritt 2 beschrieben zu verwenden, ist es möglich, API-Vorgänge zu verwenden, um den Datenkatalog zu registrieren, der zum Eigentümerkonto gehört.

Der Ersteller der DataCatalogAthena-Ressource muss über die erforderlichen Berechtigungen verfügen, um den CreateDataCatalogAthena-API-Vorgang auszuführen. Abhängig von Ihren Anforderungen kann der Zugriff auf zusätzliche API-Vorgänge erforderlich sein. Weitere Informationen finden Sie unter Datenkatalog-Beispielrichtlinien.

Der folgende `CreateDataCatalog` Anforderungstext registriert einen AWS Glue Katalog für den kontoübergreifenden Zugriff:

```
# 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"
}
```

Der folgende Beispielcode verwendet einen Java-Client, um das `DataCatalog`-Objekt zu erstellen.

```
# 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);
```

Nach diesen Schritten sollte der Kreditnehmer sehen, `ownerCatalog` wann er den ListDataCatalogsAPI-Vorgang aufruft.

Weitere Informationen finden Sie auch unter

- Registrieren eines AWS Glue Data Catalog von einem anderen Konto
- Im Leitfaden AWS Prescriptive Guidance Patterns können Sie den kontoübergreifenden Zugriff auf eine gemeinsam genutzte Website AWS Glue Data Catalog über Amazon Athena konfigurieren.
- Kontenübergreifende AWS Glue Data Catalog Abfragen mit Amazon Athena im AWS Big Data-Blog
- Gewähren von kontenübergreifendem Zugriff im AWS Glue -Entwicklerhandbuch

Zugriff auf verschlüsselte Metadaten von Athena im AWS Glue Data Catalog

Wenn Sie AWS Glue Data Catalog mit Amazon Athena verwenden, können Sie die Verschlüsselung in AWS Glue Data Catalog mithilfe der AWS Glue-Konsole oder der API aktivieren. Informationen hierzu finden Sie unter Verschlüsseln Ihres Datenkatalogs im AWS Glue-Entwicklerhandbuch.

Wenn das AWS Glue Data Catalog verschlüsselt ist, müssen Sie die folgenden Aktionen allen Richtlinien hinzufügen, die für den Zugriff auf Athena verwendet werden:

```
{
 "Version": "2012-10-17",
 "Statement": {
 "Effect": "Allow",
     "Action": [
           "kms:GenerateDataKey",
           "kms:Decrypt",
           "kms:Encrypt"
      ],
     "Resource": "(arn of the key used to encrypt the catalog)"
   }
}
```

Wenn Sie IAM-Richtlinien verwenden, stellen Sie sicher, dass Sie die bewährten Methoden von IAM befolgen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Zugriff auf Arbeitsgruppen und Tags

Eine Arbeitsgruppe ist eine von Athena verwaltete Ressource. Wenn Ihre Arbeitsgruppenrichtlinie daher Aktionen verwaltet, die `workgroup` als Eingabe verwenden, müssen Sie den ARN der Arbeitsgruppe wie folgt angeben, wobei *`workgroup-name`* der Name Ihrer Arbeitsgruppe ist:

```
"Resource": [arn:aws:athena:region:AWSAcctID:workgroup/workgroup-name]
```

Geben Sie zum Beispiel für eine Arbeitsgruppe mit dem Namen `test_workgroup` in der Region `us-west-2` für Amazon-Web-Services-Konto `123456789012` die Arbeitsgruppe als Ressource mit dem folgenden ARN an:

```
"Resource":["arn:aws:athena:us-east-2:123456789012:workgroup/test_workgroup"]
```

Um auf Arbeitsgruppen zugreifen zu können, die Trusted Identity Propagation (TIP) unterstützen, müssen IAM-Identity-Center-Benutzer dem `IdentityCenterApplicationArn` zugewiesen werden, der von der Antwort der Athena-API-Aktion GetWorkGroup zurückgegeben wird.

- Eine Liste mit Arbeitsgruppenrichtlinien finden Sie unter the section called “Beispiel-Arbeitsgruppenrichtlinien”.

- Eine Liste mit Tag-basierten Richtlinien für Arbeitsgruppen finden Sie unter Tagbasierte IAM-Zugriffssteuerungsrichtlinien.
- Weitere Informationen zum Erstellen von IAM-Richtlinien für Arbeitsgruppen finden Sie unter IAM-Richtlinien für den Zugriff auf Arbeitsgruppen.
- Eine vollständige Liste mit Amazon-Athena-Aktionen finden Sie unter den Namen von API-Aktionen in der Amazon-Athena-API-Referenz.
- Weitere Informationen zu IAM-Richtlinien finden Sie unter Erstellen von Richtlinien mit dem visuellen Editor im IAM-Benutzerhandbuch.

Wenn Sie IAM-Richtlinien verwenden, stellen Sie sicher, dass Sie die bewährten Methoden von IAM befolgen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Zugriff auf vorbereitete Anweisungen zulassen

In diesem Thema werden IAM-Berechtigungen für vorbereitete Anweisungen in Amazon Athena behandelt. Wenn Sie IAM-Richtlinien verwenden, stellen Sie sicher, dass Sie die bewährten Methoden von IAM befolgen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Weitere Informationen zu vorbereiteten Anweisungen finden Sie unter Verwenden von parametrisierten Abfragen.

Die folgenden IAM-Berechtigungen sind für das Erstellen, Verwalten und Ausführen vorbereiteter Anweisungen erforderlich.

```
athena:CreatePreparedStatement
athena:UpdatePreparedStatement
athena:GetPreparedStatement
athena:ListPreparedStatements
athena:DeletePreparedStatement
```

Verwenden Sie diese Berechtigungen, wie in der folgenden Tabelle dargestellt.

Aktion	Nutzen Sie diese Berechtigungen
Ausführen einer PREPARE-Abfrage	`athena:StartQueryExecution` `athena:CreatePreparedStatement`

Aktion	Nutzen Sie diese Berechtigungen
Führen Sie eine PREPARE-Abfrage erneut aus, um eine vorhandene vorbereitete Anweisung zu aktualisieren	`athena:StartQueryExecution` `athena:UpdatePreparedStatement`
Ausführen einer EXECUTE-Abfrage	`athena:StartQueryExecution` `athena:GetPreparedStatement`
Ausführen einer DEALLOCATE PREPARE-Abfrage	`athena:StartQueryExecution` `athena:DeletePreparedStatement`

Beispiel

Im folgenden Beispiel der IAM-Richtlinie werden Berechtigungen zum Verwalten und Ausführen vorbereiteter Anweisungen für eine angegebene Konto-ID und Arbeitsgruppe erteilt.

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

Athena mit CalledVia-Kontextschlüsseln verwenden

Wenn ein Prinzipal eine Anfrage an AWS stellt, sammelt AWS die Anfrageinformationen in einem Anfragekontext, der die Anfrage auswertet und autorisiert. Sie können das `Condition`-Element einer JSON-Richtlinie verwenden, um Schlüssel im Anforderungskontext mit Schlüsselwerten zu vergleichen, die Sie in Ihrer Richtlinie angeben. Globale Bedingungskontextschlüssel sind Bedingungsschlüssel mit einem `aws:`-Präfix.

Der aws:CalledVia-Kontextschlüssel

Sie können den globalen Bedingungskontextschlüssel aws:CalledVia verwenden, um die Services in der Richtlinie mit den Services zu vergleichen, die Anforderungen im Auftrag des IAM-Prinzipals (Benutzer oder Rolle) gestellt haben. Wenn ein Prinzipal eine Anforderung an einen AWS-Service ausgibt, kann dieser Service die Anmeldeinformationen des Auftraggebers verwenden, um nachfolgende Anforderungen an andere Services auszugeben. Der Schlüssel `aws:CalledVia` enthält eine geordnete Liste aller Services in der Kette, die im Auftrag des Auftraggebers Anforderungen ausgegeben haben.

Durch Angabe eines Serviceprinzipalnamens für den Kontextschlüssel `aws:CalledVia` können Sie den Kontextschlüssel AWS-Service-spezifisch machen. Sie können beispielsweise den Bedingungsschlüssel `aws:CalledVia` verwenden, um Anfragen auf die von Athena gestellten zu beschränken. Um den Bedingungsschlüssel `aws:CalledVia` in einer Richtlinie mit Athena zu verwenden, geben Sie den Athena-Serviceprinzipalnamen `athena.amazonaws.com` wie im folgenden Beispiel an.

```
 ...
    "Condition": {
        "ForAnyValue:StringEquals": {
            "aws:CalledVia": "athena.amazonaws.com"
        }
    }
...
```

Sie können den `aws:CalledVia`-Kontextschlüssel verwenden, um sicherzustellen, dass Aufrufer nur dann Zugriff auf eine Ressource (wie eine Lambda-Funktion) haben, wenn sie die Ressource von Athena aus aufrufen.

> **Note**
>
> Der Kontextschlüssel `aws:CalledVia` ist nicht mit der Funktion zur Weitergabe vertrauenswürdiger Identitäten kompatibel.

Hinzufügen eines optionalen CalledVia-Kontextschlüssels für detaillierten Zugriff zu einer Lambda-Funktion

Athena erfordert, dass der Aufrufer über `lambda:InvokeFunction`-Berechtigungen verfügt, um die mit der Abfrage verknüpfte Lambda-Funktion aufzurufen. Die folgende Anweisung ermöglicht einen detaillierten Zugriff auf eine Lambda-Funktion, sodass der Benutzer nur Athena verwenden kann, um die Lambda-Funktion aufzurufen.

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

Das folgende Beispiel zeigt das Hinzufügen der vorherigen Anweisung zu einer Richtlinie, die es einem Benutzer ermöglicht, eine Verbundabfrage auszuführen und zu lesen. Prinzipale, die diese Aktionen ausführen dürfen, können Abfragen ausführen, die Athena-Kataloge angeben, die einer Verbunddatenquelle zugeordnet sind. Der Prinzipal kann jedoch nicht auf die zugeordnete Lambda-Funktion zugreifen, es sei denn, die Funktion wird über Athena aufgerufen.

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

Weitere Informationen über globale Bedingungskontextschlüssel `CalledVia` finden Sie unter AWS-Globale Bedingungskontextschlüssel im IAM-Benutzerhandbuch.

Zugriff auf einen Athena-Daten-Connector für externen Hive-Metastore zulassen

Die Beispiele für Berechtigungsrichtlinien in diesem Thema veranschaulichen die erforderlichen zulässigen Aktionen und die Ressourcen, für die sie zulässig sind. Untersuchen Sie diese Richtlinien sorgfältig und ändern Sie sie entsprechend Ihren Anforderungen, bevor Sie IAM-Identitäten ähnliche Berechtigungsrichtlinien anfügen.

- 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 – Erlauben Sie einem IAM-Prinzipal, Daten mit dem Athena-Daten-Connector für externen Hive-Metastore abzufragen

Die folgende Richtlinie ist den IAM-Prinzipalen zusätzlich zur Richtlinie AWS Verwaltete Richtlinie: AmazonAthenaFullAccess angefügt, die vollen Zugriff auf Athena-Aktionen gewährt.

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

Erläuterung der Berechtigungen

Erlaubte Aktionen	Erklärung
`"s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket", "s3:PutObject", "s3:ListMultipartUploadParts", "s3:AbortMultipartUpload"`	s3-Aktionen ermöglichen das Lesen und Schreiben in die Ressource, die als "arn:aws:s3::: *MyLambdaSpillBucket* /*MyLambdaSpillLocation* " angegeben ist, wobei *MyLambdaSpillLocation* den Spill-Bucket identifiziert, der in der Konfiguration der aufgerufenen Lambda-Funktion oder -Funktionen angegeben ist. Die Ressourcenkennung *arn:aws:lambda:*: MyAWSAcctId :layer:MyAthenaLambdaLayer :** ist nur erforderlich, wenn Sie eine Lambda-Ebene zur Erstellung benutzerdefinierter Laufzeitabhängigkeiten verwenden, um die

Erlaubte Aktionen	Erklärung
	Funktionsartefaktgröße zur Bereitstellungszeit zu reduzieren. Der * in der letzten Position ist ein Platzhalter für die Ebenenversion.
`"lambda:GetFunction", "lambda:GetLayerVersion", "lambda:InvokeFunction"`	Ermöglicht Abfragen zum Aufruf der im `Resource`-Block angegebenen AWS Lambda-Funktionen. Beispielsweise `arn:aws:lambda:*:` *`MyAWSAcctId`* `:function:` *`MyAthenaLambdaFunction`* , wobei *`MyAthenaLambdaFunction`* den Namen einer Lambda-Funktion angibt, die aufgerufen werden soll. Es können mehrere Funktionen angegeben werden, wie im Beispiel gezeigt.

Example – Erlauben Sie einem IAM-Prinzipal, einen Athena-Daten-Connector für externen Hive-Metastore zu erstellen

Die folgende Richtlinie ist den IAM-Prinzipalen zusätzlich zur Richtlinie AWS Verwaltete Richtlinie: AmazonAthenaFullAccess angefügt, die vollen Zugriff auf Athena-Aktionen gewährt.

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

Erläuterung der Berechtigungen

Erlaubt Abfragen zum Aufruf der AWS Lambda-Funktionen für die AWS Lambda-Funktionen, die im `Resource`-Block angegeben sind- Beispielsweise `arn:aws:lambda:*:`*`MyAWSAcctId`*`:function:`*`MyAthenaLambdaFunction`*, wobei *`MyAthenaLambdaFunction`* den Namen einer Lambda-Funktion angibt, die aufgerufen werden soll. Es können mehrere Funktionen angegeben werden, wie im Beispiel gezeigt.

Gewährung von Lambda-Funktionszugriff auf externe Hive-Metastores

Um eine Lambda-Funktion in Ihrem Konto aufzurufen, müssen Sie eine Rolle mit den folgenden Berechtigungen erstellen:

- `AWSLambdaVPCAccessExecutionRole` – Die Berechtigung AWS Lambda-Ausführungsrolle für die Verwaltung von Elastic-Network-Schnittstellen, die Ihre Funktion mit einer VPC verbinden. Stellen Sie sicher, dass es eine ausreichende Anzahl von Netzwerkschnittstellen und IP-Adressen gibt.
- `AmazonAthenaFullAccess` – Die verwaltete Richtlinie AmazonAthenaFullAccess gewährt vollen Zugriff auf Athena.
- Eine Amazon-S3-Richtlinie, die der Lambda-Funktion das Schreiben zu S3 und Athena das Lesen aus S3 ermöglicht.

Die folgende Richtlinie definiert beispielsweise die Berechtigung für den Überlaufspeicherort `s3:\\mybucket\spill`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
```

```
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::mybucket/spill"
            ]
        }
    ]
}
```

Wenn Sie IAM-Richtlinien verwenden, stellen Sie sicher, dass Sie die bewährten Methoden von IAM befolgen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Erstellen von Lambda-Funktionen

Für die Erstellung einer Lambda-Funktion in Ihrem Konto sind Funktionsentwicklungsberechtigungen oder die Rolle `AWSLambdaFullAccess` erforderlich. Weitere Informationen finden Sie unter Identitätsbasierte IAM-Richtlinien für AWS Lambda.

Da Athena das AWS Serverless Application Repository zum Erstellen von Lambda-Funktionen verwendet, sollte der Superuser oder Administrator, der Lambda-Funktionen erstellt, auch über IAM-Richtlinien verfügen, um Athena-Verbundabfragen zuzulassen.

Katalogregistrierung und Metadaten-API-Operationen

Für den Zugriff auf Katalogregistrierungs-API- und Metadaten-API-Operationen verwenden Sie die verwaltete AmazonAthenaFullAccess-Richtlinie. Wenn Sie diese Richtlinie nicht verwenden, müssen Sie Ihren Athena-Richtlinien die folgenden API-Operationen hinzufügen:

```
{
    "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": [
                "*"
            ]
        }
    ]
}
```

Regionsübergreifender Lambda-Aufruf

Um eine Lambda-Funktion in einer anderen Region als der Region aufzurufen, in der Sie Athena-Abfragen ausführen, verwenden Sie den vollständigen ARN der Lambda-Funktion. Athena ruft standardmäßig Lambda-Funktionen auf, die in derselben Region definiert sind. Wenn Sie eine Lambda-Funktion aufrufen müssen, um auf einen Hive-Metastore in einer anderen Region als der Region zuzugreifen, in der Sie Athena-Abfragen ausführen, müssen Sie den vollständigen ARN der Lambda-Funktion angeben.

Angenommen, Sie definieren den Katalog `ehms` für die Region Europa (Frankfurt) `eu-central-1`, um die folgende Lambda-Funktion in der Region USA Ost (Nord-Virginia) zu verwenden.

```
arn:aws:lambda:us-east-1:111122223333:function:external-hms-service-new
```

Wenn Sie den vollständigen ARN auf diese Weise angeben, kann Athena die Lambda-Funktion `external-hms-service-new` für `us-east-1` aufrufen, um die Hive-Metastore-Daten aus `eu-central-1` abzurufen.

> **Note**
>
> Der Katalog `ehms` sollte in derselben Region registriert sein, in der Sie Athena-Abfragen ausführen.

Kontenübergreifende Lambda-Aufrufe

Manchmal benötigen Sie möglicherweise Zugriff auf einen Hive-Metastore über ein anderes Konto. Um beispielsweise einen Hive-Metastore auszuführen, könnten Sie einen EMR-Cluster über ein anderes Konto als das Konto starten, das Sie für Athena-Abfragen verwenden. Verschiedene Gruppen oder Teams führen den Hive-Metastore möglicherweise mit unterschiedlichen Konten in

ihrer VPC aus. Vielleicht möchten Sie auch auf Metadaten in verschiedenen Hive-Metastores in verschiedenen Gruppen oder Teams zugreifen.

Athena verwendet die AWS Lambda-Unterstützung für den kontenübergreifenden Zugriff, um einen kontenübergreifenden Zugriff für Hive-Metastores zu ermöglichen.

> **Note**
>
> Beachten Sie, dass der kontenübergreifende Zugriff für Athena normalerweise den kontenübergreifenden Zugriff für Metadaten und Daten in Amazon S3 impliziert.

Stellen Sie sich das folgende Szenario vor:

- Konto `111122223333` richtet die Lambda-Funktion `external-hms-service-new` für us-east-1 in Athena ein, um auf einen Hive-Metastore zuzugreifen, der auf einem EMR-Cluster ausgeführt wird.
- Konto `111122223333` möchte dem Konto 444455556666 Zugriff auf die Hive-Metastore-Daten erteilen.

Um dem Konto `444455556666` Zugriff auf Lambda-Funktion `external-hms-service-new` zu erteilen, verwendet das Konto `111122223333` den AWS CLI-Befehl `add-permission`. Der Befehl wurde zur besseren Lesbarkeit formatiert.

```
$ 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\"}"
}
```

Um die Lambda-Berechtigung zu überprüfen, verwenden Sie den Befehl `get-policy` wie im folgenden Beispiel gezeigt. Der Befehl wurde zur besseren Lesbarkeit formatiert.

```
$ 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\"}]}"
}
```

Nach der Hinzufügung der Berechtigung können Sie einen vollständigen ARN der Lambda-Funktion für `us-east-1` wie den folgenden verwenden, wenn Sie den Katalog `ehms` definieren:

```
arn:aws:lambda:us-east-1:111122223333:function:external-hms-service-new
```

Weitere Informationen zu regionsübergreifenden Aufrufen finden Sie unter Regionsübergreifender Lambda-Aufruf weiter oben in diesem Thema.

Gewährung eines kontenübergreifendem Zugriffs auf Daten

Bevor Sie Athena-Abfragen ausführen können, müssen Sie einen kontenübergreifenden Zugriff auf die Daten in Amazon S3 gewähren. Sie können dafür eine der folgenden Möglichkeiten auswählen:

- Aktualisieren Sie die Richtlinie für Zugriffskontrolllisten des Amazon-S3-Buckets mit einer kanonischen Benutzer-ID.
- Fügen Sie der Amazon-S3-Bucket-Richtlinie den kontenübergreifenden Zugriff hinzu.

Sie könnten beispielsweise der Amazon-S3-Bucket-Richtlinie im Konto die folgende Richtlinie `111122223333` die folgende Richtlinie hinzufügen, damit das Konto `444455556666` Daten aus dem angegebenen Amazon-S3-Speicherort lesen kann.

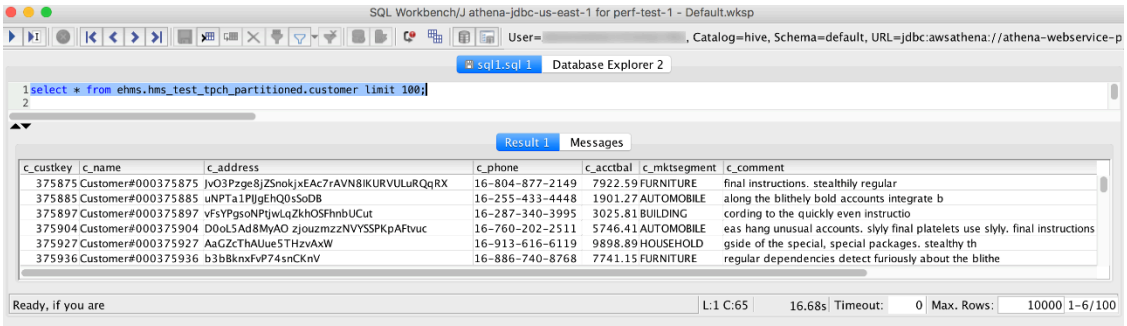
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1234567890123",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::444455556666:user/perf1-test"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::athena-test/lambda/dataset/*"
        }
    ]
}
```

Note

Möglicherweise müssen Sie den kontenübergreifenden Zugriff auf Amazon S3 nicht nur in Bezug auf Ihre Daten, sondern auch in Bezug auf Ihren Amazon-S3-Überlaufspeicherort gewähren. Ihre Lambda-Funktion speichert zusätzliche Daten am Überlaufspeicherort, wenn die Größe des Antwortobjekts einen bestimmten Schwellenwert überschreitet. Ein Beispiel für eine Richtlinie finden Sie am Anfang dieses Themas.

Im aktuellen Beispiel kann nach Gewährung des kontenübergreifenden Zugriffs für `444455556666`, das Konto `444455556666` den Katalog `ehms` im eigenen `account` verwenden, um im Konto `111122223333` definierte Tabellen abzufragen.

Im folgenden Beispiel ist das SQL Workbench-Profil `perf-test-1` für das Konto `444455556666` angegeben. Die Abfrage verwendet den Katalog `ehms`, um auf den Hive-Metastore und die Amazon-S3-Daten im Konto `111122223333` zuzugreifen.



Beispiel für IAM-Berechtigungsrichtlinien zum Zulassen von Athena Federated Query

Die Beispiele für Berechtigungsrichtlinien in diesem Thema veranschaulichen die erforderlichen zulässigen Aktionen und die Ressourcen, für die sie zulässig sind. Untersuchen Sie diese Richtlinien sorgfältig und ändern Sie sie entsprechend Ihren Anforderungen, bevor Sie diese an IAM-Identitäten anfügen.

Weitere Informationen zum Anfügen von Richtlinien an IAM-Identitäten finden Sie unter Hinzufügen und Entfernen von IAM-Identitätsberechtigungen im IAM-Benutzerhandbuch.

- 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 – Ausführen und Zurückgeben von Ergebnissen durch einen IAM-Prinzipal mithilfe von Athena Federated Query zulassen

Die folgende identitätsbasierte Berechtigungsrichtlinie ermöglicht Aktionen, die ein Benutzer oder ein anderer IAM-Prinzipal erfordert, um Athena Federated Query zu verwenden. Prinzipale, die diese Aktionen ausführen dürfen, können Abfragen ausführen, die Athena-Kataloge angeben, die einer Verbunddatenquelle zugeordnet sind.

```
{
    "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/*"
            ]
        }
    ]
}
```

Erläuterung der Berechtigungen

Erlaubte Aktionen	Erklärung
`"athena:GetQueryExecution", "athena:GetQueryResults", "athena:GetWorkGroup", "athena:StartQueryExecution",`	Athena-Berechtigungen, die zum Ausführen von Verbundabfragen erforderlich sind.

Erlaubte Aktionen	Erklärung
`"athena:StopQueryExecution"`	

`"athena:GetDataCatalog",`
`"athena:GetQueryExecution,"`
`"athena:GetQueryResults",`
`"athena:GetWorkGroup",`
`"athena:StartQueryExecution",`
`"athena:StopQueryExecution"`
	Athena-Berechtigungen, die zum Ausführen von Verbundansichtsabfragen erforderlich sind. Die `GetDataCatalog` Aktion ist für Ansichten erforderlich.

`"lambda:InvokeFunction"`	Ermöglicht Abfragen, die AWS Lambda Funktionen für die im `Resource` Block angegebenen AWS Lambda Funktionen aufzurufen. Beispiel: where `arn:aws:lambda:*:` *`MyAWSAcctId`* `:function:` *`MyAthenaLambdaFunction`* *`MyAthenaLambdaFunction`* gibt den Namen einer Lambda-Funktion an, die aufgerufen werden soll. Wie im Beispiel gezeigt, können mehrere Funktionen angegeben werden.

Erlaubte Aktionen	Erklärung
`"s3:AbortMultipartUpload", "s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket", "s3:ListMultipartUploadParts", "s3:PutObject"`	Die `s3:GetBucketLocation` Berechtigungen `s3:ListBucket` und sind für den Zugriff auf den Abfrageausgabe-Bucket für ausgeführte IAM-Prinzipale erforderlich. `StartQueryExecution` `s3:PutObject` `s3:ListMultipartUploadParts`, und `s3:AbortMultipartUpload` ermöglichen das Schreiben von Abfrageergebnissen in alle Unterordner des Abfrageergebnis-Buckets, wie durch die `arn:aws:s3:::` *`MyQueryResultsBucket`* `/*` Ressourcen-ID angegeben, wobei sich der Athena-Abfrageergebnis-Bucket *`MyQueryResultsBucket`* befindet. Weitere Informationen finden Sie unter Arbeiten mit Abfrageergebnissen, letzten Abfragen und Ausgabedateien. `s3:GetObject` ermöglicht das Lesen von Abfrageergebnissen und dem Abfrageverlauf für die angegebene Ressource`arn:aws:s3:::` *`MyQueryResultsBucket`*, wo sich der Athena-Abfrageergebnis-Bucket *`MyQueryResultsBucket`* befindet. `s3:GetObject` ermöglicht auch das Lesen von der Ressource, die als angegeben *`MyLambdaSpillPrefix`* ist"`arn:aws:s3:::` *`MyLambdaSpillBucket`* `/` *`MyLambdaSpillPrefix`* `*`", wo dies in der Konfiguration der aufgerufenen Lambda-Funktion oder der aufgerufenen Funktionen angegeben ist.

Example – Erlauben Sie einem IAM-Prinzipal, einen Datenquellen-Connector zu erstellen

```
{
    "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/*"
        }
    ]
}
```

Erläuterung der Berechtigungen

Erlaubte Aktionen	Erklärung
`"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",`	Erlauben der Erstellung und Verwaltung von Lambda-Funktionen, die als Ressourcen aufgeführt sind. In diesem Beispiel wird ein Namenspräfix in der Ressourcen-ID verwendet`arn:aws:lambda:*:` *MyAWSAcctId* `:function:` *MyAthenaLambdaFunctionsPrefix* `*`, wobei *MyAthenaLambdaFunctionsPrefix* ein gemeinsames Präfix im Namen einer Gruppe von Lambda-Funktionen verwendet wird, sodass sie nicht einzeln als Ressourcen angegeben werden müssen. Sie können eine oder mehrere Lambda-Funktionsressourcen angeben.
`"s3:GetObject"`	Ermöglicht das Lesen eines Buckets, der die in der Ressourcen-ID `arn:aws:s3:::awsserverlessrepo-changesets-`*1iiv3xa62ln3m* `/*` angegebenen Anforderungen AWS Serverless Application Repository benötigt. Dieser Bucket kann spezifisch für Ihr Konto sein.

Erlaubte Aktionen	Erklärung
`"cloudformation:*"`	Ermöglicht die Erstellung und Verwaltung von AWS CloudFormation Stacks, die durch die Ressource *StackPrefixmyCF* spezifizi ert werden. Mit diesen Stacks und Stacksets werden Konnektoren und UDFs bereitgestellt. AWS Serverless Application Repository
`"serverlessrepo:*"`	Ermöglicht das Suchen, Anzeigen, Veröffent lichen und Aktualisieren von Anwendungen in der AWS Serverless Application Repositor y, die durch die Ressourcen-ID angegeben sind. `arn:aws:serverlessrepo:*:*: applications/*`

Beispiel für IAM-Berechtigungsrichtlinien zum Zulassen von benutzerdefinierten Amazon-Athena-Funktionen (UDF)

Die Beispiele für Berechtigungsrichtlinien in diesem Thema veranschaulichen die erforderlichen zulässigen Aktionen und die Ressourcen, für die sie zulässig sind. Untersuchen Sie diese Richtlinien sorgfältig und ändern Sie sie entsprechend Ihren Anforderungen, bevor Sie IAM-Identitäten ähnliche Berechtigungsrichtlinien anfügen.

- 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 – Erlauben Sie einem IAM-Prinzipal, Abfragen auszuführen und zurückzugeben, die eine Athena-UDF-Anweisung enthalten

Die folgende identitätsbasierte Berechtigungsrichtlinie erlaubt Aktionen, die ein Benutzer oder ein anderer IAM-Prinzipal benötigt, um Abfragen auszuführen, die Athena-UDF-Anweisungen verwenden.

```
{
    "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": "*"
        }
    ]
}
```

Erläuterung der Berechtigungen

Erlaubte Aktionen	Erklärung
`"athena:StartQueryExecution", "athena:GetQueryResults", "athena:GetWorkGroup", "athena:StopQueryExecution", "athena:GetQueryExecution",`	Athena-Berechtigungen, die zum Ausführen von Abfragen in der `MyAthenaWorkGroup` - Arbeitsgruppe erforderlich sind.

Erlaubte Aktionen	Erklärung
`"s3:PutObject", "s3:GetObject", "s3:AbortMultipartUpload"`	`s3:PutObject` und `s3:AbortMultipartUpload` erlauben das Schreiben von Abfrageergebnissen in alle Unterordner des Abfrageergebnis-Buckets, wie durch den `arn:aws:s3:::` *MyQueryResultsBucket* `/*`-Ressourcenbezeichner angegeben, wobei *MyQueryResultsBucket* der Athena-Abfrageergebnis-Bucket ist. Weitere Informationen finden Sie unter Arbeiten mit Abfrageergebnissen, letzten Abfragen und Ausgabedateien. `s3:GetObject` erlaubt das Lesen der Abfrageergebnisse und des Abfrageverlaufs für die als `arn:aws:s3:::` *MyQueryResultsBucket* angegebene Ressource, wobei *MyQueryResultsBucket* der Athena-Abfrageergebnis-Bucket ist. Weitere Informationen finden Sie unter Arbeiten mit Abfrageergebnissen, letzten Abfragen und Ausgabedateien. `s3:GetObject` erlaubt auch das Lesen von der als `"arn:aws:s3:::` *MyLambdaSpillBucket* `/`*MyLambdaSpillPrefix* `*"` angegebenen Ressource, wobei *MyLambdaSpillPrefix* in der Konfiguration der aufgerufenen Lambda-Funktion oder -Funktionen angegeben wird.

Erlaubte Aktionen	Erklärung
`"lambda:InvokeFunction"`	Ermöglicht Abfragen zum Aufruf der im `Resource`-Block angegebenen AWS Lambda-Funktionen. Beispielsweise `arn:aws:lambda:*:` *`MyAWSAcctId`* `:function:` *`MyAthenaLambdaFunction`* , wobei *`MyAthenaLambdaFunction`* den Namen einer Lambda-Funktion angibt, die aufgerufen werden soll. Es können mehrere Funktionen angegeben werden, wie im Beispiel gezeigt.

Example – Erlauben Sie einem IAM-Prinzipal, eine Athena-UDF zu erstellen

```
{
    "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/*"
        }
    ]
}
```

Erläuterung der Berechtigungen

Erlaubte Aktionen	Erklärung
`"lambda:CreateFunction", "lambda:ListVersionsByFunction", "lambda:GetFunctionConfiguration", "lambda:PutFunctionConcurrency", "lambda:ListTags", "lambda:DeleteFunction", "lambda:GetAlias", "lambda:InvokeFunction", "lambda:GetFunction", "lambda:ListAliases", "lambda:UpdateFunctionConfigur ation", "lambda:UpdateFunctionCode", "lambda:AddPermission", "lambda:DeleteFunctionConcurrency",`	Erlauben der Erstellung und Verwaltung von Lambda-Funktionen, die als Ressourcen aufgeführt sind. In diesem Beispiel wird ein Namenspräfix in der Ressourcenkennung `arn:aws:lambda:*:` *MyAWSAcct Id* `:function:` *MyAthenaLambdaFunc tionsPrefix* * verwendet, wobei *MyAthenaLambdaFunctionsPrefix* ein gemeinsames Präfix ist, das im Namen einer Gruppe von Lambda-Funktionen verwendet wird, sodass diese nicht einzeln als Ressource n angegeben werden müssen. Sie können eine

Erlaubte Aktionen	Erklärung
`"lambda:RemovePermission", "lambda:GetPolicy" "lambda:GetAccountSettings", "lambda:ListFunctions", "lambda:ListEventSourceMappings",`	oder mehrere Lambda-Funktionsressourcen angeben.
`"s3:GetObject"`	Erlaubt das Lesen eines Buckets, der gemäß der Ressourcenbezeichnung `arn:aws:s3:::awsserverlessrepo-changesets-` *`1iiv3xa62ln3m`* `/*` für AWS Serverless Application Repository erforderlich ist.
`"cloudformation:*"`	Erlaubt die Erstellung und Verwaltung von AWS CloudFormation-Stacks, die von der Ressource *`MyCFStackPrefix`* angegeben werden. Diese Stacks und Stacksets stellen die Art und Weise dar, wie AWS Serverless Application Repository-Connectors und UDFs bereitstellt.
`"serverlessrepo:*"`	Erlaubt das Suchen, Anzeigen, Veröffentlichen und Aktualisieren von Anwendungen in AWS Serverless Application Repository, die durch den Ressourcenbezeichner `arn:aws:serverlessrepo:*:*:applications/*` angegeben sind.

Erlauben des Zugriffs für ML mit Athena

IAM-Prinzipale, die Athena ML-Abfragen ausführen, müssen berechtigt sein, die `sagemaker:invokeEndpoint`-Aktion für Sagemaker-Endpunkte auszuführen, die sie verwenden. Fügen Sie eine Richtlinienanweisung ähnlich der folgenden in identitätsbasierte Berechtigungsrichtlinien ein, die an Benutzeridentitäten angefügt sind. Fügen Sie außerdem die AWS Verwaltete Richtlinie: AmazonAthenaFullAccess an, die vollen Zugriff auf Athena-Aktionen gewährt, oder eine modifizierte Inline-Richtlinie, die eine Teilmenge von Aktionen ermöglicht.

Ersetzen Sie `arn:aws:sagemaker:region:AWSAcctID:ModelEndpoint` im Beispiel durch den ARN oder die ARNs von Modellendpunkten, die in Abfragen verwendet werden sollen. Weitere Informationen finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für SageMaker in der Service-Autorisierungs-Referenz.

```
{
            "Effect": "Allow",
            "Action": [
                "sagemaker:invokeEndpoint"
            ],
            "Resource": "arn:aws:sagemaker:us-west-2:123456789012:workteam/public-
crowd/default"
}
```

Wenn Sie IAM-Richtlinien verwenden, stellen Sie sicher, dass Sie die bewährten Methoden von IAM befolgen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Aktivieren des föderierten Zugriffs auf die Athena-API

In diesem Abschnitt wird der föderierte Zugriff beschrieben, der Benutzern oder Client-Anwendungen in Ihrer Organisation den Aufruf von Amazon-Athena-API-Operationen ermöglicht. In diesem Fall verfügen die Benutzer Ihrer Organisation nicht über direkten Zugriff auf Athena. Stattdessen verwalten Sie Benutzeranmeldeinformationen außerhalb von AWS in Microsoft Active Directory. Active Directory unterstützt SAML 2.0 (Security Assertion Markup Language 2.0).

In diesem Fall verwenden Sie zur Authentifizierung von Benutzern den JDBC- oder ODBC-Treiber mit SAML 2.0-Unterstützung, der auf die Active Directory Federation Services (AD FS) 3.0 zugreift und Client-Anwendungen den Zugriff auf Athena-API-Operationen ermöglicht.

Weitere Informationen zur SAML 2.0-Unterstützung auf AWS finden Sie unter Informationen zur SAML 2.0-Föderierung im IAM-Benutzerhandbuch.

> **Note**
>
> Der föderierte Zugriff auf die Athena-API wird für eine bestimmte Art von Identitätsanbieter (IdP) unterstützt, nämlich den Active Directory Federation Service (AD FS 3.0), der Teil von Windows Server ist. Der Verbundzugriff ist nicht mit der Funktion zur Weitergabe vertrauenswürdiger Identitäten des IAM Identity Center kompatibel. Der Zugriff wird durch die

Versionen von JDBC- bzw. ODBC-Treibern hergestellt, die SAML 2.0 unterstützen. Weitere Informationen finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC und Herstellen einer Verbindung mit Amazon Athena mit ODBC.

Themen

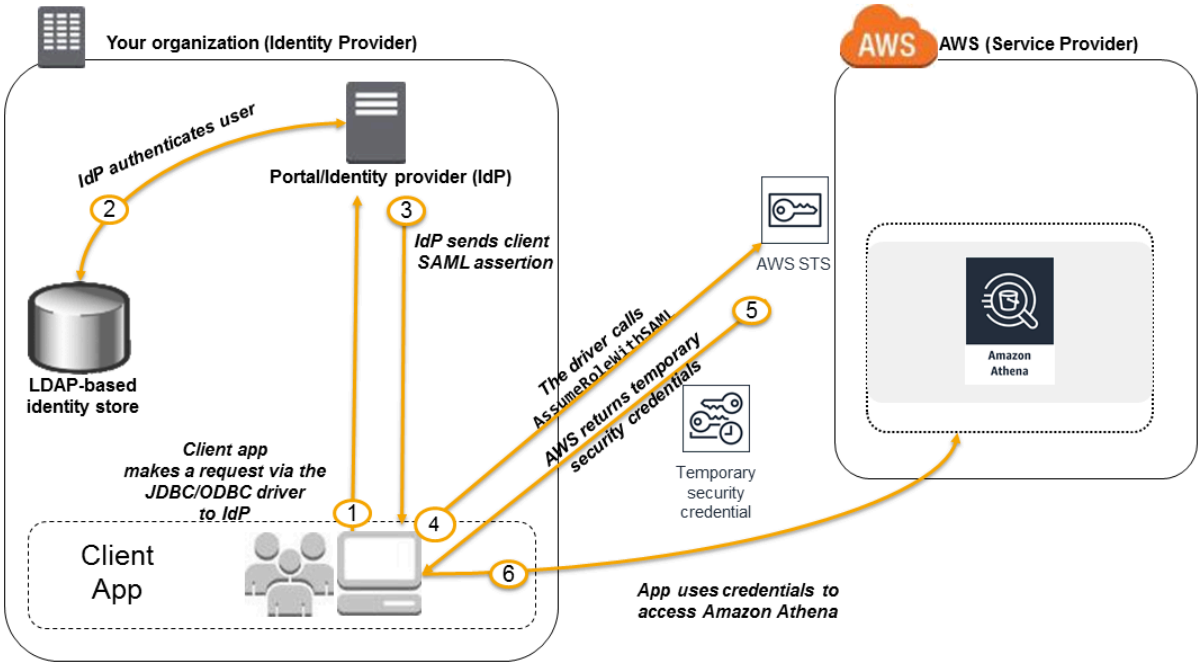
Bevor Sie beginnen

Stellen Sie vor Beginn sicher, dass die folgenden Voraussetzungen erfüllt sind:

- Installieren und konfigurieren Sie AD FS 3.0 innerhalb Ihrer Organisation als IdP.
- Installieren und konfigurieren Sie die neuesten verfügbaren Versionen der JDBC- oder ODBC-Treiber auf Clients, die für den Zugriff auf Athena verwendet werden. Der Treiber muss mit dem föderierten Zugriff mit SAML 2.0 kompatible Unterstützung bieten. Weitere Informationen finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC und Herstellen einer Verbindung mit Amazon Athena mit ODBC.

Architekturdiagramm

Das folgende Diagramm veranschaulicht diesen Prozess.



1. Ein Benutzer in Ihrer Organisation verwendet eine Client-Anwendung mit dem JDBC- oder ODBC-Treiber, um eine Authentifizierung vom Identitätsanbieter Ihrer Organisation anzufordern. Der Identitätsanbieter ist AD FS 3.0.
2. Der Identitätsanbieter authentifiziert den Benutzer anhand von Active Directory, dem Identitätsspeicher Ihrer Organisation.
3. Der Identitätsanbieter konstruiert eine SAML-Zusicherung mit Informationen über den Benutzer und sendet die Zusicherung über den JDBC- oder ODBC-Treiber an die Client-Anwendung.
4. Der JDBC- oder ODBC-Treiber ruft die AWS Security Token Service-API-Operation AssumeRoleWithSAML auf und übergibt ihr dabei die folgenden Parameter:
   - Der ARN des SAML-Anbieters
   - Den ARN der zu übernehmenden Rolle
   - Die SAML-Zusicherung des Identitätsanbieters

   Weitere Informationen finden Sie unter AssumeRoleWithSAML in der AWS Security Token Service-API-Referenz.
5. Die API-Antwort an die Client-Anwendung über den JDBC- oder ODBC-Treiber umfasst temporäre Sicherheitsanmeldeinformationen.

6. Die Client-Anwendung verwendet die temporären Sicherheitsanmeldeinformationen zum Aufrufen von Athena-API-Operationen, so dass Ihre Benutzer auf Athena-API-Operationen zugreifen können.

Verfahren: SAML-basierter föderierter Zugriff auf die Athena-API

Das Verfahren umfasst Schritte, die eine Vertrauensbeziehung zwischen dem Identitätsanbieter Ihrer Organisation und Ihrem AWS-Konto schaffen, um den SAML-basierten föderierten Zugriff auf die Amazon-Athena-API-Operation zu gewährleisten.

Den föderierten Zugriff auf die Athena-API aktivieren Sie wie folgt:

1. Registrieren Sie AWS in Ihrer Organisation als Dienstanbieter in Ihrem Identitätsanbieter. Dieser Prozess wird als Vertrauensstellung der vertrauenden Seite bezeichnet. Weitere Informationen finden Sie unter Konfigurieren des SAML 2.0-Identitätsanbieters mit der Vertrauensstellung der vertrauenden Seite im IAM-Benutzerhandbuch. Führen Sie als Teil dieser Aufgabe die folgenden Schritte aus:

   a. Rufen Sie das Beispiel-SAML-Metadatendokument von dieser URL ab: https://signin.aws.amazon.com/static/saml-metadata.xml.

   b. Erstellen Sie im Identitätsanbieter Ihrer Organisation (AD FS) eine entsprechende Metadaten-XML-Datei, die Ihren Identitätsanbieter als Identitätsanbieter für AWS beschreibt. Die Metadatendatei muss den Namen des Ausstellers, ein Erstellungsdatum, ein Ablaufdatum und Schlüssel enthalten, die AWS verwendet, um Authentifizierungsantworten (Zusicherungen) von Ihrer Organisation zu überprüfen.

2. Erstellen Sie in der IAM-Konsole eine SAML-Identitätsanbieter-Entität. Weitere Informationen finden Sie unter Erstellen von SAML-Identitätsanbietern im IAM-Benutzerhandbuch. Tun Sie im Rahmen dieses Schritts Folgendes:

   a. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.

   b. Laden Sie das SAML-Metadatendokument hoch, das der Identitätsanbieter (AD FS) im Rahmen von Schritt 1 dieses Verfahrens erstellt hat.

3. Erstellen Sie in der IAM-Konsole mindestens eine IAM-Rolle für Ihren Identitätsanbieter. Weitere Informationen finden Sie unter Erstellen von Rollen für externe Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch. Tun Sie im Rahmen dieses Schritts Folgendes:

- Führen Sie in der Berechtigungsrichtlinie der Rolle Aktionen auf, die Benutzer Ihrer Organisation in AWS ausführen dürfen.
- Legen Sie in der Vertrauensrichtlinie der Rolle die SAML-Anbieter-Entity als Prinzipal fest, die Sie in Schritt 2 dieses Verfahrens erstellt haben.

Dies baut eine Vertrauensstellung zwischen Ihrer Organisation und AWS auf.

4. Definieren Sie im Identitätsanbieter Ihrer Organisation (AD FS) Zusicherungen, in denen Benutzer oder Gruppen in Ihrer Organisation den IAM-Rollen zugeordnet werden. Das Mapping von Benutzern und Gruppen zu den IAM-Rollen wird auch als Anspruchsregel bezeichnet. Beachten Sie, dass unterschiedliche Benutzer und Gruppen in Ihrer Organisation unterschiedlichen IAM-Rollen zugeordnet werden können.

   Weitere Informationen zum Konfigurieren des Mappings in ADFS finden Sie im Blog-Beitrag Den Verbund zu AWS mithilfe von Windows Active Directory, ADFS, and SAML 2.0 ermöglichen.

5. Installieren und konfigurieren Sie den JDBC- oder ODBC-Treiber mit SAML 2.0-Unterstützung. Weitere Informationen finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC und Herstellen einer Verbindung mit Amazon Athena mit ODBC.

6. Geben Sie die Verbindungszeichenfolge aus Ihrer Anwendung im JDBC- bzw. ODBC-Treiber an. Weitere Informationen zu der Verbindungszeichenfolge, die Ihre Anwendung verwenden sollte, finden Sie unter dem Thema „Verwendung des Anmeldeinformationsanbietersvon Active Directory Federation Services (ADFS)“ im Installations- and Konfigurationshandbuch für JDBC-Treiber bzw. einem ähnlichen Thema im Installations- and Konfigurationshandbuch für ODBC-Treiber, die als PDF-Downloads über die Themen Herstellen einer Verbindung zu Amazon Athena mit JDBC und Herstellen einer Verbindung mit Amazon Athena mit ODBC verfügbar sind.

   Die allgemeine Zusammenfassung der Konfiguration der Verbindungszeichenfolge für die Treiber ist wie folgt:

   1. Stellen Sie in `AwsCredentialsProviderClass configuration` den `com.simba.athena.iamsupport.plugin.AdfsCredentialsProvider` ein, um anzugeben, dass Sie die SAML 2.0-basierte Authentifizierung über den Identitätsanbieter AD FS nutzen möchten.
   2. Geben Sie für `idp_host` den Hostnamen des AD FS-Identitätsanbieterservers an.
   3. Geben Sie für `idp_port` die Nummer des Ports an, auf dem der AD FS-Identitätsanbieter auf die SAML-Zusicherungsanfrage hört.

4. Geben Sie für `UID` und `PWD` die AD-Domain-Benutzeranmeldeinformationen an. Wenn Sie den Treiber auf Windows verwenden und `UID` und `PWD` nicht angegeben sind, versucht der Treiber, die Anmeldeinformationen des Benutzers abzurufen, der an dem Windows-Computer angemeldet ist.
5. Optional können Sie für `ssl_insecure` den Wert `true` festlegen. In diesem Fall überprüft der Treiber nicht die Authentizität des SSL-Zertifikats für den ADFS-Identitätsanbieterserver. Die Einstellung auf `true` ist erforderlich, wenn das SSL-Zertifikat des Identitätsanbieters ADFS nicht vom Treiber als vertrauenswürdig konfiguriert wurde.
6. Zum Aktivieren des Mappings eines Active Directory-Domain-Benutzers bzw. einer Gruppe zu mindestens einer IAM-Rolle (wie in Schritt 4 dieses Verfahrens erwähnt) geben Sie in der `preferred_role` für die JDBC- bzw. ODBC-Verbindung die IAM-Rolle (ARN) an, die für die Treiberverbindung übernommen werden soll. Die Angabe der `preferred_role` ist optional, aber nützlich, wenn die Rolle nicht die erste in der Anspruchsregel aufgeführte Rolle ist.

Aufgrund dieses Verfahrens werden die folgenden Aktionen ausgeführt:

1. Der JDBC- bzw. ODBC-Treiber ruft die AWS STS-API AssumeRoleWithSAML auf und übergibt ihr die Zusicherungen, wie in Schritt 4 des Architekturdiagramms abgebildet.
2. AWS stellt sicher, dass die Anforderung zum Übernehmen der Rolle vom Identitätsanbieter stammt, auf den in der SAML-Anbieter-Entity verwiesen wird.
3. Wenn die Anfrage erfolgreich ist, gibt die AWS STS-API-Operation AssumeRoleWithSAML eine Reihe temporärer Sicherheitsanmeldeinformationen zurück, die Ihre Client-Anwendung verwendet, um signierte Anforderungen an Athena zu stellen.

   Ihre Anwendung verfügt nun über Informationen zum aktuellen Benutzer und kann programmgesteuert auf Athena zugreifen.

Protokollierung und Überwachung in Athena

Um Vorfälle zu erkennen, Benachrichtigungen zu Vorfällen zu erhalten und um darauf zu reagieren, verwenden Sie diese Optionen mit Amazon Athena:

- Athena überwachen mit AWS CloudTrail – AWS CloudTrail bietet eine Aufzeichnung von Aktionen eines Benutzers, einer Rolle oder eines AWS-Service in Athena. Es erfasst Aufrufe über die Athena-Konsole und Code-Aufrufe der Athena-API-Vorgänge als Ereignisse. Auf diese Weise können Sie feststellen, welche Anfrage an Athena gestellt wurde, die IP-Adresse, von der die

Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde und weitere Details. Weitere Informationen finden Sie unter Protokollieren von Amazon-Athena-API-Aufrufen mit AWS CloudTrail.

Sie können Athena auch verwenden, um die CloudTrail-Protokolldateien nicht nur für Athena, sondern auch für andere AWS-Services abzufragen. Weitere Informationen finden Sie unter Abfragen von AWS CloudTrail-Protokollen.

- Überwachen Sie die Athena-Nutzung mit CloudTrail und Amazon QuickSight – Amazon QuickSight ist ein vollständig verwalteter, cloudbasierter Business-Intelligence-Service, mit dem Sie interaktive Dashboards erstellen können, auf die Ihr Unternehmen von jedem Gerät aus zugreifen kann. Ein Beispiel für eine Lösung, die CloudTrail und Amazon QuickSight verwendet, um die Athena-Nutzung zu überwachen, finden Sie im AWS-Big-Data-Blogbeitrag Wie Realtor.com die Amazon-Athena-Nutzung mit AWS CloudTrail und Amazon QuickSight überwacht.
- Amazon EventBridge mit Athena verwenden – Amazon EventBridge liefert nahezu in Echtzeit einen Strom von Systemereignissen, die Änderungen in AWS-Ressourcen beschreiben. EventBridge erkennt betriebliche Änderungen, sobald sie auftreten, reagiert darauf und ergreift bei Bedarf Korrekturmaßnahmen, indem es Nachrichten sendet, um an die Umgebung zu reagieren, Funktionen zu aktivieren, Änderungen vorzunehmen und Zustandsinformationen zu erfassen. Ereignisse werden auf bestmögliche Weise ausgegeben. Weitere Informationen finden Sie unter Erste Schritte mit Amazon EventBridge im Amazon EventBridge Benutzerhandbuch.
- Verwenden Sie Arbeitsgruppen zum Trennen von Benutzern, Teams, Anwendungen oder Arbeitslasten sowie zum Einrichten von Abfragegrenzwerten und zur Begrenzung von Abfragekosten – Sie können abfragebezogene Metriken in Amazon CloudWatch anzeigen, Abfragekosten durch die Konfiguration von Grenzwerten für die Menge der gescannten Daten steuern, Schwellwerte einrichten sowie Aktionen wie etwa Amazon-SNS-Alarme auslösen, wenn diese Grenzwerte überschritten werden. Ein fortgeschrittenes Verfahren finden Sie unter Einrichten von Arbeitsgruppen. Verwenden Sie IAM-Berechtigungen auf Ressourcenebene für die Steuerung des Zugriffs auf eine bestimmte Arbeitsgruppe. Weitere Informationen finden Sie unter Verwenden von Arbeitsgruppen zum Ausführen von Abfragen und Steuern von Kosten und Überwachen von Abfragen mit CloudWatch Metriken und Ereignissen.

Themen

- Protokollieren von Amazon-Athena-API-Aufrufen mit AWS CloudTrail

Protokollieren von Amazon-Athena-API-Aufrufen mit AWS CloudTrail

Athena ist in AWS CloudTrail integriert, einem Service, der eine Aufzeichnung der von einem Benutzer, einer Rolle oder einem AWS-Service durchgeführten Aktionen in Athena bereitstellt.

CloudTrail erfasst alle API-Aufrufe für Athena als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der Athena-Konsole und Code-Aufrufe der Athena-API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail-Ereignissen an einen Amazon S3-Bucket, einschließlich Ereignisse für Athena aktivieren. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse in der CloudTrail-Konsole trotzdem in Event history (Ereignisverlauf) anzeigen.

Mit den von CloudTrail erfassten Informationen können Sie die an Athena gestellte Anfrage, die IP-Adresse, von der die Anfrage gestellt wurde, den Initiator der Anfrage, den Zeitpunkt der Anfrage und zusätzliche Details bestimmen.

Weitere Informationen zu CloudTrail finden Sie im AWS CloudTrail-Benutzerhandbuch.

Sie können Athena verwenden, um CloudTrail-Protokolldateien von Athena selbst und von anderen AWS-Services abzufragen. Weitere Informationen finden Sie unter Abfragen von AWS CloudTrail-Protokollen, Hive JSON SerDe und im AWS-Big-Data-Blogbeitrag Verwenden Sie CTAS-Anweisungen mit Amazon Athena, um Kosten zu senken und die Leistung zu verbessern, die CloudTrail verwendet, um Erkenntnis für die Athena-Nutzung zu erhalten.

Athena-Informationen in CloudTrail

CloudTrail wird beim Erstellen Ihres Amazon-Web-Services-Kontos für Sie aktiviert. Die in Athena auftretenden Aktivitäten werden als CloudTrail-Ereignis zusammen mit anderen AWS-Serviceereignissen im Ereignisverlauf aufgezeichnet. Sie können die neusten Ereignisse in Ihrem Amazon-Web-Services-Konto anzeigen, suchen und herunterladen. Weitere Informationen finden Sie unter Anzeigen von Ereignissen mit dem CloudTrail-API-Ereignisverlauf.

Zur kontinuierlichen Aufzeichnung von Ereignissen in Ihrem Amazon-Web-Services-Konto, einschließlich Ereignissen für Athena, erstellen Sie einen Trail. Ein Trail ermöglicht es CloudTrail, Protokolldateien in einem Amazon-S3-Bucket bereitzustellen. Wenn Sie einen Trail in der Konsole anlegen, gilt dieser für alle AWS-Regionen-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen in der AWS-Partition und stellt die Protokolldateien in dem von Ihnen angegebenen Amazon S3 Bucket bereit. Darüber hinaus können Sie andere AWS-Services konfigurieren, um die in den CloudTrail-Protokollen erfassten Ereignisdaten weiter zu analysieren und entsprechend zu agieren. Weitere Informationen finden Sie hier:

- Übersicht zum Erstellen eines Trails
- In CloudTrail unterstützte Services und Integrationen
- Konfigurieren von Amazon SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail-Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail-Protokolldateien von mehreren Konten.

Alle Athena-Aktionen werden von CloudTrail protokolliert und sind in der Amazon-Athena-API-Referenz dokumentiert. Zum Beispiel generieren Aufrufe der Aktionen StartQueryExecution und GetQueryResults Einträge in den CloudTrail-Protokolldateien.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Anhand der Identitätsinformationen zur Benutzeridentität können Sie Folgendes bestimmen:

- Ob die Anfrage mit Stammbenutzer- oder AWS Identity and Access Management (IAM)-Anmeldeinformationen ausgeführt wurde.
- Ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer ausgeführt wurde.
- Ob die Anforderung aus einem anderen AWS-Service gesendet wurde.

Weitere Informationen finden Sie unter CloudTrail-Element userIdentity.

Grundlagen zu Athena-Protokolldateieinträgen

Ein Trail ist eine Konfiguration, durch die Ereignisse als Protokolldateien an den von Ihnen angegebenen Amazon-S3-Bucket übermittelt werden. CloudTrail-Protokolldateien können einen oder mehrere Einträge enthalten. Ein Ereignis stellt eine einzelne Anfrage aus einer beliebigen Quelle dar und enthält unter anderem Informationen über die angeforderte Aktion, das Datum und die Uhrzeit der Aktion sowie über die Anfrageparameter. CloudTrail-Protokolleinträge sind kein geordnetes Stack-Trace der öffentlichen API-Aufrufe und erscheinen daher in keiner bestimmten Reihenfolge.

> **Note**
>
> Um die unbeabsichtigte Offenlegung vertraulicher Informationen zu verhindern, hat der `queryString`-Eintrag sowohl in den `StartQueryExecution`-Protokollen als auch in den `CreateNamedQuery`-Protokollen den Wert `***OMITTED***`.

Dies ist beabsichtigt. Um auf die eigentliche Abfragezeichenfolge zuzugreifen, können Sie die Athena-API GetQueryExecution verwenden und den Wert von `responseElements.queryExecutionId` aus dem CloudTrail-Protokoll übergeben.

In den folgenden Beispielen finden Sie CloudTrail-Protokolleinträge für:

- StartQueryExecution (erfolgreich)
- StartQueryExecution (fehlgeschlagen)
- CreateNamedQuery

StartQueryExecution (erfolgreich)


```
{
 "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://athena-johndoe-test/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 (fehlgeschlagen)

```
{
 "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"
},
```


Compliance-Validierung für Amazon Athena

Externe Prüfer bewerten im Rahmen verschiedener AWS-Compliance-Programme die Sicherheit und Compliance von Amazon Athena. Hierzu zählen unter anderem SOC, PCI, FedRAMP und andere.

Eine Liste der AWS-Services, die in den Geltungsbereich bestimmter Compliance-Programme fallen, finden Sie auf der Seite AWS-Services in Scope nach Compliance-Programm. Allgemeine Informationen finden Sie unter AWS-Compliance-Programme.

Die Auditberichte von Drittanbietern lassen sich mit herunterlade AWS Artifact. Weitere Informationen finden Sie unter Herunterladen von Berichten in AWS Artifact.

Ihre Compliance-Verantwortung bei der Verwendung von Athena wird durch die Sensibilität Ihrer Daten, die Compliance-Ziele Ihres Unternehmens und die geltenden Gesetze und Vorschriften bestimmt. AWS stellt die folgenden Ressourcen zur Unterstützung der Compliance bereit:

- Kurzanleitungen für Sicherheit und Compliance – In diesen Bereitstellungsleitfäden finden Sie wichtige Überlegungen zur Architektur sowie die einzelnen Schritte zur Bereitstellung von sicherheits- und Compliance-orientierten Basisumgebungen in AWS.
- Erstellen einer Architektur für HIPAA-Sicherheit und -Konformität in Amazon Web Services – In diesem Whitepaper wird beschrieben, wie Unternehmen mithilfe von AWS HIPAA-konforme Anwendungen erstellen können.
- AWS-Compliance-Ressourcen – Diese Arbeitsbücher und Leitfäden könnten für Ihre Branche und Ihren Standort interessant sein.
- AWS Config – Dieser AWS-Service bewertet, zu welchem Grad die Konfiguration Ihrer Ressourcen den internen Vorgehensweisen, Branchenrichtlinien und Vorschriften entspricht.
- AWS Security Hub – Dieser AWS-Service liefert einen umfassenden Überblick über den Sicherheitsstatus in AWS. So können Sie die Compliance mit den Sicherheitsstandards in der Branche und den bewährten Methoden abgleichen.

Ausfallsicherheit in Athena

Die globale AWS-Infrastruktur ist um AWS-Regionen und Availability Zones herum aufgebaut. AWS-Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die mit einem Netzwerk mit geringer Latenz, hohem Durchsatz und hoher Redundanz verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Availability Zones ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser hoch verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen über AWS-Regionen und Availability Zones finden Sie unter AWSGlobale Infrastruktur.

Zusätzlich zur globalen AWS-Infrastruktur stellt Athena verschiedene Funktionen bereit, um Ihren Anforderungen in Bezug auf Ausfallsicherheit und Datensicherung zu erfüllen.

Athena ist Serverless – d. h. es gibt keine Infrastruktur, die eingerichtet oder verwaltet werden muss. Athena ist hochverfügbar und führt Abfragen mithilfe von Computerressourcen in mehreren Availability Zones durch, wobei Abfragen automatisch entsprechend weitergeleitet werden, wenn eine bestimmte Availability Zone nicht erreichbar ist. Athena verwendet Amazon S3 als zugrunde liegenden Datenspeicher, sodass Ihre Daten hochverfügbar sind und beständig verfügbar bleiben. Amazon S3 stellt eine beständige Infrastruktur zum Speichern wichtiger Daten bereit und ist für

eine Beständigkeit von 99,999999999 % der Objekte konzipiert. Ihre Daten werden redundant an mehreren Standorten und auf mehreren Geräten an jedem Standort gespeichert.

Infrastruktursicherheit in Athena

Als verwalteter Service ist Amazon Athena durch die globale Netzwerksicherheit von AWS geschützt. Informationen zu AWS-Sicherheitsdiensten und wie AWS die Infrastruktur schützt, finden Sie unter AWSCloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS-Umgebung anhand der bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastrukturschutz im Security Pillar AWS Well-Architected Framework.

Sie verwenden durch AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf Athena zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systemen wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS Security Token Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Verwenden Sie IAM-Richtlinien, um den Zugriff auf Athena-Operationen einzuschränken. Wenn Sie IAM-Richtlinien verwenden, stellen Sie sicher, dass Sie die bewährten Methoden von IAM befolgen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwaltete Athena-Richtlinien sind einfach zu nutzen und werden automatisch mit den erforderlichen Aktionen aktualisiert, wenn sich der Service weiterentwickelt. Vom Kunden verwaltete und eingebundene Richtlinien ermöglichen Ihnen die Optimierung von Richtlinien, indem Sie detailliertere Athena-Aktionen in der Richtlinie angeben. Gewähren Sie geeigneten Zugriff auf den Amazon-S3-Speicherort der Daten. Ausführliche Informationen sowie Szenarien zum Gewähren von Amazon-S3-Zugriff finden Sie unter Beispiel-Anleitungen: Verwalten des Zugriffs im Entwicklerhandbuch für Amazon Simple Storage Service. Weitere Informationen sowie ein Beispiel für Amazon-S3-Aktionen, die zuzulassen sind, finden Sie in der Beispiel-Bucket-Richtlinie unter Kontoübergreifender Zugriff.

Themen

- Herstellen einer Verbindung mit Amazon Athena über einen Schnittstellen-VPC-Endpunkt

Herstellen einer Verbindung mit Amazon Athena über einen Schnittstellen-VPC-Endpunkt

Sie können die Sicherheit Ihrer VPC verbessern, indem Sie einen Schnittstellen-VPC-Endpunkt (AWS PrivateLink) und einen AWS Glue-VPC-Endpunkt in Ihrer Virtual Private Cloud (VPC) verwenden. Ein VPC-Schnittstellen-Endpunkt verbessert die Sicherheit, indem er Ihnen die Kontrolle darüber gibt, welche Ziele innerhalb Ihrer VPC erreicht werden können. Jeder VPC-Endpunkt wird durch eine oder mehrere Elastic-Network-Schnittstellen (ENIs) mit privaten IP-Adressen in Ihren VPC-Subnetzen repräsentiert.

Der Schnittstellen-VPC-Endpunkt verbindet Ihre VPC direkt mit Athena, ohne ein Internet-Gateway, ein NAT-Gerät, eine VPN-Verbindung oder eine AWS Direct Connect-Verbindung. Die Instances in Ihrer VPC benötigen für die Kommunikation mit der Athena-API keine öffentlichen IP-Adressen.

Um Athena über Ihre VPC zu verwenden, müssen Sie für die Verbindung eine Instance innerhalb Ihrer VPC verwenden oder Ihr privates Netzwerk mit Ihrer VPC verbinden. Dies erreichen Sie mithilfe eines Amazon Virtual Private Network (VPN) oder mit AWS Direct Connect. Informationen zu Amazon VPN finden Sie unter VPN-Verbindungen im Benutzerhandbuch für Amazon Virtual Private Cloud. Informationen zu AWS Direct Connect finden Sie unter Erstellen einer Verbindung im AWS Direct Connect-Benutzerhandbuch.

Athena unterstützt VPC-Endpunkte in allen AWS-Regionen, in denen Amazon VPC und Athena verfügbar sind.

Sie können über die AWS Management Console oder AWS Command Line Interface (AWS CLI)-Befehle einen Schnittstellen-VPC-Endpunkt erstellen, um eine Verbindung zu Athena herzustellen. Weitere Informationen finden Sie unter Erstellen eines Schnittstellenendpunkts.

Wenn Sie nach dem Erstellen eines Schnittstellen-VPC-Endpunkts private DNS-Hostnamen für den Endpunkt aktivieren, wird der Athena-Standardendpunkt (https://athena.*Region*.amazonaws.com) in Ihren VPC-Endpunkt aufgelöst.

Wenn Sie keine privaten DNS-Hostnamen aktiviert haben, stellt Amazon VPC einen DNS-Endpunktnamen bereit, den Sie im folgenden Format verwenden können:

```
VPC_Endpoint_ID.athena.Region.vpce.amazonaws.com
```

Weitere Informationen finden Sie unter Schnittstellen-VPC-Endpunkte (AWS-PrivateLink) im Amazon-VPC-Benutzerhandbuch.

Athena unterstützt Aufrufe aller API-Aktionen innerhalb Ihrer VPC.

Erstellen einer VPC-Endpunktrichtlinie für Athena

Sie können eine Richtlinie für Amazon-VPC-Endpunkte für Athena erstellen, in der Sie Restriktionen wie die folgenden angeben:

- Prinzipal – Prinzipal, der die Aktionen ausführen kann.
- Aktionen – Aktionen, die ausgeführt werden können
- Ressourcen – Die Ressourcen, für die Aktionen ausgeführt werden können.
- Nur vertrauenswürdige Identitäten – Verwenden Sie die `aws:PrincipalOrgId`-Bedingung, um den Zugriff nur auf Anmeldeinformationen zu beschränken, die Teil Ihrer AWS-Organisation sind. Dies kann dazu beitragen, den Zugriff durch unbeabsichtigte Prinzipale zu verhindern.
- Nur vertrauenswürdige Ressourcen – Verwenden Sie diese `aws:ResourceOrgId`-Bedingung, um den Zugriff auf unbeabsichtigte Ressourcen zu verhindern.
- Nur vertrauenswürdige Identitäten und Ressourcen – Erstellen Sie eine kombinierte Richtlinie für einen VPC-Endpunkt, die den Zugriff auf unbeabsichtigte Prinzipale und Ressourcen verhindert.

Weitere Informationen finden Sie unter Steuern des Zugriffs auf Services mit VPC-Endpunkten im Amazon-VPC-Benutzerhandbuch und Anhang 2 – Beispiele für VPC-Endpunktrichtlinien im AWS-Whitepaper Aufbau eines Datenperimeters in AWS.

Example – VPC-Endpunktrichtlinie

Das folgende Beispiel erlaubt Anfragen von Organisationsidentitäten an Unternehmensressourcen und Anfragen von AWS-Service-Prinzipalen.

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

Wenn Sie IAM-Richtlinien verwenden, stellen Sie sicher, dass Sie die bewährten IAM-Methoden befolgen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Gemeinsame Subnetze

Sie können VPC-Endpunkte in Subnetzen, die mit Ihnen geteilt werden, nicht erstellen, beschreiben, ändern oder löschen. Sie können die VPC-Endpunkte jedoch in Subnetzen verwenden, die mit Ihnen geteilt werden. Weitere Informationen zur Freigabe von VPCs finden Sie unter Freigeben Ihrer VPC für andere Konten im Amazon-VPC-Benutzerhandbuch.

Konfiguration und Schwachstellenanalyse in Athena

Athena ist Serverless, daher muss keine Infrastruktur eingerichtet oder verwaltet werden. AWS wickelt grundlegende Sicherheitsaufgaben wie das Patching von Gastbetriebssystemen und Datenbanken, die Firewall-Konfiguration sowie die Notfallwiederherstellung ab. Diese Verfahren

wurden von qualifizierten Dritten überprüft und zertifiziert. Weitere Informationen finden Sie in den folgenden AWS-Ressourcen:

- Modell der geteilten Verantwortung
- Bewährte Methoden für Sicherheit, Identität und Compliance

Verwenden von Athena zum Abfragen von Daten, die in AWS Lake Formationregistriert sind

AWS Lake Formation ermöglicht es Ihnen, Zugriffsrichtlinien auf Datenbank-, Tabellen- und Spaltenebene zu definieren und durchzusetzen, wenn Sie Athena-Abfragen verwenden, um in Amazon S3 gespeicherte Daten zu lesen. Lake Formation bietet eine Autorisierungs- und Governance-Schicht für Daten, die in Amazon S3 gespeichert sind. Sie können in Lake Formation eine Hierarchie von Berechtigungen verwenden, um Berechtigungen zum Lesen von Datenkatalogobjekten wie Datenbanken, Tabellen und Spalten zu erteilen oder zu widerrufen. Lake Formation vereinfacht die Verwaltung von Berechtigungen und ermöglicht Ihnen die Implementierung einer detaillierten Zugriffskontrolle (FGAC) für Ihre Daten.

Mit Athena können Sie sowohl Daten abfragen, die in Lake Formation registriert sind, als auch Daten, die nicht in Lake Formation registriert sind.

Sie gelten, wenn Sie Athena verwenden, um Quelldaten aus Amazon-S3-Standorten abzufragen, die in Lake Formation registriert sind. Lake-Formation-Berechtigungen gelten auch, wenn Sie Datenbanken und Tabellen erstellen, die auf registrierte Amazon-S3-Datenspeicherorte verweisen. Um Athena mit Daten zu verwenden, die mit Lake Formation registriert wurden, muss Athena für die Verwendung von AWS Glue Data Catalog.

Lake-Formation-Berechtigungen gelten weder beim Schreiben von Objekten in Amazon S3 noch beim Abfragen von in Amazon S3 gespeicherten Daten oder Metadaten, die nicht bei Lake Formation registriert sind. Für Quelldaten in Amazon S3 und Metadaten, die nicht bei Lake Formation registriert sind, wird der Zugriff durch IAM-Berechtigungsrichtlinien für Amazon S3- und - AWS Glue Aktionen bestimmt. Athena Abfrageergebnisorte in Amazon S3 können nicht bei Lake Formation registriert werden, und IAM-Berechtigungsrichtlinien für Amazon S3 steuern den Zugriff. Darüber hinaus gelten die Berechtigungen für Lake Formation nicht für den Athena-Abfrageverlauf. Sie können Athena-Workgroups verwenden, um den Zugriff auf den Abfrageverlauf zu steuern.

Weitere Informationen zu Lake Formation finden Sie unter Häufig gestellte Fragen zu Lake Formation und im AWS Lake Formation -Entwicklerhandbuch.

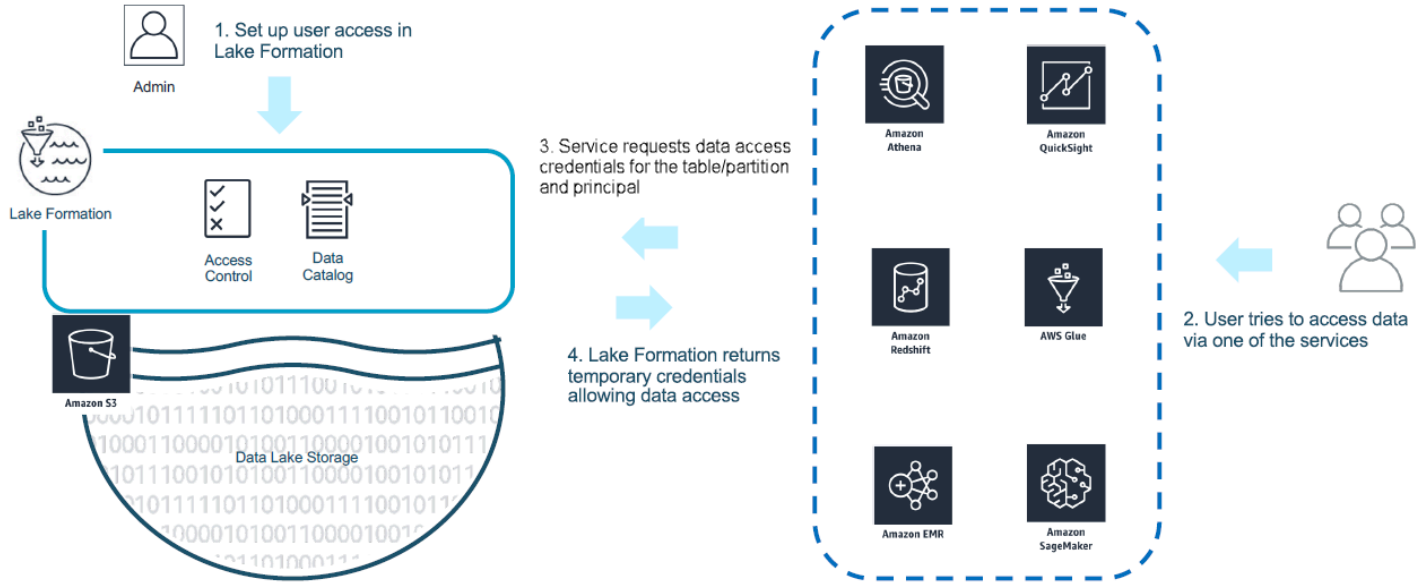
Themen

So greift Athena auf Daten zu, die bei Lake Formation angemeldet sind

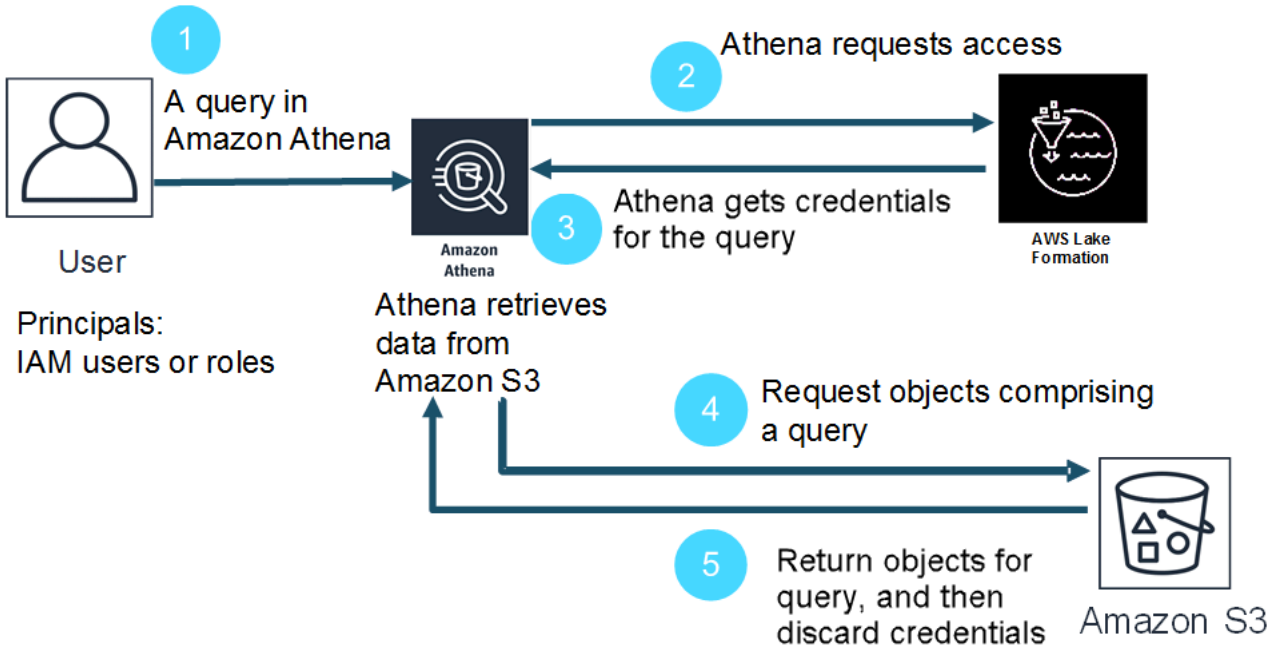
Der in diesem Abschnitt beschriebene Zugriffs-Workflow gilt nur, wenn Athena-Abfragen für Amazon-S3-Standorte und -Metadatenobjekte ausgeführt werden, die in Lake Formation registriert sind. Weitere Informationen finden Sie unter Anmelden eines Data Lake im AWS Lake Formation -Entwicklerhandbuch. Zusätzlich zum Registrieren von Daten wendet der Lake-Formation-Administrator Lake-Formation-Berechtigungen an, die den Zugriff auf Metadaten im Datenkatalog und den Datenspeicherort in Amazon S3 gewähren oder entziehen. Weitere Informationen finden Sie unter Sicherheit und Zugriffskontrolle für Metadaten und Daten im AWS Lake Formation -Entwicklerhandbuch.

Jedes Mal, wenn ein Athena-Prinzipal (Benutzer, Gruppe oder Rolle) eine Abfrage zu Daten ausführt, die mit Lake Formation registriert wurden, überprüft Lake Formation, ob der Prinzipal über die entsprechenden Lake-Formation-Berechtigungen für die Datenbank, Tabelle und den Amazon-S3-Speicherort für die Abfrage verfügt. Wenn der Prinzipal Zugriff hat, vergibt Lake Formation temporäre Anmeldeinformationen an Athena, und die Abfrage wird ausgeführt.

Das folgende Diagramm veranschaulicht den oben beschriebenen Ablauf.



Das folgende Diagramm zeigt, wie der Verkauf von Anmeldeinformationen in Athena auf der query-by-query Grundlage einer hypothetischen SELECT Abfrage für eine Tabelle mit einem in Lake Formation registrierten Amazon S3-Speicherort funktioniert:



1. Ein Prinzipal führt eine SELECT-Abfrage in Athena aus.
2. Athena analysiert die Abfrage und überprüft die Lake-Formation-Berechtigungen, um festzustellen, ob dem Prinzipal Zugriff auf die Tabelle und die Tabellenspalten gewährt wurde.

3. Wenn der Prinzipal Zugriff hat, fordert Athena Anmeldeinformationen von Lake Formation an. Wenn der Prinzipal keinen Zugriff hat, gibt Athena einen Fehler für „Zugriff verweigert“ aus.
4. Lake Formation gibt Athena Anmeldeinformationen aus, die beim Lesen von Daten aus Amazon S3 verwendet werden können, zusammen mit der Liste der zulässigen Spalten.
5. Athena verwendet die temporären Anmeldeinformationen von Lake Formation, um die Daten aus Amazon S3 abzufragen. Nachdem die Abfrage abgeschlossen ist, verwirft Athena die Anmeldeinformationen.

Überlegungen und Einschränkungen bei der Verwendung von Athena zum Abfragen von bei Lake Formation registrierten Daten

Beachten Sie Folgendes, wenn Sie Athena verwenden, um in Lake Formation registrierte Daten abzufragen. Weitere Informationen finden Sie unter Bekannte Probleme für AWS Lake Formation im AWS Lake Formation -Entwicklerhandbuch.

Überlegungen und Einschränkungen

- Spaltenmetadaten, die unter Umständen für nicht autorisierte Benutzer mit Avro und benutzerdefinierten sichtbar sind SerDe
- Arbeiten mit Lake-Formation-Berechtigungen für Ansichten
- Differenzierte Zugriffskontrolle von Lake Formation und Athena-Arbeitsgruppen
- Athena-Abfrageergebnisse-Speicherort in Amazon S3 nicht bei Lake Formation registriert
- Verwenden von Athena-Workgroups zum Einschränken des Zugriffs auf den Abfrageverlauf
- Kontoübergreifender Zugriff auf den Datenkatalog
- CSE-KMS-verschlüsselte Amazon S3-Standorte, die bei Lake Formation registriert sind
- Speicherorte für partitionierte Daten, die bei Lake Formation registriert sind, müssen sich in Tabellen-Unterverzeichnissen befinden
- Erstellen von CTAS (Table As Select)-Abfragen erfordern Amazon-S3-Schreibberechtigungen
- Die DESCRIBE-Berechtigung ist für die Standarddatenbank erforderlich

Spaltenmetadaten, die unter Umständen für nicht autorisierte Benutzer mit Avro und benutzerdefinierten sichtbar sind SerDe

Die Autorisierung auf Lake-Formation-Spaltenebene verhindert, dass ein Benutzer auf Daten in Spalten zugreift, für die er keine Lake-Formation-Berechtigungen besitzt. In bestimmten Situationen

können Benutzer jedoch auf Metadaten zugreifen, die alle Spalten in der Tabelle beschreiben, einschließlich der Spalten, für deren Daten sie keine Berechtigungen besitzen.

Dies tritt auf, wenn Spaltenmetadaten in Tabelleneigenschaften für Tabellen gespeichert werden, die entweder das Apache Avro-Speicherformat oder einen benutzerdefinierten Serializer/Deserializer (SerDe) verwenden, in dem das Tabellenschema zusammen mit der SerDe Definition in Tabelleneigenschaften definiert ist. Wenn Sie Athena mit Lake Formation verwenden, empfehlen wir, den Inhalt der Tabelleneigenschaften zu überprüfen, die Sie in Lake Formation registrieren, und nach Möglichkeit die in den Tabelleneigenschaften gespeicherten Informationen zu begrenzen, um zu verhindern, dass vertrauliche Metadaten für Benutzer sichtbar sind.

Arbeiten mit Lake-Formation-Berechtigungen für Ansichten

Für bei Lake Formation registrierte Daten kann ein Athena-Benutzer nur dann ein `VIEW` erstellen, wenn er über Lake-Formation-Berechtigungen für die Tabellen, Spalten und Amazon-S3-Quelldatenspeicherorte verfügt, auf denen `VIEW` basiert. Nachdem ein `VIEW` in Athena erstellt wurde, können die Berechtigungen für Lake Formation auf das `VIEW` angewendet werden. Berechtigungen auf Spaltenebene sind für einen `VIEW` nicht verfügbar. Benutzer mit Lake-Formation-Berechtigungen für einen `VIEW` aber ohne Berechtigungen für die Tabelle und Spalten, auf denen die Ansicht basiert, können den `VIEW` nicht zur Datenabfrage verwenden. Benutzer mit dieser Berechtigungskombination können jedoch Anweisungen wie `DESCRIBE VIEW`, `SHOW CREATE VIEW` und `SHOW COLUMNS` verwenden, um `VIEW`-Metadaten anzuzeigen. Stellen Sie daher sicher, dass Sie die Lake-Formation-Berechtigungen für jeden `VIEW` an den zugrunde liegenden Tabellenberechtigungen ausrichten. Zellfilter, die für eine Tabelle definiert sind, gelten nicht für einen `VIEW` für diese Tabelle. Die Namen der Ressourcenlinks müssen den gleichen Namen wie die Ressource im Ausgangskonto haben. Bei der Arbeit mit Ansichten in einer kontoübergreifenden Konfiguration gibt es zusätzliche Einschränkungen. Weitere Informationen zum Einrichten von Berechtigungen für freigegebene Ansichten über Konten hinweg finden Sie unter Kontoübergreifender Zugriff auf den Datenkatalog.

Differenzierte Zugriffskontrolle von Lake Formation und Athena-Arbeitsgruppen

Benutzer in derselben Athena-Arbeitsgruppe können die Daten anzeigen, die die differenzierte Zugriffskontrolle von Lake Formation so konfiguriert hat, dass sie für die Arbeitsgruppe zugänglich sind. Weitere Informationen zur Verwendung der differenzierten Zugriffskontrolle in Lake Formation finden Sie unter Verwaltung der differenzierten Zugriffskontrolle mithilfe von AWS Lake Formation im AWS -Big-Data-Blog.

Athena-Abfrageergebnisse-Speicherort in Amazon S3 nicht bei Lake Formation registriert

Die Abfrageergebnisorte in Amazon S3 für Athena können nicht bei Lake Formation registriert werden. Lake-Formation-Berechtigungen beschränken den Zugriff auf diese Standorte nicht. Wenn Sie den Zugriff nicht einschränken, können Athena-Benutzer auf Abfrageergebnisdateien und Metadaten zugreifen, wenn sie keine Lake-Formation-Berechtigungen für die Daten haben. Um dies zu vermeiden, sollten Sie Arbeitsgruppen verwenden, um den Speicherort für Abfrageergebnisse anzugeben und die Arbeitsgruppenmitgliedschaft mit den Lake-Formation-Berechtigungen auszurichten. Anschließend können Sie IAM-Berechtigungsrichtlinien verwenden, um den Zugriff auf Abfrageergebnisspeicherorte zu beschränken. Weitere Informationen zu Abfrageergebnissen finden Sie unter Arbeiten mit Abfrageergebnissen, letzten Abfragen und Ausgabedateien.

Verwenden von Athena-Workgroups zum Einschränken des Zugriffs auf den Abfrageverlauf

Der Abfrageverlauf von Athena stellt eine Liste gespeicherter Abfragen und vollständiger Abfragezeichenfolgen bereit. Sofern Sie nicht Arbeitsgruppen verwenden, um den Zugriff auf Abfrageverläufe zu trennen, können Athena-Benutzer, die nicht zum Abfragen von Daten in Lake Formation berechtigt sind, Abfragezeichenfolgen anzeigen, die für diese Daten ausgeführt werden, einschließlich Spaltennamen, Auswahlkriterien usw. Es wird empfohlen, Arbeitsgruppen zu verwenden, um Abfrageverläufe zu trennen und Athena-Arbeitsgruppenmitgliedschaft mit Lake-Formation-Berechtigungen auszurichten, um den Zugriff zu beschränken. Weitere Informationen finden Sie unter Verwendung von Arbeitsgruppen zur Kontrolle des Abfragenzugriffs und der Kosten.

Kontoübergreifender Zugriff auf den Datenkatalog

Um auf einen Datenkatalog in einem anderen Konto zuzugreifen, können Sie das kontenübergreifende AWS Glue -Feature von Athena verwenden oder den kontenübergreifenden Zugriff in Lake Formation einrichten.

Kontoübergreifender Athena-Datenkatalog-Zugriff

Sie können die kontoübergreifende AWS Glue Katalogfunktion von Athena verwenden, um den Katalog in Ihrem Konto zu registrieren. Diese Funktion ist nur in Athena-Engine-Version 2 und späteren Versionen verfügbar und auf die Verwendung in derselben Region zwischen Konten beschränkt. Weitere Informationen finden Sie unter Registrieren eines AWS Glue Data Catalog von einem anderen Konto.

Wenn für den freizugebenden Datenkatalog eine Ressourcenrichtlinie in konfiguriert ist AWS Glue, muss sie aktualisiert werden, um den Zugriff auf zu ermöglichen AWS Resource Access Manager

und Konto B Berechtigungen zur Verwendung des Datenkatalogs von Konto A zu erteilen, wie im folgenden Beispiel.

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

Weitere Informationen finden Sie unter Kontoübergreifender Zugriff auf AWS Glue -Datenkataloge.

Einrichten des kontoübergreifenden Zugriffs in Lake Formation

AWS Lake Formation Mit können Sie ein einzelnes Konto verwenden, um einen zentralen Datenkatalog zu verwalten. Sie können dieses Feature verwenden, um den kontoübergreifenden Zugriff auf Datenkatalog-Metadaten und zugrunde liegende Daten zu implementieren. Beispielsweise kann ein Besitzerkonto einem anderen (Empfänger-)Konto `SELECT` die Berechtigung für eine Tabelle erteilen.

Damit eine freigegebene Datenbank oder Tabelle im Athena-Abfrage-Editor angezeigt wird, erstellen Sie in Lake Formation einen Ressourcenlink zur freigegebenen Datenbank oder Tabelle. Wenn das Empfängerkonto in Lake Formation die Tabelle des Besitzers abfragt, CloudTrail fügt das Datenzugriffsereignis den Protokollen sowohl für das Empfängerkonto als auch für das Eigentümerkonto hinzu.

Beachten Sie bei freigegebenen Ansichten die folgenden Punkte:

- Abfragen werden auf Zielressourcen-Links ausgeführt, nicht in der Quelltabelle oder -Ansicht, und dann wird die Ausgabe für das Zielkonto freigegeben.
- Es reicht nicht aus, nur die Ansicht zu teilen. Alle Tabellen, die an der Erstellung der Ansicht beteiligt sind, müssen Teil der kontoübergreifenden Freigabe sein.
- Der Name des auf den freigegebenen Ressourcen erstellten Ressourcenlinks muss mit dem Namen der Ressource im Eigentümerkonto übereinstimmen. Wenn der Name nicht übereinstimmt, wird eine Fehlermeldung wie fehlgeschlagene Analyse der gespeicherten Ansicht 'awsdatacatalog.*my-lf-resource-link*..*my-lf-view*': Zeile 3:3: Schema *schema_name* existiert nicht angezeigt.

Weitere Informationen zum kontenübergreifenden Zugriff in Lake Formation finden Sie in den folgenden Ressourcen im AWS Lake Formation -Entwicklerhandbuch:

Kontenübergreifender Zugriff

Funktionsweise von Ressourcenverbindungen in Lake Formation

Kontoübergreifende CloudTrail Protokollierung

CSE-KMS-verschlüsselte Amazon S3-Standorte, die bei Lake Formation registriert sind

Open Table Format (OTF)-Tabellen wie Apache Iceberg mit den folgenden Merkmalen können nicht mit Athena abgefragt werden:

- Die Tabellen basieren auf Amazon S3-Datenspeicherorten, die bei Lake Formation registriert sind.
- Die Objekte in Amazon S3 werden mit clientseitiger Verschlüsselung (CSE) verschlüsselt.
- Die Verschlüsselung verwendet vom AWS KMS Kunden verwaltete Schlüssel (`CSE_KMS`).

Um Nicht-OTF-Tabellen abzufragen, die mit einem `CSE_KMS` Schlüssel verschlüsselt sind, fügen Sie der Richtlinie des AWS KMS Schlüssels, den Sie für die CSE-Verschlüsselung verwenden, den

folgenden Block hinzu. *<KMS_KEY_ARN>* ist der ARN des AWS KMS Schlüssels, der die Daten verschlüsselt. *<IAM-ROLE-ARN>* ist der ARN der IAM-Rolle, die den Amazon S3-Standort in Lake Formation registriert.

```
{
    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal": {
        "AWS": "*"
    },
    "Action": "kms:Decrypt",
    "Resource": "<KMS-KEY-ARN>",
    "Condition": {
        "ArnLike": {
            "aws:PrincipalArn": "<IAM-ROLE-ARN>"
        }
    }
}
```

Speicherorte für partitionierte Daten, die bei Lake Formation registriert sind, müssen sich in Tabellen-Unterverzeichnissen befinden

Bei Lake Formation registrierte partitionierte Tabellen müssen über partitionierte Daten in Verzeichnissen verfügen, die Unterverzeichnisse der Tabelle in Amazon S3 sind. Beispielsweise kann eine Tabelle mit dem Speicherort `s3://mydata/mytable` und den Partitionen `s3://mydata/mytable/dt=2019-07-11`, `s3://mydata/mytable/dt=2019-07-12` usw. in Lake Formation registriert und mit Athena abgefragt werden. Andererseits kann eine Tabelle mit dem Speicherort `s3://mydata/mytable` und Partitionen in `s3://mydata/dt=2019-07-11`, `s3://mydata/dt=2019-07-12` usw. nicht in Lake Formation registriert werden. Da solche Partitionen keine Unterverzeichnisse von `s3://mydata/mytable` sind, können sie auch nicht von Athena gelesen werden.

Erstellen von CTAS (Table As Select)-Abfragen erfordern Amazon-S3-Schreibberechtigungen

Create Table As Statements (CTAS) erfordern Schreibzugriff auf den Amazon-S3-Speicherort von Tabellen. Um CTAS-Abfragen für bei Lake Formation registrierte Daten auszuführen, müssen Athena-Benutzer zusätzlich zu den entsprechenden Lake-Formation-Berechtigungen zum Lesen der Datenstandorte über IAM-Berechtigungen zum Schreiben in die Tabelle von Amazon-S3-Standorten verfügen. Weitere Informationen finden Sie unter Erstellen einer Tabelle aus Abfrageergebnissen (CTAS).

Die DESCRIBE-Berechtigung ist für die Standarddatenbank erforderlich

Die Lake-Formation-Berechtigung DESCRIBE ist für die `default`-Datenbank erforderlich. Der folgende AWS CLI Beispielbefehl erteilt dem Benutzer `datalake_user1` im AWS Konto die -`DESCRIBE`Berechtigung für die `default` Datenbank `111122223333`.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --
permissions "DESCRIBE" --resource '{ "Database": {"Name":"default"}}
```

Weitere Informationen finden Sie in der Referenz zu Lake-Formation-Berechtigungen im AWS Lake Formation -Entwicklerhandbuch.

Verwalten von Lake-Formation- und Athena-Benutzerberechtigungen

Lake Formation verkauft Anmeldeinformationen, um Amazon-S3-Datenspeicher abzufragen, die bei Lake Formation registriert sind. Wenn Sie zuvor IAM-Richtlinien verwendet haben, um Berechtigungen zum Lesen von Datenspeicherorten in Amazon S3 zu gewähren oder zu verweigern, können Sie stattdessen Lake-Formation-Berechtigungen verwenden. Allerdings sind weiterhin andere IAM-Berechtigungen erforderlich.

Wenn Sie IAM-Richtlinien verwenden, stellen Sie sicher, dass Sie die bewährten Methoden von IAM befolgen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

In den folgenden Abschnitten werden die Berechtigungen zusammengefasst, die erforderlich sind, um die in Lake Formation registrierten Daten mithilfe von Athena abzufragen. Weitere Informationen finden Sie unter Sicherheit in AWS Lake Formation im AWS Lake Formation -Entwicklerhandbuch.

Berechtigungsübersicht

- Identitätsbasierte Berechtigungen für Lake Formation und Athena
- Amazon-S3-Berechtigungen für Speicherorte von Athena-Abfrageergebnissen
- Athena-Workgoup-Mitgliedschaften zum Abfragen des Verlaufs
- Lake-Formation-Berechtigungen für Daten
- IAM-Berechtigungen zum Schreiben in Amazon-S3-Speicherorte
- Berechtigungen für verschlüsselte Daten, Metadaten und Athena-Abfrageergebnisse
- Ressourcenbasierte Berechtigungen für Amazon-S3-Buckets in externen Konten (optional)

Identitätsbasierte Berechtigungen für Lake Formation und Athena

Jeder, der Athena zum Abfragen von bei Lake Formation registrierten Daten verwendet, muss über eine IAM-Berechtigungsrichtlinie verfügen, die die `lakeformation:GetDataAccess`-Aktion zulässt. AWS Verwaltete Richtlinie: AmazonAthenaFullAccess erlaubt diese Aktion. Wenn Sie eingebundenen Richtlinien verwenden, stellen Sie sicher, dass Sie die Berechtigungsrichtlinien aktualisieren, um diese Aktion zuzulassen.

In Lake Formation hat ein Data-Lake-Administrator Berechtigungen zum Erstellen von Metadatenobjekten wie Datenbanken und Tabellen, Erteilen von Lake-Formation-Berechtigungen an andere Benutzer und Registrieren neuer Amazon-S3-Speicherorte. Zur Registrierung neuer Standorte sind Berechtigungen für die serviceverknüpfte Rolle für Lake Formation erforderlich. Weitere Informationen finden Sie unter Erstellen eines Data-Lake-Administrators und Servicegebundene Rollenberechtigungen für Lake Formation im AWS Lake Formation - Entwicklerhandbuch.

Ein Lake-Formation-Benutzer kann Athena verwenden, um Datenbanken, Tabellen, Tabellenspalten und zugrunde liegende Amazon-S3-Datenspeicher basierend auf den Lake-Formation-Berechtigungen abzufragen, die ihm von Data-Lake-Administratoren erteilt wurden. Benutzer können keine Datenbanken oder Tabellen erstellen oder neue Amazon-S3-Speicherorte bei Lake Formation registrieren. Weitere Informationen finden Sie unter Erstellen eines Data Lake-Benutzers im AWS Lake Formation -Entwicklerhandbuch.

In Athena steuern identitätsbasierte Berechtigungsrichtlinien, einschließlich derer für Athena-Arbeitsgruppen, weiterhin den Zugriff auf Athena-Aktionen für Amazon-Web-Services-Kontobenutzer. Darüber hinaus kann der Verbundzugriff über die SAML-basierte Authentifizierung bereitgestellt werden, die mit Athena-Treibern verfügbar ist. Weitere Informationen finden Sie unter Verwendung von Arbeitsgruppen zur Kontrolle des Abfragenzugriffs und der Kosten, IAM-Richtlinien für den Zugriff auf Arbeitsgruppen und Aktivieren des föderierten Zugriffs auf die Athena-API.

Weitere Informationen finden Sie unter Erteilen von Lake Formation-Berechtigungen im AWS Lake Formation -Entwicklerhandbuch.

Amazon-S3-Berechtigungen für Speicherorte von Athena-Abfrageergebnissen

Die Abfrageergebnisorte in Amazon S3 für Athena können nicht bei Lake Formation registriert werden. Lake-Formation-Berechtigungen beschränken den Zugriff auf diese Standorte nicht. Wenn Sie den Zugriff nicht einschränken, können Athena-Benutzer auf Abfrageergebnisdateien und Metadaten zugreifen, wenn sie keine Lake-Formation-Berechtigungen für die Daten haben. Um dies

zu vermeiden, sollten Sie Arbeitsgruppen verwenden, um den Speicherort für Abfrageergebnisse anzugeben und die Arbeitsgruppenmitgliedschaft mit den Lake-Formation-Berechtigungen auszurichten. Anschließend können Sie IAM-Berechtigungsrichtlinien verwenden, um den Zugriff auf Abfrageergebnisspeicherorte zu beschränken. Weitere Informationen zu Abfrageergebnissen finden Sie unter Arbeiten mit Abfrageergebnissen, letzten Abfragen und Ausgabedateien.

Athena-Workgoup-Mitgliedschaften zum Abfragen des Verlaufs

Der Abfrageverlauf von Athena stellt eine Liste gespeicherter Abfragen und vollständiger Abfragezeichenfolgen bereit. Sofern Sie nicht Arbeitsgruppen verwenden, um den Zugriff auf Abfrageverläufe zu trennen, können Athena-Benutzer, die nicht zum Abfragen von Daten in Lake Formation berechtigt sind, Abfragezeichenfolgen anzeigen, die für diese Daten ausgeführt werden, einschließlich Spaltennamen, Auswahlkriterien usw. Es wird empfohlen, Arbeitsgruppen zu verwenden, um Abfrageverläufe zu trennen und Athena-Arbeitsgruppenmitgliedschaft mit Lake-Formation-Berechtigungen auszurichten, um den Zugriff zu beschränken. Weitere Informationen finden Sie unter Verwendung von Arbeitsgruppen zur Kontrolle des Abfragenzugriffs und der Kosten.

Lake-Formation-Berechtigungen für Daten

Zusätzlich zu der Grundberechtigung zur Verwendung von Lake Formation müssen Athena-Benutzer über Lake-Formation-Berechtigungen verfügen, um auf die von ihnen abgefragten Ressourcen zuzugreifen. Diese Berechtigungen werden von einem Lake-Formation-Administrator erteilt und verwaltet. Weitere Informationen finden Sie unter Sicherheit und Zugriffskontrolle für Metadaten und Daten im AWS Lake Formation -Entwicklerhandbuch.

IAM-Berechtigungen zum Schreiben in Amazon-S3-Speicherorte

Die Lake-Formation-Berechtigungen für Amazon S3 beinhalten nicht die Möglichkeit, in Amazon S3 zu schreiben. Create Table As Statements (CTAS) erfordern Schreibzugriff auf den Amazon-S3-Speicherort von Tabellen. Um CTAS-Abfragen für bei Lake Formation registrierte Daten auszuführen, müssen Athena-Benutzer zusätzlich zu den entsprechenden Lake-Formation-Berechtigungen zum Lesen der Datenstandorte über IAM-Berechtigungen zum Schreiben in die Tabelle von Amazon-S3-Standorten verfügen. Weitere Informationen finden Sie unter Erstellen einer Tabelle aus Abfrageergebnissen (CTAS).

Berechtigungen für verschlüsselte Daten, Metadaten und Athena-Abfrageergebnisse

Zugrunde liegende Quelldaten in Amazon S3 und Metadaten im Datenkatalog, der bei Lake Formation registriert ist, können verschlüsselt werden. Es gibt keine Änderung an der Art und Weise,

wie die Verschlüsselung von Abfrageergebnissen von Athena verarbeitet wird, wenn Athena zum Abfragen von Daten verwendet wird, die in Lake Formation registriert sind. Weitere Informationen finden Sie unter Verschlüsseln der in Amazon S3 gespeicherten Athena-Abfrageergebnisse.

- Verschlüsseln von Quelldaten – Die Verschlüsselung von Amazon-S3-Datenspeicherorten-Quelldaten wird unterstützt. Athena-Benutzer, die verschlüsselte Amazon-S3-Standorte abfragen, die bei Lake Formation registriert sind, benötigen Berechtigungen zum Verschlüsseln und Entschlüsseln von Daten. Weitere Informationen zu Anforderungen finden Sie unter Unterstützte Verschlüsselungsoptionen der Amazon S3 und Berechtigungen für verschlüsselte Daten in Amazon S3.
- Verschlüsseln von Metadaten – Das Verschlüsseln von Metadaten im Datenkatalog wird unterstützt. Für Prinzipale, die Athena verwenden, müssen identitätsbasierte Richtlinien die Aktionen `"kms:GenerateDataKey"`, `"kms:Decrypt"` und `"kms:Encrypt"` für den Schlüssel zulassen, mit dem Metadaten verschlüsselt werden. Weitere Informationen finden Sie unter Verschlüsseln Ihres Datenkatalogs im Entwicklerhandbuch fürAWS Glue und Zugriff auf verschlüsselte Metadaten von Athena im AWS Glue Data Catalog.

Ressourcenbasierte Berechtigungen für Amazon-S3-Buckets in externen Konten (optional)

Um einen Amazon-S3-Datenspeicherort in einem anderen Konto abzufragen, muss eine ressourcenbasierte IAM-Richtlinie (Bucket-Richtlinie) den Zugriff auf den Speicherort ermöglichen. Weitere Informationen finden Sie unter Kontoübergreifender Zugriff auf Amazon-S3-Buckets in Athena.

Informationen zum Zugriff auf einen Datenkatalog in einem anderen Konto finden Sie unter Kontoübergreifender Athena-Datenkatalog-Zugriff.

Anwenden von Lake-Formation-Berechtigungen auf vorhandene Datenbanken und Tabellen

Wenn Sie noch nicht mit Athena vertraut sind und den Zugriff auf Abfragedaten mit Lake Formation konfigurieren, müssen Sie keine IAM-Richtlinien konfigurieren, damit Benutzer Amazon-S3-Daten lesen und Metadaten erstellen können. Sie können Berechtigungen mit Lake Formation verwalten.

Das Registrieren von Daten in Lake Formation und das Aktualisieren von IAM-Berechtigungsrichtlinien ist nicht erforderlich. Wenn Daten nicht bei Lake Formation registriert sind, können Athena-Benutzer, die über die entsprechenden Berechtigungen in Amazon S3 verfügen – und gegebenenfalls – weiterhin Daten abfragen AWS Glue, die nicht bei Lake Formation registriert sind.

Wenn Sie bereits Athena-Benutzer haben, die Daten abfragen, die nicht bei Lake Formation registriert sind, können Sie IAM-Berechtigungen für Amazon S3 – und AWS Glue Data Catalogggf. – aktualisieren, sodass Sie Lake-Formation-Berechtigungen verwenden können, um den Benutzerzugriff zentral zu verwalten. Wenn Sie die Berechtigung zum Lesen von Amazon-S3-Datenverzeichnissen erhalten möchten, können Sie ressourcen- und identitätsbasierte Richtlinien aktualisieren, um Amazon-S3-Berechtigungen zu ändern. Wenn Sie für den Zugriff auf Metadaten Richtlinien auf Ressourcenebene für eine differenzierte Zugriffskontrolle mit konfiguriert haben AWS Glue, können Sie stattdessen Lake-Formation-Berechtigungen verwenden, um den Zugriff zu verwalten.

Weitere Informationen finden Sie unter Differenzierter Zugriff auf Datenbanken und Tabellen in AWS Glue Data Catalog und Aktualisieren von AWS Glue Datenberechtigungen auf das AWS Lake Formation Modell im AWS Lake Formation -Entwicklerhandbuch.

Verwenden von Lake Formation und den Athena-JDBC- und ODBC-Treibern für den Verbundzugriff auf Athena

Die Athena JDBC- und ODBC-Treiber unterstützen den SAML 2.0-basierten Verbund mit Athena mithilfe von Identitätsanbietern von Okta und Microsoft Active Directory Federation Services (AD FS). Durch die Integration von Amazon Athena mit AWS Lake Formation aktivieren Sie die SAML-basierte Authentifizierung bei Athena mit Unternehmensanmeldeinformationen. Mit Lake Formation und AWS Identity and Access Management (IAM) können Sie eine differenzierte Zugriffskontrolle auf Spaltenebene über die dem SAML-Benutzer verfügbaren Daten aufrechterhalten. Mit den Athena-JDBC- und ODBC-Treibern ist ein Verbundzugriff für den Werkzeug- oder programmatischen Zugriff verfügbar.

Um Athena für den Zugriff auf eine von Lake Formation gesteuerte Datenquelle zu verwenden, müssen Sie den auf SAML 2.0 basierenden Verbund aktivieren, indem Sie Ihre Identitätsanbieter-(IdP) und AWS Identity and Access Management-(IAM)-Rollen konfigurieren. Die detaillierten Schritte finden Sie unter Tutorial: Konfigurieren des Verbundzugriffs für Okta-Benutzer auf Athena mithilfe von Lake Formation und JDBC.

Voraussetzungen

Um Amazon Athena und Lake Formation für den Verbundzugriff zu verwenden, müssen Sie die folgenden Anforderungen erfüllen:

- Sie verwalten Ihre Unternehmensidentitäten mit einem vorhandenen SAML-basierten Identitätsanbieter wie Okta oder Microsoft Active Directory Federation Services (AD FS).

- Sie verwenden das AWS Glue Data Catalog als Metadatenspeicher.
- Sie definieren und verwalten Berechtigungen in Lake Formation, um auf Datenbanken, Tabellen und Spalten in AWS Glue Data Catalog zuzugreifen. Weitere Informationen finden Sie im AWS Lake Formation-Entwicklerhandbuch.
- Sie verwenden Version 2.0.14 oder höher des Athena-JDBC-Treibers oder Version 1.1.3 oder höher des Athena-ODBC-Treibers.

Überlegungen und Einschränkungen

Wenn Sie den Athena JDBC- oder ODBC-Treiber und Lake Formation verwenden, um den Verbundzugriff auf Athena zu konfigurieren, beachten Sie folgende Punkte:

- Derzeit unterstützen der Athena-JDBC-Treiber und ODBC-Treiber die Identitätsanbieter Okta und Microsoft Active Directory Federation Services AD FS (AD FS). Obwohl der Athena-JDBC-Treiber über eine generische SAML-Klasse verfügt, die auf andere Identitätsanbieter erweitert werden kann, kann die Unterstützung für benutzerdefinierte Erweiterungen eingeschränkt sein, die andere Identitätsanbieter (IdPs) für die Verwendung mit Athena ermöglichen.
- Der Verbundzugriff mithilfe der JDBC- und ODBC-Treiber ist nicht mit der Funktion zur Weitergabe vertrauenswürdiger Identitäten von IAM Identity Center kompatibel.
- Derzeit können Sie die Athena-Konsole nicht verwenden, um die Unterstützung für IdP - und SAML-Verwendung mit Athena zu konfigurieren. Um diese Unterstützung zu konfigurieren, verwenden Sie den Identitätsanbieter von Drittanbietern, die Verwaltungskonsolen Lake Formation und IAM sowie den JDBC- oder ODBC-Treiberclient.
- Sie sollten die SAML-2.0-Spezifikation und ihre Funktionsweise mit Ihrem Identitätsanbieter verstehen, bevor Sie Ihren Identitätsanbieter und SAML für die Verwendung mit Lake Formation und Athena konfigurieren.
- SAML-Anbieter und die Athena JDBC- und ODBC-Treiber werden von Drittanbietern bereitgestellt, daher kann die Unterstützung durch AWS bei Problemen im Zusammenhang mit ihrer Verwendung eingeschränkt sein.

Themen

- Tutorial: Konfigurieren des Verbundzugriffs für Okta-Benutzer auf Athena mithilfe von Lake Formation und JDBC

Tutorial: Konfigurieren des Verbundzugriffs für Okta-Benutzer auf Athena mithilfe von Lake Formation und JDBC

In diesem Tutorial erfahren Sie, wie Sie Okta-, AWS Lake Formation-, AWS Identity and Access Management-Berechtigungen und den Athena-JDBC-Treiber konfigurieren, um die SAML-basierte Verbundverwendung von Athena zu ermöglichen. Lake Formation bietet eine differenzierte Zugriffssteuerung über die Daten, die in Athena für den SAML-basierten Benutzer verfügbar sind. Zum Einrichten dieser Konfiguration verwendet das Tutorial die Okta-Entwicklerkonsole, die AWS-IAM- und Lake-Formation-Konsolen sowie das SQL-Workbench/J-Tool.

Voraussetzungen

In diesem Tutorial wird davon ausgegangen, dass Sie Folgendes gemacht haben:

- Erstellt ein Amazon-Web-Services-Konto. Um ein Konto zu erstellen, besuchen Sie die Amazon-Web-Services-Homepage.
- Einrichten eines Speicherorts für Abfrageergebnisse für Athena in Amazon S3.
- Registrierte einen Amazon-S3-Daten-Bucket-Standort bei Lake Formation.
- Definiert eine Datenbank und Tabellen im AWS Glue-Datenkatalog, die auf Ihre Daten in Amazon S3 verweisen.
  - Sie noch keine Tabelle definiert haben, führen Sie entweder einen AWS Glue-Crawler aus oder verwenden Sie Athena, um eine Datenbank und eine oder mehrere Tabellen für die Daten zu definieren, auf die Sie zugreifen möchten.
  - In diesem Tutorial wird eine Tabelle verwendet, die auf dem Datensatz für NYC-Taxifahrten basiert, der in der Registry der offenen Daten auf AWS verfügbar ist. Das Tutorial verwendet den Datenbanknamen `tripdb` und den Tabellennamen `nyctaxi`.

Anleitungsschritte

- Schritt 1: Erstellen eines Okta-Kontos
- Schritt 2: Hinzufügen von Benutzern und Gruppen zu Okta
- Schritt 3: Einrichten einer Okta-Anwendung für SAML-Authentifizierung
- Schritt 4: Erstellen eines AWS-SAML-Identitätsanbieters und einer IAM-Rolle für den Lake-Formation-Zugriff
- Schritt 5: IAM-Rolle und SAML-Identitätsanbieter zur Okta-Anwendung hinzufügen
- Schritt 6: Erteilen von Benutzer- und Gruppenberechtigungen durch AWS Lake Formation

Schritt 1: Erstellen eines Okta-Kontos

In diesem Tutorial wird Okta als SAML-basierten Identitätsanbieter verwendet. Wenn Sie noch kein Okta-Konto besitzen, können Sie ein kostenloses erstellen. Ein Okta-Konto ist erforderlich, damit Sie eine Okta-Anwendung für die SAML-Authentifizierung erstellen können.

So erstellen Sie ein Okta-Konto

1. Um Okta zu verwenden, navigieren Sie zur Anmeldeseite für Okta-Entwickler und erstellen Sie ein kostenloses Okta-Testkonto. Der Developer Edition Service ist bis zu den von Okta unter developer.okta.com/pricing angegebenen Grenzen kostenlos.
2. Wenn Sie die Aktivierungs-E-Mail erhalten, aktivieren Sie Ihr Konto.

   Ihnen wird ein Okta-Domain-Name zugewiesen. Speichern Sie den Domain-Namen als Referenz. Später verwenden Sie den Domain-Namen (*<okta-idp-domain>*) in der JDBC-Zeichenfolge, die eine Verbindung zu Athena herstellt.

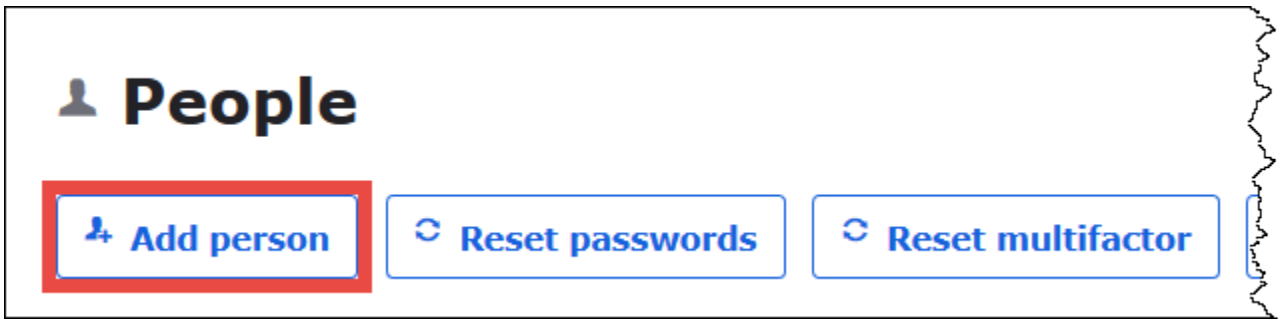
Schritt 2: Hinzufügen von Benutzern und Gruppen zu Okta

In diesem Schritt verwenden Sie die Okta-Konsole, um die folgenden Aufgaben auszuführen:

- Erstellen Sie zwei Okta-Benutzer.
- Erstellen von zwei Okta-Gruppen
- Fügen Sie jeder Okta-Gruppe einen Okta-Benutzer hinzu.

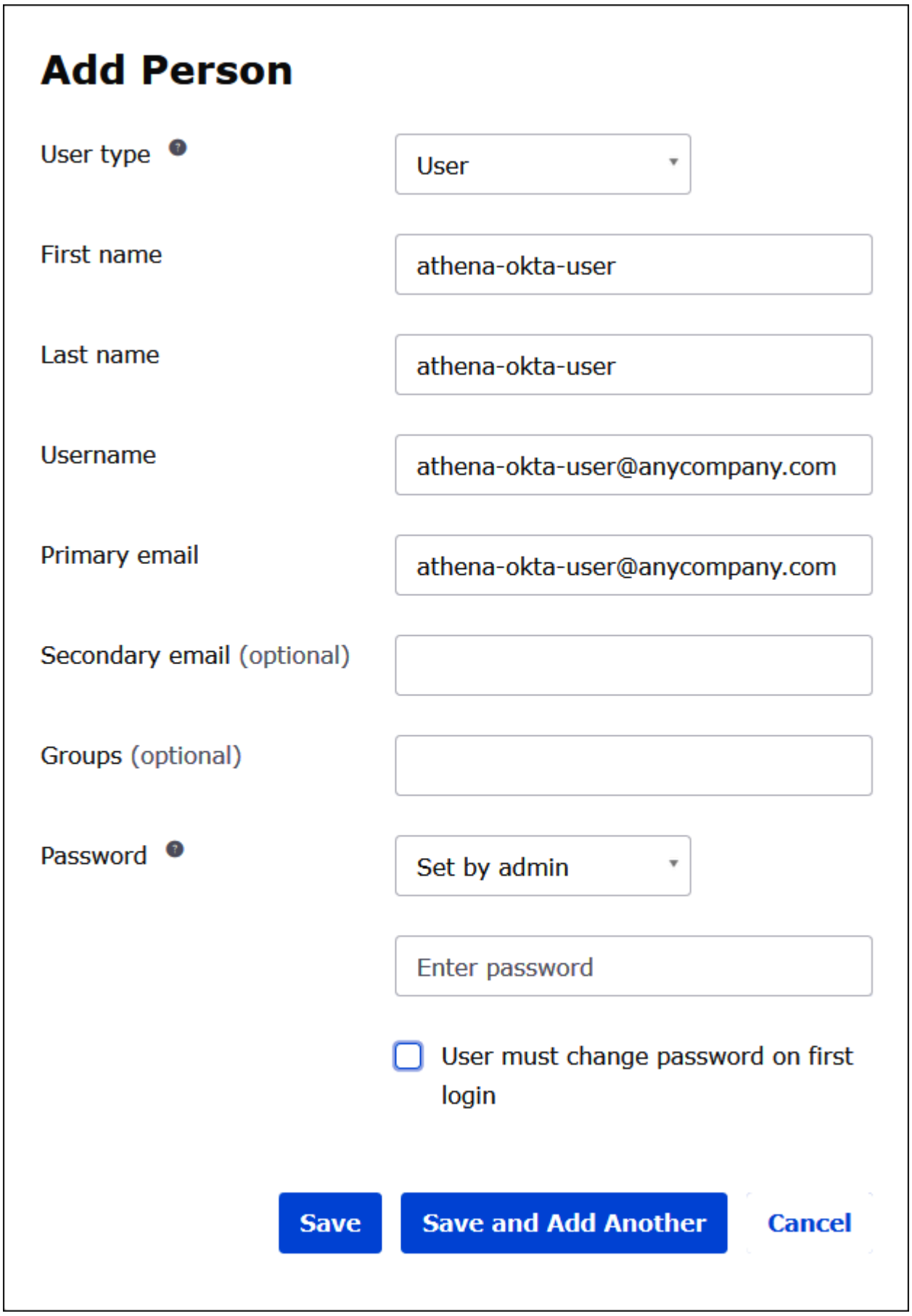
So fügen Sie Okta Benutzer hinzu

1. Nachdem Sie Ihr Okta-Konto aktiviert haben, melden Sie sich als Administrator bei der zugewiesenen Okta-Domain an.
2. Wählen Sie im linken Navigationsbereich Verzeichnis und dann Personen aus.
3. Wählen Sie Person hinzufügen, um einen neuen Benutzer hinzuzufügen, der über den JDBC-Treiber auf Athena zugreift.

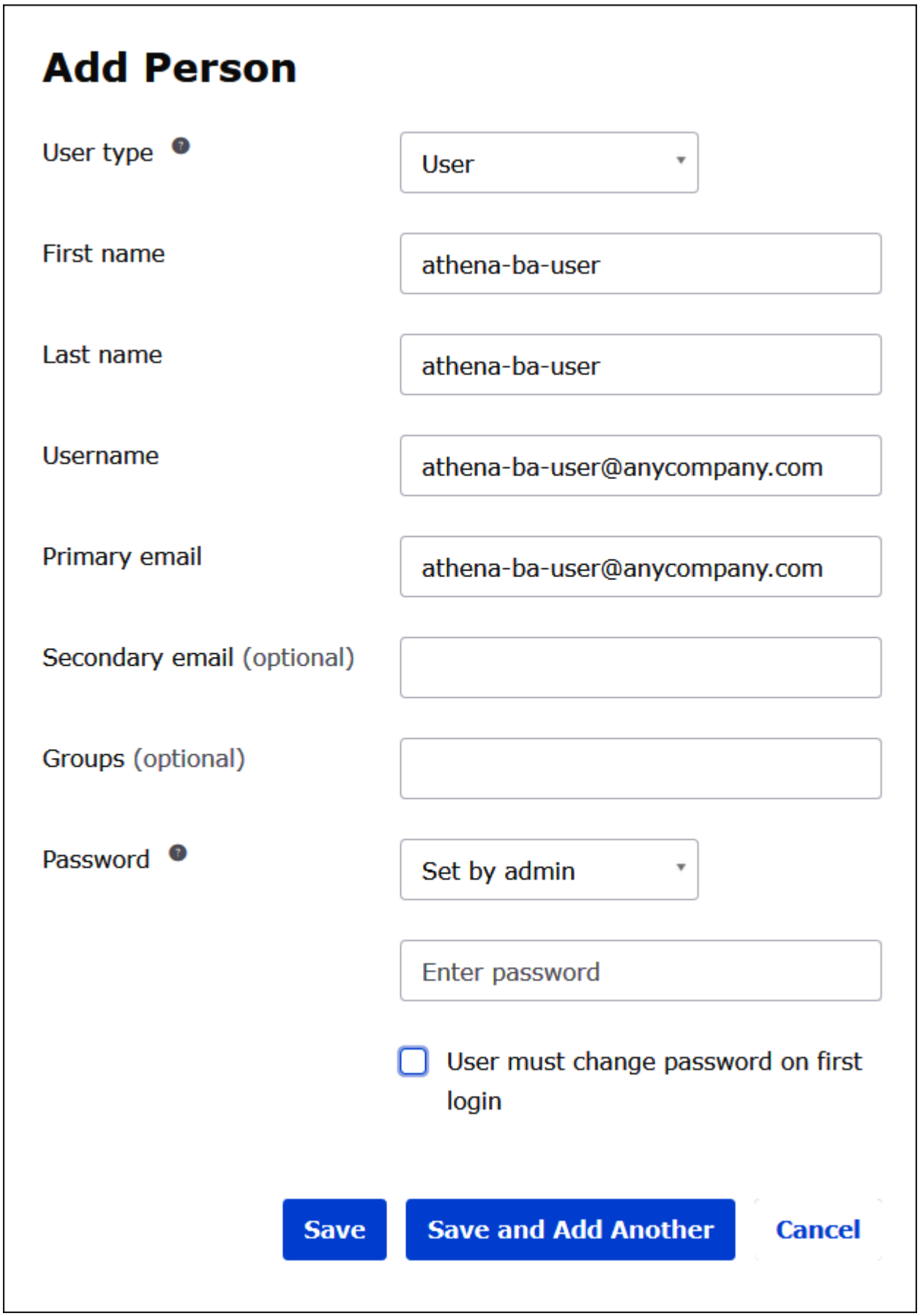


4. Geben Sie im Dialogfeld Person hinzufügen die erforderlichen Informationen ein.

   - Geben Sie die Werte für Vorname und Nachname ein. Dieses Tutorial verwendet *athena-okta-user*.
   - Geben Sie einen Benutzernamen und eine primäre E-Mail-Adresse ein. Dieses Tutorial verwendet *athena-okta-user@anycompany.com*.
   - Wählen Sie für Passwort die Option Von Administrator festgelegt aus und geben Sie dann ein Passwort ein. In diesem Tutorial wird die Option für Benutzer muss Kennwort bei der ersten Anmeldung ändern deaktiviert; Ihre Sicherheitsanforderungen können variieren.


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. Wählen Sie Save and Add Another (Speichern und weitere hinzufügen).
6. Geben Sie die Informationen für einen anderen Benutzer ein. In diesem Beispiel wird der Business-Analyst-Benutzer *athena-ba-user@anycompany.com* hinzugefügt.

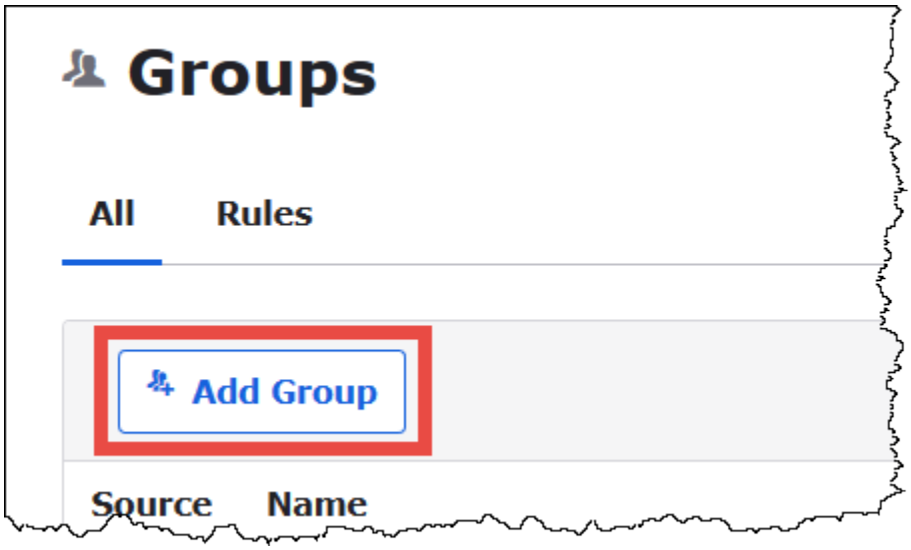

Add Person
User type
User
First name
athena-ba-user
Last name
athena-ba-user
Username
athena-ba-user@anycompany.com
Primary email
athena-ba-user@anycompany.com
Secondary email (optional)
Groups (optional)
Password
Set by admin
Enter password
User must change password on first login
Save
Save and Add Another
Cancel

7. Wählen Sie Save (Speichern) aus.

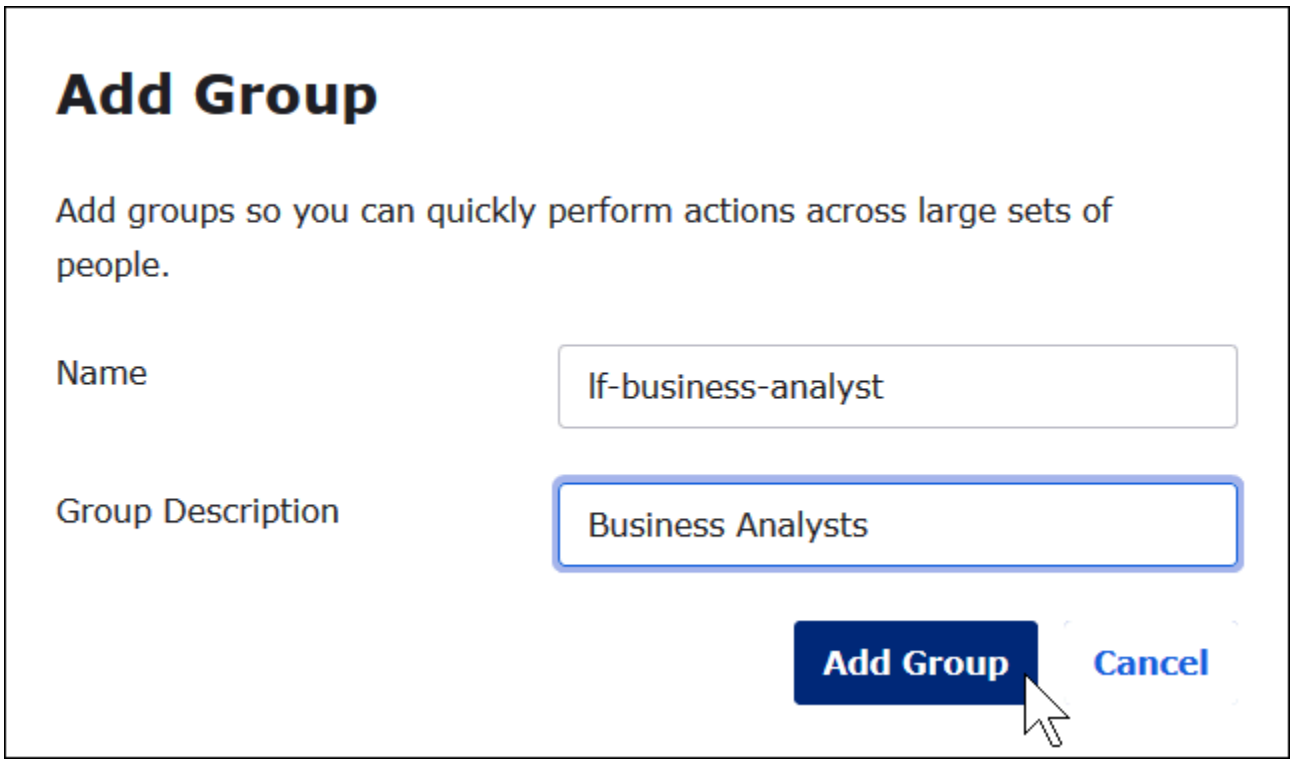
Im folgenden Verfahren gewähren Sie über den Athena-JDBC-Treiber Zugriff für zwei Okta-Gruppen, indem Sie eine Gruppe „Business Analysts“ und eine Gruppe „Entwickler“ hinzufügen.

So fügen Sie Okta-Gruppen hinzu

1. Wählen Sie im Okta-Navigationsbereich Verzeichnis und dann Gruppen aus.
2. Wählen Sie auf der Seite Gruppen die Option Gruppe hinzufügen aus.



3. Geben Sie im Dialogfeld Gruppe hinzufügen die erforderlichen Informationen ein.
   - Geben Sie für Name *`lf-business-analyst`* ein.
   - Geben Sie für Gruppenbeschreibung *`Business Analysts`* ein.

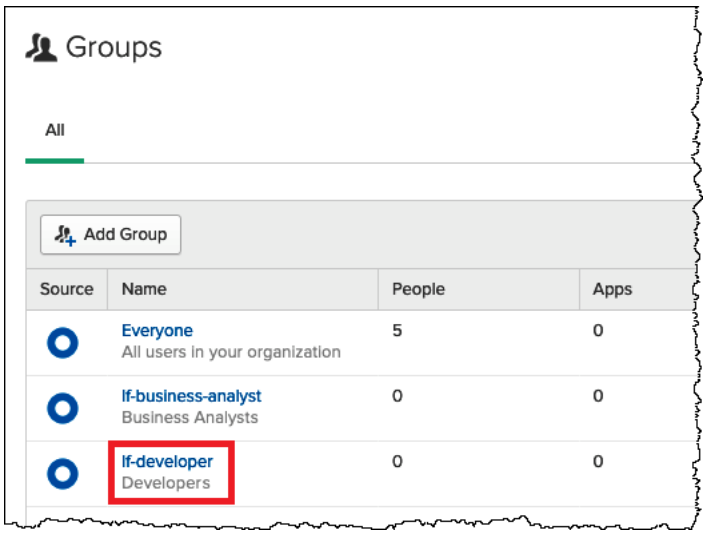


4. Wählen Sie Add Group (Gruppe hinzufügen).
5. Wählen Sie auf der Seite Gruppen erneut Gruppe hinzufügen aus. Dieses Mal geben Sie Informationen für die Entwicklergruppe ein.
6. Geben Sie die erforderlichen Informationen ein.
   - Geben Sie als Name *`lf-developer`* ein.
   - Geben Sie für Gruppenbeschreibung *`Entwickler`* ein.
7. Wählen Sie Add Group (Gruppe hinzufügen).

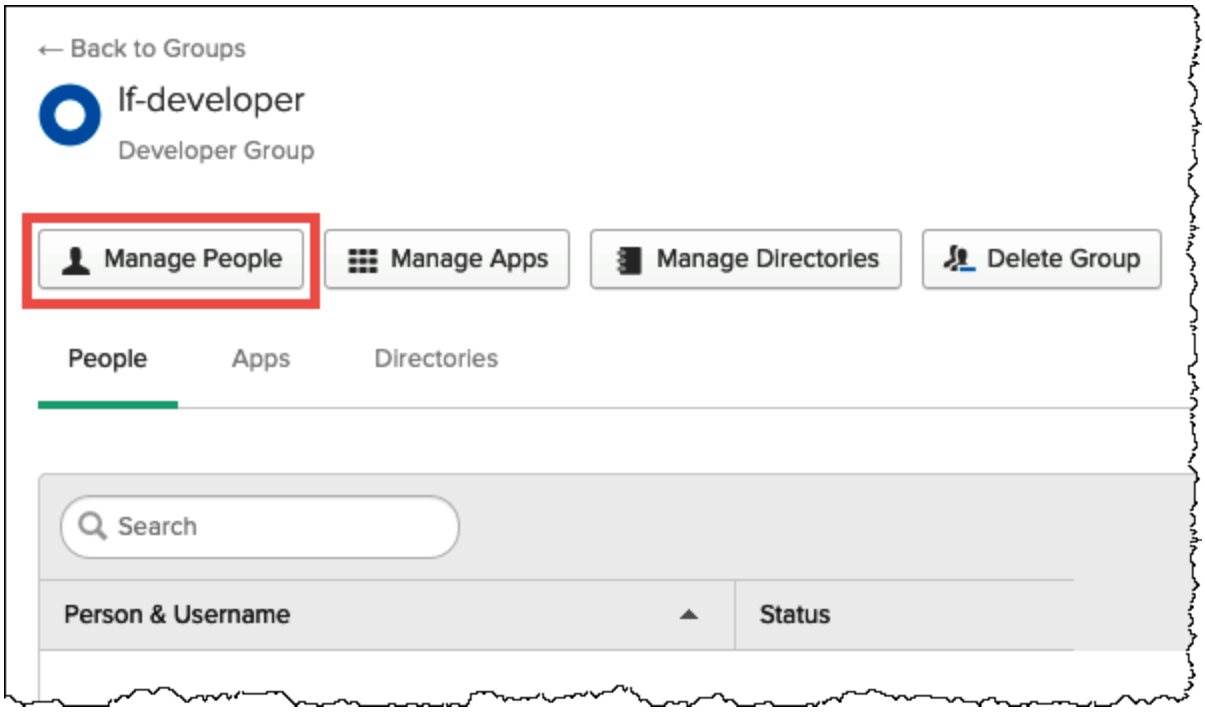
Nachdem Sie zwei Benutzer und zwei Gruppen haben, können Sie jeder Gruppe einen Benutzer hinzufügen.

So fügen Sie Benutzer Gruppen hinzu

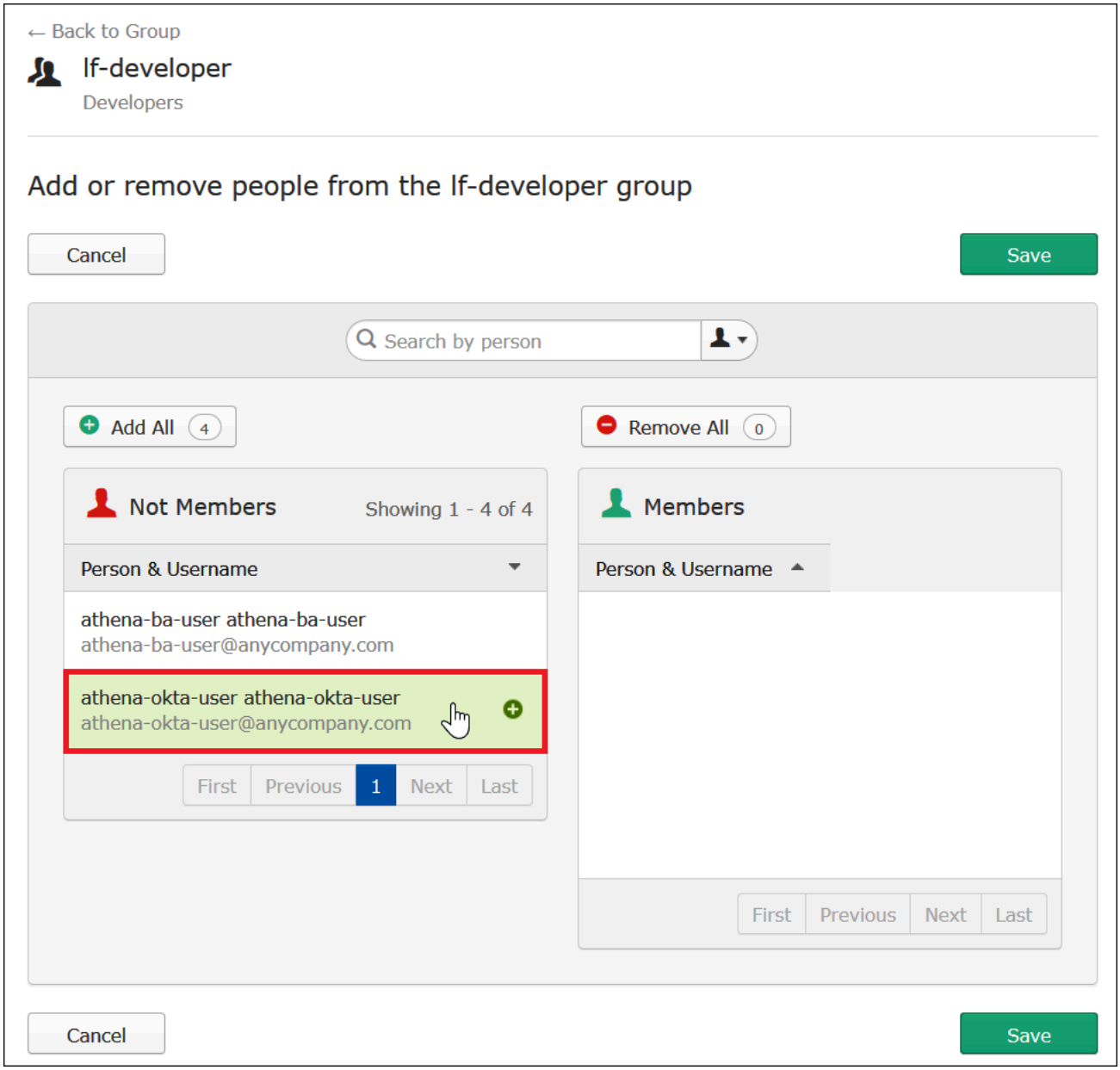
1. Wählen Sie auf der Seite Gruppen die soeben erstellte lf-developer-Gruppe aus. Sie fügen dieser Gruppe einen der Okta-Benutzer hinzu, die Sie als Entwickler erstellt haben.



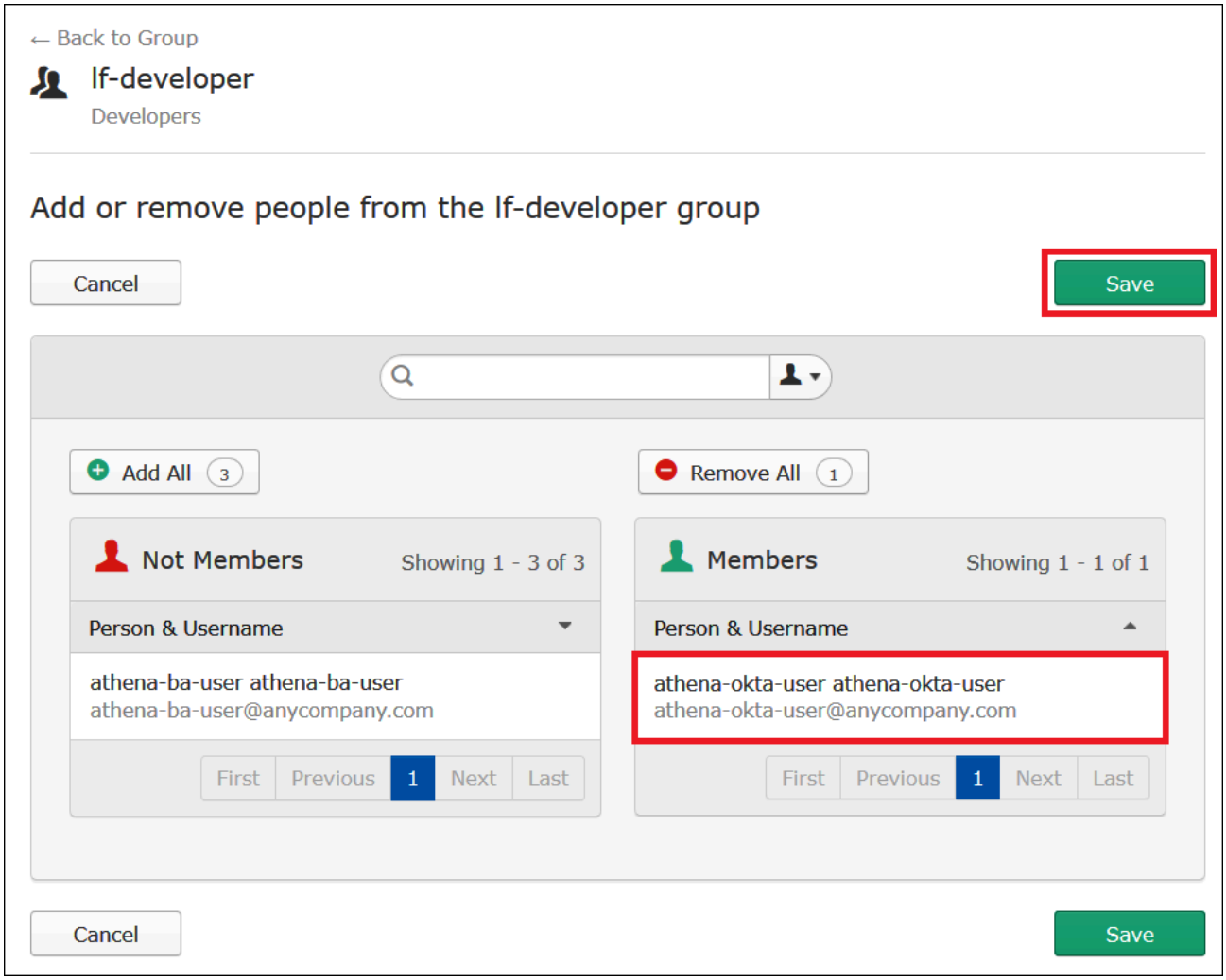
2. Wählen Sie Manage People (Verwalten von Personen).



3. Wählen Sie aus der Liste Keine Mitglieder die Option athena-okta-user aus.

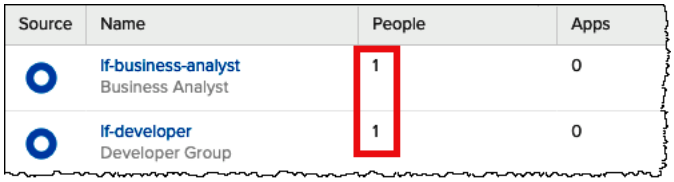


Der Eintrag für den Benutzer wird von der Liste Keine Mitglieder links in die Liste Mitglieder rechts verschoben.



4. Wählen Sie Save (Speichern) aus.
5. Wählen Sie Zurück zur Gruppe oder wählen Sie Verzeichnis und dann Gruppen aus.
6. Wählen Sie die Gruppe lf-business-analyst.
7. Wählen Sie Manage People (Verwalten von Personen).
8. Fügen Sie den athena-ba-user der Mitgliederliste der Gruppe lf-business-analyst hinzu und wählen Sie dann Speichern.
9. Wählen Sie Zurück zur Gruppe oder wählen Sie Verzeichnis, Gruppen.

   Auf der Seite Gruppen wird jetzt angezeigt, dass jede Gruppe einen Okta-Benutzer hat.



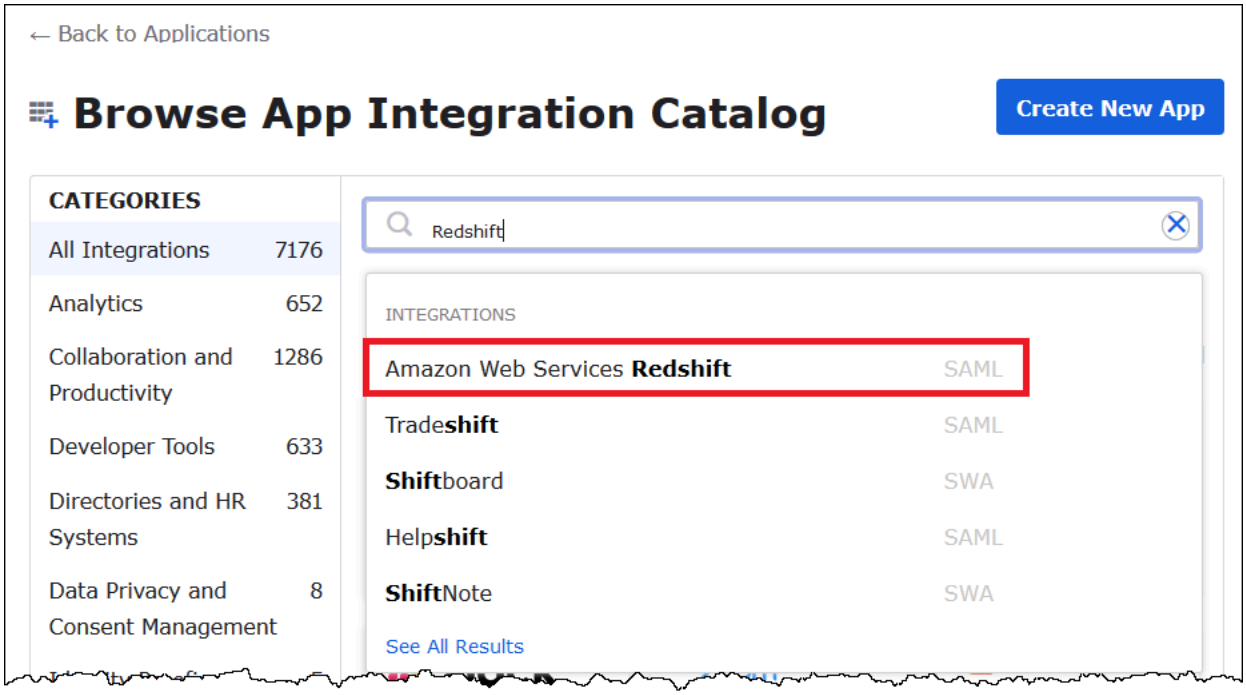
Schritt 3: Einrichten einer Okta-Anwendung für SAML-Authentifizierung

In diesem Schritt verwenden Sie die Okta-Entwicklerkonsole, um die folgenden Aufgaben auszuführen:

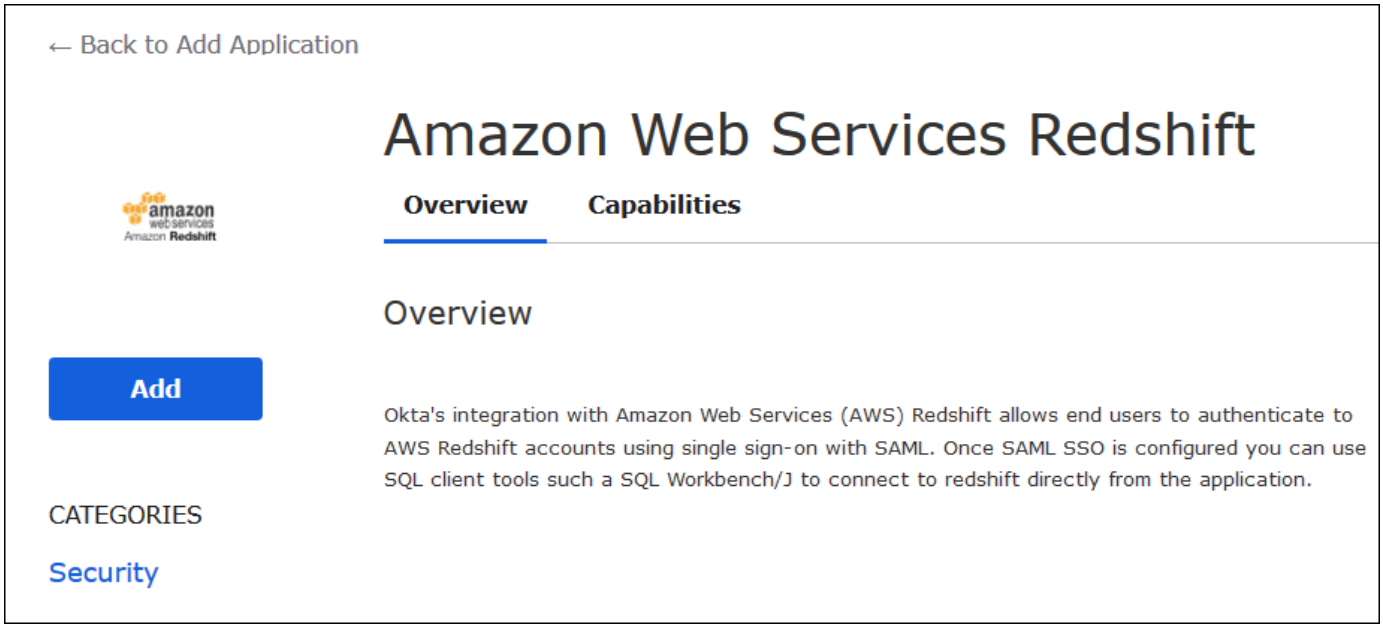
- Hinzufügen einer SAML-Anwendung zur Verwendung mit AWS.
- Weisen Sie die Anwendung dem Okta-Benutzer zu.
- Weisen Sie die Anwendung einer Okta-Gruppe zu.
- Laden Sie die resultierenden Metadaten des Identitätsanbieters zur späteren Verwendung mit AWS herunter.

So fügen Sie eine Anwendung für die SAML-Authentifizierung hinzu

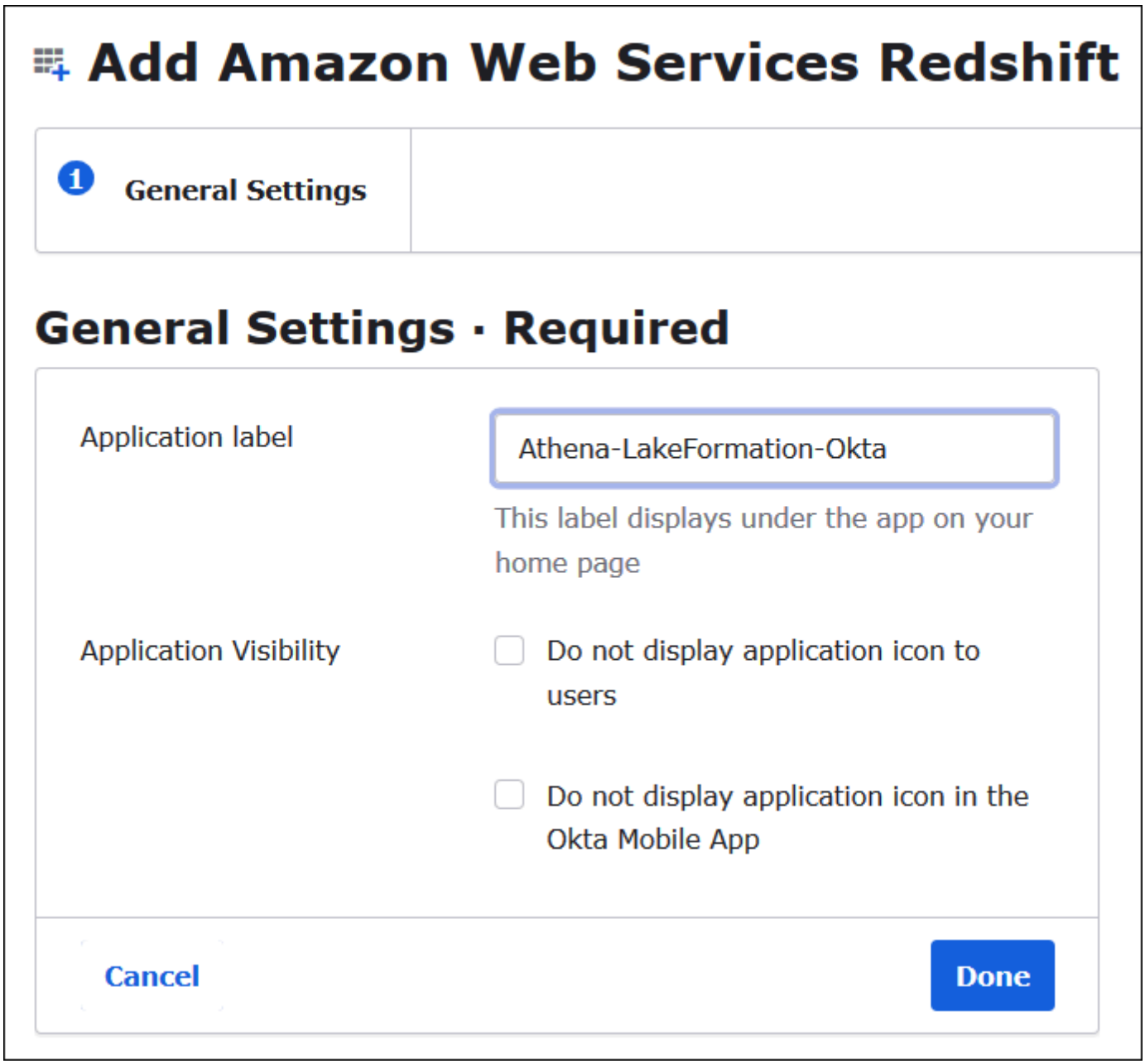
1. Wählen Sie im Okta-Navigationsbereich Anwendungen, Anwendungen, damit Sie eine Okta-Anwendung für die SAML-Authentifizierung bei Athena konfigurieren können.
2. Klicken Sie auf Browse App Catalog (Durchsuchen von App-Katalog).
3. Geben Sie in das Suchfeld ei **`Redshift`**.
4. Wählen Sie Amazon Web Services Redshift. Die Okta-Anwendung in diesem Tutorial verwendet die vorhandene SAML-Integration für Amazon Redshift.



5. Wählen Sie auf der Seite Amazon Web Services Redshift Hinzufügen aus, um eine SAML-basierte Anwendung für Amazon Redshift zu erstellen.



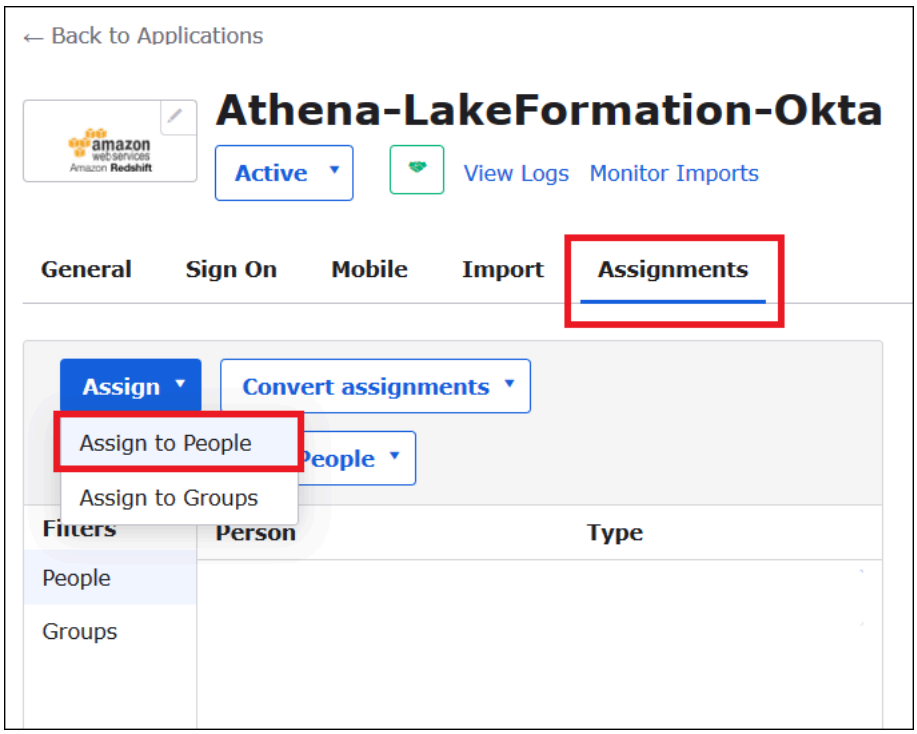
6. Geben Sie für Anwendungsmarkierung `Athena-LakeFormation-Okta` ein und wählen Sie dann Fertig aus.



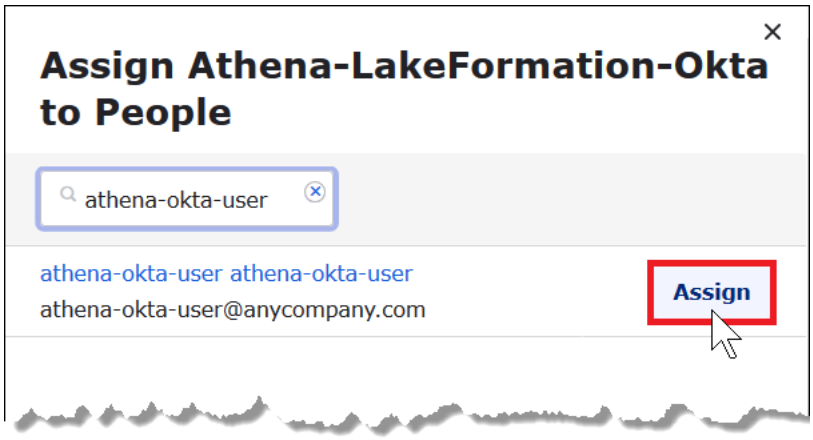
Nachdem Sie nun eine Okta-Anwendung erstellt haben, können Sie sie den von Ihnen erstellten Benutzern und Gruppen zuweisen.

So weisen Sie die Anwendung Benutzern und Gruppen zu

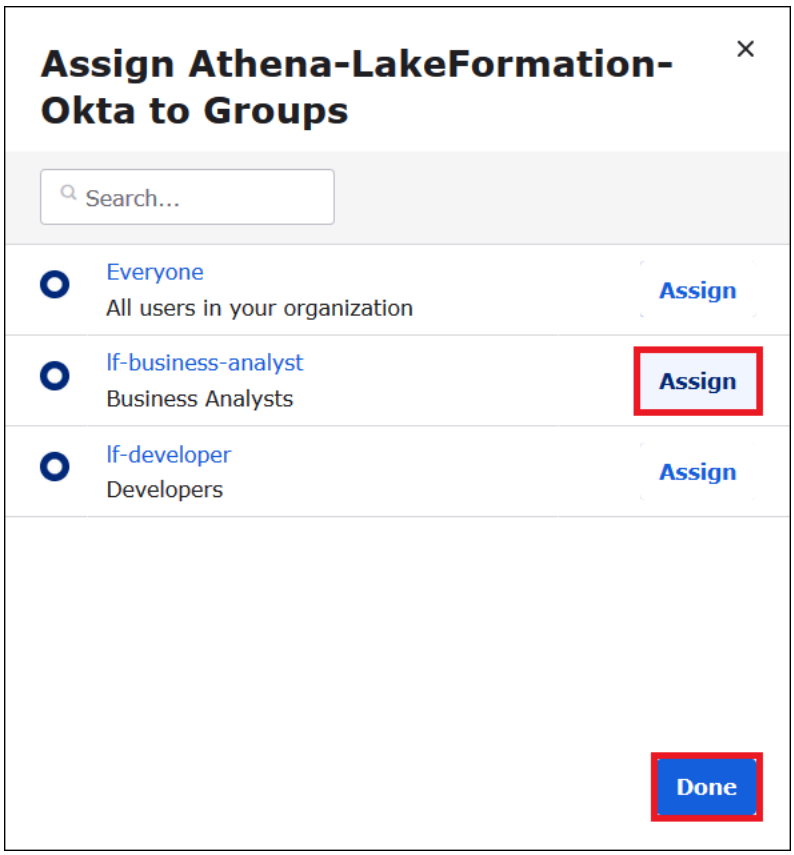
1. Wählen Sie auf der Seite Anwendungen die Anwendung Athena-LakeFormation-Okta aus.
2. Wählen Sie auf der Registerkarte Zuweisungen Zuweisen, Personen zuweisen.



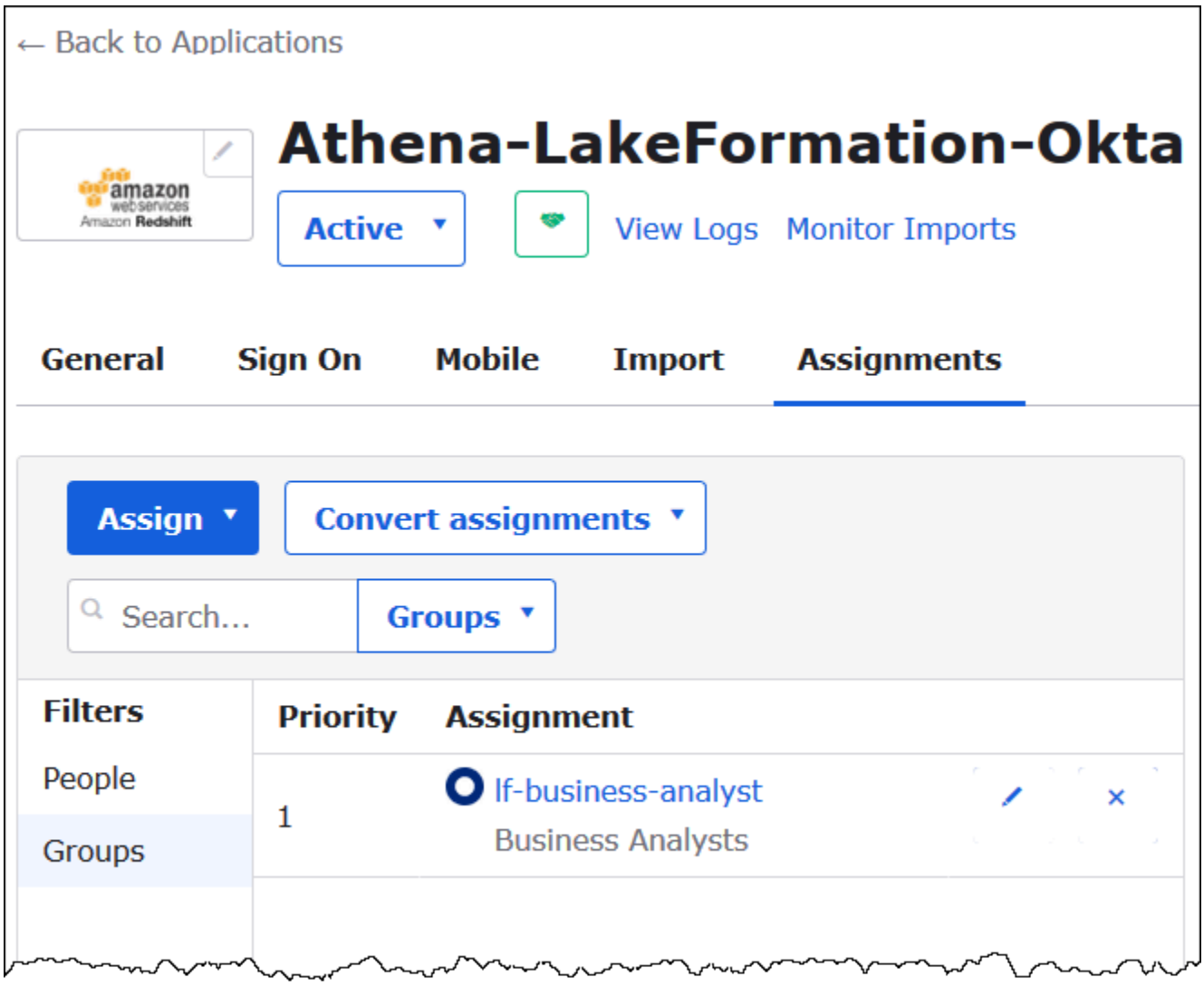
3. Suchen Sie im Dialogfeld Athena-LakeFormation-Okta zu Personen zuweisen den Benutzer athena-okta-user, den Sie zuvor erstellt haben.
4. Wählen Sie Zuordnen, um den Benutzer der Anwendung zuzuordnen.



5. Wählen Sie Save and Go Back (Speichern und zurückkehren).
6. Wählen Sie Done (Erledigt) aus.
7. Wählen Sie auf der Registerkarte Zuweisungen für die Anwendung Athena-LakeFormation-Okta die Option Zuweisen, Zu Gruppen zuweisen.
8. Wählen Sie für lf-business-analyst Zuweisen, um die Anwendung Athena-LakeFormation-Okta der Gruppe lf-business-analyst zuzuweisen und wählen Sie dann Fertig.



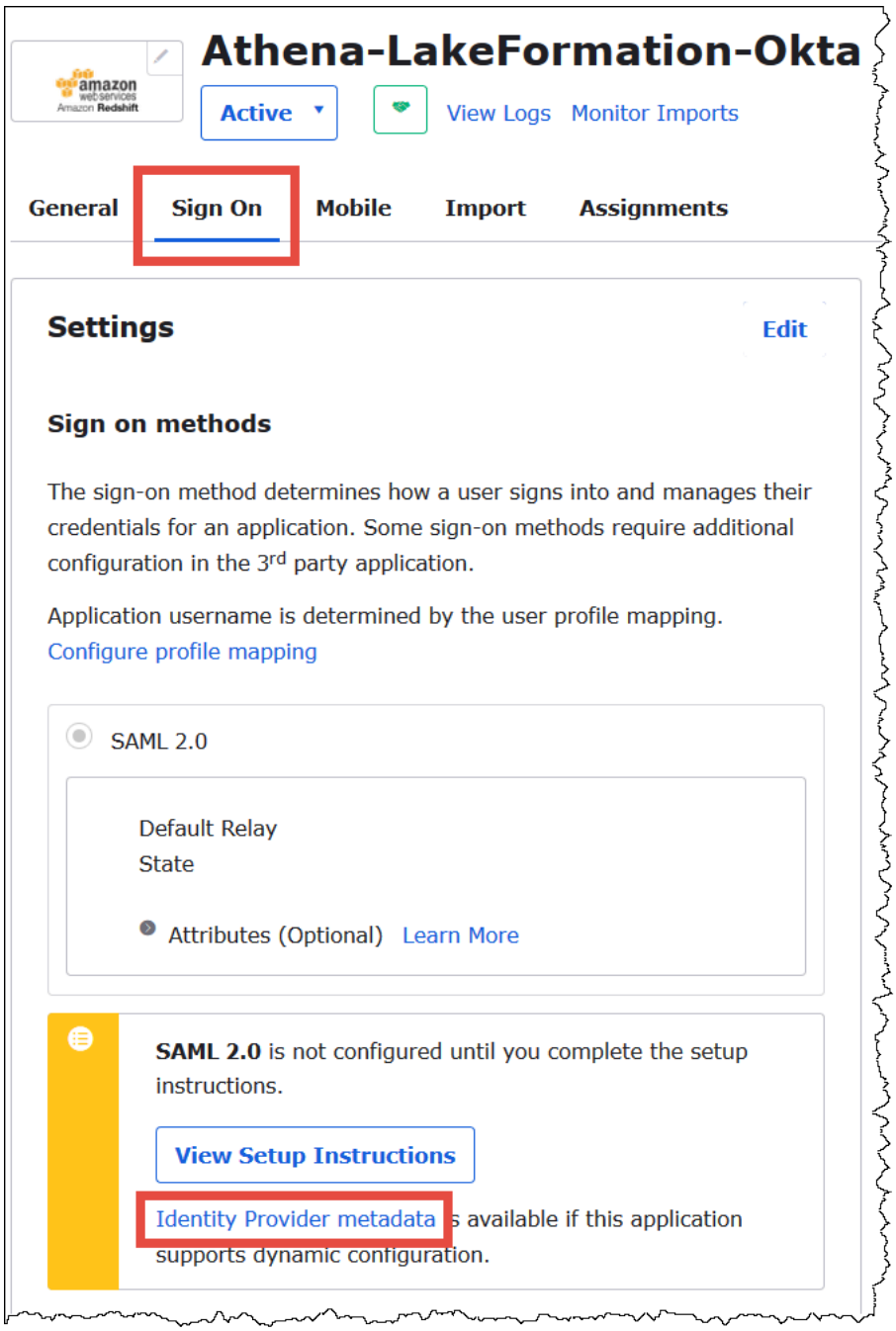
Die Gruppe wird in der Liste der Gruppen für die Anwendung angezeigt.



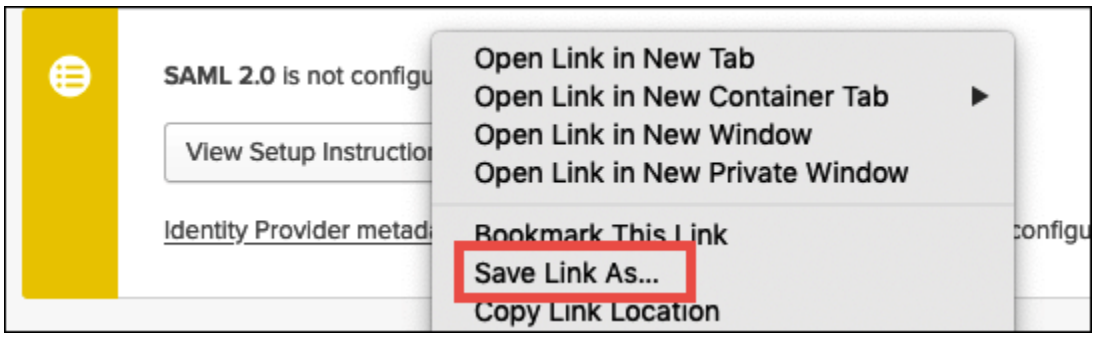
Jetzt können Sie die Metadaten der Identitätsanbieteranwendung für die Verwendung mit AWS herunterladen.

So laden Sie die Anwendungsmetadaten herunter

1. Wählen Sie die Registerkarte Sign On (Anmelden) der Okta-Anwendung und klicken Sie dann mit der rechten Maustaste auf Identity Provider metadata (Metadaten des Identitätsanbieters).



2. Wählen Sie Save Link As (Link speichern unter), um die Metadaten des Identitätsanbieters im XML-Format in einer Datei zu speichern. Geben Sie ihm einen Namen, den Sie kennen (z. B. `Athena-LakeFormation-idp-metadata.xml`).



Schritt 4: Erstellen eines AWS-SAML-Identitätsanbieters und einer IAM-Rolle für den Lake-Formation-Zugriff

In diesem Schritt verwenden Sie die AWS-Konsole für Identity and Access Management (IAM), um die folgenden Aufgaben auszuführen:

- Erstellen Sie einen Identitätsanbieter für AWS.
- Erstellen Sie eine IAM-Rolle für den Lake-Formation-Zugriff.
- Fügen Sie der Rolle die verwaltete AmazonAthenaFullAccess-Richtlinie hinzu.
- Fügen Sie der Rolle eine Richtlinie für Lake Formation und AWS Glue hinzu.
- Fügen Sie der Rolle eine Richtlinie für Athena-Abfrageergebnisse hinzu.

So erstellen Sie einen AWS-SAML-Identitätsanbieter

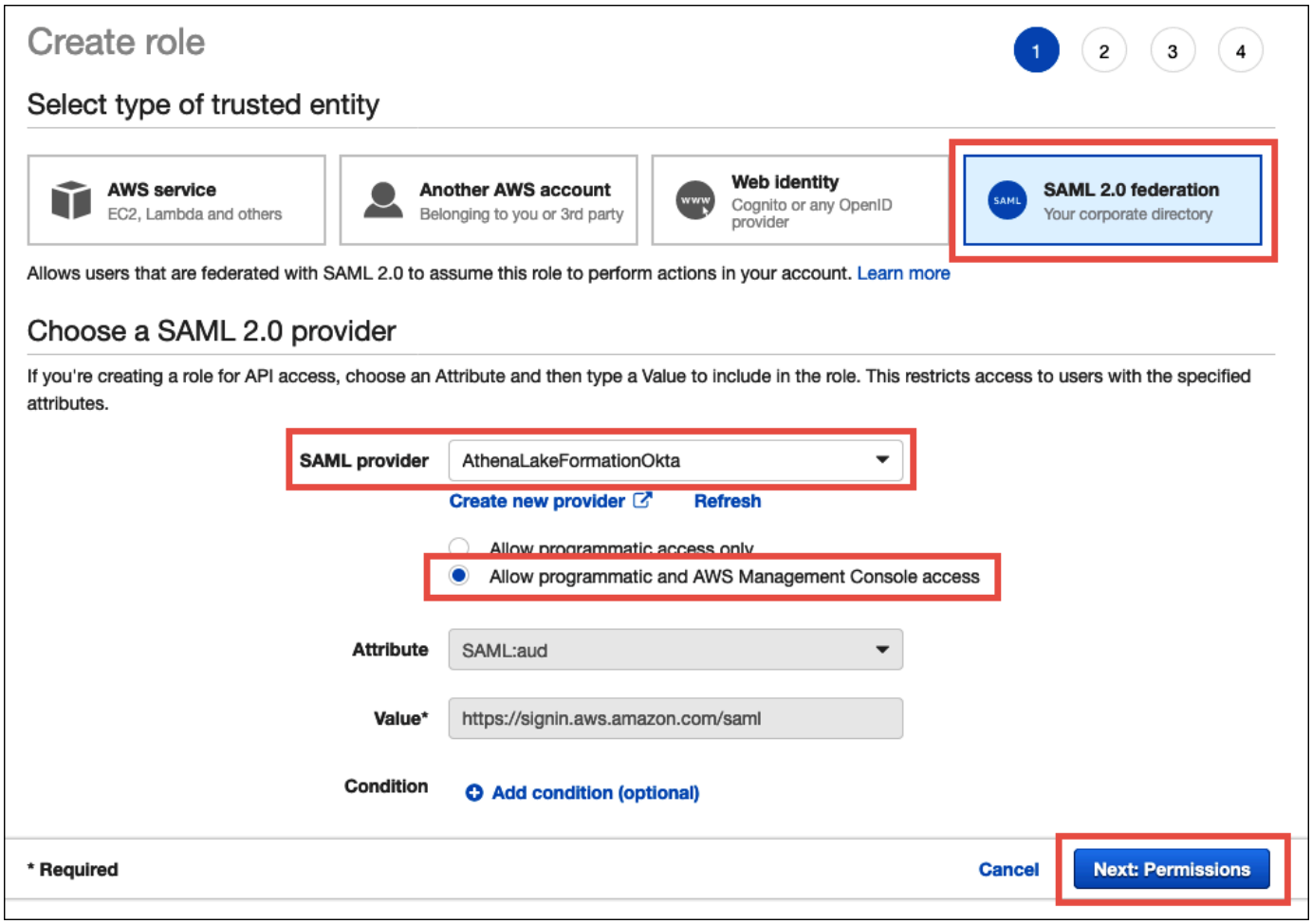
1. Melden Sie sich als Amazon-Web-Services-Kontoadministrator bei der Amazon-Web-Services-Kontokonsole an und navigieren Sie zur IAM-Konsole (https://console.aws.amazon.com/iam/).
2. Wählen Sie im Navigationsbereich Identitätsanbieter und dann Anbieter hinzufügen aus.
3. Geben Sie auf dem Bildschirm Anbieter konfigurieren die folgenden Informationen ein:
   - Wählen Sie als Anbietertyp SAML aus.
   - Geben Sie für Anbietername `AthenaLakeFormationOkta` ein.
   - Verwenden Sie für das Metadatendokument die Option Datei auswählen, um die heruntergeladene XML-Metadatendatei des Identitätsanbieters (IdP) hochzuladen.
4. Wählen Sie Add provider (Anbieter hinzufügen).

Als Nächstes erstellen Sie eine IAM-Rolle für den AWS Lake Formation-Zugriff. Sie fügen der Rolle zwei Inline-Richtlinien hinzu. Eine Richtlinie bietet Berechtigungen für den Zugriff auf Lake Formation

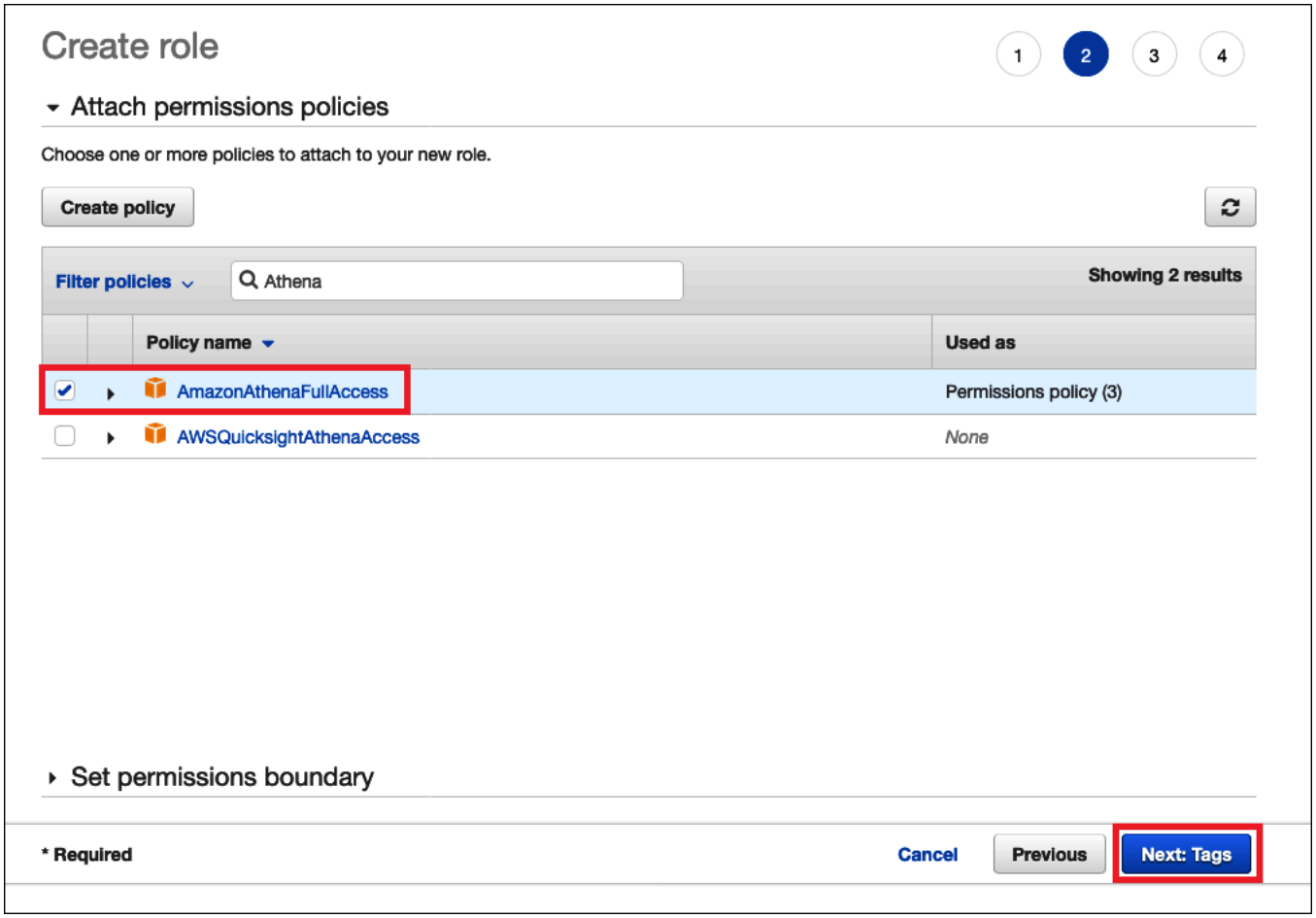
und die AWS Glue-APIs. Die andere Richtlinie bietet Zugriff auf Athena und den Speicherort der Athena-Abfrageergebnisse in Amazon S3.

So erstellen Sie eine IAM-Rolle zum Zugriff auf AWS Lake Formation

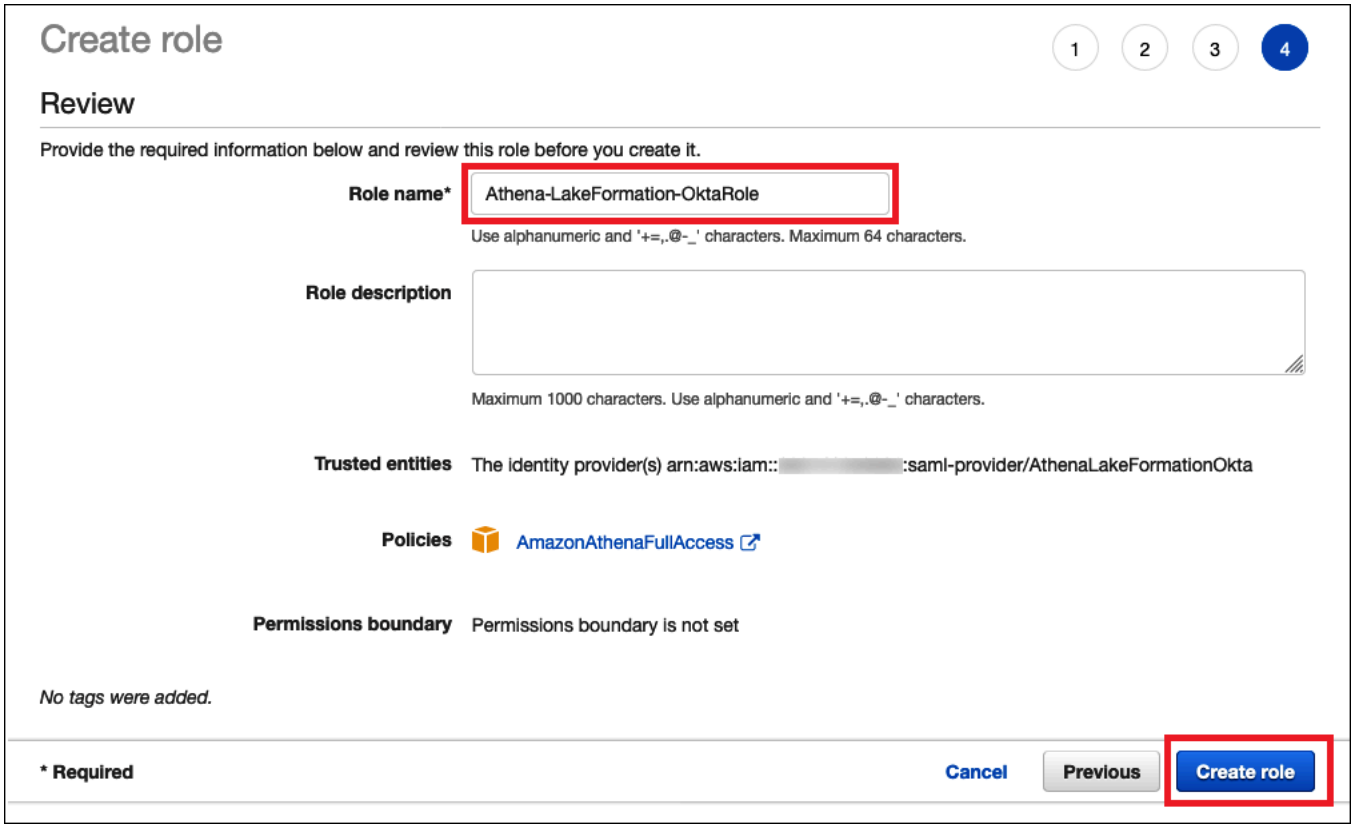
1. Wählen Sie im Navigationsbereich der IAM-Konsole Rollen und dann Rolle erstellen aus.
2. Führen Sie auf der Seite Rolle erstellen die folgenden Schritte aus:



   a. Wählen Sie für Typ der vertrauenswürdigen Entität auswählen die Option SAML-2.0-Verbund aus.
   b. Wählen Sie für den SAML-Anbieter AthenaLakeFormationOkta aus.
   c. Wählen Sie für SAML-Anbieter die Option Programmgesteuerten und AWS Management Console-Zugriff zulassen.
   d. Wählen Sie Next: Permissions (Weiter: Berechtigungen) aus.
3. Geben Sie auf der Seite Berechtigungsrichtlinien anhängen für Filterrichtlinien **`Athena`** ein.
4. Wählen Sie die verwaltete AmazonAthenaFullAccess-Richtlinie und dann Weiter: Tags aus.



5. Wählen Sie auf der Seite Add tags (Tags hinzufügen) die Option Next: Review (Weiter: Prüfen) aus.
6. Geben Sie auf der Seite Prüfen für Rollenname einen Namen für die Rolle ein (z. B. *`Athena-LakeFormation-OktaRole`*) und wählen Sie dann Rolle erstellen aus.



Als Nächstes fügen Sie Inline-Richtlinien hinzu, die den Zugriff auf Lake Formation, AWS Glue APIs und Athena-Abfrageergebnisse in Amazon S3 ermöglichen.

Wenn Sie IAM-Richtlinien verwenden, stellen Sie sicher, dass Sie die bewährten Methoden von IAM befolgen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

So fügen Sie der Rolle für Lake Formation und AWS Glue eine Inline-Richtlinie hinzu

1. Wählen Sie aus der Liste der Rollen in der IAM-Konsole die neu erstellte `Athena-LakeFormation-OktaRole`.
2. Wählen Sie auf der Seite Zusammenfassung für die Rolle auf der Registerkarte Berechtigungen die Option Inline-Richtlinie hinzufügen aus.
3. Wählen Sie auf der Seite Create policy (Richtlinie erstellen) die Option JSON aus.
4. Fügen Sie eine Inline-Richtlinie wie die folgende hinzu, die Zugriff auf Lake Formation und die AWS Glue-APIs bietet.

```
{
```

```
    "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. Wählen Sie Review policy (Richtlinie prüfen).
6. Geben Sie unter Name einen Namen für die Richtlinie ein (z. B. **`LakeFormationGlueInlinePolicy`**).
7. Wählen Sie Create Policy (Richtlinie erstellen) aus.

So fügen Sie der Rolle für den Speicherort der Athena Abfrageergebnisse eine Inline-Richtlinie hinzu

1. Wählen Sie auf der Seite Zusammenfassung für die `Athena-LakeFormation-OktaRole`-Rolle auf der Registerkarte Berechtigungen die Option Inline-Richtlinie hinzufügen aus.
2. Wählen Sie auf der Seite Create policy (Richtlinie erstellen) die Option JSON aus.
3. Fügen Sie eine Inline-Richtlinie wie die folgende hinzu, die der Rolle Zugriff auf den Speicherort der Athena-Abfrageergebnisse ermöglicht. Ersetzen Sie die Platzhalter *`<athena-query-results-bucket>`* im Beispiel durch den Namen Ihres Amazon-S3-Buckets.

```
{
    "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. Wählen Sie Review policy (Richtlinie prüfen).
5. Geben Sie unter Name einen Namen für die Richtlinie ein (z. B. **`AthenaQueryResultsInlinePolicy`**).
6. Wählen Sie Create Policy (Richtlinie erstellen) aus.

Als Nächstes kopieren Sie den ARN der Lake Formation-Zugriffsrolle und den ARN des von Ihnen erstellten SAML-Anbieters. Diese sind erforderlich, wenn Sie die Okta-SAML-Anwendung im nächsten Abschnitt des Lernprogramms konfigurieren.

So kopieren Sie den Rollen-ARN und den SAML-Identitätsanbieter-ARN

1. Wählen Sie in der IAM-Konsole auf der Seite Zusammenfassung für die `Athena-LakeFormation-OktaRole`-Rolle das Symbol In Zwischenablage kopieren neben Rollen-ARN aus. Der ARN hat das folgende Format:

   ```
   arn:aws:iam::<account-id>:role/Athena-LakeFormation-OktaRole
   ```

2. Speichern Sie den vollständigen ARN sicher für einen späteren Verweis.
3. Wählen Sie im Navigationsbereich der IAM-Konsole Identitätsanbieter aus.
4. Wählen Sie den AthenaLakeFormationOkta-Anbieter aus.
5. Wählen Sie auf der Seite Zusammenfassung neben Anbieter-ARN das Symbol In Zwischenablage kopieren aus. Der ARN sollte wie folgt aussehen:

   ```
   arn:aws:iam::<account-id>:saml-provider/AthenaLakeFormationOkta
   ```

6. Speichern Sie den vollständigen ARN sicher für einen späteren Verweis.

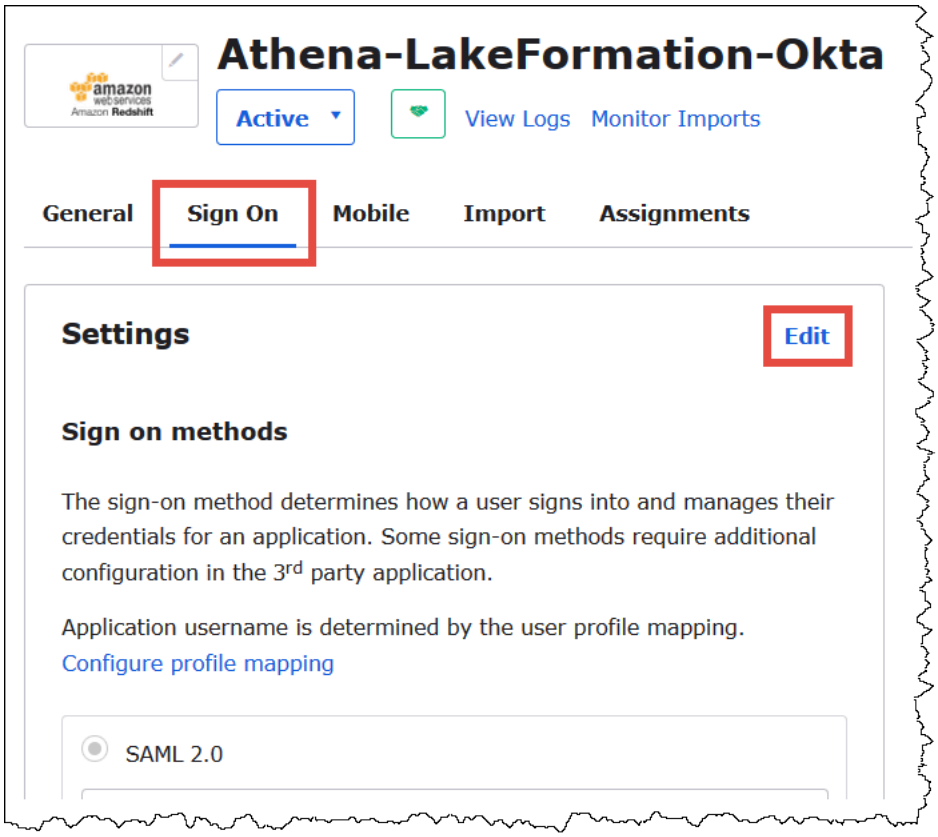
Schritt 5: IAM-Rolle und SAML-Identitätsanbieter zur Okta-Anwendung hinzufügen

In diesem Schritt kehren Sie zur Okta-Entwicklerkonsole zurück und führen die folgenden Aufgaben aus:

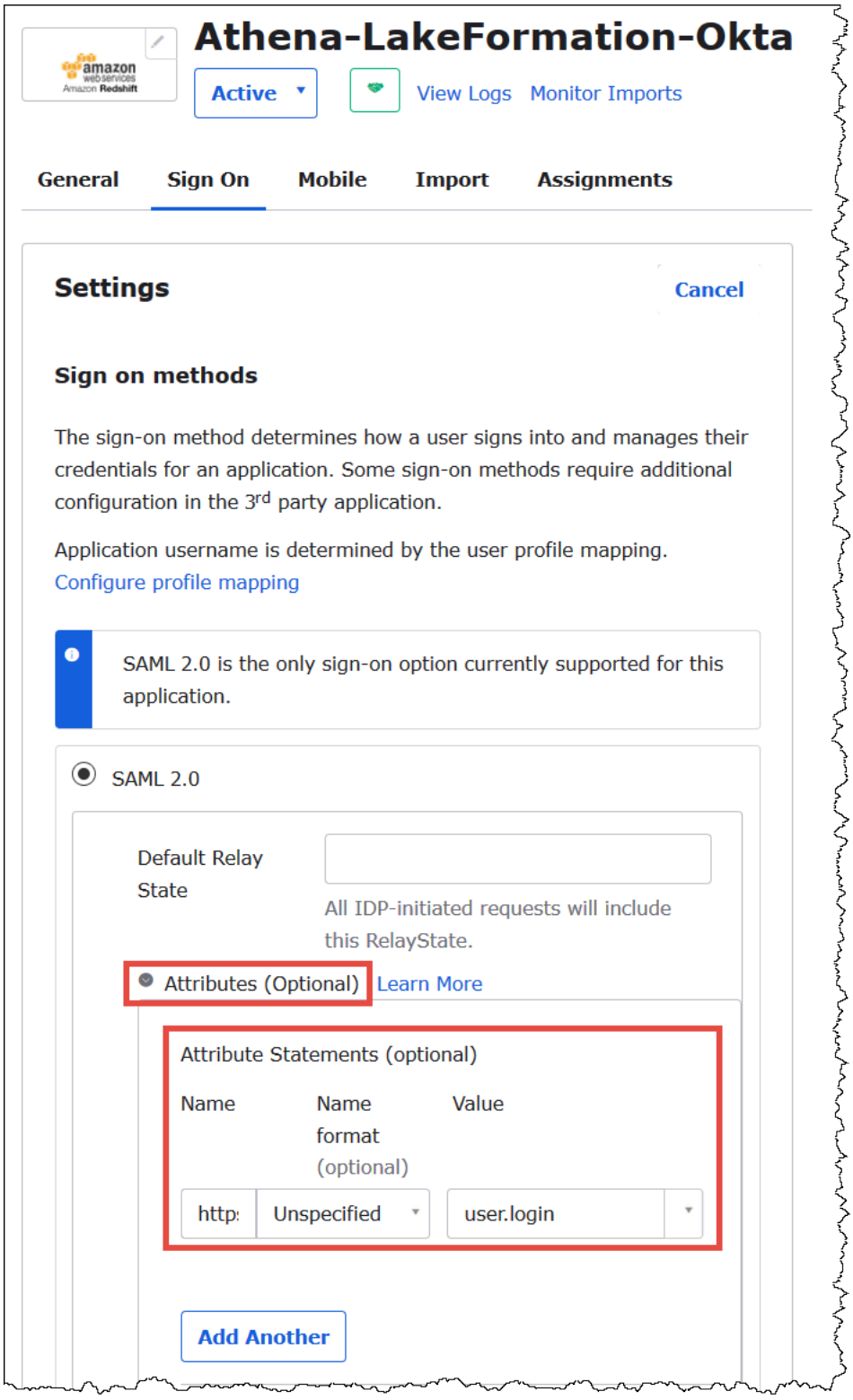
- Fügen Sie der Okta-Anwendung Benutzer- und Gruppen-Lake-Formation-URL-Attribute hinzu.
- Fügen Sie der Okta-Anwendung den ARN für den Identitätsanbieter und den ARN für die IAM-Rolle hinzu.
- Kopieren Sie die Okta-Anwendungs-ID. Die Okta-Anwendungs-ID ist im JDBC-Profil erforderlich, das eine Verbindung zu Athena herstellt.

So fügen Sie der Okta-Anwendung Benutzer- und Gruppen-URL-Attribute für Lake Formation hinzu

1. Melden Sie sich bei der Okta-Entwicklerkonsole an.
2. Wählen Sie die Registerkarte Anwendungen und dann die `Athena-LakeFormation-Okta`-Anwendung aus.
3. Wählen Sie auf der Registerkarte Sign On (Anmelden) für die Anwendung und dann Edit (Bearbeiten).



4. Wählen Sie Attribute (optional), um es zu erweitern.

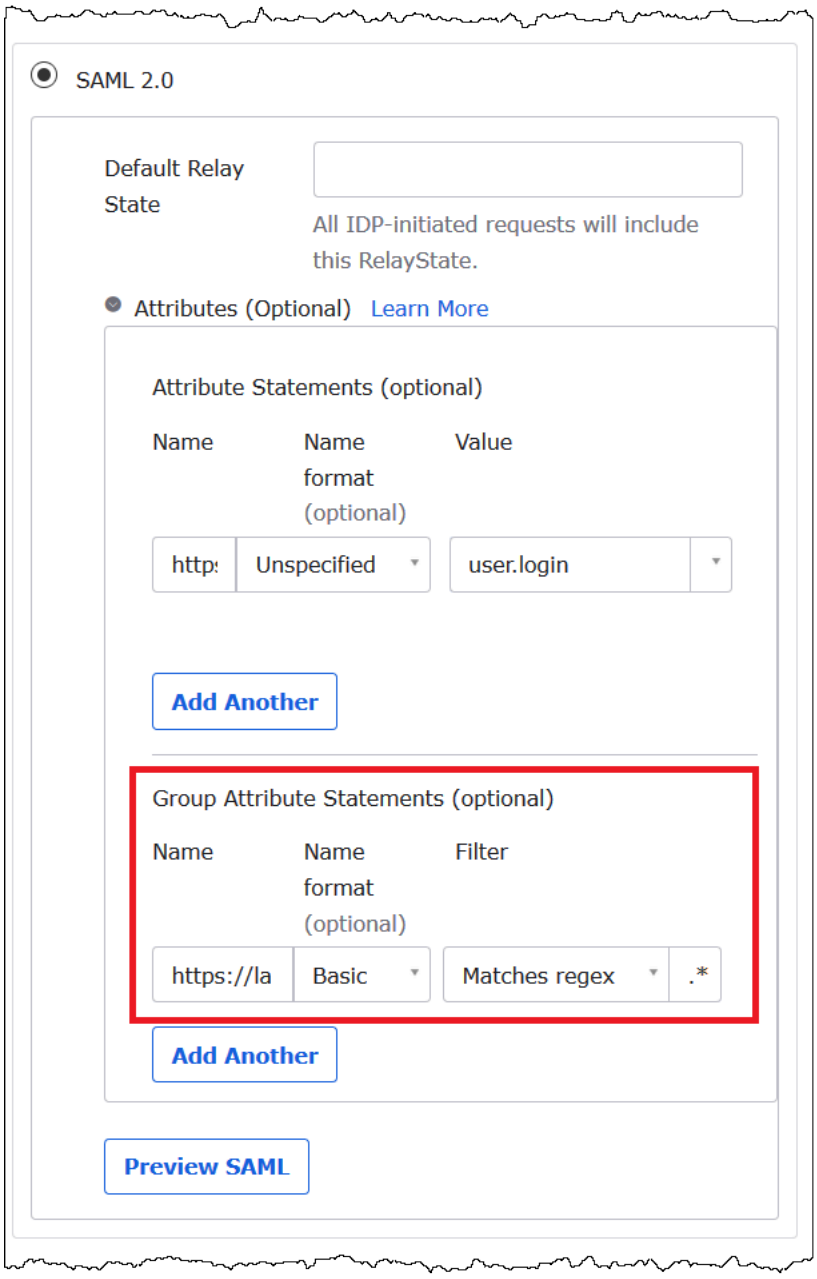


5. Fügen Sie für Attributanweisungen (optional) das folgende Attribut hinzu:

- Geben Sie unter Name **`https://lakeformation.amazon.com/SAML/Attributes/Username`** ein.
- Geben Sie für Wert **`user.login`** ein

6. Fügen Sie unter Gruppenattributanweisungen (optional) das folgende Attribut hinzu:

   - Geben Sie unter Name **`https://lakeformation.amazon.com/SAML/Attributes/Groups`** ein.
   - Geben Sie für Namensformat **`Basic`** ein
   - Wählen Sie für Filter die Option Entspricht Regex und geben Sie dann **`.*`** in das Filterfeld ein.

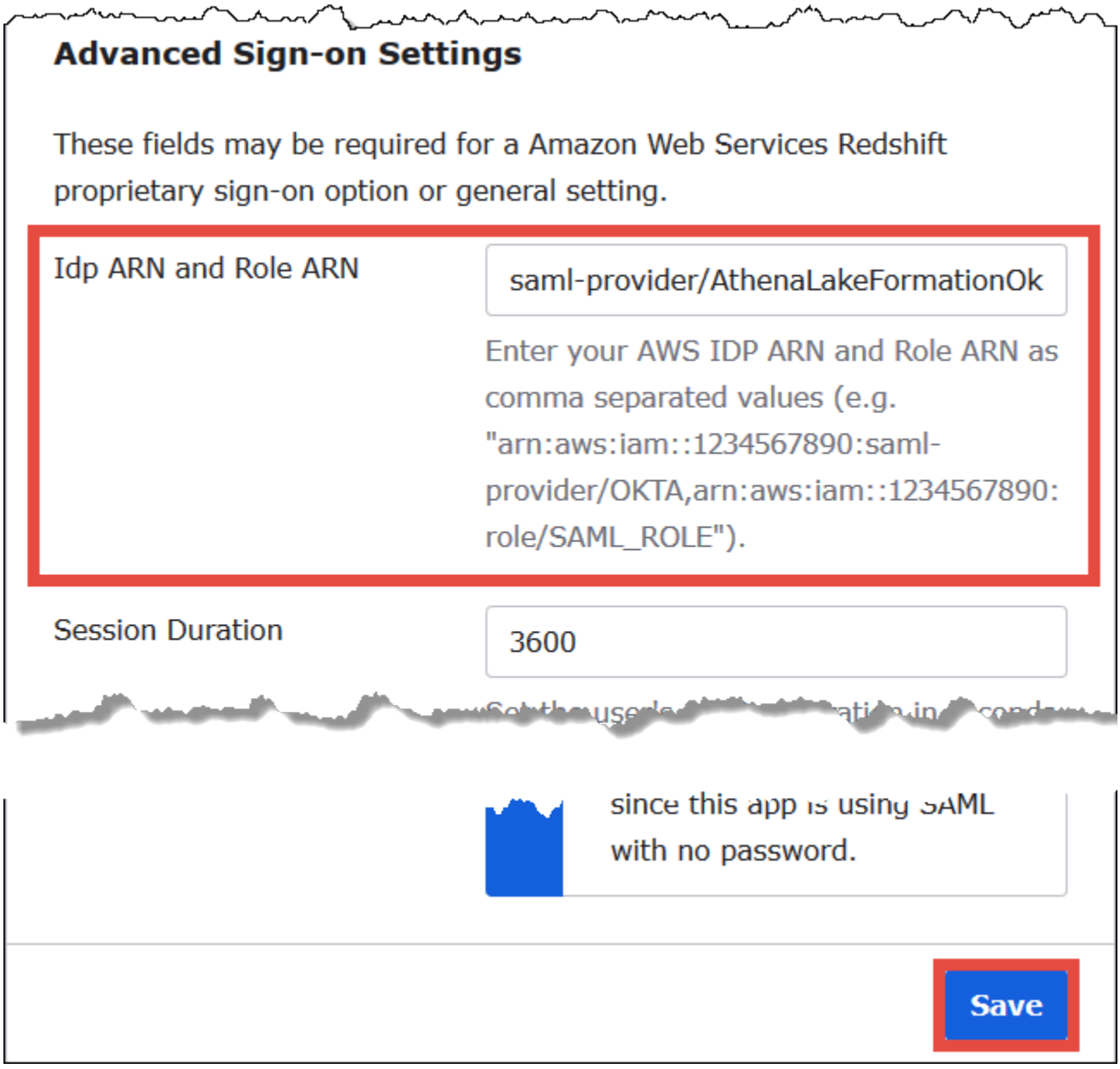


7. Scrollen Sie nach unten zum Abschnitt Erweiterte Anmeldeeinstellungen, in dem Sie den Identitätsanbieter und die IAM-Rollen-ARNs zur Okta-Anwendung hinzufügen.

So fügen Sie der Okta-Anwendung die ARNs für den Identitätsanbieter und die IAM-Rolle hinzu

1. Geben Sie für Idp-ARN und Rollen-ARN den AWS-Identity-Anbieter-ARN und den Rollen-ARN als kommagetrennte Werte im Format *<saml-arn>*,*<role-arn>* ein. Die kombinierte Zeichenfolge sollte wie folgt aussehen:

```
arn:aws:iam::<account-id>:saml-provider/
AthenaLakeFormationOkta,arn:aws:iam::<account-id>:role/Athena-LakeFormation-
OktaRole
```

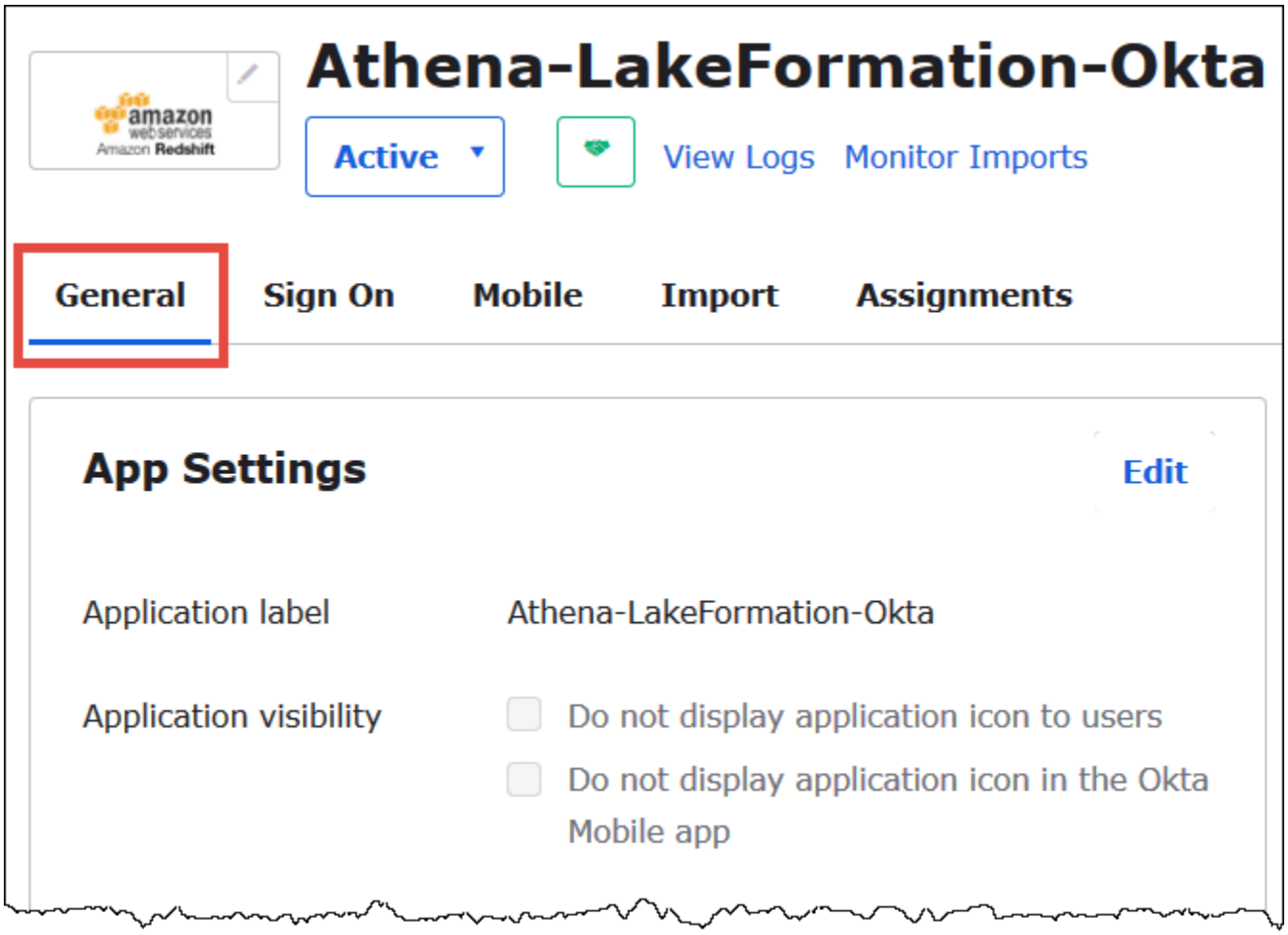


2. Wählen Sie Save (Speichern) aus.

Als Nächstes kopieren Sie die Okta-Anwendungs-ID. Sie benötigen dies später für die JDBC-Zeichenfolge, die sich mit Athena verbindet.

So suchen und kopieren Sie die Okta-Anwendung-ID

1. Wählen Sie das Symbol Allgemein der Okta-Anwendung.



2. Scrollen Sie nach unten zum Abschnitt App-Einbettungslink.
3. Kopieren Sie von Embed Link (Link einbetten) aus den Okta-Anwendungs-ID-Teil der URL und speichern Sie ihn sicher. Die Okta-Anwendungs-ID ist der Teil der URL nach `amazon_aws_redshift/`, aber vor dem nächsten Schrägstrich. Wenn die URL beispielsweise `amazon_aws_redshift/aaa/bbb` enthält, lautet die Anwendungs-ID `aaa`.

> **Note**
>
> Sie können den Einbettungslink nicht verwenden, um sich direkt bei der Athena-Konsole anzumelden, um Datenbanken anzuzeigen. Die Lake-Formation-Berechtigungen für SAML-Benutzer und -Gruppen werden nur erkannt, wenn Sie den JDBC- oder ODBC-Treiber verwenden, um Anfragen an Athena zu senden. Um die Datenbanken anzuzeigen, können Sie das Tool SQL Workbench/J verwenden, das über den JDBC-Treiber eine Verbindung mit Athena herstellt. Das Tool SQL Workbench/J wird in Schritt 7: Überprüfen des Zugriffs über den Athena-JDBC-Client erläutert.

Schritt 6: Erteilen von Benutzer- und Gruppenberechtigungen durch AWS Lake Formation

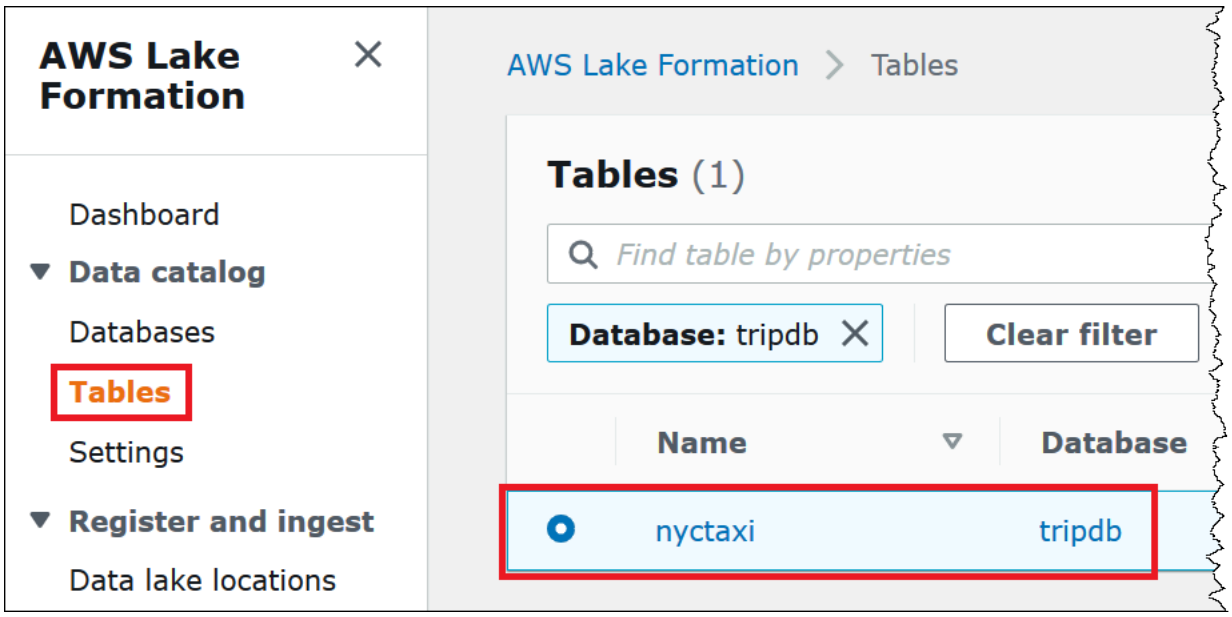
In diesem Schritt verwenden Sie die Lake-Formation-Konsole, um dem SAML-Benutzer und der SAML-Gruppe Berechtigungen für eine Tabelle zu erteilen. Sie können folgende Aufgaben ausführen:

- Geben Sie den ARN des Okta SAML-Benutzers und die zugehörigen Benutzerberechtigungen für die Tabelle an.
- Geben Sie den ARN der Okta SAML-Gruppe und die zugehörigen Gruppenberechtigungen für die Tabelle an.

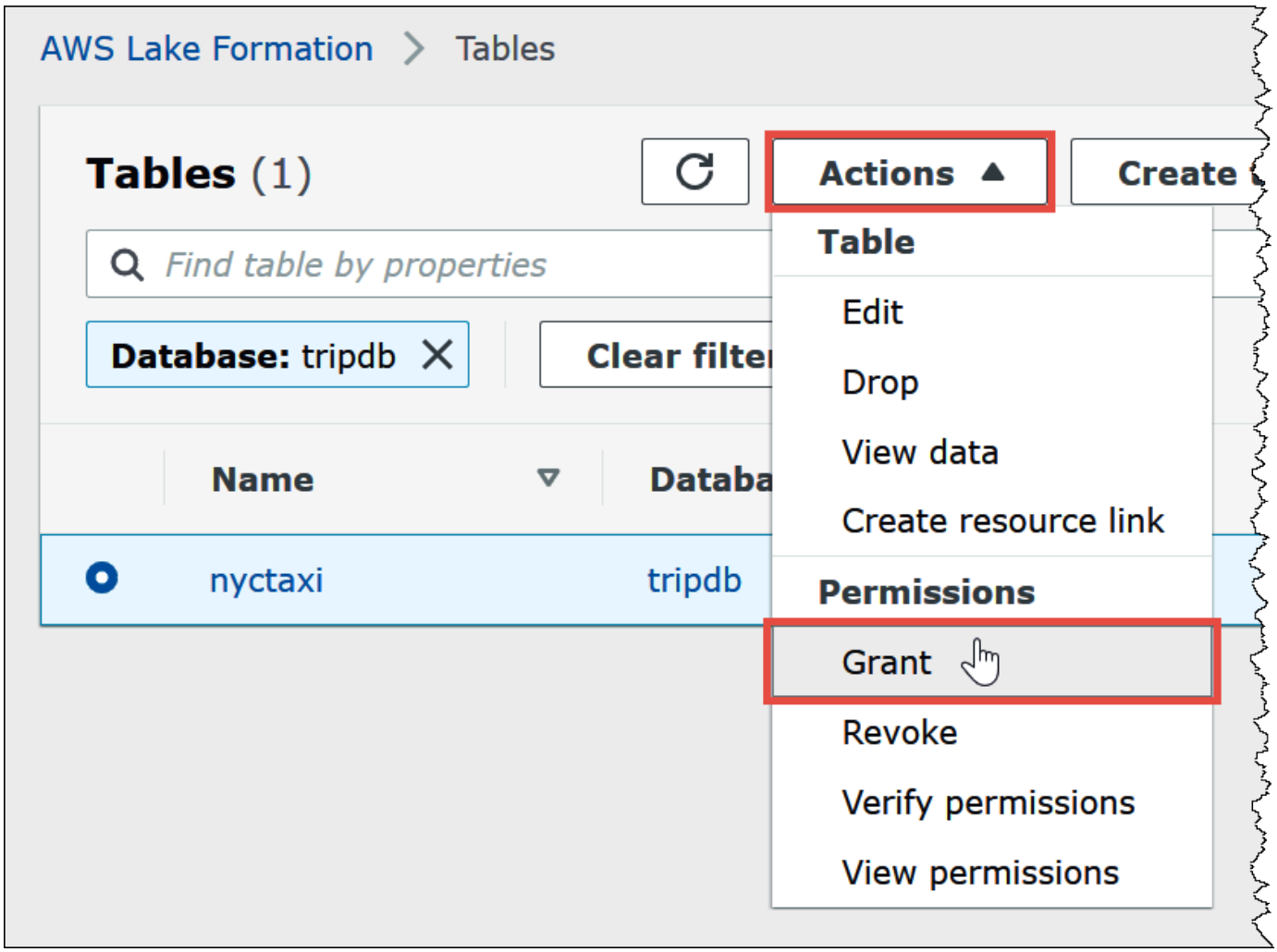
- Überprüfen Sie die Berechtigungen, die Sie erteilt haben.

So erteilen Sie dem Okta-Benutzer Berechtigungen in Lake Formation

1. Melden Sie sich als Data-Lake-Administrator bei der AWS Management Console an.
2. Öffnen Sie die Lake-Formation-Konsole unter https://console.aws.amazon.com/lakeformation/.
3. Wählen Sie im Navigationsbereich Tabellen aus, und wählen Sie dann die Tabelle aus, für die Sie Berechtigungen erteilen möchten. Dieses Tutorial verwendet die `nyctaxi`-Tabelle aus der `tripdb`-Datenbank.



4. Wählen Sie unter Aktionen die Option Gewähren aus.



5. Geben Sie im Dialogfeld Berechtigungen gewähren die folgenden Informationen ein:

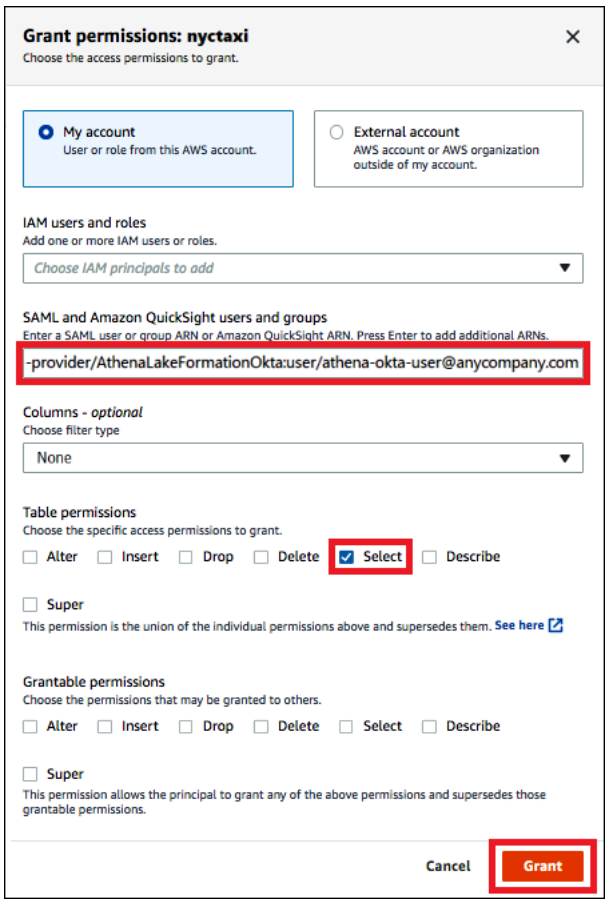
   a. Geben Sie unter SAML- und Amazon-QuickSight-Benutzer und -Gruppen den Okta-SAML-Benutzer-ARN im folgenden Format ein:

   ```
   arn:aws:iam::<account-id>:saml-provider/AthenaLakeFormationOkta:user/<athena-okta-user>@<anycompany.com>
   ```

   b. Wählen Sie für Spalten für Filtertyp auswählen und optional Spalten einschließen oder Spalten ausschließen aus.

   c. Verwenden Sie das Dropdown-Menü Eine oder mehrere Spalten auswählen unter dem Filter, um die Spalten anzugeben, die Sie für den Benutzer ein- oder ausschließen möchten.

   d. Wählen Sie für Tabellenberechtigungen die Option Auswählen aus. Dieses Tutorial gewährt nur die SELECT-Berechtigung; Ihre Anforderungen können variieren.



6. Wählen Sie Gewähren.

Jetzt führen Sie ähnliche Schritte für die Okta-Gruppe aus.

So erteilen Sie Berechtigungen in Lake Formation für die Okta-Gruppe

1. Stellen Sie auf der Seite Tabellen der Lake-Formation-Konsole sicher, dass die nyctaxi-Tabelle noch ausgewählt ist.
2. Wählen Sie unter Aktionen die Option Gewähren aus.
3. Geben Sie im Dialogfeld Berechtigungen gewähren die folgenden Informationen ein:

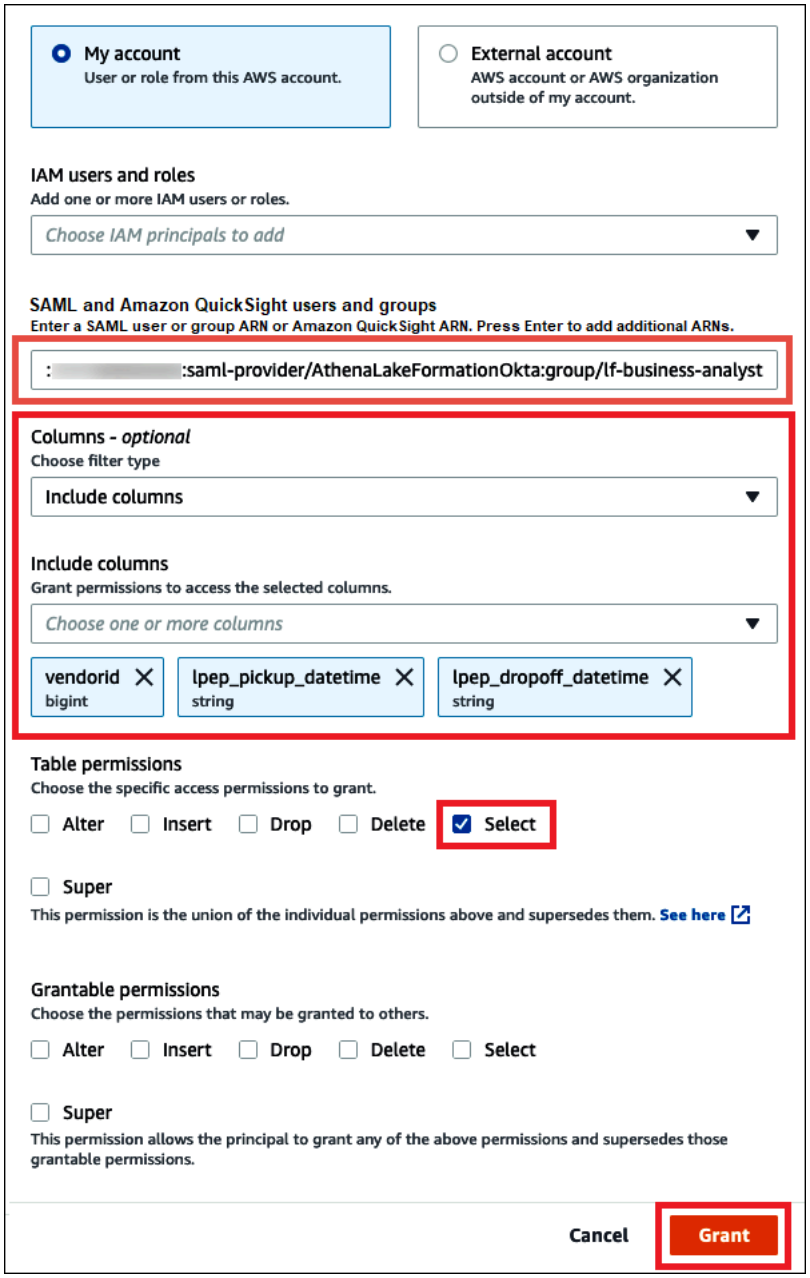
   a. Geben Sie unter SAML- und Amazon-QuickSight-Benutzer und -Gruppen den Okta-SAML-Gruppen-ARN im folgenden Format ein:

   ```
   arn:aws:iam::<account-id>:saml-provider/AthenaLakeFormationOkta:group/lf-business-analyst
   ```

   b. Wählen Sie für Spalten, Filtertyp auswählen, Spalten einschließen aus.

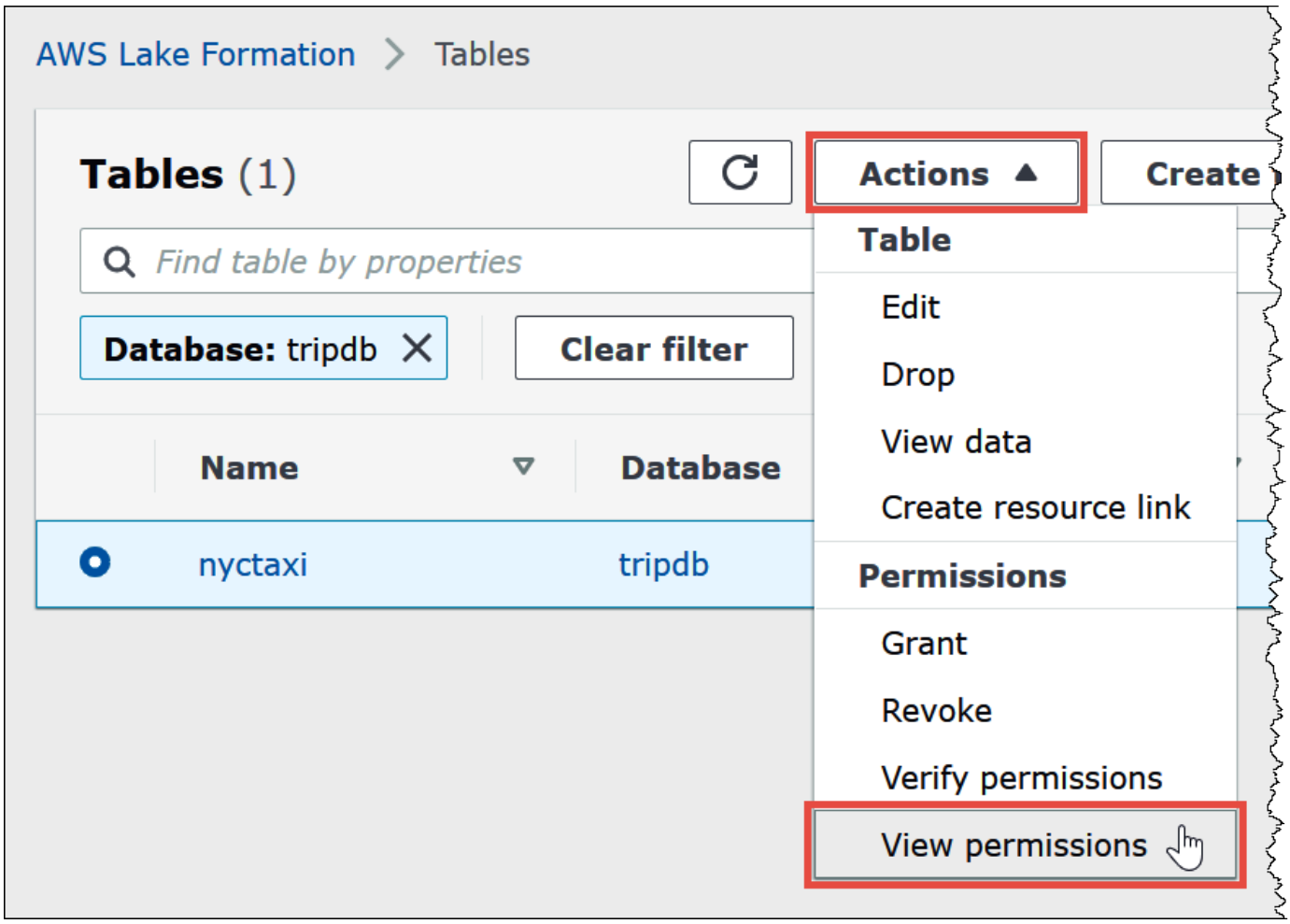
c. Wählen Sie für Eine oder mehrere Spalten auswählen die ersten drei Spalten der Tabelle aus.

d. Wählen Sie für Tabellenberechtigungen die zu erteilenden spezifischen Zugriffsberechtigungen aus. Dieses Tutorial gewährt nur die SELECT-Berechtigung; Ihre Anforderungen können variieren.

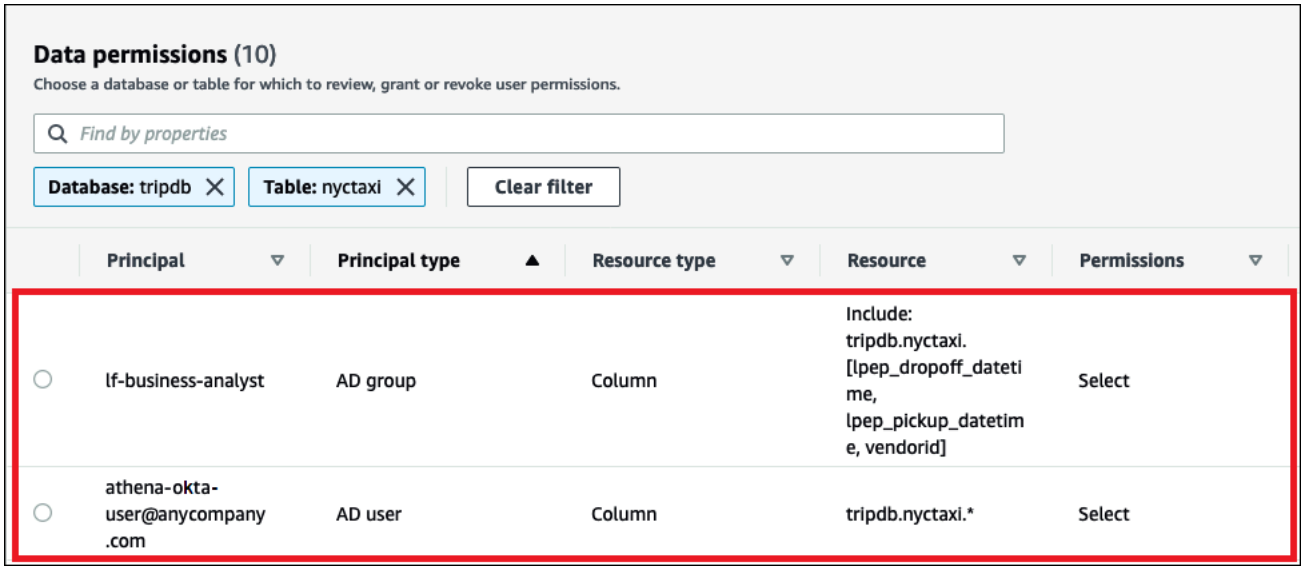


4. Wählen Sie Gewähren.

5. Um die von Ihnen erteilten Berechtigungen zu überprüfen, wählen Sie Aktionen, Berechtigungen anzeigen.



Auf der Seite Data permissions (Datenberechtigungen) für die `nyctaxi`-Tabelle werden die Berechtigungen für athena-okta-user und die lf-business-analyst-Gruppe angezeigt.



Schritt 7: Überprüfen des Zugriffs über den Athena-JDBC-Client

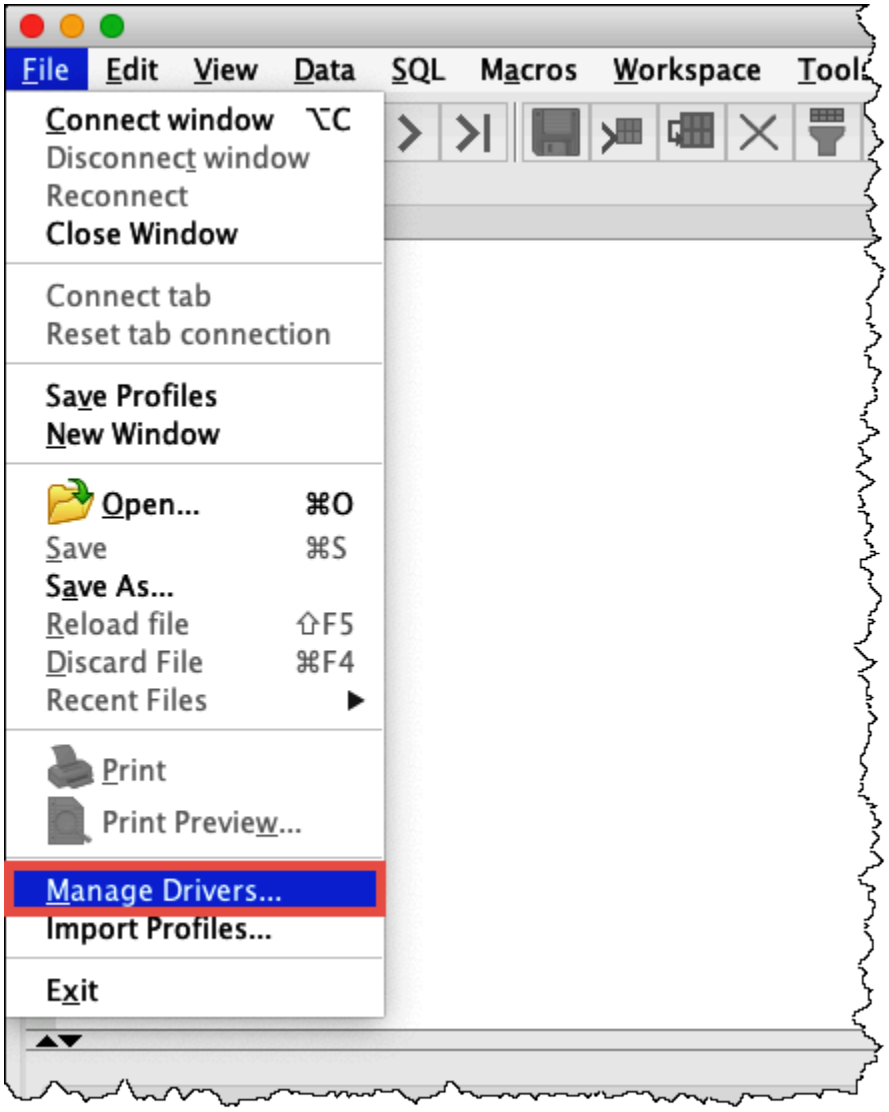
Jetzt können Sie einen JDBC-Client verwenden, um eine Testverbindung mit Athena als Okta-SAML-Benutzer durchzuführen.

In diesem Abschnitt führen Sie die folgenden Aufgaben aus:

- Vorbereiten des Testclients – Laden Sie den Athena-JDBC-Treiber herunter, installieren Sie SQL Workbench und fügen Sie den Treiber zu Workbench hinzu. In diesem Lernprogramm wird SQL Workbench verwendet, um über Okta-Authentifizierung auf Athena zuzugreifen und Lake-Formation-Berechtigungen zu überprüfen.
- In SQL Workbench:
  - Erstellen Sie eine Verbindung für den Athena-Okta-Benutzer.
  - Führen Sie Testabfragen als Athena-Okta-Benutzer aus.
  - Erstellen und testen Sie eine Verbindung für den Benutzer des Business Analyst.
- Fügen Sie in der Okta-Konsole den Business-Analyst-Benutzer der Entwicklergruppe hinzu.
- Konfigurieren Sie in der Lake-Formation-Konsole Tabellenberechtigungen für die Entwicklergruppe.
- Führen Sie in SQL Workbench Testabfragen als Business-Analyst-Benutzer aus und überprüfen Sie, wie sich die Änderung der Berechtigungen auf die Ergebnisse auswirkt.

So bereiten Sie den Test-Client vor

1. Laden Sie den Lake-Formation-kompatiblen Athena-JDBC-Treiber (2.0.14 oder höher) von Herstellen einer Verbindung zu Amazon Athena mit JDBC herunter und extrahieren Sie ihn.
2. Laden Sie das kostenlose SQL Workbench/J SQL-Abfragetool herunter und installieren Sie es, das unter einer modifizierten Apache-2.0-Lizenz verfügbar ist.
3. Wählen Sie in SQL Workbench die Optionen Datei und Treiber verwalten aus.



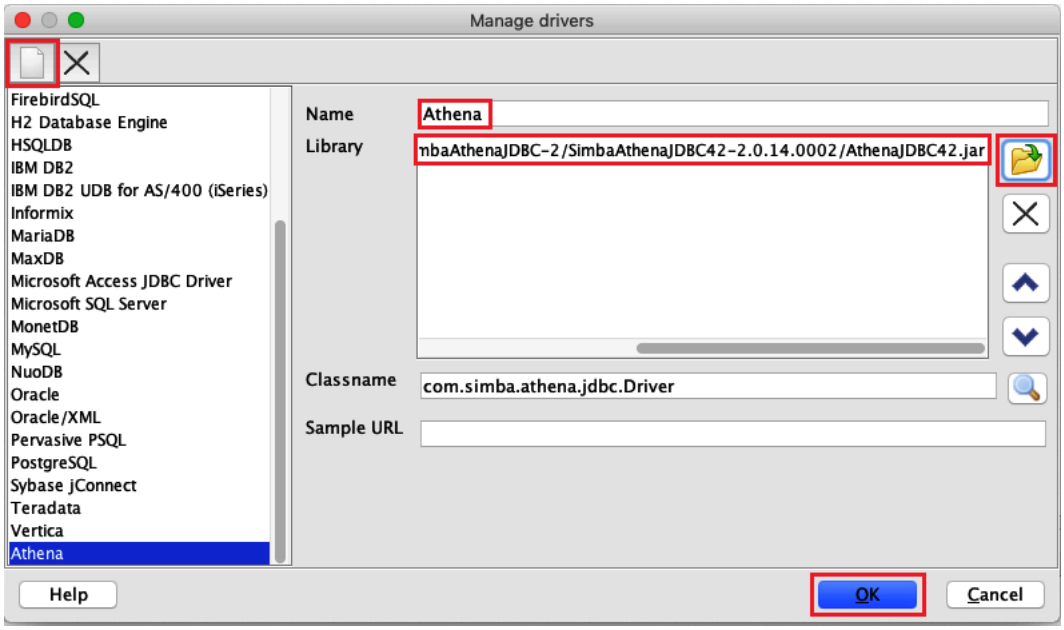
4. Führen Sie im Dialogfeld Treiber verwalten die folgenden Schritte aus:

   a. Wählen Sie das Symbol für den neuen Treiber.

   b. Geben Sie unter Name **`Athena`** ein.

   c. Navigieren Sie für Bibliothek zu der `.jar`-Simba-Athena-JDBC-Datei, die Sie gerade heruntergeladen haben und wählen Sie sie aus.

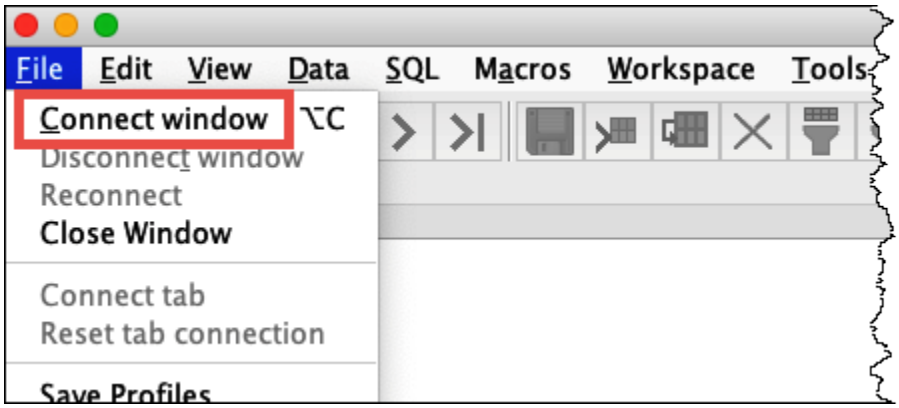
   d. Wählen Sie OK.



Sie können nun eine Verbindung für den Athena-Okta-Benutzer erstellen und testen.

So erstellen Sie eine Verbindung für den Okta-Benutzer

1. Wählen Sie Datei, Verbindungsfenster.



2. Erstellen Sie im Dialogfeld Verbindungsprofil eine Verbindung, indem Sie die folgenden Informationen eingeben:

   - Geben Sie im Feld Name **`Athena_Okta_User_Connection`** ein.
   - Wählen Sie als Treiber den Simba-Athena-JDBC-Treiber aus.
   - Führen Sie für URL einen der folgenden Schritte aus:

- Um eine Verbindungs-URL zu verwenden, geben Sie eine einzeilige Verbindungszeichenfolge ein. Das folgende Beispiel fügt Zeilenumbrüche für eine bessere Lesbarkeit hinzu.

```
jdbc:awsathena://AwsRegion=region-id;
S3OutputLocation=s3://athena-query-results-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;
```

- Führen Sie die folgenden Schritte aus, um eine AWS-profilbasierte URL zu verwenden:

  1. Konfigurieren Sie ein AWS-Profil, das über eine AWS-Anmeldeinformationendatei verfügt, wie im folgenden Beispiel.

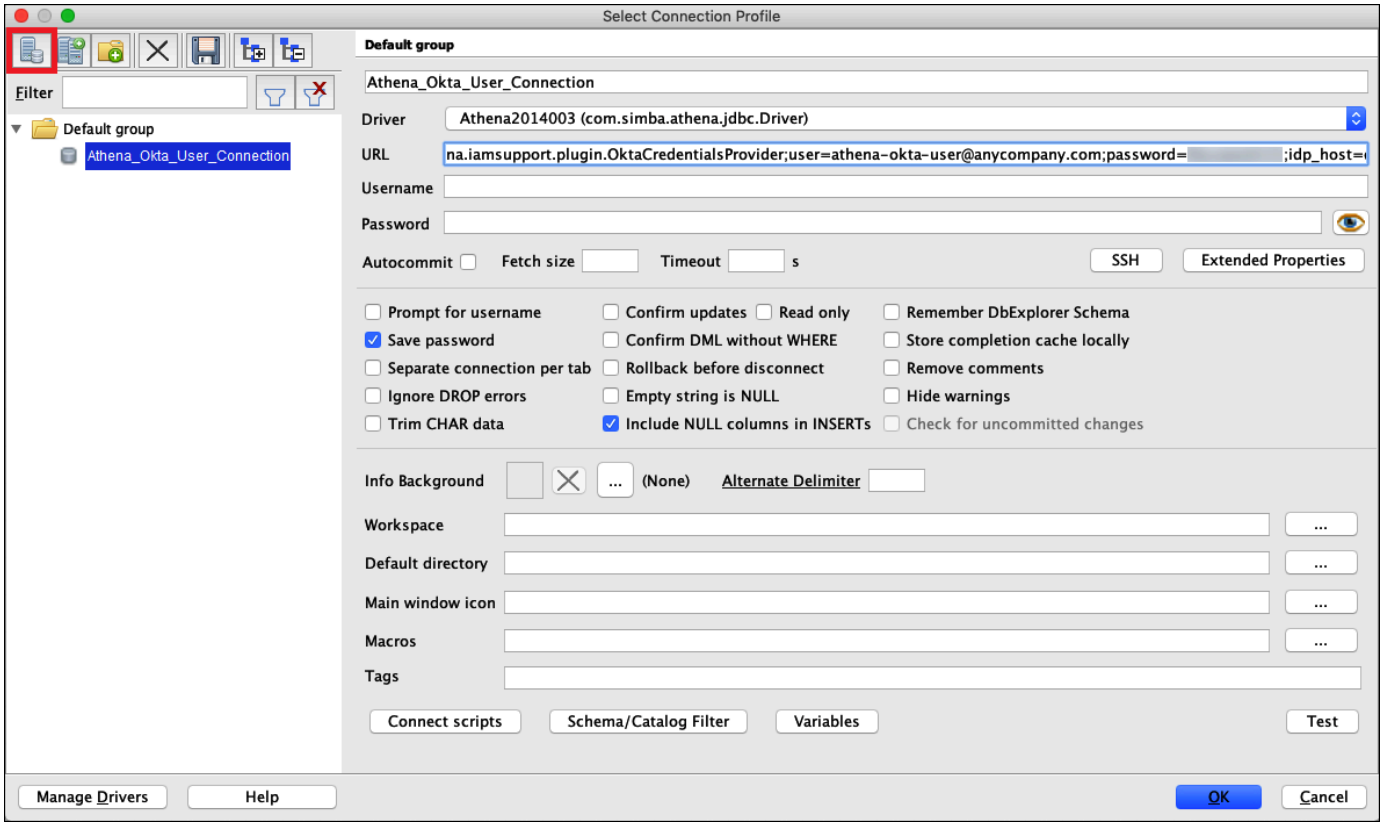
```
[athena_lf_dev]
plugin_name=com.simba.athena.iamsupport.plugin.OktaCredentialsProvider
idp_host=okta-idp-domain
app_id=okta-app-id
uid=athena-okta-user@anycompany.com
pwd=password
```

  2. Geben Sie für URL eine einzeilige Verbindungszeichenfolge wie im folgenden Beispiel ein. Das Beispiel fügt Zeilenumbrüche für eine bessere Lesbarkeit hinzu.

```
jdbc:awsathena://AwsRegion=region-id;
S3OutputLocation=s3://athena-query-results-bucket/athena_results;
profile=athena_lf_dev;
SSL_Insecure=true;
LakeFormationEnabled=true;
```

  Beachten Sie, dass diese Beispiele grundlegende Darstellungen der URL sind, die zum Herstellen einer Verbindung mit Athena erforderlich ist. Eine vollständige Liste der in der URL unterstützten Parameter finden Sie in JDBC-Dokumentation.

Das folgende Image zeigt ein SQL-Workbench-Verbindungsprofil, das eine Verbindungs-URL verwendet.

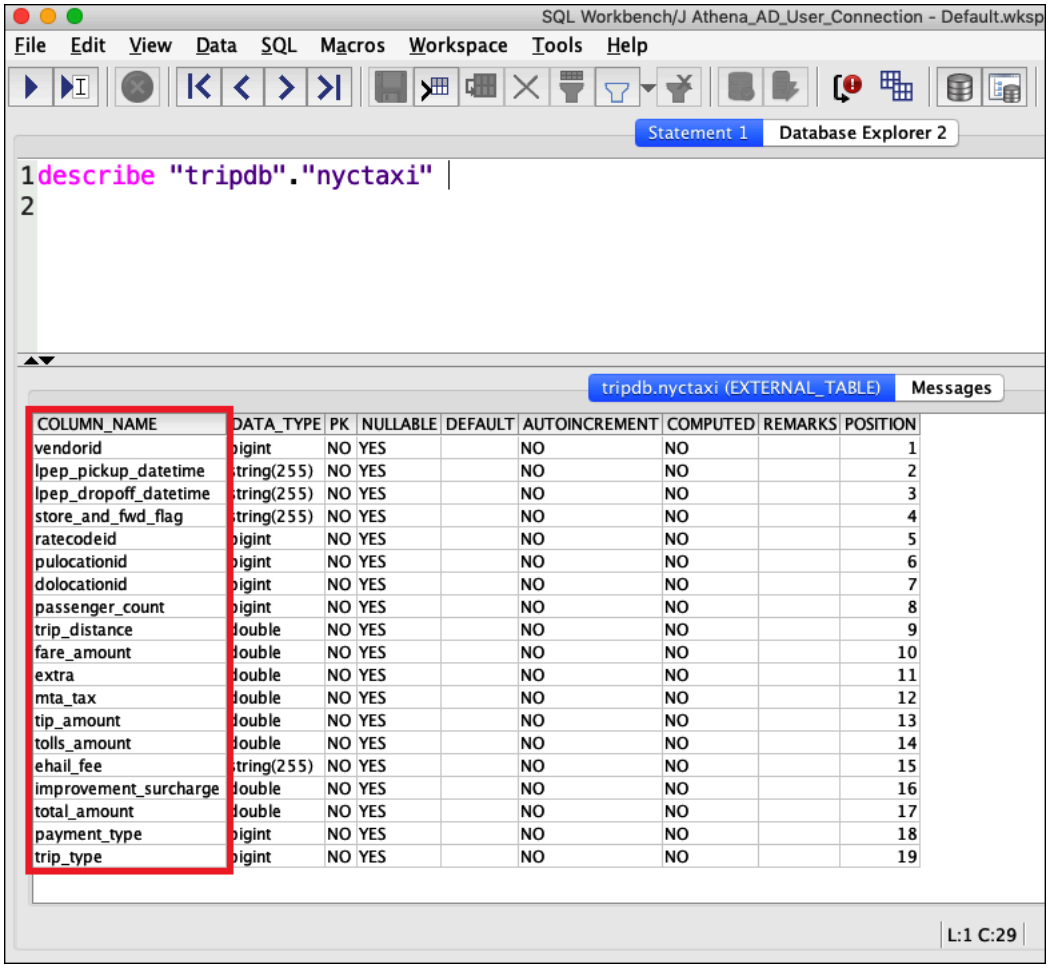


Nachdem Sie eine Verbindung für den Okta-Benutzer hergestellt haben, können Sie diese testen, indem Sie einige Daten abrufen.

So testen Sie die Verbindung für den Okta-Benutzer

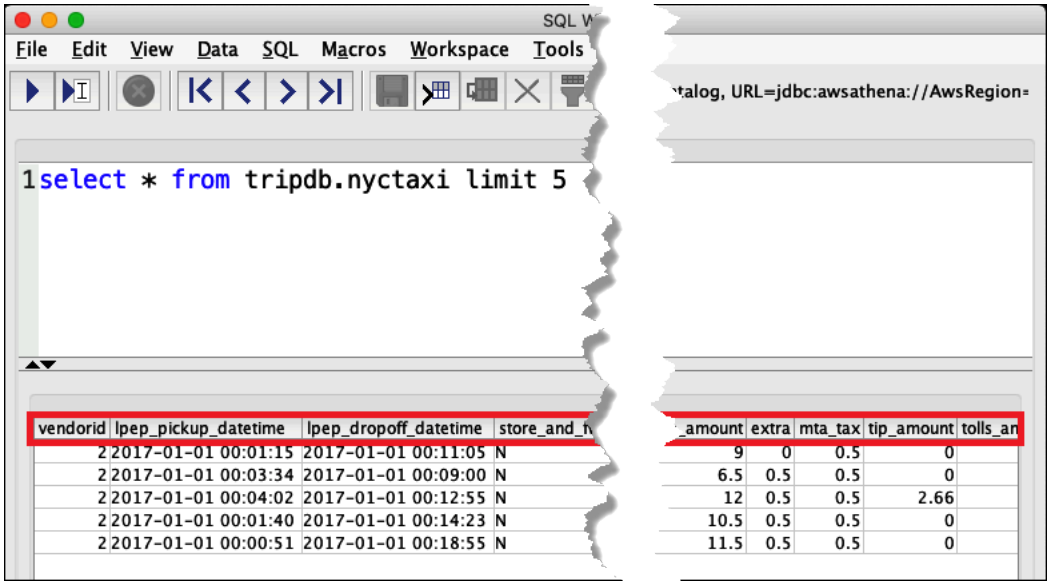
1. Wählen Sie Test (Testen) und überprüfen Sie dann, ob die Verbindung erfolgreich ist.
2. Führen Sie im Fenster SQL-Workbench-Anweisung den folgenden SQL-Befehl `DESCRIBE` aus. Stellen Sie sicher, dass alle Spalten angezeigt werden.

```
DESCRIBE "tripdb"."nyctaxi"
```



3. Führen Sie im Fenster SQL-Workbench-Anweisung den folgenden SQL-Befehl SELECT aus. Stellen Sie sicher, dass alle Spalten angezeigt werden.

```
SELECT * FROM tripdb.nyctaxi LIMIT 5
```



Als Nächstes überprüfen Sie, ob der athena-ba-user als Mitglied der lf-business-analyst-Gruppe nur Zugriff auf die ersten drei Spalten der Tabelle hat, die Sie zuvor in Lake Formation angegeben haben.

So überprüfen Sie den Zugriff auf die athena-ba-user

1. Erstellen Sie in SQL Workbench im Dialogfeld Verbindungsprofil ein weiteres Verbindungsprofil.

   - Geben Sie als Verbindungsprofilnamen **`Athena_Okta_Group_Connection`** ein.
   - Wählen Sie für Treiber den Simba-Athena-JDBC-Treiber aus.
   - Führen Sie für URL einen der folgenden Schritte aus:
     - Um eine Verbindungs-URL zu verwenden, geben Sie eine einzeilige Verbindungszeichenfolge ein. Das folgende Beispiel fügt Zeilenumbrüche für eine bessere Lesbarkeit hinzu.

```
jdbc:awsathena://AwsRegion=region-id;
S3OutputLocation=s3://athena-query-results-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;
```

- Führen Sie die folgenden Schritte aus, um eine AWS-profilbasierte URL zu verwenden:

  1. Konfigurieren Sie ein AWS-Profil, das über eine Anmeldeinformationendatei verfügt, wie im folgenden Beispiel.

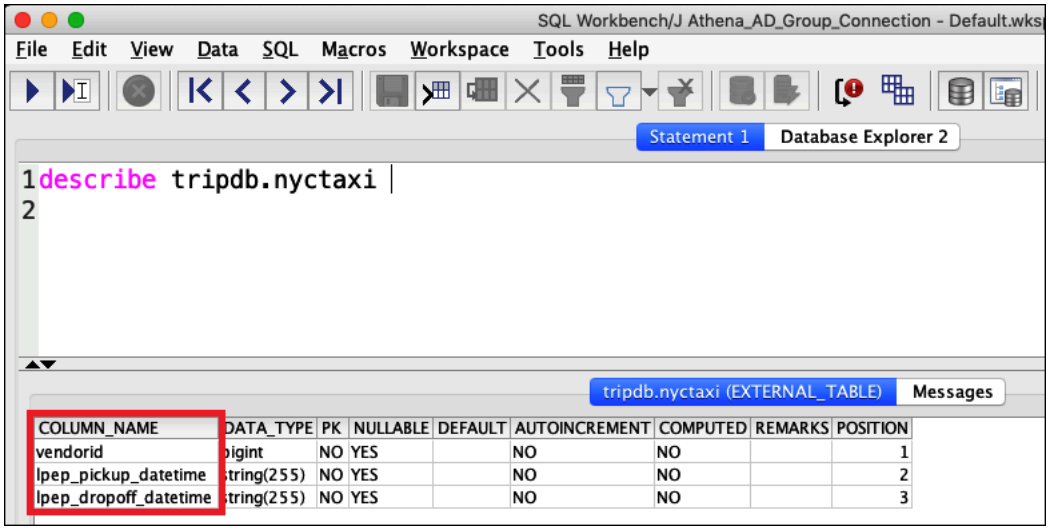
     ```
     [athena_lf_ba]
     plugin_name=com.simba.athena.iamsupport.plugin.OktaCredentialsProvider
     idp_host=okta-idp-domain
     app_id=okta-application-id
     uid=athena-ba-user@anycompany.com
     pwd=password
     ```

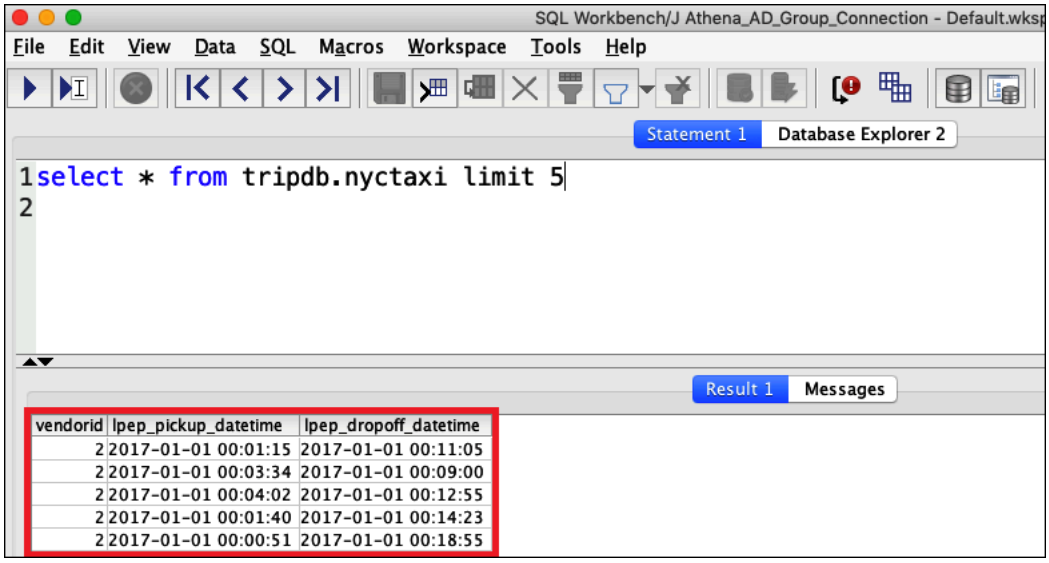
  2. Geben Sie für URL eine einzeilige Verbindungszeichenfolge wie im Folgenden ein. Das Beispiel fügt Zeilenumbrüche für eine bessere Lesbarkeit hinzu.

     ```
     jdbc:awsathena://AwsRegion=region-id;
     S3OutputLocation=s3://athena-query-results-bucket/athena_results;
     profile=athena_lf_ba;
     SSL_Insecure=true;
     LakeFormationEnabled=true;
     ```

2. Wählen Sie Testen, um zu bestätigen, dass die Verbindung erfolgreich ist.
3. Führen Sie im Fenster SQL-Anweisung dieselben `DESCRIBE`- und `SELECT`-SQL-Befehle wie zuvor aus, und überprüfen Sie die Ergebnisse.

   Da athena-ba-user Mitglied der lf-business-analyst-Gruppe ist, werden nur die ersten drei Spalten zurückgegeben, die Sie in der Lake-Formation-Konsole angegeben haben.

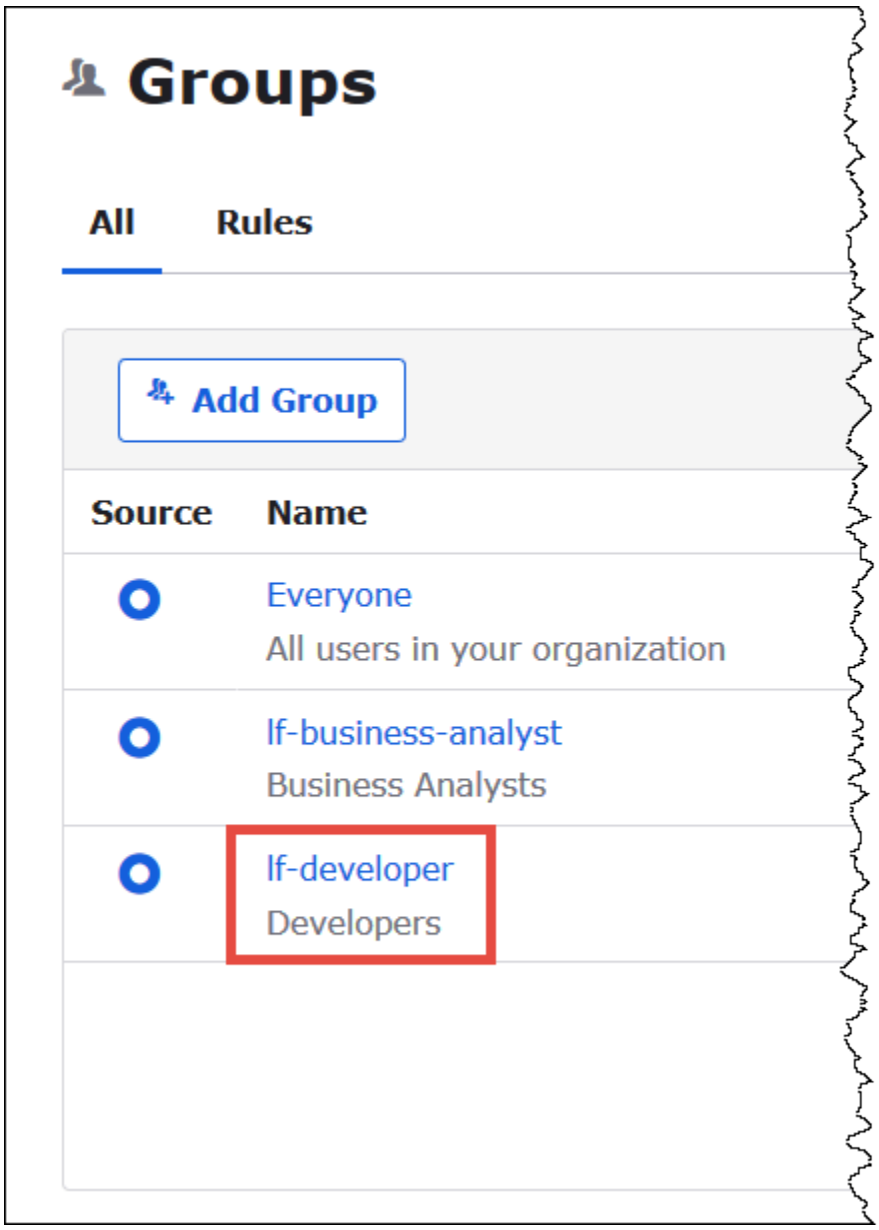




Als nächstes kehren Sie zur Okta-Konsole zurück, um den `athena-ba-user` zur `lf-developer`-Okta-Gruppe hinzuzufügen.

So fügen Sie der Gruppe lf-developer den athena-ba-user hinzu

1. Melden Sie sich bei der Okta-Konsole als Administrator der zugewiesenen Okta-Domain an.
2. Wählen Sie Verzeichnis und dann Gruppen aus.
3. Wählen Sie auf der Seite Gruppen die Gruppe lf-developer aus.



4. Wählen Sie Manage People (Verwalten von Personen).
5. Wählen Sie aus der Liste Keine Mitglieder den athena-ba-user aus, um ihn der lf-developer-Gruppe hinzuzufügen.
6. Wählen Sie Save (Speichern) aus.

Nun kehren Sie zur Lake Formation-Konsole zurück, um die Tabellenberechtigungen für die lf-developer-Gruppe zu konfigurieren.

So konfigurieren Sie Tabellenberechtigungen für die lf-developer-group

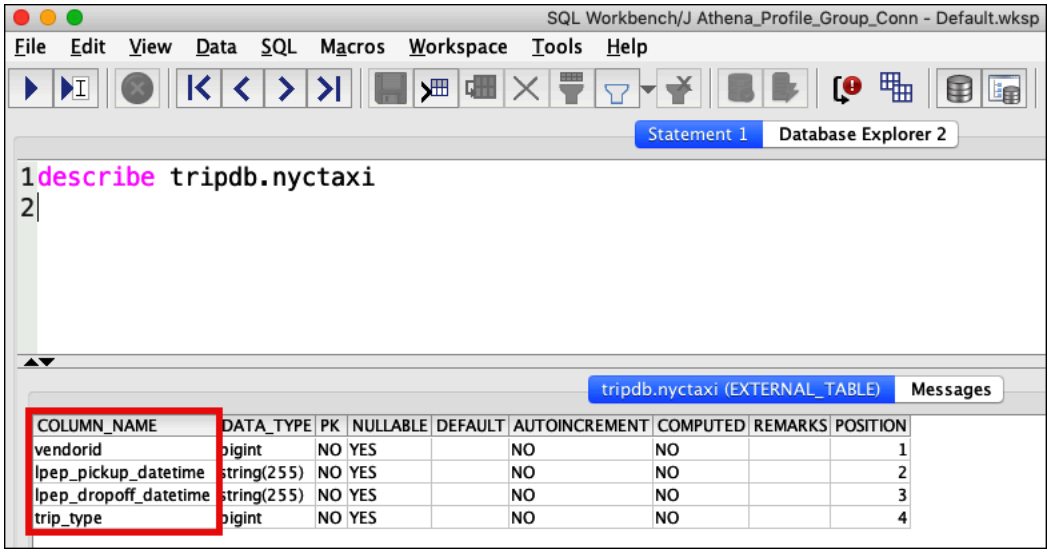
1. Melden Sie sich als Data-Lake-Administrator bei der Lake-Formation-Konsole an.
2. Wählen Sie im Navigationsbereich Tables (Tabellen) aus.
3. Wählen Sie die nyctaxi-Tabelle aus.
4. Wählen Sie Aktionen, Gewähren.
5. Geben Sie im Dialogfeld Berechtigungen gewähren die folgenden Informationen ein:
   - Geben Sie unter SAML- und Amazon-QuickSight-Benutzer und -Gruppen den ARN der Okta SAML lf-developer-Gruppe im folgenden Format ein:
   - Wählen Sie für Spalten, Filtertyp auswählen, Spalten einschließen aus.
   - Wählen Sie die Spalte trip_type aus.
   - Wählen Sie für Tabellenberechtigungen die Option Auswählen aus.
6. Wählen Sie Gewähren.

Jetzt können Sie SQL Workbench verwenden, um die Änderung der Berechtigungen für die lf-developer-Gruppe zu überprüfen. Die Änderung sollte sich in den Daten widerspiegeln, die dem athena-ba-user zur Verfügung stehen, der jetzt Mitglied der lf-developer-Gruppe ist.

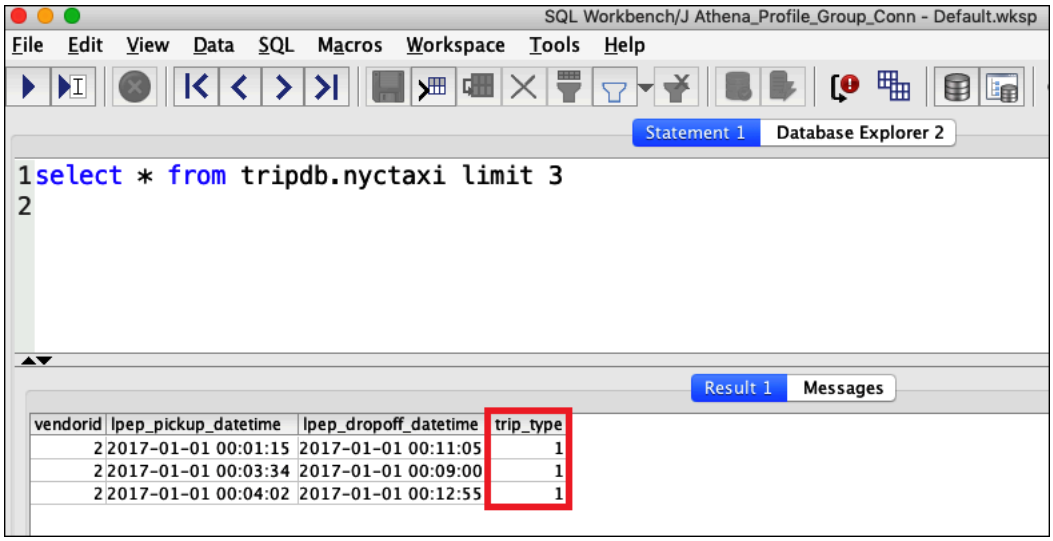
So überprüfen Sie die Änderung der Berechtigungen für athena-ba-user

1. Schließen Sie das SQL-Workbench-Programm und öffnen Sie es dann erneut.
2. Verbinden Sie sich mit dem Profil für athena-ba-user.
3. Geben Sie im Anweisungsfenster dieselben SQL-Anweisungen aus, die Sie zuvor ausgeführt haben:

   Dieses Mal wird die Spalte trip_type angezeigt.



Da athena-ba-user nun sowohl Mitglied der lf-developer- als auch der lf-business-analyst-Gruppe ist, bestimmt die Kombination der Lake-Formation-Berechtigungen für diese Gruppen die zurückgegebenen Spalten.



Schlussfolgerung

In diesem Tutorial haben Sie die Athena-Integration mit AWS Lake Formation mit Okta als SAML-Anbieter konfiguriert. Sie haben Lake Formation und IAM verwendet, um die Ressourcen zu steuern, die dem SAML-Benutzer in Ihrem Data-Lake-AWS Glue-Datenkatalog zur Verfügung stehen.

Zugehörige Ressourcen

Weitere Informationen finden Sie in den folgenden Ressourcen.

- Herstellen einer Verbindung zu Amazon Athena mit JDBC
- Aktivieren des föderierten Zugriffs auf die Athena-API
- AWS Lake Formation-Entwicklerhandbuch
- Erteilen und Widerrufen von Datenkatalog-Berechtigungen im AWS Lake Formation-Entwicklerhandbuch.
- Identitätsanbieter und Verbund im IAM-Benutzerhandbuch.
- Erstellen von IAM-SAML-Identitätsanbietern im IAM-Benutzerhandbuch.
- Aktivieren des Verbunds zu AWS mit Windows Active Directory, ADFS und SAML 2.0 im AWS-Sicherheitsblog.

Workload-Management

Sie können die Features von Athena für Arbeitsgruppen, Kapazitätsmanagement, Leistungsoptimierung, Komprimierungsunterstützung, Tags und Servicekontingent-Features verwenden, um Ihren Workload zu verwalten.

Themen

- Verwendung von Arbeitsgruppen zur Kontrolle des Abfragenzugriffs und der Kosten
- Kapazität zur Abfrageverarbeitung verwalten
- Leistungsoptimierung in Athena
- Athena-Komprimierungs-Support
- Markieren von Athena-Ressourcen
- Service Quotas

Verwendung von Arbeitsgruppen zur Kontrolle des Abfragenzugriffs und der Kosten

Verwenden Sie Arbeitsgruppen zum Trennen von Benutzern, Teams, Anwendungen oder Workloads sowie zur Einrichtung von Grenzwerten für die Datenmenge, die jede Abfrage oder die gesamte Arbeitsgruppe verarbeiten kann, und zur Nachverfolgung von Kosten. Da Arbeitsgruppen als Ressourcen fungieren, können Sie identitätsbasierte Richtlinien auf Ressourcenebene verwenden, um den Zugriff auf eine bestimmte Arbeitsgruppe zu steuern. Sie können auch abfragebezogene Metriken in Amazon anzeigen CloudWatch, Kosten kontrollieren, indem Sie Limits für die Menge der

gescannten Daten konfigurieren, Schwellenwerte erstellen und Aktionen auslösen, z. B. Amazon SNS, wenn diese Schwellenwerte überschritten werden.

Zur weiteren Kostenkontrolle können Sie Kapazitätsreservierungen mit der von Ihnen angegebenen Anzahl von Datenverarbeitungseinheiten erstellen und der Reservierung eine oder mehrere Arbeitsgruppen hinzufügen. Weitere Informationen finden Sie unter Kapazität zur Abfrageverarbeitung verwalten.

Arbeitsgruppen lassen sich wie folgt in IAM, CloudWatchAmazon Simple Notification Service und AWS Kosten- und Nutzungsberichte integrieren:

- Identitätsbasierte IAM-Richtlinien mit Berechtigungen auf Ressourcenebene steuern, wer in einer Arbeitsgruppe Abfragen ausführen kann.
- Athena veröffentlicht die Arbeitsgruppenabfragemetriken in CloudWatch, wenn Sie Abfragemetriken aktivieren.
- In Amazon SNS, können Sie Amazon-SNS-Themen erstellen, die Alarme für bestimmte Benutzer der Arbeitsgruppe ausgeben, wenn die Datennutzungskontrollen für Abfragen in einer Arbeitsgruppe Ihre festgelegten Schwellenwerte überschreiten.
- Wenn Sie eine Arbeitsgruppe mit einem Tag kennzeichnen, das in der Rechnungsstellungs- und Kostenmanagementkonsole als Kostenzuordnungs-Tag konfiguriert ist, werden die Kosten, die mit den laufenden Abfragen in dieser Arbeitsgruppe verknüpft sind, in Ihren Kosten- und Nutzungsberichten mit diesem Kostenzuordnungs-Tag angezeigt.

Themen

- Verwenden von Arbeitsgruppen zum Ausführen von Abfragen
- Steuern von Kosten und Überwachen von Abfragen mit CloudWatch Metriken und Ereignissen

Weitere Informationen finden Sie auch im AWS -Big-Data-Blogbeitrag Separate Abfragen und Kostenmanagement mithilfe von Amazon Athena-Arbeitsgruppen, der Ihnen zeigt, wie Sie Arbeitsgruppen verwenden, um Workloads zu trennen, den Benutzerzugriff zu steuern und die Abfragenutzung und -kosten zu verwalten.

Verwenden von Arbeitsgruppen zum Ausführen von Abfragen

Wir empfehlen, Arbeitsgruppen zu verwenden, um Abfragen für Teams, Anwendungen oder verschiedene Arbeitslasten zu isolieren. Sie können beispielsweise separate Arbeitsgruppen für zwei verschiedene Teams in Ihrer Organisation erstellen. Sie können auch Arbeitslasten trennen.

So können Sie beispielsweise zwei unabhängige Arbeitsgruppen erstellen, eine für automatisierte geplante Anwendungen wie beispielsweise die Berichterstellung und eine andere zur Ad-hoc-Nutzung durch Analysten. Sie können zwischen Arbeitsgruppen umschalten.

Themen

Vorteile der Verwendung von Arbeitsgruppen

Arbeitsgruppen bieten Ihnen folgende Möglichkeiten:

Isolieren von Benutzern, Teams, Anwendungen oder Arbeitslasten in Gruppen	Jede Arbeitsgruppe verfügt über ihren eigenen Abfrageverlauf und eine Liste gespeicherter Abfragen. Weitere Informationen finden Sie unter Funktionsweise von Arbeitsgruppen. Für alle Abfragen in der Arbeitsgruppe können Sie Arbeitsgruppeneinstellungen konfigurieren. Dazu gehören ein Amazon-S3-Speicherort für Abfrageergebnisse, der erwartete Bucket-Eigentümer, die Verschlüsselung und die Steuerung von Objekten, die in den Abfrageergebnis-Bucket geschrieben werden. Sie können auch Arbeitsgruppeneinstellungen durchsetzen. Weitere Informationen finden Sie unter Arbeitsgruppen-Einstellungen.
Durchsetzen von Kosteneinschränkungen	Sie können zwei Arten von Kosteneinschränkungen für Abfragen in einer Arbeitsgruppe festlegen: • Das Limit pro Abfrage ist ein Schwellenwert für die Datenmenge, die für jede Abfrage gescannt wird. Athena bricht Abfragen ab,

	wenn sie den angegebenen Schwellenwert überschreiten. Das Limit gilt für jede ausgeführte Abfrage innerhalb einer Arbeitsgruppe. Sie können nur ein Limit pro Abfrage festlegen und dieses bei Bedarf aktualisieren. • Das Per-workgroup limit (Limit pro Arbeitsgruppe) ist ein Schwellenwert, den Sie für jede Arbeitsgruppe festlegen können und der die Menge der bei Abfragen in der Arbeitsgruppe gescannten Daten betrifft. Bei Überschreiten eines Schwellenwerts wird ein Amazon-SNS-Alarm aktiviert, der eine Aktion Ihrer Wahl auslöst, beispielsweise das Senden einer E-Mail an einen bestimmten Benutzer. Sie können mehrere Limits pro Arbeitsgruppe für jede Arbeitsgruppe festlegen. Die detaillierten Schritte finden Sie unter Festlegen von Limits zur Kontrolle der Datennutzung.
Verfolgen Sie abfragebezogene Metriken für alle Arbeitsgruppenabfragen in CloudWatch.	Wenn Sie die Arbeitsgruppe für jede Abfrage konfigurieren, die in einer Arbeitsgruppe ausgeführt wird, um Metriken zu veröffentlichen, veröffentlicht Athena sie in CloudWatch. Sie können Abfragemetriken für Ihre Arbeitsgruppen in der Athena-Konsole anzeigen. In können CloudWatchSie benutzerdefinierte Dashboards erstellen und Schwellenwerte und Alarme für diese Metriken festlegen.

Funktionsweise von Arbeitsgruppen

Arbeitsgruppen in Athena haben folgende Merkmale:

- Standardmäßig verfügt jedes Konto über eine primäre Arbeitsgruppe und alle authentifizierten Benutzer können mit den Standardberechtigungen auf diese Arbeitsgruppe zugreifen. Die primäre Arbeitsgruppe kann nicht gelöscht werden.
- In jeder von Ihnen erstellten Arbeitsgruppe werden gespeicherte Abfragen und der Abfrageverlauf nur für die in der Gruppe ausgeführten Abfragen, nicht für alle Abfragen in dem Konto angezeigt. So sind Ihre Abfragen von anderen Abfragen in einem Konto getrennt und Sie können Ihre eigenen gespeicherten Abfragen und Abfragen im Verlauf effizienter lokalisieren.

- Wenn eine Arbeitsgruppe deaktiviert wird, können keine Abfragen darin ausgeführt werden, bis Sie die Gruppe aktivieren. Abfragen an eine deaktivierte Arbeitsgruppe schlagen fehl, bis Sie die Gruppe erneut aktivieren.
- Wenn Sie über Berechtigungen verfügen, können Sie eine leere Arbeitsgruppe und eine Arbeitsgruppe mit gespeicherten Abfragen löschen. In diesem Fall werden Sie vor dem Löschen einer Arbeitsgruppe von Athena gewarnt, dass gespeicherte Abfragen gelöscht werden. Stellen Sie vor dem Löschen einer Arbeitsgruppe, auf die andere Benutzer Zugriff haben, sicher, dass ihre Benutzer Zugriff auf andere Arbeitsgruppen haben, in denen sie weiterhin Abfragen ausführen können.
- Sie können Einstellungen für eine gesamte Arbeitsgruppe festlegen und deren Nutzung bei allen in der Arbeitsgruppe ausgeführten Abfragen durchsetzen. Die Einstellungen umfassen einen Abfrageergebnisstandort in Amazon S3, den erwarteten Bucket-Eigentümer, die Verschlüsselung und die Steuerung von Objekten, die in den Abfrageergebnis-Bucket geschrieben werden.

> **Important**
>
> Wenn Sie Einstellungen für eine gesamte Arbeitsgruppe durchsetzen, verwenden alle in dieser Arbeitsgruppe ausgeführten Abfragen die Arbeitsgruppeneinstellungen. Das gilt auch, wenn ihre clientseitigen Einstellungen möglicherweise von den Arbeitsgruppeneinstellungen abweichen. Weitere Informationen finden Sie unter Arbeitsgruppen-Einstellungen überschreiben clientseitige Einstellungen.

Einschränkungen für Arbeitsgruppen

- Sie können bis zu 1000 Arbeitsgruppen pro Region in Ihrem Konto erstellen.
- Die primäre Arbeitsgruppe kann nicht gelöscht werden.
- Sie können innerhalb jeder Arbeitsgruppe bis zu zehn Abfrageregisterkarten öffnen. Wenn Sie zwischen Arbeitsgruppen wechseln, bleiben Ihre Abfrageregisterkarten für bis zu drei Arbeitsgruppen geöffnet.

Einrichten von Arbeitsgruppen

Die Einrichtung von Arbeitsgruppen umfasst das Erstellen der Gruppen und die Festlegung von Berechtigungen für ihre Nutzung. Entscheiden Sie zunächst, welche Arbeitsgruppen Ihre Organisation braucht, und erstellen Sie diese. Legen Sie dann IAM-Arbeitsgruppenrichtlinien fest, die

den Benutzerzugriff und die Aktionen auf einer `workgroup`-Ressource steuern. Benutzer mit Zugriff auf diese Arbeitsgruppen können nun Abfragen darin ausführen.

> **Note**
>
> Verwenden Sie diese Tasks, um Arbeitsgruppen einzurichten, wenn Sie diese zum ersten Mal verwenden. Wenn Ihr Athena-Konto bereits Arbeitsgruppen verwendet, benötigt jeder Benutzer des Kontos Berechtigungen zur Ausführung von Abfragen in einer oder mehreren Arbeitsgruppen in dem Konto. Überprüfen Sie vor der Ausführung von Abfragen Ihre IAM-Richtlinien, um festzustellen, auf welche Arbeitsgruppen Sie zugreifen können. Passen Sie Ihre Richtlinie bei Bedarf an und wechseln Sie zu einer Arbeitsgruppe, die Sie verwenden möchten.

Wenn Sie keine Arbeitsgruppen erstellt haben, werden alle Abfragen in Ihrem Konto standardmäßig in der primären Arbeitsgruppe ausgeführt.

Athena zeigt die aktuelle Arbeitsgruppe in der Arbeitsgruppe-Option rechts oben in der Konsole an. Sie können diese Option verwenden, um Arbeitsgruppen zu wechseln. Wenn Sie Abfragen ausführen, werden sie in der aktuellen Arbeitsgruppe ausgeführt. Sie können Abfragen im Kontext einer Arbeitsgruppe in der Konsole, über API-Vorgänge, über die Befehlszeilenschnittstelle oder über eine Clientanwendung mithilfe des JDBC- oder ODBC-Treibers ausführen. Wenn Sie Zugriff auf eine Arbeitsgruppe haben, können Sie die Einstellungen der Arbeitsgruppe, ihre Metriken und Limits für die Datennutzungskontrolle anzeigen. Mit zusätzlichen Berechtigungen können Sie die Einstellungen und Grenzwerte für die Datenverwendungssteuerung bearbeiten.

So richten Sie Arbeitsgruppen ein

1. Legen Sie fest, welche Arbeitsgruppen Sie erstellen möchten. Sie können beispielsweise Folgendes entscheiden:

   - Wer Abfragen in den einzelnen Arbeitsgruppen ausführen kann und wer für die Arbeitsgruppenkonfiguration verantwortlich ist. Dies legt die von Ihnen erstellten IAM-Richtlinien fest. Weitere Informationen finden Sie unter IAM-Richtlinien für den Zugriff auf Arbeitsgruppen.
   - Welche Speicherorte Amazon S3 für die Ergebnisse der Abfragen verwendet werden sollen, die in jeder Arbeitsgruppe ausgeführt werden. In Amazon S3 muss ein Speicherort vorhanden sein, bevor Sie ihn für die Ergebnisse der Arbeitsgruppenabfrage angeben können. Alle

Benutzer, die eine Arbeitsgruppe verwenden, müssen Zugriff auf diesen Speicherort haben. Weitere Informationen finden Sie unter Arbeitsgruppen-Einstellungen.

- Ob der Besitzer des Amazon-S3-Abfrageergebnis-Bucket die volle Kontrolle über neue Objekte hat, die in den Bucket geschrieben werden. Wenn der Standort Ihres Abfrageergebnisses beispielsweise einem anderen Konto gehört, können Sie dem anderen Konto das Eigentum und die volle Kontrolle über Ihre Abfrageergebnisse gewähren. Weitere Informationen finden Sie unter AclConfiguration.
- Geben Sie die ID des an AWS-Konto , von dem Sie erwarten, dass es der Eigentümer des Ausgabespeicherort-Buckets ist. Dies ist eine optionale zusätzliche Sicherheitsmaßnahme. Wenn die Konto-ID des Bucket-Eigentümers nicht mit der ID übereinstimmt, die Sie hier angeben, schlagen Versuche, in den Bucket auszugeben, fehl. Weitere Informationen finden Sie unter Überprüfen der Bucket-Eigentümerschaft mit Bucket-Eigentümer-Bedingung im Amazon-S3-Benutzerhandbuch. Diese Einstellung gilt nicht für CTAS-, INSERT-INTO- oder UNLOAD-Anweisungen.
- Welche Verschlüsselungseinstellungen erforderlich sind und welche Arbeitsgruppen Abfragen haben, die verschlüsselt werden müssen. Wir empfehlen Ihnen, eine separate Arbeitsgruppen für verschlüsselte und nicht-verschlüsselte Abfragen zu erstellen. Auf diese Weise können Sie eine Verschlüsselung für eine Arbeitsgruppe durchzusetzen, die dann für alle darin ausgeführten Abfragen gilt. Weitere Informationen finden Sie unter Verschlüsseln der in Amazon S3 gespeicherten Athena-Abfrageergebnisse.

2. Erstellen Sie nach Bedarf Arbeitsgruppen und fügen Sie diesen Tags hinzu. Informationen zu den erforderlichen Schritten finden Sie unter Erstellen von Arbeitsgruppen.
3. Erstellen Sie IAM-Richtlinien für Ihre Benutzer, Gruppen oder Rollen, um deren Zugriff auf Arbeitsgruppen zu aktivieren. Die Richtlinien legen die Arbeitsgruppenzugehörigkeit und den Zugriff auf Aktionen auf einer `workgroup`-Ressource fest. Die detaillierten Schritte finden Sie unter IAM-Richtlinien für den Zugriff auf Arbeitsgruppen. Beispiele für JSON-Richtlinien finden Sie unter Zugriff auf Arbeitsgruppen und Tags.
4. Richten Sie Arbeitsgruppeneinstellungen ein. Geben Sie in Amazon S3 einen Standort für Abfrageergebnisse an sowie optional den erwarteten Bucket-Eigentümer, die Verschlüsselungseinstellungen und die Steuerung von Objekten, die in den Abfrageergebnis-Bucket geschrieben werden. Sie können Arbeitsgruppeneinstellungen durchsetzen. Weitere Informationen finden Sie unter Arbeitsgruppeneinstellungen.

> **Important**
>
> Wenn Sie clientseitige Einstellungen überschreiben, wird Athena die Einstellungen der Arbeitsgruppe verwenden. Dies betrifft Abfragen, die Sie in der Konsole über die Treiber, die Befehlszeilenschnittstelle oder die API-Vorgänge ausführen.
> Während Abfragen weiterhin ausgeführt werden, kann die Automatisierung, die auf der Verfügbarkeit von Ergebnissen in einem bestimmten Amazon-S3-Bucket basiert, möglicherweise fehlschlagen. Wir empfehlen Ihnen, vor einer Überschreibung Ihre Benutzer zu informieren. Nachdem die Arbeitsgruppeneinstellungen für eine Überschreibung eingestellt wurden, können Sie die Angabe clientseitiger Einstellungen in den Treibern oder in der API auslassen.

5. Benachrichtigen Sie die Benutzer, welche Arbeitsgruppen sie für die Ausführung von Abfragen verwenden sollen. Senden Sie eine E-Mail, um die Benutzer Ihres Kontos über zu verwendende Arbeitsgruppennamen, die erforderlichen IAM-Richtlinien und die Arbeitsgruppeneinstellungen zu informieren.
6. Konfigurieren Sie Limits für die Kostenkontrolle – auch als Limits für die Datennutzungskontrolle bezeichnet – für Abfragen und Arbeitsgruppen. Erstellen Sie ein Amazon-SNS-Thema und konfigurieren Sie Abonnements, damit Sie bei Überschreitung eines Schwellenwerts benachrichtigt werden. Ausführliche Schritte finden Sie unter Festlegen von Limits zur Kontrolle der Datennutzung und Erste Schritte mit Amazon SNS im Entwicklerhandbuch für Amazon Simple Notification Service.
7. Wechseln Sie zu der Arbeitsgruppe, damit Sie Abfragen ausführen können. Um Abfragen auszuführen, wechseln Sie zu der entsprechenden Arbeitsgruppe. Die detaillierten Schritte finden Sie unter the section called “Angeben einer Arbeitsgruppe, in der Abfragen ausgeführt werden sollen”.

IAM-Richtlinien für den Zugriff auf Arbeitsgruppen

Verwenden Sie zur Steuerung des Zugriffs auf Arbeitsgruppen IAM-Berechtigungen auf Ressourcenebene oder identitätsbasierte IAM-Richtlinien. Wenn Sie IAM-Richtlinien verwenden, stellen Sie sicher, dass Sie die bewährten IAM-Methoden befolgen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

> **Note**
>
> Um auf Arbeitsgruppen zuzugreifen, für die die Verbreitung vertrauenswürdiger Identitäten aktiviert ist, müssen IAM-Identity-Center-Benutzer dem zugewiesen werden`IdentityCenterApplicationArn`, der von der Antwort der AthenaGetWorkGroup-API-Aktion zurückgegeben wird.

Das folgende Verfahren gilt speziell für Athena.

IAM-spezifische Informationen finden Sie unter den Links am Ende dieses Abschnitts. Weitere Informationen zu JSON-Arbeitsgruppen-Beispielrichtlinien finden Sie unter Beispiel-Arbeitsgruppenrichtlinien.

Verwenden Sie den visuellen Editor in der IAM-Konsole, um eine Arbeitsgruppenrichtlinie zu erstellen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich auf der linken Seite Policies (Richtlinien) und dann Create Policy (Richtlinie erstellen) aus.
3. Wählen Sie auf der Registerkarte Visual editor (Visueller Editor) die Option Choose a service (Wählen Sie einen Service) aus. Wählen Sie dann Athena aus, um es der Richtlinie hinzuzufügen.
4. Wählen Sie Select actions (Aktionen auswählen) und dann die Aktionen aus, die Sie der Richtlinie hinzufügen möchten. Im visuellen Editor werden die in Athena verfügbaren Aktionen angezeigt. Weitere Informationen finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Athena in der Service-Autorisierungs-Referenz.
5. Wählen Sie Add actions (Aktionen hinzufügen) aus, um eine bestimmte Aktion einzugeben, oder verwenden Sie Platzhalter (*), um mehrere Aktionen anzugeben.

   Standardmäßig lässt die Richtlinie, die Sie erstellen, die Aktionen zu, die Sie auswählen. Wenn Sie eine oder mehrere Aktionen auswählen, die Berechtigungen auf Ressourcenebene für die `workgroup`-Ressource in Athena unterstützen, listet der Editor die `workgroup`-Ressource auf.
6. Wählen Sie Resources (Ressourcen) aus, um die Arbeitsgruppen für Ihre Richtlinie anzugeben. Beispiele für JSON-Arbeitsgruppen-Richtlinien finden Sie unter Beispiel-Arbeitsgruppenrichtlinien.
7. Geben Sie die `workgroup`-Ressource wie folgt an:

```
arn:aws:athena:<region>:<user-account>:workgroup/<workgroup-name>
```

8. Wählen Sie Review policy (Richtlinie prüfen) aus und geben Sie Name und Description (Beschreibung) (optional) für die zu erstellende Richtlinie ein. Prüfen Sie die Richtlinienübersicht, um sicherzustellen, dass Sie die beabsichtigten Berechtigungen erteilt haben.
9. Wählen Sie Create policy (Richtlinie erstellen) aus, um Ihre neue Richtlinie zu speichern.
10. Hängen Sie diese identitätsbasierte Richtlinie an einen Benutzer, eine Gruppe oder eine Rolle an.

Weitere Informationen finden Sie in den folgenden Themen in der Service Authorization Reference und im IAM-Benutzerhandbuch:

- Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Athena
- Erstellen von Richtlinien mit dem visuellen Editor
- Hinzufügen und Entfernen von IAM-Richtlinien
- Steuern des Zugriffs auf Ressourcen

Beispiele für JSON-Arbeitsgruppen-Richtlinien finden Sie unter Beispiel-Arbeitsgruppenrichtlinien.

Eine vollständige Liste mit Amazon-Athena-Aktionen finden Sie unter den Namen von API-Aktionen in der Amazon-Athena-API-Referenz.

Beispiel-Arbeitsgruppenrichtlinien

In diesem Abschnitt sind Beispielrichtlinien enthalten, die Sie verwenden können, um verschiedene Aktionen in Arbeitsgruppen zu aktivieren. Wenn Sie IAM-Richtlinien verwenden, stellen Sie sicher, dass Sie die bewährten IAM-Methoden befolgen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Eine Arbeitsgruppe ist eine IAM-Ressource, die von Athena verwaltet wird. Wenn Ihre Arbeitsgruppenrichtlinie daher Aktionen verwaltet, die `workgroup` als Eingabe verwenden, müssen Sie den ARN der Arbeitsgruppe wie folgt angeben:

```
"Resource": [arn:aws:athena:<region>:<user-account>:workgroup/<workgroup-name>]
```

Hierbei ist *<workgroup-name>* der Name Ihrer Arbeitsgruppe. Eine Arbeitsgruppe mit dem Namen `test_workgroup` geben Sie beispielsweise wie folgt als Ressource an:

```
"Resource": ["arn:aws:athena:us-east-1:123456789012:workgroup/test_workgroup"]
```

Eine vollständige Liste mit Amazon-Athena-Aktionen finden Sie unter den Namen von API-Aktionen in der Amazon-Athena-API-Referenz. Weitere Informationen zu IAM-Richtlinien finden Sie unter Erstellen von Richtlinien mit dem visuellen Editor im IAM-Benutzerhandbuch. Weitere Informationen zum Erstellen von IAM-Richtlinien für Arbeitsgruppen finden Sie unter IAM-Richtlinien für den Zugriff auf Arbeitsgruppen.

- 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 Beispielrichtlinie für Vollzugriff auf alle Arbeitsgruppen

Die folgende Richtlinie erlaubt den vollständigen Zugriff auf alle möglicherweise in dem Konto vorhandenen Arbeitsgruppenressourcen. Wir empfehlen Ihnen, diese Richtlinie für die Benutzer in Ihrem Konto zu verwenden, die Arbeitsgruppen für alle anderen Benutzer verwalten müssen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:*"
            ],
            "Resource": [
                "*"
            ]
        }
```

```
    ]
}
```

Example Beispielrichtlinie für Vollzugriff auf eine angegebene Arbeitsgruppe

Die folgende Richtlinie gewährt vollständigen Zugriff auf eine einzelne Arbeitsgruppenressource namens `workgroupA`. Sie können diese Richtlinie für Benutzer mit vollständiger Kontrolle über eine bestimmte Arbeitsgruppe verwenden.

```
{
    "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 Beispielrichtlinie für die Ausführung von Abfragen in einer angegebenen Arbeitsgruppe

In der folgenden Richtlinie ist ein Benutzer berechtigt, Abfragen in der angegebenen `workgroupA` auszuführen und anzuzeigen. Der Benutzer darf keine Verwaltungsaufgaben für die Arbeitsgruppe selbst durchführen, sie also beispielsweise nicht aktualisieren oder löschen.

```
{
    "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 Beispielrichtlinie für die Ausführung von Abfragen in der primären Arbeitsgruppe

Sie können das vorherige Beispiel ändern, um es einem bestimmten Benutzer zu gestatten, auch Abfragen in der primären Arbeitsgruppe auszuführen.

> **Note**
>
> Wir empfehlen, dass Sie die primäre Arbeitsgruppenressource für alle Benutzer hinzufügen, die ansonsten für die Ausführung von Abfragen in ihren angegebenen Arbeitsgruppen konfiguriert sind. Es ist sinnvoll, diese Ressource ihren Arbeitsgruppen-Benutzerrichtlinien hinzuzufügen, falls ihre bezeichnete Arbeitsgruppe gelöscht oder deaktiviert wird. Sie können dann weiterhin Abfragen in der primären Arbeitsgruppe ausführen.

Um Benutzern in Ihrem Konto das Ausführen von Abfragen in der primären Arbeitsgruppe zu ermöglichen, fügen Sie wie im folgenden Beispiel eine Zeile mit dem ARN der primären Arbeitsgruppe zum Ressourcenabschnitt des Example policy for running queries in a specified workgroup hinzu.

```
arn:aws:athena:us-east-1:123456789012:workgroup/primary"
```

Example Beispielrichtlinie für Verwaltungsvorgänge in einer angegebenen Arbeitsgruppe

In der folgenden Richtlinie ist ein Benutzer berechtigt, eine Arbeitsgruppe zu erstellen, zu löschen, Details darüber abzurufen und die Arbeitsgruppe `test_workgroup` zu aktualisieren.

```
{
    "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 Beispielrichtlinie für die Auflistung von Arbeitsgruppen

Mit der folgenden Richtlinie erhalten alle Benutzer die Berechtigung, alle Arbeitsgruppen aufzulisten:

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListWorkGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

Example Beispielrichtlinie für die Ausführung und Anhalten von Abfragen in einer angegebenen Arbeitsgruppe

In dieser Richtlinie ist ein Benutzer berechtigt, Abfragen in der Arbeitsgruppe auszuführen:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:StartQueryExecution",
                "athena:StopQueryExecution"
            ],
            "Resource": [
                "arn:aws:athena:us-east-1:123456789012:workgroup/test_workgroup"
            ]
        }
    ]
}
```

Example Beispielrichtlinie für die Arbeit mit benannten Abfragen in einer angegebenen Arbeitsgruppe

In der folgenden Richtlinie verfügt ein Benutzer über Berechtigungen zum Erstellen, Löschen und Abrufen von Informationen über benannte Abfragen in der angegebenen Arbeitsgruppe:

```
{
    "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 -Beispielrichtlinie für die Nutzung von Spark-Notebooks in Athena

Verwenden Sie für die Arbeit mit Spark-Notebooks in Athena beispielsweise die folgende Richtlinie.

```
{
    "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": "*"
        }
    ]
}
```

Arbeitsgruppen-Einstellungen

Jede Arbeitsgruppe hat folgende Einstellungen:

- Einen eindeutigen Namen: Dieser kann zwischen 1 und 128 Zeichen umfassen, einschließlich alphanumerischer Zeichen, Bindestriche und Unterstriche. Der Name einer erstellten Arbeitsgruppe kann nicht nachträglich geändert werden. Sie können jedoch eine neue Arbeitsgruppe mit denselben Einstellungen und einem anderen Namen erstellen.
- Einstellungen, die für alle in der Arbeitsgruppe ausgeführten Abfragen gelten. Dazu gehören die Folgenden:
  - Ein Speicherort in Amazon S3 für Abfrageergebnisse aller Abfragen, die in dieser Arbeitsgruppe ausgeführt werden. Beim Erstellen der Arbeitsgruppe können Sie nur einen bereits vorhandenen Speicherort angeben. Informationen zum Erstellen eines Amazon-S3-Buckets finden Sie unter Create a Bucket (Erstellen eines Buckets).
  - Bucker-Eigentümer hat Kontrolle über Abfrageergebnisse – Ob der Eigentümer des Amazon-S3-Abfrageergebnis-Buckets die volle Kontrolle über neue Objekte hat, die in den Bucket

geschrieben werden. Wenn der Standort Ihres Abfrageergebnisses beispielsweise einem anderen Konto gehört, können Sie dem anderen Konto das Eigentum und die volle Kontrolle über Ihre Abfrageergebnisse gewähren.

- Erwarteter Bucket-Eigentümer – Die ID des AWS-Konto , von dem Sie erwarten, dass es der Eigentümer des Abfrageergebnis-Buckets ist. Dies ist eine zusätzliche Sicherheitsmaßnahme. Wenn die Konto-ID des Bucket-Eigentümers nicht mit der ID übereinstimmt, die Sie hier angeben, schlagen Versuche, in den Bucket auszugeben, fehl. Ausführliche Informationen finden Sie unter Überprüfen der Bucket-Eigentümerschaft mit Bucket-Eigentümer-Bedingung im Amazon-S3-Benutzerhandbuch.

  > **Note**
  >
  > Die erwartete Bucket-Eigentümereinstellung gilt nur für den Amazon-S3-Ausgabespeicherort, den Sie für Athena-Abfrageergebnisse angeben. Sie gilt nicht für andere Amazon-S3-Speicherorte wie Datenquellenspeicherorte in externen Amazon-S3-Buckets, `CTAS`- und `INSERT INTO`-Speicherorte der Zieltabelle, `UNLOAD`-Speicherorte der Anweisungsangaben, Vorgänge zum Verschütten von Buckets für Verbundabfragen oder `SELECT`-Abfragen, die für eine Tabelle in einem anderen Konto ausgeführt werden.

- Eine Verschlüsselungseinstellung, wenn Sie eine Verschlüsselung für alle Arbeitsgruppenabfragen verwenden. Sie haben nur die Möglichkeit, alle Abfragen in einer Arbeitsgruppe, nicht nur einige zu verschlüsseln. Am besten erstellen Sie separate Arbeitsgruppen für verschlüsselte und nicht verschlüsselte Abfragen.

Darüber hinaus kann Ihre Arbeitsgruppe clientseitige Einstellungen überschreiben. Vor der Freigabe von Arbeitsgruppen können Sie den Speicherort für Ergebnisse sowie Verschlüsselungsoptionen als Parameter im JDBC- oder ODBC-Treiber oder auf der Registerkarte Properties (Eigenschaften) in der Athena-Konsole angeben. Diese Einstellungen könnten auch direkt über die API-Vorgänge angegeben werden. Die Einstellungen werden auch als „clientseitige Einstellungen” bezeichnet. Bei Arbeitsgruppen können Sie diese Einstellungen auf Arbeitsgruppenebene konfigurieren, um die auf Client-Ebene verfügbaren Optionen zu steuern. Durch die Durchsetzung von Einstellungen auf Arbeitsgruppenebene müssen Benutzer ihre clientseitigen Einstellungen auch nicht einzeln konfigurieren. Wenn Sie Clientseitige Einstellungen überschreiben auswählen, verwenden die Abfragen die Arbeitsgruppeneinstellungen und ignorieren die clientseitigen Einstellungen.

Bei Auswahl von Override Client-Side Settings (Clientseitige Einstellungen überschreiben) werden die Benutzer auf der Konsole darüber informiert, dass ihre Einstellungen geändert wurden. Wenn

die Arbeitsgruppeneinstellungen auf diese Weise durchgesetzt werden, können die Benutzer die entsprechenden clientseitigen Einstellungen auslassen. Abfragen, die in der Konsole ausgeführt werden, verwenden dann die Einstellungen der Arbeitsgruppe, auch wenn clientseitige Einstellungen vorhanden sind. Wenn Abfragen in der Arbeitsgruppe über die AWS CLI, API-Operationen oder JDBC- oder ODBC-Treiber ausgeführt werden, werden außerdem clientseitige Einstellungen wie Speicherort und Verschlüsselung der Abfrageergebnisse durch Arbeitsgruppeneinstellungen überschrieben. Um die Einstellungen für die Arbeitsgruppe zu sehen, rufen Sie die Details der Arbeitsgruppe auf.

Sie können auch Abfragelimits für Abfragen in Arbeitsgruppen festlegen.

Arbeitsgruppen-Einstellungen überschreiben clientseitige Einstellungen

In den Dialogfenstern Create workgroup (Arbeitsgruppe erstellen) und Edit workgroup (Arbeitsgruppe bearbeiten) gibt es ein Feld mit der Bezeichnung Override client-side settings (Clientseitige Einstellungen überschreiben). Diese Funktion ist standardmäßig nicht ausgewählt. Je nachdem, ob Sie das Feld auswählen, führt Athena die folgenden Schritte aus:

- Wenn Clientseitige Einstellungen überschreiben nicht ausgewählt ist, werden die Arbeitsgruppeneinstellungen nicht durchgesetzt. Wenn die Option „Clientseitige Einstellungen überschreiben“ für die Arbeitsgruppe nicht ausgewählt ist, verwendet Athena die Einstellungen des Clients für alle Abfragen, die in der Arbeitsgruppe ausgeführt werden, einschließlich der Einstellungen für den Abfrageergebnisspeicherort, den erwarteten Bucket-Besitzer, die Verschlüsselung und die Steuerung von Objekten, welche in den Abfrageergebnis-Bucket geschrieben werden. Jeder Benutzer kann clientseitige Einstellungen im Menü Einstellungen auf der Konsole angeben. Wenn die clientseitigen Einstellungen nicht verwendet werden, gelten die arbeitsgruppenweiten Einstellungen. Wenn Sie die AWS CLI, API-Aktionen oder JDBC- und ODBC-Treiber verwenden, um Abfragen in einer Arbeitsgruppe auszuführen, die keine clientseitigen Einstellungen überschreibt, verwenden Ihre Abfragen die Einstellungen, die Sie in Ihren Abfragen angeben.
- Wenn Clientseitige Einstellungen überschreiben ausgewählt ist, werden die Arbeitsgruppeneinstellungen auf Arbeitsgruppenebene für alle Clients in der Arbeitsgruppe durchgesetzt. Wenn die Option „Clientseitige Einstellungen überschreiben“ für die Arbeitsgruppe ausgewählt ist, verwendet Athena die Einstellungen der Arbeitsgruppe für alle Abfragen, die in der Arbeitsgruppe ausgeführt werden, einschließlich der Einstellungen für den Abfrageergebnisspeicherort, den erwarteten Bucket-Besitzer, die Verschlüsselung und die Steuerung von Objekten, welche in den Abfrageergebnis-Bucket geschrieben werden.

Arbeitsgruppeneinstellungen haben Vorrang vor allen clientseitigen Einstellungen, die Sie für eine Abfrage angeben, wenn Sie die Konsole, API-Aktionen oder JDBC- und ODBC-Treiber verwenden.

Wenn Sie die clientseitigen Einstellungen überschreiben, werden Sie beim nächsten Öffnen der Athena-Konsole durch Sie oder einen beliebigen Arbeitsgruppenbenutzer darüber informiert, dass Abfragen in der Arbeitsgruppe die Einstellungen der Arbeitsgruppe verwenden, und Sie werden aufgefordert, diese Änderung zu bestätigen.

> **Important**
>
> Wenn Sie API-Aktionen, die oder die JDBC- und ODBC-Treiber verwenden AWS CLI, um Abfragen in einer Arbeitsgruppe auszuführen, die clientseitige Einstellungen überschreibt, stellen Sie sicher, dass Sie entweder die clientseitigen Einstellungen in Ihren Abfragen weglassen oder sie so aktualisieren, dass sie den Einstellungen der Arbeitsgruppe entsprechen. Wenn Sie in Ihren Abfragen clientseitige Einstellungen angeben, diese aber in einer Arbeitsgruppe ausführen, die die Einstellungen überschreibt, werden die Abfragen ausgeführt, aber die Arbeitsgruppeneinstellungen werden verwendet. Informationen zum Anzeigen der Einstellungen für eine Arbeitsgruppe finden Sie unter Anzeigen der Arbeitsgruppen-Details.

Verwalten von Arbeitsgruppen

In https://console.aws.amazon.com/athena/ können Sie die folgenden Aufgaben ausführen:

Statement	Beschreibung
Erstellen von Arbeitsgruppen	Erstellen einer neuen Arbeitsgruppe.
Bearbeiten von Arbeitsgruppen	Bearbeiten einer Arbeitsgruppe und Ändern ihrer Einstellungen. Sie können den Namen einer Arbeitsgruppe nicht ändern, aber eine neue Arbeitsgruppe mit denselben Einstellungen und einem anderen Namen erstellen.
Anzeigen der Arbeitsgruppen-Details	Rufen Sie die Details der Arbeitsgruppe auf, z. B. Name, Beschreibung, Limits für die Datennutzung, Speicherort von Abfrageergebnissen, erwarteter Abfrageergebnis-Bucket-Eigentümer, Verschlüsselung und

Statement	Beschreibung
	Steuerung von Objekten, die in den Abfrageergebnis-Bucket geschrieben werden. Sie können auch überprüfen, ob diese Arbeitsgruppe ihre Einstellungen erzwingt, wenn Override client-side settings (Clientseitige Einstellungen überschreiben) markiert ist.
Löschen von Arbeitsgruppen	Löschen von Arbeitsgruppen. Wenn Sie eine Arbeitsgruppe löschen, werden der Abfrageverlauf, die gespeicherten Abfragen, die Einstellungen der Arbeitsgruppe und die Limits zur Kontrolle der Datennutzung ebenfalls gelöscht. Die arbeitsgruppenweiten Datenlimit-Steuerelemente verbleiben in CloudWatchund Sie können sie einzeln löschen. Die primäre Arbeitsgruppe kann nicht gelöscht werden.
So wechseln Sie Arbeitsgruppen	Wechseln zwischen Arbeitsgruppen, auf die Sie Zugriff haben.
Kopieren einer gespeicherte Abfrage zwischen Arbeitsgruppen	Kopieren Sie eine gespeicherte Abfrage zwischen Arbeitsgruppen. Sie können dies beispielsweise tun, wenn Sie eine Abfrage in einer Vorschauarbeitsgruppe erstellt haben und sie in einer Arbeitsgruppe ohne Vorschauansicht verfügbar machen möchten.
Aktivieren und Deaktivieren von Arbeitsgruppen	Aktivieren oder Deaktivieren einer Arbeitsgruppe. Wenn eine Arbeitsgruppe deaktiviert wird, können ihre Benutzer keine Abfragen ausführen oder neue benannte Abfragen erstellen. Wenn Sie auf sie zugreifen können, können Sie weiter Metriken, Limits zur Kontrolle der Datennutzung, Arbeitsgruppeneinstellungen, den Abfrageverlauf und gespeicherte Abfragen anzeigen.
Angeben einer Arbeitsgruppe, in der Abfragen ausgeführt werden sollen	Bevor Sie Abfragen ausführen können, müssen Sie Athena angeben, welche Arbeitsgruppe verwendet werden soll. Sie müssen über Berechtigungen für die Arbeitsgruppe verfügen.

Statement	Beschreibung
Erstellen einer Athena-Arbeitsgruppe, die die IAM-Identity-Center-Authentifizierung verwendet	Um IAM-Identity-Center-Identitäten mit Athena verwenden zu können, müssen Sie eine für IAM Identity Center aktivierte Arbeitsgruppe erstellen. Nachdem Sie die Arbeitsgruppe erstellt haben, können Sie die IAM-Identity-Center-Konsole oder -API verwenden, um der Arbeitsgruppe IAM-Identity-Center-Benutzer oder -Gruppen zuzuweisen.

Erstellen von Arbeitsgruppen

Das Erstellen einer Arbeitsgruppe erfordert Berechtigungen für `CreateWorkgroup`-API-Aktionen. Siehe Zugriff auf Arbeitsgruppen und Tags und IAM-Richtlinien für den Zugriff auf Arbeitsgruppen. Wenn Sie Tags hinzufügen, müssen Sie auch Berechtigungen für `TagResource` hinzufügen. Siehe Tag-Richtlinienbeispiele für Arbeitsgruppen.

So erstellen Sie eine Arbeitsgruppe in der Konsole

1. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.

2. Wählen Sie im Navigationsbereich der Athena-Konsole Workgroups (Arbeitsgruppen) aus.
3. Wählen Sie auf der Seite Workgroups (Arbeitsgruppen) die Option Create workgroup (Arbeitsgruppe erstellen) aus.
4. Füllen Sie auf der Seite Create workgroup (Arbeitsgruppe erstellen) die Felder wie folgt aus:

Feld	Beschreibung
Workgroup name (Name der Arbeitsgruppe)	Erforderlich Geben Sie einen eindeutigen Namen für Ihre Arbeitsgruppe ein. Verwenden Sie zwischen 1 und 128 Zeichen. (A – Z, a – z, 0 – 9, _, -, .). Dieser Name kann nicht geändert werden.
Beschreibung	Optional. Geben Sie eine Beschreibung für Ihre Arbeitsgruppe ein. Sie kann bis zu 1024 Zeichen enthalten.
Choose the type of engine (Engine-Typ auswählen)	Wählen Sie Athena SQL, wenn Sie Ad-hoc-SQL-Abfragen für Daten in Amazon S3 ausführen möchten, oder verwenden Sie einen vorgefertigten Datenquellen-Konnektor, um Verbundabfragen für eine Vielzahl von Datenquellen außerhalb von Amazon S3 auszuführen. Sie können Abfragen mit dem Athena-Abfrageeditor, AWS CLI oder mit Athena-APIs ausführen. Benutzen Sie Apache Spark, wenn Sie Jupyter-Notebook-Anwendungen mit Python und Apache Spark erstellen, bearbeiten und ausführen möchten. Jupyter Notebooks enthalten eine Liste von Zellen, die Code, Text, Markdown, Mathematik, Darstellungen und Multimedia enthalten können. In einer interaktiven Notebook-Sitzung in Athena werden die Zellen der Reihe nach als Berechnungen ausgeführt. Informationen zum Erstellen und Konfigurieren einer Spark-fähigen Arbeitsgruppe finden Sie unter Erstellen einer Spark-fähigen Arbeitsgruppe in Athena. Nachdem Sie eine Arbeitsgruppe erstellt haben, kann deren Analyse-Engine aktualisiert werden (z. B. von Athena-Engine-Version 2 auf Athena-Engine-Version 3). Der Engine-Typ kann jedoch nicht geändert werden. Beispielsweise kann eine Arbeitsgruppe der Athena-Engine-Version 3 nicht in eine Arbeitsgruppe der PySpark Engine-Version 3 geändert werden.

Feld	Beschreibung
Abfrage-Engine aktualisieren	Wählen Sie aus, wie Sie Ihre Arbeitsgruppe aktualisieren möchten, wenn eine neue Athena-Engine-Version veröffentlicht wird. Sie können Athena entscheiden lassen, wann Sie Ihre Arbeitsgruppe aktualisieren oder manuell eine Engine-Version auswählen. Weitere Informationen finden Sie unter Athena-Engine-Versionierung.
Authentifizierungsmodus	Wählen Sie AWS Identity and Access Management -(IAM), um die IAM-Authentifizierung oder den Verbund für die Arbeitsgruppe zu verwenden. Wählen Sie IAM Identity Center, wenn Sie Workforce -Identitäten wie Benutzer und Gruppen von SAML-2.0-Identität sanbietern wie Microsoft Active Directory unterstützen möchten. Weitere Informationen finden Sie unter Vertrauenswürdige Identität sweitergabe über Anwendungen hinweg im AWS IAM Identity Center -Benutzerhandbuch.
Servicerolle für den Zugriff auf das IAM Identity Center	Athena benötigt IAM-Berechtigungen, um in Ihrem Namen auf IAM Identity Center zugreifen zu können. Weitere Informationen zu IAM-Servicerollen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS -Service im IAM-Benutzerhandbuch.

Feld	Beschreibung
Speicherort des Abfrageergebnisses	Optional. Geben Sie einen Pfad zu einem Amazon-S3-Bucket oder -Präfix ein. Dieser Bucket und dieses Präfix müssen bereits vorhanden sein, bevor Sie sie angeben können. **Note** Wenn Sie Abfragen in der Konsole ausführen, ist die Angabe des Speicherorts von Abfrageergebnissen optional. Wenn Sie ihn für die Arbeitsgruppe oder in Settings (Einstell ungen) nicht angeben, verwendet Athena den Standards peicherort für Abfrageergebnisse. Wenn Sie Abfragen mit der API oder den Treibern ausführen, müssen Sie den Speicherort der Abfrageergebnisse an mindestens einem der beiden Stellen angeben: für einzelne Abfragen mit OutputLocationoder für die Arbeitsgruppe mit WorkGroup Configuration.

Feld	Beschreibung
Erwarteter Bucket-Eigentümer	Optional. Geben Sie die ID des ein AWS-Konto , von dem Sie erwarten, dass es der Eigentümer des Ausgabespeicherort-Buckets ist. Dies ist eine zusätzliche Sicherheitsmaßnahme. Wenn die Konto-ID des Bucket-Eigentümers nicht mit der ID übereinstimmt, die Sie hier angeben, schlagen Versuche, in den Bucket auszugeben, fehl. Ausführliche Informationen finden Sie unter Überprüfen der Bucket-Eigentümerschaft mit Bucket-Eigentümer-Bedingung im Amazon-S3-Benutzerhandbuch. **Note** Die erwartete Bucket-Eigentümereinstellung gilt nur für den Amazon-S3-Ausgabespeicherort, den Sie für Athena-Abfrageergebnisse angeben. Sie gilt nicht für andere Amazon-S3-Speicherorte wie Datenquellenspeicherorte in externen Amazon-S3-Buckets, `CTAS`- und `INSERT INTO`-Speicherorte der Zieltabelle, `UNLOAD`-Speicherorte der Anweisung sangaben, Vorgänge zum Verschütten von Buckets für Verbundabfragen oder `SELECT`-Abfragen, die für eine Tabelle in einem anderen Konto ausgeführt werden.

Feld	Beschreibung
Assign bucket owner full control over query results (Bucket-Eigentümer die volle Kontrolle über Abfrageergebnisse zuweisen)	Diese Funktion ist standardmäßig nicht ausgewählt. Wenn Sie sie auswählen und für den Speicherort des Abfrageergebnis-Bucket ACLs aktiviert sind, gewähren Sie dem Bucket-Eigentümer die volle Kontrolle über Abfrageergebnisse. Wenn der Standort Ihres Abfrageergebnisses beispielsweise einem anderen Konto gehört, können Sie diese Option nutzen, um dem anderen Konto das Eigentum und die volle Kontrolle über Ihre Abfrageergebnisse zu gewähren. Wenn die Einstellung zur S3-Objekteigentümerschaft des Bucket Bucket owner preferred (Bucket-Eigentümer bevorzugt) lautet, besitzt der Bucket-Eigentümer auch alle Abfrageergebnisobjekte, die aus dieser Arbeitsgruppe geschrieben wurden. Wenn beispielsweise die Arbeitsgruppe eines externen Kontos diese Option aktiviert und den Speicherort des Abfrageergebnisses auf den Amazon-S3-Bucket Ihres Kontos festlegt, der eine Einstellung zur S3-Objekteigentümerschaft von Bucket owner preferred (Bucket-Eigentümer bevorzugt) hat, sind Sie Eigetümer der Abfrageergebnisse der externen Arbeitsgruppe und haben die volle Kontrolle über sie. Wenn Sie diese Option auswählen, wenn die Einstellung zur S3-Objekteigentümerschaft des Bucket Bucket owner enforced (Bucket-Eigentümer durchgesetzt) ist, hat dies keine Auswirkungen. Weitere Informationen finden Sie unter Einstellungen zur Objekteigentümerschaft im Amazon-S3-Benutzerhandbuch.

Feld	Beschreibung
Encrypt query results (Abfrageergebnisse verschlüsseln)	Optional. Verschlüsseln der in Amazon S3 gespeicherten Ergebnisse. Bei Auswahl werden alle Abfragen in der Arbeitsgruppe verschlüsselt. Bei Auswahl dieser Option können Sie den Encryption type (Verschlüsselungstyp) und den Encryption key (Verschlüsselungsschlüssel) auswählen und den KMS Key ARN (KMS-Schlüssel-ARN) eingeben. Wenn Sie keinen Schlüssel besitzen, öffnen Sie die AWS KMS -Konsole, um einen Schlüssel zu erstellen. Weitere Informationen finden Sie unter Erstellen von Schlüsseln im AWS Key Management Service -Entwicklerhandbuch.
Stellen Sie ***encryption_type*** als Mindestverschlüsselung ein	Optional. Wählen Sie diese Option, um einen Mindestverschlüsselungstyp für Abfrageergebnisse für alle Benutzer der Arbeitsgruppe zu erzwingen. Wenn Sie diese Option auswählen, wird eine Tabelle mit der Hierarchie der Verschlüsselungstypen angezeigt. Die Tabelle zeigt Ihnen auch, welche Verschlüsselungstypen Arbeitsgruppenbenutzer verwenden dürfen, wenn Sie einen bestimmten Verschlüsselungstyp als Minimum angeben. Um diese Option verwenden zu können, darf die Option Clientseitige Einstellungen überschreiben nicht aktiviert sein. Weitere Informationen finden Sie unter Konfiguration der Mindestverschlüsselung für eine Arbeitsgruppe.
Aktivieren von S3-Zugriffsberechtigungen	Dieses Feld ist standardmäßig ausgewählt, wenn Sie IAM Identity Center als Authentifizierungsmodus wählen. Wenn diese Option ausgewählt ist, wendet sie benutzer- oder gruppenbasierte IAM-Identity-Center-Berechtigungen auf Amazon-S3-Standorte an.
Erstellen eines auf Benutzeridentitäten basierenden S3-Präfixes	Wenn diese Option ausgewählt ist, erstellt Athena beim Speichern von Abfrageergebnissen ein Amazon-S3-Präfix. Das Präfix basiert auf der Benutzeridentität des IAM Identity Center des Benutzers.

Feld	Beschreibung
Veröffentlichen von Abfragemetriken in CloudWatch	Dieses Feld ist standardmäßig ausgewählt. Veröffentlichen von Abfragemetriken zu CloudWatch. Siehe Überwachung von Athena-Abfragen mit CloudWatch Metriken.
Override client-side settings (Clientse itige Einstellungen überschreiben)	Diese Funktion ist standardmäßig nicht ausgewählt. Bei Auswahl werden die Arbeitsgruppeneinstellungen für alle Abfragen in der Arbeitsgruppe übernommen und die clientseitigen Einstellungen überschrieben. Weitere Informationen finden Sie unter Arbeitsgr uppen-Einstellungen überschreiben clientseitige Einstellungen.
S3-Buckets mit Zahlung durch den Anforderer	Optional. Wählen Sie Turn on queries on requester pays buckets in Amazon S3 (Abfragen zu Buckets mit Zahlung durch den Anforderer in Amazon S3 aktivieren), wenn Arbeitsgruppenbenu tzer Abfragen zu Daten ausführen, die in Amazon S3-Buckets gespeichert sind, die als Zahlung durch den Anforderer konfiguriert sind. Dem Konto des Benutzers, der die Abfrage ausführt, werden die entsprechenden Datenzugriffs- und Datenübertragungsg ebühren in Rechnung gestellt, die mit der Abfrage verbunden sind. Weitere Informationen finden Sie unter Buckets mit Zahlung durch den Anforderer im Benutzerhandbuch zu Amazon Simple Storage Service.
Manage per query data usage control (Datennutzungskont rolle pro Abfrage verwalten)	Optional. Legt das Limit für die maximale Datenmenge fest, die eine Abfrage scannen darf. Sie können nur ein Limit pro Abfrage für eine Arbeitsgruppe festlegen. Das Limit gilt für alle Abfragen in der Arbeitsgruppe und wenn die Abfrage das Limit überschreitet, wird sie abgebrochen. Weitere Informationen finden Sie unter Festlegen von Limits zur Kontrolle der Datennutzung.

Feld	Beschreibung
Workgroup data usage alerts (Warnungen zur Datennutzung von Arbeitsgruppen)	Optional. Legen Sie mehrere Warnungsschwellenwerte fest, wenn Abfragen, die in dieser Arbeitsgruppe ausgeführt werden, eine bestimmte Datenmenge innerhalb eines bestimmten Zeitraums scannen. Warnungen werden mithilfe von Amazon- CloudWatch Alarmen implementiert und gelten für alle Abfragen in der Arbeitsgruppe. Weitere Informationen finden Sie unter Verwenden von Amazon- CloudWatch Alarmen im Amazon- CloudWatch Benutzerhandbuch.
Tags	Optional. Hinzufügen von einem oder mehreren Tags zu einer Arbeitsgruppe. Ein Tag ist eine Markierung, die Sie einer Athena-Arbeitsgruppenressource zuordnen. Sie besteht aus einem Schlüssel und einem Wert. Verwenden Sie AWS bewährte Methoden für die Markierung, um einen konsistenten Satz von Tags zu erstellen und Arbeitsgruppen nach Zweck, Eigentümer oder Umgebung zu kategorisieren. Sie können Tags auch in IAM-Richtlinien und zur Kontrolle von Abrechnungskosten verwenden. Verwenden Sie keine doppelten Tag-Schlüssel in derselben Arbeitsgruppe. Weitere Informationen finden Sie unter the section called “Markieren von Ressourcen”.

5. Wählen Sie Create workgroup (Arbeitsgruppe erstellen) aus. Die Arbeitsgruppe wird in der Liste auf der Seite Workgroups (Arbeitsgruppen) angezeigt.

Sie können auch die CreateWorkGroup -API-Operation verwenden, um eine Arbeitsgruppe zu erstellen.

> **Important**
>
> Nachdem Sie Arbeitsgruppen erstellt haben, erstellen Sie IAM-Richtlinien für den Zugriff auf Arbeitsgruppen-IAM, mit dem Sie arbeitsgruppenbezogene Aktionen ausführen können.

Bearbeiten von Arbeitsgruppen

Das Bearbeiten einer Arbeitsgruppe erfordert Berechtigungen für `UpdateWorkgroup`-API-Vorgänge. Siehe Zugriff auf Arbeitsgruppen und Tags und IAM-Richtlinien für den Zugriff auf Arbeitsgruppen. Wenn Sie Tags hinzufügen oder bearbeiten, müssen Sie auch Berechtigungen für `TagResource` besitzen. Siehe Tag-Richtlinienbeispiele für Arbeitsgruppen.

So bearbeiten Sie eine Arbeitsgruppe in der Konsole

1. Wählen Sie im Navigationsbereich der Athena-Konsole Workgroups (Arbeitsgruppen) aus.
2. Klicken Sie auf der Seite Workgroups (Arbeitsgruppen) auf die Schaltfläche für die Arbeitsgruppe, die Sie bearbeiten möchten.
3. Wählen Sie Actions (Aktionen) und Edit (Bearbeiten).
4. Ändern Sie die Felder wie gewünscht. Die Liste der Felder finden Sie unter Create workgroup (Arbeitsgruppe erstellen). Sie können alle Felder mit Ausnahme des Namens der Arbeitsgruppe ändern. Wenn Sie den Namen ändern müssen, erstellen Sie eine andere Arbeitsgruppe mit dem neuen Namen und denselben Einstellungen.
5. Wählen Sie Save Changes (Änderungen speichern). Die aktualisierte Arbeitsgruppe wird in der Liste auf der Seite Workgroups (Arbeitsgruppen) angezeigt.

Anzeigen der Arbeitsgruppen-Details

Sie können die Details jeder Arbeitsgruppe anzeigen. Die Details umfassen den Namen der Arbeitsgruppe, die Beschreibung der Arbeitsgruppe, die Angabe, ob sie aktiviert oder deaktiviert ist, sowie die für in der Arbeitsgruppe ausgeführte Abfragen verwendeten Einstellungen. Zu diesen gehören der Speicherort der Abfrageergebnisse, der erwartete Bucket-Eigentümer, die Verschlüsselung und die Steuerung von Objekten, die in den Abfrageergebnis-Bucket geschrieben werden. Wenn es für eine Arbeitsgruppe Limits für die Datennutzung gibt, werden diese ebenfalls angezeigt.

So zeigen Sie die Arbeitsgruppendetails an

1. Wählen Sie im Navigationsbereich der Athena-Konsole Workgroups (Arbeitsgruppen) aus.
2. Wählen Sie auf der Seite der Workgroups (Arbeitsgruppen) den Link der Arbeitsgruppe aus, die Sie anzeigen möchten. Die Seite Overview Details (Übersicht über Details) für die Arbeitsgruppe wird angezeigt.

Löschen von Arbeitsgruppen

Sie können eine Arbeitsgruppe löschen, wenn Sie die entsprechende Berechtigung besitzen. Die primäre Arbeitsgruppe kann nicht gelöscht werden.

Wenn Sie die Berechtigungen besitzen, können Sie eine leere Arbeitsgruppe jederzeit löschen. Sie können auch eine Arbeitsgruppe löschen, die gespeicherte Abfragen enthält. In diesem Fall werden Sie vor dem Löschen der Arbeitsgruppe von Athena gewarnt, dass gespeicherte Abfragen gelöscht werden.

Wenn Sie eine Arbeitsgruppe löschen, während Sie sich in der Arbeitsgruppe befinden, wechselt die Konsole zur primären Arbeitsgruppe. Wenn Sie auf diese zugreifen können, können Sie Abfragen ausführen und ihre Einstellungen anzeigen.

Wenn Sie eine Arbeitsgruppe löschen, werden ihre Einstellungen und die Limits zur Kontrolle der Datennutzung pro Abfrage gelöscht. Die arbeitsgruppenweiten Datenlimit-Steuerelemente verbleiben in CloudWatchund Sie können sie dort bei Bedarf löschen.

> **Important**
>
> Stellen Sie vor dem Löschen einer Arbeitsgruppe sicher, dass ihre Benutzer auch zu anderen Arbeitsgruppen gehören, in denen sie weiter Abfragen ausführen können. Wenn die IAM-Richtlinien der Benutzer diese ausschließlich zur Ausführung von Abfragen in dieser Arbeitsgruppe berechtigt haben und Sie diese Arbeitsgruppe löschen, sind sie nicht mehr zum Ausführen von Abfragen berechtigt. Weitere Informationen finden Sie unter Example policy for running queries in the primary workgroup.

So löschen Sie eine Arbeitsgruppe in der Konsole

1. Wählen Sie im Navigationsbereich der Athena-Konsole Workgroups (Arbeitsgruppen) aus.
2. Klicken Sie auf der Seite Workgroups (Arbeitsgruppen) auf die Schaltfläche für die Arbeitsgruppe, die Sie löschen möchten.
3. Wählen Sie Actions (Aktionen), Delete (Löschen) aus.
4. Geben Sie an der Bestätigungsaufforderung Delete workgroup (Arbeitsgruppe löschen) den Namen der Arbeitsgruppe ein und wählen Sie dann Delete (Löschen) aus.

Um eine Arbeitsgruppe mit dem API-Vorgang zu löschen, verwenden Sie die Aktion `DeleteWorkGroup`.

So wechseln Sie Arbeitsgruppen

Sie können von einer Arbeitsgruppe zu einer anderen wechseln, wenn Sie für beide Arbeitsgruppen Berechtigungen besitzen.

Sie können innerhalb jeder Arbeitsgruppe bis zu zehn Abfrageregisterkarten öffnen. Wenn Sie zwischen Arbeitsgruppen wechseln, bleiben Ihre Abfrageregisterkarten für bis zu drei Arbeitsgruppen geöffnet.

So wechseln Sie Arbeitsgruppen

1. Verwenden Sie in der Athena-Konsole die Option Arbeitsgruppe rechts oben, um eine Arbeitsgruppe auszuwählen.
2. Wenn das Dialogfeld Einstellungen für ***Arbeitsgruppennamen*** für Arbeitsgruppe angezeigt wird, wählen Sie Bestätigen.

Die Option Arbeitsgruppe zeigt den Namen der Arbeitsgruppe an, zu der Sie gewechselt haben. Sie können nun Abfragen in dieser Arbeitsgruppe ausführen.

Kopieren einer gespeicherte Abfrage zwischen Arbeitsgruppen

Derzeit verfügt die Athena-Konsole nicht über die Möglichkeit, eine gespeicherte Abfrage direkt von einer Arbeitsgruppe in eine andere zu kopieren, aber Sie können dieselbe Aufgabe manuell ausführen, indem Sie das folgende Verfahren verwenden.

So kopieren Sie eine gespeicherte Abfrage zwischen Arbeitsgruppen

1. Wählen Sie in der Athena-Konsole aus der Arbeitsgruppe, aus der Sie die Abfrage kopieren möchten, die Registerkarte Saved queries (Gespeicherte Abfragen) aus.
2. Wählen Sie den Link der gespeicherten Abfrage aus, die Sie kopieren möchten. Athena öffnet die Abfrage im Abfrage-Editor.
3. Wählen Sie im Abfrage-Editor den Abfragetext aus, und drücken Sie dann **`Ctrl+C`**, um ihn zu kopieren.
4. Wechseln Sie zur Zielarbeitsgruppe oder Create workgroup (Arbeitsgruppe erstellen) und wechseln Sie dann zu dieser.

5. Öffnen Sie eine neue Registerkarte im Abfrage-Editor und drücken Sie **`Ctrl+V`**, um den Text in die neue Registerkarte einzufügen.
6. Wählen Sie im Abfrage-Editor Save as (Speichern unter) aus, um die Abfrage in der Zielarbeitsgruppe zu speichern.
7. Geben Sie im Dialogfeld Namen auswählen einen Namen für die Abfrage und eine optionale Beschreibung ein.
8. Wählen Sie Save (Speichern) aus.

Aktivieren und Deaktivieren von Arbeitsgruppen

Wenn Sie über die entsprechenden Berechtigungen verfügen, können Sie in der Konsole mittels der API-Vorgänge oder JDBC- oder ODBC-Treiber Arbeitsgruppen aktivieren oder deaktivieren.

So aktivieren oder deaktivieren Sie eine Arbeitsgruppe

1. Wählen Sie im Navigationsbereich der Athena-Konsole Workgroups (Arbeitsgruppen) aus.
2. Wählen Sie auf der Seite Workgroups (Arbeitsgruppen) den Link für die Arbeitsgruppe aus.
3. Wählen Sie rechts oben Aktivieren von Arbeitsgruppe oder Deaktivieren von Arbeitsgruppe aus.
4. Wählen Sie an der Bestätigungsaufforderung Aktivieren oder Deaktivieren von aus. Wenn Sie eine Arbeitsgruppe deaktivieren, können ihre Benutzer keine Abfragen in ihr ausführen oder neue benannte Abfragen erstellen. Wenn Sie eine Arbeitsgruppe aktivieren, können Benutzer sie zum Ausführen von Abfragen verwenden.

Angeben einer Arbeitsgruppe, in der Abfragen ausgeführt werden sollen

Um eine Arbeitsgruppe anzugeben, die verwendet werden soll, müssen Sie über Berechtigungen für die Arbeitsgruppe verfügen.

So geben Sie die zu verwendende Arbeitsgruppe an

1. Stellen Sie sicher, dass Ihre Berechtigungen Ihnen das Ausführen von Abfragen in der Arbeitsgruppe ermöglichen, die Sie verwenden möchten. Weitere Informationen finden Sie unter the section called “ IAM-Richtlinien für den Zugriff auf Arbeitsgruppen”.
2. Verwenden Sie eine der folgenden Optionen, um die Arbeitsgruppe anzugeben:

   - Wenn Sie die Athena-Konsole verwenden, legen Sie die Arbeitsgruppe über Wechseln zwischen Arbeitsgruppen fest.

- Wenn Sie die Athena-API-Vorgänge verwenden, geben Sie den Arbeitsgruppennamen in der API-Aktion an. Sie können beispielsweise den Namen der Arbeitsgruppe in StartQueryExecutionwie folgt festlegen:

```
StartQueryExecutionRequest startQueryExecutionRequest = new
 StartQueryExecutionRequest()
              .withQueryString(ExampleConstants.ATHENA_SAMPLE_QUERY)
              .withQueryExecutionContext(queryExecutionContext)
              .withWorkGroup(WorkgroupName)
```

- Wenn Sie den JDBC- oder ODBC-Treiber verwenden, legen Sie den Arbeitsgruppennamen mithilfe des `Workgroup`-Konfigurationsparameters in der Verbindungszeichenfolge fest. Der Treiber übergibt den Namen der Arbeitsgruppe an Athena. Sie geben die Arbeitsgruppenparameter in der Verbindungszeichenfolge wie im folgenden Beispiel gezeigt an:

```
jdbc:awsathena://AwsRegion=<AWSREGION>;UID=<ACCESSKEY>;
PWD=<SECRETKEY>;S3OutputLocation=s3://<athena-output>-<AWSREGION>/;
Workgroup=<WORKGROUPNAME>;
```

Konfiguration der Mindestverschlüsselung für eine Arbeitsgruppe

Als Administrator einer Athena-SQL-Arbeitsgruppe können Sie eine minimale Verschlüsselungsstufe in Amazon S3 für alle Abfrageergebnisse der Arbeitsgruppe erzwingen. Sie können dieses Feature verwenden, um sicherzustellen, dass Abfrageergebnisse niemals unverschlüsselt in einem Amazon-S3-Bucket gespeichert werden.

Wenn Benutzer in einer Arbeitsgruppe mit aktivierter Mindestverschlüsselung eine Abfrage einreichen, können sie die Verschlüsselung nur auf die von Ihnen konfigurierte Mindeststufe oder auf eine höhere Stufe einstellen, falls eine verfügbar ist. Athena verschlüsselt Abfrageergebnisse entweder auf der Ebene, die angegeben wurde, wenn der Benutzer die Abfrage ausführt, oder auf der Ebene, die in der Arbeitsgruppe festgelegt wurde.

Die folgenden Stufen sind verfügbar:

- Basis – Serverseitige Verschlüsselung mit von Amazon S3 verwalteten Schlüsseln (SSE-S3)
- Mittelstufe – Serverseitige Verschlüsselung mit KMS-verwalteten Schlüsseln (SSE-KMS).
- Erweitert – Clientseitige Verschlüsselung mit von KMS verwalteten Schlüsseln (CSE-KMS).

Überlegungen und Einschränkungen

- Das Mindestverschlüsselungs-Feature ist für Apache-Spark-fähige Arbeitsgruppen nicht verfügbar.
- Das Mindestverschlüsselungs-Feature funktioniert nur, wenn die Arbeitsgruppe die Option Clientseitige Einstellungen überschreiben nicht aktiviert.
- Wenn für die Arbeitsgruppe die Option Clientseitige Einstellungen überschreiben aktiviert ist, hat die Einstellung für die Arbeitsgruppenverschlüsselung Vorrang, und die Einstellung für die Mindestverschlüsselung hat keine Auswirkung.
- Die Aktivierung dieses Features ist kostenlos.

Aktivierung der Mindestverschlüsselung für eine Arbeitsgruppe

Sie können eine Mindestverschlüsselungsstufe für die Abfrageergebnisse Ihrer Athena-SQL-Arbeitsgruppe aktivieren, wenn Sie die Arbeitsgruppe erstellen oder aktualisieren. Dazu können Sie die Athena-Konsole, die Athena-API oder verwenden AWS CLI.

Verwendung der Athena-Konsole zur Aktivierung der Mindestverschlüsselung

Informationen zum Erstellen oder Bearbeiten Ihrer Arbeitsgruppe mit der Athena-Konsole finden Sie unter Eine Arbeitsgruppe erstellen oder Eine Arbeitsgruppe bearbeiten. Gehen Sie bei der Konfiguration Ihrer Arbeitsgruppe wie folgt vor, um die Mindestverschlüsselung zu aktivieren.

So konfigurieren Sie die Mindestverschlüsselungsstufe für Arbeitsgruppenabfrageergebnisse

1. Erweitern Sie im Abschnitt Zusätzliche Konfigurationen die Option Einstellungen.
2. Deaktivieren Sie die Option Clientseitige Einstellungen überschreiben, oder stellen Sie sicher, dass sie nicht ausgewählt ist.
3. Erweitern Sie im Abschnitt Zusätzliche Konfigurationen die Option Konfiguration der Abfrageergebnisse.
4. Wählen Sie die Option Abfrageergebnisse verschlüsseln aus.
5. Wählen Sie unter Verschlüsselungstyp die Verschlüsselungsmethode aus, die Athena für die Abfrageergebnisse Ihrer Arbeitsgruppe verwenden soll (SSE_S3, SSE_KMS oder CSE_KMS). Diese Verschlüsselungstypen entsprechen den Sicherheitsstufen „Basis“, „Mittelstufe“ und „Erweitert“.
6. Um die Verschlüsselungsmethode durchzusetzen, die Sie als Mindestverschlüsselungsstufe für alle Benutzer ausgewählt haben, wählen Sie ***encryption_method*** als Mindestverschlüsselung aus.

Wenn Sie diese Option auswählen, werden in einer Tabelle die Verschlüsselungshierarchie und die Verschlüsselungsstufen aufgeführt, die Benutzern gewährt werden, wenn der von Ihnen gewählte Verschlüsselungstyp zum Mindestverschlüsselungstyp wird.

7. Nachdem Sie Ihre Arbeitsgruppe erstellt oder Ihre Arbeitsgruppenkonfiguration aktualisiert haben, wählen Sie Arbeitsgruppe erstellen oder Änderungen speichern.

Verwenden der Athena-API oder AWS CLI zum Aktivieren der Mindestverschlüsselung

Wenn Sie die CreateWorkGroup oder UpdateWorkGroup API verwenden, um eine Athena-SQL-Arbeitsgruppe zu erstellen oder zu aktualisieren, setzen Sie EnforceWorkGroupConfiguration auf `false`, EnableMinimumEncryptionConfiguration auf und verwenden Sie die `true`, EncryptionOption um den Verschlüsselungstyp anzugeben.

AWS CLIVerwenden Sie in der den `update-work-group` Befehl `create-work-group` oder mit den `--configuration-updates` Parametern `--configuration` oder und geben Sie die Optionen an, die denen für die API entsprechen.

Verwendung von für IAM Identity Center aktivierten Athena-Arbeitsgruppen

Die Funktion zur Verbreitung vertrauenswürdiger Identitäten von AWS IAM Identity Center ermöglicht es Ihren Mitarbeitern, Identitäten über Analyseservices hinweg AWS zu verwenden. Die Weitergabe vertrauenswürdiger Identitäten erspart Ihnen die Durchführung Service-spezifischer Konfigurationen von Identitätsanbietern oder IAM-Rolleneinrichtungen.

Mit IAM Identity Center können Sie die Anmeldesicherheit für Ihre Workforce-Identitäten, auch Workforce-Benutzer genannt, verwalten. IAM Identity Center bietet einen Ort, an dem Sie Mitarbeiter erstellen oder verbinden und ihren Zugriff über alle AWS Konten und Anwendungen hinweg zentral verwalten können. Mithilfe von Berechtigungen für mehrere Konten können Sie diesen Benutzern Zugriff auf AWS-Kontenzuweisen. Sie können Anwendungszuweisungen verwenden, um Ihren Benutzern Zugriff auf für IAM Identity Center aktivierte Anwendungen, Cloud-Anwendungen und Security Assertion Markup Language (SAML 2.0)-Anwendungen des Kunden zuzuweisen. Weitere Informationen finden Sie unter Vertrauenswürdige Identitätsweitergabe über Anwendungen hinweg im AWS IAM Identity Center -Benutzerhandbuch.

Derzeit ermöglicht die Athena-SQL-Unterstützung für die Weitergabe vertrauenswürdiger Identitäten die Verwendung derselben Identität für Amazon EMR Studio und die Athena-SQL-Schnittstelle in EMR Studio. Um IAM-Identity-Center-Identitäten mit Athena SQL in EMR Studio zu verwenden,

müssen Sie für IAM Identity Center aktivierte Arbeitsgruppen in Athena erstellen. Anschließend können Sie die IAM-Identity-Center-Konsole oder die API verwenden, um IAM-Identity-Center-Benutzer oder -Gruppen den für IAM Identity Center aktivierten Athena-Arbeitsgruppen zuzuweisen. Abfragen von einer Athena-Arbeitsgruppe, die die Weitergabe vertrauenswürdiger Identitäten verwendet, müssen über die Athena-SQL-Schnittstelle in einem EMR Studio ausgeführt werden, in dem IAM Identity Center aktiviert ist.

Überlegungen und Einschränkungen

Berücksichtigen Sie bei Verwendung der Weitergabe vertrauenswürdiger Identitäten mit Amazon Athena verwenden, die folgenden Punkte:

- Sie können die Authentifizierungsmethode für die Arbeitsgruppe nach der Erstellung der Arbeitsgruppe nicht mehr ändern.
  - Vorhandene Athena-SQL-Arbeitsgruppen können nicht zur Unterstützung von für IAM Identity Center aktivierten Arbeitsgruppen geändert werden.
  - Arbeitsgruppen, die für IAM Identity Center aktiviert sind, können nicht geändert werden, um IAM-Berechtigungen auf Ressourcenebene oder identitätsbasierte IAM-Richtlinien zu unterstützen.
- Für den Zugriff auf Arbeitsgruppen mit aktivierter Identitätsverbreitung müssen IAM-Identity-Center-Benutzer dem zugewiesen werden`IdentityCenterApplicationArn`, der von der Antwort der AthenaGetWorkGroup-API-Aktion zurückgegeben wird.
- Amazon-S3-Zugriffsberechtigungen müssen für die Verwendung der Weitergabe vertrauenswürdiger Identitäten konfiguriert sein. Weitere Informationen finden Sie unter S3-Zugriffsberechtigungen und Identitäten im Unternehmensverzeichnis im Amazon-S3-Benutzerhandbuch.
- Athena-Arbeitsgruppen, die für IAM Identity Center aktiviert sind, erfordern die Konfiguration von Lake Formation für die Verwendung von IAM-Identity-Center-Identitäten. Informationen zur Konfiguration finden Sie unter Integration von IAM Identity Center im AWS Lake Formation - Entwicklerhandbuch.
- In Arbeitsgruppen, die die Weitergabe vertrauenswürdiger Identitäten verwenden, kommt es bei Abfragen standardmäßig nach 30 Minuten zu einer Zeitüberschreitung. Sie können eine Erhöhung des Abfrage-Timeouts beantragen. Die maximale Ausführung einer Abfrage in Arbeitsgruppen zur Weitergabe vertrauenswürdiger Identitäten beträgt jedoch eine Stunde.
- Es kann bis zu einer Stunde dauern, bis Änderungen der Benutzer- oder Gruppenberechtigungen in Arbeitsgruppen zur Weitergabe vertrauenswürdiger Identitäten wirksam werden.

- Abfragen in einer Athena-Arbeitsgruppe, die die Weitergabe vertrauenswürdiger Identitäten verwendet, können nicht direkt über die Athena-Konsole ausgeführt werden. Sie müssen über die Athena-Schnittstelle in einem EMR Studio ausgeführt werden, in dem IAM Identity Center aktiviert ist. Weitere Informationen zur Verwendung von Athena in EMR Studio finden Sie unter Verwenden des Amazon-Athena-SQL-Editors in EMR Studio im Amazon-EMR-Verwaltungshandbuch.
- Die Weitergabe von vertrauenswürdigen Identitäten ist nicht mit den folgenden Athena-Features kompatibel.
  - `aws:CalledVia`-Kontextschlüssel.
  - Athena für Spark-Arbeitsgruppen.
  - Verbundzugriff auf die Athena-API.
  - Verbundzugriff auf Athena mithilfe von Lake Formation und den Athena-JDBC- und ODBC-Treibern.
- Sie können die Verbreitung vertrauenswürdiger Identitäten mit Athena nur in den folgenden verwenden AWS-Regionen:
  - `us-east-2` – USA Ost (Ohio)
  - `us-east-1` – USA Ost (Nord-Virginia)
  - `us-west-1` – USA West (Nordkalifornien)
  - `us-west-2` – USA West (Oregon)
  - `af-south-1` – Afrika (Kapstadt)
  - `ap-east-1` – Asien-Pazifik (Hongkong)
  - `ap-southeast-3` – Asien-Pazifik (Jakarta)
  - `ap-south-1` – Asien-Pazifik (Mumbai)
  - `ap-northeast-3` – Asien-Pazifik (Osaka)
  - `ap-northeast-2` – Asien-Pazifik (Seoul)
  - `ap-southeast-1` – Asien-Pazifik (Singapur)
  - `ap-southeast-2` – Asien-Pazifik (Sydney)
  - `ap-northeast-1` – Asien-Pazifik (Tokio)
  - `ca-central-1` – Kanada (Zentral)
  - `eu-central-1` – Europa (Frankfurt)
  - `eu-west-1` – Europa (Irland)
  - `eu-west-2` – Europa (London)
  - `eu-south-1` – Europa (Mailand)

- `eu-west-3` – Europa (Paris)
- `eu-north-1` – Europa (Stockholm)
- `me-south-1` – Naher Osten (Bahrain)
- `sa-east-1` – Südamerika (São Paulo)

Erforderliche Berechtigungen

Dem IAM-Benutzer des Administrators, der die für IAM Identity Center aktivierte Arbeitsgruppe in der Athena-Konsole erstellt, müssen die folgenden Richtlinien angefügt sein.

- Die von `AmazonAthenaFullAccess` verwaltete Richtlinie. Details hierzu finden Sie unter AWS Verwaltete Richtlinie: AmazonAthenaFullAccess.
- Die folgende Inline-Richtlinie, die IAM- und IAM-Identity-Center-Aktionen zulässt:

```
{
    "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": [
                "*"
            ]
        }
    ]
}
```

Erstellen einer für IAM Identity Center aktivierten Arbeitsgruppe

Das folgende Verfahren zeigt die Schritte und Optionen zum Erstellen einer für IAM Identity Center aktivierten Athena-Arbeitsgruppe. Eine Beschreibung der anderen für Athena-Arbeitsgruppen verfügbaren Konfigurationsoptionen finden Sie unter Erstellen von Arbeitsgruppen.

So erstellen Sie eine SSO-fähige Arbeitsgruppe in der Athena-Konsole

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wählen Sie im Navigationsbereich der Athena-Konsole Workgroups (Arbeitsgruppen) aus.
3. Wählen Sie auf der Seite Workgroups (Arbeitsgruppen) die Option Create workgroup (Arbeitsgruppe erstellen) aus.
4. Geben Sie auf der Seite Arbeitsgruppe erstellen unter Arbeitsgruppenname einen Namen für die Arbeitsgruppe ein.
5. Verwenden Sie für die Analytics-Engine die Standardeinstellung von Athena SQL.
6. Wählen Sie für Authentifizierung die Option IAM Identity Center aus.

7. Wählen Sie für Servicerolle für Zugriff auf das IAM Identity Center eine vorhandene Servicerolle aus oder erstellen Sie eine neue.

   Athena benötigt Berechtigungen, um für Sie auf IAM Identity Center zugreifen zu können. Hierzu ist eine Servicerolle für Athena erforderlich. Eine Servicerolle ist eine von Ihnen verwaltete IAM-Rolle, die einen - AWS Service autorisiert, in Ihrem Namen auf andere - AWS Services zuzugreifen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen - AWS Service im IAM-Benutzerhandbuch.
8. Erweitern Sie die Konfiguration des Abfrageergebnisses und geben Sie dann einen Amazon-S3-Pfad für Speicherort des Abfrageergebnisses ein oder wählen Sie ihn aus.
9. (Optional) Wählen Sie Abfrageergebnisse verschlüsseln aus.
10. (Optional) Wählen Sie S3-Präfix basierend auf Benutzeridentität erstellen aus.

    Wenn Sie eine für IAM Identity Center aktivierte Arbeitsgruppe erstellen, ist standardmäßig die Option S3-Zugriffsgewährungen aktivieren ausgewählt. Sie können Amazon-S3-Zugriffsberechtigungen verwenden, um den Zugriff auf Speicherorte (Präfixe) der Athena-Abfrageergebnisse in Amazon S3 zu steuern. Weitere Informationen zu Amazon-S3-Zugriffsberechtigungen finden Sie unter Verwalten des Zugriffs mit Amazon-S3-Zugriffsberechtigungen.

    In Athena-Arbeitsgruppen, die die IAM-Identity-Center-Authentifizierung verwenden, können Sie die Erstellung von identitätsbasierten Speicherorten für Abfrageergebnisse aktivieren, die von Amazon-S3-Zugriffsberechtigungen verwaltet werden. Mit diesen auf der Benutzeridentität basierenden Amazon-S3-Präfixen können Benutzer in einer Athena-Arbeitsgruppe ihre Abfrageergebnisse von anderen Benutzern in derselben Arbeitsgruppe isoliert halten.

    Wenn Sie die Option Benutzerpräfix ermöglichen, fügt Athena die Benutzer-ID als Amazon-S3-Pfadpräfix an den Standort der Abfrageergebnisse für die Arbeitsgruppe an (z. B. `s3://DOC-EXAMPLE-BUCKET/${user_id}`). Um dieses Feature nutzen zu können, müssen Sie Zugriffsberechtigungen so konfigurieren, dass sie nur dem Benutzer Berechtigungen für den Standort gewähren, der das `user_id`-Präfix hat.

    > **Note**
    >
    > Durch die Auswahl der S3-Präfixoption für die Benutzeridentität wird automatisch die Option Clientseitige Einstellungen überschreiben für die Arbeitsgruppe aktiviert, wie im

nächsten Schritt beschrieben. Die Option Clientseitige Einstellungen außer Kraft setzen ist eine Voraussetzung für das Feature „Benutzeridentitätspräfix“.

11. Erweitern Sie Einstellungen und bestätigen Sie dann, dass Clientseitige Einstellungen überschreiben ausgewählt ist.

    Wenn Sie Clientseitige Einstellungen überschreiben auswählen, werden Arbeitsgruppeneinstellungen auf Arbeitsgruppenebene für alle Clients in der Arbeitsgruppe erzwungen. Weitere Informationen finden Sie unter Arbeitsgruppen-Einstellungen überschreiben clientseitige Einstellungen.
12. (Optional) Nehmen Sie alle erforderlichen Konfigurationseinstellungen vor, wie unter Erstellen von Arbeitsgruppen beschrieben.
13. Wählen Sie Create workgroup (Arbeitsgruppe erstellen) aus.
14. Verwenden Sie den Abschnitt Arbeitsgruppen der Athena-Konsole, um Benutzer oder Gruppen aus Ihrem IAM-Identity-Center-Verzeichnis Ihrer für IAM Identity Center aktivierten Athena-Arbeitsgruppe zuzuweisen.

Athena-Arbeitsgruppen-APIs

Im Folgenden werden einige der REST-API-Vorgänge aufgelistet, die für Athena-Arbeitsgruppen verwendet werden. Für alle folgenden Vorgänge (mit Ausnahme von `ListWorkGroups`) müssen Sie eine Arbeitsgruppe angeben. In anderen Vorgänge, z. B. `StartQueryExecution`, ist der Arbeitsgruppenparameter optional und die Vorgänge werden hier nicht aufgelistet. Eine vollständige Liste der Vorgänge finden Sie in der Amazon-Athena-API-Referenz.

- CreateWorkGroup
- DeleteWorkGroup
- GetWorkGroup
- ListWorkGroups
- UpdateWorkGroup

Fehlerbehebung bei Arbeitsgruppen

Verwenden Sie die folgenden Tipps, um Fehler für Arbeitsgruppen zu beheben.

- Überprüfen Sie die Berechtigungen für einzelne Benutzer in Ihrem Konto. Sie müssen Zugriff auf den Speicherort für Abfrageergebnisse und auf die Arbeitsgruppe besitzen, in der sie Abfragen ausführen möchten. Wenn sie zwischen zwei Arbeitsgruppen wechseln möchten, müssen sie über Berechtigungen für beide Arbeitsgruppen verfügen. Weitere Informationen finden Sie unter IAM-Richtlinien für den Zugriff auf Arbeitsgruppen.
- Achten Sie auf den Kontext in der Athena-Konsole, um zu sehen, in welcher Arbeitsgruppe Sie Abfragen ausführen werden. Wenn Sie den Treiber verwenden, müssen Sie die Arbeitsgruppe auf die von Ihnen benötigte Arbeitsgruppe festlegen. Weitere Informationen finden Sie unter the section called “Angeben einer Arbeitsgruppe, in der Abfragen ausgeführt werden sollen”.
- Wenn Sie die API oder die Treiber zum Ausführen von Abfragen verwenden, müssen Sie den Speicherort der Abfrageergebnisse auf eine der folgenden Arten angeben: Verwenden Sie für einzelne Abfragen OutputLocation (clientseitig). Verwenden Sie in der Arbeitsgruppe WorkGroupConfiguration. Wenn der Speicherort nicht mit einem dieser Verfahren angegeben wird, gibt Athena bei der Abfrageausführung einen Fehler aus.
- Wenn Sie clientseitige Einstellungen mit Arbeitsgruppeneinstellungen überschreiben, treten möglicherweise Fehler für den Speicherort der Abfrageergebnisse auf. Beispielsweise könnte der Benutzer einer Arbeitsgruppe möglicherweise nicht über die nötigen Berechtigungen für den Speicherort der Arbeitsgruppe in Amazon S3 zum Speichern von Abfrageergebnissen verfügen. In diesem Fall müssen Sie die notwendigen Berechtigungen hinzufügen.
- Arbeitsgruppen führen Änderungen in Bezug auf das Verhalten der API-Vorgänge ein. Aufrufe der folgenden vorhandenen API-Vorgänge erfordern, dass Benutzer in Ihrem Konto in IAM über ressourcenbasierte Berechtigungen für die Arbeitsgruppen verfügen, in denen sie diese Aufrufe ausführen. Wenn keine Berechtigungen für die Arbeitsgruppe und für Arbeitsgruppenaktionen vorhanden sind, lösen die folgenden API-Aktionen `AccessDeniedException`CreateNamedQuery, DeleteNamedQuery, GetNamedQuery, ListNamedQueriesStopQueryExecution, , StartQueryExecutionListQueryExecutions, GetQueryExecution, GetQueryResults, und aus GetQueryResultsStream (diese API-Aktion ist nur für die Verwendung mit dem Treiber verfügbar und wird nicht anderweitig für die öffentliche Verwendung verfügbar gemacht). Weitere Informationen finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Athena in der Service-Autorisierungs-Referenz.

  Aufrufe der BatchGetNamedQuery API-Operationen BatchGetQueryExecution und geben nur Informationen zu Abfragen zurück, die in Arbeitsgruppen ausgeführt werden, auf die Benutzer Zugriff haben. Wenn ein Benutzer keinen Zugriff auf eine Arbeitsgruppe hat, geben diese API-Vorgänge die ID der nicht autorisierten Abfrage als Teil der Liste nicht verarbeiteter IDs zurück. Weitere Informationen finden Sie unter the section called “ Athena-Arbeitsgruppen-APIs”.

- Wenn die Arbeitsgruppe, in der eine Abfrage ausgeführt wird, mit einem erzwungenen Speicherort für Abfrageergebnisse konfiguriert wurde und Sie keinen `external_location` für die CTAS-Abfrage angeben. Athena gibt einen Fehler aus und schlägt in diesem Fall eine Abfrage fehl, die einen `external_location` angibt. Die folgende Abfrage schlägt beispielsweise fehl, wenn Sie die clientseitigen Einstellungen für den Abfrageergebnisspeicherort überschreiben und die Verwendung eines eigenen Speicherorts für die Arbeitsgruppe erzwingen: `CREATE TABLE <DB>.<TABLE1> WITH (format='Parquet', external_location='s3://my_test/test/') AS SELECT * FROM <DB>.<TABLE2> LIMIT 10;`

Ihnen werden möglicherweise die folgenden Fehler angezeigt. Diese Tabelle enthält eine Liste mit einigen der Fehler, die im Zusammenhang mit Arbeitsgruppen auftreten können, und entsprechende Lösungsvorschläge.

Fehler für Arbeitsgruppen

Fehler	Tritt auf, wenn ...
Abfragestatus STORNIERT. Das Limit für gescannte Bytes wurde überschritten.	Eine Abfrage das Limit für die Datennutzung pro Abfrage erreicht und abgebrochen wird. Sie sollten die Abfrage neu schreiben, damit sie weniger Daten liest, oder sich an den Kontoadministrator wenden.
Benutzer: *arn:aws:iam::123456789012:user/abc* ist nicht berechtigt, Folgendes auszuführen: athena:StartQueryExecution on-Ressource: *arn:aws:athena:us-east-1:123456789012:workgroup/workgroupname*	Ein Benutzer führt eine Abfrage in einer Arbeitsgruppe aus, auf die er nicht zugreifen kann. Aktualisieren Sie Ihre Richtlinie, um auf die Arbeitsgruppe zugreifen zu können.
INVALID_INPUT. WorkGroup <name> ist deaktiviert.	Ein Benutzer führt eine Abfrage in einer Arbeitsgruppe aus, aber die Arbeitsgruppe ist deaktiviert. Ihre Arbeitsgruppe könnte von Ihrem Administrator deaktiviert worden sein. Es ist auch möglich, dass Sie keinen Zugriff auf die Arbeitsgruppe haben. In beiden Fällen sollten

Fehler	Tritt auf, wenn ...
	Sie sich an einen Administrator wenden, der Arbeitsgruppen ändern kann.
INVALID_INPUT. WorkGroup <name> wurde nicht gefunden.	Ein Benutzer führt eine Abfrage in einer Arbeitsgruppe aus, aber die Arbeitsgruppe ist nicht vorhanden. Dies kann der Fall sein, wenn die Arbeitsgruppe gelöscht wurde. Wechseln Sie zu einer anderen Arbeitsgruppe, um Ihre Abfrage auszuführen.
InvalidRequestException: beim Aufrufen der StartQueryExecution -Operation: Kein Ausgabespeicherort angegeben. Ein Ausgabespeicherort ist entweder über die Konfigurationseinstellung für das Arbeitsgruppenergebnis oder als API-Eingabe erforderlich.	Ein Benutzer führt eine Abfrage mit der API aus, ohne den Speicherort für Abfrageergebnisse anzugeben. Sie müssen den Ausgabespeicherort für Abfrageergebnisse auf eine der beiden Arten festlegen: entweder für einzelne Abfragen, mit OutputLocation (clientseitig) oder in der Arbeitsgruppe mit WorkGroup Configuration.
Die Abfrage „Tabelle als Auswahl erstellen“ ist fehlgeschlagen, da sie mit der Eigenschaft 'external_location' an eine Athena-Arbeitsgruppe übermittelt wurde, die einen zentralen Ausgabeort für alle Abfragen erzwingt. Entfernen Sie die Eigenschaft 'external_location' und übermitteln Sie die Anfrage erneut.	Wenn die Arbeitsgruppe, in der eine Abfrage ausgeführt wird, mit einem erzwungenen Speicherort für Abfrageergebnisse konfiguriert wurde und Sie einen `external_location` für die CTAS-Abfrage angeben. Entfernen Sie in diesem Fall den `external_location` und führen Sie die Abfrage erneut aus.
Die vorbereitete Anweisung *`prepared_statement_name`* kann nicht erstellt werden. Die Anzahl der vorbereiteten Anweisungen in dieser Arbeitsgruppe überschreitet den Grenzwert von 1 000.	Die Arbeitsgruppe enthält mehr als das Limit von 1 000 vorbereiteten Anweisungen. Sie können mit DEALLOCATE PREPARE eine oder mehrere vorbereitete Anweisungen aus der Arbeitsgruppe entfernen, um dieses Problem zu umgehen. Alternativ, können Sie eine neue Arbeitsgruppe erstellen.

Steuern von Kosten und Überwachen von Abfragen mit CloudWatch Metriken und Ereignissen

Arbeitsgruppen ermöglichen Ihnen das Festlegen von Limits zur Kontrolle der Datennutzung pro Abfrage oder Arbeitsgruppe, das Einrichten von Alarmen bei Überschreitung dieser Limits und das Veröffentlichen von Abfragemetriken zu CloudWatch.

In jeder Arbeitsgruppe können Sie folgende Aktionen ausführen:

- Konfigurieren von Data usage controls (Datennutzungskontrollen) pro Abfrage und Arbeitsgruppe und Einrichten von Aktionen, die ausgeführt werden sollen, wenn Abfragen die Schwellenwerte verletzen.
- Zeigen Sie Abfragemetriken an, analysieren Sie sie und veröffentlichen Sie sie in CloudWatch. Wenn Sie eine Arbeitsgruppe in der Konsole erstellen, CloudWatch wird die Einstellung für die Veröffentlichung der Metriken in für Sie ausgewählt. Wenn Sie die API-Operationen verwenden, müssen Sie die Veröffentlichung der Metriken aktivieren. Wenn Metriken veröffentlicht werden, werden sie auf der Registerkarte Metrics (Metriken) im Bereich Workgroups (Arbeitsgruppen) angezeigt. Metriken sind standardmäßig für die primäre Arbeitsgruppe deaktiviert.

Video

Das folgende Video zeigt, wie Sie benutzerdefinierte Dashboards erstellen und Alarme und Auslöser für Metriken in festlegen CloudWatch. Sie können vorab ausgefüllte Dashboards direkt über die Athena-Konsole verwenden, um diese Abfragemetriken zu verwenden.

Überwachen von Amazon Athena-Abfragen mit Amazon CloudWatch

Themen

- Aktivieren von CloudWatch Abfragemetriken
- Überwachung von Athena-Abfragen mit CloudWatch Metriken
- Überwachung von Athena-Abfragen mit Amazon EventBridge-Ereignissen
- Überwachung der Athena-Nutzungsmetriken
- Festlegen von Limits zur Kontrolle der Datennutzung

Aktivieren von CloudWatch Abfragemetriken

Wenn Sie eine Arbeitsgruppe in der Konsole erstellen, CloudWatch ist die Einstellung für die Veröffentlichung von Abfragemetriken in standardmäßig ausgewählt.

So aktivieren oder deaktivieren Sie Abfragemetriken in der Athena-Konsole für eine Arbeitsgruppe

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.

3. Wählen Sie im Navigationsbereich die Option Arbeitsgruppen aus.
4. Wählen Sie den Link der Arbeitsgruppe aus, die Sie ändern möchten.
5. Wählen Sie auf der Detailseite für die Arbeitsgruppe Edit (Bearbeiten) aus.
6. Wählen oder deaktivieren Sie im Abschnitt Einstellungen die Option Abfragemetriken in veröffentlichen AWS CloudWatch.

Wenn Sie API-Operationen, die Befehlszeilenschnittstelle oder die Clientanwendung mit dem JDBC-Treiber verwenden, um Arbeitsgruppen zu erstellen, um die Veröffentlichung von Abfragemetriken zu ermöglichen, setzen Sie `PublishCloudWatchMetricsEnabled` `true` in auf WorkGroupConfiguration. Das folgende Beispiel zeigt ausschließlich die Konfiguration von Metriken und lässt andere Konfigurationen aus:

```
"WorkGroupConfiguration": {
      "PublishCloudWatchMetricsEnabled": "true"
```

```
    ....
    }
```

Überwachung von Athena-Abfragen mit CloudWatch Metriken

Athena veröffentlicht abfragebezogene Metriken in Amazon CloudWatch, wenn die Option Abfragemetriken in veröffentlichen CloudWatch ausgewählt ist. Sie können benutzerdefinierte Dashboards erstellen, Alarme und Auslöser für Metriken in festlegen CloudWatchoder vorab ausgefüllte Dashboards direkt von der Athena-Konsole aus verwenden.

Wenn Sie Abfragemetriken für Abfragen in Arbeitsgruppen aktivieren, werden die Metriken auf der Registerkarte Metrics (Metriken) im Bedienfeld Workgroups (Arbeitsgruppen) für jede Arbeitsgruppe in der Athena-Konsole angezeigt.

Athena veröffentlicht die folgenden Metriken in der CloudWatch Konsole:

- `DPUAllocated` – Die Gesamtzahl der DPUs (Datenverarbeitungseinheiten), die im Rahmen einer Kapazitätsreservierung für die Ausführung von Abfragen bereitgestellt wurden.
- `DPUConsumed` – Die Anzahl der DPUs, die zu einem bestimmten Zeitpunkt in einer Reservierung aktiv durch Abfragen in einem `RUNNING`-Zustand genutzt werden. Metrik, die nur ausgegeben wird, wenn die Arbeitsgruppe mit einer Kapazitätsreservierung verknüpft ist, und umfasst alle Arbeitsgruppen, die einer Reservierung zugeordnet sind.
- `DPUCount` – Die maximale Anzahl von DPUs, die von Ihrer Abfrage konsumiert werden. Sie wird genau einmal veröffentlicht, sobald die Abfrage abgeschlossen ist.
- `EngineExecutionTime` – Die Anzahl der Millisekunden, die die Abfrage zur Ausführung benötigt hat.
- `ProcessedBytes` – Die Anzahl der von Athena pro DML-Abfrage gescannten Byte.
- `QueryPlanningTime` – Die Anzahl der Millisekunden, die Athena zur Planung des Abfrageverarbeitungsflusses benötigt hat.
- `QueryQueueTime` – Die Anzahl der Millisekunden, während denen sich die Abfrage in der Abfragewarteschlange befunden und auf Ressourcen gewartet hat.
- `ServicePreProcessingTime` – Die Anzahl der Millisekunden, die Athena für die Vorverarbeitung der Abfrage benötigt hat, bevor sie an die Abfrage-Engine gesendet wurde.
- `ServiceProcessingTime` – Die Anzahl der Millisekunden, die Athena für die Verarbeitung der Abfrageergebnisse benötigt hat, nachdem die Abfrage-Engine die Abfrageausführung abgeschlossen hatte.

- `TotalExecutionTime` – Die Anzahl der Millisekunden, die Athena zum Ausführen einer DDL- oder DML-Abfrage benötigt hat.

Weitere Informationen finden Sie unter Liste der CloudWatch Metriken und Dimensionen für Athena weiter unten in diesem Dokument.

Diese Metriken haben die folgenden Dimensionen:

- `CapacityReservation` – Der Name der Kapazitätsreservierung, die zur Ausführung der Abfrage verwendet wurde, falls zutreffend.
- `QueryState` – `SUCCEEDED`, `FAILED` oder `CANCELED`
- `QueryType` – `DML`, `DDL` oder `UTILITY`
- `WorkGroup` – Name der Arbeitsgruppe

Athena veröffentlicht die folgende Metrik in der CloudWatch Konsole unter dem `AmazonAthenaForApacheSpark` Namespace :

- `DPUCount` – Anzahl der DPUs, die während der Sitzung zur Ausführung der Berechnungen verbraucht wurden.

Diese Metrik besitzt die folgenden Dimensionen:

- `SessionId` – Die ID der Sitzung, in der die Berechnungen übermittelt werden.
- `WorkGroup` – Name der Arbeitsgruppe.

Weitere Informationen finden Sie unter Liste der CloudWatch Metriken und Dimensionen für Athena an späterer Stelle in diesem Thema. Weitere Informationen zu Athena-Nutzungsmetriken finden Sie unter Überwachung der Athena-Nutzungsmetriken.

So zeigen Sie Abfragemetriken für eine Arbeitsgruppe in der Konsole an

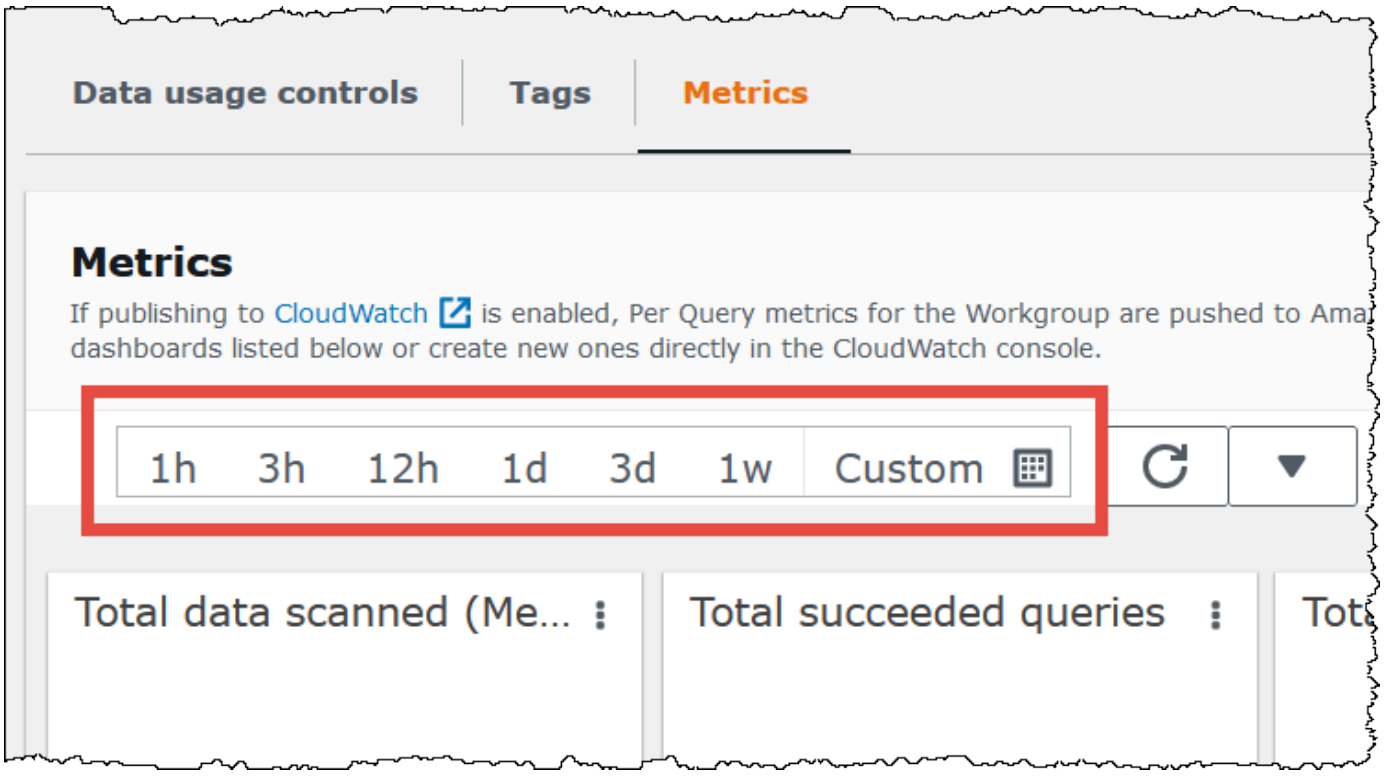
1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.

3. Wählen Sie im Navigationsbereich die Option Workgroups (Arbeitsgruppen) aus.
4. Wählen Sie in der Liste die Arbeitsgruppe aus, die Sie möchten, und wählen Sie dann das Metrics (Metriken)-Tab aus.

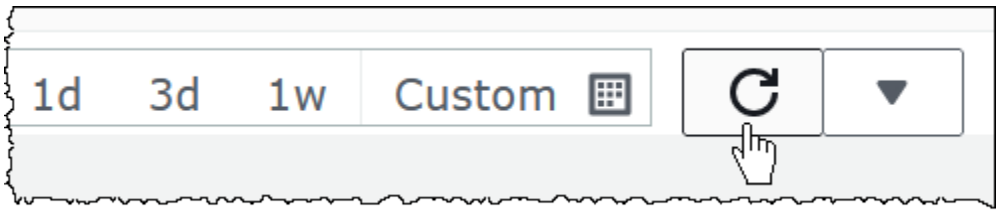
   Das Dashboard mit den Metriken wird angezeigt.

   > **Note**
   >
   > Wenn Sie erst kürzlich Metriken für die Arbeitsgruppe aktiviert haben und/oder keine aktuelle Abfrageaktivität stattgefunden hat, sind die Diagramme im Dashboard möglicherweise leer. Die Abfrageaktivität wird von abgerufen, CloudWatch abhängig vom Intervall, das Sie im nächsten Schritt angeben.

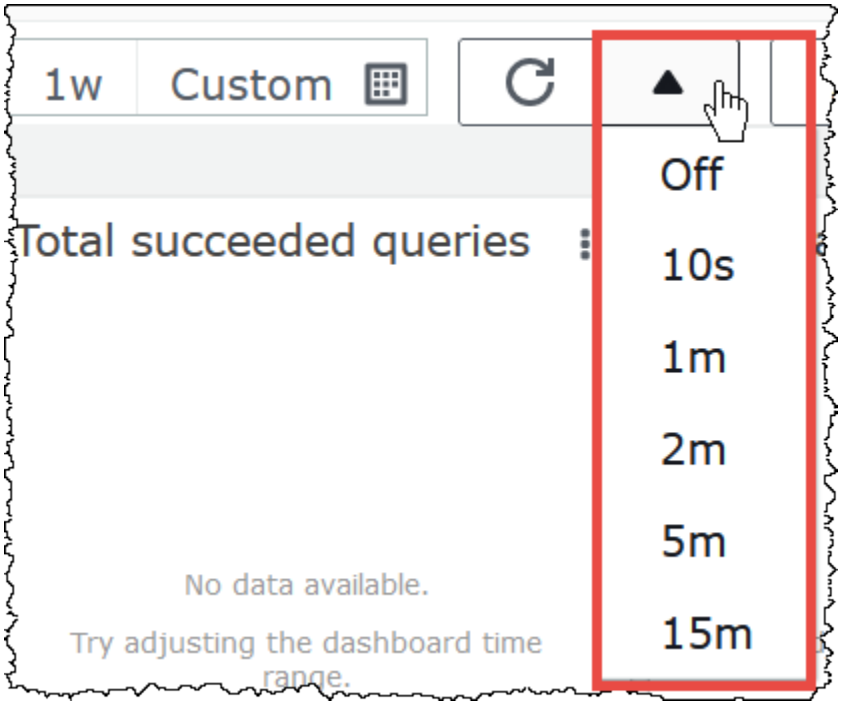
5. Wählen Sie im Abschnitt Metriken das Metrikintervall aus, das Athena zum Abrufen der Abfragemetriken von verwenden soll CloudWatch, oder geben Sie ein benutzerdefiniertes Intervall an.



6. Um die angezeigten Metriken zu aktualisieren, wählen Sie das Aktualisierungssymbol aus.



7. Wählen Sie den Pfeil neben dem Aktualisierungssymbol, um auszuwählen, wie oft die Anzeige der Metriken aktualisiert werden soll.



So zeigen Sie Metriken in der Amazon- CloudWatch Konsole an

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im Navigationsbereich Metrics (Metriken) All metrics (Alle Metriken) aus.
3. Wählen Sie das AWS/Athena-Namespace.

So zeigen Sie Metriken im CLI-Dashboard an

- Führen Sie eine der folgenden Aktionen aus:
  - Um die Metriken für Athena aufzulisten, öffnen Sie eine Eingabeaufforderung und verwenden Sie den folgenden Befehl:

```
aws cloudwatch list-metrics --namespace "AWS/Athena"
```

  - Verwenden Sie den folgenden Befehl, um alle verfügbaren Metriken aufzulisten:

```
aws cloudwatch list-metrics"
```

Liste der CloudWatch Metriken und Dimensionen für Athena

Wenn Sie CloudWatch Metriken in Athena aktiviert haben, werden die folgenden Metriken CloudWatch pro Arbeitsgruppe an gesendet. Die folgenden Metriken benutzen den `AWS/Athena`-Namespace.

Metrikname	Beschreibung
DPUAllocated	Die Gesamtzahl der DPUs (Datenverarbeitungseinheiten), die im Rahmen einer Kapazitätsreservierung für die Ausführung von Abfragen bereitgestellt wurden.
DPUConsumed	Die Anzahl der DPUs, die zu einem bestimmten Zeitpunkt in einer Reservierung aktiv durch Abfragen in einem `RUNNING`-Zustand genutzt werden. Diese Metrik wird nur ausgegeben, wenn die Arbeitsgruppe mit einer Kapazitätsreservierung verknüpft ist, und umfasst alle Arbeitsgruppen, die einer Reservierung zugeordnet sind. Wenn Sie eine Arbeitsgruppe von einer Reservierung in eine andere verschieben, enthält die Metrik Daten aus dem Zeitpunkt, zu dem die Arbeitsgruppe zur ersten Reservierung gehörte. Weitere Informationen über Kapazitätsreservierungen finden Sie unter Kapazität zur Abfrageverarbeitung verwalten.
DPUCount	Die maximale Anzahl von DPUs, die von Ihrer Abfrage konsumiert werden. Sie wird genau einmal veröffentlicht, sobald die Abfrage abgeschlossen ist. Diese Metrik wird nur für Arbeitsgruppen ausgegeben, die einer Kapazitätsreservierung zugeordnet sind.
EngineExecutionTime	Die Anzahl der Millisekunden, die die Abfrage zur Ausführung benötigt hat.
ProcessedBytes	Die Anzahl der von Athena pro DML-Abfrage gescannten Byte. Bei Abfragen, die storniert wurden (entweder durch die Benutzer oder automatisch, wenn sie das Limit erreicht hatten), schließt dies die Menge der vor der Stornierung durchsuchten Daten ein. Diese Metrik wird für DDL-Abfragen nicht gemeldet.

Metrikname	Beschreibung
QueryPlanningTime	Die Anzahl der Millisekunden, die Athena zur Planung des Abfrageverarbeitungsflusses benötigt hat. Dies enthält die Zeit, die zum Abrufen von Tabellenpartitionen aus der Datenquelle benötigt wurde. Beachten Sie, dass die Abfrageplanungszeit eine Teilmenge von ist, da die Abfrage-Engine die Abfrageplanung durchführt EngineExecutionTime.
QueryQueueTime	Die Anzahl der Millisekunden, während denen sich die Abfrage in der Abfragewarteschlange befunden und auf Ressourcen gewartet hat. Wenn vorübergehende Fehler auftreten, kann die Abfrage der Warteschlange automatisch wieder hinzugefügt werden.
ServicePreProcessingTime	Die Anzahl der Millisekunden, die Athena für die Vorverarb eitung der Abfrage benötigt hat, bevor sie an die Abfrage-Engine gesendet wurde.
ServiceProcessingTime	Die Anzahl der Millisekunden, die Athena für die Verarbeitung der Abfrageergebnisse benötigt hat, nachdem die Abfrage-E ngine die Abfrageausführung abgeschlossen hatte.
TotalExecutionTime	Die Anzahl der Millisekunden, die Athena zum Ausführen der Abfrage benötigt hat. TotalExecutionTime umfasst QueryQueu eTime, QueryPlanningTime EngineExecutionTime, und ServiceProcessingTime.

Diese Metriken für Athena haben die folgenden Dimensionen.

Dimension	Beschreibung
CapacityReservation	Der Name der Kapazitätsreservierung, die zur Ausführung der Abfrage verwendet wurde, falls zutreffend. Wenn eine Kapazität sreservierung nicht verwendet wird, gibt diese Dimension keine Daten zurück.

Dimension	Beschreibung
QueryState	Der Abfragestatus. Gültige Statistiken: ERFOLGT, GESCHEITERT oder ABGEBROCHEN.
QueryType	Der Abfragetyp. Gültige Statistiken: `DDL`, `DML` oder `UTILITY`. Der Typ der Abfrageanweisung, die ausgeführt wurde. `DDL` gibt DDL-Abfrageanweisungen (Data Definition Language) an. `DML` weist auf DML-Abfrageanweisungen (Data Manipulation Language) hin, wie z. B. `CREATE TABLE AS SELECT`. `UTILITY`kennzeichnet andere Abfrageanweisungen als DDL und DML, z. B. `SHOW CREATE TABLE` oder `DESCRIBE TABLE`.
WorkGroup	Der Name der Arbeitsgruppe.

Überwachung von Athena-Abfragen mit Amazon EventBridge-Ereignissen

Sie können Amazon Athena mit Amazon verwenden EventBridge , um Echtzeitbenachrichtigungen über den Status Ihrer Abfragen zu erhalten. Wenn eine von Ihnen übermittelte Abfrage den Status wechselt, veröffentlicht Athena ein Ereignis in , das Informationen über diesen Abfragestatusübergang EventBridge enthält. Sie können einfache Regeln für Ereignisse schreiben, die für Sie von Interesse sind, und automatisierte Aktionen ausführen, wenn ein Ereignis mit einer Regel übereinstimmt. Sie können beispielsweise eine Regel erstellen, die eine - AWS Lambda Funktion aufruft, wenn eine Abfrage einen Beendigungsstatus erreicht. Ereignisse werden auf bestmögliche Weise ausgegeben.

Führen Sie vor dem Erstellen von Ereignisregeln für Athena die folgenden Schritte aus:

- Machen Sie sich mit Ereignissen, Regeln und Zielen in vertraut EventBridge. Weitere Informationen finden Sie unter Was ist Amazon EventBridge? Weitere Informationen zum Einrichten von Regeln finden Sie unter Erste Schritte mit Amazon EventBridge.
- Erstellen Sie die Ziele für die Ereignisregeln.

> **Note**
>
> Athena bietet aktuell einen Ereignistyp, nämlich die Athena-Abfragezustandsänderung, wird aber möglicherweise um weitere Ereignistypen und Details erweitert. Wenn Sie JSON-Ereignisdaten programmgesteuert de-serialisieren, vergewissern Sie sich, dass Ihre Anwendung unbekannte Eigenschaften verarbeiten kann, falls diese zusätzlichen Eigenschaften hinzugefügt werden.

Ereignisformat von Athena

Im Folgenden finden Sie das grundlegende Muster für ein Amazon-Athena-Ereignis.

```
{
    "source":[
        "aws.athena"
    ],
    "detail-type":[
        "Athena Query State Change"
    ],
    "detail":{
        "currentState":[
            "SUCCEEDED"
        ]
    }
}
```

Athena-Abfragestatus-Änderungsereignis

Das folgende Beispiel zeigt ein Athena-Abfragezustand-Änderungsereignis-Ereignis mit dem `currentState`-Wert von `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"
    }
}
```

Das folgende Beispiel zeigt ein Athena-Abfragezustand-Änderungsereignis-Ereignis mit dem `currentState`-Wert von `FAILED`. Der `athenaError`-Block wird nur angezeigt, wenn `currentState` `FAILED` ist. Informationen zu den Werten für `errorCategory` und `errorType` finden Sie unter Athena-Fehlerkatalog.

```
{
    "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"
    }
```

```
}
```

Ausgabe-Eigenschaften

Die JSON-Ausgabe enthält die folgenden Eigenschaften.

Property (Eigenschaft)	Description (Beschreibung)
`athenaError`	Wird nur angezeigt, wenn `currentState` `FAILED` ist. Enthält Informationen über den aufgetretenen Fehler, einschließlich Fehlerkategorie, Fehlertyp, Fehlermeldung und ob die Aktion, die zum Fehler geführt hat, wiederholt werden kann. Die Werte für jedes dieser Felder hängen von der Art des aufgetretenen Fehlers ab. Informationen zu den Werten für `errorCategory` und `errorType` finden Sie unter Athena-Fehlerkatalog.
`versionId`	Die Versionsnummer für das Schema des Detailobjekts.
`currentState`	Der Zustand, in den die Abfrage zum Zeitpunkt des Ereignisses übergegangen ist.
`previousState`	Der Zustand, aus dem die Abfrage zum Zeitpunkt des Ereignisses übergegangen ist.
`statementType`	Der Typ der Abfrageanweisung, die ausgeführt wurde.
`queryExecutionId`	Der eindeutige Bezeichner für die ausgeführte Abfrage.
`workgroupName`	Der Name der Arbeitsgruppe, in der die Abfrage ausgeführt wurde.
`sequenceNumber`	Eine monoton zunehmende Zahl, die die Deduplizierung und Anordnung eingehender Ereignisse ermöglicht, die eine einzelne Abfrageausführung erfordern. Wenn doppelte Ereignisse für denselben Statusübergang veröffentlicht werden, ist der Wert `sequenceNumber` derselbe. Wenn bei einer Abfrage mehr als einmal ein Zustandsübergang auftritt, z. B. bei Abfragen mit dem selten vorkommenden erneuten Einreihen in die Warteschlange, können mit `sequenceNumber` Ereignisse nach

Property (Eigenschaft)	Description (Beschreibung)
	identischen `currentState` - und `previousState` -Werten angeordnet werden.

Beispiel

Im folgenden Beispiel werden Ereignisse in einem Amazon-SNS-Thema veröffentlicht, das Sie abonniert haben. Wenn Athena abgefragt wird, erhalten Sie eine E-Mail. Im Beispiel wird davon ausgegangen, dass das Thema Amazon SNS vorhanden ist und Sie es abonniert haben.

So veröffentlichen Sie Athena-Ereignisse in einem Amazon-SNS-Thema

1. Erstellen Sie das Ziel für Ihr Amazon-SNS-Thema. Erteilen Sie den EventBridge Ereignissen die `events.amazonaws.com` Berechtigung, in Ihrem Amazon SNS-Thema zu veröffentlichen, wie im folgenden Beispiel.

   ```
   {
       "Effect":"Allow",
       "Principal":{
           "Service":"events.amazonaws.com"
       },
       "Action":"sns:Publish",
       "Resource":"arn:aws:sns:us-east-1:111111111111:your-sns-topic"
   }
   ```

2. Verwenden Sie den AWS CLI `events put-rule` Befehl , um eine Regel für Athena-Ereignisse zu erstellen, wie im folgenden Beispiel.

   ```
   aws events put-rule --name {ruleName} --event-pattern '{"source": ["aws.athena"]}'
   ```

3. Verwenden Sie den AWS CLI `events put-targets` Befehl , um das Amazon SNS-Themenziel an die Regel anzuhängen, wie im folgenden Beispiel.

   ```
   aws events put-targets --rule {ruleName} --targets Id=1,Arn=arn:aws:sns:us-
   east-1:111111111111:your-sns-topic
   ```

4. Fragen Sie Athena ab und achten Sie auf das aufgerufene Ziel ab. Sie sollten entsprechende E-Mails von dem Amazon-SNS-Thema erhalten.

Verwenden von AWS-Benutzerbenachrichtigungen mit Amazon Athena

Sie können mit AWS-Benutzerbenachrichtigungen Lieferkanäle festlegen, um über Amazon-Athena-Ereignisse benachrichtigt zu werden. Sie erhalten eine Benachrichtigung, wenn ein Ereignis einer von Ihnen angegebenen Regel entspricht. Sie können Benachrichtigungen für Ereignisse über mehrere Kanäle erhalten, einschließlich E-Mail-, AWS Chatbot-Chat- oder AWS Console Mobile Application-Push-Benachrichtigungen. Sie können Benachrichtigungen auch im Console Notifications Center anzeigen. Benutzerbenachrichtigungen unterstützt die Aggregation, wodurch die Anzahl der Benachrichtigungen, die Sie bei bestimmten Ereignissen erhalten, reduziert werden kann.

Weitere Informationen finden Sie im AWS-Benutzerbenachrichtigungen -Benutzerhandbuch.

Überwachung der Athena-Nutzungsmetriken

Sie können CloudWatch Nutzungsmetriken verwenden, um einen Einblick in die Ressourcennutzung Ihres Kontos zu erhalten, indem Sie Ihre aktuelle Servicenutzung in CloudWatch Diagrammen und Dashboards anzeigen.

Für Athena entsprechen Metriken zur Nutzungsverfügbarkeit den AWS-Service Kontingenten für Athena. Sie können Alarme konfigurieren, mit denen Sie benachrichtigt werden, wenn sich Ihre Nutzung einem Servicekontingent nähert. Weitere Hinweise zu Athena-Servicekontingenten finden Sie unter Service Quotas. Weitere Informationen zu - AWS Nutzungsmetriken finden Sie unter -AWS Nutzungsmetriken im Amazon- CloudWatch Benutzerhandbuch.

Athena veröffentlicht die folgenden Metriken im `AWS/Usage`-Namespace.

Metrikname	Beschreibung
`ResourceCount`	Die Summe aller in die Warteschlange gestellten und ausführen den Abfragen für ein Konto, getrennt nach Abfragetyp (DML oder DDL). Maximum ist die einzige nützliche Statistik für diese Metrik. Diese Metrik wird regelmäßig jede Minute veröffentlicht. Wenn Sie keine Abfragen ausführen, meldet die Metrik nichts (nicht einmal 0). Die Metrik wird nur veröffentlicht, wenn zum Zeitpunkt der Metrik aktive Abfragen ausgeführt werden.

Die folgenden Dimensionen werden verwendet, um die Nutzungsmetriken zu verfeinern, die von Athena veröffentlicht werden.

Dimension	Beschreibung
`Service`	Der Name der , die die Ressource AWS-Service enthält. Der Wert für Athena für diese Dimension ist `Athena`.
`Resource`	Der Typ der Ressource, die ausgeführt wird. Der Ressource nwert für die Verwendung von Athena-Abfragen ist `ActiveQueryCount` .
`Type`	Der Typ von Entität, die gemeldet wird. Derzeit ist der einzige gültige Wert für Athena-Nutzungsmetriken `Resource`.
`Class`	Die Klasse der nachverfolgten Ressource. Für Athena kann `Class` ein `DML` oder ein `DDL` sein.

Anzeigen von Athena-Ressourcennutzungsmetriken in der CloudWatch Konsole

Sie können die CloudWatch Konsole verwenden, um ein Diagramm der Athena-Nutzungsmetriken anzuzeigen und Alarme zu konfigurieren, die Sie warnen, wenn sich Ihre Nutzung einem Servicekontingent nähert.

So zeigen Sie Metriken für die Nutzung von Athena an

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im Navigationsbereich Metrics (Metriken) All metrics (Alle Metriken) aus.
3. Wählen Sie Usage (Verwendung) und danach By AWS Resource (Nach Ressource).

   Die Liste der Service Quotas-Nutzungsmetriken wird angezeigt.
4. Aktivieren Sie das Kontrollkästchen neben Athena und ActiveQueryCount.
5. Wählen Sie die Registerkarte Graphed metrics (Grafisch dargestellte Metriken) aus.

   Das obige Diagramm zeigt Ihre aktuelle Nutzung der AWS Ressource an.

Informationen zum Hinzufügen von Service Quotas zum Diagramm und zum Einstellen eines Alarms, der Sie benachrichtigt, wenn Sie sich dem Service Quota nähern, finden Sie unter Visualisieren Ihrer Service Quotas und Einstellen von Alarmen im Amazon- CloudWatch Benutzerhandbuch. Informationen zum Festlegen von Nutzungslimits pro Arbeitsgruppe finden Sie unter Festlegen von Limits zur Kontrolle der Datennutzung.

Festlegen von Limits zur Kontrolle der Datennutzung

Athena ermöglicht Ihnen das Festlegen von zwei Arten von Kostenkontrollen: Limit pro Abfrage und Limit pro Arbeitsgruppe. Für jede Arbeitsgruppe können Sie nur ein Limit pro Abfrage festlegen. Sie können jedoch mehrere Limits pro Arbeitsgruppe festlegen.

- Das Limit zur Kontrolle pro Abfrage gibt die Gesamtmenge der pro Abfrage gescannten Daten an. Wenn eine in der Arbeitsgruppe ausgeführte Abfrage das Limit überschreitet, wird sie abgebrochen. Sie können in einer Arbeitsgruppe nur ein Limit zur Kontrolle pro Abfrage erstellen. Dieses Limit gilt für jede in der Arbeitsgruppe ausgeführte Abfrage. Bearbeiten Sie das Limit, wenn Sie es ändern müssen. Detaillierte Anweisungen finden Sie unter So erstellen Sie eine Kontrolle der Datennutzung pro Abfrage.
- Das arbeitsgruppenweite Limit zur Kontrolle der Datennutzung gibt die Gesamtmenge der gescannten Daten für alle Abfragen an, die in dieser Arbeitsgruppe während des angegebenen Zeitraums ausgeführt werden. Sie können mehrere Limits pro Arbeitsgruppe erstellen. Das arbeitsgruppenweite Limit für Abfragen ermöglicht Ihnen das Festlegen verschiedener Schwellenwerte für stündliche oder tägliche Datenaggregate, die von in der Arbeitsgruppe ausgeführten Abfragen gescannt werden.

  Wenn die Gesamtmenge der gescannten Daten den Schwellenwert überschreitet, können Sie eine Benachrichtigung an ein Amazon-SNS-Thema senden. Dazu konfigurieren Sie einen Amazon-SNS-Alarm und eine Aktion in der Athena-Konsole, um einen Administrator zu benachrichtigen, wenn das Limit überschritten wird. Detaillierte Anweisungen finden Sie unter So erstellen Sie eine Kontrolle der Datennutzung pro Arbeitsgruppe. Sie können auch einen Alarm und eine Aktion für jede Metrik erstellen, die Athena über die CloudWatch Konsole veröffentlicht. Beispielsweise können Sie eine Warnung für eine Reihe von fehlgeschlagenen Abfragen festlegen. Diese Warnung kann eine E-Mail an einen Administrator auslösen, wenn die Zahl einen bestimmten Schwellenwert überschreitet. Wenn das Limit überschritten wird, sendet eine Aktion eine Amazon-SNS-Alarmbenachrichtigung an die angegebenen Benutzer.

  Weitere Aktionen, die Sie ergreifen können:

- Rufen Sie eine Lambda-Funktion auf. Weitere Informationen finden Sie unter Aufrufen von Lambda-Funktionen bei Amazon-SNS-Benachrichtigungen im Enwicklerhandbuch von Amazon Simple Notification Service.
- Durch Deaktivieren der Arbeitsgruppe wird die Ausführung aller weiteren Abfragen gestoppt. Informationen zu den erforderlichen Schritten finden Sie unter Aktivieren und Deaktivieren von Arbeitsgruppen.

Die Limits pro Abfrage und pro Arbeitsgruppe sind voneinander unabhängig. Die angegebene Aktion wird ausgeführt, wenn eines der beiden Limits überschritten wird. Wenn zwei oder mehr Benutzer zur selben Zeit in derselben Arbeitsgruppe Abfragen ausführen, ist es möglich, dass die einzelnen Abfragen zwar keines der angegebenen Limits überschreiten, die Gesamtmenge der gescannter Daten jedoch das Limit für die Datennutzung pro Arbeitsgruppe überschreitet. In diesem Fall wird ein Amazon-SNS-Alarm an die Benutzer gesendet.

So erstellen Sie eine Kontrolle der Datennutzung pro Abfrage

Das Limit zur Kontrolle pro Abfrage gibt die Gesamtmenge der pro Abfrage gescannten Daten an. Wenn eine in der Arbeitsgruppe ausgeführte Abfrage das Limit überschreitet, wird sie abgebrochen. Abgebrochene Abfragen werden entsprechend den Amazon-Athena-Preisen berechnet.

> **Note**
>
> Im Fall von abgebrochenen oder fehlgeschlagenen Abfragen hat Athena möglicherweise bereits partielle Ergebnisse zu Amazon S3 geschrieben. In diesen Fällen löscht Athena die partiellen Ergebnisse nicht aus dem Amazon-S3-Präfix, in dem Ergebnisse gespeichert werden. Sie müssen das Amazon-S3-Präfix mit Teilergebnissen entfernen. Athena verwendet mehrteilige Amazon-S3-Uploads, um Daten in Amazon S3 zu schreiben. Wir empfehlen, die Bucket-Lebenszyklusrichtlinie so festzulegen, dass beim Fehlschlagen von Abfragen mehrteilige Uploads abgebrochen werden. Weitere Informationen finden Sie unter Unvollständige mehrteilige Uploads mit einer Bucket-Lebenszyklusrichtlinie abbrechen und im Benutzerhandbuch zu Amazon Simple Storage Service.

Sie können in einer Arbeitsgruppe nur ein Limit zur Kontrolle pro Abfrage erstellen. Dieses Limit gilt für jede in der Arbeitsgruppe ausgeführte Abfrage. Bearbeiten Sie das Limit, wenn Sie es ändern müssen.

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.

3. Wählen Sie im Navigationsbereich die Option Arbeitsgruppen aus.
4. Wählen Sie den Namen der Arbeitsgruppe aus der Liste aus.
5. Wählen Sie im Tab Data usage controls (Datennutzungssteuerungen) im Abschnitt Per query data usage control (Steuerung der Datennutzung pro Abfrage) Manage (Verwalten) aus.
6. Geben Sie auf der Seite Manage per query data usage control (Verwalten der Datenverwendungssteuerung pro Abfrage) die folgenden Werte an:

   - Geben Sie in Data limit (Datenlimit) einen Wert zwischen 10 MB (Minimum) und 7 EB (Maximum) an.

     > **Note**
     >
     > Dies sind von der Konsole für Datennutzungskontrollen innerhalb von Arbeitsgruppen vorgegebene Limits. Es handelt sich nicht um Abfragenlimits in Athena.

   - Wählen Sie für Einheiten den Einheitenwert aus der Dropdown-Liste aus (z. B. Kilobyte KB oder Exabyte EB).

   Der Standardwert für Action (Aktion) ist der Abbruch der Abfrage bei Überschreiten des Limits. Diese Einstellung kann nicht geändert werden.

7. Wählen Sie Speichern.

So erstellen oder bearbeiten Sie eine Warnung der Datennutzung pro Arbeitsgruppe

Sie können mehrere Warnungsschwellenwerte festlegen, wenn Abfragen, die in einer Arbeitsgruppe ausgeführt werden, eine bestimmte Datenmenge innerhalb eines bestimmten Zeitraums scannen. Warnungen werden mithilfe von Amazon- CloudWatch Alarmen implementiert und gelten für alle Abfragen in der Arbeitsgruppe. Wenn ein Schwellenwert erreicht ist, können Sie Amazon SNS eine E-Mail an von Ihnen angegebene Benutzer senden lassen. Abfragen werden nicht automatisch abgebrochen, wenn ein Schwellenwert erreicht wird.

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.
3. Wählen Sie im Navigationsbereich die Option Workgroups (Arbeitsgruppen) aus.
4. Wählen Sie den Namen der Arbeitsgruppe aus der Liste aus.
5. Wählen Sie Edit (Bearbeiten) aus, um die Einstellungen der Arbeitsgruppe zu bearbeiten.
6. Scrollen Sie nach unten zu Workgroup data usage alerts – optional (Warnungen zur Datennutzung von Arbeitsgruppen – optional) und erweitern Sie die Ansicht.
7. Wählen Sie Add alert (Warnung hinzufügen) aus.
8. Geben Sie für Data usage threshold configuration (Konfiguration der Datennutzungsschwelle) die Werte wie folgt an:

   - Geben Sie für Data threshold (Datenschwelle) eine Zahl an und wählen Sie dann einen Einheitenwert aus der Dropdown-Liste aus.
   - Wählen Sie in Zeitraum einen Zeitraum aus der Dropdown-Liste aus.
   - Wählen Sie für SNS-Thema auswählen ein Amazon-SNS-Thema aus der Dropdown-Liste aus. Oder wählen Sie Create SNS topic (SNS-Thema erstellen), um direkt zur Amazon SNS console (Amazon-SNS-Konsole) zu gelangen, das Amazon-SNS-Thema zu erstellen und ein Abonnement dafür für einen der Benutzer in Ihrem Athena-Konto einzurichten. Weitere Informationen finden Sie unter Erste Schritte mit Amazon SNS im Benutzerhandbuch für Amazon Simple Notification Service.
9. Wählen Sie Add alert (Warnung hinzufügen) aus, wenn Sie eine neue Warnung erstellen, oder Save (Speichern), um eine vorhandene Warnung zu speichern.

Kapazität zur Abfrageverarbeitung verwalten

Sie können Kapazitätsreservierungen verwenden, um dedizierte Verarbeitungskapazität für die Abfragen zu erhalten, die Sie in Athena ausführen. Mit Kapazitätsreservierungen können Sie die Vorteile von Workload-Management-Funktionen nutzen, mit denen Sie Ihre wichtigsten interaktiven Workloads priorisieren, kontrollieren und skalieren können. Sie können beispielsweise jederzeit Kapazität hinzufügen, um die Anzahl der Abfragen zu erhöhen, die Sie gleichzeitig ausführen können, zu kontrollieren, welche Workloads die Kapazität nutzen können, und die Kapazität auf mehrere Workloads verteilen. Die Kapazität wird vollständig von Athena verwaltet und so lange für Sie bereitgestellt, wie Sie sie benötigen. Die Einrichtung ist einfach und es sind keine Änderungen an Ihren SQL-Anweisungen erforderlich.

Um die Verarbeitungskapazität für Ihre Abfragen zu erhalten, erstellen Sie eine Kapazitätsreservierung, geben die Anzahl der benötigten Datenverarbeitungseinheiten (DPUs) an und weisen der Reservierung eine oder mehrere Arbeitsgruppen zu.

Arbeitsgruppen spielen eine wichtige Rolle, wenn Sie Kapazitätsreservierungen verwenden. Arbeitsgruppen ermöglichen es Ihnen, Abfragen in logischen Gruppierungen zu organisieren. Mit Kapazitätsreservierungen weisen Sie Arbeitsgruppen selektiv Kapazität zu, sodass Sie kontrollieren können, wie sich die Abfragen für jede Arbeitsgruppe verhalten und wie sie abgerechnet werden. Weitere Informationen zu Arbeitsgruppen finden Sie unter Verwendung von Arbeitsgruppen zur Kontrolle des Abfragenzugriffs und der Kosten.

Durch das Zuweisen von Arbeitsgruppen zu Reservierungen können Sie den Anfragen, die Sie an die zugewiesenen Arbeitsgruppen senden, Priorität einräumen. Sie könnten beispielsweise einer Arbeitsgruppe, die für zeitkritische Finanzberichterstattungsabfragen verwendet wird, Kapazität zuweisen, um diese Abfragen von weniger kritischen Abfragen in einer anderen Arbeitsgruppe zu isolieren. Dies ermöglicht eine konsistente Abfrageausführung für kritische Workloads, während andere Workloads unabhängig voneinander ausgeführt werden können.

Sie können Kapazitätsreservierungen und Arbeitsgruppen zusammen verwenden, um unterschiedlichen Anforderungen gerecht zu werden. Es folgen einige Beispielszenarien:

- Isolierung – Um einen wichtigen Workload zu isolieren, weisen Sie einer Reservierung eine einzelne Arbeitsgruppe zu. Nur Anfragen der zugewiesenen Arbeitsgruppe nutzen die Verarbeitungskapazität der ausgewählten Reservierung.
- Teilen – Mehrere Workloads nutzen die Kapazität einer Reservierung. Wenn Sie beispielsweise vorhersehbare monatliche Kosten für eine bestimmte Gruppe von Workloads wünschen,

können Sie einer einzigen Reservierung mehrere Arbeitsgruppen zuweisen. Die zugewiesenen Arbeitsgruppen teilen sich die reservierte Kapazität.

- Gemischtes Modell – Sie können Kapazitätsreservierungen und die Abrechnung pro Abfrage gleichzeitig in demselben Konto verwenden. Um beispielsweise die zuverlässige Ausführung von Abfragen zu gewährleisten, die eine Produktionsanwendung unterstützen, weisen Sie einer Kapazitätsreservierung eine Arbeitsgruppe für diese Abfragen zu. Wenn Sie die Abfragen entwickeln, bevor Sie sie in die Produktionsarbeitsgruppe verschieben, verwenden Sie eine separate Arbeitsgruppe, die nicht mit einer Reservierung verbunden ist und daher eine Abrechnung pro Abfrage verwendet.

DPUs verstehen

Die Kapazität wird in Datenverarbeitungseinheiten (DPUs) gemessen. DPUs stellen die Rechen- und Speicherressourcen dar, die Athena verwendet, um in Ihrem Namen auf Daten zuzugreifen und diese zu verarbeiten. Ein DPU bietet 4 vCPUs und 16 GB Arbeitsspeicher. Die Anzahl der von Ihnen angegebenen DPUs hat Einfluss auf die Anzahl der Abfragen, die Sie gleichzeitig ausführen können. Beispielsweise kann eine Reservierung mit 256 DPUs etwa die doppelte Anzahl gleichzeitiger Abfragen unterstützen als eine Reservierung mit 128 DPUs.

Sie können bis zu 100 Kapazitätsreservierungen mit insgesamt bis zu 1 000 DPUs pro Konto und Region erstellen. Sie müssen mindestens 24 DPUs anfordern. Wenn Sie für Ihren Anwendungsfall mehr als 1 000 DPUs benötigen, wenden Sie sich bitte an athena-feedback@amazon.com.

Informationen zur Schätzung Ihres Kapazitätsbedarfs finden Sie unter Kapazitätsanforderungen bestimmen. Preisinformationen finden Sie unter Amazon-Athena-Preisinformationen.

Überlegungen und Einschränkungen

- Das Feature erfordert die Athena-Engine-Version 3.
- Eine einzelne Arbeitsgruppe kann jeweils höchstens einer Reservierung zugewiesen werden, und Sie können einer Reservierung maximal 20 Arbeitsgruppen hinzufügen.
- Sie können Spark-fähige Arbeitsgruppen nicht zu einer Kapazitätsreservierung hinzufügen.
- Um eine Arbeitsgruppe zu löschen, die einer Reservierung zugewiesen wurde, entfernen Sie zuerst die Arbeitsgruppe aus der Reservierung.
- Sie müssen mindestens 24 DPUs anfordern.
- Sie können bis zu 100 Kapazitätsreservierungen mit insgesamt bis zu 1 000 DPUs pro Konto und Region erstellen.

- Kapazitätsanfragen können nicht garantiert werden und können bis zu 30 Minuten dauern.
- Es gibt einen Mindestabrechnungszeitraum von 1 Stunde. Nach einer Stunde wird die Kapazität pro Minute abgerechnet. Preisinformationen finden Sie unter Amazon-Athena-Preisinformationen.
- Reservierte Kapazität ist nicht auf eine andere Kapazitätsreservierung, AWS-Konto oder AWS-Region übertragbar.
- DDL-Abfragen zu Kapazitätsreservierungen verbrauchen DPUs.
- Abfragen, die mit bereitgestellter Kapazität ausgeführt werden, werden nicht auf Ihre aktiven Abfragegrenzwerte für DDL und DML angerechnet.
- Wenn alle DPUs verwendet werden, werden übermittelte Abfragen in die Warteschlange gestellt. Solche Abfragen werden nicht abgewiesen und werden nicht für On-Demand-Kapazitäten genutzt.
- Die `DPUConsumed`-CloudWatch-Metrik gilt pro Arbeitsgruppe und nicht pro Reservierung. Wenn Sie also eine Arbeitsgruppe von einer Reservierung in eine andere verschieben, enthält die `DPUConsumed`-Metrik Daten aus dem Zeitpunkt, zu dem die Arbeitsgruppe zur ersten Reservierung gehörte. Weitere Informationen zum Verwenden von CloudWatch-Metriken in Athena finden Sie unter Überwachung von Athena-Abfragen mit CloudWatch Metriken
- Das Feature ist aktuell in den folgenden AWS-Regionen verfügbar:
  - US East (Ohio)
  - USA Ost (Nord-Virginia)
  - USA West (Oregon)
  - Asien-Pazifik (Singapur)
  - Asien-Pazifik (Sydney)
  - Asien-Pazifik (Tokio)
  - Europa (Irland)
  - Europa (Stockholm)

Themen

Kapazitätsanforderungen bestimmen

Bevor Sie eine Kapazitätsreservierung erstellen, können Sie die benötigte Kapazität schätzen, sodass Sie ihr die richtige Anzahl von DPUs zuweisen können. Und wenn eine Reservierung genutzt wurde, sollten Sie die Reservierung möglicherweise auf unzureichende oder überschüssige Kapazität überprüfen. In diesem Thema werden Techniken beschrieben, mit denen Sie diese Schätzungen vornehmen können, und es werden auch einige AWS-Tools zur Bewertung von Nutzung und Kosten beschrieben.

Themen

Schätzung der erforderlichen Kapazität

Bei der Schätzung des Kapazitätsbedarfs ist es sinnvoll, zwei Aspekte zu berücksichtigen: wie viel Kapazität eine bestimmte Abfrage möglicherweise benötigt, und wie viel Kapazität Sie im Allgemeinen benötigen könnten.

Schätzung des Kapazitätsbedarfs pro Abfrage

Um zu ermitteln, wie viele DPUs für eine Abfrage erforderlich sein könnten, können Sie die folgenden Richtlinien verwenden:

- DDL-Abfragen verbrauchen 4 DPUs.
- DML-Abfragen verbrauchen in der Regel zwischen 4 und 124 DPUs.

Athena bestimmt die Anzahl der DPUs, die für eine DML-Abfrage erforderlich sind, wenn die Anfrage gesendet wird. Die Anzahl variiert je nach Datengröße, Speicherformat, Abfragekonstruktion und anderen Faktoren. Im Allgemeinen versucht Athena, die niedrigste und effizienteste DPU-Nummer auszuwählen. Wenn Athena feststellt, dass mehr Rechenleistung erforderlich ist, damit die Abfrage erfolgreich abgeschlossen werden kann, wird die Anzahl der der Abfrage zugewiesenen DPUs erhöht.

Schätzung der für die Workload spezifischen Kapazitätsanforderungen

Beachten Sie die allgemeinen Richtlinien in der folgenden Tabelle, um zu ermitteln, wie viel Kapazität Sie möglicherweise benötigen, um mehrere Abfragen gleichzeitig auszuführen:

Gleichzeitige Abfragen	DPUs erforderlich
10	40 oder mehr
20	96 oder mehr
30 oder mehr	240 oder mehr

Beachten Sie, dass die tatsächliche Anzahl der benötigten DPUs von Ihren Zielen und Analysemustern abhängt. Wenn Sie beispielsweise möchten, dass Abfragen sofort und ohne Warteschlangen gestartet werden, ermitteln Sie den höchsten Bedarf an gleichzeitigen Abfragen und stellen Sie dann die Anzahl der DPUs entsprechend bereit.

Sie können weniger DPUs bereitstellen, als Sie zu Spitzenzeiten benötigen. Bei Spitzenauslastung kann es jedoch zu Warteschlangen kommen. Wenn es zu Warteschlangen kommt, hält Athena Ihre Abfragen in einer Warteschlange und führt sie aus, sobald Kapazität verfügbar ist.

Wenn Sie Abfragen innerhalb eines festen Budgets ausführen möchten, können Sie den AWS-Preisrechner verwenden, um die Anzahl der DPUs zu ermitteln, die Ihrem Budget entsprechen.

Denken Sie abschließend daran, dass die Datengröße, das Speicherformat und die Art und Weise, wie eine Abfrage geschrieben wird, die DPUs beeinflussen, die für eine Abfrage benötigt werden. Um die Abfrageleistung zu erhöhen, können Sie Ihre Daten komprimieren, partitionieren oder in spaltenförmige Formate konvertieren. Weitere Informationen finden Sie unter Leistungsoptimierung in Athena.

Anzeichen dafür, dass mehr Kapazität benötigt wird

Fehlermeldungen zu unzureichender Kapazität und Queuing bei Abfragen sind zwei Anzeichen dafür, dass die zugewiesene Kapazität unzureichend ist.

Wenn Ihre Abfragen mit einer Fehlermeldung über unzureichende Kapazität fehlschlagen, ist die DPU-Anzahl Ihrer Kapazitätsreservierung zu niedrig für Ihre Abfrage. Wenn Sie beispielsweise eine Reservierung mit 24 DPUs haben und eine Abfrage ausführen, für die mehr als 24 DPUs erforderlich

sind, schlägt die Abfrage fehl. Um diesen Abfragefehler zu überwachen, können Sie EventBridge-Ereignisse von Athena verwenden. Versuchen Sie, weitere DPUs hinzuzufügen und Ihre Abfrage erneut auszuführen.

Wenn sich viele Abfragen in der Warteschlange befinden, bedeutet dies, dass Ihre Kapazität durch andere Abfragen voll ausgelastet ist. Um die Warteschlangen zu reduzieren, gehen Sie wie folgt vor:

- Fügen Sie Ihrer Reservierung DPUs hinzu, um die Parallelität der Abfragen zu erhöhen.
- Entfernen Sie Arbeitsgruppen aus Ihrer Reservierung, um Kapazitäten für andere Abfragen freizugeben.

Um zu prüfen, ob die Abfrage-Warteschlangen übermäßig lang sind, verwenden Sie die CloudWatch-Metrik „Athena-Abfrage-Warteschlangenzeit“ für die Arbeitsgruppen in Ihrer Kapazitätsreservierung. Wenn der Wert über Ihrem bevorzugten Schwellenwert liegt, können Sie der Kapazitätsreservierung DPUs hinzufügen.

Auf ungenutzte Kapazität prüfen

Um nach ungenutzter Kapazität zu suchen, können Sie entweder die Anzahl der DPUs in der Reservierung verringern oder den Workload erhöhen und dann die Ergebnisse beobachten.

Auf untätige Kapazität prüfen

1. Führen Sie eine der folgenden Aktionen aus:
   - Reduzieren Sie die Anzahl der DPUs in Ihrer Reservierung (reduzieren Sie die verfügbaren Ressourcen)
   - Fügen Sie Ihrer Reservierung Arbeitsgruppen hinzu (erhöhen Sie den Workload)
2. Verwenden Sie CloudWatch, um die Zeit der Abfragewarteschlange zu messen.
3. Wenn die Warteschlangenzeit einen gewünschten Wert überschreitet, gehen Sie wie folgt vor:
   - Entfernen Sie Arbeitsgruppen
   - Fügen Sie Ihrer Kapazitätsreservierung DPUs hinzu
4. Überprüfen Sie nach jeder Änderung die Leistung und die Warteschlangenzeit für Abfragen.
5. Passen Sie den Workload und/oder die DPU-Anzahl weiter an, um das gewünschte Gleichgewicht zu erreichen.

Wenn Sie die Kapazität außerhalb eines bevorzugten Zeitraums nicht aufrechterhalten möchten, können Sie die Reservierung stornieren und später eine weitere Reservierung erstellen. Selbst wenn Sie kürzlich Kapazitäten für eine andere Reservierung storniert haben, können Anfragen nach neuen Kapazitäten nicht garantiert werden, und die Erstellung neuer Reservierungen dauert einige Zeit.

Tools zur Bewertung von Kapazitätsanforderungen und Kosten

Sie können die folgenden Services und Feature in AWS verwenden, um Ihre Nutzung und Kosten von Athena zu messen.

CloudWatch-Metriken

Sie können Athena so konfigurieren, dass abfragebezogene Metriken auf Arbeitsgruppenebene in Amazon CloudWatch veröffentlicht werden. Nachdem Sie Metriken für die Arbeitsgruppe aktiviert haben, werden die Metriken für die Abfragen der Arbeitsgruppe in der Athena-Konsole auf der Detailseite der Arbeitsgruppe angezeigt.

Informationen zu den auf CloudWatch veröffentlichten Athena-Metriken und ihren Dimensionen finden Sie unter Überwachung von Athena-Abfragen mit CloudWatch Metriken.

CloudWatch-Nutzungsmetriken

Sie können CloudWatch-Nutzungsmetriken verwenden, um einen Einblick in Ihr Konto zu gewähren, indem Sie Ihre aktuelle Servicenutzung in CloudWatch-Diagrammen und Dashboards anzeigen. Für Athena entsprechen Metriken zur Nutzungsverfügbarkeit AWS Servicekontingente für Athena. Sie können Alarme konfigurieren, mit denen Sie benachrichtigt werden, wenn sich Ihre Nutzung einem Servicekontingent nähert.

Weitere Informationen finden Sie unter Überwachung der Athena-Nutzungsmetriken.

Amazon-EventBridge-Ereignisse

Sie können Amazon Athena mit Amazon EventBridge verwenden, um Echtzeitbenachrichtigungen zum Status Ihrer Abfragen zu erhalten. Wenn eine von Ihnen übermittelte Abfrage in andere Zustände übergeht, veröffentlicht Athena ein Ereignis in EventBridge, das Informationen über diesen Abfragezustandübergang enthält. Sie können einfache Regeln für Ereignisse schreiben, die für Sie von Interesse sind, und automatisierte Aktionen ausführen, wenn ein Ereignis mit einer Regel übereinstimmt.

Weitere Informationen finden Sie in den folgenden Ressourcen.

- Überwachung von Athena-Abfragen mit Amazon EventBridge-Ereignissen

- Was ist Amazon EventBridge?
- Amazon-EventBridge-Ereignisse

Tags (Markierungen)

In Athena unterstützen Kapazitätsreservierungen Tags. Ein Tag besteht aus einem Schlüssel und einem Wert. Um Ihre Kosten in Athena zu verfolgen, können Sie AWS-generierte Kostenzuordnungs-Tags verwenden. AWS verwendet Kostenzuordnungs-Tags, um Ihre Ressourcenkosten in Ihrem Kosten- und Nutzungsbericht zu gruppieren. Dies erleichtert Ihnen die Kategorisierung und Nachverfolgung Ihrer AWS-Kosten. Um Kostenzuweisungs-Tags für Athena zu aktivieren, verwenden Sie die AWS Billing and Cost Management-Konsole.

Weitere Informationen finden Sie in den folgenden Ressourcen.

- Markieren von Athena-Ressourcen
- Aktivieren der von AWS generierten Kostenzuordnungs-Tags
- Verwenden von AWS-Kostenzuordnungs-Tags

Kapazitätsreservierungen erstellen

Zunächst erstellen Sie eine Kapazitätsreservierung mit der Anzahl von DPUs, die Sie benötigen, und weisen dann eine oder mehrere Arbeitsgruppen zu, die diese Kapazität für ihre Abfragen verwenden. Sie können Ihre Kapazität später nach Bedarf anpassen, um eine konsistentere Leistung zu erzielen oder die Kosten besser zu verwalten. Informationen zur Schätzung Ihres Kapazitätsbedarfs finden Sie unter Kapazitätsanforderungen bestimmen.

> **Important**
>
> Kapazitätsanfragen können nicht garantiert werden und können bis zu 30 Minuten dauern.

So erstellen Sie eine Kapazitätsreservierung

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.
3. Wählen Sie Administration, Kapazitätsreservierungen aus.

4. Wählen Sie Kapazitätsreservierung erstellen.
5. Geben Sie auf der Seite Kapazitätsreservierung erstellen für den Namen der Kapazitätsreservierung den Namen ein. Der Name muss eindeutig sein, zwischen 1 und 128 Zeichen lang sein und darf nur die Zeichen a-z, A-Z, 0-9, _ (Unterstrich), enthalten. (Punkt) und - (Bindestrich). Sie können den Namen nicht mehr ändern, nachdem Sie die Reservierung erstellt haben.
6. Wählen Sie für DPU die gewünschte Anzahl von Datenverarbeitungseinheiten (DPUs) in Schritten von 4 aus. Weitere Informationen finden Sie unter DPUs verstehen.
7. (Optional) Erweitern Sie die Option Tags und wählen Sie dann Neues Tag hinzufügen aus, um ein oder mehrere benutzerdefinierte Schlüssel/Wert-Paare hinzuzufügen, die der Kapazitätsreservierungsressource zugeordnet werden sollen. Weitere Informationen finden Sie unter Markieren von Athena-Ressourcen.
8. Wählen Sie Review (Überprüfen).
9. Bestätigen Sie bei der Aufforderung zur Bestätigung der Erstellung der Kapazitätsreservierung die Anzahl der DPUs und AWS-Region weitere Informationen. Wenn Sie damit einverstanden sind, wählen Sie Senden aus.

   Auf der Detailseite wird der Status Ihrer Kapazitätsreservierung als Ausstehend angezeigt. Wenn Ihre Reservierungskapazität für Abfragen verfügbar ist, wird ihr Status als Aktiv angezeigt.

Sie können Ihrer Reservierung nun eine oder mehrere Arbeitsgruppen hinzufügen. Informationen zu den erforderlichen Schritten finden Sie unter Arbeitsgruppen zu einer Reservierung hinzufügen.

Reservierungen verwalten

Sie können Ihre Kapazitätsreservierungen auf der Seite Kapazitätsreservierungen einsehen und verwalten. Sie können Verwaltungsaufgaben wie das Hinzufügen oder Reduzieren von DPUs, das Ändern von Arbeitsgruppenzuweisungen und das Markieren oder Abbrechen von Reservierungen ausführen.

So zeigen Sie Kapazitätsreservierungen an und verwalten sie

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.
3. Wählen Sie Administration, Kapazitätsreservierungen aus.

4. Auf der Seite Kapazitätsreservierungen können Sie die folgenden Aufgaben ausführen:

   - Um eine Kapazitätsreservierung zu erstellen, wählen Sie Kapazitätsreservierung erstellen.
   - Verwenden Sie das Suchfeld, um Reservierungen nach Namen oder Anzahl der DPUs zu filtern.
   - Wählen Sie das Status-Dropdown-Menü, um nach dem Status der Kapazitätsreservierung zu filtern (z. B. Aktiv oder Storniert). Weitere Informationen zum Reservierungsstatus finden Sie unter Reservierungsstatus verstehen.
   - Um Details zu einer Kapazitätsreservierung anzuzeigen, wählen Sie den Link für die Reservierung. Die Detailseite für die Reservierung enthält Optionen zum Bearbeiten von Kapazitäten, zum Hinzufügen von Arbeitsgruppen, zum Entfernen von Arbeitsgruppen und zum Abbrechen der Reservierung.
   - Um eine Reservierung zu bearbeiten (z. B. durch Hinzufügen oder Entfernen von DPUs), klicken Sie auf die Schaltfläche für die Reservierung und wählen Sie dann Bearbeiten.
   - Um eine Reservierung zu stornieren, klicken Sie auf die Schaltfläche für die Reservierung und wählen Sie dann Abbrechen.

Reservierungsstatus verstehen

Die folgende Tabelle beschreibt die möglichen Statuswerte für eine Kapazitätsreservierung.

Status	Description
Ausstehend	Athena bearbeitet Ihre Kapazitätsanfrage. Kapazität ist nicht bereit, Abfragen auszuführen.
Aktiv	Kapazität ist verfügbar, um Abfragen auszuführen.
Fehlgeschlagen	Ihre Kapazitätsanfrage wurde nicht erfolgreich abgeschlossen. Beachten Sie, dass die Erfüllung von Kapazitätsanfragen nicht garantier t werden kann. Fehlgeschlagene Reservierungen werden auf die DPU-Limits Ihres Kontos angerechnet. Um die Nutzung freizugeben, müssen Sie die Reservierung abbrechen.
Aktualisierung ausstehend	Athena bearbeitet gerade eine Änderung der Reservierung. Dieser Status tritt beispielsweise auf, nachdem Sie die Reservierung bearbeitet haben, um DPUs hinzuzufügen oder zu entfernen.

Status	Description
Abbrechen	Athena bearbeitet eine Anfrage zur Stornierung der Reservierung. Abfragen, die in den Arbeitsgruppen, die die Reservierung verwendet haben, noch ausgeführt werden, können abgeschlossen werden, aber andere Abfragen in der Arbeitsgruppe verwenden On-Demand-Kapazität (nicht bereitgestellte).
Abgebrochen	Die Stornierung der Kapazitätsreservierung ist abgeschlossen. Stornierte Reservierungen bleiben 45 Tage lang in der Konsole. Nach 45 Tagen löscht Athena die Reservierung. Während der 45 Tage können Sie die Reservierung nicht wiederverwenden oder anderweitig verwenden. Sie können jedoch anhand der zugehörigen Tags die Details einsehen, um historische Informationen zu erhalten. Es kann nicht garantiert werden, dass stornierte Kapazitäten zu einem zukünftigen Zeitpunkt erneut reserviert werden können. Kapazität kann nicht auf eine andere Reservierung, AWS-Konto oder AWS-Region übertragen werden.

Grundlegendes zu aktiven DPUs und Ziel-DPUs

In der Liste der Kapazitätsreservierungen in der Athena-Konsole werden für Ihre Reservierung zwei DPU-Werte angezeigt: Aktive DPU und Ziel-DPU.

- Aktive DPU – Die Anzahl der DPUs, die in Ihrer Reservierung für die Ausführung von Abfragen verfügbar sind. Wenn Sie beispielsweise 100 DPUs anfordern und Ihre Anfrage erfüllt ist, zeigt aktive DPU 100 an.
- Ziel-DPU – Die Anzahl der DPUs, auf die Ihre Reservierung gerade umgestellt wird. Die Ziel-DPU zeigt einen anderen Wert als die aktive DPU an, wenn eine Reservierung erstellt wird oder eine Erhöhung oder Verringerung der Anzahl der DPUs aussteht.

Wenn Sie beispielsweise eine Anfrage zur Erstellung einer Reservierung mit 24 DPUs eingereicht haben, lautet der Reservierungsstatus Ausstehend, die aktive DPU ist 0 und die Ziel-DPU ist 24.

Wenn Sie eine Reservierung mit 100 DPUs haben und Ihre Reservierung bearbeiten, um eine Erhöhung um 20 DPUs zu beantragen, lautet der Status Aktualisierung ausstehend, die aktive DPU ist 100 und die Ziel-DPU ist 120.

Wenn Sie eine Reservierung mit 100 DPUs haben und Ihre Reservierung bearbeiten, um eine Erhöhung um 20 DPUs zu beantragen, lautet der Status Aktualisierung ausstehend, die aktive DPU ist 100 und die Ziel-DPU ist 80.

Während dieser Übergänge arbeitet Athena aktiv daran, die Anzahl der DPUs auf Ihre Anfrage hin zu erwerben oder zu reduzieren. Wenn die aktive DPU der Ziel-DPU entspricht, wurde die Zielzahl erreicht und es stehen keine Änderungen mehr aus.

Um diese Werte programmgesteuert abzurufen, können Sie die API-Aktion GetCapacityReservation aufrufen. Die API bezeichnet aktive DPU und Ziel-DPU als `AllocatedDpus` und `TargetDpus`.

Themen

Kapazitätsreservierungen bearbeiten

Nachdem Sie eine Kapazitätsreservierung erstellt haben, können Sie die Anzahl der DPUs anpassen und ihre benutzerdefinierten Tags hinzufügen oder entfernen.

So bearbeiten Sie eine Kapazitätsreservierung

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.
3. Wählen Sie Administration, Kapazitätsreservierungen aus.
4. Führen Sie in der Liste der Kapazitätsreservierungen einen der folgenden Schritte aus:
   - Wählen Sie die Schaltfläche neben dem Katalognamen und anschließend Bearbeiten aus.
   - Wählen Sie den Reservierungslink und dann Bearbeiten aus.

5. Wählen Sie für DPU die gewünschte Anzahl von Datenverarbeitungseinheiten in Schritten von 4 aus. Sie müssen mindestens 24 DPUs haben. Weitere Informationen finden Sie unter DPUs verstehen.

> **Note**
>
> Sie können einer bestehenden Kapazitätsreservierung jederzeit DPUs hinzufügen. Sie können die Anzahl der DPUs jedoch erst 1 Stunde nach der Erstellung der Reservierung oder dem Hinzufügen von DPUs verringern.

6. (Optional) Wählen Sie bei Tags die Option Entfernen, um ein Tag zu entfernen, oder wählen Sie Neues Tag hinzufügen, um ein neues Tag hinzuzufügen.
7. Wählen Sie Absenden aus. Die Detailseite für die Reservierung zeigt die aktualisierte Konfiguration.

Arbeitsgruppen zu einer Reservierung hinzufügen

Nachdem Sie eine Kapazitätsreservierung erstellt haben, können Sie der Reservierung bis zu 20 Arbeitsgruppen hinzufügen. Das Hinzufügen einer Arbeitsgruppe zu einer Reservierung teilt Athena mit, welche Abfragen auf Ihrer reservierten Kapazität ausgeführt werden sollen. Abfragen von Arbeitsgruppen, die keiner Reservierung zugeordnet sind, werden weiterhin mit dem standardmäßigen Preismodell pro gescanntem Terabyte (TB) ausgeführt.

Wenn eine Reservierung zwei oder mehr Arbeitsgruppen umfasst, können Abfragen von diesen Arbeitsgruppen die Kapazität der Reservierung beanspruchen. Sie können Arbeitsgruppen jederzeit hinzufügen oder entfernen. Wenn Sie Arbeitsgruppen hinzufügen oder entfernen, werden laufende Abfragen nicht unterbrochen.

Wenn sich Ihre Reservierung im Status „Ausstehend“ befindet, werden Abfragen von Arbeitsgruppen, die Sie hinzugefügt haben, weiterhin mit dem Standardpreismodell pro gescanntem Terabyte (TB) ausgeführt, bis die Reservierung aktiv wird.

Um Ihrer Kapazitätsreservierung eine oder mehrere Arbeitsgruppen hinzuzufügen

1. Wählen Sie auf der Detailseite für die Kapazitätsreservierung die Option Arbeitsgruppen hinzufügen aus.

2. Wählen Sie auf der Seite Arbeitsgruppen hinzufügen die Arbeitsgruppen aus, die Sie hinzufügen möchten, und klicken Sie dann auf Arbeitsgruppen hinzufügen. Sie können eine Arbeitsgruppe nicht mehr als einer Reservierung zuordnen.

   Auf der Detailseite für Ihre Kapazitätsreservierung sind die Arbeitsgruppen aufgeführt, die Sie hinzugefügt haben. Abfragen, die in diesen Arbeitsgruppen ausgeführt werden, verwenden die Kapazität, die Sie reserviert haben, wenn die Reservierung aktiv ist.

Eine Arbeitsgruppe aus einer Reservierung entfernen

Wenn Sie keine dedizierte Kapazität länger benötigen oder eine Arbeitsgruppe dementsprechend nicht länger benötigen, können Sie sie jederzeit entfernen. Das Entfernen einer Arbeitsgruppe aus einer Reservierung ist ein unkomplizierter Vorgang. Nachdem Sie eine Arbeitsgruppe aus einer Reservierung entfernt haben, verwenden Abfragen der entfernten Arbeitsgruppe standardmäßig On-Demand-Kapazität (nicht bereitgestellte) und werden auf der Grundlage der gescannten Terabyte (TB) abgerechnet.

So entfernen Sie eine oder mehrere Arbeitsgruppen aus einer Reservierung

1. Wählen Sie auf der Detailseite für die Kapazitätsreservierung die Arbeitsgruppen aus, die Sie entfernen möchten.
2. Wählen Sie Arbeitsgruppen entfernen. Die Eingabeaufforderung Arbeitsgruppen entfernen? informiert Sie darüber, dass alle derzeit aktiven Abfragen beendet werden müssen, bevor die Arbeitsgruppe aus der Reservierung entfernt wird.
3. Wählen Sie Remove (Entfernen) aus. Auf der Detailseite für Ihre Kapazitätsreservierung wird angezeigt, dass die entfernten Arbeitsgruppen nicht mehr vorhanden sind.

Eine Kapazitätsreservierung abbrechen

Wenn Sie eine Kapazitätsreservierung nicht länger benötigen, können Sie sie abbrechen. Abfragen, die noch in den Arbeitsgruppen laufen, die die Reservierung verwendet haben, können beendet werden, aber andere Abfragen in der Arbeitsgruppe verwenden die Reservierung nicht mehr.

> **Note**
>
> Es kann nicht garantiert werden, dass stornierte Kapazitäten zu einem zukünftigen Zeitpunkt erneut reserviert werden können. Kapazität kann nicht auf eine andere Reservierung, AWS-Konto oder AWS-Region übertragen werden.

So brechen Sie eine Kapazitätsreservierung ab

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.
3. Wählen Sie Administration, Kapazitätsreservierungen aus.
4. Führen Sie in der Liste der Kapazitätsreservierungen einen der folgenden Schritte aus:
   - Wählen Sie die Schaltfläche neben dem Katalognamen und anschließend Bearbeiten aus.
   - Wählen Sie den Link zur Reservierung und wählen Sie dann Kapazitätsreservierung abbrechen.
5. Unter Kapazitätsreservierung abbrechenen? Geben Sie in der Befehlszeile Abbrechen ein und wählen Sie dann Kapazitätsreservierung abbbrechen aus.

   Der Status der Reservierung ändert sich in Storniere, und ein Fortschrittsbanner informiert Sie darüber, dass die Stornierung im Gange ist.

   Wenn die Stornierung abgeschlossen ist, bleibt die Kapazitätsreservierung bestehen, ihr Status wird jedoch als Storniert angezeigt. Die Reservierung wird 45 Tage nach der Stornierung gelöscht. Während der 45 Tage können Sie die beendete Reservierung nicht wiederverwenden oder anderweitig verwenden. Sie können jedoch anhand der zugehörigen Tags die Details einsehen, um historische Informationen zu erhalten.

Eine Kapazitätsreservierung löschen

Wenn Sie alle Verweise auf eine stornierte Kapazitätsreservierung entfernen möchten, können Sie die Reservierung löschen. Eine Reservierung muss storniert werden, bevor sie gelöscht werden kann. Eine gelöschte Reservierung wird sofort aus Ihrem Konto entfernt und kann nicht mehr referenziert werden, auch nicht anhand ihres ARN.

So löschen Sie eine Kapazitätsreservierung

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.
3. Wählen Sie Administration, Kapazitätsreservierungen aus.
4. Führen Sie in der Liste der Kapazitätsreservierungen einen der folgenden Schritte aus:
   - Wählen Sie die Schaltfläche neben dem Katalognamen und anschließend Aktionen, Löschen aus.
   - Wählen Sie den Reservierungslink und dann Löschen aus.
5. Bei der Eingabeaufforderung Kapazitätsreservierung löschen?, wählen Sie Löschen.

   Ein Banner informiert Sie darüber, dass die Kapazitätsreservierung erfolgreich gelöscht wurde. Die gelöschte Reservierung erscheint nicht mehr in der Liste der Kapazitätsreservierungen.

IAM-Richtlinien für Kapazitätsreservierungen

Verwenden Sie zur Steuerung des Zugriffs auf Kapazitätsreservierungen IAM-Berechtigungen auf Ressourcenebene oder identitätsbasierte IAM-Richtlinien. Wenn Sie IAM-Richtlinien verwenden, stellen Sie sicher, dass Sie die bewährten IAM-Methoden befolgen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Das folgende Verfahren gilt speziell für Athena.

IAM-spezifische Informationen finden Sie unter den Links am Ende dieses Abschnitts. Weitere Informationen zu JSON-Kapazitätsreservierungsrichtlinien finden Sie unter Beispielrichtlinien für Kapazitätsreservierungen.

Verwenden Sie den visuellen Editor in der IAM-Konsole, um eine Kapazitätsreservierungsrichtlinie zu erstellen

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich auf der linken Seite Policies (Richtlinien) und dann Create Policy (Richtlinie erstellen) aus.

3. Wählen Sie auf der Registerkarte Visual editor (Visueller Editor) die Option Choose a service (Wählen Sie einen Service) aus. Wählen Sie dann Athena aus, um es der Richtlinie hinzuzufügen.
4. Wählen Sie Select actions (Aktionen auswählen) und dann die Aktionen aus, die Sie der Richtlinie hinzufügen möchten. Im visuellen Editor werden die in Athena verfügbaren Aktionen angezeigt. Weitere Informationen finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Athena in der Service-Autorisierungs-Referenz.
5. Wählen Sie Aktionen hinzufügen aus, um eine bestimmte Aktion einzugeben, oder verwenden Sie Platzhalter (*), um mehrere Aktionen anzugeben.

   Standardmäßig lässt die Richtlinie, die Sie erstellen, die Aktionen zu, die Sie auswählen. Wenn Sie eine oder mehrere Aktionen auswählen, die Berechtigungen auf Ressourcenebene für die `capacity-reservation`-Ressource in Athena unterstützen, listet der Editor die `capacity-reservation`-Ressource auf.
6. Wählen Sie Ressourcen aus, um die Kapazitätsreservierungen für Ihre Richtlinie festzulegen. Beispiele für JSON-Kapazitätsreservierungsrichtlinien finden Sie unter Beispielrichtlinien für Kapazitätsreservierungen.
7. Geben Sie die `capacity-reservation`-Ressource wie folgt an:

   ```
   arn:aws:athena:<region>:<user-account>:capacity-reservation/<capacity-reservation-name>
   ```
8. Wählen Sie Review policy (Richtlinie prüfen) aus und geben Sie Name und Description (Beschreibung) (optional) für die zu erstellende Richtlinie ein. Prüfen Sie die Richtlinienübersicht, um sicherzustellen, dass Sie die beabsichtigten Berechtigungen erteilt haben.
9. Wählen Sie Create policy (Richtlinie erstellen) aus, um Ihre neue Richtlinie zu speichern.
10. Hängen Sie diese identitätsbasierte Richtlinie an einen Benutzer, eine Gruppe oder eine Rolle an.

Weitere Informationen finden Sie in den folgenden Themen in der Service Authorization Reference und im IAM-Benutzerhandbuch:

- Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Athena
- Erstellen von Richtlinien mit dem visuellen Editor
- Hinzufügen und Entfernen von IAM-Richtlinien
- Steuern des Zugriffs auf Ressourcen

Beispiele für JSON-Kapazitätsreservierungsrichtlinien finden Sie unter Beispielrichtlinien für Kapazitätsreservierungen.

Eine vollständige Liste mit Amazon-Athena-Aktionen finden Sie unter den Namen von API-Aktionen in der Amazon-Athena-API-Referenz.

Beispielrichtlinien für Kapazitätsreservierungen

Dieser Abschnitt enthält Beispielrichtlinien, mit denen Sie verschiedene Aktionen für Kapazitätsreservierungen aktivieren können. Wenn Sie IAM-Richtlinien verwenden, stellen Sie sicher, dass Sie die bewährten IAM-Methoden befolgen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Eine Kapazitätsreservierung ist eine IAM-Ressource, die von Athena verwaltet wird. Wenn Ihre Kapazitätsreservierungsrichtlinie Aktionen verwendet, die `capacity-reservation` als Eingabe verwenden, müssen Sie daher den ARN der Kapazitätsreservierung wie folgt angeben:

```
"Resource": [arn:aws:athena:<region>:<user-account>:capacity-reservation/<capacity-reservation-name>]
```

Dabei ist *<capacity-reservation-name>* der Name Ihrer Kapazitätsreservierung. Geben Sie diesen beispielsweise für eine Kapazitätsreservierung mit dem Namen `test_capacity_reservation` wie folgt als Ressource an:

```
"Resource": ["arn:aws:athena:us-east-1:123456789012:capacity-reservation/test_capacity_reservation"]
```

Eine vollständige Liste mit Amazon-Athena-Aktionen finden Sie unter den Namen von API-Aktionen in der Amazon-Athena-API-Referenz. Weitere Informationen zu IAM-Richtlinien finden Sie unter Erstellen von Richtlinien mit dem visuellen Editor im IAM-Benutzerhandbuch.

- Example policy to list capacity reservations
- Example policy for management operations

Example Beispielrichtlinie zum Auflisten von Kapazitätsreservierungen

Die folgende Richtlinie ermöglicht es allen Benutzern, alle Kapazitätsreservierungen aufzulisten.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListCapacityReservations"
            ],
            "Resource": "*"
        }
    ]
}
```

Example Beispielrichtlinie für Verwaltungsoperationen

Die folgende Richtlinie ermöglicht es einem Benutzer, die Kapazitätsreservierung `test_capacity_reservation` zu erstellen, zu stornieren, Details abzurufen und zu aktualisieren. Die Richtlinie ermöglicht es einem Benutzer auch, Zuweisungen `workgroupA` und `workgroupB` zu `test_capacity_reservation` zuzuweisen.

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

Athena-APIs zur Kapazitätsreservierung

Die folgende Liste enthält Referenzlinks zu den Athena-Kapazitätsreservierungs-API-Aktionen. Datenstrukturen und andere Athena-API-Aktionen finden Sie in der Amazon-Athena-API-Referenz.

- CancelCapacityReservation
- CreateCapacityReservation
- GetCapacityAssignmentConfiguration
- GetCapacityReservation
- ListCapacityReservations
- PutCapacityAssignmentConfiguration
- UpdateCapacityReservation

Leistungsoptimierung in Athena

Dieses Thema enthält allgemeine Informationen und spezifische Vorschläge zur Verbesserung der Leistung Ihrer Athena-Abfragen und zur Umgehung von Fehlern im Zusammenhang mit Beschränkungen und der Ressourcennutzung.

Service Quotas

Athena setzt Kontingente für Metriken wie die Laufzeit von Abfragen, die Anzahl gleichzeitiger Abfragen in einem Konto und die API-Anforderungsraten durch. Weitere Hinweise zu diesen Kontingenten finden Sie unter Service Quotas. Eine Überschreitung dieser Kontingente führt dazu, dass eine Abfrage fehlschlägt – entweder beim Absenden oder während der Abfrageausführung.

Viele der Tipps zur Leistungsoptimierung auf dieser Seite können dazu beitragen, die Laufzeit von Abfragen zu reduzieren. Durch die Optimierung wird Kapazität freigesetzt, sodass Sie mehr Abfragen innerhalb des Parallelitätskontingents ausführen können, und verhindert, dass Abfragen abgebrochen werden, weil sie zu lange laufen.

Die Kontingente für die Anzahl gleichzeitiger Abfragen und API-Anfragen gelten pro AWS-Konto und AWS-Region. Wir empfehlen, einen Workload pro Workload auszuführen AWS-Konto (oder separate bereitgestellte Kapazitätsreservierungen zu verwenden), um zu verhindern, dass Workloads um dasselbe Kontingent konkurrieren.

Wenn Sie zwei Workloads in demselben Konto ausführen, kann einer der Workloads eine ganze Reihe von Abfragen ausführen. Dies kann dazu führen, dass der verbleibende Workload gedrosselt

oder daran gehindert wird, Abfragen auszuführen. Um dies zu vermeiden, können Sie die Workloads auf separate Konten verschieben, um jedem Workload ein eigenes Parallelitätskontingent zuzuweisen. Durch das Erstellen einer bereitgestellten Kapazitätsreservierung für eine oder beide Workloads wird dasselbe Ziel erreicht.

Kontingente in anderen Services

Wenn Athena eine Abfrage ausführt, kann es andere Services aufrufen, die Kontingente durchsetzen. Während der Abfrageausführung kann Athena API-Aufrufe an Amazon S3 und andere AWS Dienste wie IAM und tätigen. AWS Glue Data Catalog AWS KMS Wenn Sie föderierte Abfragen verwenden, ruft Athena auch auf. AWS Lambda Alle diese Services haben ihre eigenen Grenzwerte und Kontingente, die überschritten werden können. Wenn bei der Ausführung einer Abfrage Fehler von diesen Services auftreten, schlägt sie fehl und schließt den Fehler des Quellservices mit ein. Behebbare Fehler werden wiederholt, aber Abfragen können trotzdem fehlschlagen, wenn sich das Problem nicht rechtzeitig von selbst löst. Lesen Sie sich die Fehlermeldungen sorgfältig durch, um festzustellen, ob sie von Athena oder einem anderen Service stammen. Einige der relevanten Fehler werden in diesem Dokument behandelt.

Weitere Informationen zur Umgehung von Fehlern, die durch Amazon-S3-Servicekontingente verursacht werden, finden Sie unter Zu viele Dateien vermeiden weiter unten in diesem Dokument. Weitere Informationen zur Amazon-S3-Leistungsoptimierung finden Sie unter Bewährte Entwurfsmuster: Optimierung der Leistung von Amazon S3 im Amazon-S3-Benutzerhandbuch.

Ressourcenlimits

Athena führt Abfragen in einer verteilten Abfrage-Engine aus. Wenn Sie eine Abfrage einreichen, schätzt der Athena-Engine-Abfrageplaner die Rechenkapazität, die für die Ausführung der Abfrage erforderlich ist, und bereitet entsprechend einen Cluster von Rechenknoten vor. Einige Abfragen wie DDL-Abfragen werden nur auf einem Knoten ausgeführt. Komplexe Abfragen über große Datensätze werden auf viel größeren Clustern ausgeführt. Die Knoten sind einheitlich und haben dieselben Speicher-, CPU- und Festplattenkonfigurationen. Athena aufskaliert, nicht ab, um anspruchsvollere Anfragen zu bearbeiten.

Manchmal übersteigen die Anforderungen einer Abfrage die Ressourcen, die dem Cluster zur Verfügung stehen, auf dem die Abfrage ausgeführt wird. In diesem Fall schlägt die Abfrage fehl und es wird folgende Fehlermeldung angezeigt: Abfrage erschöpfte Ressourcen mit diesem Skalierungsfaktor.

Die am häufigsten erschöpfte Ressource ist Arbeitsspeicher, in seltenen Fällen kann es sich aber auch um Festplattenspeicher handeln. Speicherfehler treten häufig auf, wenn die Engine eine

Join- oder Fensterfunktion ausführt. Sie können jedoch auch in unterschiedlichen Zählungen und Aggregationen auftreten.

Selbst wenn eine Abfrage einmal mit dem Fehler „Nicht genügend Ressourcen“ fehlschlägt, kann sie erfolgreich sein, wenn Sie sie erneut ausführen. Die Abfrageausführung ist nicht deterministisch. Faktoren wie die Dauer des Ladens von Daten und die Verteilung der Zwischendatensätze auf die Knoten können zu einer unterschiedlichen Ressourcennutzung führen. Stellen Sie sich beispielsweise eine Abfrage vor, die zwei Tabellen miteinander verbindet und die Verteilung der Werte für die Join-Bedingung stark verzerrt ist. Eine solche Abfrage kann in den meisten Fällen erfolgreich sein, schlägt aber gelegentlich fehl, wenn die häufigsten Werte von demselben Knoten verarbeitet werden.

Verwenden Sie die in diesem Dokument genannten Tipps zur Leistungsoptimierung, um zu verhindern, dass Ihre Abfragen die verfügbaren Ressourcen überschreiten. Insbesondere Tipps zur Optimierung von Abfragen, die die verfügbaren Ressourcen erschöpfen, finden Sie unter Verknüpfungen optimieren, Fensterfunktionen optimieren und Optimieren von Abfragen mithilfe von Näherungen.

Methoden zur Abfragenoptimierung

Verwenden Sie die in diesem Abschnitt beschriebenen Techniken zur Abfrageoptimierung, um Abfragen schneller ausführen zu lassen oder um das Problem für Abfragen zu umgehen, die die Ressourcenlimits in Athena überschreiten.

Verknüpfungen optimieren

Es gibt viele verschiedene Strategien für die Ausführung von Verknüpfungen in einer verteilten Abfrage-Engine. Zwei der gängigsten sind verteilte Hash-Joins und Abfragen mit komplexen Join-Bedingungen.

Verteilter Hash-Join

Der gebräuchlichste Join-Typ verwendet einen Gleichheitsvergleich als Join-Bedingung. Athena führt diese Art von Join als verteilten Hash-Join aus.

Bei einem verteilten Hash-Join erstellt die Engine aus einer der Seiten des Joins eine Nachschlagetabelle (Hashtabelle). Diese Seite wird die Build-Seite genannt. Die Datensätze der Build-Seite sind auf die Knoten verteilt. Jeder Knoten erstellt eine Nachschlagetabelle für seine Teilmenge. Die andere Seite des Joins, die so genannte Test-Seite, wird dann durch die Knoten

gestreamt. Die Datensätze von der Testseite werden auf die gleiche Weise wie auf der Build-Seite auf die Knoten verteilt. Auf diese Weise kann jeder Knoten die Verknüpfung durchführen, indem er die entsprechenden Datensätze in seiner eigenen Nachschlagetabelle nachschlägt.

Wenn die auf der Build-Seite der Verknüpfung erstellten Nachschlagetabellen nicht in den Arbeitsspeicher passen, können Abfragen fehlschlagen. Selbst wenn die Gesamtgröße der Build-Seite kleiner als der verfügbare Speicher ist, können Abfragen fehlschlagen, wenn die Verteilung der Datensätze stark verzerrt ist. Im Extremfall könnten alle Datensätze denselben Wert für die Join-Bedingung haben und müssen in den Speicher eines einzelnen Knotens passen. Selbst eine Abfrage mit weniger Verzerrungen kann fehlschlagen, wenn ein Satz von Werten an denselben Knoten gesendet wird und die Summe der Werte den verfügbaren Speicher übersteigt. Knoten können zwar Datensätze auf die Festplatte übertragen, aber ein Datenverlust verlangsamt die Abfrageausführung und kann nicht ausreichen, um zu verhindern, dass die Abfrage fehlschlägt.

Athena versucht, Verbindungen neu anzuordnen, um die größere Relation als Testseite und die kleinere Relation als Build-Seite zu verwenden. Da Athena die Daten in Tabellen jedoch nicht verwaltet, verfügt es nur über begrenzte Informationen und muss häufig davon ausgehen, dass die erste Tabelle größer und die zweite Tabelle kleiner ist.

Gehen Sie beim Schreiben von Verknüpfungen mit gleichheitsbasierten Join-Bedingungen davon aus, dass die Tabelle links vom `JOIN`-Schlüsselwort die Testseite und die Tabelle rechts die Build-Seite ist. Stellen Sie sicher, dass die rechte Tabelle, die Build-Seite, die kleinere der Tabellen ist. Wenn es nicht möglich ist, die Build-Seite der Verknüpfung so klein zu machen, dass sie in den Arbeitsspeicher passt, sollten Sie mehrere Abfragen ausführen, die Teilmengen der Build-Tabelle verknüpfen.

Andere Join-Typen

Abfragen mit komplexen Verbindungsbedingungen (z. B. Abfragen, die `LIKE`, > oder andere Operatoren verwenden) sind häufig rechenintensiv. Im schlimmsten Fall muss jeder Datensatz von einer Seite der Verknüpfung mit jedem Datensatz auf der anderen Seite der Verknüpfung verglichen werden. Da die Ausführungszeit mit dem Quadrat der Anzahl der Datensätze wächst, besteht bei solchen Abfragen das Risiko, dass die maximale Ausführungszeit überschritten wird.

Um im Voraus herauszufinden, wie Athena Ihre Anfrage ausführt, können Sie die `EXPLAIN`-Anweisung verwenden. Weitere Informationen finden Sie unter Verwenden von EXPLAIN und EXPLAIN ANALYZE in Athena und Die Ergebnisse der Athena-EXPLAIN-Anweisung verstehen.

Fensterfunktionen optimieren

Da es sich bei Fensterfunktionen um ressourcenintensive Vorgänge handelt, können sie dazu führen, dass Abfragen langsam ausgeführt werden oder sogar fehlschlagen und die Meldung Erschöpfte Ressourcen bei diesem Skalierungsfaktor abfragen angezeigt wird. Fensterfunktionen behalten alle Datensätze, mit denen sie arbeiten, im Speicher, um ihr Ergebnis zu berechnen. Wenn das Fenster sehr groß ist, kann der Speicherplatz der Fensterfunktion knapp werden.

Um sicherzustellen, dass Ihre Abfragen innerhalb der verfügbaren Speichergrenzen ausgeführt werden, reduzieren Sie die Größe der Fenster, in denen Ihre Fensterfunktionen ausgeführt werden. Dazu können Sie eine `PARTITIONED BY`-Klausel hinzufügen oder den Geltungsbereich vorhandener Partitionierungsklauseln einschränken.

Verwenden Sie stattdessen Funktionen, die keine Fenster sind

Manchmal können Abfragen mit Fensterfunktionen ohne Fensterfunktionen neu geschrieben werden. So können Sie z. B. anstelle von `row_number` für die Suche nach den ersten `N` Datensätzen `ORDER BY` und `LIMIT` verwenden. Anstatt `row_number` oder `rank` zu verwenden, um Datensätze zu deduplizieren, können Sie Aggregatfunktionen wie max_by, min_by und arbiträr verwenden.

Nehmen wir zum Beispiel an, Sie haben einen Datensatz mit Aktualisierungen von einem Sensor. Der Sensor meldet regelmäßig seinen Batteriestatus und enthält einige Metadaten wie den Standort. Wenn Sie den letzten Batteriestatus für jeden Sensor und seinen Standort wissen möchten, können Sie diese Abfrage verwenden:

```
SELECT sensor_id,
       arbitrary(location) AS location,
       max_by(battery_status, updated_at) AS battery_status
FROM sensor_readings
GROUP BY sensor_id
```

Da Metadaten wie der Standort für jeden Datensatz identisch sind, können Sie die `arbitrary`-Funktion verwenden, um einen beliebigen Wert aus der Gruppe auszuwählen.

Um den letzten Batteriestatus abzurufen, können Sie die `max_by`-Funktion verwenden. Die `max_by`-Funktion wählt den Wert für eine Spalte aus dem Datensatz aus, in dem der Maximalwert einer anderen Spalte gefunden wurde. In diesem Fall gibt sie den Batteriestatus für den Datensatz mit dem Zeitpunkt der letzten Aktualisierung innerhalb der Gruppe zurück. Diese Abfrage wird schneller ausgeführt und benötigt weniger Speicher als eine entsprechende Abfrage mit einer Fensterfunktion.

Aggregationen optimieren

Wenn Athena eine Aggregation durchführt, verteilt es die Datensätze mithilfe der Spalten in der `GROUP BY`-Klausel auf die Worker-Knoten. Um die Zuordnung von Datensätzen zu Gruppen so effizient wie möglich zu gestalten, versuchen die Knoten, Datensätze im Speicher zu behalten, sie aber bei Bedarf auf die Festplatte zu übertragen.

Es ist auch eine gute Idee, die Aufnahme überflüssiger Spalten in `GROUP BY`-Klauseln zu vermeiden. Da weniger Spalten weniger Speicher benötigen, ist eine Abfrage, die eine Gruppe mit weniger Spalten beschreibt, effizienter. Numerische Spalten verbrauchen auch weniger Speicher als Zeichenketten. Wenn Sie beispielsweise einen Datensatz aggregieren, der sowohl eine numerische Kategorie-ID als auch einen Kategorienamen hat, verwenden Sie in der `GROUP BY`-Klausel nur die Kategorie-ID-Spalte.

Manchmal enthalten Abfragen Spalten in der `GROUP BY`-Klausel, um die Tatsache zu umgehen, dass eine Spalte entweder Teil der `GROUP BY`-Klausel oder ein Aggregatausdruck sein muss. Wenn diese Regel nicht befolgt wird, erhalten Sie möglicherweise eine Fehlermeldung wie die folgende:

EXPRESSION_NOT_AGGREGATE: Zeile 1:8: 'Kategorie' muss ein Aggregatausdruck sein oder in der GROUP-BY-Klausel vorkommen

Um zu vermeiden, dass der `GROUP BY`-Klausel redundante Spalten hinzugefügt werden müssen, können Sie die arbitrary-Funktion verwenden, wie im folgenden Beispiel.

```
SELECT country_id,
       arbitrary(country_name) AS country_name,
       COUNT(*) AS city_count
FROM world_cities
GROUP BY country_id
```

Die `ARBITRARY`-Funktion gibt einen beliebigen Wert aus der Gruppe zurück. Die Funktion ist nützlich, wenn Sie wissen, dass alle Datensätze in der Gruppe denselben Wert für eine Spalte haben, der Wert die Gruppe jedoch nicht identifiziert.

Die wichtigsten N-Abfragen optimieren

Die `ORDER BY`-Klausel gibt die Ergebnisse einer Abfrage in sortierter Reihenfolge zurück. Athena verwendet verteilte Sortierung, um den Sortiervorgang parallel auf mehreren Knoten auszuführen.

Wenn Sie nicht unbedingt möchten, dass Ihr Ergebnis sortiert wird, vermeiden Sie das Hinzufügen einer `ORDER BY`-Klausel. Vermeiden Sie auch, innere Abfragen mit `ORDER BY` zu versehen, wenn

sie nicht unbedingt notwendig sind. In vielen Fällen kann der Abfrageplaner überflüssige Sortierungen entfernen, dies kann jedoch nicht garantiert werden. Eine Ausnahme von dieser Regel ist, wenn eine interne Abfrage einen wichtigen N-Vorgang ausführt, z. B. die Suche nach den N neuesten oder N gängigsten Werten.

Wenn Athena `ORDER BY` zusammen mit `LIMIT` sieht, versteht es, dass Sie eine N-Top-Abfrage ausführen, und verwendet entsprechend dedizierte Vorgänge.

> **Note**
>
> Obwohl Athena häufig auch Fensterfunktionen wie `row_number`, die Top-N verwendet, erkennen kann, empfehlen wir die einfachere Version, die `ORDER BY` und `LIMIT` verwendet. Weitere Informationen finden Sie unter Fensterfunktionen optimieren.

Nur die erforderlichen Spalten einschließen

Wenn Sie eine Spalte nicht unbedingt benötigen, nehmen Sie sie nicht in Ihre Abfrage auf. Je weniger Daten eine Abfrage verarbeiten muss, desto schneller wird sie ausgeführt. Dies reduziert sowohl den benötigten Speicher als auch die Datenmenge, die zwischen den Knoten gesendet werden muss. Wenn Sie ein spaltenorientiertes Dateiformat verwenden, reduziert die Reduzierung der Anzahl der Spalten auch die Datenmenge, die aus Amazon S3 gelesen wird.

Athena hat keine spezifische Begrenzung für die Anzahl der Spalten in einem Ergebnis, aber die Art und Weise, wie Abfragen ausgeführt werden, begrenzt die mögliche kombinierte Größe von Spalten. Die kombinierte Größe von Spalten umfasst ihre Namen und Typen.

Der folgende Fehler wird beispielsweise durch eine Beziehung verursacht, die die Größenbeschränkung für einen Beziehungsdeskriptor überschreitet:

GENERIC_INTERNAL_ERROR: io.airlift.bytecode. CompilationException

Um dieses Problem zu umgehen, reduzieren Sie die Anzahl der Spalten in der Abfrage, oder erstellen Sie Unterabfragen und verwenden Sie eine `JOIN`, die eine kleinere Datenmenge abruft. Wenn Sie Abfragen haben, die `SELECT *` in der äußersten Abfrage ausführen, sollten Sie die * in eine Liste mit nur den Spalten ändern, die Sie brauchen.

Optimieren von Abfragen mithilfe von Näherungen

Athena unterstützt Näherungsaggregatfunktionen zum Zählen verschiedener Werte, der häufigsten Werte, Perzentile (einschließlich ungefährer Mediane) und zum Erstellen von Histogrammen. Verwenden Sie diese Funktionen immer dann, wenn keine exakten Werte benötigt werden.

Im Gegensatz zu `COUNT(DISTINCT col)`-Vorgängen benötigt approx_distinct viel weniger Speicher und läuft schneller. In ähnlicher Weise werden bei der Verwendung von numeric_histogram anstelle von Histogramm Näherungsmethoden und damit weniger Speicher verwendet.

LIKE optimieren

Sie können `LIKE` es verwenden, um passende Zeichenketten zu finden, aber bei langen Zeichenketten ist das rechenintensiv. Die Funktion regexp_like ist in den meisten Fällen eine schnellere Alternative und bietet auch mehr Flexibilität.

Oft können Sie eine Suche optimieren, indem Sie die gesuchte Teilzeichenfolge verankern. Wenn Sie beispielsweise nach einem Präfix suchen, ist es viel besser, '*substr*%' anstelle von '%*substr*%' zu verwenden. *Oder, wenn Sie regexp_like verwenden, '^substr*'.

Verwenden Sie UNION ALL anstelle von UNION

`UNION ALL` und `UNION` sind zwei Möglichkeiten, die Ergebnisse von zwei Abfragen zu einem Ergebnis zu kombinieren. `UNION ALL` verkettet die Datensätze aus der ersten Abfrage mit der zweiten und `UNION` macht dasselbe, entfernt aber auch Duplikate. `UNION` muss alle Datensätze verarbeiten und die Duplikate finden, was speicher- und rechenintensiv ist, aber `UNION ALL` ist ein relativ schneller Vorgang. Sofern Sie Datensätze nicht deduplizieren müssen, verwenden Sie `UNION ALL`, um die beste Leistung zu erzielen.

Verwenden Sie UNLOAD für große Ergebnismengen

Wenn die Ergebnisse einer Abfrage voraussichtlich umfangreich sein werden (z. B. Zehntausende von Zeilen oder mehr), verwenden Sie UNLOAD, um die Ergebnisse zu exportieren. In den meisten Fällen ist dies schneller als das Ausführen einer normalen Abfrage, und die Verwendung von `UNLOAD` gibt Ihnen auch mehr Kontrolle über die Ausgabe.

Wenn die Ausführung einer Abfrage abgeschlossen ist, speichert Athena das Ergebnis als einzelne unkomprimierte CSV-Datei auf Amazon S3. Dies dauert länger als `UNLOAD`, nicht nur, weil das Ergebnis unkomprimiert ist, sondern auch, weil der Vorgang nicht parallelisiert werden kann. Im Gegensatz dazu schreibt `UNLOAD` die Ergebnisse direkt von den Worker-Knoten und nutzt die Parallelität des Rechenclusters voll aus. Darüber hinaus können Sie `UNLOAD` so konfigurieren, dass

die Ergebnisse in komprimiertem Format und in anderen Dateiformaten wie JSON und Parquet geschrieben werden.

Weitere Informationen finden Sie unter UNLOAD.

Verwenden Sie CTAS oder Glue ETL, um häufig verwendete Aggregationen zu materialisieren

Das „Materialisieren“ einer Abfrage ist eine Möglichkeit, die Abfrageleistung zu beschleunigen, indem vorab berechnete komplexe Abfrageergebnisse (z. B. Aggregationen und Verknüpfungen) zur Wiederverwendung in nachfolgenden Abfragen gespeichert werden.

Wenn viele Ihrer Abfragen dieselben Verknüpfungen und Aggregationen enthalten, können Sie die allgemeine Unterabfrage als neue Tabelle materialisieren und dann Abfragen für diese Tabelle ausführen. Sie können die neue Tabelle mit Erstellen einer Tabelle aus Abfrageergebnissen (CTAS) oder einem speziellen ETL-Tool wie Glue ETL erstellen.

Nehmen wir zum Beispiel an, Sie haben ein Dashboard mit Widgets, die verschiedene Aspekte eines Auftragsdatensatzes zeigen. Jedes Widget hat seine eigene Abfrage, aber die Abfragen verwenden alle dieselben Verknüpfungen und Filter. Eine Bestelltabelle wird mit einer Einzelpostentabelle verknüpft, und es gibt einen Filter, der nur die letzten drei Monate anzeigt. Wenn Sie die gemeinsamen Features dieser Abfragen identifizieren, können Sie eine neue Tabelle erstellen, die von den Widgets verwendet werden kann. Dadurch wird Duplikation reduziert und die Leistung verbessert. Der Nachteil ist, dass Sie die neue Tabelle auf dem neuesten Stand halten müssen.

Abfrageergebnisse wiederverwenden

Es ist üblich, dass dieselbe Abfrage innerhalb eines kurzen Zeitraums mehrmals ausgeführt wird. Dies kann beispielsweise der Fall sein, wenn mehrere Personen dasselbe Daten-Dashboard öffnen. Wenn Sie eine Abfrage ausführen, können Sie Athena anweisen, zuvor berechnete Ergebnisse wiederzuverwenden. Sie geben das maximale Alter der Ergebnisse an, die wiederverwendet werden sollen. Wenn dieselbe Abfrage zuvor innerhalb dieses Zeitrahmens ausgeführt wurde, gibt Athena diese Ergebnisse zurück, anstatt die Abfrage erneut auszuführen. Weitere Informationen finden Sie unter Wiederverwenden von Abfrageergebnissen hier im Amazon-Athena-Benutzerhandbuch und im AWS -Big-Data-Blog Kosten reduzieren und die Abfrageleistung verbessern mit Amazon Athena Query Result Reuse.

Methoden zur Datenoptimierung

Die Leistung hängt nicht nur von Abfragen ab, sondern vor allem auch davon, wie Ihr Datensatz organisiert ist und welches Dateiformat und welche Komprimierung er verwendet.

Ihre Daten partitionieren

Durch die Partitionierung wird Ihre Tabelle in Teile aufgeteilt und die zugehörigen Daten werden auf der Grundlage von Eigenschaften wie Datum, Land oder Region zusammengefasst. Partitionsschlüssel fungieren als virtuelle Spalten. Sie definieren Partitionsschlüssel bei der Tabellenerstellung und verwenden sie zum Filtern Ihrer Abfragen. Wenn Sie nach Partitionsschlüsselspalten filtern, werden nur Daten aus übereinstimmenden Partitionen gelesen. Wenn Ihr Datensatz beispielsweise nach Datum partitioniert ist und Ihre Abfrage über einen Filter verfügt, der nur auf die letzte Woche zutrifft, werden nur die Daten der letzten Woche gelesen. Weitere Informationen zur Partitionierung finden Sie unter Daten in Athena partitionieren.

Wählen Sie Partitionsschlüssel aus, die Ihre Abfragen unterstützen

Da die Partitionierung erhebliche Auswirkungen auf die Abfrageleistung hat, sollten Sie beim Entwerfen Ihres Datensatzes und Ihrer Tabellen darauf achten, wie Sie partitionieren. Zu viele Partitionsschlüssel können zu fragmentierten Datensätzen mit zu vielen und zu kleinen Dateien führen. Umgekehrt führen zu wenige Partitionsschlüssel oder gar keine Partitionierung zu Abfragen, bei denen mehr Daten gescannt werden als nötig.

Vermeiden Sie die Optimierung für seltene Abfragen

Eine gute Strategie besteht darin, für die häufigsten Abfragen zu optimieren und die Optimierung für seltene Abfragen zu vermeiden. Wenn sich Ihre Abfragen beispielsweise auf Zeiträume von Tagen beziehen, sollten Sie sie nicht nach Stunden partitionieren, auch wenn einige Abfragen auf diese Ebene filtern. Wenn Ihre Daten über eine detaillierte Zeitstempelspalte verfügen, können die seltenen Abfragen, die nach Stunden filtern, die Zeitstempelspalte verwenden. Auch wenn in seltenen Fällen etwas mehr Daten als nötig gescannt werden, ist eine Reduzierung der Gesamtleistung in seltenen Fällen in der Regel kein guter Kompromiss.

Um die Datenmenge, die Abfragen scannen müssen, zu reduzieren und dadurch die Leistung zu verbessern, sollten Sie ein spaltenförmiges Dateiformat verwenden und die Datensätze sortiert halten. Anstatt nach Stunden zu partitionieren, sollten Sie die Datensätze nach Zeitstempel sortieren. Bei Abfragen mit kürzeren Zeitfenstern ist die Sortierung nach Zeitstempel fast so effizient wie die Partitionierung nach Stunden. Darüber hinaus beeinträchtigt die Sortierung nach Zeitstempeln in der Regel nicht die Leistung von Abfragen in Zeitfenstern, die in Tagen gezählt werden. Weitere Informationen finden Sie unter Spaltenorientierte Dateiformate verwenden.

Beachten Sie, dass Abfragen in Tabellen mit Zehntausenden von Partitionen besser abschneiden, wenn alle Partitionsschlüssel Prädikate enthalten. Dies ist ein weiterer Grund, Ihr

Partitionierungsschema für die häufigsten Abfragen zu entwerfen. Weitere Informationen finden Sie unter Partitionen nach Gleichheit abfragen.

Partitionsprojektion verwenden

Die Partitionsprojektion ist eine Athena-Funktion, die Partitionsinformationen nicht in der AWS Glue Data Catalog, sondern als Regeln in den Eigenschaften der Tabelle in AWS Glue speichert. Wenn Athena eine Abfrage für eine mit Partitionsprojektion konfigurierte Tabelle plant, liest es die Partitionsprojektionsregeln der Tabelle. Athena berechnet die Partitionen, die im Speicher gelesen werden sollen, basierend auf der Abfrage und den Regeln, anstatt nach Partitionen in der AWS Glue Data Catalog zu suchen.

Die Partitionsprojektion vereinfacht nicht nur die Partitionsverwaltung, sondern kann auch die Leistung von Datensätzen mit einer großen Anzahl von Partitionen verbessern. Wenn eine Abfrage Bereiche anstelle bestimmter Werte für Partitionsschlüssel enthält, dauert die Suche nach passenden Partitionen im Katalog umso länger, je mehr Partitionen vorhanden sind. Mit der Partitionsprojektion kann der Filter im Speicher berechnet werden, ohne den Katalog aufrufen zu müssen, und das kann viel schneller sein.

Unter bestimmten Umständen kann die Partitionsprojektion zu einer schlechteren Leistung führen. Ein Beispiel ist, wenn eine Tabelle „spärlich“ ist. Eine spärliche Tabelle enthält nicht Daten für jede Permutation der Partitionsschlüsselwerte, die in der Konfiguration der Partitionsprojektion beschrieben werden. Bei einer spärlichen Tabelle werden die anhand der Abfrage berechneten Partitionen und die Konfiguration der Partitionsprojektion alle in Amazon S3 aufgeführt, auch wenn sie keine Daten enthalten.

Wenn Sie die Partitionsprojektion verwenden, stellen Sie sicher, dass alle Partitionsschlüssel Prädikate enthalten. Grenzen Sie den Umfang der möglichen Werte ein, um unnötige Amazon-S3-Auflistungen zu vermeiden. Stellen Sie sich einen Partitionsschlüssel mit einem Bereich von einer Million Werten und eine Abfrage ohne Filter für diesen Partitionsschlüssel vor. Um die Abfrage auszuführen, muss Athena mindestens eine Million Amazon-S3-Listenoperationen ausführen. Abfragen sind am schnellsten, wenn Sie bestimmte Werte abfragen, unabhängig davon, ob Sie Partitionsprojektion verwenden oder Partitionsinformationen im Katalog speichern. Weitere Informationen finden Sie unter Partitionen nach Gleichheit abfragen.

Wenn Sie eine Tabelle für die Partitionsprojektion konfigurieren, stellen Sie sicher, dass die von Ihnen angegebenen Bereiche angemessen sind. Wenn eine Abfrage kein Prädikat für einen Partitionsschlüssel enthält, werden alle Werte im Bereich für diesen Schlüssel verwendet. Wenn Ihr Datensatz an einem bestimmten Datum erstellt wurde, verwenden Sie dieses Datum als

Ausgangspunkt für alle Datumsbereiche. Verwenden Sie `NOW` als das Ende von Datumsbereichen. Vermeiden Sie numerische Bereiche mit einer großen Anzahl von Werten und ziehen Sie in Betracht, stattdessen den injizierten Typ zu verwenden.

Weitere Informationen zur Partitionsprojektion finden Sie unter Partitionsprojektion mit Amazon Athena.

Verwenden Sie Partitionsindizes

Partitionsindizes sind eine Funktion in der AWS Glue Data Catalog , die Leistung bei der Partitionssuche für Tabellen mit einer großen Anzahl von Partitionen verbessert.

Die Liste der Partitionen im Katalog ist wie eine Tabelle in einer relationalen Datenbank. Die Tabelle enthält Spalten für die Partitionsschlüssel und eine zusätzliche Spalte für den Speicherort der Partition. Wenn Sie eine partitionierte Tabelle abfragen, werden die Partitionsspeicherorte durch Scannen dieser Tabelle gesucht.

Wie bei relationalen Datenbanken können Sie die Leistung von Abfragen erhöhen, indem Sie Indizes hinzufügen. Sie können mehrere Indizes hinzufügen, um unterschiedliche Abfragemuster zu unterstützen. Der AWS Glue Data Catalog Partitionsindex unterstützt sowohl Gleichheits- als auch Vergleichsoperatoren wie >>=,, in < Kombination mit dem `AND` Operator. Weitere Informationen finden Sie unter Arbeiten mit Partitionsindizes AWS Glue im AWS Glue Entwicklerhandbuch und Verbessern der Abfrageleistung von Amazon Athena mithilfe von AWS Glue Data Catalog Partitionsindizes im AWS Big Data-Blog.

Immer STRING als Typ für Partitionsschlüssel verwenden

Wenn Sie Partitionsschlüssel abfragen, denken Sie daran, dass Athena verlangt, dass die Partitionsschlüssel vom Typ `STRING` sind, um die Partitionsfilterung nach unten in AWS Glue zu schieben. Wenn die Anzahl der Partitionen nicht gering ist, kann die Verwendung anderer Typen zu einer schlechteren Leistung führen. Wenn Ihre Partitionsschlüsselwerte einem Datum oder einer Zahl ähneln, wandeln Sie sie in Ihrer Abfrage in den entsprechenden Typ um.

Alte und leere Partitionen entfernen

Wenn Sie Daten aus einer Partition auf Amazon S3 entfernen (z. B. mithilfe des Amazon-S3-Lebenszyklus), sollten Sie auch den Partitionseintrag aus dem AWS Glue Data Catalog entfernen. Während der Abfrageplanung wird jede Partition, der die Abfrage entspricht, in Amazon S3 aufgeführt. Wenn Sie viele leere Partitionen haben, kann sich der Mehraufwand beim Auflisten dieser Partitionen nachteilig auswirken.

Wenn Sie viele tausend Partitionen haben, sollten Sie außerdem in Erwägung ziehen, Partitionsmetadaten für alte Daten zu entfernen, die nicht mehr relevant sind. Wenn Abfragen beispielsweise nie Daten berücksichtigen, die älter als ein Jahr sind, können Sie in regelmäßigen Abständen Partitionsmetadaten für die älteren Partitionen entfernen. Wenn die Anzahl der Partitionen auf Zehntausende ansteigt, kann das Entfernen ungenutzter Partitionen Abfragen beschleunigen, die nicht für alle Partitionsschlüssel Prädikate enthalten. Hinweise zum Einbeziehen von Prädikaten für alle Partitionsschlüssel in Ihre Abfragen finden Sie unter Partitionen nach Gleichheit abfragen.

Partitionen nach Gleichheit abfragen

Abfragen, die Gleichheitsprädikate für alle Partitionsschlüssel enthalten, werden schneller ausgeführt, da die Partitionsmetadaten direkt geladen werden können. Vermeiden Sie Abfragen, bei denen einer oder mehrere Partitionsschlüssel kein Prädikat haben oder bei denen das Prädikat einen Wertebereich auswählt. Für solche Abfragen muss die Liste aller Partitionen gefiltert werden, um passende Werte zu finden. Bei den meisten Tabellen ist der Overhead minimal, bei Tabellen mit Zehntausenden oder mehr Partitionen kann der Overhead jedoch erheblich werden.

Wenn es nicht möglich ist, Ihre Abfragen so umzuschreiben, dass Partitionen nach Gleichheit gefiltert werden, können Sie es mit Partitionsprojektion versuchen. Weitere Informationen finden Sie unter Partitionsprojektion verwenden.

Vermeiden Sie es, MSCK REPAIR TABLE für die Partitionsverwaltung zu verwenden

Da die Ausführung von `MSCK REPAIR TABLE` sehr lange dauern kann, nur neue Partitionen hinzugefügt und alte Partitionen nicht entfernt werden, ist dies keine effiziente Methode zur Verwaltung von Partitionen (siehe Überlegungen und Einschränkungen).

Partitionen lassen sich besser manuell mithilfe der AWS Glue Data Catalog -APIs ALTER TABLE ADD PARTITION oder AWS Glue -Crawlers verwalten. Als Alternative können Sie die Partitionsprojektion verwenden, sodass Sie keine Partitionen mehr verwalten müssen. Weitere Informationen finden Sie unter Partitionsprojektion mit Amazon Athena.

Stellen Sie sicher, dass Ihre Abfragen mit dem Partitionierungsschema kompatibel sind

Mithilfe der Anweisung `EXPLAIN` können Sie im Voraus überprüfen, welche Partitionen eine Abfrage scannt. Stellen Sie Ihrer Abfrage das `EXPLAIN`-Schlüsselwort voran und suchen Sie dann am Ende der `EXPLAIN`-Ausgabe nach dem Quellfragment (z. B. `Fragment 2 [SOURCE]`) für jede Tabelle. Suchen Sie nach Zuweisungen, bei denen die rechte Seite als Partitionsschlüssel definiert ist. Die Zeile darunter enthält eine Liste aller Werte für diesen Partitionsschlüssel, die bei der Ausführung der Abfrage gescannt werden.

Angenommen, Sie haben eine Abfrage für eine Tabelle mit einem `dt`-Partitionsschlüssel und dem Präfix `EXPLAIN`. Wenn es sich bei den Werten in der Abfrage um Datumswerte handelt und ein Filter einen Bereich von drei Tagen auswählt, könnte die `EXPLAIN` Ausgabe etwa so aussehen:

```
dt := dt:string:PARTITION_KEY
    :: [[2023-06-11], [2023-06-12], [2023-06-13]]
```

Die `EXPLAIN`-Ausgabe zeigt, dass der Planer drei Werte für diesen Partitionsschlüssel gefunden hat, die der Abfrage entsprachen. Sie zeigt Ihnen auch, was diese Werte sind. Weitere Informationen zur Verwendung von `EXPLAIN` und Verwenden von EXPLAIN und EXPLAIN ANALYZE in Athena finden Sie unter Die Ergebnisse der Athena-EXPLAIN-Anweisung verstehen.

Spaltenorientierte Dateiformate verwenden

Spaltenförmige Dateiformate wie Parquet und ORC sind für verteilte Analyse-Workloads konzipiert. Sie organisieren Daten nach Spalten statt nach Zeilen. Das Organisieren von Daten im Spaltenformat bietet die folgenden Vorteile:

- Nur die für die Abfrage benötigten Spalten werden geladen
- Die Gesamtmenge der Daten, die geladen werden müssen, wird reduziert
- Spaltenwerte werden zusammen gespeichert, sodass Daten effizient komprimiert werden können
- Dateien können Metadaten enthalten, die es der Engine ermöglichen, das Laden nicht benötigter Daten zu überspringen

Als Beispiel dafür, wie Dateimetadaten verwendet werden können, können Dateimetadaten Informationen über die Mindest- und Höchstwerte auf einer Datenseite enthalten. Wenn die abgefragten Werte nicht in dem in den Metadaten angegebenen Bereich liegen, kann die Seite übersprungen werden.

Eine Möglichkeit, diese Metadaten zur Verbesserung der Leistung zu verwenden, besteht darin, sicherzustellen, dass die Daten in den Dateien sortiert sind. Angenommen, Sie haben Abfragen, die nach Datensätzen suchen, bei denen sich der `created_at`-Eintrag innerhalb einer kurzen Zeitspanne befindet. Wenn Ihre Daten nach der `created_at`-Spalte sortiert sind, kann Athena die Minimal- und Maximalwerte in den Dateimetadaten verwenden, um die nicht benötigten Teile der Datendateien zu überspringen.

Achten Sie bei der Verwendung von spaltenorientierten Dateiformaten darauf, dass Ihre Dateien nicht zu klein sind. Wie in Zu viele Dateien vermeiden erwähnt, verursachen Datensätze mit vielen kleinen

Dateien Leistungsprobleme. Dies gilt insbesondere für spaltenförmige Dateiformate. Bei kleinen Dateien überwiegt der Aufwand des spaltenförmigen Dateiformats die Vorteile.

Beachten Sie, dass Parquet und ORC intern nach Zeilengruppen (Parquet) und Streifen (ORC) organisiert sind. Die Standardgröße für Zeilengruppen beträgt 128 MB und für Streifen 64 MB. Wenn Sie viele Spalten haben, können Sie die Zeilengruppe und die Streifen-Größe erhöhen, um die Leistung zu verbessern. Es wird nicht empfohlen, die Zeilengruppen- oder Streifen-Größe auf weniger als die Standardwerte zu reduzieren.

Um andere Datenformate in Parquet oder ORC zu konvertieren, können Sie AWS Glue ETL oder Athena verwenden. Weitere Informationen zur Verwendung von Athena für ETL finden Sie unter Verwenden von CTAS und INSERT INTO für ETL und Datenanalyse.

Daten komprimieren

Athena unterstützt eine Vielzahl von Komprimierungsformaten. Das Abfragen komprimierter Daten ist schneller und auch günstiger, da Sie für die Anzahl der vor der Dekomprimierung gescannten Byte zahlen.

Das GZIP-Format bietet gute Komprimierungsraten und bietet eine breite Unterstützung für andere Tools und Services. Das Format zstd (Zstandard) ist ein neueres Komprimierungsformat mit einem ausgewogenen Verhältnis zwischen Leistung und Kompressionsrate.

Versuchen Sie beim Komprimieren von Textdateien wie JSON- und CSV-Daten, ein Gleichgewicht zwischen der Anzahl der Dateien und der Größe der Dateien zu erreichen. Bei den meisten Komprimierungsformaten muss der Leser Dateien von Anfang an lesen. Das bedeutet, dass komprimierte Textdateien im Allgemeinen nicht parallel verarbeitet werden können. Große unkomprimierte Dateien werden häufig zwischen Workern aufgeteilt, um eine höhere Parallelität bei der Abfrageverarbeitung zu erreichen. Dies ist jedoch bei den meisten Komprimierungsformaten nicht möglich.

Wie in Zu viele Dateien vermeiden diskutiert, ist es besser, weder zu viele noch zu wenige Dateien zu haben. Da die Anzahl der Dateien das Limit dafür ist, wie viele Worker die Abfrage verarbeiten können, gilt diese Regel insbesondere für komprimierte Dateien.

Weitere Informationen zur Verwendung der Komprimierung in Athena finden Sie unter Athena-Komprimierungs-Support.

Verwenden Sie Bucketing für die Suche nach Schlüsseln mit hoher Kardinalität

Beim Bucketing handelt es sich um eine Technik zur Verteilung von Datensätzen in separate Dateien, die auf dem Wert einer der Spalten basieren. Dadurch wird sichergestellt, dass sich alle Datensätze mit demselben Wert in derselben Datei befinden. Bucketing ist nützlich, wenn Sie einen Schlüssel mit hoher Kardinalität haben und viele Ihrer Abfragen nach bestimmten Werten des Schlüssels suchen.

Nehmen wir beispielsweise an, Sie fragen eine Reihe von Datensätzen für einen bestimmten Benutzer ab. Wenn die Daten nach Benutzer-IDs zusammengefasst sind, weiß Athena im Voraus, welche Dateien Datensätze für eine bestimmte ID enthalten und welche nicht. Dadurch kann Athena nur die Dateien lesen, die die ID enthalten können, wodurch die Menge der gelesenen Daten erheblich reduziert wird. Es reduziert auch die Rechenzeit, die andernfalls erforderlich wäre, um die Daten nach der spezifischen ID zu durchsuchen.

Nachteile von Bucketing

Bucketing ist weniger nützlich, wenn Abfragen häufig nach mehreren Werten in der Spalte suchen, nach der die Daten gruppiert sind. Je mehr Werte abgefragt werden, desto höher ist die Wahrscheinlichkeit, dass alle oder die meisten Dateien gelesen werden müssen. Wenn Sie beispielsweise drei Buckets haben und eine Abfrage nach drei verschiedenen Werten sucht, müssen möglicherweise alle Dateien gelesen werden. Bucketing funktioniert am besten, wenn Abfragen nach einzelnen Werten suchen.

Weitere Informationen finden Sie unter Partitionierung und Bucketing in Athena.

Zu viele Dateien vermeiden

Datensätze, die aus vielen kleinen Dateien bestehen, führen insgesamt zu einer schlechten Abfrageleistung. Wenn Athena eine Abfrage plant, listet es alle Partitionsspeicherorte auf, was einige Zeit in Anspruch nimmt. Die Bearbeitung und Anforderung jeder Datei ist ebenfalls mit einem Rechenaufwand verbunden. Daher ist das Laden einer einzigen größeren Datei aus Amazon S3 schneller als das Laden derselben Datensätze aus vielen kleineren Dateien.

In extremen Fällen können Sie auf Servicebeschränkungen in Amazon S3 stoßen. Amazon S3 unterstützt bis zu 5 500 Anfragen pro Sekunde an eine einzelne Indexpartition. Anfänglich wird ein Bucket als eine einzelne Indexpartition behandelt, aber wenn die Anforderungslast zunimmt, kann er in mehrere Indexpartitionen aufgeteilt werden.

Amazon S3 untersucht Anforderungsmuster und Aufteilungen auf der Grundlage von Schlüsselpräfixen. Wenn Ihr Datensatz aus vielen tausend Dateien besteht, können die Anfragen von Athena das Anforderungskontingent überschreiten. Selbst bei weniger Dateien kann das Kontingent

überschritten werden, wenn mehrere gleichzeitige Abfragen für denselben Datensatz durchgeführt werden. Andere Anwendungen, die auf dieselben Dateien zugreifen, können zur Gesamtzahl der Anfragen beitragen.

Wenn die Anforderungsrate `limit` überschritten wird, gibt Amazon S3 den folgenden Fehler zurück. Dieser Fehler ist in den Statusinformationen für die Abfrage in Athena enthalten.

SlowDown: Bitte reduzieren Sie Ihre Anfragerate

Stellen Sie zur Fehlerbehebung zunächst fest, ob der Fehler durch eine einzelne Abfrage oder durch mehrere Abfragen verursacht wurde, die dieselben Dateien lesen. Wenn letzteres der Fall ist, koordinieren Sie die Ausführung von Abfragen, sodass sie nicht gleichzeitig ausgeführt werden. Um dies zu erreichen, fügen Sie Ihrer Anwendung einen Warteschlangenmechanismus oder sogar Wiederholungsversuche hinzu.

Wenn das Ausführen einer einzelnen Abfrage den Fehler auslöst, versuchen Sie, Datendateien zu kombinieren oder die Abfrage so zu ändern, dass weniger Dateien gelesen werden. Kleine Dateien lassen sich am besten kombinieren, bevor sie geschrieben werden. Ziehen Sie dazu die folgenden Techniken in Betracht:

- Ändern Sie den Prozess, der die Dateien schreibt, um größere Dateien zu schreiben. Sie könnten beispielsweise Datensätze für einen längeren Zeitraum zwischenspeichern, bevor sie geschrieben werden.
- Speichern Sie Dateien an einem Speicherort auf Amazon S3 und verwenden Sie ein Tool wie Glue ETL, um sie zu größeren Dateien zu kombinieren. Verschieben Sie dann die größeren Dateien an den Speicherort, auf den die Tabelle verweist. Weitere Informationen finden Sie unter Lesen von Eingabedateien in größeren Gruppen im AWS Glue Entwicklerhandbuch und Wie kann ich einen AWS Glue ETL-Job für die Ausgabe größerer Dateien konfigurieren? im AWS re:POST Knowledge Center.
- Reduzieren Sie die Anzahl der Partitionsschlüssel. Wenn Sie zu viele Partitionsschlüssel haben, enthält jede Partition möglicherweise nur wenige Datensätze, was zu einer übermäßigen Anzahl kleiner Dateien führt. Hinweise zur Entscheidung, welche Partitionen erstellt werden sollen, finden Sie unter Wählen Sie Partitionsschlüssel aus, die Ihre Abfragen unterstützen.

Vermeiden Sie zusätzliche Speicherhierarchien außerhalb der Partition

Um den Aufwand bei der Planung von Abfragen zu vermeiden, speichern Sie Dateien in einer flachen Struktur an jedem Partitionsspeicherort. Verwenden Sie keine zusätzlichen Verzeichnishierarchien.

Wenn Athena eine Abfrage plant, listet es alle Dateien in allen Partitionen auf, denen die Abfrage entspricht. Obwohl Amazon S3 per se keine Verzeichnisse hat, ist es üblich, den `/`-Schrägstrich als Verzeichnistrennzeichen zu interpretieren. Wenn Athena Partitionsspeicherorte auflistet, listet es rekursiv alle gefundenen Verzeichnisse auf. Wenn Dateien innerhalb einer Partition in einer Hierarchie organisiert sind, kommt es zu mehreren Auflistungsrunden.

Wenn sich alle Dateien direkt am Speicherort der Partition befinden, muss meistens nur ein Listenvorgang ausgeführt werden. Allerdings sind mehrere sequentielle Listenoperationen erforderlich, wenn Sie mehr als 1 000 Dateien in einer Partition haben, da Amazon S3 nur 1 000 Objekte pro Listenvorgang zurückgibt. Mehr als 1 000 Dateien in einer Partition können auch zu anderen, schwerwiegenderen Leistungsproblemen führen. Weitere Informationen finden Sie unter Zu viele Dateien vermeiden.

SymlinkTextInputFormat nur bei Bedarf verwenden

Die Verwendung der SymlinkTextInputFormat-Technik kann eine Möglichkeit sein, Situationen zu umgehen, in denen die Dateien für eine Tabelle nicht übersichtlich in Partitionen organisiert sind. Beispielsweise können Symlinks nützlich sein, wenn sich alle Dateien im selben Präfix oder Dateien mit unterschiedlichen Schemas am selben Ort befinden.

Durch die Verwendung von Symlinks wird die Abfrageausführung jedoch um ein gewisses Maß an Indirektion erweitert. Diese Ebenen der Indirektion wirken sich auf die Gesamtleistung aus. Die Symlink-Dateien müssen gelesen und die Speicherorte, die sie definieren, müssen aufgelistet werden. Dadurch werden mehrere Roundtrips zu Amazon S3 hinzugefügt, die für herkömmliche Hive-Tabellen nicht erforderlich sind. Zusammenfassend lässt sich sagen, dass Sie `SymlinkTextInputFormat` nur verwenden sollten, wenn bessere Optionen wie das Reorganisieren von Dateien nicht verfügbar sind.

Weitere Ressourcen

Weitere Informationen zur Leistungsoptimierung in Athena finden Sie in den folgenden Ressourcen:

- Lesen Sie den AWS Big-Data-Blogbeitrag Die 10 besten Tipps zur Leistungsoptimierung für Amazon Athena
- Einen Artikel über die Verwendung von Prädikat-Pushdown zur Verbesserung der Leistung bei Verbundabfragen finden Sie unter Verbessern von Verbundabfragen mit Prädikat-Pushdown in Amazon Athena im AWS -Big-Data-Blog.

- Einen Artikel über die Leistungsoptimierungen in der Athena-Abfrage-Engine finden Sie im AWS Big Data-Blog unter Abfragen dreimal schneller ausführen und dabei bis zu 70% Kosten sparen mit der neuesten Amazon Athena Athena-Engine.
- Lesen Sie weitere Athena-Beiträge im AWS Big-Data-Blog
- Stellen Sie ein Frage auf AWS -re:Post mit Hilfe des Amazon-Athena-Tags
- Konsultieren Sie die Athena-Themen im AWS Knowledge Center
- Kontakt AWS Support (klicken Sie im AWS Management ConsoleSupport auf Support, Support Center)

Drosselung durch Amazon S3 verhindern

Bei der Drosselung wird die Geschwindigkeit begrenzt, mit der Sie einen Service, eine Anwendung oder ein System nutzen. In AWS können Sie die Drosselung verwenden, um eine übermäßige Nutzung des Amazon-S3-Service zu verhindern und die Verfügbarkeit und Reaktionsfähigkeit von Amazon S3 für alle Benutzer zu erhöhen. Da die Drosselung jedoch die Geschwindigkeit begrenzt, mit der die Daten zu oder von Amazon S3 übertragen werden können, ist es wichtig, zu verhindern, dass Ihre Interaktionen gedrosselt werden.

Drosselung auf Service-Ebene reduzieren

Um eine Drosselung von Amazon S3 auf Service-Ebene zu vermeiden, können Sie Ihre Nutzung überwachen und Ihre Servicekontingente anpassen oder Sie verwenden bestimmte Techniken wie Partitionierung. Im Folgenden sind einige der Bedingungen aufgeführt, die zu einer Drosselung führen können:

- Überschreitung der API-Anforderungslimits Ihres Kontos – Amazon S3 hat standardmäßige API-Anforderungslimits, die auf Kontotyp und -nutzung basieren. Wenn Sie die maximale Anzahl von Anfragen pro Sekunde für ein einzelnes Objekt überschreiten, werden Ihre Anfragen möglicherweise gedrosselt, um eine Überlastung des Amazon-S3-Service zu verhindern.
- Unzureichende Datenpartitionierung – Wenn Sie Ihre Daten nicht richtig partitionieren und große Datenmengen übertragen, kann Amazon S3 Ihre Anfragen drosseln. Weitere Informationen finden Sie im Abschnitt Partitionierung verwenden in diesem Dokument.
- Große Anzahl kleiner Objekte – Vermeiden Sie nach Möglichkeit eine große Anzahl kleiner Dateien. Amazon S3 hat ein Limit von 5 500 GET-Anfragen pro Sekunde pro partitioniertem Präfix, und Ihre Athena-Abfragen haben dasselbe Limit. Wenn Sie Millionen von kleinen Objekten in einer einzigen Abfrage scannen müssen, wird Ihre Abfrage wahrscheinlich von Amazon S3 gedrosselt werden.

Um übermäßiges Scannen zu vermeiden, verwenden Sie AWS Glue-ETL, um Ihre Dateien regelmäßig zu komprimieren oder die Tabelle zu partitionieren und Partitionsschlüsselfilter hinzuzufügen. Weitere Informationen finden Sie in den folgenden Ressourcen.

- Wie kann ich einen AWS Glue-ETL-Auftrag für die Ausgabe größerer Dateien konfigurieren? (AWS Knowledge Center)
- Eingabedateien in größeren Gruppen lesen (AWS Glue-Entwicklerhandbuch)

Ihre Tabellen optimieren

Die Strukturierung Ihrer Daten ist wichtig, wenn Sie auf Probleme mit der Drosselung stoßen. Obwohl Amazon S3 große Datenmengen verarbeiten kann, kommt es manchmal aufgrund der Art und Weise, wie die Daten strukturiert sind, zu einer Drosselung.

In den folgenden Abschnitten finden Sie einige Vorschläge zur Strukturierung Ihrer Daten in Amazon S3, um Drosselungsprobleme zu vermeiden.

Partitionierung verwenden

Sie können die Partitionierung verwenden, um die Drosselung zu reduzieren, indem Sie die Datenmenge einschränken, auf die zu einem bestimmten Zeitpunkt zugegriffen werden muss. Durch die Partitionierung von Daten in bestimmten Spalten können Sie Anfragen gleichmäßig auf mehrere Objekte verteilen und die Anzahl der Anfragen für ein einzelnes Objekt reduzieren. Durch die Reduzierung der Datenmenge, die gescannt werden muss, wird die Abfrageleistung verbessert und die Kosten gesenkt.

Sie können beim Erstellen einer Tabelle Partitionen definieren, die als virtuelle Spalten fungieren. Um eine Tabelle mit Partitionen in einer `CREATE TABLE`-Anweisung zu erstellen, verwenden Sie die `PARTITIONED BY (`*`column_name data_type`*`)`-Klausel, um die Schlüssel für die Partitionierung Ihrer Daten zu definieren.

Um die von einer Abfrage gescannten Partitionen einzuschränken, können Sie sie als Prädikate in einer `WHERE`-Klausel der Abfrage angeben. Daher eignen sich Spalten, die häufig als Filter verwendet werden, gut für die Partitionierung. In der Regel werden die Daten zeitbasiert partitioniert und das kann zu einem Multi-Level-Partitionierungsschema führen.

Beachten Sie, dass die Partitionierung auch mit Kosten verbunden ist. Wenn Sie die Anzahl der Partitionen in Ihrer Tabelle erhöhen, erhöht sich auch der Zeitaufwand für das Abrufen und Verarbeiten von Partitionsmetadaten. Durch eine zu starke Partitionierung können also die Vorteile zunichte gemacht werden, die Sie durch eine umsichtigere Partitionierung erzielen. Wenn Ihre Daten

stark auf einen Partitionswert ausgerichtet sind und die meisten Abfragen diesen Wert verwenden, kann Ihnen der zusätzliche Mehraufwand entstehen.

Weitere Informationen über Partitionierung in Athena finden Sie unter Was ist Partitionierung?

Ihre Daten in Buckets sammeln

Eine andere Möglichkeit, Ihre Daten zu partitionieren, besteht darin, die Daten in einer einzigen Partition zu sammeln. Beim Bucketing geben Sie eine oder mehrere Spalten an, die Zeilen enthalten, die Sie gruppieren möchten. Anschließend ordnen Sie diese Zeilen mehreren Gruppen zu. Auf diese Weise fragen Sie nur den Bucket ab, der gelesen werden muss, wodurch die Anzahl der Datenzeilen, die gescannt werden müssen, reduziert wird.

Wenn Sie eine Spalte auswählen, die für das Bucketing verwendet werden soll, wählen Sie die Spalte mit hoher Kardinalität (d. h. mit vielen unterschiedlichen Werten), die gleichmäßig verteilt ist und häufig zum Filtern der Daten verwendet wird. Ein Beispiel für eine Spalte, die sich gut für das Bucketing eignet, ist ein Primärschlüssel, z. B. eine ID-Spalte.

Weitere Informationen zu Bucketing in Athena finden Sie unter Was ist Bucketing?

Verwenden Sie AWS Glue-Partitionsindizes

Sie können AWS Glue-Partitionsindizes verwenden, um Daten in einer Tabelle auf der Grundlage der Werte einer oder mehrerer Partitionen zu organisieren. AWS Glue-Partitionsindizes können die Anzahl der Datenübertragungen, den Umfang der Datenverarbeitung und die Verarbeitungszeit von Abfragen reduzieren.

Ein AWS Glue-Partitionsindex ist eine Metadatendatei, die Informationen über die Partitionen in der Tabelle enthält, einschließlich der Partitionsschlüssel und ihrer Werte. Der Partitionsindex wird in einem Amazon-S3-Bucket gespeichert und automatisch von AWS Glue aktualisiert, wenn der Tabelle neue Partitionen hinzugefügt werden.

Wenn ein AWS Glue-Partitionsindex vorhanden ist, versuchen Abfragen, eine Teilmenge der Partitionen abzurufen, anstatt alle Partitionen in der Tabelle zu laden. Abfragen werden nur für die Teilmenge der Daten ausgeführt, die für die Abfrage relevant sind.

Wenn Sie eine Tabelle in AWS Glue erstellen, können Sie einen Partitionsindex für eine beliebige Kombination von Partitionsschlüsseln erstellen, die in der Tabelle definiert sind. Nachdem Sie einen oder mehrere Partitionsindizes für eine Tabelle erstellt haben, müssen Sie der Tabelle eine Eigenschaft hinzufügen, die die Partitionsfilterung ermöglicht. Anschließend können Sie die Tabelle von Athena abfragen.

Schritte zum Erstellen eines Partitionsindex in AWS Glue finden Sie unter Arbeiten mit Partitionsindizes in AWS Glue im AWS Glue-Entwicklerhandbuch. Hinweise zum Hinzufügen einer Tabelleneigenschaft zur Aktivierung der Partitionsfilterung finden Sie unter AWS Glue Indizierung und Filterung von Partitionen.

Datenkomprimierung und Dateiaufteilung verwenden

Datenkomprimierung kann Abfragen erheblich beschleunigen, wenn Dateien ihre optimale Größe haben oder wenn sie in logische Gruppen aufgeteilt werden können. Im Allgemeinen erfordern höhere Komprimierungsraten mehr CPU-Zyklen, um die Daten zu komprimieren und zu dekomprimieren. Für Athena empfehlen wir, entweder Apache Parquet oder Apache ORC zu verwenden, die Daten standardmäßig komprimieren. Weitere Informationen zur Datenkomprimierung in Athena finden Sie unter Athena-Komprimierungs-Support.

Das Aufteilen von Dateien erhöht die Parallelität, da Athena das Lesen einer einzelnen Datei auf mehrere Leser verteilen kann. Wenn eine einzelne Datei nicht aufgeteilt werden kann, kann nur ein einziger Leser die Datei lesen, während andere Leser inaktiv sind. Apache Parquet und Apache ORC unterstützen auch teilbare Dateien.

Verwenden Sie optimierte spaltenförmige Datenspeicher

Die Athena-Abfrageleistung verbessert sich erheblich, wenn Sie Ihre Daten in ein Spaltenformat konvertieren. Wenn Sie spaltenförmige Dateien generieren, ist eine Optimierungstechnik, die Sie in Betracht ziehen sollten, darin, die Daten auf der Grundlage des Partitionsschlüssels zu ordnen.

Apache Parquet und Apache ORC sind häufig verwendete spaltenorientierte Open-Source-Datenspeicher. Informationen zur Konvertierung vorhandener Amazon-S3-Datenquellen in eines dieser Formate finden Sie unter Konvertieren in spaltenbasierte Formate.

Verwenden Sie eine größere Parquet-Blockgröße oder ORC-Streifengröße

Parquet und ORC verfügen über Datenspeicherparameter, die Sie zur Optimierung anpassen können. In Parquet können Sie die Blockgröße optimieren. In ORC können Sie die Streifengröße optimieren. Je größer der Block oder der Streifen, desto mehr Zeilen können Sie in jedem Block oder Streifen speichern. Standardmäßig beträgt die Parquet-Blockgröße 128 MB und die ORC-Streifen-Größe 64 MB.

Wenn ein ORC-Streifen weniger als 8 MB groß ist (der Standardwert von `hive.orc.max_buffer_size`), liest Athena den gesamten ORC-Streifen. Dies ist der

Kompromiss, den Athena zwischen Spaltenselektivität und Eingabe-/Ausgabeoperationen pro Sekunde für kleinere Streifen eingeht.

Wenn Sie Tabellen mit einer sehr großen Anzahl von Spalten haben, kann eine kleine Block- oder Streifen-Größe dazu führen, dass mehr Daten gescannt werden als nötig. In diesen Fällen kann eine größere Blockgröße effizienter sein.

Verwenden Sie ORC für komplexe Typen

Wenn Sie derzeit in Parquet gespeicherte Spalten mit komplexen Datentypen (z. B. `array`, `map` oder `struct`) abfragen, liest Athena eine gesamte Datenzeile, anstatt nur die angegebenen Spalten selektiv zu lesen. Dies ist ein bekanntes Problem in Athena. Um dieses Problem zu umgehen, sollten Sie ORC verwenden.

Wählen Sie einen Komprimierungsalgorithmus

Ein weiterer Parameter, den Sie konfigurieren können, ist der Komprimierungsalgorithmus für Datenblöcke. Informationen zu den in Athena für Parquet und ORC unterstützten Komprimierungsalgorithmen finden Sie unter Athena-Komprimierungsunterstützung.

Weitere Informationen zur Optimierung von spaltenförmigen Speicherformaten in Athena finden Sie im Abschnitt „Optimieren Sie die Generierung von spaltenförmigen Datenspeichern“ im AWS Big-Data-Blogbeitrag Die 10 besten Tipps zur Leistungsoptimierung für Amazon Athena.

Verwenden von Iceberg-Tabellen

Apache Iceberg ist ein offenes Tabellenformat für sehr große Analysedatensätze, das für eine optimierte Nutzung auf Amazon S3 konzipiert ist. Sie können Iceberg-Tabellen verwenden, um die Drosselung in Amazon S3 zu reduzieren.

Iceberg-Tabellen bieten Ihnen folgende Vorteile:

- Sie können Iceberg-Tabellen auf eine oder mehrere Spalten partitionieren. Dadurch wird der Datenzugriff optimiert und die Datenmenge reduziert, die durch Abfragen gescannt werden muss.
- Da der Iceberg-Objektspeichermodus die Iceberg-Tabellen für die Verwendung mit Amazon S3 optimiert, kann er große Datenmengen und umfangreiche Abfrage-Workloads verarbeiten.
- Iceberg-Tabellen im Objektspeichermodus sind skalierbar, fehlertolerant und robust, was dazu beitragen kann, die Drosselung zu reduzieren.
- Die ACID-Transaktionsunterstützung bedeutet, dass mehrere Benutzer Amazon-S3-Objekte auf atomare Weise hinzufügen und löschen können.

Weitere Informationen zu Apache Iceberg finden Sie unter Apache Iceberg. Weitere Informationen zur Verwendung von Apache-Iceberg-Tabellen in Athena finden Sie unter Verwendung von Iceberg-Tabellen.

Abfragen optimieren

Verwenden Sie die Vorschläge in diesem Abschnitt für die Optimierung Ihrer SQL-Abfragen in Athena.

Verwenden Sie LIMIT mit der ORDER-BY-Klausel

Die `ORDER BY`-Klausel gibt Daten in einer sortierten Reihenfolge zurück. Dazu muss Athena alle Datenzeilen an einen einzelnen Worker-Knoten senden und die Zeilen dann sortieren. Diese Art von Abfrage kann lange laufen oder sogar fehlschlagen.

Um die Effizienz Ihrer Abfragen zu erhöhen, sollten Sie sich die oberen oder unteren *N*-Werte ansehen und dann auch eine `LIMIT`-Klausel verwenden. Dadurch werden die Kosten für die Sortierung erheblich reduziert, da sowohl die Sortierung als auch die Beschränkung auf einzelne Worker-Knoten und nicht auf einen einzelnen Worker verlagert werden.

JOIN-Klauseln optimieren

Wenn Sie zwei Tabellen verbinden, verteilt Athena die Tabelle auf der rechten Seite an die Worker-Knoten und streamt dann die Tabelle auf der linken Seite, um den Join durchzuführen.

Geben Sie aus diesem Grund die größere Tabelle auf der linken Seite des Joins und die kleinere Tabelle auf der rechten Seite des Joins an. Auf diese Weise verwendet Athena weniger Speicher und führt die Abfrage mit geringerer Latenz aus.

Beachten Sie auch folgende Punkte:

- Wenn Sie mehrere `JOIN`-Befehle verwenden, geben Sie die Tabellen vom größten zum kleinsten an.
- Vermeiden Sie Kreuzverknüpfungen, sofern sie nicht für die Abfrage erforderlich sind.

Optimieren Sie GROUP-BY-Klauseln

Der `GROUP BY`-Operator verteilt Zeilen auf der Grundlage der `GROUP BY`-Spalten an die Worker-Knoten. Auf diese Spalten wird im Speicher verwiesen, und die Werte werden verglichen, während

die Zeilen aufgenommen werden. Die Werte werden zusammen aggregiert, wenn die `GROUP BY`-Spalte übereinstimmt. Angesichts der Funktionsweise dieses Verfahrens ist es ratsam, die Spalten von der höchsten zur niedrigsten Kardinalität anzuordnen.

Verwenden Sie Zahlen statt Zeichenketten

Zahlen benötigen weniger Speicher und sind im Vergleich zu Zeichenketten schneller zu verarbeiten. Verwenden Sie nach Möglichkeit Zahlen anstelle von Zeichenketten.

Die Gesamtanzahl der Spalten reduzieren

Um den Gesamtspeicherbedarf zum Speichern Ihrer Daten zu reduzieren, begrenzen Sie die Anzahl der in Ihrer `SELECT`-Anweisung angegebenen Spalten.

Verwenden Sie reguläre Ausdrücke anstelle von LIKE

Abfragen, die Klauseln enthalten, z. B. `LIKE '%string%'` bei großen Zeichenketten, können sehr rechenintensiv sein. Wenn Sie in einer Zeichenkettenspalte nach mehreren Werten filtern, verwenden Sie stattdessen die Funktion regexp_like() und einen regulären Ausdruck. Dies ist besonders nützlich, wenn Sie eine lange Werteliste vergleichen.

Die LIMIT-Klausel verwenden

Anstatt beim Ausführen einer Abfrage alle Spalten auszuwählen, verwenden Sie die `LIMIT`-Klausel, um nur die Spalten zurückzugeben, die Sie benötigen. Dadurch wird die Größe des Datensatzes reduziert, der über die Pipeline zur Abfrageausführung verarbeitet wird. `LIMIT`-Klauseln sind hilfreicher, wenn Sie Tabellen mit einer großen Anzahl von Spalten abfragen, die auf Zeichenfolgen basieren. `LIMIT`-Klauseln sind auch hilfreich, wenn Sie mehrere Verknüpfungen oder Aggregationen für eine Abfrage durchführen.

Weitere Informationen finden Sie auch unter

Bewährte Methoden für Designmuster: Optimieren der Leistung von Amazon S3 finden Sie im Benutzerhandbuch für Amazon Simple Storage Service.

Leistungsoptimierung in Athena

Athena-Komprimierungs-Support

Themen

Athena unterstützt eine Vielzahl von Komprimierungsformate zum Lesen und Schreiben von Daten, einschließlich des Lesens aus einer Tabelle, die mehrere Komprimierungsformate verwendet. Zum Beispiel kann Athena die Daten in einer Tabelle erfolgreich lesen, die das Parquet-Dateiformat verwendet, wenn einige Parquet-Dateien mit Snappy komprimiert werden und andere Parquet-Dateien mit GZIP komprimiert werden. Das gleiche Prinzip gilt für ORC-, Textfile- und JSON-Speicherformate.

Athena unterstützt folgende Komprimierungsformate:

- BZIP2 – Format, das den Burrows-Wheeler-Algorithmus verwendet.
- ENTLEEREN (Deflate) – Komprimierungsalgorithmus basierend auf LZSS und Huffman-Kodierung. Entleeren (Deflate) ist nur für das Avro-Dateiformat relevant.
- GZIP – Komprimierungsalgorithmus basierend auf Entleeren (Deflate). Für Hive-Tabellen in Athena-Engine-Version 2 und 3 und Iceberg-Tabellen in Athena-Engine-Version 2 ist GZIP das standardmäßige Schreibkomprimierungsformat für Dateien in den Parquet- und Textdateispeicherformaten. Dateien im `tar.gz`-Format werden nicht unterstützt.
- LZ4 – Dieses Mitglied der Lempel-Ziv 77 (LZ7)-Familie konzentriert sich auch auf die Komprimierungs- und Dekomprimierungsgeschwindigkeit und nicht auf die maximale Komprimierung von Daten. LZ4 hat die folgenden Framing-Formate:
  - LZ4 Roh/ungerahmt – Eine ungerahmte Standardimplementierung des LZ4-Blockkomprimierungsformats. Weitere Informationen finden Sie in der Beschreibung des LZ4-Blockformats auf GitHub.
  - LZ4 framed – Die übliche Framing-Implementierung von LZ4. Weitere Informationen finden Sie in der Beschreibung des LZ4-Frame-Formats auf GitHub.
  - LZ4 hadoop-kompatibel – Die Apache-Hadoop-Implementierung von LZ4. Diese Implementierung umschließt die LZ4-Komprimierung mit der Klasse BlockCompressorStream.java.

- LZOO – Format, das den Lempel-Ziv-Oberhumer-Algorithmus verwendet, der sich eher auf eine hohe Komprimierungs- und Dekompressionsgeschwindigkeit als auf die maximale Komprimierung von Daten konzentriert. LZO hat zwei Implementierungen:
  - LZO-Standard – Weitere Informationen hierzu finden Sie im LZO-Abstract auf der Oberhumer-Website.
  - LZO hadoop-kompatibel – Diese Implementierung umschließt den LZO-Algorithmus mit der BlockCompressorStream.java-Klasse.
- SNAPPY – Komprimierungsalgorithmus, der Teil der Lempel-Ziv-77(LZ7)-Familie ist. Snappy konzentriert sich eher auf eine hohe Komprimierungs- und Dekomprimierungsgeschwindigkeit als auf die maximale Komprimierung von Daten.
- ZLIB – ZLIB basiert auf Deflate das Standard-Schreibkomprimierungsformat für Dateien im ORC-Datenspeicherformat. Weitere Informationen finden Sie auf der zlib-Seite auf GitHub.
- ZSTD – Der Zstandard-Echtzeit-Datenkomprimierungsalgorithmus ist ein schneller Komprimierungsalgorithmus, der hohe Komprimierungsverhältnisse bietet. Die Zstandard-Bibliothek (ZSTD) wird als Open-Source-Software unter Verwendung einer BSD-Lizenz bereitgestellt. ZSTD ist die Standardkomprimierung für Iceberg-Tabellen. Beim Schreiben komprimierter ZSTD-Daten verwendet Athena standardmäßig ZSTD-Komprimierungsstufe 3. Weitere Informationen zur Verwendung von ZSTD-Komprimierungsstufen in Athena finden Sie unter Verwendung von ZSTD-Komprimierungsstufen in Athena.

Festlegen von Komprimierungsformaten

Wenn Sie CREATE TABLE- oder CTAS-Anweisungen schreiben, können Sie Komprimierungseigenschaften angeben, die den Komprimierungstyp angeben, der verwendet werden soll, wenn Athena in diese Tabellen schreibt.

- Für CTAS siehe CTAS-Tabelleneigenschaften. Beispiele finden Sie unter Beispiele für CTAS-Abfragen.
- Für Informationen zu CREATE TABLE finden Sie unter ALTER TABLE SET TBLPROPERTIES eine Liste der Eigenschaften von Komprimierungstabellen.

Festlegen keiner Komprimierung

CREATE-TABLE-Anweisungen unterstützen das Schreiben unkomprimierter Dateien. Verwenden Sie die folgende Syntax, um unkomprimierte Dateien zu schreiben:

- CREATE TABLE (Textdatei oder JSON) — In `TBLPROPERTIES`, spezifizieren `write.compression = NONE`.
- CREATE TABLE (Parquet) — In `TBLPROPERTIES`, spezifizieren `parquet.compression = UNCOMPRESSED`.
- CREATE TABLE (ORC) — In `TBLPROPERTIES`, spezifizieren `orc.compress = NONE`.

Notizen und Ressourcen

- Derzeit werden Großbuchstaben-Dateierweiterungen wie `.GZ` oder `.BZIP2` von Athena nicht anerkannt. Vermeiden Sie die Verwendung von Datensätzen mit Dateierweiterungen in Großbuchstaben oder benennen Sie die Datendateierweiterungen in Kleinbuchstaben um.
- Athena bestimmt den Komprimierungstyp für Daten in den Formaten CSV, TSV und JSON anhand der Dateierweiterung. Wenn keine Dateierweiterung vorhanden ist, behandelt Athena die Daten als unkomprimierten Klartext. Wenn Ihre Daten komprimiert sind, stellen Sie sicher, dass der Dateiname die Komprimierungserweiterung enthält, z. B. `gz`.
- Das ZIP-Dateiformat wird nicht unterstützt.
- Für die Abfrage von Amazon-Data-Firehose-Protokollen von Athena umfassen unterstützte Formate GZIP-Komprimierung oder ORC-Dateien mit SNAPPY-Komprimierung.
- Weitere Informationen zur Verwendung der Komprimierung finden Sie in Abschnitt 3 („Komprimieren und Aufteilen von Dateien“) des AWS -Big-Data-Blogbeitrags Top 10 Tipps zur Leistungsoptimierung für Amazon Athena.

Unterstützung der Komprimierung von Hive-Tabellen nach Dateiformaten

Die Unterstützung der Hive-Komprimierung in Athena hängt von der Engine-Version ab.

Unterstützung der Hive-Komprimierung in Athena-Engine-Version 3

Die folgende Tabelle gibt einen Überblick über die Unterstützung von Komprimierungsformaten in der Athena-Engine-Version 3 für jedes Speicherdateiformat in Apache Hive. Das Textdateiformat umfasst TSV, CSV, JSON und benutzerdefinierte SerDes für Text. „Ja“ oder „Nein“ in einer Zelle gelten gleichermaßen für Lese- und Schreibvorgänge, sofern nicht anders angegeben. Für die Zwecke dieser Tabelle gelten CREATE TABLE, CTAS und INSERT INTO als Schreibvorgänge. Weitere Informationen zur Verwendung von ZSTD-Komprimierungsstufen in Athena finden Sie unter Verwendung von ZSTD-Komprimierungsstufen in Athena.

	Avro	Ion	ORC	Parquet	Textdatei
BZIP2	Ja	Ja	Nein	Nein	Ja
DEFLATE	Ja	Nein	Nein	Nein	Nein
GZIP	Nein	Ja	Nein	Ja	Ja
LZ4	Nein	Ja	Ja	Ja	Ja
LZO	Nein	Schreiben – Nein Lesen – Ja	Nein	Ja	Schreiben – Nein Lesen – Ja
SNAPPY	Ja	Ja	Ja	Ja	Ja
ZLIB	Nein	Nein	Ja	Nein	Nein
ZSTD	Ja	Ja	Ja	Ja	Ja
NONE	Ja	Ja	Ja	Ja	Ja

Unterstützung der Hive-Komprimierung in Athena-Engine-Version 2

In der folgenden Tabelle sind die unterstützten Komprimierungsformate in der Athena-Engine-Version 2 für Apache Hive zusammengefasst. Das Textdateiformat umfasst TSV, CSV, JSON und benutzerdefinierte SerDes für Text. „Ja“ oder „Nein“ in einer Zelle gelten gleichermaßen für Lese- und Schreibvorgänge, sofern nicht anders angegeben. Für die Zwecke dieser Tabelle gelten CREATE TABLE, CTAS und INSERT INTO als Schreibvorgänge.

	Avro	Ion	ORC	Parquet	Textdatei
BZIP2	Ja	Ja	Nein	Nein	Ja
DEFLATE	Ja	Nein	Nein	Nein	Nein
GZIP	Nein	Ja	Nein	Ja	Ja

	Avro	Ion	ORC	Parquet	Textdatei
LZ4	Nein	Nein	Ja	Schreiben – Ja Lesen – Nein	Schreiben – Nein Lesen – Ja
LZO	Nein	Schreiben – Nein Lesen – Ja	Nein	Ja	Schreiben – Nein Lesen – Ja
SNAPPY	Ja	Ja	Ja	Ja	Ja
ZLIB	Nein	Nein	Ja	Nein	Nein
ZSTD	Nein	Ja	Ja	Ja	Ja
NONE	Ja	Ja	Ja	Ja	Ja

Unterstützung der Komprimierung von Iceberg-Tabellen nach Dateiformaten

Die Unterstützung der Apache-Iceberg-Komprimierung in Athena hängt von der Engine-Version ab.

Unterstützung der Iceberg-Komprimierung in Athena-Engine-Version 3

Die folgende Tabelle fasst die Unterstützung des Komprimierungsformats in Athena-Engine-Version 3 für Speicherdateiformate in Apache Iceberg zusammen. „Ja“ oder „Nein“ in einer Zelle gelten gleichermaßen für Lese- und Schreibvorgänge, sofern nicht anders angegeben. Für die Zwecke dieser Tabelle gelten CREATE TABLE, CTAS und INSERT INTO als Schreibvorgänge. Das Standardspeicherformat für Iceberg in Athena-Engine-Version 3 ist Parquet. Das Standardkompressionsformat für Iceberg in Athena-Engine-Version 3 ist ZSTD. Weitere Informationen zur Verwendung von ZSTD-Komprimierungsstufen in Athena finden Sie unter Verwendung von ZSTD-Komprimierungsstufen in Athena.

	Avro	ORC	Parquet (Standard)
BZIP2	Nein	Nein	Nein
GZIP	Ja	Nein	Ja
LZ4	Nein	Ja	Nein
SNAPPY	Ja	Ja	Ja
ZLIB	Nein	Ja	Nein
ZSTD	Ja	Ja	Ja (Standard)
NONE	Ja (`None` oder `Deflate` angeben)	Ja	Ja (`None` oder `Uncompressed` angeben)

Unterstützung der Iceberg-Komprimierung in Athena-Engine-Version 2

Die folgende Tabelle gibt einen Überblick über die Unterstützung von Komprimierungsformaten in Athena-Engine-Version 2 für Apache Iceberg. „Ja“ oder „Nein“ in einer Zelle gelten gleichermaßen für Lese- und Schreibvorgänge, sofern nicht anders angegeben. Für die Zwecke dieser Tabelle gelten CREATE TABLE, CTAS und INSERT INTO als Schreibvorgänge. Das Standardspeicherformat für Iceberg in Athena-Engine-Version 2 ist Parquet. Das Standardkompressionsformat für Iceberg in Athena-Engine-Version 2 ist GZIP.

	Avro (Nicht unterstützt)	ORC (Nicht unterstützt)	Parquet (Standard)
BZIP2	Nein	Nein	Nein
GZIP	Nein	Nein	Ja (Standard)
LZ4	Nein	Nein	Nein
SNAPPY	Nein	Nein	Ja
ZLIB	Nein	Nein	Nein

	Avro (Nicht unterstützt)	ORC (Nicht unterstützt)	Parquet (Standard)
ZSTD	Nein	Nein	Ja
NONE	Nein	Nein	Ja

Verwendung von ZSTD-Komprimierungsstufen in Athena

Der Zstandard-Algorithmus zur Datenkomprimierung in Echtzeit ist ein schneller Kompressionsalgorithmus, der hohe Komprimierungsraten bereitstellt. Die Zstandard (ZSTD)-Bibliothek ist Open-Source-Software und verwendet eine BSD-Lizenz. Athena unterstützt das Lesen und Schreiben von ZSTD-komprimierten ORC-, Parquet- und Textdatei-Daten.

Sie können die ZSTD-Komprimierungsstufen verwenden, um das Komprimierungsverhältnis und die Geschwindigkeit entsprechend Ihren Anforderungen anzupassen. Die ZSTD-Bibliothek unterstützt Komprimierungsstufen von 1 bis 22. Athena verwendet standardmäßig die ZSTD-Komprimierungsstufe 3.

Komprimierungsstufen stellen differenzierte Kompromisse zwischen der Komprimierungsgeschwindigkeit und dem erreichten Komprimierungsgrad bereit. Niedrigere Komprimierungsstufen bieten eine höhere Geschwindigkeit aber größere Dateigrößen. Sie können beispielsweise Stufe 1 verwenden, wenn Geschwindigkeit am wichtigsten ist und Stufe 22, wenn Größe am wichtigsten ist. Stufe 3 eignet sich für viele Anwendungsfälle und ist die Standardeinstellung. Verwenden Sie Stufen über 19 mit Vorsicht, da sie mehr Arbeitsspeicher benötigen. Die ZSTD-Bibliothek bietet auch negative Komprimierungsstufen, die den Bereich der Komprimierungsgeschwindigkeiten und -verhältnisse erweitern. Weitere Informationen finden Sie im Zstandard-Komprimierungs-RFC.

Die Fülle an Komprimierungsstufen bietet erhebliche Möglichkeiten zur Feinabstimmung. Stellen Sie jedoch sicher, dass Sie Ihre Daten messen und die Kompromisse berücksichtigen, wenn Sie sich für eine Komprimierungsstufe entscheiden. Wir empfehlen, die Standardstufe 3 oder eine Stufe im Bereich von 6 bis 9 zu verwenden für einen angemessenen Kompromiss zwischen Komprimierungsgeschwindigkeit und komprimierter Datengröße zu erzielen. Reservieren Sie die Stufen 20 und höher für Fälle, in denen die Größe am wichtigsten ist und die Kompressionsgeschwindigkeit kein Problem darstellt.

Überlegungen und Einschränkungen

Berücksichtigen Sie bei der Verwendung der ZSTD-Komprimierungsstufe in Athena die folgenden Punkte.

- Die ZSTD-Eigenschaft `compression_level` wird nur in der Athena-Engine-Version 3 unterstützt.
- Die ZSTD-Eigenschaft `compression_level` wird für die `ALTER TABLE`-, `CREATE TABLE`-, `CREATE TABLE AS` (CTAS)- und `UNLOAD`-Anweisungen unterstützt.
- Die `compression_level`-Eigenschaft ist optional.
- Die `compression_level`-Eigenschaft wird nur für ZSTD-Komprimierung unterstützt.
- Mögliche Komprimierungsstufen sind 1 bis 22.
- Die Standardkomprimierungsstufe ist 3.

Informationen zur Unterstützung für Apache-Hive-ZSTD-Komprimierung in Athena finden Sie unter Unterstützung der Komprimierung von Hive-Tabellen nach Dateiformaten. Informationen zur Verwendung der Apache-Iceberg-ZSTD-Komprimierung in Athena finden Sie unter Unterstützung der Komprimierung von Iceberg-Tabellen nach Dateiformaten.

Festlegen der ZSTD-Komprimierungsstufen

Um die ZSTD-Komprimierungsstufe für die `ALTER TABLE`-, `CREATE TABLE`-, `CREATE TABLE AS`-, and `UNLOAD`-Anweisungen anzugeben, verwenden Sie die `compression_level`-Eigenschaft. Um die ZSTD-Komprimierung selbst anzugeben, müssen Sie die individuelle Komprimierungseigenschaft verwenden, die in der Syntax für die Anweisung verwendet wird.

ALTER TABLE SET TBLPROPERTIES

Geben Sie in der ALTER TABLE SET TBLPROPERTIES-Anweisung der `SET TBLPROPERTIES`-Klausel die ZSTD-Komprimierung mit `'write.compression' = 'ZSTD'` oder `'parquet.compression' = 'ZSTD'` an. Verwenden Sie anschließend die `compression_level`-Eigenschaft, um einen Wert zwischen 1 und 22 anzugeben (z. B. `compression_level' = 5`). Wenn Sie keine Eigenschaft für die Komprimierungsstufe angeben, wird die Komprimierungsstufe standardmäßig auf 3 gesetzt.

Beispiel

Im folgenden Beispiel wird die Tabelle `existing_table` so geändert, dass das Parquet-Dateiformat mit ZSTD-Komprimierung und ZSTD-Komprimierungsstufe 4 verwendet wird. Beachten Sie, dass der Wert für die Komprimierungsstufe als Zeichenfolge und nicht als Ganzzahl eingegeben werden muss.

```
ALTER TABLE existing_table
SET TBLPROPERTIES ('parquet.compression' = 'ZSTD', 'compression_level' = 4)
```

CREATE TABLE

Geben Sie in der CREATE TABLE-Anweisung der TBLPROPERTIES-Klausel `write.compression' = 'ZSTD'` oder `'parquet.compression' = 'ZSTD'` an, verwenden Sie anschließend `compression_level = compression_level` und geben Sie einen Wert zwischen 1 und 22 an. Wenn die `compression_level`-Eigenschaft nicht angegeben ist, ist die Standardkomprimierungsstufe 3.

Beispiel

Im folgenden Beispiel wird eine Tabelle im Parquet-Dateiformat mit ZSTD-Komprimierung und ZSTD-Komprimierungsstufe 4 erstellt.

```
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)

Geben Sie in der CREATE TABLE AS-Anweisung der `WITH`-Klausel `write_compression = 'ZSTD'` oder `parquet_compression = 'ZSTD'` an, verwenden Sie anschließend `compression_level = compression_level` und geben Sie einen Wert zwischen 1 und 22 an. Wenn die `compression_level`-Eigenschaft nicht angegeben ist, ist die Standardkomprimierungsstufe 3.

Beispiel

Das folgende CTAS-Beispiel gibt Parquet als Dateiformat an, das die ZSTD-Komprimierung mit der Komprimierungsstufe 4 verwendet.

```
CREATE TABLE new_table
WITH ( format = 'PARQUET', write_compression = 'ZSTD', compression_level = 4)
AS SELECT * FROM old_table
```

UNLOAD

Geben Sie in der UNLOAD-Anweisung der `WITH`-Klausel `compression = 'ZSTD'` an, verwenden Sie anschließend `compression_level = compression_level` und geben Sie einen Wert zwischen 1 und 22 an. Wenn die `compression_level`-Eigenschaft nicht angegeben ist, ist die Standardkomprimierungsstufe 3.

Beispiel

Im folgenden Beispiel werden die Abfrageergebnisse unter Verwendung des Parquet-Dateiformats, der ZSTD-Komprimierung und der ZSTD-Komprimierungsstufe 4 an den angegebenen Speicherort entladen.

```
UNLOAD (SELECT * FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/'
WITH (format = 'PARQUET', compression = 'ZSTD', compression_level = 4)
```

Markieren von Athena-Ressourcen

Ein Tag besteht aus einem Schlüssel und einem Wert, die Sie beide selbst definieren können. Wenn Sie eine Athena-Ressource mit Tags markieren, weisen Sie ihr benutzerdefinierte Metadaten zu. Mit Tags können Sie AWS-Ressourcen auf unterschiedliche Weise kategorisieren (z. B. nach Zweck, Eigentümer oder Umgebung). In Athena sind Ressourcen wie Arbeitsgruppen, Datenkataloge und Kapazitätsreservierungen markierbare Ressourcen. So können Sie beispielsweise eine Reihe von Tags für Arbeitsgruppen in Ihrem Konto erstellen, mit deren Hilfe Sie Eigentümer von Arbeitsgruppen nachverfolgen oder Arbeitsgruppen anhand ihres Zwecks identifizieren können. Wenn Sie die Tags auch als Kostenzuordnungs-Tags in der Billing and Cost Management Kostenmanagementkonsole aktivieren, werden die mit laufenden Abfragen verbundenen Kosten in Ihrem Kosten- und Nutzungsbericht mit diesem Kostenzuordnungs-Tag angezeigt. Wir empfehlen Ihnen, für das AWS-Tagging bewährte Methoden zu verwenden, um eine einheitliche Reihe von Tags festzulegen, die den Anforderungen Ihres Unternehmens entsprechen.

Sie können mit Tags arbeiten, indem Sie die Athena-Konsole oder die API-Operationen verwenden.

Themen

- Grundlagen zu Tags (Markierungen)
- Tag (Markierung)-Einschränkungen
- Arbeiten mit Tags auf Arbeitsgruppen in der Konsole

- Verwenden von Tag-Operationen
- Tagbasierte IAM-Zugriffssteuerungsrichtlinien

Grundlagen zu Tags (Markierungen)

Ein Tag (Markierung) ist eine Markierung, die Sie einer Athena-Ressource zuordnen. Jeder Tag (Markierung) besteht aus einem Schlüssel und einem optionalen Wert, beides können Sie bestimmen.

Mit Tags können Sie AWS-Ressourcen auf unterschiedliche Weise kategorisieren. Sie können zum Beispiel eine Reihe von Tags für die Arbeitsgruppen in Ihrem Konto definieren, um die Eigentümer oder den Zweck der einzelnen Arbeitsgruppen nachzuverfolgen.

Sie können Tags beim Erstellen einer neuen Athena-Arbeitsgruppe oder eines neuen Datenkatalogs hinzufügen. Außerdem können Sie Tags bearbeiten oder daraus entfernen. Sie können einen Tag in der Konsole bearbeiten. Wenn Sie API-Operationen zur Bearbeitung eines Tags verwenden, entfernen Sie den alten Tag und fügen Sie einen neuen hinzu. Wenn Sie eine Ressource löschen, werden alle Tags (Markierungen) der Ressource ebenfalls gelöscht.

Athena weist Ihren Ressourcen nicht automatisch Tags zu. Sie können Tag (Markierung)-Schlüssel und -Werte bearbeiten und Tags (Markierungen) jederzeit von einer Ressource entfernen. Sie können den Wert eines Tags (Markierung) zwar auf eine leere Zeichenfolge, jedoch nicht Null festlegen. Fügen Sie der gleichen Ressource keine doppelten Tag-Schlüssel hinzu. In diesem Fall gibt Athena eine Fehlermeldung aus. Wenn Sie die TagResource-Aktion verwenden, um eine Ressource mit einem vorhandenen Tag-Schlüssel zu markieren, überschreibt der neue Tag-Wert den alten.

In IAM können Sie steuern, welche Benutzer in Ihrem Amazon-Web-Services-Konto zum Erstellen, Bearbeiten, Entfernen oder Auflisten von Tags berechtigt sind. Weitere Informationen finden Sie unter Tagbasierte IAM-Zugriffssteuerungsrichtlinien.

Eine vollständige Liste mit Amazon-Athena-Tag-Aktionen finden Sie unter den Namen von API-Aktionen in der Amazon-Athena-API-Referenz.

Sie können Tags für Fakturierungszwecke verwenden. Weitere Informationen finden Sie unter Verwendung von Tags für die Fakturierung im AWS Billing and Cost Management-Benutzerhandbuch.

Weitere Informationen finden Sie unter Tag (Markierung)-Einschränkungen.

Tag (Markierung)-Einschränkungen

Für Tags gelten zwei Einschränkungen:

- In Athena können Sie Arbeitsgruppen und Datenkataloge mit Tags markieren. Abfragen können nicht markiert werden.
- Die maximale Anzahl an Tags pro Ressource beträgt 50. Um innerhalb dieser grenze zu bleiben, prüfen und entfernen Sie nicht verwendete Tags.
- Jeder Tag (Markierung) muss für jede Ressource eindeutig sein. Jeder Tag (Markierung) kann nur einen Wert haben. Fügen Sie derselben Ressource keine doppelten Tag-Schlüssel gleichzeitig hinzu. In diesem Fall gibt Athena eine Fehlermeldung aus. Wenn Sie eine Ressource mit einem vorhandenen Tag-Schlüssel in einer separaten `TagResource`-Aktion markieren, überschreibt der neue Tag-Wert den alten.
- Die Tagschlüssellänge ist 1-128 Unicode-Zeichen in UTF-8.
- Die Tagwertlänge ist 0-256 Unicode-Zeichen in UTF-8.

  Tagging-Operationen, wie etwa das Hinzufügen, Bearbeiten, Entfernen oder Auflisten von Tags, erfordern, dass Sie für die Arbeitsgruppenressource einen ARN angeben.
- Athena ermöglicht die Verwendung von Buchstaben, Ziffern, Leerräumen in UTF-8 sowie der folgenden Zeichen: + - = . _ : / @.
- Bei Tag-Schlüsseln und -Werten wird zwischen Groß- und Kleinschreibung unterschieden.
- Das `"aws:"`-Präfix in Tag-Schlüsseln ist zur Verwendung durch AWS reserviert. Sie können keine Tag-Schlüssel oder mit diesem Präfix bearbeiten oder löschen. Tags mit diesem Präfix werden nicht für Ihr Tags-pro-Ressource-Limit angerechnet.
- Die von Ihnen zugewiesenen Tags sind nur für Ihr Amazon-Web-Services-Konto verfügbar.

Arbeiten mit Tags auf Arbeitsgruppen in der Konsole

Mithilfe der Athena-Konsole können Sie sehen, welche Tags von den einzelnen Arbeitsgruppen in Ihrem Konto verwendet werden. Sie können Tags nur nach Arbeitsgruppe anzeigen. Sie können auch die Athena-Konsole verwenden, um Tags von jeweils einer Arbeitsgruppe anzuwenden, zu bearbeiten oder zu entfernen.

Sie können Arbeitsgruppen mithilfe der Tags, die Sie erstellt haben, suchen.

Themen

Anzeigen von Tags für einzelne Arbeitsgruppen

Um Tags für eine einzelne Arbeitsgruppe in der Athena-Konsole anzuzeigen

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.

3. Wählen Sie im Navigationsmenü Workgroups (Arbeitsgruppen) und dann die gewünschte Arbeitsgruppe aus.
4. Führen Sie eine der folgenden Aktionen aus:
   - Wählen Sie die Registerkarte Tags aus. Verwenden Sie das Suchfeld, wenn die Liste der Tags lang ist.
   - Wählen Sie Edit (Bearbeiten) und scrollen Sie dann nach unten zum Abschnitt Tags.

Hinzufügen und Löschen von Tags für einzelne Arbeitsgruppen

Sie können Tags für einzelne Arbeitsgruppen direkt auf der Registerkarte Workgroups (Arbeitsgruppen) verwalten.

> **Note**
>
> Wenn Benutzer Tags hinzufügen sollen, wenn sie eine Arbeitsgruppe in der Konsole erstellen, oder Tags übergeben sollen, wenn sie die CreateWorkGroup-Aktion verwenden, stellen Sie sicher, dass Sie den Benutzern IAM-Berechtigungen für die Aktionen TagResource und CreateWorkGroup erteilen.

So fügen Sie ein Tag hinzu, wenn Sie eine neue Arbeitsgruppe erstellen

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wählen Sie im Navigationsmenü Workgroups (Arbeitsgruppen).
3. Wählen Sie Create workgroup (Arbeitsgruppe erstellen), und füllen Sie die Werte nach Bedarf ein. Die detaillierten Schritte finden Sie unter Erstellen von Arbeitsgruppen.
4. Fügen Sie im Abschnitt Tags einen oder mehrere Tags hinzu, indem Sie Schlüssel und Werte angeben. Fügen Sie keine doppelten Tag-Schlüssel gleichzeitig derselben Arbeitsgruppe hinzu. In diesem Fall gibt Athena eine Fehlermeldung aus. Weitere Informationen finden Sie unter Tag (Markierung)-Einschränkungen.
5. Wählen Sie anschließend Create workgroup (Arbeitsgruppe erstellen).

So fügen Sie einen Tag einer vorhandenen Arbeitsgruppe hinzu oder bearbeiten ihn:

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wählen Sie im Navigationsbereich die Option Workgroups (Arbeitsgruppen) aus.
3. Wählen Sie die Arbeitsgruppe aus, die Sie ändern möchten.
4. Führen Sie eine der folgenden Aktionen aus:
   - Wählen Sie die Registerkarte Tags und dann Manage tags (Tags verwalten).
   - Wählen Sie Edit (Bearbeiten) und scrollen Sie dann nach unten zum Abschnitt Tags.
5. Geben Sie einen Schlüssel und den Wert für jedes Tag an. Weitere Informationen finden Sie unter Tag (Markierung)-Einschränkungen.
6. Wählen Sie Save (Speichern) aus.

So löschen Sie einen Tag aus einer einzelnen Arbeitsgruppe:

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wählen Sie im Navigationsbereich die Option Workgroups (Arbeitsgruppen) aus.
3. Wählen Sie die Arbeitsgruppe aus, die Sie ändern möchten.
4. Führen Sie eine der folgenden Aktionen aus:
   - Wählen Sie die Registerkarte Tags und dann Manage tags (Tags verwalten).
   - Wählen Sie Edit (Bearbeiten) und scrollen Sie dann nach unten zum Abschnitt Tags.
5. Wählen Sie in der Liste der Tags Remove (Entfernen) für das Tag, das Sie löschen möchten, und wählen Sie dann Save (Speichern).

Verwenden von Tag-Operationen

Verwenden Sie die folgenden Tag-Operationen, um Tags für eine Ressource hinzuzufügen, zu entfernen oder aufzulisten.

API	CLI	Aktionsbeschreibung
`TagResource`	`tag-resource`	Fügen Sie ein oder mehrere Tags für die Ressource hinzu, die den angegebenen ARN hat, oder überschreiben Sie diese.
`UntagResource`	`untag-resource`	Löschen Sie ein oder mehrere Tags aus der Ressource, die den angegebenen ARN hat.
`ListTagsForResource`	`list-tags-for-resource`	Listen Sie ein oder mehrere Tags für die Ressource auf, die den angegebenen ARN hat.

Hinzufügen von Tags beim Erstellen einer Ressource

Wenn Sie beim Erstellen einer Arbeitsgruppe oder eines Datenkatalogs Tags hinzufügen möchten, verwenden Sie den `tags`-Parameter mit den `CreateWorkGroup`- oder `CreateDataCatalog`-API-Operationen oder mit den `create-data-catalog`-Befehlen AWS CLI oder `create-work-group`.

Verwalten von Tags mithilfe von API-Operationen

Die Beispiele in diesem Abschnitt zeigen, wie Tag-API-Operationen zum Verwalten von Tags auf Arbeitsgruppen und Datenkatalogen verwendet werden. Die Beispiele werden in der Programmiersprache Java gegeben.

Example TagResource

Im folgenden Beispiel werden der Arbeitsgruppe `workgroupA` zwei Tags hinzugefügt:

```
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);
```

Im folgenden Beispiel werden dem Datenkatalog `datacatalogA` zwei Tags hinzugefügt:

```
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**
>
> Fügen Sie der gleichen Ressource keine doppelten Tag-Schlüssel hinzu. In diesem Fall gibt Athena eine Fehlermeldung aus. Wenn Sie eine Ressource mit einem vorhandenen Tag-Schlüssel in einer separaten `TagResource`-Aktion markieren, überschreibt der neue Tag-Wert den alten.

Example UntagResource

Im folgenden Beispiel wird `tagKey2` aus der Arbeitsgruppe `workgroupA` entfernt:

```
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);
```

Im folgenden Beispiel wird `tagKey2` aus dem Datenkatalog `datacatalogA` entfernt:

```
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

Im folgenden Beispiel werden Tags für die Arbeitsgruppe `workgroupA` aufgelistet:

```
ListTagsForResourceRequest request = new ListTagsForResourceRequest()
    .withResourceARN("arn:aws:athena:us-east-1:123456789012:workgroup/workgroupA");

ListTagsForResourceResult result = client.listTagsForResource(request);

List<Tag> resultTags = result.getTags();
```

Im folgenden Beispiel werden Tags für den Datenkatalog `datacatalogA` aufgelistet:

```
ListTagsForResourceRequest request = new ListTagsForResourceRequest()
    .withResourceARN("arn:aws:athena:us-east-1:123456789012:datacatalog/datacatalogA");

ListTagsForResourceResult result = client.listTagsForResource(request);
```

```
List<Tag> resultTags = result.getTags();
```

Verwalten von Tags mit dem AWS CLI

In den folgenden Abschnitten wird gezeigt, wie Sie mit AWS CLI Tags für Datenkataloge erstellen und verwalten.

Hinzufügen von Tags zu einer Ressource: Tag-Ressource

Der `tag-resource`-Befehl fügt einzelne oder mehrere Tags einer angegebenen Ressource hinzu

Syntax

```
aws athena tag-resource --resource-arn
arn:aws:athena:region:account_id:datacatalog/catalog_name --tags
Key=string,Value=string Key=string,Value=string
```

Der `--resource-arn`-Parameter gibt die Ressource an, der die Tags hinzugefügt werden. Der `--tags`-Parameter gibt eine Liste von durch Leerzeichen getrennten Schlüssel-Wert-Paaren an, die der Ressource als Tags hinzugefügt werden sollen.

Example

Im folgenden Beispiel werden dem `mydatacatalog`-Datenkatalog Tags hinzugefügt.

```
aws athena tag-resource --resource-arn arn:aws:athena:us-
east-1:111122223333:datacatalog/mydatacatalog --tags Key=Color,Value=Orange
 Key=Time,Value=Now
```

Um das Ergebnis anzuzeigen, verwenden Sie den `list-tags-for-resource`-Befehl.

Weitere Informationen zum Hinzufügen von Tags bei Verwendung des `create-data-catalog`-Befehls finden Sie unter Registrierung eines Katalogs: Create-data-catalog.

Auflisten der Tags für eine Ressource: List-tags-for-resource

Der `list-tags-for-resource`-Befehl listet die Tags für die angegebene Ressource auf.

Syntax

```
aws athena list-tags-for-resource --resource-arn
arn:aws:athena:region:account_id:datacatalog/catalog_name
```

Der `--resource-arn`-Parameter gibt die Ressource an, für die die Tags aufgelistet werden.

Im folgenden Beispiel werden die Tags für den `mydatacatalog`-Datenkatalog aufgelistet.

```
aws athena list-tags-for-resource --resource-arn arn:aws:athena:us-
east-1:111122223333:datacatalog/mydatacatalog
```

Das folgende Beispielergebnis verwendet das JSON-Format.

```
{
    "Tags": [
        {
            "Key": "Time",
            "Value": "Now"
        },
        {
            "Key": "Color",
            "Value": "Orange"
        }
    ]
}
```

Entfernen von Tags von einer Ressource: Untag-resource

Der `untag-resource`-Befehl entfernt die angegebenen Tag-Schlüssel und die zugehörigen Werte von der angegebenen Ressource.

Syntax

```
aws athena untag-resource --resource-arn
arn:aws:athena:region:account_id:datacatalog/catalog_name --tag-keys
key_name [key_name ...]
```

Der `--resource-arn`-Parameter gibt die Ressource an, von der die Tags entfernt werden. Der `--tag-keys`-Parameter erstellt eine durch Leerzeichen getrennte Liste von Schlüsselnamen. Für jeden angegebenen Schlüsselnamen entfernt der `untag-resource`-Befehl sowohl den Schlüssel als auch seinen Wert.

Im folgenden Beispiel werden die Schlüssel `Color` und `Time` sowie deren Werte aus der `mydatacatalog`-Katalogressource entfernt.

```
aws athena untag-resource --resource-arn arn:aws:athena:us-
east-1:111122223333:datacatalog/mydatacatalog --tag-keys Color Time
```

Tagbasierte IAM-Zugriffssteuerungsrichtlinien

Mit Tags können Sie eine IAM-Richtlinie schreiben, die den `Condition`-Block enthält, um den Zugriff auf eine Ressource basierend auf ihren Tags zu steuern.

Tag-Richtlinienbeispiele für Arbeitsgruppen

Example 1. Grundlegende Markierungsrichtlinie

Die folgende IAM-Richtlinie ermöglicht Ihnen das Ausführen von Abfragen und die Interaktion mit Tags für die Arbeitsgruppe mit dem Namen `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: Richtlinienblock, der Aktionen auf einer Arbeitsgruppe auf der Grundlage eines Tagschlüssel-/Tagwert-Paares verweigert

Tags, die einer bestehenden Ressource, beispielsweise einer Arbeitsgruppe, zugeordnet sind, werden als Ressourcen-Tags bezeichnet. Mit Ressourcen-Tags können Sie Richtlinienblöcke wie die folgenden schreiben, die die aufgelisteten Aktionen für jede Arbeitsgruppe ablehnen, die mit einem Schlüssel-Wert-Paar wie `stack`, `production` gekennzeichnet sind.

```
{
    "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. Richtlinienblock, der Tags ändernde Aktionsanfragen auf die angegebenen Tags beschränkt

Tags, die als Parameter an Operationen übergeben werden, die Tags ändern (z. B. `TagResource`, `UntagResource` oder `CreateWorkGroup` mit Tags) werden als Anforderungs-Tags bezeichnet. Der folgende Beispielrichtlinienblock erlaubt die `CreateWorkGroup`-Operation nur, wenn eines der übergebenen Tags den Schlüssel `costcenter` und den Wert `1`, `2` oder `3` hat.

> **Note**
>
> Wenn Sie zulassen möchten, dass eine IAM-Rolle Tags als Teil einer `CreateWorkGroup`-Operation weitergibt, stellen Sie sicher, dass Sie der Rolle Berechtigungen für die `TagResource`- und `CreateWorkGroup`-Aktionen erteilen.

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

Tag-Richtlinienbeispiele für Datenkataloge

Example 1. Grundlegende Markierungsrichtlinie

Mit der folgenden IAM-Richtlinie können Sie mit Tags für den Datenkatalog mit dem Namen `datacatalogA` interagieren:

```
{
    "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: Richtlinienblock, der Aktionen in einem Datenkatalog auf der Grundlage eines Tag-Schlüssel-/Tag-Wert-Paares verweigert

Sie können Ressourcen-Tags verwenden, um Richtlinienblöcke zu schreiben, die bestimmte Aktionen in Datenkatalogen verweigern, die mit bestimmten Tag-Schlüssel-Wert-Paaren markiert sind. In der folgenden Beispielrichtlinie werden Aktionen für Datenkataloge verweigert, die das Tag-Schlüssel-Wert-Paar `stack`, `production` haben.

```
{
    "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. Richtlinienblock, der Tags ändernde Aktionsanfragen auf die angegebenen Tags beschränkt

Tags, die als Parameter an Operationen übergeben werden, die Tags ändern (z. B. `TagResource`, `UntagResource` oder `CreateDataCatalog` mit Tags) werden als Anforderungs-Tags bezeichnet.

Der folgende Beispielrichtlinienblock erlaubt die `CreateDataCatalog`-Operation nur, wenn eines der übergebenen Tags den Schlüssel `costcenter` und den Wert 1, 2 oder 3 hat.

> **Note**
>
> Wenn Sie zulassen möchten, dass eine IAM-Rolle Tags als Teil einer `CreateDataCatalog`-Operation weitergibt, stellen Sie sicher, dass Sie der Rolle Berechtigungen für die `TagResource`- und `CreateDataCatalog`-Aktionen erteilen.

```
{
    "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**
>
> Die Konsole „Service Quotas“ stellt Informationen zu Amazon-Athena-Kontingenten bereit. Sie können auch die Service-Quotas-Konsole verwenden, um Kontingenterhöhungen für Kontingente anzufordern, die anpassbar sind. Informationen zu AWS Glue -bezogenen

Schemaeinschränkungen finden Sie auf der Seite Endpunkte und Kontingente vonAWS Glue. Allgemeine Informationen zu AWS Service Quotas finden Sie unter AWS Service Quotas im Allgemeine AWS-Referenz.

Abfragen

Ihr Konto verfügt über die folgenden abfragebezogenen Kontingente für Amazon Athena. Details dazu finden Sie auf der Seite Endpunkte und Kontingente von Amazon Athena in der Allgemeine AWS-Referenz.

- Active DDL queries (Aktive DDL-Abfragen) – Die Anzahl der aktiven DDL-Abfragen. Zu DDL-Abfragen gehören `CREATE TABLE`- und `ALTER TABLE ADD PARTITION`-Abfragen.
- DDL query timeout (DDL-Abfrage-Timeout) – Die maximale Zeit in Minuten, die eine DDL-Abfrage ausgeführt werden kann, bevor sie abgebrochen wird.
- Aktive DML-Abfragen – Die Anzahl der aktiven DML-Abfragen. Zu DML-Abfragen gehören `SELECT`- und `CREATE TABLE AS`(CTAS)-, und `INSERT INTO`-Abfragen. Die spezifischen Kontingente variieren je AWS -Region.
- DML-Abfrage-Timeout – Die maximale Zeit in Minuten, die eine DML-Abfrage ausgeführt werden kann, bevor sie abgebrochen wird. Sie können eine Erhöhung dieses Timeouts auf maximal 240 Minuten beantragen.

Um Kontingenterhöhungen anzufordern, können Sie die Konsole von Athena Service Quotas verwenden.

Athena verarbeitet Abfragen, indem Ressourcen je nach der Gesamtauslastung des Service sowie der Anzahl eingehender Anforderungen zugewiesen werden. Ihre Abfragen werden möglicherweise vorübergehend in die Warteschlange gestellt, bevor sie ausgeführt werden. Asynchrone Prozesse nehmen die Abfragen aus Warteschlangen auf und führen sie auf physischen Ressourcen aus, sobald die Ressourcen verfügbar sind und solange die Kontokonfiguration dies zulässt.

Ein DML- oder DDL-Abfragekontingent umfasst sowohl laufende als auch Queues in der Warteschlange. Wenn Ihr DML-Abfragekontingent beispielsweise 25 beträgt und Ihre Gesamtzahl der ausgeführten und in die Warteschlange eingestellten Abfragen 26 beträgt, führt Abfrage 26 zu einem TooManyRequestsException Fehler.

> **Note**
>
> Wenn Sie die Parallelität der Abfragen, die Sie in Athena ausführen, direkt steuern möchten, können Sie Kapazitätsreservierungen verwenden. Weitere Informationen finden Sie unter Kapazität zur Abfrageverarbeitung verwalten.

Länge der Abfragezeichenfolge

Die maximal zulässige Länge einer Abfragezeichenfolge ist 262144 Byte, wobei die Zeichenfolgen in UTF-8 kodiert sind. Dies ist kein anpassbares Kontingent. Sie können diese Einschränkung jedoch umgehen, indem Sie lange Abfragen in mehrere kleinere Abfragen aufteilen. Weitere Informationen finden Sie unter Wie kann ich die maximale Abfragezeichenfolgenlänge in Athena erhöhen? im AWS -Wissenscenter.

Arbeitsgruppen

Beachten Sie beim Arbeiten mit Athena-Arbeitsgruppen die folgenden Punkte:

- Athena-Service-Quotas werden für alle Arbeitsgruppen in einem Konto freigegeben.
- Die maximale Anzahl der Arbeitsgruppen, die Sie pro Region in einem Konto erstellen können, ist 1.000.
- Die maximale Anzahl von vorbereiteten Anweisungen in einer Arbeitsgruppe beträgt 1 000.
- Die maximale Anzahl der Tags pro Arbeitsgruppe ist 50. Weitere Informationen finden Sie unter Tag (Markierung)-Einschränkungen.

Datenbanken, Tabellen und Partitionen

- Wenn Sie die AWS Glue Data Catalog mit Athena verwenden, finden Sie unter -AWS Glue Endpunkte und -Kontingente Informationen zu Servicekontingenten für Tabellen, Datenbanken und Partitionen, z. B. die maximale Anzahl von Datenbanken oder Tabellen pro Konto.
  - Obwohl Athena das Abfragen von AWS Glue Tabellen mit 10 Millionen Partitionen unterstützt, kann Athena nicht mehr als 1 Million Partitionen in einem einzigen Scan lesen.
- Wenn Sie nicht verwenden AWS Glue Data Catalog, beträgt die Anzahl der Partitionen pro Tabelle 20.000. Sie können eine Kontingenterhöhung beantragen.

Amazon-S3-Buckets

Wenn Sie mit Amazon-S3-Buckets arbeiten, beachten Sie die folgenden Punkte:

- Amazon S3 verfügt über ein Standard-Servicekontingent von 100 Buckets pro Konto.
- Athena erfordert für das Protokollieren von Ergebnissen einen separaten Bucket.
- Sie können eine Kontingenterhöhung um bis zu 1.000 Amazon-S3-Buckets pro AWS -Konto beantragen.

API-Aufrufkontingente pro Konto

Für Athena-APIs gelten die folgenden Standardkontingente für die Anzahl der Aufrufe an die API pro Konto (nicht pro Abfrage):

API-Name	Standardanzahl der Aufrufe pro Sekunde	Burst-Kapazität
`BatchGetNamedQuery` , `ListNamedQueries` , `ListQueryExecutions`	5	bis zu 10
`CreateNamedQuery` , `DeleteNamedQuery` , `GetNamedQuery`	5	bis zu 20
`BatchGetQueryExecution`	20	bis zu 40
`StartQueryExecution` , `StopQueryExecution`	20	bis zu 80
`GetQueryExecution` , `GetQueryResults`	100	bis zu 200

Beispielsweise können Sie bis zu 20 Aufrufe pro Sekunde für `StartQueryExecution` tätigen. Wenn diese API 4 Sekunden lang nicht aufgerufen wird, sammelt Ihr Konto zudem eine Burst-Kapazität von bis zu 80 Aufrufen. In diesem Fall kann Ihre Anwendung im Burst-Modus bis zu 80 Aufrufe für diese API durchführen.

Wenn Sie eine dieser APIs verwenden und das Standardkontingent für die Anzahl der Aufrufe pro Sekunde oder die Burst-Kapazität in Ihrem Konto überschreiten, gibt die Athena-API einen

Fehler ähnlich dem folgenden aus: „“ClientError: Beim ThrottlingExceptionAufrufen der Operation <API_name> ist ein Fehler aufgetreten (): Rate überschritten. Reduzieren Sie die Anzahl der Aufrufe pro Sekunde oder die Burst-Kapazität für die API für dieses Konto.

Das Kontingent von Athena für API-Aufrufe pro Konto kann nicht in der Athena-Service-Quotas-Konsole geändert werden. Um eine Erhöhung des Kontingents für Athena-API-Aufrufe zu beantragen, navigieren Sie zur Seite zur Erhöhung des AWS Support -Service-Limits, füllen Sie das Formular aus und senden Sie es ab.

Athena-Engine-Versionierung

Athena veröffentlicht gelegentlich eine neue Engine-Version, um verbesserte Leistung, Funktionalität und Code-Fehlerbehebungen bereitzustellen. Wenn eine neue Motorversion verfügbar ist, benachrichtigt Athena Sie über die Athena-Konsole und Ihre AWS Health Dashboard. Ihr AWS Health Dashboard benachrichtigt Sie über Ereignisse, die sich auf Ihre AWS Services oder Ihr Konto auswirken können. Weitere Informationen zu finden Sie AWS Health Dashboardunter Erste Schritte mit der AWS Health Dashboard.

Die Engine-Versionierung wird über die Arbeitsgruppe konfiguriert. Sie können Arbeitsgruppen verwenden, um zu steuern, welche Abfrage-Engine Ihre Abfragen verwenden und ob Athena Ihre Arbeitsgruppen automatisch aktualisieren soll. Die verwendete Abfrage-Engine wird im Abfrage-Editor, auf der Arbeitsgruppendetailseite angezeigt und ist über die Athena-APIs verfügbar.

- Standardmäßig sind Arbeitsgruppen für ein automatisches Upgrade konfiguriert. Wenn eine Arbeitsgruppe auf automatisches Upgrade eingestellt ist, aktualisiert Athena die Arbeitsgruppe für Sie, es sei denn, es findet Inkompatibilitäten.
- Wenn Sie eine Arbeitsgruppe für die Verwendung einer bestimmten Version konfigurieren, ändert Athena die Version der Arbeitsgruppe nicht.

In beiden Fällen aktualisiert Athena Ihre Arbeitsgruppen, wenn eine Version nicht mehr verfügbar ist. Athena benachrichtigt Sie AWS Health Dashboard darüber, wann eine Engine-Version nicht mehr angeboten wird. Ihr AWS Health Dashboard benachrichtigt Sie über Ereignisse, die sich auf Ihre AWS Services oder Ihr Konto auswirken können. Weitere Informationen zu finden Sie AWS Health Dashboardunter Erste Schritte mit der AWS Health Dashboard.

Wenn Sie eine neue Modulversion verwenden, kann eine kleine Teilmenge von Abfragen aufgrund von Inkompatibilitäten unterbrochen werden. Breaking Changes werden angekündigt, wenn eine

neue Athena-Version veröffentlicht wird. Sie sollten Arbeitsgruppen verwenden, um Ihre Abfragen vor dem Upgrade zu testen, indem Sie eine Testarbeitsgruppe erstellen, die die neue Engine verwendet, oder indem Sie das Upgrade einer vorhandenen Arbeitsgruppe testen. Weitere Informationen finden Sie unter Testen von Abfragen im Voraus eines Engine-Versions-Upgrades.

Themen

Ändern von Athena-Engine-Versionen

Athena veröffentlicht gelegentlich eine neue Engine-Version, um verbesserte Leistung, Funktionalität und Code-Fehlerbehebungen bereitzustellen. Wenn eine neue Engine-Version verfügbar ist, benachrichtigt Athena Sie über die Konsole. Sie können Athena entscheiden lassen, wann ein Upgrade durchgeführt werden soll, oder manuell eine Athena-Engine-Version pro Arbeitsgruppe angeben.

Themen

Suchen der Version der Abfrage-Engine für eine Arbeitsgruppe

Sie können die Workgroups (Arbeitsgruppen) verwenden, um die aktuelle Engine-Version für jede Arbeitsgruppe zu finden.

So suchen Sie die aktuelle Engine-Version für eine beliebige Arbeitsgruppe

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.

3. Klicken Sie im Navigationsbereich der Athena-Konsole auf Workgroups (Arbeitsgruppen).
4. Suchen Sie auf der Seite Workgroups (Arbeitsgruppen) die gewünschte Arbeitsgruppe. Die Abfrage-Engine-Version-Spalte für die Arbeitsgruppe zeigt die Version der Abfrage-Engine an.

Ändern der Engine-Version in der Athena-Konsole

Wenn eine neue Engine-Version verfügbar ist, können Sie Athena entscheiden lassen, wann die Arbeitsgruppe aktualisiert werden soll, oder manuell die Athena-Engine-Version angeben, die von der Arbeitsgruppe verwendet wird. Wenn derzeit nur eine Version verfügbar ist, ist die manuelle Angabe einer anderen Version nicht möglich.

> **Note**
>
> Um die Engine-Version für eine Arbeitsgruppe zu ändern, benötigen Sie die Berechtigung zum Ausführen der `athena:ListEngineVersions`-Aktion für die Arbeitsgruppe. Eine IAM-Beispielrichtlinie finden Sie unter Beispiel-Arbeitsgruppenrichtlinien.

Athena entscheiden lassen, wann die Arbeitsgruppe aktualisiert werden soll

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.
3. Klicken Sie im Navigationsbereich der Konsole auf Workgroups (Arbeitsgruppen).

4. Wählen Sie in der Liste der Arbeitsgruppen den Link für die Arbeitsgruppe aus, den sie konfigurieren möchten.
5. Wählen Sie Edit (Bearbeiten).
6. Wählen Sie im Abschnitt Version der Abfrage-Engine für Abfrage-Engine aktualisieren die Option Automatisch aus, damit Athena auswählen kann, wann Ihre Arbeitsgruppe aktualisiert werden soll. Dies ist die Standardeinstellung.
7. Wählen Sie Save Changes (Änderungen speichern).

   In der Liste der Arbeitsgruppen zeigt der Query engine update status (Aktualisierungsstatus des Abfrage-Engines) für die Arbeitsgruppe Automatic (Automatisch) an.

So wählen Sie manuell eine Engine-Version

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.
3. Klicken Sie im Navigationsbereich der Konsole auf Workgroups (Arbeitsgruppen).
4. Wählen Sie in der Liste der Arbeitsgruppen den Link für die Arbeitsgruppe aus, den sie konfigurieren möchten.
5. Wählen Sie Edit (Bearbeiten).
6. Wählen Sie im Abschnitt Query engine version (Abfrage-Engine-Version) für Update query engine (Abfrage-Engine aktualisieren) die Option Manual (Manuell), um manuell eine Engine-Version auszuwählen.
7. Verwenden Sie die Option, Abfrage-Engine-Version, um die Engine-Version auszuwählen, die die Arbeitsgruppe verwenden soll. Wenn eine andere Engine-Version nicht verfügbar ist, kann keine andere Engine-Version angegeben werden.
8. Wählen Sie Save Changes (Änderungen speichern).

   In der Liste der Arbeitsgruppen zeigt der Query engine update status (Aktualisierungsstatus des Abfrage-Engines) für die Arbeitsgruppe Manual (Manuell) an.

Ändern der Engine-Version mithilfe der AWS CLI

Verwenden Sie die Syntax im folgenden Beispiel AWS CLI, um die Engine-Version mit der zu ändern.

```
aws athena update-work-group --work-group workgroup-name --configuration-updates
 EngineVersion={SelectedEngineVersion='Athena engine version 3'}
```

Angeben der Engine-Version beim Erstellen einer Arbeitsgruppe

Wenn Sie eine Arbeitsgruppe erstellen, können Sie die von der Arbeitsgruppe verwendete Engine-Version angeben oder Athena entscheiden lassen, wann die Arbeitsgruppe aktualisiert werden soll. Wenn eine neue Engine-Version verfügbar ist, besteht eine bewährte Methode darin, eine Arbeitsgruppe zu erstellen, um die neue Engine zu testen, bevor Sie die anderen Arbeitsgruppen aktualisieren. Um die Engine-Version für eine Arbeitsgruppe anzugeben, müssen Sie über die `athena:ListEngineVersions`-Berechtigung für die Arbeitsgruppe verfügen. Eine IAM-Beispielrichtlinie finden Sie unter Beispiel-Arbeitsgruppenrichtlinien.

So geben Sie die Engine-Version beim Erstellen einer Arbeitsgruppe an

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.
3. Klicken Sie im Navigationsbereich der Konsole auf Workgroups (Arbeitsgruppen).
4. Wählen Sie auf der Seite Workgroups (Arbeitsgruppen) die Option Create workgroup (Arbeitsgruppe erstellen) aus.
5. Führen Sie auf der Seite Create workgroup (Arbeitsgruppe erstellen) im Abschnitt Query engine version (Version der Abfrage-Engine) einen der folgenden Schritte aus:
   - Wählen Sie Automatisch aus, damit Athena entscheiden kann, wann Ihre Arbeitsgruppe aktualisiert werden soll. Dies ist die Standardeinstellung.
   - Wählen Sie Manual (Manuell), um manuell eine andere Engine-Version auszuwählen, falls eine verfügbar ist.
6. Geben Sie bei Bedarf Informationen für die anderen Felder ein. Informationen zu den anderen Feldern finden Sie unter Erstellen von Arbeitsgruppen.
7. Wählen Sie Create workgroup (Arbeitsgruppe erstellen) aus.

Testen von Abfragen im Voraus eines Engine-Versions-Upgrades

Wenn eine Arbeitsgruppe auf eine neue Engine-Version aktualisiert wird, können einige Ihrer Abfragen aufgrund von Inkompatibilitäten unterbrochen werden. Um sicherzustellen, dass das Upgrade der Engine-Version reibungslos funktioniert, können Sie Ihre Abfragen im Voraus testen.

So testen Sie Ihre Abfragen vor einem Upgrade der Engine-Version

1. Überprüfen Sie, ob die Engine-Version der Arbeitsgruppe verwendet wird. Die von Ihnen verwendete Engine-Version wird auf der Workgroups (Arbeitsgruppen)-Seite in der Query engine version (Abfrage-Engine-Version)-Spalte für die Arbeitsgruppe angezeigt. Weitere Informationen finden Sie unter Suchen der Version der Abfrage-Engine für eine Arbeitsgruppe.
2. Erstellen Sie eine Testarbeitsgruppe, die die neue Engine-Version verwendet. Weitere Informationen finden Sie unter Angeben der Engine-Version beim Erstellen einer Arbeitsgruppe.
3. Verwenden Sie die neue Arbeitsgruppe, um die Abfragen auszuführen, die Sie testen möchten.
4. Wenn eine Abfrage fehlschlägt, verwenden Sie die Versionsreferenz der Athena-Engine, um zu überprüfen, ob Änderungen geändert werden, die sich auf die Abfrage auswirken könnten. Einige Änderungen erfordern möglicherweise, dass Sie die Syntax Ihrer Abfragen aktualisieren.
5. Wenn Ihre Abfragen immer noch fehlschlagen, wenden Sie sich an , um Unterstützung AWS Support zu erhalten. Wählen Sie in der AWS Management ConsoleSupport, Supportcenter, oder stellen Sie eine Frage zu AWS re:Post, indem Sie das Amazon-Athena-Tag verwenden.

Fehlerbehebung bei Abfragen, die fehlschlagen

Wenn eine Abfrage nach einem Upgrade der Engine-Version fehlschlägt, verwenden Sie Versionsreferenz der Athena-Engine, um nach wichtigen Änderungen zu suchen, einschließlich Änderungen, die sich auf die Syntax in Ihren Abfragen auswirken können.

Wenn Ihre Abfragen immer noch fehlschlagen, wenden Sie sich an , um Unterstützung AWS Support zu erhalten. Wählen Sie in der die Option Support AWS Management Console, Support Center oder stellen Sie mithilfe des Amazon Athena-Tags eine Frage zu AWS re:Post.

Versionsreferenz der Athena-Engine

In diesem Abschnitt werden die Änderungen an der Athena-Abfrage-Engine aufgeführt.

Themen

Athena-Engine-Version 3

Für die Engine-Version 3 hat Athena einen kontinuierlichen Integrationsansatz für die Verwaltung von Open-Source-Software eingeführt, der die Aktualität mit den Trino- und Presto-Projekten verbessert. So erhalten Sie schnelleren Zugriff auf Verbesserungen der Community, die in die Athena-Engine integriert und darauf abgestimmt sind.

Diese Version der Athena-Engine-Version 3 unterstützt alle Features der Athena-Engine-Version 2. Dieses Dokument hebt die wichtigsten Unterschiede zwischen der Athena-Engine-Version 2 und der Athena-Engine-Version 3 hervor. Weitere Informationen finden Sie im AWS-Big-Data-Blogartikel Upgrade auf Athena Engine Version 3, um die Abfrageleistung zu erhöhen und auf mehr Analysefeatures zuzugreifen.

Erste Schritte

Erstellen Sie zunächst entweder eine neue Athena-Arbeitsgruppe, die Athena engine version 3 (Athena-Engine-Version 3) verwendet, oder konfigurieren Sie eine vorhandene Arbeitsgruppe für die Verwendung von Version 3. Jede Athena-Arbeitsgruppe kann eine Aktualisierung von der Engine-

Version 2 auf die Engine-Version 3 durchführen, ohne dass Ihre Fähigkeit zur Übermittlung von Abfragen unterbrochen wird.

Weitere Informationen finden Sie unter Ändern von Athena-Engine-Versionen.

Neue Features und Verbesserungen

Die aufgeführten Features und Aktualisierungen umfassen Verbesserungen von Athena selbst und von Funktionen, die von Open-Source-Trino integriert wurden. Eine vollständige Liste der SQL-Abfrageoperatoren und -Funktionen finden Sie in der Trino-Dokumentation.

Hinzugefügte Features

Unterstützung für den Apache-Spark-Bucketing-Algorithmus

Athena kann Buckets lesen, die vom Spark-Hash-Algorithmus generiert wurden. Um anzugeben, dass Daten ursprünglich vom Spark-Hash-Algorithmus geschrieben wurden, fügen Sie `('bucketing_format'='spark')` in die `TBLPROPERTIES`-Klausel Ihrer `CREATE TABLE`-Anweisung ein. Wenn diese Eigenschaft nicht angegeben wird, wird der Hive-Hash-Algorithmus verwendet.

```
CREATE EXTERNAL TABLE `spark_bucket_table`(
  `id` int,
  `name` string
  )
CLUSTERED BY (`name`)
INTO 8 BUCKETS
STORED AS PARQUET
LOCATION
  's3://path/to/bucketed/table/'
TBLPROPERTIES ('bucketing_format'='spark')
```

Erweiterte Funktionen

Die Funktionen in diesem Abschnitt sind neu in der Athena-Engine-Version 3.

Aggregationsfunktionen

listagg(x, separator) – Gibt die verketteten Eingabewerte zurück, getrennt durch die Trennzeichenfolge.

```
SELECT listagg(value, ',') WITHIN GROUP (ORDER BY value) csv_value
```

```
FROM (VALUES 'a', 'c', 'b') t(value);
```

Array-Funktionen

contains_sequence(x, seq) – Gibt wahr zurück, wenn Array x alle Array seq als sequentielle Teilmenge enthält (alle Werte in derselben aufeinanderfolgenden Reihenfolge).

```
SELECT contains_sequence(ARRAY [1,2,3,4,5,6], ARRAY[1,2]);
```

Binäre Funktionen

murmur3(binary) – Berechnet den 128-Bit-MurmurHash3-Hash der Binärdatei.

```
SELECT murmur3(from_base64('aaaaaa'));
```

Konvertierungs-Funktionen

format_number(number) – Gibt eine formatierte Zeichenfolge mithilfe eines Einheitensymbols zurück.

```
SELECT format_number(123456); -- '123K'
```

```
SELECT format_number(1000000); -- '1M'
```

Datums- und Zeitfunktionen

timezone_hour(timestamp) – Gibt die Stunde des Zeitzonen-Offsets aus dem Zeitstempel zurück.

```
SELECT EXTRACT(TIMEZONE_HOUR FROM TIMESTAMP '2020-05-10 12:34:56 +08:35');
```

timezone_minute(Zeitstempel) – Gibt die Minute des Zeitzonen-Offsets aus dem Zeitstempel zurück.

```
SELECT EXTRACT(TIMEZONE_MINUTE FROM TIMESTAMP '2020-05-10 12:34:56 +08:35');
```

Geodatenfunktionen

to_encoded_polyline(Geometry) – Codiert eine Linienfolge oder einen Mehrpunkt in eine Polylinie.

```
SELECT to_encoded_polyline(ST_GeometryFromText(
   'LINESTRING (-120.2 38.5, -120.95 40.7, -126.453 43.252)'));
```

from encoded polyline(varchar) – Decodiert eine Polylinie in eine Linienfolge.

```
SELECT ST_AsText(from_encoded_polyline('_p~iF~ps|U_ulLnnqC_mqNvxq`@'));
```

to_geojson_geometry(SphericalGeography) – Gibt die angegebene sphärische Geographie im GeoJSON-Format zurück.

```
SELECT to_geojson_geometry(to_spherical_geography(ST_GeometryFromText(
   'LINESTRING (0 0, 1 2, 3 4)')));
```

from_geojson_geometry(varchar) – Gibt das Objekt vom Typ sphärische Geografie aus der GeoJSON-Darstellung zurück und entfernt dabei Schlüssel/Werte, die nicht geometrisch sind. `Feature` und `FeatureCollection` werden nicht unterstützt.

```
SELECT
 from_geojson_geometry(to_geojson_geometry(to_spherical_geography(ST_GeometryFromText(
   'LINESTRING (0 0, 1 2, 3 4)'))));
```

geometry_nearest_points(Geometry, Geometry) – Gibt die Punkte auf jeder Geometrie zurück, die einander am nächsten liegen. Wenn eine der Geometrien leer ist, wird NULL zurückgegeben. Gibt andernfalls eine Reihe von zwei `Point`-Objekten zurück, die den Mindestabstand von zwei beliebigen Punkten auf den Geometrien aufweisen. Der erste Punkt stammt aus dem ersten Geometrie-Argument, der zweite aus dem zweiten Geometrie-Argument. Bei mehreren Paaren mit gleichem Mindestabstand wird ein Paar willkürlich gewählt.

```
SELECT geometry_nearest_points(ST_GeometryFromText(
   'LINESTRING (50 100, 50 200)'), ST_GeometryFromText(
   'LINESTRING (10 10, 20 20)'));
```

Festlegen von Digest-Funktionen

make_set_digest(x) – Fasst alle Eingabewerte von x in einem setdigest zusammen.

```
SELECT make_set_digest(value) FROM (VALUES 1, 2, 3) T(value);
```

Zeichenfolgenfunktionen

soundex(char) – Gibt eine Zeichenfolge zurück, die die phonetische Darstellung von char enthält.

```
SELECT name
FROM nation
WHERE SOUNDEX(name) = SOUNDEX('CHYNA'); -- CHINA
```

concat_ws(string0, string1,..., stringN) – Gibt die Verkettung von `string1, string2, ..., stringN` mit `string0` als Trennzeichen zurück. Wenn `string0` null ist, ist der Rückgabewert null. Alle in den Argumenten nach dem Trennzeichen angegebenen Nullwerte werden übersprungen.

```
SELECT concat_ws(',', 'def', 'pqr', 'mno');
```

Fensterfunktionen

GROUPS – Fügt Unterstützung für Fensterrahmen basierend auf Gruppen hinzu.

```
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);
```

Leistungsverbesserungen

Die Leistungsverbesserungen in Athena-Engine-Version 3 umfassen Folgendes.

- Schnellerer Metadatenabruf für AWS Glue-Tabellen – Bei Abfragen, die mehrere Tabellen umfassen, verringert sich die Zeit für die Abfrageplanung.
- Dynamische Filterung für RIGHT JOIN – Die dynamische Filterung ist jetzt für Rechtsverknüpfungen mit Equality-Join-Bedingungen aktiviert, wie im folgenden Beispiel.

  ```
  SELECT *
  FROM lineitem RIGHT JOIN tpch.tiny.supplier
  ON lineitem.suppkey = supplier.suppkey
  WHERE supplier.name = 'abc';
  ```

- Große vorbereitete Anweisungen – Die Standardgröße des HTTP-Anforderungs-/Antwort-Headers wurde auf 2 MB erhöht, um große vorbereitete Anweisungen zu ermöglichen.
- approx_percentile () – Die `approx_percentile`-Funktion verwendet nun `tdigest` anstelle von `qdigest`, um ungefähre Quantilwerte aus Verteilungen abzurufen. Dies führt zu höherer Leistung und geringerem Speicherverbrauch. Beachten Sie, dass die Funktion aufgrund dieser Änderung andere Ergebnisse zurückgibt als in Athena-Engine-Version 2. Weitere Informationen finden Sie unter Die Funktion approx_percentile gibt unterschiedliche Ergebnisse zurück.

Verbesserungen der Zuverlässigkeit

Die allgemeine Engine-Speichernutzung und -Nachverfolgung in Athena-Engine-Version 3 wurde verbessert. Große Abfragen sind weniger anfällig für Ausfälle durch Knotenabstürze.

Verbesserungen der Abfragesyntax

INTERSECT ALL – Unterstützung für `INTERSECT ALL` hinzugefügt.

```
SELECT * FROM (VALUES 1, 2, 3, 4) INTERSECT ALL SELECT * FROM (VALUES 3, 4);
```

EXCEPT ALL – Unterstützung für `EXCEPT ALL` hinzugefügt.

```
SELECT * FROM (VALUES 1, 2, 3, 4) EXCEPT ALL SELECT * FROM (VALUES 3, 4);
```

RANGE PRECEDING – Unterstützung für `RANGE PRECEDING` in Fensterfunktionen hinzugefügt.

```
SELECT sum(x) over (order by x range 1 preceding)
FROM (values (1), (1), (2), (2)) t(x);
```

MATCH_RECOGNIZE – Unterstützung für Zeilenmusterabgleich hinzugefügt, wie im folgenden Beispiel gezeigt.

```
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;
```

Verbesserungen des Datenformats und des Datentyps

Athena-Engine-Version 3 verfügt über die folgenden Verbesserungen für Datenformat und Datentyp.

- LZ4 und ZSTD – Unterstützung für das Lesen von LZ4- und ZSTD-komprimierten Parquet-Daten hinzugefügt. Unterstützung für das Schreiben von ZSTD-komprimierten ORC-Daten hinzugefügt.

- Symlink-basierten Tabellen – Unterstützung für das Erstellen von Symlink-basierten Tabellen in Avro-Dateien hinzugefügt. Ein Beispiel folgt.

```
CREATE TABLE test_avro_symlink
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
...
INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
```

- SphericalGeography – Der Typ SphericalGeography bietet native Unterstützung für räumliche Features, die auf geographischen Koordinaten dargestellt werden (manchmal auch als geodätische Koordinaten, Breitengrad/Längengrad oder Längengrad/Breitengrad bezeichnet). Geografische Koordinaten sind sphärische Koordinaten, die in Winkeleinheiten (Grad) ausgedrückt werden.

  Die `to_spherical_geography`-Funktion gibt geographische (sphärische) Koordinaten aus geometrischen (planaren) Koordinaten zurück, wie im folgenden Beispiel gezeigt.

```
SELECT to_spherical_geography(ST_GeometryFromText(
   'LINESTRING (-40.2 28.9, -40.2 31.9, -37.2 31.9)'));
```

Abwärtskompatible Änderungen

Wenn Sie von Athena-Engine-Version 2 zu Athena-Engine-Version 3 migrieren, können sich bestimmte Änderungen auf das Tabellenschema, die Syntax oder die Verwendung von Datentypen auswirken. Dieser Abschnitt listet die zugehörigen Fehlermeldungen auf und enthält Vorschläge für Problemumgehungen.

Änderungen der Abfragesyntax

IGNORE NULLS kann nicht mit Fensterfunktionen ohne Wert verwendet werden

Fehlermeldung: Für die `bool_or`-Funktion kann keine Nullbehandlungsklausel angegeben werden.

Ursache: `IGNORE NULLS` kann jetzt nur mit den Wertfunktionen `first_value`, `last_value`, `nth_value`, `lead` und `lag` verwendet werden. Diese Änderung wurde vorgenommen, um der ANSI-SQL-Spezifikation zu entsprechen.

Vorgeschlagene Lösung: Entfernen Sie `IGNORE NULLS` in Abfragezeichenfolgen von Fensterfunktionen ohne Wert.

Die CONCAT-Funktion muss über zwei oder mehr Argumente verfügen

Fehlermeldung: INVALID_FUNCTION_ARGUMENT: Es müssen zwei oder mehr Verkettungsargumente vorhanden sein

Ursache: Zuvor akzeptierte die `CONCAT`-Zeichenfolgenfunktion ein einzelnes Argument. In Athena-Engine-Version 3 erfordert die `CONCAT`-Funktion mindestens zwei Argumente.

Lösungsvorschlag: Ändern Sie das Vorkommen von `CONCAT(str)` auf `CONCAT(str, '')`.

In Athena-Engine-Version 3 können Funktionen nicht mehr als 127 Argumente haben. Weitere Informationen finden Sie unter Zu viele Argumente für den Funktionsaufruf.

Die Funktion approx_percentile gibt unterschiedliche Ergebnisse zurück

Die `approx_percentile`-Funktion liefert in Athena-Engine-Version 3 andere Ergebnisse als in Athena-Engine-Version 2.

Fehlermeldung: Keine.

Ursache: Die `approx_percentile`-Funktion unterliegt Versionsänderungen.

> **Important**
>
> Da es sich bei den Ausgaben der `approx_percentile`-Funktion um Näherungen handelt und sich die Näherungen von einer Version zur nächsten ändern können, sollten Sie sich bei kritischen Anwendungen nicht auf die `approx_percentile`-Funktion verlassen.

Vorgeschlagene Lösung: Um das Verhalten von Athena-Engine-Version 2 `approx_percentile` anzunähern, können Sie in Athena-Engine-Version 3 einen anderen Funktionsumfang verwenden. Nehmen wir zum Beispiel an, Sie haben die folgende Abfrage in Athena-Engine-Version 2:

```
SELECT approx_percentile(somecol, 2E-1)
```

Um die gleiche Ausgabe in Athena-Engine-Version 3 anzunähern, können Sie die Funktionen `value_at_quantile` und `qdigest_agg` ausprobieren, wie im folgenden Beispiel. Beachten Sie, dass das gleiche Verhalten auch mit dieser Problemumgehung nicht garantiert werden kann.

```
SELECT value_at_quantile(qdigest_agg(somecol, 1), 2E-1)
```

Die Geodatenfunktion unterstützt keine varbinary-Eingabe

Fehlermeldung: FUNCTION_NOT_FOUND für st_XXX

Ursache: Einige Geodatenfunktionen unterstützen den Legacy-Eingabetyp `VARBINARY` oder textbezogene Funktionssignaturen nicht mehr.

Lösungsvorschlag: Verwenden Sie Geodatenfunktionen, um die Eingabetypen in unterstützte Typen zu konvertieren. Unterstützte Eingabetypen sind in der Fehlermeldung angegeben.

In GROUP-BY-Klauseln müssen verschachtelte Spalten in doppelte Anführungszeichen gesetzt werden

Fehlermeldung: "*`column_name`"."*`nested_column`" muss ein Aggregatausdruck sein oder in der GROUP-BY-Klausel vorkommen

Ursache: Athena-Engine-Version 3 erfordert, dass verschachtelte Spaltennamen in `GROUP BY`-Klauseln in doppelten Anführungszeichen stehen. Beispielsweise erzeugt die folgende Abfrage den Fehler, weil sie in der `GROUP BY`-Klausel `user.name` nicht in doppelten Anführungszeichen steht.

```
SELECT "user"."name" FROM dataset
GROUP BY user.name
```

Lösungsvorschlag: Setzen Sie geschachtelte Spaltennamen in `GROUP BY`-Klauseln in doppelte Anführungszeichen, wie im folgenden Beispiel.

```
SELECT "user"."name" FROM dataset
GROUP BY "user"."name"
```

Reihenfolge der Argumente der Funktion Log()

In Athena-Engine-Version 2 lautete die Reihenfolge der Argumente für die `log()`-Funktion `log(`*`value`*`, `*`base`*`)`. In Athena-Engine-Version 3 wurde dies in `log(`*`base`*`, `*`value`*`)` geändert und entspricht nun den SQL-Standards.

Die Funktion minute() unterstützt nicht das Intervall Jahr bis Monat

Fehlermeldung: Unerwartete Parameter (Intervall von Jahr zu Monat) für die Funktion Minute. Erwartet: minute(Zeitstempel mit Zeitzone), minute(Zeit mit Zeitzone), minute(Zeitstempel), minute(Zeit), minute(Intervall Tag zu Sekunde).

Ursache: In Athena-Engine-Version 3 wurden Typprüfungen für `EXTRACT` gemäß der ANSI-SQL-Spezifikation präzisiert.

Lösungsvorschlag: Aktualisieren Sie die Abfragen, um sicherzustellen, dass Typen mit den vorgeschlagenen Funktionssignaturen übereinstimmen.

ORDER-BY-Ausdrücke müssen in der SELECT-Liste angezeigt werden

Fehlermeldung: Für SELECT DISTINCT müssen ORDER BY-Ausdrücke in der SELECT-Liste angezeigt werden

Ursache: In einer `SELECT`-Klausel wird ein falscher Tabellen-Aliasing verwendet.

Lösungsvorschlag: Vergewissern Sie sich, dass alle Spalten im `ORDER BY`-Ausdruck die richtigen Verweise in der `SELECT DISTINCT`-Klausel haben.

Abfragefehler beim Vergleich mehrerer Spalten, die von einer Unterabfrage zurückgegeben wurden

Beispiel für eine Fehlermeldung: Wertausdruck und Ergebnis der Unterabfrage müssen vom gleichen Typ sein: row(varchar, varchar) vs row(row(varchar, varchar))

Ursache: Aufgrund einer Syntaxaktualisierung in Athena-Engine-Version 3 tritt dieser Fehler auf, wenn eine Abfrage versucht, mehrere von einer Unterabfrage zurückgegebene Werte zu vergleichen, und die `SELECT`-Unterabfrage-Anweisung ihre Spaltenliste in Klammern einschließt, wie im folgenden Beispiel.

```
SELECT *
FROM table1
WHERE (t1_col1, t1_col2)
IN (SELECT (t2_col1, t2_col2) FROM table2)
```

Lösung: Entfernen Sie in Athena-Engine-Version 3 die Klammern um die Liste der Spalten in der `SELECT`-Unterabfrageanweisung, wie in der folgenden aktualisierten Beispielabfrage.

```
SELECT *
FROM table1
WHERE (t1_col1, t1_col2)
IN (SELECT t2_col1, t2_col2 FROM table2)
```

SKIP ist ein reserviertes Wort für DML-Abfragen

Das Wort `SKIP` ist jetzt ein reserviertes Wort für DML-Abfragen wie `SELECT`. Um `SKIP` als Bezeichner in einer DML-Abfrage zu verwenden, schließen Sie es in doppelte Anführungszeichen ein.

Weitere Informationen zu reservierten Wörtern in Athena finden Sie unter Reservierte Schlüsselwörter.

Die Klauseln SYSTEM_TIME und SYSTEM_VERSION sind für Zeitreisen veraltet

Fehlermeldung: nicht übereinstimmende Eingabe 'SYSTEM_TIME'. Erwartet: 'TIMESTAMP', 'VERSION'

Ursache: In Athena-Engine-Version2 verwendeten Iceberg-Tabellen die `FOR SYSTEM_TIME AS OF`- und `FOR SYSTEM_VERSION AS OF`-Klauseln für Zeitstempel- und Versionszeitreisen. Athena-Engine-Version 3 verwendet die `FOR TIMESTAMP AS OF`- und `FOR VERSION AS OF`-Klauseln.

Lösungsvorschlag: Aktualisieren Sie die SQL-Abfrage, um die `TIMESTAMP AS OF`- und `VERSION AS OF`-Klauseln für Zeitreiseoperationen zu verwenden, wie in den folgenden Beispielen.

Zeitreise nach Zeitstempel:

```
SELECT * FROM TABLE FOR TIMESTAMP AS OF (current_timestamp - interval '1' day)
```

Zeitreise nach Version:

```
SELECT * FROM TABLE FOR VERSION AS OF 949530903748831860
```

Zu viele Argumente für einen Array-Konstruktor

Fehlermeldung: TOO_MANY_ARGUMENTS: Zu viele Argumente für den Array-Konstruktor.

Ursache: Die maximale Anzahl von Elementen in einem Array-Konstruktor ist jetzt auf 254 festgelegt.

Vorgeschlagene Lösung: Teilen Sie die Elemente in mehrere Arrays auf, die jeweils 254 oder weniger Elemente enthalten, und verwenden Sie die `CONCAT`-Funktion, um die Arrays zu verketten, wie im folgenden Beispiel gezeigt.

```
CONCAT(
ARRAY[x1,x2,x3...x254],
```

```
ARRAY[y1,y2,y3...y254],
...
)
```

Bezeichner mit Null-Längenbegrenzung ist nicht zulässig

Fehlermeldung: Durch Null getrennte Kennung nicht zulässig.

Ursache: Eine Abfrage verwendete eine leere Zeichenfolge als Spalten-Alias.

Lösungsvorschlag: Aktualisieren Sie die Abfrage so, dass für die Spalte ein nicht leerer Alias verwendet wird.

Änderungen der Datenverarbeitung

Bucket-Validierung

Fehlermeldung: HIVE_INVALID_BUCKET_FILES: Die Hive-Tabelle ist beschädigt.

Ursache: Die Tabelle ist möglicherweise beschädigt. Um die Richtigkeit der Abfragen für Bucket-Tabellen sicherzustellen, ermöglicht die Athena-Engine-Version 3 eine zusätzliche Validierung von Bucket-Tabellen, um die Richtigkeit der Abfragen sicherzustellen und unerwartete Fehler zur Laufzeit zu vermeiden.

Vorgeschlagene Lösung: Erstellen Sie die Tabelle mit Athena-Engine-Version 3 neu.

Das Umwandeln einer Struktur in JSON gibt jetzt Feldnamen zurück

Wenn Sie in Athena-Engine-Version 3 in einer `SELECT`-Abfrage ein `struct` in JSON umwandeln, gibt die Umwandlung jetzt sowohl die Feldnamen als auch die Werte (z. B. `useragent":null`) statt nur die Werte (z. B. `null`) zurück.

Änderung der Sicherheitserzwingung auf Spaltenebene der Iceberg-Tabelle

Fehlermeldung: Zugriff verweigert: Kann nicht aus Spalten ausgewählt werden

Ursache: Die Iceberg-Tabelle wurde außerhalb von Athena erstellt und verwendet eine Apache Iceberg SDK-Version vor 0.13.0. Da frühere SDK-Versionen keine Spalten in AWS Glue füllen, konnte Lake Formation die für den Zugriff autorisierten Spalten nicht ermitteln.

Lösungsvorschlag: Führen Sie ein Update mit der FESTGELEGTE TABELLENEIGENSCHAFTEN ÄNDERN-Anweisung von Athena durch oder verwenden Sie das neueste Iceberg-SDK, um die Tabelle zu reparieren und die Spalteninformationen in AWS Glue zu aktualisieren.

Nullen in Listendatentypen werden jetzt an UDFs weitergegeben

Fehlermeldung: Null-Zeiger-Ausnahme

Ursache: Dieses Problem kann Sie betreffen, wenn Sie den UDF-Konnektor verwenden und eine benutzerdefinierte Lambda-Funktion implementiert haben.

Athena-Engine-Version 2 hat die Nullen in Listendatentypen herausgefiltert, die an eine benutzerdefinierte Funktion übergeben wurden. In Athena-Engine-Version 3 werden die Nullen jetzt beibehalten und an die UDF weitergegeben. Dies kann zu einer Null-Zeiger-Ausnahme führen, wenn die UDF versucht, das Null-Element ohne Überprüfung zu dereferenzieren.

Wenn sich beispielsweise die Daten `[null, 1, null, 2, 3, 4]` in einer Ursprungsdatenquelle wie DynamoDB befinden, werden die folgenden Daten an die benutzerdefinierte Lambda-Funktion übergeben:

Athena-Engine-Version 2: `[1,2,3,4]`

Athena-Engine-Version 3: `[null, 1, null, 2, 3, 4]`

Lösungsvorschlag: Stellen Sie sicher, dass Ihre benutzerdefinierte Lambda-Funktion Nullelemente in Listendatentypen verarbeitet.

Teilzeichenfolgen aus Zeichen-Arrays enthalten keine aufgefüllten Leerzeichen mehr

Fehlermeldung: Es wird kein Fehler ausgegeben, aber die zurückgegebene Zeichenfolge enthält keine aufgefüllten Leerzeichen mehr. Beispielsweise gibt `substr(char[20],1,100)` jetzt eine Zeichenfolge mit der Länge 20 statt 100 zurück.

Lösungsvorschlag: Es sind keine Maßnahmen erforderlich.

Nicht unterstützte Umwandlung des Dezimalspaltentyps

Fehlermeldungen: HIVE_CURSOR_ERROR: Parquet-Datei konnte nicht gelesen werden: s3://*DOC-EXAMPLE-BUCKET*/*path*/*file_name*.parquet oder Nicht unterstützter Spaltentyp (varchar) für die Parquet-Spalte (*[column_name*]

Ursache: Athena-Engine-Version 2 war gelegentlich erfolgreich (schlug aber häufig fehl), wenn versucht wurde, Datentypzwänge von `varchar` zu dezimal zu ändern. Da die Athena-Engine-Version 3 über eine Typüberprüfung verfügt, die die Kompatibilität des Typs prüft, bevor sie versucht, den Wert zu lesen, schlagen solche Zwangsverknüpfungen jetzt immer fehl.

Vorgeschlagene Lösung: Ändern Sie Ihr Schema in AWS Glue sowohl für Athena-Engine-Version 2 als auch für Athena-Engine-Version 3 so, dass es einen numerischen Datentyp anstelle von `varchar` für Dezimalspalten in Parquet-Dateien verwendet. Durchforsten Sie entweder die Daten erneut und stellen Sie sicher, dass der neue Spaltendatentyp ein Dezimaltyp ist, oder erstellen Sie die Tabelle in Athena manuell neu und verwenden Sie die Syntax `decimal(precision, scale)`, um einen decimal-Datentyp für die Spalte anzugeben.

Float- oder Double-NaN-Werte können nicht mehr in bigint umgewandelt werden

Fehlermeldung: INVALID_CAST_ARGUMENT: Real/Double NaN kann nicht in bigint umgewandelt werden

Ursache: `NaN` kann in Athena-Engine-Version 3 nicht mehr auf 0 als `bigint` gecastet werden als.

Lösungsvorschlag: Stellen Sie sicher, dass beim Umwandeln in `bigint` keine `NaN`-Werte in den Spalten `double` oder `float` vorhanden sind.

Änderung des Rückgabetyps der Funktion uuid ()

Das folgende Problem betrifft sowohl Tabellen als auch Ansichten.

Fehlermeldung: Hive-Typ: uuid wird nicht unterstützt

Ursache: In Athena-Engine-Version 2 gab die `uuid()`-Funktion eine Zeichenfolge zurück, aber in Athena-Engine-Version 3 gibt sie eine zufällig generierte Pseudo-UUID (Typ 4) zurück. Da der UUID-Spaltendatentyp in Athena nicht unterstützt wird, kann die `uuid()`-Funktion in Athena-Engine-Version 3 nicht mehr direkt in CTAS-Abfragen zum Generieren von UUID-Spalten verwendet werden.

Die folgende `CREATE TABLE`-Anweisung wird beispielsweise in Athena-Engine-Version 2 erfolgreich abgeschlossen, gibt aber NOT_SUPPORTED: Nicht unterstützter Hive-Type: uuid in Athena-Engine-Version 3:

```
CREATE TABLE uuid_table AS
   SELECT uuid() AS myuuid
```

In ähnlicher Weise wird die folgende Anweisung `CREATE VIEW` in Athena-Engine-Version 2 erfolgreich abgeschlossen, gibt aber den ungültigen Spaltentyp für die Spalte myuuid zurück: Nicht unterstützter Hive-Typ: uuid in Athena-Engine-Version 3:

```
CREATE VIEW uuid_view AS
   SELECT uuid() AS myuuid
```

Wenn eine so in Athena-Engine-Version 2 erstellte Ansicht in Athena-Engine-Version 3 abgefragt wird, tritt ein Fehler wie der folgende auf:

VIEW_IS_STALE: Zeile 1:15: Die Ansicht 'awsdatacatalog.mydatabase.uuid_view' ist veraltet oder befindet sich in einem ungültigen Zustand: Die Spalte [myuuid] vom Typ uuid, die von der Abfrageansicht an Position 0 projiziert wird, kann nicht in die Spalte [myuuid] vom Typ varchar, die in der Ansichtsdefinition gespeichert ist, umgewandelt werden

Vorgeschlagene Lösung: Verwenden Sie beim Erstellen der Tabelle oder Ansicht die `cast()`-Funktion, um die Ausgabe von `uuid()` in ein `varchar` zu konvertieren, wie in den folgenden Beispielen:

```
CREATE TABLE uuid_table AS
   SELECT CAST(uuid() AS VARCHAR) AS myuuid
```

```
CREATE VIEW uuid_view AS
   SELECT CAST(uuid() AS VARCHAR) AS myuuid
```

Probleme mit Zwangsmaßnahmen bei CHAR und VARCHAR

Verwenden Sie die Problemumgehungen in diesem Abschnitt, wenn Sie mit `varchar` und `char` in der Athena-Engine-Version 3 auf Zwangsprobleme stoßen. Wenn Sie diese Problemumgehungen nicht nutzen können, wenden Sie sich bitte an AWS Support.

CONCAT-Funktionsfehler bei gemischten CHAR- und VARCHAR-Eingaben

Problem: Die folgende Abfrage ist auf Athena-Engine-Version 2 erfolgreich.

```
SELECT concat(CAST('abc' AS VARCHAR(20)), '12', CAST('a' AS CHAR(1)))
```

In der Athena-Engine-Version 3 schlägt dieselbe Abfrage jedoch wie folgt fehl:

Fehlermeldung: FUNCTION_NOT_FOUND: Zeile 1:8: Unerwartete Parameter (varchar(20), varchar(2), char(1)) für die Funktion concat. Erwartet: concat (char (x), char (y)), concat (array (E), E) E, concat (E, array (E)) E, concat (array (E)) E, concat (array (E)) E, concat (varchar), concat (varbinary)

Vorgeschlagene Lösung: Wenn Sie die `concat`-Funktion verwenden, konvertieren Sie entweder in `char` oder `varchar`, aber nicht in eine Mischung aus beidem.

SQL || Fehler bei der Verkettung mit CHAR- und VARCHAR-Eingaben

In der Athena-Engine-Version 3 erfordert der `||`-Operator für die Verkettung von doppelten vertikalen Balken `varchar` als Eingaben. Bei den Eingaben darf es sich nicht um eine Kombination von `varchar`- und `char`-Typen handeln.

Fehlermeldung: TYPE_NOT_FOUND: Zeile 1:26: Unbekannter Typ: char(65537)

Ursache: Eine Abfrage, die `||` verwendet, um ein `char` und ein `varchar` zu verketten, kann zu dem Fehler führen, wie im folgenden Beispiel.

```
SELECT CAST('a' AS CHAR) || CAST('b' AS VARCHAR)
```

Vorgeschlagene Lösung: Verketten Sie `varchar` mit `varchar`, wie im folgenden Beispiel.

```
SELECT CAST('a' AS VARCHAR) || CAST('b' AS VARCHAR)
```

Fehler bei der CHAR- und VARCHAR-UNION-Abfrage

Fehlermeldung: NOT_SUPPORTED: Hive-Typ wird nicht unterstützt: char(65536). Unterstützte CHAR-Typen: CHAR(<=255)

Ursache: Eine Abfrage, die versucht, `char` und `varchar` zu kombinieren, wie im folgenden Beispiel:

```
CREATE TABLE t1 (c1) AS SELECT CAST('a' as CHAR) as c1 UNION ALL SELECT CAST('b' AS
 VARCHAR) AS c1
```

Vorgeschlagene Lösung: Verwenden Sie in der Beispielabfrage die `'a'`-Umwandlung zu `varchar` und nicht `char`.

Unerwünschte Leerzeichen nach CHAR- oder VARCHAR-Zwang

In Athena-Engine-Version 3 ist der Zieltyp, wenn `char(X)`- und `varchar`-Daten bei der Bildung eines Arrays oder einer einzelnen Spalte zu einem einzigen Typ gezwungen werden `char(65535)` und jedes Feld enthält viele unerwünschte Leerzeichen am Ende.

Ursache: Athena-Engine-Version 3 wandelt `varchar` und `char(X)` zu `char(65535)` und füllt die Daten dann nach rechts mit Leerzeichen auf.

Vorgeschlagene Lösung: Wandeln Sie jedes Feld explizit in `varchar` um.

Änderungen des Zeitstempels

Umwandlung eines Zeitstempels mit Zeitzone auf Varchar-Verhaltensänderung

In der Athena-Engine-Version 2 führte das Umwandeln von `Timestamp` mit Zeitzone in `varchar` dazu, dass sich einige Zeitzonenliterale änderten (z. B. wurde `US/Eastern` in `America/New_York` geändert). Dieses Problem tritt in Athena-Engine-Version 3 nicht auf.

Überlauf des Datums-Zeitstempels löst Fehler aus

Fehlermeldung: Millis-Überlauf: XXX

Ursache: Da ISO 8601-Datumsangaben in Athena-Engine-Version 2 nicht auf Überlauf geprüft wurden, erzeugten einige Daten einen negativen Zeitstempel. Athena-Engine-Version 3 prüft auf diesen Überlauf und löst eine Ausnahme aus.

Lösungsvorschlag: Stellen Sie sicher, dass der Zeitstempel innerhalb des Bereichs liegt.

Politische Zeitzonen mit TIME werden nicht unterstützt

Fehlermeldung: INVALID LITERAL

Ursache: Abfragen wie `SELECT TIME '13:21:32.424 America/Los_Angeles'`.

Lösungsvorschlag: Vermeiden Sie die Verwendung politischer Zeitzonen mit `TIME`.

Genauigkeitsabweichung in Timestamp-Spalten verursacht Serialisierungsfehler

Fehlermeldung: SERIALIZATION_ERROR: Spalte '*COLUMNZ*' vom Typ ‚timestamp(3)' an Position *X*:*Y* konnte nicht serialisiert werden

COLUMNZ ist der Ausgabename der Spalte, die das Problem verursacht. Die Zahlen *X*:*Y* geben die Position der Spalte in der Ausgabe an.

Ursache: Athena-Engine-Version 3 prüft, ob die Genauigkeit der Zeitstempel in den Daten mit der Genauigkeit übereinstimmt, die für den Spaltendatentyp in der Tabellenspezifikation angegeben ist. Derzeit liegt diese Genauigkeit immer bei 3. Wenn die Daten eine höhere Genauigkeit aufweisen, schlagen Abfragen mit dem angegebenen Fehler fehl.

Lösungsvorschlag: Überprüfen Sie Ihre Daten, um sicherzustellen, dass Ihre Zeitstempel auf die Millisekunde genau sind.

Falsche Zeitstempelgenauigkeit in UNLOAD- und CTAS-Abfragen für Iceberg-Tabellen

Fehlermeldung: Falsche Zeitstempelgenauigkeit für Zeitstempel(6); die konfigurierte Genauigkeit ist MILLISEKUNDEN

Ursache: Athena-Engine-Version 3 prüft, ob die Genauigkeit der Zeitstempel in den Daten mit der Genauigkeit übereinstimmt, die für den Spaltendatentyp in der Tabellenspezifikation angegeben ist. Derzeit liegt diese Genauigkeit immer bei 3. Wenn die Daten eine höhere Genauigkeit haben (z. B. Mikrosekunden statt Millisekunden), können Abfragen mit dem angegebenen Fehler fehlschlagen.

Lösung: Um dieses Problem zu umgehen, `CAST` Sie zunächst die Genauigkeit des Zeitstempels auf 6 ein, wie im folgenden CTAS-Beispiel, das eine Iceberg-Tabelle erstellt. Beachten Sie, dass die Genauigkeit mit 6 statt mit 3 angegeben werden muss, um den Fehler Zeitstempelpräzision (3) nicht unterstützt für Iceberg zu vermeiden.

```
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
```

Da Athena Zeitstempel 6 nicht unterstützt, wandeln Sie den Wert dann erneut in Zeitstempel um (z. B. in einer Ansicht). Das folgende Beispiel erstellt eine Ansicht aus der `my_iceberg_ctas`-Tabelle.

```
CREATE OR REPLACE VIEW my_iceberg_ctas_view AS
SELECT cast(dt AS timestamp) AS dt
FROM my_iceberg_ctas
```

Das Lesen des Long-Typs als Zeitstempel in ORC-Dateien verursacht jetzt einen Fehler „fehlerhafte-ORC-Datei“

Fehlermeldung: Fehler beim Öffnen von Hive Split ‘FILE (SPLIT POSITION)’ Fehlerhafte ORC-Datei. Der Zeitstempel des SQL-Typs kann nicht aus dem ORC-Stream .long_type vom Typ LONG gelesen werden

Ursache: Athena-Engine-Version 3 lehnt nun impliziten Zwang vom `Long`-Datentyp zu `Timestamp` oder von `Timestamp` zu `Long` ab. Zuvor wurden `Long`-Werte implizit in Zeitstempel umgewandelt, als wären sie Epochen-Millisekunden.

Lösungsvorschlag: Verwenden Sie die `from_unixtime`-Funktion, um die Spalte explizit umzuwandeln, oder verwenden Sie die `from_unixtime`-Funktion, um eine zusätzliche Spalte für zukünftige Abfragen zu erstellen.

Zeit und Intervall von Jahr zu Monat werden nicht unterstützt

Fehlermeldung: TYPE MISMATCH

Ursache: Athena-Engine-Version 3 unterstützt Zeit und Intervall von Jahr zu Monat nicht (z. B. `SELECT TIME '01:00' + INTERVAL '3' MONTH`).

Zeitstempelüberlauf für das int96-Parquet-Format

Fehlermeldung: Ungültige timeOfDayNanos

Ursache: Ein Zeitstempelüberlauf für das `int96`-Parquet-Format.

Lösungsvorschlag: Identifizieren Sie die spezifischen Dateien, bei denen das Problem auftritt. Generieren Sie dann die Datendatei erneut mit einer aktuellen, bekannten Parquet-Bibliothek oder verwenden Sie Athena CTAS. Wenn das Problem weiterhin besteht, wenden Sie sich an den Athena-Support und teilen Sie uns mit, wie die Datendateien generiert werden.

Bei der Umwandlung von einer Zeichenfolge in einen Zeitstempel ist Platz zwischen Datums- und Uhrzeitwerten erforderlich

Fehlermeldung: INVALID_CAST_ARGUMENT: Der Wert kann nicht in einen Zeitstempel umgewandelt werden.

Ursache: Athena-Engine-Version 3 akzeptiert keinen Bindestrich mehr als gültiges Trennzeichen zwischen Datums- und Uhrzeitwerten in der Eingabezeichenfolge zu `cast`. Die folgende Abfrage funktioniert beispielsweise in Athena-Engine-Version 2, aber nicht in Athena-Engine-Version 3:

```
SELECT CAST('2021-06-06-23:38:46' AS timestamp) AS this_time
```

Vorgeschlagene Lösung: Ersetzen Sie in Athena-Engine-Version 3 den Bindestrich zwischen Datum und Uhrzeit durch ein Leerzeichen, wie im folgenden Beispiel.

```
SELECT CAST('2021-06-06 23:38:46' AS timestamp) AS this_time
```

Änderung des Zeitstempel-Rückgabewerts von to_iso8601 ()

Fehlermeldung: Keine.

Ursache: In Athena-Engine-Version 2 gibt die `to_iso8601`-Funktion einen Zeitstempel mit Zeitzone zurück, auch wenn der an die Funktion übergebene Wert die Zeitzone nicht enthält. In Athena-Engine-Version 3 gibt die `to_iso8601`-Funktion nur dann einen Zeitstempel mit Zeitzone zurück, wenn das übergebene Argument die Zeitzone enthält.

Die folgende Abfrage übergibt beispielsweise das aktuelle Datum zweimal an die `to_iso8601`-Funktion: zuerst als Zeitstempel mit Zeitzone und dann als Zeitstempel.

```
SELECT TO_ISO8601(CAST(CURRENT_DATE AS TIMESTAMP WITH TIME ZONE)),
 TO_ISO8601(CAST(CURRENT_DATE AS TIMESTAMP))
```

Die folgende Ausgabe zeigt das Ergebnis der Abfrage in jeder Engine.

Athena-Engine-Version 2:

#	_col0	_col1
1	2023-02-24T00:00:00.000Z	2023-02-24T00:00:00.000Z

Athena-Engine-Version 3

#	_col0	_col1
1	2023-02-24T00:00:00.000Z	2023-02-24T00:00:00.000

Lösungsvorschlag: Um das vorherige Verhalten zu replizieren, können Sie den Zeitstempelwert an die `with_timezone`-Funktion übergeben, bevor Sie ihn an `to_iso8601` übergeben, wie im folgenden Beispiel:

```
SELECT to_iso8601(with_timezone(TIMESTAMP '2023-01-01 00:00:00.000', 'UTC'))
```

Ergebnis

#	_col0
1	2023-01-01T00:00:00.000Z

Der erste Parameter at_timezone () muss ein Datum angeben

Problem: In Athena-Engine-Version 3 kann die `at_timezone`-Funktion keinen `time_with_timezone`-Wert als ersten Parameter annehmen.

Ursache: Ohne Datumsinformationen kann nicht festgestellt werden, ob es sich bei dem übergebenen Wert um Sommerzeit oder Normalzeit handelt. `at_timezone('12:00:00 UTC', 'America/Los_Angeles')` ist beispielsweise mehrdeutig, da nicht bestimmt werden kann, ob es sich bei dem übergebenen Wert um Pacific Daylight Time (PDT) oder Pacific Standard Time (PST) handelt.

Einschränkungen

Athena-Engine-Version 3 weist die folgenden Einschränkungen auf.

- Abfrageleistung – Viele Abfragen werden mit der Athena-Engine-Version 3 schneller ausgeführt, aber einige Abfragepläne können sich von denen der Athena-Engine-Version 2 unterscheiden. Daher können sich einige Abfragen in Bezug auf Latenz oder Kosten unterscheiden.
- Trino- und Presto-Konnektoren – Weder Trino- noch Presto-Konnektoren werden unterstützt. Sie können mit Amazon Athena Federated Query verbinden. Weitere Informationen finden Sie unter Nutzung von Amazon-Athena-Verbundabfrage.
- Fehlertolerante Ausführung – Die fehlertolerante Ausführung von Trino (Trino Tardigrade) wird nicht unterstützt.
- Grenzwert für Funktionsparameter – Funktionen können nicht mehr als 127 Parameter annehmen. Weitere Informationen finden Sie unter Zu viele Argumente für den Funktionsaufruf.

Die folgenden Grenzwerte wurden in Athena-Engine-Version 2 eingeführt, um sicherzustellen, dass Abfragen aufgrund von Ressourcenbeschränkungen nicht fehlschlagen. Diese Grenzwerte sind von Benutzern nicht konfigurierbar.

- Anzahl der Ergebniselemente – Die Anzahl der Ergebniselemente n ist für die folgenden Funktionen auf 10.000 oder weniger beschränkt: `min(col, n)`, `max(col, n)`, `min_by(col1, col2, n)` und `max_by(col1, col2, n)`.
- GROUPING SETS – Die maximale Anzahl von Slices in einem Gruppierungssatz ist 2048.
- Maximale Zeilenlänge der Textdatei – Die standardmäßige maximale Zeilenlänge für Textdateien beträgt 200 MB.
- Sequenzfunktion maximale Ergebnisgröße – Die maximale Ergebnisgröße einer Sequenzfunktion beträgt 50000 Einträge. Zum Beispiel ist `SELECT sequence(0,45000,1)` erfolgreich, aber

`SELECT sequence(0,55000,1)` schlägt mit der Fehlermeldung fehl. Das Ergebnis der Sequenzfunktion darf nicht mehr als 50000 Einträge haben. Diese Beschränkung gilt für alle Eingabetypen für Sequenzfunktionen, auch für Zeitstempel.

Athena-Engine-Version 2

Athena-Engine-Version 2 führte die folgenden Änderungen ein.

- Neue Features und Verbesserungen
  - Verbesserungen bei Gruppierung, Verknüpfung und Unterabfrage
  - Erweiterungen von Datentypen
  - Erweiterte Funktionen
  - Leistungsverbesserungen
  - JSON-bezogene Verbesserungen
- Abwärtskompatible Änderungen
  - Fehlerbehebungen
  - Änderungen an Geodatenfunktionen
  - ANSI-SQL-Compliance
  - Ersetzte Funktionen
  - Einschränkungen

Neue Features und Verbesserungen

- EXPLAIN und EXPLAIN ANALYZE – Sie können jetzt die `EXPLAIN`-Anweisung in Athena verwenden, um die Ausführungspläne für Ihre SQL-Abfragen anzuzeigen. Verwenden Sie `EXPLAIN ANALYZE`, um den verteilten Ausführungsplan für Ihre SQL-Abfragen und die Kosten für jeden Vorgang anzuzeigen. Weitere Informationen finden Sie unter Verwenden von EXPLAIN und EXPLAIN ANALYZE in Athena.
- Verbundabfragen – Verbundabfragen werden in Athena-Engine-Version 2 unterstützt. Weitere Informationen finden Sie unter Nutzung von Amazon-Athena-Verbundabfrage.
- Räumliche Funktionen – Mehr als 25 räumliche Funktionen wurden hinzugefügt. Weitere Informationen finden Sie unter Neue Geodaten-Funktionen in Athena-Engine-Version 2.
- Verschachteltes Schema – Support für das Lesen verschachtelter Schemas hinzugefügt, wodurch die Kosten gesenkt werden.

- Vorbereitete Anweisungen – Verwenden Sie vorbereitete Anweisungen für die wiederholte Ausführung derselben Abfrage mit unterschiedlichen Abfrageparametern. Eine vorbereitete Anweisung enthält Platzhalterparameter, deren Werte Sie zur Laufzeit übergeben. Vorbereitete Anweisungen verhindern SQL-Injections-Angriffe. Weitere Informationen finden Sie unter Verwenden von parametrisierten Abfragen.
- Support der Schema-Evolution – Support für Schema-Evolution wurde hinzugefügt, um Daten im Parquet-Format zu unterstützen.
  - Support für das Lesen von Array-, Map- oder Zeilentypspalten von Partitionen, bei denen sich das Partitionsschema vom Tabellenschema unterscheidet, wurde hinzugefügt. Dies kann auftreten, wenn das Tabellenschema nach dem Erstellen der Partition aktualisiert wurde. Die geänderten Spaltentypen müssen kompatibel sein. Bei Zeilentypen können nachfolgende Felder hinzugefügt oder gelöscht werden, aber die entsprechenden Felder (nach Ordinalzahl) müssen denselben Namen haben.
  - ORC-Dateien können nun Strukturspalten mit fehlenden Feldern enthalten. Dadurch kann das Tabellenschema geändert werden, ohne die ORC-Dateien neu zu schreiben.
  - ORC-Strukturspalten werden nun nach Namen und nicht nach Ordinalnamen zugeordnet. Dadurch werden fehlende oder zusätzliche Strukturfelder in der ORC-Datei korrekt behandelt.
- SQL-OFFSET – Die SQL-`OFFSET`-Klausel wird jetzt in `SELECT`-Anweisungen unterstützt. Weitere Informationen finden Sie unter SELECT.
- UNLOAD-Anweisung – Sie können die `UNLOAD`-Anweisung verwenden, um die Ausgabe einer `SELECT`-Abfrage in die Formate PARQUET, ORC, AVRO und JSON zu schreiben. Weitere Informationen finden Sie unter UNLOAD.

Verbesserungen bei Gruppierung, Verknüpfung und Unterabfrage

- Komplexe Gruppierung – Support für komplexe Gruppierungsoperationen wurde hinzugefügt.
- Korrelierte Unterabfragen – Support für korrelierte Unterabfragen in `IN`-Prädikaten und für korrelierte Unterabfragen, die Zwang erfordern, wurde hinzugefügt.
- CROSS-JOIN – Unterstützung für `CROSS JOIN` gegen `LATERAL` abgeleitete Tabellen hinzugefügt.
- GROUPING SETS – Unterstützung für `ORDER BY`-Klauseln in Aggregationen für Abfragen hinzugefügt, die `GROUPING SETS` verwenden.
- Lambda Ausdrücke – Unterstützung für die Dereferenzierung von Zeilenfeldern in Lambda-Ausdrücken hinzugefügt.

- Null-Werte in Semijoins – Unterstützung für Nullwerte auf der linken Seite eines Semijoin hinzugefügt (dh ein `IN`-Prädikat mit Unterabfragen).
- Räumliche Joins – Unterstützung für Räumliche Broadcast-Joins und Räumliche linke Joins hinzugefügt.
- Auf Festplatte auslagern – Für speicherintensive Vorgänge `INNER JOIN`- und `LEFT JOIN`-Operationen lagert Athena Zwischenergebnisse auf die Festplatte aus. Dies ermöglicht die Ausführung von Abfragen, die große Mengen an Speicher benötigen.

Erweiterungen von Datentypen

- INT für INTEGER – Unterstützung für `INT` hinzugefügt als Alias für den `INTEGER`-Datentyp.
- Intervall-Typen – Unterstützung für das Umwandeln in `INTERVAL`-Typen hinzugefügt.
- IPADDRESS – Es wurde ein neuer `IPADDRESS` Typ hinzugefügt, um IP-Adressen in DML-Abfragen darzustellen. Unterstützung für die Umwandlung zwischen `VARBINARY`-Typ und `IPADDRESS`-Typ Der `IPADDRESS` Typ wird in DDL-Abfragen nicht erkannt.
- IS DISTINCT FROM – `IS DISTINCT FROM`-Unterstützung für die `JSON`- und `IPADDRESS`-Typen hinzugefügt.
- Nullgleichsprüfungen – Gleichheitsüberprüfungen auf Nullwerte in `ARRAY`-, `MAP`- und `ROW`-Datenstrukturen werden jetzt unterstützt. Der Ausdruck `ARRAY ['1', '3', null] = ARRAY ['1', '2', null]` gibt beispielsweise `false` zurück. Zuvor gab ein null-Element die Fehlernachricht Vergleich nicht unterstützt zurück.
- Zeilentyp-Zwang – Zwang zwischen Zeilentypen unabhängig von Feldnamen ist jetzt erlaubt. Bisher war ein Zeilentyp nur dann auf einen anderen umsetzbar, wenn der Feldname im Quelltyp mit dem Zieltyp übereinstimmte oder wenn der Zieltyp einen anonymen Feldnamen hatte.
- Zeitsubtraktion – Implementiert Subtraktion für alle `TIME`- und `TIMESTAMP`-Typen.
- Unicode – Unterstützung für Escape-Unicode-Sequenzen in Zeichenfolgen-Literalen hinzugefügt.
- Verkettung von VARBINARY – Unterstützung für die Verkettung von `VARBINARY`-Werten hinzugefügt.

  Fensterwertfunktionen – Fensterwertfunktionen unterstützen jetzt `IGNORE NULLS` und `RESPECT NULLS`.

Zusätzliche Eingabetypen für Funktionen

Die folgenden Funktionen akzeptieren jetzt zusätzliche Eingabetypen. Weitere Informationen zu jeder Funktion finden Sie unter dem entsprechenden Link zur Presto-Dokumentation.

- approx_distinct () – Die approx_distinct () unterstützt jetzt die folgenden Typen: `INTEGER`, `SMALLINT`, `TINYINT`, `DECIMAL`, `REAL`, `DATE`, `TIMESTAMP`, `TIMESTAMP WITH TIME ZONE`, `TIME`, `TIME WITH TIME ZONE`, `IPADDRESS` und `CHAR`.
- Avg(), sum() – Die avg()- und sum()-Aggregatfunktionen unterstützen nun den `INTERVAL`-Datentyp.
- Lpad(), rpad() – Die lpad- und rpad-Funktionen funktionieren jetzt auf `VARBINARY`-Eingaben.
- Min(), max() – Die min()- und max()-Aggregationsfunktionen erlauben nun unbekannte Eingabetypen zur Abfrageanalyse, sodass Sie die Funktionen mit `NULL`-Literalen verwenden können.
- regexp_replace () – Variante der regexp_replace ()-Funktion hinzugefügt, die eine Lambda-Funktion für jede Ersetzung ausführen kann.
- Sequence() – `DATE`-Varianten für die sequence()-Funktion hinzugefügt, einschließlich Variante mit einer impliziten Schrittweite von einem Tag.
- ST_Area() – Die Geodatenfunktion ST_Area() unterstützt jetzt alle Geometrietypen.
- Substr() – Die substr()-Funktion funktioniert jetzt auf `VARBINARY`-Eingaben.
- zip_with() – Arrays mit nicht übereinstimmender Länge können jetzt mit zip_with() verwendet werden. Fehlende Positionen werden mit null gefüllt. Zuvor wurde ein Fehler ausgelöst, wenn Arrays unterschiedlicher Länge übergeben wurden. Diese Änderung kann es schwierig machen, zwischen Werten, die ursprünglich null waren, von Werten zu unterscheiden, die hinzugefügt wurden, um die Arrays auf die gleiche Länge aufzufüllen.

Erweiterte Funktionen

Die folgende Liste enthält Funktionen, die ab Version 2 der Athena-Engine neu sind. Die Liste enthält keine räumlichen Funktionen. Eine Liste der Geodatenfunktionen finden Sie unter Neue Geodaten-Funktionen in Athena-Engine-Version 2.

Weitere Informationen zu jeder Funktion finden Sie unter dem entsprechenden Link zur Presto-Dokumentation.

Aggregationsfunktionen

reduce_agg()

Array-Funktionen und -Operatoren

array_sort () – Variante dieser Funktion hinzugefügt, die eine Lambda-Funktion als Komparator nimmt.

ngrams()

Binäre Funktionen und Operatoren

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()

to_ieee754_32()

to_ieee754_64()

Datums- und Zeitfunktionen und -Operatoren

millisecond()

parse_duration()

to_milliseconds()

Zuordnungs-Funktionen und -Operatoren

multimap_from_entries()

Mathematische Funktionen und Operatoren

inverse_normal_cdf()

wilson_interval_lower()

wilson_interval_upper()

Quantildigest-Funktionen

Quantildigest-Funktionen und `qdigest`-Quantil-Digest-Typ hinzugefügt.

Zeichenfolgen-Funktionen und -Operatoren

hamming_distance()

split_to_multimap()

Leistungsverbesserungen

Die Leistung der folgenden Features hat sich in Athena-Engine-Version 2 verbessert.

Abfrageleistung

- Broadcast-Beitrittleistung – Verbesserte Broadcast-Beitrittleistung durch Anwenden des dynamischen Partitionsschnitts im Worker-Knoten.
- Tabellen mit Buckets – Verbesserte Leistung beim Schreiben in Bucket-Tabellen, wenn die zu schreibenden Daten bereits entsprechend partitioniert sind (z. B. wenn die Ausgabe von einem Bucket-Boot stammt).
- DISTINCT – Verbesserte Leistung für einige Abfragen, die `DISTINCT` verwenden.

  Dynamische Filterung und Partitionsbereinigung – Verbesserungen erhöhen die Leistung und reduzieren die bei Abfragen gescannte Datenmenge.
- Filter- und Projektionsoperationen – Filter- und Projektionsoperationen werden nun immer nach Möglichkeit von Spalten verarbeitet. Die Engine nutzt automatisch Wörterbuchkodierungen, wo sie wirksam sind.

- Sammelbörsen – Verbesserte Leistung für Abfragen mit Sammelbörsen.
- Globale Aggregationen – Verbesserte Leistung für einige Abfragen, die gefilterte globale Aggregationen durchführen.
- GROUPING SETS, CUBE, ROLLUP – Verbesserte Leistung für Abfragen mit `GROUPING SETS`, `CUBE` oder `ROLLUP`, mit dem Sie mehrere Sätze von Spalten in einer einzelnen Abfrage aggregieren können.
- Sehr selektive Filter – Die Leistung von Anfragen mit hochselektiven Filtern wurde verbessert.
- Operationen JOIN und AGREGATE – Die Leistung von `JOIN` und `AGGREGATE`-Operationen wurde verbessert.
- LIKE – Die Leistung von Abfragen, die `LIKE`-Prädikate für die Spalten von `information_schema`-Tabellen verwenden, wurde verbessert.
- ORDER BY UND LIMIT – Verbesserte Pläne, Leistung und Speichernutzung für Abfragen mit `ORDER BY` und `LIMIT`, um unnötigen Datenaustausch zu vermeiden.
- ORDER BY – `ORDER BY`-Operationen werden jetzt standardmäßig verteilt, sodass größere `ORDER BY`-Klauseln verwendet werden können.
- Konvertierungen von Zeilentypen – Verbesserte Leistung bei der Konvertierung zwischen `ROW`-Typen.
- Strukturelle Typen – Verbesserte Leistung von Abfragen, die Strukturtypen verarbeiten und Scan-, Join-, Aggregations- oder Tabellenschreibvorgänge enthalten.
- Tabellen-Scans – Eine Optimierungsregel wurde eingeführt, um in bestimmten Fällen doppelte Tabellenscans zu vermeiden.
- UNION – Verbesserte Leistung für `UNION`-Abfragen.

Abfrageplanungsleistung

- Planungsleistung – Verbesserte Planungsleistung für Anfragen, die mehrere Tabellen mit einer großen Anzahl von Spalten verknüpfen.
- Prädikatsauswertungen – Verbesserte Leistung bei der Prädikatauswertung während des Prädikaten-Pushdowns in der Planung.
- Prädikat Pushdown-Unterstützung für Umwandlung – Prädikat-Pushdown für das Prädikat *`<column>`* `IN` *`<values list>`* unterstützen, bei dem Werte in der Werteliste umgewandelt werden müssen, um dem Spaltentyp zu entsprechen.
- Prädikatinferenz und Pushdown – Prädikatinferenz und Pushdown erweitert für Abfragen, die ein Prädikat *`<symbol>`* `IN` *`<subquery>`* verwenden.

- Zeitüberschreitungen – Es wurde ein Fehler behoben, der in seltenen Fällen Timeouts für die Abfrageplanung verursachen konnte.

Join-Leistung

- Joins mit Map-Spalten – Verbesserte Leistung von Joins und Aggregationen, die Map-Spalten enthalten.
- Joins mit ausschließlich Ungleichheitsbedingungen – Verbesserte Leistung von Joins mit ausschließlich Ungleichheitsbedingungen durch Verwendung eines verschachtelten Loop-Joins anstelle eines Hash-Joins.
- Externe Joins – Der Join-Verteilungstyp wird jetzt automatisch für Abfragen mit äußeren Joins ausgewählt.
- Bereich über Funktions-Joins – Verbesserte Leistung von Joins, bei denen die Bedingung ein Bereich über einer Funktion ist (z. B. `a JOIN b ON b.x < f(a.x) AND b.x > g(a.x)`).
- Spill-to-disk – spill-to-disk Verwandte Fehler und Speicherprobleme wurden behoben, um die Leistung zu verbessern und Speicherfehler in -`JOIN`Operationen zu reduzieren.

Unterabfrageleistung

- Korrelierte EXIST-Unterabfragen – Verbesserte Leistung von korrelierten `EXISTS`-Unterabfragen.
- Korrelierte Unterabfragen mit Gleichheiten – Verbesserte Unterstützung für korrelierte Unterabfragen, die Gleichheitsprädikate enthalten.
- Korrelierte Unterabfragen mit Ungleichheiten – Verbesserte Leistung für korrelierte Unterabfragen, die Ungleichheiten enthalten.
- Count(*) Aggregationen über Unterabfragen – Verbesserte Leistung von `count(*)`-Aggregationen über Unterabfragen mit bekannter konstanter Kardinalität.
- Ausbreitung des äußeren Abfragefilters – Verbesserte Leistung korrelierter Unterabfragen, wenn Filter aus der äußeren Abfrage an die Unterabfrage weitergegeben werden können.

Funktions-Leistung

- Aggregationsfensterfunktionen – Verbesserte Leistung von Aggregatfensterfunktionen.
- element_at () – Verbesserte Leistung von `element_at()`, für Maps mit konstanter Zeit anstatt proportional zur Größe der Map.

- Grouping() – Verbesserte Leistung für Abfragen mit `grouping()`.
- JSON-Umwandlung – Verbesserte Leistung bei der Umwandlung von `JSON`- auf `ARRAY`- oder `MAP`-Typen.
- Rückgabefunktionen – Verbesserte Leistung von Funktionen, die Karten zurückgeben.
- Map-to-map -Umwandlung – Die Leistung von map-to-map Casting wurde verbessert.
- Min() und max() – Die `min()`- und `max()`-Funktionen wurden optimiert, um unnötige Objekterstellung zu vermeiden und somit den Garbage-Collection-Overhead zu reduzieren.
- row_number () – Verbesserte Leistung und Speichernutzung für Abfragen mit `row_number()` gefolgt von einem Filter für die generierten Zeilennummern.
- Fensterfunktionen – Verbesserte Leistung von Abfragen, die Fensterfunktionen mit identischen `PARTITION BY`- und `ORDER BY`-Klauseln.
- Fensterfunktionen – Verbesserte Leistung bestimmter Fensterfunktionen (z. B. `LAG`), die ähnliche Spezifikationen haben.

Räumliche Leistung

- Geometrieserialisierung – Die Serialisierungsleistung von Geometriewerten wurde verbessert.
- Räumliche Funktionen – Die Leistung von `ST_Intersects()`, `ST_Contains()`, `ST_Touches()`, `ST_Within()`, `ST_Overlaps()`, `ST_Disjoint()`, `transform_values()`, `ST_XMin()`, `ST_XMax()`, `ST_YMin()`, `ST_YMax()`, `ST_Crosses()` und `array_intersect()` wurde verbessert.
- ST_Distance () – Verbesserte Leistung von Join-Abfragen, die die `ST_Distance()`-Funktion verwenden.
- ST_Intersection() – Optimierte `ST_Intersection()`-Funktion für Rechtecke, die an Koordinatenachsen ausgerichtet sind (z. B. Polygone, die von der `ST_Envelope()`- und `bing_tile_polygon()`-Funktionen produziert wurden).

JSON-bezogene Verbesserungen

Map-Funktionen

- Verbesserte Leistung von Map tiefgestellt von `O(n)` auf `O(1)` in allen Fällen. Bisher nutzten nur Karten, die von bestimmten Funktionen und Lesern erstellt wurden, diese Verbesserung.
- Funktionen `map_from_entries()` und `map_entries()` hinzugefügt.

Umwandlung

- Es wurde die Möglichkeit hinzugefügt, in `JSON` von `REAL`, `TINYINT` oder `SMALLINT` umzuwandeln.
- Sie können jetzt `JSON` auf `ROW` umwandeln, selbst wenn die `JSON` enthält nicht jedes Feld in der `ROW` aus.
- Verbesserte Leistung von `CAST(json_parse(...) AS ...)`.
- Verbesserte Leistung bei der Umwandlung von `JSON`- auf `ARRAY`- oder `MAP`-Typen.

Neue JSON-Funktionen

- is_json_scalar()

Abwärtskompatible Änderungen

Unterbrechende Änderungen umfassen Fehlerbehebungen, Änderungen an räumlichen Funktionen, ersetzte Funktionen und die Einführung von Grenzwerten. Verbesserungen bei der ANSI-SQL-Konformität können Abfragen brechen, die von nicht standardmäßigem Verhalten abhängen.

Fehlerbehebungen

Die folgenden Änderungen korrigieren Verhaltensprobleme, die dazu führten, dass Abfragen erfolgreich ausgeführt wurden, aber mit ungenauen Ergebnissen.

- fixed_len_byte_array Parquet-Spalten werden jetzt als DECIMAL akzeptiert – Abfragen von Parquet-Spalten vom Typ `fixed_len_byte_array` sind erfolgreich und geben korrekte Werte zurück, wenn sie im Parquet-Schema als `DECIMAL` annotiert sind. Fragt ab `fixed_len_byte_array`-Spalten ohne die `DECIMAL`-Notation schlägt mit einem Fehler fehl. Zuvor wurden Abfragen auf `fixed_len_byte_array`-Spalten ohne die DECIMAL-Anmerkung erfolgreich, aber unverständliche Werte zurückgegeben.
- json_parse() ignoriert keine nachgestellten Zeichen mehr – Zuvor wurden Eingaben wie `[1,2]abc` würde erfolgreich als `[1,2]` geparst. Die Verwendung nachgestellter Zeichen erzeugt nun die Fehlermeldung '[1, 2] abc' kann nicht in JSON konvertiert werden.
- Round() Dezimalgenauigkeit korrigiert – `round(x, d)` rundet `x` jetzt korrekt, wenn `x` ein DEZIMAL ist oder wenn `x` ein DEZIMAL mit Skala 0 und `d` eine negative Ganzzahl ist. Zuvor trat in diesen Fällen keine Rundung auf.
- round(x, d) and truncate(x, d) – Der Parameter `d` in der Signatur von Funktionen `round(x, d)` und `truncate(x, d)` ist jetzt vom Typ `INTEGER`. Zuvor konnte `d` vom Typ `BIGINT` sein.

- Map() mit doppelten Schlüsseln – `map()` löst jetzt einen Fehler bei doppelten Schlüsseln aus, anstatt eine beschädigte Karte im Hintergrund zu erzeugen. Abfragen, die derzeit Kartenwerte mit doppelten Schlüsseln erstellen, schlagen jetzt mit einem Fehler fehl.
- map_from_entries() löst einen Fehler mit Nulleinträgen aus – `map_from_entries()`löst nun einen Fehler aus, wenn das Eingabe-Array einen Nulleintrag enthält. Abfragen, die eine Karte erstellen, indem `NULL` als Wert übergeben wird, schlagen jetzt fehl.
- Tabellen – Tabellen mit nicht unterstützten Partitionstypen können nicht mehr erstellt werden.
- Verbesserte numerische Stabilität in statistischen Funktionen – Die numerische Stabilität der statistischen Funktionen `corr()`, `covar_samp()`, `regr_intercept()` und `regr_slope()` wurde verbessert.
- TIMESTAMP-Präzision, die in Parquet definiert ist, wird jetzt eingehalten – Die Präzision von `TIMESTAMP`-Werten und die Genauigkeit, die für die `TIMESTAMP` im Parquet-Schema muss nun übereinstimmen. Nicht übereinstimmende Präzisionen führen zu falschen Zeitstempeln.
- Zeitzoneninformation – Zeitzoneninformationen werden jetzt mithilfe des java.time-Pakets des Java 1.8 SDK berechnet.
- SUMME der Datentypen INTERVAL_DAY_TO_SECOND und INTERVAL_YEAR_TO_MONTH – Sie können `SUM(NULL)` nicht mehr direkt verwenden. Um `SUM(NULL)` zu benutzen, wandeln Sie `NULL` in einen Datentyp wie `BIGINT`, `DECIMAL`, `REAL`, `DOUBLE`, `INTERVAL_DAY_TO_SECOND` oder `INTERVAL_YEAR_TO_MONTH` um.

Änderungen an Geodatenfunktionen

Folgende Änderungen an räumlichen Funktionen wurden vorgenommen.

- Änderungen der Funktionsnamen – Einige Funktionsnamen haben sich geändert. Weitere Informationen finden Sie unter Änderungen des Geodaten-Funktionennamens in Athena-Engine-Version 2.
- VARBINARY – `VARBINARY` wird nicht mehr direkt für die Eingabe in Geodatenfunktionen unterstützt. Um beispielsweise die Fläche einer Geometrie direkt zu berechnen, muss die Geometrie nun entweder im `VARCHAR`- oder im `GEOMETRY`-Format eingegeben werden. Die Problemumgehung besteht darin, Transformationsfunktionen zu verwenden, wie in den folgenden Beispielen.
  - Um `ST_area()` zum Berechnen der Fläche für die `VARBINARY`-Eingabe im Well-Known-Binary-(WKB)-Format zu verwenden, übergeben Sie die Eingabe zuerst an `ST_GeomFromBinary()`, zum Beispiel:

```
ST_area(ST_GeomFromBinary(<wkb_varbinary_value>))
```

- Um `ST_area()` zum Berechnen der Fläche für die `VARBINARY`-Eingabe im Legacy-Binärformat zu verwenden, übergeben Sie dieselbe Eingabe zuerst an die `ST_GeomFromLegacyBinary()`-Funktion, zum Beispiel:

```
ST_area(ST_GeomFromLegacyBinary(<legacy_varbinary_value>))
```

- ST_ExteriorRing() und ST_Polygon() – `ST_ExteriorRing()` und akzeptieren `ST_Polygon()` jetzt nur Polygone als Eingaben. Zuvor akzeptierten diese Funktionen fälschlicherweise andere Geometrien.
- ST_Distance() – Wie von der SQL/MM-Spezifikation erforderlich, gibt die `ST_Distance()`-Funktion jetzt `NULL` zurück, wenn eine der Eingaben eine leere Geometrie ist. Zuvor wurde `NaN` zurückgegeben.

ANSI-SQL-Compliance

Die folgenden Syntax- und Verhaltensprobleme wurden entsprechend dem ANSI-SQL-Standard korrigiert.

- Cast()-Operationen – Cast()-Operationen von REAL oder DOUBLE nach DECIMAL entsprechen nun dem SQL-Standard. Zum Beispiel hat `cast (double '10000000000000000000000000000000' as decimal(38))` zuvor `10000000000000000536616220439 3472` zurückgegeben, jetzt aber `10000000000000000000000000000000`.
- JOIN ... USING – `JOIN ... USING` entspricht jetzt der Standard-SQL-Semantik. Zuvor musste `JOIN ... USING` den Tabellennamen in Spalten qualifizieren, und die Spalte aus beiden Tabellen war in der Ausgabe vorhanden. Tabellenqualifikationen sind jetzt ungültig und die Spalte ist nur einmal in der Ausgabe vorhanden.
- Literale vom Typ ROW entfernt – Das Literalformat des ROW-Typs`ROW<int, int>(1, 2)` wird nicht mehr unterstützt. Verwenden Sie stattdessen die `ROW(1 int, 2 int)`-Syntax.
- Semantik für gruppierte Aggregation – Gruppierte Aggregationen verwenden `IS NOT DISTINCT FROM`-Semantik statt Gleichheitssemantik. Gruppierte Aggregationen geben jetzt korrekte Ergebnisse zurück und zeigen eine verbesserte Leistung beim Gruppieren nach `NaN`-Gleitkommawerten. Gruppierung auf Karten-, Listen- und Zeilentypen, die Nullen enthalten, wird unterstützt.

- Typen mit Anführungszeichen sind nicht mehr zulässig – Gemäß ANSI-SQL-Standard dürfen Datentypen nicht mehr in Anführungszeichen eingeschlossen werden. `SELECT "date" '2020-02-02'` ist beispielsweise keine gültige Abfrage mehr. Verwenden Sie stattdessen die `SELECT date '2020-02-02'`-Syntax.
- Zugriff auf anonyme Zeilenfelder – Auf anonyme Zeilenfelder kann nicht mehr mit der Syntax `[.field0, .field1, ...]` zugegriffen werden.
- Komplexe Gruppierungsvorgänge – Die komplexen Gruppierungsoperationen `GROUPING SETS`, `CUBE` und `ROLLUP` unterstützen keine Gruppierung von Ausdrücken, die aus Eingabespalten bestehen. Es sind nur Spaltennamen zulässig.

Ersetzte Funktionen

Die folgenden Funktionen werden nicht mehr unterstützt und wurden durch Syntax ersetzt, die dieselben Ergebnisse liefert.

- Information_schema. __internal_partitions__ – Die Verwendung von `__internal_partitions__` wird in der Athena-Engine Version 2 nicht mehr unterstützt. Verwenden Sie für eine äquivalente Syntax `SELECT * FROM "<table_name>$partitions"` oder `SHOW PARTITIONS`. Weitere Informationen finden Sie unter Auflisten von Partitionen für eine bestimmte Tabelle.
- Ersetzte Geodatenfunktionen – Eine Liste der Geodatenfunktionen, deren Namen sich geändert haben, finden Sie unter Änderungen des Geodaten-Funktionennamens in Athena-Engine-Version 2.

Einschränkungen

Die folgenden Grenzwerte wurden in Athena-Engine-Version 2 eingeführt, um sicherzustellen, dass Abfragen aufgrund von Ressourcenbeschränkungen nicht fehlschlagen. Diese Grenzwerte sind von Benutzern nicht konfigurierbar.

- Anzahl der Ergebniselemente – Die Anzahl der Ergebniselemente n ist für die folgenden Funktionen auf 10.000 oder weniger beschränkt: `min(col, n)`, `max(col, n)`, `min_by(col1, col2, n)` und `max_by(col1, col2, n)`.
- GROUPING SETS – Die maximale Anzahl von Slices in einem Gruppierungssatz ist 2048.
- Maximale Zeilenlänge der Textdatei – Die standardmäßige maximale Zeilenlänge für Textdateien beträgt 200 MB.

- Sequenzfunktion maximale Ergebnisgröße – Die maximale Ergebnisgröße einer Sequenzfunktion beträgt 50000 Einträge. Zum Beispiel ist `SELECT sequence(0,45000,1)` erfolgreich, aber `SELECT sequence(0,55000,1)` schlägt mit der Fehlermeldung fehl. Das Ergebnis der Sequenzfunktion darf nicht mehr als 50000 Einträge haben. Diese Beschränkung gilt für alle Eingabetypen für Sequenzfunktionen, auch für Zeitstempel.

SQL-Referenz für Athena

Amazon Athena unterstützt eine Teilmenge von Data-Definition-Language-(DDL)- und Data-Manipulation-Language-(DML)-Anweisungen, -Funktionen, -Operatoren und -Datentypen. Mit einigen Ausnahmen basiert Athena-DDL auf DDL-HiveQL-DDL. Weitere Informationen über Athena-Engine-Versionen finden Sie unter Athena-Engine-Versionierung.

Themen

Datentypen in Amazon Athena

Wenn Sie CREATE TABLE ausführen, geben Sie Spaltennamen und den Datentyp an, den jede Spalte enthalten kann. Athena unterstützt die unten aufgeführten Datentypen. Weitere Informationen zu den Datentyp-Zuordnungen, die der JDBC-Treiber zwischen Athena, JDBC und Java unterstützt, finden Sie unter Datentypen im Installations- und Konfigurationsleitfaden für JDBC-Treiber. Weitere Informationen zu den Datentyp-Zuordnungen, die der ODBC-Treiber zwischen Athena und SQL unterstützt, finden Sie unter Datentypen im Installations- und Konfigurationsleitfaden für ODBC-Treiber.

- **`boolean`** – Die Werte sind `true` und `false`.
- **`tinyint`** – Eine 8-Bit signierte Ganzzahl im Zweierkomplement-Format mit einem Mindestwert von -2^7 und einem Höchstwert von 2^7-1.
- **`smallint`** – Eine 16-Bit signierte Ganzzahl im Zweierkomplement-Format mit einem Mindestwert von -2^{15} und einem Höchstwert von $2^{15}-1$.

- **`int`** und **`integer`** – Athena verwendet je nach Art der Abfrage verschiedene Ausdrücke für Ganzzahl.
  - **`int`** – In DDL-Abfragen (Data Definition Language) wie `CREATE TABLE` verwenden Sie den Datentyp `int`.
  - **`integer`** – In DML-Abfragen wie `SELECT * FROM` verwendet Athena den `integer` Datentyp. `integer` wird als signierter 32-Bit-Wert im Zweierkomplement-Format mit einem Mindestwert von -2^{31} und einem Höchstwert von $2^{31}-1$ dargestellt.
    - Um Kompatibilität mit den geschäftlichen Analyseanwendungen zu gewährleisten, gibt der JDBC-Treiber den Typ `integer` zurück.
- **`bigint`** – Eine 64-Bit signierte Ganzzahl im Zweierkomplement-Format mit einem Mindestwert von -2^{63} und einem Höchstwert von $2^{63}-1$.
- **`double`** – Eine signierte 64-Bit-Gleitkommazahl mit doppelter Genauigkeit. Der Bereich liegt zwischen 4,94065645841246544e-324d bis 1,79769313486231570e+308d, positiv oder negativ. `double` folgt dem IEEE-Standard für Gleitkommaarithmetik (IEEE 754).
- **`float`** – Eine signierte 32-Bit-Gleitkommazahl mit einfacher Genauigkeit. Der Bereich liegt zwischen 1,40129846432481707e-45 bis 3,40282346638528860e+38, positiv oder negativ. `float` folgt dem IEEE-Standard für Gleitkommaarithmetik (IEEE 754). Entspricht dem `real` in Presto. Verwenden Sie in Athena `float` in DDL-Anweisungen wie `CREATE TABLE` und `real` in SQL-Funktionen wie `SELECT CAST`. Der AWS Glue Crawler gibt Werte in zurück `float`und Athena übersetzt die `float` Typen `real` und intern (siehe 5. Juni 2018 Versionshinweise).
- **`decimal`**(*`precision`*, *`scale`*) – *`precision`* ist die Gesamtanzahl der Stellen. *`scale`* (optional) ist die Anzahl der Nachkommastellen mit einem Standardwert von 0. Verwenden Sie z. B. diese Definitionen: `decimal(11,5)`, `decimal(15)`. Die maximale Wert der *`Genauigkeit`* ist 38 und der maximale Wert für die *`Skalierung`* beträgt 38.

  Um Dezimalwerte als Literale anzugeben, z. B. bei der Auswahl von Zeilen mit einem bestimmten Dezimalwert im DDL-Ausdruck einer Abfrage, legen Sie als Typdefinition `decimal` fest und listen Sie die Dezimalwerte als Literalwert (in einfachen Anführungszeichen) in Ihrer Abfrage auf, wie in diesem Beispiel: `decimal_value = decimal '0.12'`.
- **`char`** – Zeichendaten mit fester Länge, die zwischen 1 und 255 Zeichen liegen muss, z. B `char(10)`. Weitere Informationen finden Sie unter CHAR-Hive-Datentyp.

> **Note**
>
> Wenn Sie die `substr`-Funktion verwenden möchten, um eine Teilzeichenfolge der angegebenen Länge aus einem `char`-Datentyp zurückzugeben, müssen Sie den `char`-Wert, wie im folgenden Beispiel, zuerst in einen `varchar` umwandeln.
>
> ```
> substr(cast(col1 as varchar), 1, 4)
> ```

- **`varchar`** – Zeichendaten mit variabler Länge, die zwischen 1 und 65535 Zeichen liegen muss, z. B `varchar(10)`. Weitere Informationen finden Sie unter VARCHAR-Hive-Datentyp.
- **`string`** – Ein Zeichenfolgenliteral, das in einfache oder doppelte Anführungszeichen eingeschlossen ist. Weitere Informationen finden Sie unter STRING-Hive-Datentyp.

> **Note**
>
> Nicht-Zeichenfolgen-Datentypen können nicht zu `string` in Athena umgewandelt werden. Wandeln Sie sie stattdessen zu `varchar` um.

- **`ipaddress`** – Stellt eine IP-Adresse in DML-Abfragen dar. Wird für DDL nicht unterstützt. Weitere Informationen finden Sie unter IPADDRESS.
- **`binary`** – Wird für Daten in Parquet verwendet.
- **`date`** – Ein Datum im ISO-Format, z. B. *`YYYY`*`-`*`MM`*`-`*`DD`*. Beispiel: `date '2008-09-15'` Eine Ausnahme ist OpenCSV,SerDe das die Anzahl der Tage verwendet, die seit dem 1. Januar 1970 verstrichen sind. Weitere Informationen finden Sie unter OpenCSVSerDe für CSV-Verarbeitung.
- **`timestamp`**– Datum und Uhrzeit in einem `java.sql.Timestamp`-kompatiblen Format bis zu einer maximalen Auflösung von Millisekunden, wie *`yyyy`*`-`*`MM`*`-`*`dd HH`*`:`*`mm`*`:`*`ss`*`[.f...]`. Beispiel: `timestamp '2008-09-15 03:04:05.324'` Eine Ausnahme ist OpenCSVSerDe, das `timestamp` Daten im numerischen UNIX-Format verwendet (z. B. `1579059880000`).

  Weitere Informationen zum Arbeiten mit Zeitstempeln finden Sie unter Mit Zeitstempeldaten arbeiten später in dieser Dokumentation.
- **`array`**<*`data_type`*> – Ein Array des angegebenen Komponententyps.

  Beispiel

  ```
  CREATE TABLE table array_table (c1 array<integer>) LOCATION '...';
  ```

```
INSERT INTO array_table values(ARRAY[1,2,3]);
```

- **map**<*primitive_type*, *data_type*> – Eine Zuordnung zwischen den angegebenen Komponententypen.

  Beispiel

  ```
  CREATE TABLE map_table(c1 map<string, integer>) LOCATION '...';
  INSERT INTO map_table values(MAP(ARRAY['foo', 'bar'], ARRAY[1, 2]));
  ```

- **struct**<*col_name* : *data_type*, *...*> – Eine Sammlung von Elementen verschiedener Komponententypen.

  Beispiel

  ```
  CREATE TABLE struct_table(c1 struct<name:varchar(10), age:integer>) LOCATION '...';
  INSERT INTO struct_table SELECT CAST(ROW('Bob', 38) AS ROW(name VARCHAR(10), age
   INTEGER));
  ```

Mit Zeitstempeldaten arbeiten

In diesem Abschnitt werden einige Überlegungen zur Arbeit mit Zeitstempeldaten in Athena beschrieben.

> **Note**
>
> Die Behandlung von Zeitstempeln hat sich zwischen der Athena-Engine-Version 2 und der Athena-Engine-Version 3 etwas geändert. Informationen zu Fehlern im Zusammenhang mit Zeitstempeln, die in Athena-Engine-Version 3 auftreten können, sowie Lösungsvorschläge finden Sie unter Änderungen des Zeitstempels in der Athena-Engine-Version 3-Referenz.

Format für das Schreiben von Zeitstempeldaten in Amazon-S3-Objekte

Das Format, in dem Zeitstempeldaten in Amazon S3-Objekte geschrieben werden sollen, hängt sowohl vom Spaltendatentyp als auch von der verwendeten SerDe Bibliothek ab.

- Wenn Sie eine Tabellenspalte vom Typ `DATE` haben, erwartet Athena, dass es sich bei der entsprechenden Spalte oder Eigenschaft der Daten um eine Zeichenfolge im ISO-Format `YYYY-MM-DD` oder um einen integrierten Datumstyp wie bei Parquet oder ORC handelt.

- Wenn Sie eine Tabellenspalte vom Typ `TIME` haben, erwartet Athena, dass es sich bei der entsprechenden Spalte oder Eigenschaft der Daten um eine Zeichenfolge im ISO-Format `HH:MM:SS` oder um einen integrierten Zeittyp wie bei Parquet oder ORC handelt.
- Wenn Sie eine Tabellenspalte des Typs `TIMESTAMP` haben, erwartet Athena, dass die entsprechende Spalte oder Eigenschaft der Daten eine Zeichenkette im Format `YYYY-MM-DD HH:MM:SS.SSS` ist (beachten Sie das Leerzeichen zwischen Datum und Uhrzeit), oder ein eingebauter Zeittyp wie die für Parquet, ORC oder Ion.

> **Note**
>
> OpenCSVSerDe -Zeitstempel sind eine Ausnahme und müssen als UNIX-Epochen mit Millisekundenauflösung codiert werden.

Sicherstellen, dass zeitlich aufgeteilte Daten mit dem Zeitstempelfeld in einem Datensatz übereinstimmen

Der Hersteller der Daten muss sicherstellen, dass die Partitionswerte mit den Daten innerhalb der Partition übereinstimmen. Wenn Ihre Daten beispielsweise eine `-timestamp`Eigenschaft haben und Sie Firehose verwenden, um die Daten in Amazon S3 zu laden, müssen Sie die dynamische Partitionierung verwenden, da die Standardpartitionierung von Firehose ist wall-clock-based.

String als Datentyp für Partitionsschlüssel verwenden

Aus Leistungsgründen ist es vorzuziehen, `STRING` als Datentyp für Partitionsschlüssel zu verwenden. Obwohl Athena Partitionswerte im Format `YYYY-MM-DD` als Datum erkennt, wenn Sie den `DATE`-Typ verwenden, kann dies zu einer schlechten Leistung führen. Aus diesem Grund empfehlen wir, stattdessen den `STRING`-Datentyp für Partitionsschlüssel zu verwenden.

Wie Sie Abfragen für Zeitstempelfelder schreiben, die ebenfalls zeitpartitioniert sind

Wie Sie Abfragen für Zeitstempelfelder schreiben, die zeitpartitioniert sind, hängt vom Tabellentyp ab, den Sie abfragen möchten.

Hive-Tabellen

Bei den Hive-Tabellen, die in Athena am häufigsten verwendet werden, hat die Abfrage-Engine keine Kenntnis von den Beziehungen zwischen Spalten und Partitionsschlüsseln. Aus diesem Grund müssen Sie Ihren Abfragen immer Prädikate sowohl für die Spalte als auch für den Partitionsschlüssel hinzufügen.

Nehmen wir beispielsweise an, Sie haben eine `event_time`-Spalte und einen `event_date`-Partitionsschlüssel und möchten Ereignisse zwischen 23:00 und 03:00 Uhr abfragen. In diesem Fall müssen Sie Prädikate in Ihre Abfrage sowohl für die Spalte als auch für den Partitionsschlüssel aufnehmen, wie im folgenden Beispiel.

```
WHERE event_time BETWEEN start_time AND end_time
  AND event_date BETWEEN start_time_date AND end_time_date
```

Iceberg-Tabellen

Bei Iceberg-Tabellen können Sie berechnete Partitionswerte verwenden, was Ihre Abfragen vereinfacht. Nehmen wir beispielsweise an, Ihre Iceberg-Tabelle wurde mit einer `PARTITIONED BY`-Klausel wie der folgenden erstellt:

```
PARTITIONED BY (event_date month(event_time))
```

In diesem Fall löscht die Abfrage-Engine automatisch Partitionen auf der Grundlage der Werte der `event_time`-Prädikate. Aus diesem Grund muss Ihre Abfrage nur ein Prädikat für `event_time` angeben, wie im folgenden Beispiel.

```
WHERE event_time BETWEEN start_time AND end_time
```

Weitere Informationen finden Sie unter Erstellen von Iceberg-Tabellen.

DML-Abfragen, -Funktionen und -Operatoren

Die Athena-DML-Abfrage-Engine unterstützt im Allgemeinen die Trino- und Presto-Syntax und fügt eigene Verbesserungen hinzu. Nicht alle Trino- oder Presto-Features werden von Athena unterstützt. Weitere Informationen finden Sie in den Themen für spezifische Anweisungen in diesem Abschnitt und Überlegungen und Einschränkungen. Informationen zu Funktionen finden Sie unter Funktionen in Amazon Athena. Weitere Informationen über Athena-Engine-Versionen finden Sie unter Athena-Engine-Versionierung.

Informationen zu DDL-Anweisungen finden Sie unter DDL-Anweisungen. Eine Liste der nicht unterstützten DDL-Anweisungen finden Sie unter Nicht unterstützte DDLs.

SELECT

Ruft Datenzeilen aus null oder mehr Tabellen ab.

> **Note**
>
> Dieses Thema enthält zusammenfassende Informationen zur Referenz. Umfassende Informationen über die Verwendung von `SELECT` und der SQL-Sprache gehen über den Rahmen dieser Dokumentation hinaus. Weitere Informationen zur Verwendung von SQL speziell für Athena finden Sie unter Überlegungen und Einschränkungen für SQL-Abfragen in Amazon Athena und Ausführen von SQL-Abfragen mit Amazon Athena. Für ein Beispiel für das Erstellen einer Datenbank, das Erstellen einer Tabelle und das Ausführen einer `SELECT`-Abfrage auf dem Tisch in Athena siehe Erste Schritte.

Syntax

```
[ 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**
>
> Reservierte Wörter in SQL SELECT-Anweisungen müssen in doppelte Anführungszeichen eingeschlossen werden. Weitere Informationen finden Sie unter Liste reservierter Schlüsselwörter in SQL-SELECT-Anweisungen.

Parameter

[WITH with_query [, ...]]

Mit `WITH` können Sie verschachtelte Abfragen reduzieren oder Unterabfragen vereinfachen.

Die Verwendung der `WITH`-Klausel zur Erstellung rekursiver Abfragen wird ab der Athena-Engine-Version 3 unterstützt. Die maximale Rekursionstiefe beträgt 10.

Die Klausel `WITH` geht der `SELECT`-Liste in einer Abfrage voraus und legt eine oder mehrere Unterabfragen für die Verwendung in der `SELECT`-Abfrage fest.

Jede Unterabfrage definiert eine temporäre Tabelle ähnlich wie eine Ansichtdefinition, die mit der `FROM`-Klausel referenziert werden kann. Die Tabellen werden nur verwendet, wenn die Abfrage ausgeführt wird.

Die Syntax von `with_query` lautet:

```
subquery_table_name [ ( column_name [, ...] ) ] AS (subquery)
```

Wobei gilt:

- `subquery_table_name` ist ein eindeutiger Name für eine temporäre Tabelle, die die Ergebnisse der Unterabfrage mit `WITH`-Klausel definiert. Jede `subquery` muss einen Tabellennamen erhalten, der in der `FROM`-Klausel referenziert werden kann.
- `column_name [, ...]` ist eine optionale Liste der ausgegebenen Spaltennamen. Die Anzahl der Spaltennamen muss größer als oder gleich der Anzahl der Spalten sein, die von der `subquery` definiert wird.
- `subquery` ist eine Abfrageanweisung.

[ALL | DISTINCT] select_expr

`select_expr` legt die auszuwählenden Zeilen fest.

`ALL` ist die Standardeinstellung. Bei Verwendung von `ALL` wird dies so behandelt, als wäre kein Wert angegeben worden. Alle Zeilen aller Spalten werden einschließlich Duplikaten ausgewählt.

Verwenden Sie `DISTINCT`, um nur eindeutige Werte zurückzugeben, wenn eine Spalte Duplikatwerte enthält.

FROM from_item [,...]

Gibt die Eingabe der Abfrage an, wobei `from_item` eine Ansicht, ein JOIN-Konstrukt oder eine Unterabfrage wie unten beschrieben sein kann.

`from_item` kann eines der Folgenden sein:

- `table_name [ [ AS ] alias [ (column_alias [, ...]) ] ]`

  Hier ist `table_name` der Name der Zieltabelle, aus der Zeilen ausgewählt werden. `alias` ist der Name für die Ausgabe der `SELECT`-Anweisung und `column_alias` definiert die Spalten für den angegebenen `alias`.

-ODER-

- `join_type from_item [ ON join_condition | USING ( join_column [, ...] ) ]`

  Hierbei ist `join_type` eines der Folgenden:

  - `[ INNER ] JOIN`
  - `LEFT [ OUTER ] JOIN`
  - `RIGHT [ OUTER ] JOIN`
  - `FULL [ OUTER ] JOIN`
  - `CROSS JOIN`
  - `ON join_condition | USING (join_column [, ...])` Wobei Sie durch Verwenden von `join_condition` die Spaltennamen für JOIN-Schlüssel in mehreren Tabellen angeben können. Für die Verwendung von `join_column` muss `join_column` in beiden Tabellen vorhanden sein.

[WHERE condition]

Filtert Ergebnisse nach dem von Ihnen angegebenen `condition`, wobei `condition` im Allgemeinen die folgende Syntax hat.

```
column_name operator value [[[AND | OR] column_name operator value] ...]
```

Der *`Operator`* kann einer der Komparatoren =, >, <, >=, <=, <>, != sein.

Die folgenden Unterabfrageausdrücke können auch in der `WHERE`-Klausel verwendet werden.

- `[NOT] BETWEEN` *`integer_A`* `AND` *`integer_B`* – Gibt einen Bereich zwischen zwei Ganzzahlen an, wie im folgenden Beispiel. Wenn der Spaltendatentyp `varchar` ist, muss die Spalte zuerst in eine Ganzzahl umgewandelt werden.

  ```
  SELECT DISTINCT processid FROM "webdata"."impressions"
  WHERE cast(processid as int) BETWEEN 1500 and 1800
  ORDER BY processid
  ```

- `[NOT] LIKE` *`value`* – Sucht nach dem angegebenen Muster. Verwenden Sie das Prozentzeichen (%) als Platzhalterzeichen, wie im folgenden Beispiel.

  ```
  SELECT * FROM "webdata"."impressions"
  ```

```
WHERE referrer LIKE '%.org'
```

- `[NOT] IN (value[, value[, ...])` – Gibt eine Liste möglicher Werte für eine Spalte an, wie im folgenden Beispiel gezeigt.

```
SELECT * FROM "webdata"."impressions"
WHERE referrer IN ('example.com','example.net','example.org')
```

[GROUP BY [ALL | DISTINCT] grouping_expressions [, ...]]

Teilt die Ausgabe der `SELECT`-Anweisung in Zeilen mit passenden Werten.

Über `ALL` und `DISTINCT` wird festgelegt, ob für Duplikate jeweils eine eigene Ausgabezeile angelegt wird. Wenn nichts angegeben ist, wird von `ALL` ausgegangen.

`grouping_expressions` ermöglicht Ihnen, komplexe Gruppierungsoperationen auszuführen. Sie können komplexe Gruppierungsvorgänge verwenden, um Analysen durchzuführen, die eine Aggregation mehrerer Spaltengruppen in einer einzigen Abfrage erfordern.

Das `grouping_expressions`-Element kann eine beliebige Funktion sein, z. B. `SUM`, `AVG` oder `COUNT`, die auf Eingabespalten ausgeführt wird.

Mithilfe von `GROUP BY`-Ausdrücken kann die Ausgabe nach Eingabespaltennamen gruppiert werden, die nicht in der Ausgabe der `SELECT`-Anweisung auftauchen.

Alle Ausgabeausdrücke müssen entweder Aggregatfunktionen oder Spalten sein, die in der `GROUP BY`-Klausel vorhanden sind.

Sie können mit einer einzelnen Abfrage eine Analyse durchführen, für die mehrere Spaltensätze zusammengefasst werden müssen.

Athena unterstützt komplexe Aggregationen mit `GROUPING SETS`, `CUBE` und `ROLLUP`. `GROUP BY GROUPING SETS` gibt mehrere Spaltenlisten an, nach denen gruppiert werden soll. `GROUP BY CUBE` generiert alle möglichen Gruppierungssätze für einen gegebenen Satz von Spalten. `GROUP BY ROLLUP` generiert alle möglichen Zwischensummen für einen gegebenen Satz von Spalten. Komplexe Gruppierungsoperationen unterstützen keine Gruppierung von Ausdrücken, die aus Eingabespalten bestehen. Es sind nur Spaltennamen zulässig.

Oft können Sie mit `UNION ALL` dieselben Ergebnisse erzielen wie mit diesen `GROUP BY`-Operationen. Abfragen, in denen `GROUP BY` verwendet wird, haben jedoch den Vorteil, dass

die Daten einmalig ausgelesen werden, wohingegen bei `UNION ALL` die zugrunde liegenden Daten dreimal ausgelesen werden, was zu uneinheitlichen Ergebnissen führen kann, wenn die Datenquelle sich geändert hat.

[HAVING condition]

Wird in Aggregatfunktionen und der `GROUP BY`-Klausel verwendet. Kontrolliert, welche Gruppen ausgewählt werden und schließt Gruppen aus, die die `condition` nicht erfüllen. Diese Filterung wird nach der Berechnung der Gruppierungs- und Aggregatfunktionen ausgeführt.

[{ UNION | INTERSECT | EXCEPT } [ALL | DISTINCT] union_query]]

`UNION`, `INTERSECT` und`EXCEPT` kombinieren die Ergebnisse mehrerer `SELECT`-Anweisungen in eine einzelne Abfrage. `ALL` oder `DISTINCT` steuern die Eindeutigkeit der Zeilen, die in der endgültigen Ergebnismenge enthalten sind.

`UNION` kombiniert die Zeilen, die sich aus der ersten Abfrage ergeben, mit den Zeilen, die aus der zweiten Abfrage resultieren. Um Duplikate zu eliminieren, erstellt `UNION` eine Hash-Tabelle, die Speicher verbraucht. Ziehen Sie für eine bessere Leistung die Verwendung von `UNION ALL` in Betracht, wenn Ihre Abfrage die Eliminierung von Duplikaten nicht erfordert. Mehrere `UNION`-Klauseln werden von links nach rechts verarbeitet, sofern Sie keine Klammern verwenden, um die Verarbeitungsreihenfolge explizit festzulegen.

`INTERSECT` gibt nur die Zeilen zurück, die in den Ergebnissen der ersten und der zweiten Abfrage vorhanden sind.

`EXCEPT` gibt die Zeilen aus den Ergebnissen der ersten Abfrage zurück, ausgenommen die Zeilen, die von der zweiten Abfrage gefunden wurden.

`ALL` bewirkt, dass alle Zeilen aufgenommen werden, auch wenn die Zeilen identisch sind.

`DISTINCT` bewirkt, dass nur eindeutige Zeilen in die kombinierte Ergebnismenge aufgenommen werden.

[ORDER BY expression [ASC | DESC] [NULLS FIRST | NULLS LAST] [, ...]]

Sortiert eine Ergebnisgruppe nach einem oder mehreren Ausgabe-`expression`.

Wenn die Klausel mehrere Ausdrücke enthält, wird die Ergebnisgruppe nach dem ersten `expression` sortiert. Danach wird der zweite `expression` auf die Zeilen mit passenden Werten aus dem ersten Ausdruck angewendet usw.

Jeder `expression` kann Ausgabespalten aus `SELECT` oder eine Ordinalzahl für eine Ausgabespalte nach Position (beginnend mit 1) festlegen.

`ORDER BY` wird im letzten Schritt nach `GROUP BY`- und `HAVING`-Klauseln ausgewertet. Über `ASC` und `DESC` wird festgelegt, ob die Ergebnisse in auf- oder absteigender Reihenfolge sortiert werden.

Die Standardnullsortierung ist `NULLS LAST` unabhängig von auf- oder absteigender Sortierreihenfolge.

[OFFSET count [ROW | ROWS]]

Verwenden Sie die `OFFSET`-Klausel, um eine Reihe führender Zeilen aus der Ergebnismenge zu verwerfen. Wenn die `ORDER BY`-Klausel vorhanden ist, wird die `OFFSET`-Klausel über eine sortierte Ergebnismenge ausgewertet und die Menge bleibt sortiert, nachdem die übersprungenen Zeilen verworfen wurden. Wenn die Abfrage keine `ORDER BY`-Klausel hat, ist es willkürlich, welche Zeilen verworfen werden. Wenn die durch `OFFSET` angegebene Anzahl der Größe der Ergebnismenge entspricht oder diese überschreitet, ist das Endergebnis leer.

LIMIT [count | ALL]

Beschränkt die Anzahl der Zeilen in der Ergebnisgruppe auf `count`. `LIMIT ALL` entspricht dem Weglassen der `LIMIT`-Klausel. Wenn die Abfrage keine `ORDER BY`-Klausel enthält, werden die Ergebnisse zufällig angezeigt.

TABLESAMPLE [BERNOULLI | SYSTEM] (percentage)

Optionaler Operator zur Auswahl von Zeilen aus einer Tabelle basierend auf einer Stichprobenmethode.

Mit `BERNOULLI` wird jede Zeile mit einer Wahrscheinlichkeit von `percentage` als Teil der Stichprobe ausgewählt. Alle physischen Blöcke der Tabelle werden gescannt und bestimmte Zeilen werden basierend auf einem Vergleich zwischen dem `percentage` der Stichproben und einem zufälligen, zur Laufzeit berechneten Wert übersprungen.

Mit `SYSTEM` wird die Tabelle in logische Datensegmente unterteilt. Aus der Tabelle werden Stichproben mit dieser Granularität entnommen.

Es werden entweder alle Zeilen aus einem bestimmten Segment ausgewählt oder das Segment wird basierend auf einem Vergleich zwischen dem `percentage` der Stichprobe und einem zufälligen, während der Laufzeit berechneten Wert übersprungen. `SYSTEM`-

Stichproben sind abhängig vom Connector. Mit dieser Methode kann keine unabhängige Stichprobenwahrscheinlichkeit garantiert werden.

[UNNEST (array_or_map) [WITH ORDINALITY]]

Erweitert ein Array oder eine Zuordnung in eine Beziehung. Arrays werden in eine einzelne Spalte erweitert. Zuordnungen werden in zwei Spalten (Schlüssel und Wert) erweitert.

Sie können `UNNEST` mit mehreren Argumenten verwenden, die in mehrere Spalten mit so vielen Zeilen wie das höchste Kardinalitätsargument erweitert werden.

Andere Spalten werden mit Nullen aufgefüllt.

Die `WITH ORDINALITY`-Klausel fügt eine Ordinalitätsspalte am Ende hinzu.

`UNNEST` wird normalerweise mit einem `JOIN` verwendet und kann auf Spalten von Beziehungen links des `JOIN` verweisen.

Abrufen der Dateispeicherorte für Quelldaten in Amazon S3

Um den Speicherort der Amazon-S3-Datei für die Daten in einer Tabellenzeile anzuzeigen, können Sie `"$path"` in einer `SELECT`-Abfrage verwenden, wie im folgenden Beispiel:

```
SELECT "$path" FROM "my_database"."my_table" WHERE year=2019;
```

Dies gibt ein Ergebnis wie das folgende zurück:

```
s3://DOC-EXAMPLE-BUCKET/datasets_mytable/year=2019/data_file1.json
```

Um eine sortierte, eindeutige Liste der S3-Dateinamenspfade für die Daten in einer Tabelle zurückzugeben, können Sie wie im folgenden Beispiel `SELECT DISTINCT` und `ORDER BY` verwenden.

```
SELECT DISTINCT "$path" AS data_source_file
FROM sampledb.elb_logs
ORDER By data_source_file ASC
```

Um nur die Dateinamen ohne Pfad zurückzugeben, können Sie `"$path"` wie im folgenden Beispiel als Parameter an eine `regexp_extract`-Funktion übergeben.

```
SELECT DISTINCT regexp_extract("$path", '[^/]+$') AS data_source_file
```

```
FROM sampledb.elb_logs
ORDER By data_source_file ASC
```

Um die Daten aus einer bestimmten Datei zurückzugeben, geben Sie die Datei in der `WHERE`-Klausel an, wie im folgenden Beispiel.

```
SELECT *,"$path" FROM my_database.my_table WHERE "$path" = 's3://DOC-EXAMPLE-BUCKET/
my_table/my_partition/file-01.csv'
```

Weitere Informationen und Beispiele finden Sie im Wissenscenter-Artikel Wie kann ich die Amazon-S3-Quelldatei für eine Zeile in einer Athena-Tabelle anzeigen?.

> **Note**
>
> In Athena, Hive oder Iceberg werden die versteckten Metadatenspalten `$bucket`, `$file_modified_time`, `$file_size` und `$partition` für Ansichten nicht unterstützt.

Maskieren von einfachen Anführungszeichen mit Escape

Um ein einzelnes Anführungszeichen mit Escape zu maskieren, stellen Sie ihm ein weiteres einfaches Anführungszeichen voran, wie im folgenden Beispiel. Verwechseln Sie dies nicht mit einem doppelten Anführungszeichen.

```
Select 'O''Reilly'
```

Ergebnisse

```
O'Reilly
```

Weitere Ressourcen

Weitere Informationen zur Verwendung von `SELECT`-Anweisungen in Athena finden Sie in den folgenden Ressourcen.

Weitere Informationen darüber	Sehen Sie dies an
Ausführen von Abfragen in Athena	Ausführen von SQL-Abfragen mit Amazon Athena

Weitere Informationen darüber	Sehen Sie dies an
Mit SELECT eine Tabelle erstellen	Erstellen einer Tabelle aus Abfrageergebnissen (CTAS)
Einfügen von Daten aus einer SELECT-Abfrage in eine andere Tabelle	INSERT INTO
Integrierte Funktionen in SELECT-Anweisungen verwenden	Funktionen in Amazon Athena
Verwenden von benutzerdefinierten Funktionen in SELECT-Anweisungen	Abfragen mit benutzerdefinierten Funktionen (User Defined Functions, UDFs)
Metadaten des Datenkatalogs abfragen	Abfragen von AWS Glue Data Catalog

INSERT INTO

Fügt neue Zeilen in eine Zieltabelle ein, basierend auf einer SELECT-Abfrageanweisung, die in einer Quelltabelle ausgeführt wird, oder basierend auf einem Satz von VALUES, der als Teil der Anweisung bereitgestellt wird. Wenn die Quelltabelle auf zugrunde liegenden Daten in einem Format wie CSV oder JSON basiert und die Zieltabelle auf einem anderen Format wie Parquet oder ORC basiert, können Sie INSERT INTO-Abfragen verwenden, um ausgewählte Daten in das Format der Zieltabelle zu transformieren.

Überlegungen und Einschränkungen

Beachten Sie Folgendes, wenn Sie INSERT-Abfragen mit Athena verwenden.

- Wenn Sie eine INSERT-Abfrage für eine Tabelle mit zugrunde liegenden Daten ausführen, die in Amazon S3 verschlüsselt sind, werden die Ausgabedateien, die die INSERT-Abfrage schreibt, standardmäßig nicht verschlüsselt. Wir empfehlen, INSERT-Abfrageergebnisse zu verschlüsseln, wenn Sie in Tabellen mit verschlüsselten Daten einfügen.

  Weitere Informationen zum Verschlüsseln von Abfrageergebnissen mithilfe der Konsole finden Sie unter Verschlüsseln der in Amazon S3 gespeicherten Athena-Abfrageergebnisse. Um die Verschlüsselung mit der AWS CLI oder der Athena-API zu aktivieren, verwenden Sie die EncryptionConfiguration Eigenschaften der -StartQueryExecutionAktion, um Amazon S3-Verschlüsselungsoptionen entsprechend Ihren Anforderungen anzugeben.

- Für `INSERT INTO`-Anweisungen gilt die erwartete Bucket-Eigentümereinstellung nicht für den Speicherort der Zieltabelle in Amazon S3. Die erwartete Bucket-Eigentümereinstellung gilt nur für den Amazon-S3-Ausgabespeicherort, den Sie für Athena-Abfrageergebnisse angeben. Weitere Informationen finden Sie unter Angeben eines Speicherorts des Abfrageergebnisses mithilfe der Athena-Konsole.
- ACID-konforme `INSERT INTO`-Anweisungen finden Sie im `INSERT INTO`-Abschnitt von Aktualisieren von Iceberg-Datentabellen.

Unterstützte Formate und SerDes

Sie können eine -`INSERT`Abfrage für Tabellen ausführen, die aus Daten mit den folgenden Formaten und erstellt wurden SerDes.

Data format (Datenformat)	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
Textdatei	org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
	Note CSV-, TSV- und benutzerdefinierte, durch Trennzeichen getrennte Dateien werden unterstützt.

Bucket-Tabellen werden nicht unterstützt

`INSERT INTO` wird für Bucket-Tabellen nicht unterstützt. Weitere Informationen finden Sie unter Partitionierung und Bucketing in Athena.

Verbundabfragen werden nicht unterstützt

`INSERT INTO` wird für Verbundabfragen nicht unterstützt. Ein Versuch, dies zu tun, kann zur Fehlermeldung führen: Dieser Vorgang wird derzeit für externe Kataloge nicht unterstützt. Weitere Informationen zu Verbundabfragen finden Sie unter Nutzung von Amazon-Athena-Verbundabfrage.

Partitionierung

Beachten Sie die Punkte in diesem Abschnitt, wenn Sie die Partitionierung mit `INSERT INTO`- oder `CREATE TABLE AS SELECT`-Abfragen verwenden.

Einschränkungen

Die Anweisung `INSERT INTO` unterstützt das Schreiben von maximal 100 Partitionen in die Zieltabelle. Wenn Sie die `SELECT`-Klausel für eine Tabelle mit mehr als 100 Partitionen ausführen, schlägt die Abfrage fehl, es sei denn, die `SELECT`-Abfrage ist auf 100 Partitionen oder weniger beschränkt.

Hinweise zum Umgehen dieser Einschränkung finden Sie unter Verwenden von CTAS und INSERT INTO zum Umgehen des Limits von 100 Partitionen.

Spalten-Reihenfolge

`INSERT INTO`- oder `CREATE TABLE AS SELECT`-Anweisungen erwarten, dass die partitionierte Spalte die letzte Spalte in der Liste der projizierten Spalten in einer `SELECT`-Anweisung ist.

Wenn die Quelltabelle nicht partitioniert ist oder im Vergleich zur Zieltabelle auf verschiedene Spalten partitioniert ist, betrachten Abfragen wie `INSERT INTO` *`destination_table`* `SELECT * FROM` *`source_table`* die Werte in der letzten Spalte der Quelltabelle als Werte für eine Partitionsspalte in der Zieltabelle. Beachten Sie dies beim Versuch, eine partitionierte Tabelle aus einer nicht partitionierten Tabelle zu erstellen.

Ressourcen

Weitere Informationen zur Verwendung von `INSERT INTO` mit Partitionierung finden Sie in den folgenden Ressourcen.

- Hinweise zum Einfügen von partitionierten Daten in eine partitionierte Tabelle finden Sie unter Verwenden von CTAS und INSERT INTO zum Umgehen des Limits von 100 Partitionen.
- Hinweise zum Einfügen von nicht-partitionierten Daten in eine partitionierte Tabelle finden Sie unter Verwenden von CTAS und INSERT INTO für ETL und Datenanalyse.

In Amazon S3 geschriebene Dateien

Athena schreibt Dateien in die Quelldatenspeicherorte in Amazon S3 als Ergebnis des `INSERT`-Befehls. Jede `INSERT`-Operation erstellt eine neue Datei, anstatt an eine vorhandene Datei anzuhängen. Die Dateispeicherorte hängen von der Struktur der Tabelle und ggf. der `SELECT`-Abfrage ab. Athena generiert für jede `INSERT`-Abfrage eine Daten-Manifest-Datei. Das Manifest verfolgt die Dateien, die die Abfrage geschrieben hat. Es wird im Verzeichnis der Athena-Abfrageergebnisse in Amazon S3 gespeichert. Weitere Informationen finden Sie unter Identifizieren von Abfrageausgabedateien.

Vermeiden Sie hohe Transaktionsaktualisierungen

Wenn Sie verwenden`INSERT INTO`, um einer Tabelle in Amazon S3 Zeilen hinzuzufügen, schreibt Athena vorhandene Dateien nicht neu oder ändert sie nicht. Stattdessen werden die Zeilen als eine oder mehrere neue Dateien geschrieben. Da Tabellen mit vielen kleinen Dateien zu einer geringeren Abfrageleistung und Schreib- und Lesevorgängen wie führen `PutObject` und zu höheren Kosten für Amazon S3 `GetObject` führen, sollten Sie bei der Verwendung von die folgenden Optionen berücksichtigen`INSERT INTO`:

- Führen Sie -`INSERT INTO`Operationen weniger häufig auf größeren Stapeln von Zeilen aus.
- Bei Volumes zur Datenaufnahme großer Datenmengen sollten Sie einen Service wie Amazon Data Firehose verwenden.
- Vermeiden Sie die vollständige Verwendung von `INSERT INTO` . Sammeln Sie stattdessen Zeilen in größeren Dateien und laden Sie sie direkt in Amazon S3 hoch, wo sie von Athena abgefragt werden können.

Suchen verwaister Dateien

Wenn eine `CTAS`- oder `INSERT INTO`-Anweisung fehlschlägt, ist es möglich, dass verwaiste Daten am Datenspeicherort belassen werden. Da Athena keine Daten (auch keine Teildaten) aus Ihrem Bucket löscht, können Sie diese Teildaten möglicherweise in nachfolgenden Abfragen lesen. Zur Suche nach verwaisten Dateien zwecks Überprüfung oder Löschung können Sie die Daten-Manifest-

Datei verwenden, die Athena zur Verfügung stellt, um die Liste der zu schreibenden Dateien zu verfolgen. Weitere Informationen finden Sie unter Identifizieren von Abfrageausgabedateien und DataManifestLocation.

INSERT INTO...SELECT

Gibt die Abfrage an, die für eine Tabelle ausgeführt werden soll, `source_table`, die festlegt, welche Zeilen in eine zweite Tabelle eingefügt werden sollen, `destination_table`. Wenn die `SELECT`-Abfrage Spalten in der `source_table` angibt, müssen die Spalten genau mit denen in der `destination_table` übereinstimmen.

Weitere Informationen zu `SELECT`-Abfragen finden Sie unter SELECT.

Syntax

```
INSERT INTO destination_table
SELECT select_query
FROM source_table_or_view
```

Beispiele

Wählen Sie alle Zeilen in der `vancouver_pageviews`-Tabelle aus und fügen Sie sie in die `canada_pageviews`-Tabelle ein:

```
INSERT INTO canada_pageviews
SELECT *
FROM vancouver_pageviews;
```

Wählen Sie nur die Zeilen in der `vancouver_pageviews`-Tabelle aus, in denen die `date`-Spalte einen Wert zwischen `2019-07-01` und `2019-07-31` hat, und fügen Sie sie dann in `canada_july_pageviews` ein:

```
INSERT INTO canada_july_pageviews
SELECT *
FROM vancouver_pageviews
WHERE date
    BETWEEN date '2019-07-01'
        AND '2019-07-31';
```

Wählen Sie die Werte in den Spalten `city` und `state` in der `cities_world`-Tabelle nur aus den Zeilen mit dem Wert `usa` in der Spalte `country` aus und fügen Sie sie in die Spalten `city` und `state` in der `cities_usa`-Tabelle ein:

```
INSERT INTO cities_usa (city,state)
SELECT city,state
FROM cities_world
    WHERE country='usa'
```

INSERT INTO...VALUES

Fügt Zeilen in eine vorhandene Tabelle ein, indem Spalten und Werte angegeben werden. Angegebene Spalten und zugehörige Datentypen müssen genau mit den Spalten und Datentypen in der Zieltabelle übereinstimmen.

> **Important**
>
> Wir raten davon ab, Zeilen mit `VALUES` einzufügen, da Athena Dateien für jede `INSERT`-Operation generiert. Dies kann dazu führen, dass viele kleine Dateien erstellt werden und die Abfrageleistung der Tabelle beeinträchtigen. Um Dateien zu identifizieren, die eine `INSERT`-Abfrage erstellt, überprüfen Sie die Daten-Manifest-Datei. Weitere Informationen finden Sie unter Arbeiten mit Abfrageergebnissen, letzten Abfragen und Ausgabedateien.

Syntax

```
INSERT INTO destination_table [(col1,col2,...)]
VALUES (col1value,col2value,...)[,
       (col1value,col2value,...)][,
       ...]
```

Beispiele

In den folgenden Beispielen hat die Tabelle "cities" drei Spalten: `id`, `city`, `state`, `state_motto`. Die `id`-Spalte hat den Typ `INT` und alle anderen Spalten haben den Typ `VARCHAR`.

Fügen Sie eine einzelne Zeile in die `cities`-Tabelle ein, wobei alle Spaltenwerte angegeben sind:

```
INSERT INTO cities
```

```
VALUES (1,'Lansing','MI','Si quaeris peninsulam amoenam circumspice')
```

Fügen Sie zwei Zeilen in die `cities`-Tabelle ein:

```
INSERT INTO cities
VALUES (1,'Lansing','MI','Si quaeris peninsulam amoenam circumspice'),
       (3,'Boise','ID','Esto perpetua')
```

DELETE

Löscht Zeilen in einer Apache-Iceberg-Tabelle. `DELETE` ist transaktionsbasiert und wird nur für Apache-Iceberg-Tabellen unterstützt.

Syntax

Verwenden Sie die folgende Syntax, um die Zeilen in einer Iceberg-Tabelle zu löschen.

```
DELETE FROM [db_name.]table_name [WHERE predicate]
```

Weitere Informationen und Beispiele finden Sie im `DELETE`-Abschnitt von Aktualisieren von Iceberg-Datentabellen.

UPDATE

Aktualisiert Zeilen in einer Apache-Iceberg-Tabelle. `UPDATE` ist transaktionsbasiert und wird nur für Apache-Iceberg-Tabellen unterstützt.

Syntax

Verwenden Sie die folgende Syntax, um die Zeilen in einer Iceberg-Tabelle zu aktualisieren.

```
UPDATE [db_name.]table_name SET xx=yy[,...] [WHERE predicate]
```

Weitere Informationen und Beispiele finden Sie im `UPDATE`-Abschnitt von Aktualisieren von Iceberg-Datentabellen.

MERGE INTO

Aktualisiert, löscht oder fügt Zeilen in eine Apache-Iceberg-Tabelle ein. Eine einzige Anweisung kann Aktionen zum Aktualisieren, Löschen und Einfügen kombinieren.

> **Note**
>
> `MERGE INTO` ist transaktionsbasiert und wird nur für Apache-Iceberg-Tabellen in Athena-Engine-Version 3 unterstützt.

Syntax

Verwenden Sie die folgende Syntax, um Zeilen in einer Iceberg-Tabelle bedingt zu aktualisieren, zu löschen oder einzufügen.

```
MERGE INTO target_table [ [ AS ]  target_alias ]
USING { source_table | query } [ [ AS ] source_alias ]
ON search_condition
when_clause [...]
```

Die *when_clause* ist eine der folgenden:

```
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` unterstützt eine beliebige Anzahl von `WHEN`-Klauseln mit unterschiedlichen `MATCHED`-Bedingungen. Die Bedingungsklauseln führen die `DELETE`, `UPDATE` oder `INSERT`-Operation in der ersten `WHEN`-Klausel aus, die durch den `MATCHED`-Zustand und die Übereinstimmungsbedingung ausgewählt wurde.

Für jede Quellzeile werden die `WHEN`-Klauseln der Reihe nach verarbeitet. Nur die erste übereinstimmende `WHEN`-Klausel wird ausgeführt. Nachfolgende Klauseln werden ignoriert. Ein Benutzerfehler wird ausgegeben, wenn eine einzelne Zeile der Zieltabelle mit mehr als einer Quellzeile übereinstimmt.

Wenn eine Quellzeile mit keiner `WHEN`-Klausel übereinstimmt und es keine `WHEN NOT MATCHED`-Klausel gibt, wird die Quellzeile ignoriert.

In WHEN-Klauseln mit UPDATE-Operationen können sich die Spaltenwertausdrücke auf jedes Feld des Ziels oder der Quelle beziehen. Im NOT MATCHED-Fall können sich die INSERT-Ausdrücke auf ein beliebiges Feld der Quelle beziehen.

Beispiel

Im folgenden Beispiel werden Zeilen aus der zweiten Tabelle mit der ersten Tabelle zusammengeführt, wenn die Zeilen in der ersten Tabelle nicht vorhanden sind. Beachten Sie, dass den in der VALUES-Klausel aufgeführten Spalten der Alias der Quelltabelle vorangestellt werden muss. Den in der INSERT-Klausel aufgeführten Zielspalten darf kein solches Präfix vorangestellt werden.

```
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)
```

Weitere MERGE INTO-Beispiele finden Sie unter Aktualisieren von Iceberg-Datentabellen.

OPTIMIZE

Optimiert Zeilen in einer Apache-Iceberg-Tabelle durch Umschreiben von Datendateien in ein optimiertes Layout basierend auf deren Größe und der Anzahl der zugeordneten Löschdateien.

> **Note**
>
> OPTIMIZE ist transaktionsbasiert und wird nur für Apache-Iceberg-Tabellen unterstützt.

Syntax

Die folgende Syntaxzusammenfassung zeigt, wie das Datenlayout für eine Iceberg-Tabelle optimiert wird.

```
OPTIMIZE [db_name.]table_name REWRITE DATA USING BIN_PACK
  [WHERE predicate]
```

Die Verdichtungsaktion wird durch die Datenmenge berechnet, die während des Umschreibungsvorgangs gescannt wurde. Die REWRITE DATA-Aktion verwendet Prädikate zur

Auswahl für Dateien, die übereinstimmende Zeilen enthalten. Wenn eine Zeile in der Datei mit dem Prädikat übereinstimmt, wird die Datei zur Optimierung ausgewählt. Um also die Anzahl der von der Verdichtungsoperation betroffenen Dateien zu steuern, können Sie eine `WHERE`-Klausel angeben.

Konfigurieren von Komprimierungs-Eigenschaften

Um die Größe der für die Verdichtung zu wählenden Dateien und die resultierende Dateigröße nach der Verdichtung zu steuern, können Sie Tabelleneigenschaftsparameter verwenden. Sie können den FESTGELEGTE TABELLENEIGENSCHAFTEN ÄNDERN-Befehl verwenden, um die zugehörigen Tabelleneigenschaften zu konfigurieren.

Weitere Informationen finden Sie auch unter

Optimieren von Iceberg-Tabellen

VACUUM

Die `VACUUM`-Anweisung führt Wartung von Apache-Iceberg-Tabellen durch, indem sie nicht mehr benötigte Datendateien entfernt.

> **Note**
>
> `VACUUM` ist transaktionsbasiert und wird nur für Apache-Iceberg-Tabellen in Athena-Engine-Version 3 unterstützt.

Syntax

Verwenden Sie die folgende Syntax, um Datendateien zu entfernen, die für eine Iceberg-Tabelle nicht mehr benötigt werden.

```
VACUUM target_table
```

Es wird empfohlen, die `VACUUM`-Anweisung für Iceberg-Tabellen auszuführen, um Datendateien zu entfernen, die nicht mehr relevant sind, und um die Metadatengröße und den Speicherverbrauch zu reduzieren. Beachten Sie, dass für die damit verbundenen Anfragen an Amazon S3 Gebühren anfallen, da die `VACUUM`-Anweisung API-Aufrufe an Amazon S3 tätigt.

> **Warning**
>
> Wenn Sie einen Snapshot-Ablauf ausführen, können Sie nicht mehr zu abgelaufenen Snapshots reisen.

`VACUUM` führt die folgenden Operationen aus:

- Entfernt Snapshots, die älter sind als die durch die `vacuum_max_snapshot_age_seconds`-Tabelleneigenschaft angegebene Zeitdauer. Standardmäßig ist diese Eigenschaft auf 432 000 Sekunden (5 Tage) festgelegt.
- Entfernt Snapshots, die nicht innerhalb des Aufbewahrungszeitraums liegen und die die in der `vacuum_min_snapshots_to_keep`-Tabelleneigenschaft angegebene Anzahl überschreiten. Der Standardwert ist 1.

  Sie können diese Tabelleneigenschaften in Ihrer `CREATE TABLE`-Anweisung angeben. Nachdem die Tabelle erstellt wurde, können Sie diese mit der FESTGELEGTE TABELLENEIGENSCHAFTEN ÄNDERN-Anweisung aktualisieren.
- Entfernt alle Metadaten und Datendateien, die aufgrund des Entfernens des Snapshots nicht mehr erreichbar sind. Sie können die Anzahl der alten Metadatendateien, die beibehalten werden sollen, konfigurieren, indem Sie die `vacuum_max_metadata_files_to_keep`-Tabelleneigenschaft festlegen. Der Standardwert lautet 100.
- Entfernt verwaiste Dateien, die älter sind als die in der `vacuum_max_snapshot_age_seconds`-Tabelleneigenschaft angegebene Zeit. Verwaiste Dateien sind Dateien im Datenverzeichnis der Tabelle, die nicht mehr Teil des Tabellenstatus sind.

Weitere Informationen zur Erstellung und Verwaltung von Apache-Iceberg-Tabellen in Athena finden Sie unter Erstellen von Iceberg-Tabellen und Verwalten von Iceberg-Tabellen.

Verwenden von EXPLAIN und EXPLAIN ANALYZE in Athena

Die `EXPLAIN`-Anweisung zeigt den logischen oder verteilten Ausführungsplan einer angegebenen SQL-Anweisung oder validiert die SQL-Anweisung. Sie können die Ergebnisse im Textformat oder in einem Datenformat für das Rendern in einem Diagramm ausgeben.

> **Note**
>
> Sie können grafische Darstellungen logischer und verteilter Pläne für Ihre Abfragen in der Athena-Konsole anzeigen, ohne die `EXPLAIN`-Dateisyntax zu verwenden. Weitere Informationen finden Sie unter Anzeigen von Ausführungsplänen für SQL-Abfragen.

Die `EXPLAIN ANALYZE`-Anweisung zeigt sowohl den verteilten Ausführungsplan einer angegebenen SQL-Anweisung als auch die Rechenkosten jeder Operation in einer SQL-Abfrage an. Sie können die Ergebnisse im Text- oder JSON-Format ausgeben.

Überlegungen und Einschränkungen

Die `EXPLAIN`- und `EXPLAIN ANALYZE`-Anweisungen in Athena unterliegen den folgenden Einschränkungen.

- Da `EXPLAIN`-Abfragen keine Daten scannen, berechnet Athena keine Gebühren. Da `EXPLAIN`-Abfragen jedoch AWS Glue aufrufen, um Tabellen-Metadaten abzurufen, können Gebühren von Glue anfallen, wenn die Anrufe das kostenlose Kontingent für Glue überschreiten.
- Weil `EXPLAIN ANALYZE`-Abfragen ausgeführt werden, scannen sie Daten und Athena berechnet die gescannte Datenmenge.
- Die in Lake Formation definierten Zeilen- oder Zellenfilterinformationen und die Abfragestatistikinformationen werden in der Ausgabe von `EXPLAIN` und `EXPLAIN ANALYZE` nicht angezeigt.

EXPLAIN-Syntax

```
EXPLAIN [ ( option [, ...]) ] statement
```

`Option` kann eine der folgenden sein:

```
FORMAT { TEXT | GRAPHVIZ | JSON }
TYPE { LOGICAL | DISTRIBUTED | VALIDATE | IO }
```

Wenn die Option `FORMAT` nicht angegeben ist, wird standardmäßig das `TEXT`-Format ausgegeben. Der `IO`-Typ stellt Informationen zu den Tabellen und Schemas bereit, die die Abfrage liest. `IO` wird nur in Athena-Engine-Version 2 unterstützt und kann nur im JSON-Format zurückgegeben werden.

EXPLAIN-ANALYZE-Syntax

Zusätzlich zu der in `EXPLAIN` enthaltenen Ausgabe enthält die `EXPLAIN ANALYZE`-Ausgabe auch Laufzeitstatistiken für die angegebene Abfrage, z. B. die CPU-Auslastung, die Anzahl der Eingabezeilen und die Anzahl der Ausgabezeilen.

```
EXPLAIN ANALYZE [ ( option [, ...]) ] statement
```

`Option` kann eine der folgenden sein:

```
FORMAT { TEXT | JSON }
```

Wenn die Option `FORMAT` nicht angegeben ist, wird standardmäßig das `TEXT`-Format ausgegeben. Weil alle Abfragen für `EXPLAIN ANALYZE DISTRIBUTED` sind, ist die Option `TYPE` für `EXPLAIN ANALYZE` nicht verfügbar.

`Anweisung` kann einer der folgenden Werte sein:

```
SELECT
CREATE TABLE AS SELECT
INSERT
UNLOAD
```

EXPLAIN-Beispiele

Die folgenden Beispiele für `EXPLAIN` entwickeln sich von den einfacheren zu den komplexeren Beispielen.

EXPLAIN-Beispiel 1: Verwenden der EXPLAIN-Anweisung, um einen Abfrageplan im Textformat anzuzeigen

Das folgende Beispiel `EXPLAIN` zeigt den Ausführungsplan für eine `SELECT`-Abfrage für Elastic-Load-Balancing-Protokolle. Das Format ist standardmäßig die Textausgabe.

```
EXPLAIN
SELECT
   request_timestamp,
   elb_name,
   request_ip
```

```
FROM sampledb.elb_logs;
```

Ergebnisse

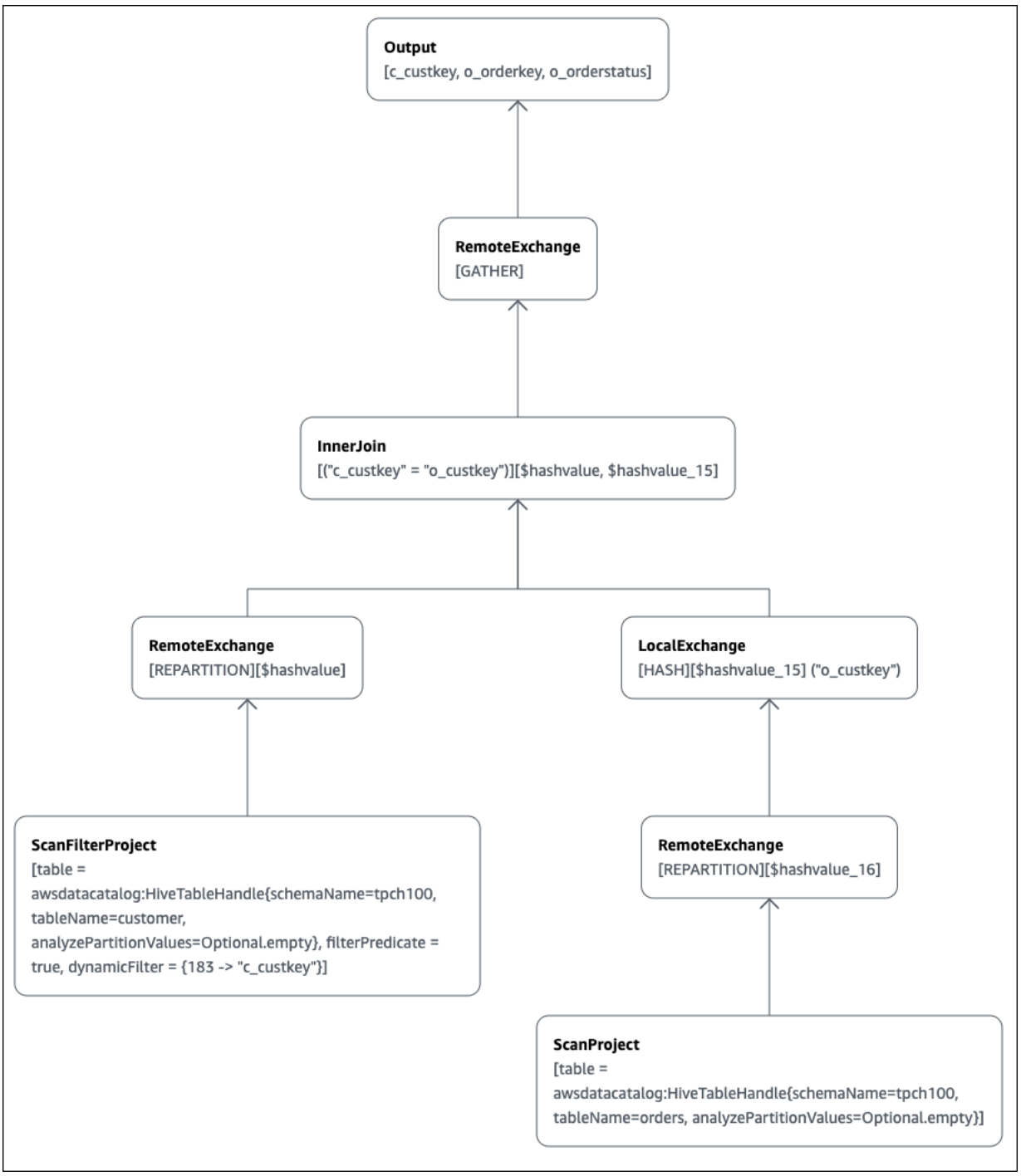
```
- Output[request_timestamp, elb_name, request_ip] => [[request_timestamp, elb_name,
 request_ip]]
    - RemoteExchange[GATHER] => [[request_timestamp, elb_name, request_ip]]
        - TableScan[awsdatacatalog:HiveTableHandle{schemaName=sampledb,
 tableName=elb_logs,
analyzePartitionValues=Optional.empty}] => [[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
```

EXPLAIN-Beispiel 2: Einen Abfrageplan grafisch darstellen

Sie können mit der Athena-Konsole einen Abfrageplan grafisch darstellen. Geben Sie eine SELECT-Anweisung wie die folgende im Athena-Abfrage-Editor ein und wählen Sie dann EXPLAIN (Erklären) aus.

```
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
```

Die Explain-Seite des Athena-Abfrage-Editors wird geöffnet und zeigt Ihnen einen verteilten Plan und einen logischen Plan für die Abfrage. Das folgende Diagramm zeigt den logischen Plan für das Beispiel.



Important

Derzeit sind einige Partitionsfilter im verschachtelten Operator-Baumdiagramm möglicherweise nicht sichtbar, obwohl Athena sie auf Ihre Abfrage anwendet. Um die Wirkung

solcher Filter zu überprüfen, führen Sie `EXPLAIN` oder `EXPLAIN ANALYZE` auf Ihre Anfrage aus und sehen Sie sich die Ergebnisse an.

Weitere Informationen zur Verwendung der Abfrageplan-Diagrammfeatures in der Athena-Konsole finden Sie unter Anzeigen von Ausführungsplänen für SQL-Abfragen.

EXPLAIN-Beispiel 3. Verwenden der EXPLAIN-Anweisung, um das Partition Pruning zu überprüfen

Wenn Sie ein Filterprädikat für einen partitionierten Schlüssel verwenden, um eine partitionierte Tabelle abzufragen, wendet das Abfragemodul das Prädikat auf den partitionierten Schlüssel an, um die Menge der gelesenen Daten zu reduzieren.

Im folgenden Beispiel wird eine `EXPLAIN`-Abfrage verwendet, um die Partitionsbereinigung für eine `SELECT`-Abfrage in einer partitionierten Tabelle zu überprüfen. Zuerst erstellt eine `CREATE TABLE`-Anweisung die `tpch100.orders_partitioned`-Tabelle. Die Tabelle ist nach Spalte `o_orderdate` partitioniert.

```
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://<your_s3_bucket>/<your_directory_path>/'
```

Die `tpch100.orders_partitioned`-Tabelle hat mehrere Partitionen auf `o_orderdate`, wie der Befehl `SHOW PARTITIONS` zeigt.

```
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
```

Die folgende EXPLAIN-Abfrage überprüft die Partitionsbereinigung für die angegebene SELECT-Anweisung.

```
EXPLAIN
SELECT
   o_orderkey,
   o_custkey,
   o_orderdate
FROM tpch100.orders_partitioned
WHERE o_orderdate = '1995'
```

Ergebnisse

```
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
```

Der fett gedruckte Text im Ergebnis zeigt, dass das Prädikat o_orderdate = '1995' auf den PARTITION_KEY angewendet wurde.

EXPLAIN-Beispiel 4. Verwenden einer EXPLAIN-Abfrage, um die Join-Reihenfolge und den Join-Typ zu überprüfen

Die folgende EXPLAIN-Abfrage überprüft die Join-Reihenfolge und den Join-Typ der SELECT-Anweisung. Verwenden Sie eine Abfrage wie diese, um die Speicherauslastung der Abfrage zu untersuchen, um die Wahrscheinlichkeit eines EXCEEDED_LOCAL_MEMORY_LIMIT-Fehlers zu verringern.

```
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
```

Ergebnisse

```
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
```

Die Beispielabfrage wurde für eine bessere Leistung zu einem Cross-Join optimiert. Die Ergebnisse zeigen, dass `tpch100.orders` als BROADCAST-Verteilungstyp verteilt wird. Dies impliziert, dass die `tpch100.orders`-Tabelle an alle Knoten verteilt wird, die den Join-Vorgang ausführen. Der BROADCAST-Verteilungstyp erfordert, dass alle gefilterten Ergebnisse der `tpch100.orders`-Tabelle in den Speicher jedes Knotens passen, der die Join-Operation ausführt.

Die `tpch100.customer`-Tabelle ist jedoch kleiner als `tpch100.orders`. Da `tpch100.customer` weniger Speicher benötigt, können Sie die Abfrage in BROADCAST `tpch100.customer` statt in `tpch100.orders` umschreiben. Dies verringert die Wahrscheinlichkeit, dass die Abfrage den EXCEEDED_LOCAL_MEMORY_LIMIT-Fehler erhält. Diese Strategie setzt folgende Punkte voraus:

- Das `tpch100.customer.c_custkey` ist in der `tpch100.customer`-Tabelle eindeutig.
- Zwischen `tpch100.customer` und `tpch100.orders` besteht eine Eins-zu-Viele-Mappingbeziehung.

Im folgenden Beispiel wird die neu geschriebene Abfrage dargestellt.

```
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
```

EXPLAIN-Beispiel 5. Verwenden einer EXPLAIN-Abfrage zum Entfernen von Prädikaten, die keine Auswirkungen haben

Sie können eine EXPLAIN-Abfrage verwenden, um die Wirksamkeit des Filterns von Prädikaten zu überprüfen. Sie können die Ergebnisse verwenden, um Prädikate zu entfernen, die keine Auswirkung haben, wie im folgenden Beispiel.

```
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
```

Ergebnisse

```
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
```

Das `filterPredicate` in den Ergebnissen zeigt, dass der Optimierer die ursprünglichen drei Prädikate zu zwei Prädikaten zusammengeführt und ihre Anwendungsreihenfolge geändert hat.

```
filterPredicate = (("c_custkey" = 1500) AND ("c_custkey" = CAST(("random"() * 1E3) AS
 int)))
```

Da die Ergebnisse zeigen, dass das Prädikat `AND c.c_custkey BETWEEN 1000 AND 2000` keine Auswirkung hat, können Sie dieses Prädikat entfernen, ohne die Abfrageergebnisse zu ändern.

Informationen zu den in den Ergebnissen von EXPLAIN-Abfragen verwendeten Begriffen finden Sie unter Die Ergebnisse der Athena-EXPLAIN-Anweisung verstehen.

EXPLAIN-ANALYZE-Beispiele

In den nachstehenden Beispielen wird ein Beispiel für `EXPLAIN ANALYZE`-Abfragen und Ausgaben gezeigt.

EXPLAIN-ANALYZE-Beispiel 1. Verwenden von EXPLAIN ANALYZE, um einen Abfrageplan und Rechenkosten im Textformat anzuzeigen

Das folgenden Beispiel `EXPLAIN ANALYZE` zeigt den Ausführungsplan und die Rechenkosten für eine `SELECT`-Abfrage in CloudFront-Protokollen. Das Format ist standardmäßig die Textausgabe.

```
EXPLAIN ANALYZE SELECT FROM cloudfront_logs LIMIT 10
```

Ergebnisse

```
 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
```

EXPLAIN-ANALYZE-Beispiel 2. Verwenden von EXPLAIN ANALYZE, um einen Abfrageplan im JSON-Format anzuzeigen

Das folgende Beispiel zeigt den Ausführungsplan und die Rechenkosten für eine SELECT-Abfrage in CloudFront-Protokollen. Im Beispiel wird JSON als Ausgabeformat angegeben.

```
EXPLAIN ANALYZE (FORMAT JSON) SELECT * FROM cloudfront_logs LIMIT 10
```

Ergebnisse

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

```
}
```

Weitere Ressourcen

Weitere Informationen finden Sie in den folgenden Ressourcen.

- Die Ergebnisse der Athena-EXPLAIN-Anweisung verstehen
- Anzeigen von Ausführungsplänen für SQL-Abfragen
- Anzeigen von Statistiken und Ausführungsdetails für abgeschlossene Abfragen
- Trino-Dokumentation `EXPLAIN`
- Trino-Dokumentation `EXPLAIN ANALYZE`
- Optimieren Sie die Leistung von Verbundabfragen mit EXPLAIN und EXPLAIN ANALYZE in Amazon Athena im AWS-Big-Data-Blog.

Die Ergebnisse der Athena-EXPLAIN-Anweisung verstehen

Dieses Thema bietet eine kurze Anleitung zu den operativen Begriffen, die in den Athena-`EXPLAIN`-Anweisungsergebnissen verwendet werden.

EXPLAIN-Anweisungs-Ausgabetypen

`EXPLAIN`-Ausgabetypen können zwei Typen haben:

- Logischer Plan – Zeigt den logischen Plan an, den die SQL-Engine zum Ausführen einer Anweisung verwendet. Die Syntax für diese Option lautet `EXPLAIN` oder `EXPLAIN (TYPE LOGICAL)`.
- Verteilter Plan – Zeigt einen Ausführungsplan in einer verteilten Umgebung an. Die Ausgabe zeigt Fragmente, die Verarbeitungsschritte sind. Jedes Planfragment wird von einem oder mehreren Knoten verarbeitet. Daten können zwischen den Knoten ausgetauscht werden, die die Fragmente verarbeiten. Die Syntax für diese Option lautet `EXPLAIN (TYPE DISTRIBUTED)`

  In der Ausgabe für einen verteilten Plan werden Fragmente (Verarbeitungsphasen) durch *`number`* [*`fragment_type`*] `Fragment` angegeben, wobei *`number`* eine nullbasierte ganze Zahl ist und *`fragment_type`* angibt, wie das Fragment von den Knoten ausgeführt wird. Die Fragmenttypen, die Einblicke in das Layout des Datenaustauschs geben, werden in der folgenden Tabelle beschrieben.

Fragment-Typen für verteilte Pläne

Fragment-Typ	Beschreibung
SINGLE	Das Fragment wird auf einem einzelnen Knoten ausgeführt.
HASH	Das Fragment wird auf einer festen Anzahl von Knoten ausgeführt. Die Eingabedaten werden mit einer Hash-Funktion verteilt.
ROUND_ROBIN	Das Fragment wird auf einer festen Anzahl von Knoten ausgeführt. Die Eingabedaten werden auf Round-Robin-Weise verteilt.
BROADCAST	Das Fragment wird auf einer festen Anzahl von Knoten ausgeführt. Die Eingabedaten werden an alle Knoten übertragen.
SOURCE	Das Fragment wird auf Knoten ausgeführt, auf denen Eingabesplits zugegriffen wird.

Exchange

Die austauschbezogenen Begriffe beschreiben, wie Daten zwischen Worker-Knoten ausgetauscht werden. Übertragungen können entweder lokal oder remote erfolgen.

LocalExchange [***exchange_type***]

Überträgt Daten lokal innerhalb von Worker-Knoten für verschiedene Phasen einer Abfrage. Der Wert für *exchange_type* kann einer der logischen oder verteilten Austausch-Typen sein, wie später in diesem Abschnitt beschrieben.

RemoteExchange [***exchange_type***]

Überträgt Daten zwischen Worker-Knoten für verschiedene Phasen einer Abfrage. Der Wert für *exchange_type* kann einer der logischen oder verteilten Austausch-Typen sein, wie später in diesem Abschnitt beschrieben.

Logische Austausch-Typen

Die folgenden Austausch-Typen beschreiben Aktionen, die während der Austauschphase eines logischen Plans durchgeführt wurden.

- **GATHER** – Ein einzelner Worker-Knoten sammelt die Ausgabe von allen anderen Worker-Knoten. Beispielsweise sammelt die letzte Stufe einer Auswahlabfrage Ergebnisse von allen Knoten und schreibt die Ergebnisse in Amazon S3.
- **REPARTITION** – Sendet die Zeilendaten an einen bestimmten Worker basierend auf dem Partitionierungsschema, das für den nächsten Operator erforderlich ist.
- **REPLICATE** – Kopiert die Zeilendaten in alle Worker.

Verteilte Austausch-Typen

Die folgenden Austauschtypen geben das Layout der Daten an, wenn sie zwischen Knoten in einem verteilten Plan ausgetauscht werden.

- **HASH** – Der Austausch verteilt Daten an mehrere Ziele mithilfe einer Hash-Funktion.
- **SINGLE** – Der Austausch verteilt Daten an ein einzelnes Ziel.

Scanning

Die folgenden Bedingungen beschreiben, wie Daten während einer Abfrage gescannt werden.

TableScan

Scannt die Quelldaten einer Tabelle von Amazon S3 oder einem Apache-Hive-Connector und wendet die aus dem Filterprädikat generierte Partitionsbereinigung an.

ScanFilter

Scannt die Quelldaten einer Tabelle von Amazon S3 oder einem Hive-Connector und wendet die Partitionsbereinigung an, die aus dem Filterprädikat und aus zusätzlichen Filterprädikaten generiert wurde, die nicht durch die Partitionsbereinigung angewendet wurden.

ScanFilterProject

Scannt zunächst die Quelldaten einer Tabelle von Amazon S3 oder einem Hive-Connector und wendet die Partitionsbereinigung an, die aus dem Filterprädikat und aus zusätzlichen Filterprädikaten generiert wurde, die nicht durch die Partitionsbereinigung angewendet wurden. Ändert dann das Speicherlayout der Ausgabedaten in eine neue Projektion, um die Leistung späterer Phasen zu verbessern.

Join

Verknüpft Daten zwischen zwei Tabellen. Verknüpfungen können nach Verknüpfungstyp und nach Verteilungstyp kategorisiert werden.

JOIN-Typen

Verknüpfungstypen definieren die Art und Weise, in der der Join-Vorgang ausgeführt wird.

CrossJoin – Erzeugt das kartesische Produkt der beiden verbundenen Tabellen.

InnerJoin – Wählt Datensätze aus, die übereinstimmende Werte in beiden Tabellen aufweisen.

LeftJoin – Wählt alle Datensätze aus der linken Tabelle und die übereinstimmenden Datensätze aus der rechten Tabelle aus. Wenn keine Übereinstimmung auftritt, ist das Ergebnis auf der rechten Seite NULL.

RightJoin – Wählt alle Datensätze aus der rechten Tabelle und die passenden Datensätze aus der linken Tabelle aus. Wenn keine Übereinstimmung auftritt, ist das Ergebnis auf der linken Seite NULL.

FullJoin – Wählt alle Datensätze aus, bei denen eine Übereinstimmung in den linken oder rechten Tabellendatensätzen vorliegt. Die verknüpfte Tabelle enthält alle Datensätze aus beiden Tabellen und füllt NULLs für fehlende Übereinstimmungen auf beiden Seiten aus.

> **Note**
>
> Aus Leistungsgründen kann das Abfragemodul eine Join-Abfrage in einen anderen Join-Typ umschreiben, um dieselben Ergebnisse zu erzielen. Beispielsweise kann eine Inner-Join-Abfrage mit einem Prädikat für eine Tabelle in ein `CrossJoin` umgeschrieben werden. Dadurch wird das Prädikat in die Scanphase der Tabelle verschoben, sodass weniger Daten gescannt werden.

Join-Verteilungs-Typen

Verteilungstypen definieren, wie Daten zwischen Worker-Knoten ausgetauscht werden, wenn der Join-Vorgang ausgeführt wird.

Partitioniert – Sowohl die linke als auch die rechte Tabelle sind über alle Worker-Knoten hinweg hashpartitioniert. Partitionierte Verteilung verbraucht weniger Speicher in jedem Knoten. Partitionierte Verteilung kann viel langsamer sein als replizierte Joins. Partitionierte Joins eignen sich, wenn Sie zwei große Tabellen verbinden.

Repliziert – Eine Tabelle wird über alle Worker-Knoten hinweg hashpartitioniert, und die andere Tabelle wird auf alle Worker-Knoten repliziert, um den Join-Vorgang auszuführen. Die replizierte Verteilung kann viel schneller sein als partitionierte Joins, verbraucht jedoch mehr Speicher in jedem Worker-Knoten. Wenn die replizierte Tabelle zu groß ist, kann der Worker-Knoten einen Fehler aufgrund von fehlendem Speicherplatz auftreten. Replizierte Joins sind geeignet, wenn eine der verknüpften Tabellen klein ist.

PREPARE

Erstellt eine SQL-Anweisung mit dem Namen `statement_name`, die zu einem späteren Zeitpunkt ausgeführt werden soll. Die Anweisung kann Parameter enthalten, die durch Fragezeichen dargestellt werden. Um Werte für die Parameter bereitzustellen und die vorbereitete Anweisung auszuführen, verwenden Sie EXECUTE.

Syntax

```
PREPARE statement_name FROM statement
```

Die folgende Tabelle beschreibt diese Parameter.

Parameter	Beschreibung
`statement_name`	Der Name der zu erstellenden Anweisung. Der Name muss innerhalb der Arbeitsgruppe eindeutig sein.
`statement`	`SELECT`-, `CTAS`- oder `INSERT INTO`-Abfrage.

> **Note**
>
> Die maximale Anzahl von vorbereiteten Anweisungen in einer Arbeitsgruppe liegt bei 1 000.

Beispiele

Das folgende Beispiel bereitet eine ausgewählte Abfrage ohne Parameter vor.

```
PREPARE my_select1 FROM
SELECT * FROM nation
```

Das folgende Beispiel bereitet eine ausgewählte Abfrage mit Parametern vor. Die Werte für `productid` und `quantity` werden von der `USING`-Klausel einer `EXECUTE`-Aussage bereitgestellt:

```
PREPARE my_select2 FROM
SELECT order FROM orders WHERE productid = ? and quantity < ?
```

Im folgenden Beispiel wird eine Einfügeabfrage vorbereitet.

```
PREPARE my_insert FROM
INSERT INTO cities_usa (city, state)
SELECT city, state
FROM cities_world
WHERE country = ?
```

Weitere Informationen finden Sie auch unter

Abfragen mit vorbereiteten Anweisungen

EXECUTE

DEALLOCATE PREPARE

INSERT INTO

EXECUTE

Führt eine vorbereitete Anweisung mit dem Namen `statement_name` aus. Parameterwerte für die Fragezeichen in der vorbereiteten Anweisung sind in der`USING`-Klausel in einer kommagetrennten Liste definiert. Um eine vorbereitete Anweisung zu erstellen, verwenden Sie PREPARE.

Syntax

```
EXECUTE statement_name [ USING parameter1[, parameter2, ... ] ]
```

Beispiele

Das folgende Beispiel bereitet eine Abfrage ohne Parameter vor und führt sie aus.

```
PREPARE my_select1 FROM
SELECT name FROM nation
EXECUTE my_select1
```

Das folgende Beispiel bereitet eine Abfrage mit einem einzigen Parameter vor und führt sie aus.

```
PREPARE my_select2 FROM
SELECT * FROM "my_database"."my_table" WHERE year = ?
EXECUTE my_select2 USING 2012
```

Dies entspricht:

```
SELECT * FROM "my_database"."my_table" WHERE year = 2012
```

Das folgende Beispiel bereitet eine Abfrage mit zwei Parametern vor und führt sie aus.

```
PREPARE my_select3 FROM
SELECT order FROM orders WHERE productid = ? and quantity < ?
EXECUTE my_select3 USING 346078, 12
```

Weitere Informationen finden Sie auch unter

Abfragen mit vorbereiteten Anweisungen

PREPARE

INSERT INTO

DEALLOCATE PREPARE

Entfernt die vorbereitete Anweisung mit dem angegebenen Namen aus den vorbereiteten Anweisungen in der aktuellen Arbeitsgruppe.

Syntax

```
DEALLOCATE PREPARE statement_name
```

Beispiele

Im folgenden Beispiel wird die vorbereitete Anweisung `my_select1` aus der aktuellen Arbeitsgruppe entfernt.

```
DEALLOCATE PREPARE my_select1
```

Weitere Informationen finden Sie auch unter

Abfragen mit vorbereiteten Anweisungen

PREPARE

UNLOAD

Schreibt Abfrageergebnisse aus einer SELECT-Anweisung in das angegebene Datenformat. Zu den unterstützten Formaten für UNLOAD gehören Apache Parquet, ORC, Apache Avro und JSON. CSV ist das einzige Ausgabeformat, das vom Athena-SELECTBefehl unterstützt wird. Sie können jedoch den UNLOAD Befehl verwenden, der eine Vielzahl von Ausgabeformaten unterstützt, um Ihre SELECT Abfrage einzuschließen und ihre Ausgabe in eines der von UNLOAD unterstützten Formate umzuschreiben.

Obwohl Sie die CTAS-Anweisung verwenden können, um Daten in anderen Formaten als CSV auszugeben, erfordern diese Anweisungen auch die Erstellung einer Tabelle in Athena. Die UNLOAD-Anweisung ist nützlich, wenn Sie die Ergebnisse einer SELECT-Abfrage in einem Nicht-CSV-Format ausgeben möchten, aber die zugehörige Tabelle nicht benötigen. Beispielsweise kann eine nachgelagerte Anwendung erfordern, dass die Ergebnisse einer SELECT-Abfrage im JSON-Format vorliegen, und Parquet oder ORC können einen Leistungsvorteil gegenüber CSV bieten, wenn Sie die Ergebnisse der SELECT-Abfrage für zusätzliche Analysen verwenden möchten.

Überlegungen und Einschränkungen

Beachten Sie bei der Verwendung der UNLOAD-Anweisung in Athena die folgenden Punkte:

- Keine globale Reihenfolge von Dateien – UNLOAD-Ergebnisse werden parallel in mehrere Dateien geschrieben. Wenn die SELECT-Abfrage in der UNLOAD-Anweisung eine Sortierreihenfolge angibt, wird der Inhalt jeder Datei sortiert, aber die Dateien werden nicht relativ zueinander sortiert.
- Verwaiste Daten nicht gelöscht – Im Falle eines Fehlers versucht Athena nicht, verwaiste Daten zu löschen. Dieses Verhalten ist das gleiche wie bei CTAS und INSERT INTO-Anweisungen.
- Maximale Partitionen – Die maximale Anzahl von Partitionen, die mit UNLOAD verwendet werden können, beträgt 100.
- Metadaten- und Manifestdateien– Athena generiert für jede UNLOAD-Abfrage eine Metadatendatei und eine Datenmanifestdatei. Das Manifest verfolgt die Dateien, die die Abfrage geschrieben hat. Beide Dateien werden in Ihrem Athena-Abfrageergebnisspeicherort in Amazon S3 gespeichert. Weitere Informationen finden Sie unter Identifizieren von Abfrageausgabedateien.

- Verschlüsselung – UNLOAD-Ausgabedateien werden gemäß der für Amazon S3 verwendeten Verschlüsselungskonfiguration verschlüsselt. Um die Verschlüsselungskonfiguration für die Verschlüsselung Ihres UNLOAD Ergebnisses einzurichten, können Sie die EncryptionConfiguration API verwenden.
- Vorbereitete Anweisungen – UNLOADkann mit vorbereiteten Anweisungen verwendet werden. Informationen zu vorbereiteten Anweisungen in Athena finden Sie unter Verwenden von parametrisierten Abfragen.
- Service Quotas – UNLOAD verwendet DML-Abfragekontingente. Kontigentinformationen finden Sie unter Service Quotas.
- Erwarteter Bucket-Eigentümer – Die Einstellung für den erwarteten Bucket-Eigentümer gilt nicht für den Amazon-S3-Speicherort, der in der UNLOAD-Abfrage angegeben wurde. Die erwartete Bucket-Eigentümereinstellung gilt nur für den Amazon-S3-Ausgabespeicherort, den Sie für Athena-Abfrageergebnisse angeben. Weitere Informationen finden Sie unter Angeben eines Speicherorts des Abfrageergebnisses mithilfe der Athena-Konsole.

Syntax

Die UNLOAD-Anweisung verwendet die folgende Syntax.

```
UNLOAD (SELECT col_name[, ...] FROM old_table)
TO 's3://my_athena_data_location/my_folder/'
WITH ( property_name = 'expression' [, ...] )
```

> **Note**
>
> Das TO-Ziel muss einen Standort in Amazon S3 angeben, der keine Daten enthält. Bevor die UNLOAD-Abfrage in den angegebenen Speicherort schreibt, überprüft sie, ob der Bucket-Speicherort leer ist. Da UNLOAD keine Daten an den angegebenen Speicherort schreibt, wenn der Speicherort bereits Daten enthält, überschreibt UNLOAD keine vorhandenen Daten. Um einen Bucket-Speicherort als Ziel für UNLOAD wiederzuverwenden, löschen Sie die Daten am Bucket-Speicherort und führen Sie die Abfrage dann erneut aus.

Parameter

Die möglichen Werte für *property_name* sind wie folgt.

format = '*file_format*'

Erforderlich Gibt das Dateiformat der Ausgabe an. Mögliche Werte für *file_format* sind ORC, PARQUET, AVRO, JSON oder TEXTFILE.

Kompression = '*compression_format*'

Optional. Diese Option ist spezifisch für die Formate ORC und Parquet. Für ORC ist der Standardwert zlib und für Parquet der Standardwert gzip. Informationen über unterstützte Komprimierungsformate finden Sie unter Unterstützung der Athena-Komprimierung.

> **Note**
>
> Diese Option gilt nicht für das Format AVRO. Athena verwendet gzip für JSON- und TEXTFILE-Formate.

compression_level = *compression_level*

Optional. Die Komprimierungsstufe, die für die ZSTD-Komprimierung verwendet werden soll. Diese Eigenschaft gilt nur für die ZSTD-Komprimierung. Weitere Informationen finden Sie unter Verwendung von ZSTD-Komprimierungsstufen in Athena.

field_delimiter = '*delimiter*'

Optional. Gibt ein Feldtrennzeichen aus einem einzigen Zeichen für Dateien in CSV, TSV und anderen Textformaten an. Das folgende Beispiel verwendet ein Komma als Trennzeichen:

```
WITH (field_delimiter = ',')
```

Derzeit werden mehrzeilige Feldtrennzeichen nicht unterstützt. Wenn Sie kein Feldtrennzeichen angeben, wird das Oktalzeichen \001 (^A) verwendet.

partitioned_by = ARRAY[*col_name*[,...]]

Optional. Eine Array-Liste der Spalten, nach denen die Ausgabe partitioniert ist.

> **Note**
>
> Stellen Sie in Ihrer SELECT-Anweisung sicher, dass die Namen der partitionierten Spalten in Ihrer Spaltenliste an letzter Stelle stehen.

Beispiele

Im folgenden Beispiel wird die Ausgabe einer SELECT-Abfrage im JSON-Format an den Amazon-S3-Speicherort s3://*DOC-EXAMPLE-BUCKET*/unload_test_1/ geschrieben.

```
UNLOAD (SELECT * FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/unload_test_1/'
WITH (format = 'JSON')
```

Im folgenden Beispiel wird die Ausgabe einer SELECT-Abfrage im Parquet-Format mithilfe der Snappy-Komprimierung geschrieben.

```
UNLOAD (SELECT * FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/'
WITH (format = 'PARQUET',compression = 'SNAPPY')
```

Im folgenden Beispiel werden vier Spalten im Textformat geschrieben, wobei die Ausgabe durch die letzte Spalte partitioniert ist.

```
UNLOAD (SELECT name1, address1, comment1, key1 FROM table1)
TO 's3://DOC-EXAMPLE-BUCKET/ partitioned/'
WITH (format = 'TEXTFILE', partitioned_by = ARRAY['key1'])
```

Im folgenden Beispiel werden die Abfrageergebnisse unter Verwendung des Parquet-Dateiformats, der ZSTD-Komprimierung und der ZSTD-Komprimierungsstufe 4 an den angegebenen Speicherort entladen.

```
UNLOAD (SELECT * FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/'
WITH (format = 'PARQUET', compression = 'ZSTD', compression_level = 4)
```

Weitere Informationen finden Sie auch unter

- Vereinfachen Sie Ihre ETL- und ML-Pipelines mithilfe des Amazon Athena UNLOAD-Features im AWS -Big-Data-Blog.

Funktionen in Amazon Athena

Weitere Informationen über Athena-Engine-Versionen finden Sie im Versionsreferenz der Athena-Engine. Eine Liste der Zeitzonen, die mit dem `AT TIME ZONE`-Operator verwendet werden können, finden Sie unter Unterstützte Zeitzonen.

Athena-Engine-Version 3

Funktionen in Athena-Engine-Version 3 basieren auf Trino. Informationen zu Funktionen, Operatoren und Ausdrücken von Trino finden Sie unter Funktionen und Operatoren und in den folgenden Unterabschnitten der Trino-Dokumentation.

- Aggregate
- Array
- Binary
- Bitweise
- Farbe
- Vergleich
- Bedingt
- Konvertierung
- Datum und Uhrzeit
- Dezimal
- Geodaten
- HyperLogLog
- IP-Adresse
- JSON
- Lambda
- Logisch
- Machine Learning
- Zuordnung
- Math (Mathematik)
- Quantil-Digest
- Regulärer Ausdruck
- Sitzung

- Digest festlegen
- Zeichenfolge
- System (System)
- Tabelle
- Teradata
- T-Digest
- URL
- UUID
- Window

Athena-Engine-Version 2

Funktionen in Athena-Engine-Version 2 basieren auf Presto 0.217. Die räumlichen Funktionen in Athena-Engine-Version 2 finden Sie unter Geodaten-Funktionen in Athena-Engine-Version 2.

> **Note**
>
> Versionsspezifische Dokumentation für Presto 0.217-Funktionen ist nicht mehr verfügbar. Hinweise zu aktuellen Presto-Funktionen, -Operatoren und Ausdrücken finden Sie unter Presto-Funktionen und -Operatoren oder besuchen Sie die Links zu den Unterkategorien in diesem Abschnitt.

- Logische Operatoren
- Vergleichsfunktionen und Operatoren
- Bedingte Ausdrücke
- Konvertierungs-Funktionen
- Mathematische Funktionen und Operatoren
- Bitweise-Funktionen
- Dezimale Funktionen und Operatoren
- Zeichenfolgen-Funktionen und -Operatoren
- Binäre Funktionen
- Datums- und Zeitfunktionen und -Operatoren
- Funktionen für reguläre Ausdrücke

- JSON-Funktionen und -Operatoren
- URL-Funktionen
- Aggregationsfunktionen
- Fensterfunktionen
- Farb-Funktionen
- Array-Funktionen und -Operatoren
- Zuordnungs-Funktionen und -Operatoren
- Lambda-Ausdrücke und -Funktionen
- Teradata-Funktionen

Unterstützte Zeitzonen

Sie können den Operator `AT TIME ZONE` in einer `SELECT timestamp`-Anweisung verwenden, um die Zeitzone für den zurückgegebenen Zeitstempel anzugeben, wie im folgenden Beispiel:

```
SELECT timestamp '2012-10-31 01:00 UTC' AT TIME ZONE 'America/Los_Angeles' AS la_time;
```

Ergebnisse

```
la_time

2012-10-30 18:00:00.000 America/Los_Angeles
```

Die folgende Liste enthält die Zeitzonen, die mit dem `AT TIME ZONE`-Operator in Athena verwendet werden können. Weitere Funktionen und Beispiele für die Zeitzone finden Sie unter Funktionen und Beispiele für Zeitzonen.

```
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
```

Funktionen und Beispiele für Zeitzonen

Im Folgenden finden Sie weitere Funktionen und Beispiele für Zeitzonen.

- at_timezone(***timestamp***, ***zone***) – Gibt den Wert von *timestamp* in der entsprechenden Ortszeit für *zone* zurück.

  Beispiel

  ```
  SELECT at_timezone(timestamp '2021-08-22 00:00 UTC', 'Canada/Newfoundland')
  ```

  Ergebnis

  ```
  2021-08-21 21:30:00.000 Canada/Newfoundland
  ```

- timezone_hour(***timestamp***) – Gibt die Stunde des Zeitzonenoffset aus dem Zeitstempel als einen `bigint` zurück.

  Beispiel

  ```
  SELECT timezone_hour(timestamp '2021-08-22 04:00 UTC' AT TIME ZONE 'Canada/
  Newfoundland')
  ```

  Ergebnis

  ```
  -2
  ```

- timezone_minute(***timestamp***) – Gibt die Minute des Zeitzonenoffset aus dem *timestamp* als einen `bigint` zurück.

  Beispiel

  ```
  SELECT timezone_minute(timestamp '2021-08-22 04:00 UTC' AT TIME ZONE 'Canada/
  Newfoundland')
  ```

  Ergebnis

```
-30
```

- with_timezone(***timestamp***, ***zone***) – Gibt den Zeitstempel mit Zeitzone aus deu angegebenen Werten für *timestamp* (Zeitstmepel) und *zone* zurück.

  Beispiel

  ```
  SELECT with_timezone(timestamp '2021-08-22 04:00', 'Canada/Newfoundland')
  ```

  Ergebnis

  ```
  2021-08-22 04:00:00.000 Canada/Newfoundland
  ```

DDL-Anweisungen

Nutzen Sie die folgenden DDL-Anweisungen direkt in Athena.

Die Athena-Abfrage-Engine basiert auf HiveQL DDL.

Athena unterstützt nicht alle DDL-Anweisungen und es gibt einige Unterschiede zwischen HiveQL DDL und Athena DDL. Weitere Informationen finden Sie in den Referenzthemen in diesem Abschnitt und Nicht unterstützte DDLs.

Themen

- Nicht unterstützte DDLs
- 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
- CREATE VIEW
- DESCRIBE
- DESCRIBE VIEW
- DROP DATABASE
- DROP TABLE
- DROP VIEW
- MSCK REPAIR TABLE
- SHOW COLUMNS
- SHOW CREATE TABLE
- SHOW CREATE VIEW
- SHOW DATABASES
- SHOW PARTITIONS
- SHOW TABLES
- SHOW TBLPROPERTIES
- SHOW VIEWS

Nicht unterstützte DDLs

Die folgenden DDL-Anweisungen werden von Athena nicht unterstützt:

- 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
- ANZEIGEN VON STATISTIKEN
- SHOW TRANSACTIONS
- START TRANSACTION
- UNLOCK DATABASE
- UNLOCK TABLE

ALTER DATABASE SET DBPROPERTIES

Erstellt eine oder mehrere Eigenschaften für eine Datenbank. Sie können `DATABASE` und `SCHEMA` verwenden, beides hat dieselbe Bedeutung.

Syntax

```
ALTER {DATABASE|SCHEMA} database_name
  SET DBPROPERTIES ('property_name'='property_value' [, ...] )
```

Parameter

SET DBPROPERTIES ('property_name'='property_value' [, ...]

Gibt eine Eigenschaft (oder mehrere) für die Datenbank mit dem Namen `property_name` an. Der Wert für jede dieser Eigenschaften wird entsprechend als `property_value` festgelegt. Sollte `property_name` bereits vorhanden sein, wird der alte Wert mit `property_value` überschrieben.

Beispiele

```
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

Fügt einer vorhandenen Tabelle eine oder mehrere Spalten hinzu. Bei Verwendung der optionalen Syntax `PARTITION` werden Partitionsmetadaten aktualisiert.

Syntax

```
ALTER TABLE table_name
  [PARTITION
   (partition_col1_name = partition_col1_value
   [,partition_col2_name = partition_col2_value][,...])]
  ADD COLUMNS (col_name data_type)
```

Parameter

PARTITION (partition_col_name = partition_col_value [,...])

Erstellt eine Partition mit den von Ihnen angegebenen Spaltenname/Wert-Kombinationen. Umschließen Sie `partition_col_value` nur dann mit Anführungszeichen, wenn es sich beim Datentyp der Spalte um eine Zeichenfolge handelt.

ADD COLUMNS (col_name data_type [,col_name data_type,…])

Fügt Spalten nach vorhandenen Spalten, jedoch nicht vor Partitionsspalten hinzu.

Beispiele

```
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)
```

Hinweise

- Um im Navigationsbereich des Athena-Abfrage-Editors nach der Ausführung von `ALTER TABLE ADD COLUMNS` eine neue Tabellenspalte anzuzeigen, aktualisieren Sie die Tabellenliste im Editor manuell und erweitern die Tabelle anschließend erneut.
- `ALTER TABLE ADD COLUMNS` funktioniert nicht für Spalten mit dem `date`-Datentyp Um dieses Problem zu umgehen, verwenden Sie den `timestamp`-Datentyp

ALTER TABLE ADD PARTITION

Erstellt eine oder mehrere Partitionsspalten für die Tabelle. Jede Partition besteht aus einer oder mehreren unterschiedlichen Spaltenname/Wert-Kombinationen. Für jede angegebene Kombination wird ein eigenes Datenverzeichnis erstellt. Dies kann die Abfrageleistung in einigen Fällen verbessern. Partitionierte Spalten sind nicht in den Tabellendaten selbst vorhanden. Falls Sie also einen Spaltennamen angeben, der mit einer Spalte in der Tabelle übereinstimmt, wird ein Fehler ausgegeben. Weitere Informationen finden Sie unter Daten in Athena partitionieren.

In Athena müssen eine Tabelle und ihre Partitionen die gleichen Datenformate verwenden, ihre Schemata können aber unterschiedlich sein. Weitere Informationen finden Sie unter Updates in Tabellen mit Partitionen.

Informationen zu den Berechtigungen auf Ressourcenebene, die in IAM-Richtlinien (einschließlich `glue:CreatePartition`) erforderlich sind, finden Sie unter AWS Glue -API-Berechtigungen: Referenz zu Aktionen und Ressourcen und Differenzierter Zugriff auf Datenbanken und Tabellen in AWS Glue Data Catalog. Informationen zur Fehlerbehebung bei Berechtigungen bei der Verwendung von Athena finden Sie im Abschnitt Berechtigungen des Themas Athena-Fehlerbehebung.

Syntax

```
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']
```

```
  [,...]
```

Parameter

Wenn Sie eine Partition hinzufügen, geben Sie ein oder mehrere Spaltenname/Wert-Paare für die Partition und den Amazon-S3-Pfad an, in dem sich die Datendateien für diese Partition befinden.

[IF NOT EXISTS]

Führt in dem Fall, dass bereits eine Partition mit derselben Definition vorhanden ist, zu einer Fehlerunterdrückung.

PARTITION (partition_col_name = partition_col_value [,...])

Erstellt eine Partition mit den von Ihnen angegebenen Spaltenname/Wert-Kombinationen. Binden Sie `partition_col_value` nur als Zeichenfolgenzeichen ein, wenn der Datentyp der Spalte eine Zeichenfolge ist.

[LOCATION 'location']

Gibt das Verzeichnis an, in dem die in der vorherigen Anweisung definierten Partitionen gespeichert werden. Die `LOCATION`-Klausel ist optional, wenn die Daten Partitionierung (`pk1=v1/pk2=v2/pk3=v3`) im Hive-Stil verwenden. Bei der Partitionierung im Hive-Stil wird der vollständige Amazon-S3-URI automatisch aus dem Speicherort der Tabelle, den Partitionsschlüsselnamen und den Partitionsschlüsselwerten erstellt. Weitere Informationen finden Sie unter Daten in Athena partitionieren.

Überlegungen

Amazon Athena legt kein bestimmtes Limit für die Anzahl der Partitionen fest, die Sie in einer einzelnen `ALTER TABLE ADD PARTITION` DDL-Anweisung hinzufügen können. Wenn Sie jedoch eine beträchtliche Anzahl von Partitionen hinzufügen müssen, sollten Sie erwägen, den Vorgang in kleinere Batches aufzuteilen, um potenzielle Leistungsprobleme zu vermeiden. Das folgende Beispiel verwendet aufeinanderfolgende Befehle, um Partitionen einzeln hinzuzufügen, und verwendet, um das Hinzufügen von Duplikaten `IF NOT EXISTS` zu vermeiden.

```
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')
```

Beachten Sie bei der Arbeit mit Partitionen in Athena auch die folgenden Punkte:

- Athena unterstützt zwar das Abfragen von AWS Glue Tabellen mit 10 Millionen Partitionen, Athena kann jedoch nicht mehr als 1 Million Partitionen in einem einzigen Scan lesen.
- Um Ihre Abfragen zu optimieren und die Anzahl der gescannten Partitionen zu reduzieren, sollten Sie Strategien wie das Bereinigen von Partitionen oder die Verwendung von Partitionsindizes in Betracht ziehen.
- Wenn Sie dies nicht verwenden AWS Glue Data Catalog, beträgt die maximale Anzahl von Partitionen pro Tabelle 20.000. Sie können eine Kontingenterhöhung beantragen.

Weitere Überlegungen zur Arbeit mit Partitionen in Athena finden Sie unterDaten in Athena partitionieren.

Beispiele

Das folgende Beispiel fügt einer Tabelle für partitionierte Daten im Hive-Stil eine einzelne Partition hinzu.

```
ALTER TABLE orders ADD
  PARTITION (dt = '2016-05-14', country = 'IN');
```

Das folgende Beispiel fügt einer Tabelle für partitionierte Daten im Hive-Stil mehrere Partitionen hinzu.

```
ALTER TABLE orders ADD
  PARTITION (dt = '2016-05-31', country = 'IN')
  PARTITION (dt = '2016-06-01', country = 'IN');
```

Wenn die Tabelle nicht für partitionierte Daten im Hive-Stil bestimmt ist, ist die `LOCATION`-Klausel erforderlich und sollte der vollständige Amazon-S3-URI für das Präfix sein, das die Daten der Partition enthält.

```
ALTER TABLE orders ADD
  PARTITION (dt = '2016-05-31', country = 'IN') LOCATION 's3://mystorage/path/to/
INDIA_31_May_2016/'
  PARTITION (dt = '2016-06-01', country = 'IN') LOCATION 's3://mystorage/path/to/
INDIA_01_June_2016/';
```

Um Fehler zu ignorieren, wenn die Partition bereits existiert, verwenden Sie die `IF NOT EXISTS`-Klausel, wie folgt.

```
ALTER TABLE orders ADD IF NOT EXISTS
  PARTITION (dt = '2016-05-14', country = 'IN');
```

Null Byte **_$folder$**-Datei

Wenn du eine `ALTER TABLE ADD PARTITION`-Anweisung ausführst und fälschlicherweise eine Partition angibts, die bereits vorhanden ist, und einen falschen Ort für Simple Storage Service (Amazon S3), Null-Byte-Platzhalterdateien des Formats *`partition_value`*`_$folder$` werden in Simple Storage Service (Amazon S3) erstellt. Sie müssen diese Dateien manuell entfernen.

Um dies zu verhindern, verwenden Sie die `ADD IF NOT EXISTS`-Syntax in Ihrer `ALTER TABLE ADD PARTITION`-Anweisung wie folgt.

```
ALTER TABLE table_name ADD IF NOT EXISTS PARTITION […]
```

ALTER TABLE DROP PARTITION

Löscht eine oder mehrere angegebene Partitionen für die benannte Tabelle.

Syntax

```
ALTER TABLE table_name DROP [IF EXISTS] PARTITION (partition_spec) [, PARTITION
 (partition_spec)]
```

Parameter

[IF EXISTS]

Unterdrückt die Fehlermeldung, wenn die angegebene Partition nicht vorhanden ist.

PARTITION (partition_spec)

Jede `partition_spec` gibt eine Spaltenname/Spaltenwert-Kombination in Form von `partition_col_name = partition_col_value [,...]` an.

Beispiele

```
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');
```

Hinweise

Die `ALTER TABLE DROP PARTITION`-Anweisung bietet keine einzige Syntax zum Löschen aller Partitionen auf einmal oder unterstützt Filterkriterien, um einen Bereich von zu löschenden Partitionen anzugeben.

Um dieses Thema zu umgehen, können Sie die AWS Glue-API GetPartitions- und BatchDeletePartition-Aktionen in Skriptsprache verwenden. Die `GetPartitions`-Aktion unterstützt komplexe Filterausdrücke wie die in einem SQL-`WHERE`-Ausdruck. Nachdem Sie `GetPartitions` verwendet haben, um eine gefilterte Liste der zu löschenden Partitionen zu erstellen, können Sie mit der `BatchDeletePartition`-Aktion die Partitionen in Batches von 25 löschen.

> **Important**
>
> Aufgrund eines bekannten Problems werden alle Partitionen für die Tabelle gelöscht, wenn eine ungültige Partition für die `ALTER TABLE DROP PARTITION`-Anweisung in AWS Glue angegeben wird. Mit der folgenden Anweisung werden beispielsweise alle Partitionen für die Tabelle *`my_table`* gelöscht, obwohl die angegebene Partition nicht existiert. Um das Problem zu umgehen, stellen Sie sicher, dass Sie die Partitionsinformationen korrekt eingeben, bevor Sie die `ALTER TABLE DROP PARTITION`-Anweisung ausführen.
>
> ```
> ALTER TABLE my_table DROP IF EXISTS PARTITION(zzz='');
> ```

ALTER TABLE RENAME PARTITION

Benennt die Partitionsspalte `partition_spec` für die Tabelle namens `table_name` in `new_partition_spec` um.

Weitere Informationen zur Partitionierung finden Sie unter Daten in Athena partitionieren.

Syntax

```
ALTER TABLE table_name PARTITION (partition_spec) RENAME TO PARTITION
 (new_partition_spec)
```

Parameter

PARTITION (partition_spec)

Jede `partition_spec` gibt eine Spaltenname/Spaltenwert-Kombination in Form von `partition_col_name = partition_col_value [,...]` an.

Beispiele

```
ALTER TABLE orders
PARTITION (dt = '2014-05-14', country = 'IN') RENAME TO PARTITION (dt = '2014-05-15',
 country = 'IN');
```

ALTER TABLE REPLACE COLUMNS

Entfernt alle vorhandenen Spalten aus einer mit LazySimpleSerDe erstellten Tabelle und ersetzt sie durch den angegebenen Spaltensatz. Bei Verwendung der optionalen Syntax `PARTITION` werden Partitionsmetadaten aktualisiert. Sie können `ALTER TABLE REPLACE COLUMNS` auch verwenden, um Spalten zu löschen, indem Sie nur die Spalten angeben, die Sie behalten möchten.

Syntax

```
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, ...])
```

Parameter

PARTITION (partition_col_name = partition_col_value [,...])

Gibt eine Partition mit den von Ihnen angegebenen Spaltennamen/Wert-Kombinationen an. Umschließen Sie `partition_col_value` nur dann mit Anführungszeichen, wenn es sich beim Datentyp der Spalte um eine Zeichenfolge handelt.

REPLACE COLUMNS (col_name data_type [,col_name data_type,…])

Ersetzt vorhandene Spalten durch die angegebenen Spaltennamen und Datentypen.

Hinweise

- Um die Änderung der Tabellenspalten im Navigationsbereich des Athena-Abfrage-Editors anzuzeigen, nachdem Sie `ALTER TABLE REPLACE COLUMNS` ausgeführt haben, müssen Sie möglicherweise die Tabellenliste im Editor manuell aktualisieren und die Tabelle dann erneut erweitern.
- `ALTER TABLE REPLACE COLUMNS` funktioniert nicht für Spalten mit dem `date`-Datentyp Um dieses Problem zu umgehen, verwenden Sie stattdessen den `timestamp`-Datentyp in der Tabelle.
- Beachten Sie, dass die Syntax `ALTER TABLE table-name REPLACE COLUMNS` sein muss, auch wenn Sie nur eine einzelne Spalte ersetzen, mit Spalten im Plural. Sie müssen nicht nur die Spalte angeben, die Sie ersetzen möchten, sondern auch die Spalten, die Sie beibehalten möchten. Andernfalls werden die Spalten, die Sie nicht angeben, gelöscht. Diese Syntax und dieses Verhalten leitet sich von Apache Hive DDL ab. Weitere Informationen finden Sie unter Spalten hinzufügen/ersetzen in der Apache-Dokumentation.

Beispiel

Im folgenden Beispiel wird die Tabelle `names_cities`, die mit dem LazySimpleSerde verwendet, sie hat drei Spalten mit den Namen `col1`, `col2` und `col3`. Alle Spalten sind vom Typ `string`. Um die Spalten in der Tabelle anzuzeigen, verwendet der folgende Befehl die SHOW COLUMNS-Anweisung.

```
SHOW COLUMNS IN names_cities
```

Ergebnis der Abfrage:

```
col1
col2
col3
```

Der folgende `ALTER TABLE REPLACE COLUMNS`-Befehl ersetzt die Spaltennamen durch `first_name`, `last_name`, und `city`. Die zugrunde liegenden Quelldaten sind nicht betroffen.

```
ALTER TABLE names_cities
REPLACE COLUMNS (first_name string, last_name string, city string)
```

Um das Ergebnis zu testen, wird `SHOW COLUMNS` erneut ausgeführt.

```
SHOW COLUMNS IN names_cities
```

Ergebnis der Abfrage:

```
first_name
last_name
city
```

Eine andere Möglichkeit, die neuen Spaltennamen anzuzeigen, besteht darin, eine Vorschau der Tabelle im Athena-Abfrage-Editor anzuzeigen oder Ihre eigene SELECT-Abfrage auszuführen.

ALTER TABLE SET LOCATION

Ändert den Speicherort für die Tabelle namens `table_name`und optional eine Partition mit `partition_spec`.

Syntax

```
ALTER TABLE table_name [ PARTITION (partition_spec) ] SET LOCATION 'new location'
```

Parameter

PARTITION (partition_spec)

Gibt die Partition mit `partition_spec`-Parametern an, dessen Speicherort Sie ändern möchten. `partition_spec` gibt eine Name/Wert-Kombination einer Spalte im Formular aus `partition_col_name = partition_col_value`.

SET LOCATION 'new location'

Gibt den neuen Speicherort an. Es muss sich dabei um einen Amazon S3-Speicherort handeln. Weitere Informationen zur Syntax finden Sie unter Tabellenort in Amazon S3.

Beispiele

```
ALTER TABLE customers PARTITION (zip='98040', state='WA') SET LOCATION 's3://mystorage/
custdata/';
```

ALTER TABLE SET TBLPROPERTIES

Fügt benutzerdefinierte oder vordefinierte Metadateneigenschaften zu einer Tabelle hinzu und legt deren zugewiesene Werte fest. Um die Eigenschaften in einer Tabelle anzuzeigen, verwenden Sie den Befehl SHOW TBLPROPERTIES.

Apache Hive Verwaltete Tabellen werden nicht unterstützt, so dass die Einstellung `'EXTERNAL'='FALSE'` keine Auswirkungen hat.

Syntax

```
ALTER TABLE table_name SET TBLPROPERTIES ('property_name' = 'property_value' [ , ... ])
```

Parameter

SET TBLPROPERTIES ('property_name' = 'property_value' [, ...])

Gib die Metadateneigenschaften, die als `property_name` hinzugefügt werden, und den jeweiligen Wert als `property value` an. Wenn `property_name` bereits vorhanden ist, wird der Wert auf den neu angegebenen `property_value` festgelegt.

Die folgenden vordefinierten Tabelleneigenschaften haben besondere Verwendungszwecke.

Vordefinierte Eigenschaft	Beschreibung
`classification`	Gibt den Datentyp für AWS Glue an. Mögliche Werte sind `csv`, `parquet`, `orc`, `avro`, oder `json`. Tabellen, die für Athena in der CloudTrail-Konsole erstellt wurden, fügen `cloudtrail` als einen Wert für die Eigenschaft `classification` hinzu. Weitere Informationen finden Sie im Abschnitt TBLPROPERTIES von CREATE TABLE.
`has_encrypted_data`	Gibt an, ob der durch `LOCATION` angegebene Datensatz verschlüsselt ist. Weitere Informationen finden Sie im Abschnitt TBLPROPERTIES von CREATE TABLE und Erstellen von Tabellen basierend auf verschlüsselten Datensätzen in Amazon S3.
`orc.compress`	Gibt ein Komprimierungsformat für Daten im ORC-Format an. Weitere Informationen finden Sie unter ORC SerDe.
`parquet.compression`	Legt ein Komprimierungsformat für Daten im Parquet-Format fest. Weitere Informationen finden Sie unter Parquet SerDe.

Vordefinierte Eigenschaft	Beschreibung
`write.compression`	Gibt ein Komprimierungsformat für Daten in den Textdatei- oder JSON-Formaten an. Verwenden Sie für die Formate Parquet und ORC die Eigenschaften `parquet.compression` bzw. `orc.compress` .
`compression_level`	Gibt eine zu verwendende Komprimierungsstufe an. Diese Eigenschaft gilt nur für die ZSTD-Komprimierung. Mögliche Werte liegen zwischen 1 und 22. Der Standardwert ist 3. Weitere Informationen finden Sie unter Verwendung von ZSTD-Komprimierungsstufen in Athena.
`projection.*`	Benutzerdefinierte Eigenschaften, die in der Partitionsprojektion verwendet werden, damit Athena weiß, welche Partitionsmuster zu erwarten sind, wenn eine Abfrage für eine Tabelle ausgeführt wird. Weitere Informationen finden Sie unter Partitionsprojektion mit Amazon Athena.
`skip.header.line.count`	Ignoriert Kopfzeilen in Daten, wenn Sie eine Tabelle definieren. Weitere Informationen finden Sie unter Ignorieren von Kopfzeilen.
`storage.location.template`	Gibt eine benutzerdefinierte Amazon-S3-Pfadvorlage für projizierte Partitionen an. Weitere Informationen finden Sie unter Einrichten der Partitionsprojektion.

Beispiele

Im folgenden Beispiel wird den Tabelleneigenschaften eine Kommentarnotiz hinzugefügt.

```
ALTER TABLE orders
SET TBLPROPERTIES ('notes'="Please don't drop this table.");
```

Im folgenden Beispiel wird die Tabelle `existing_table` so geändert, dass das Parquet-Dateiformat mit ZSTD-Komprimierung und ZSTD-Komprimierungsstufe 4 verwendet wird.

```
ALTER TABLE existing_table
SET TBLPROPERTIES ('parquet.compression' = 'ZSTD', 'compression_level' = 4)
```

CREATE DATABASE

Erstellt eine Datenbank. Sie können `DATABASE` oder `SCHEMA` gleichwertig verwenden. Sie haben dieselbe Bedeutung.

> **Note**
>
> Für ein Beispiel für das Erstellen einer Datenbank, das Erstellen einer Tabelle und das Ausführen einer `SELECT`-Abfrage auf dem Tisch in Athena siehe Erste Schritte.

Syntax

```
CREATE {DATABASE|SCHEMA} [IF NOT EXISTS] database_name
  [COMMENT 'database_comment']
  [LOCATION 'S3_loc']
  [WITH DBPROPERTIES ('property_name' = 'property_value') [, ...]]
```

Parameter

[IF NOT EXISTS]

Führt dazu, dass der Fehler unterdrückt wird, wenn eine Datenbank mit dem Namen `database_name` bereits vorhanden ist.

[COMMENT database_comment]

Legt den Metadatenwert für die integrierte Metadateneigenschaft namens `comment` und den Wert, den Sie für `database_comment` angeben, fest. In AWS Glue wird der `COMMENT`-Inhalt in das `Description`-Feld der Datenbankeigenschaften geschrieben.

[LOCATION S3_loc]

Legt den Speicherort fest, an dem Datenbankdateien und Metaspeicher als `S3_loc` abgelegt werden. Der Speicherort muss ein Amazon S3-Speicherort sein.

[WITH DBPROPERTIES ('property_name' = 'property_value') [, ...]]

Ermöglicht Ihnen, benutzerdefinierte Metadateneigenschaften für die Datenbankdefinition festzulegen.

Beispiele

```
CREATE DATABASE clickstreams;
```

```
CREATE DATABASE IF NOT EXISTS clickstreams
  COMMENT 'Site Foo clickstream data aggregates'
  LOCATION 's3://myS3location/clickstreams/'
  WITH DBPROPERTIES ('creator'='Jane D.', 'Dept.'='Marketing analytics');
```

Anzeigen von Datenbankeigenschaften

Um die Datenbankeigenschaften für eine Datenbank anzuzeigen, die Sie in AWSDataCatalog mit `CREATE DATABASE` erstellen, können Sie den AWS CLI-Befehl `aws glue get-database` wie im folgenden Beispiel verwenden:

```
aws glue get-database --name <your-database-name>
```

In der JSON-Ausgabe sieht das Ergebnis wie folgt aus:

```
{
    "Database": {
        "Name": "<your-database-name>",
        "Description": "<your-database-comment>",
        "LocationUri": "s3://<your-database-location>",
        "Parameters": {
            "<your-database-property-name>": "<your-database-property-value>"
        },
        "CreateTime": 1603383451.0,
        "CreateTableDefaultPermissions": [
            {
                "Principal": {
                    "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                },
                "Permissions": [
                    "ALL"
                ]
            }
        ]
    }
}
```

Weitere Informationen zur AWS CLI finden Sie im AWS Command Line Interface-Benutzerhandbuch.

CREATE TABLE

Erstellt eine Tabelle mit dem Namen und den Parametern, den bzw. die Sie angeben.

> **Note**
>
> Diese Seite enthält zusammenfassende Referenzinformationen. Weitere Informationen zum Erstellen von Tabellen in Athena und ein Beispiel für eine `CREATE TABLE`-Anweisung finden Sie unter Erstellen von Tabellen in Athena. Für ein Beispiel für das Erstellen einer Datenbank, das Erstellen einer Tabelle und das Ausführen einer `SELECT`-Abfrage auf dem Tisch in Athena siehe Erste Schritte.

Syntax

```
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://bucket_name/[folder]/']
 [TBLPROPERTIES ( ['has_encrypted_data'='true | false',]
 ['classification'='aws_glue_classification',] property_name=property_value [, ...] ) ]
```

Parameter

EXTERNAL

Gibt an, dass die Tabelle auf einer Datendatei basiert, die in Amazon S3 an dem von Ihnen angegebenen `LOCATION` vorhanden ist. Verwenden Sie außer beim Erstellen von Iceberg-Tabellen immer das `EXTERNAL`-Schlüsselwort. Wenn Sie `CREATE TABLE` ohne das `EXTERNAL`-Schlüsselwort für Nicht-Iceberg-Tabellen verwenden, gibt Athena einen Fehler aus. Wenn Sie eine externe Tabelle erstellen, müssen die referenzierten Daten mit dem Standardformat oder dem Format übereinstimmen, das Sie über die Klauseln `ROW FORMAT`, `STORED AS` und `WITH SERDEPROPERTIES` festlegen.

[IF NOT EXISTS]

Dieser Parameter prüft, ob bereits eine Tabelle mit demselben Namen vorhanden ist. Ist dies der Fall, gibt der Parameter TRUE zurück und Amazon Athena bricht die CREATE TABLE-Aktion ab. Da eine Stornierung erfolgt, bevor Athena den Datenkatalog aufruft, gibt es kein AWS CloudTrail Ereignis aus.

[db_name.]table_name

Gibt einen Namen für die zu erstellende Tabelle an. Der optionale Parameter db_name gibt die Datenbank an, in der die Tabelle gespeichert ist. Ist nichts angegeben, wird von der aktuellen Datenbank ausgegangen. Schließen Sie table_name in Anführungszeichen ein, wenn der Tabellenname Zahlen enthält, zum Beispiel "table123". Wenn table_name mit einem Unterstrich beginnt, verwenden Sie Backticks, beispielsweise `_mytable`. Sonderzeichen (außer Unterstriche) werden nicht unterstützt.

Bei Athena-Tabellennamen wird die Groß-/Kleinschreibung berücksichtigt. Wenn Sie mit Apache Spark arbeiten, beachten Sie, dass Spark Tabellennamen in Kleinbuchstaben erfordert.

[(col_name data_type [COMMENT col_comment] [, ...])]

Gibt den Datentyp und den Namen jeder zu erstellenden Spalten an. Spaltennamen lassen außer Unterstrichen (_) keine Sonderzeichen zu. Falls col_name mit einem Unterstrich beginnt, schließen Sie den Spaltennamen in Backticks ein, beispielsweise `_mycolumn`.

Beim Wert data_type kann es sich um den folgenden Typ handeln:

- boolean – Die Werte sind true und false.
- tinyint – Eine 8-Bit signierte Ganzzahl im Zweierkomplement-Format mit einem Mindestwert von -2^7 und einem Höchstwert von 2^7-1.
- smallint – Eine 16-Bit signierte Ganzzahl im Zweierkomplement-Format mit einem Mindestwert von -2^15 und einem Höchstwert von 2^15-1.
- int – Benutzen Sie in DDL-Abfragen (Data Definition Language) wie CREATE TABLE das int-Schlüsselwort, um eine Ganzzahl darzustellen. Verwenden Sie in anderen Abfragen das Schlüsselwort integer, wobei integer als 32-Bit-Wert mit Vorzeichen im Zweierkomplementformat mit einem Mindestwert von -2^31 und einem Höchstwert von 2^31-1 dargestellt wird. Im JDBC-Treiber wird integer zurückgegeben, um Kompatibilität mit den geschäftlichen Analyseanwendungen zu gewährleisten.
- bigint – Eine 64-Bit signierte Ganzzahl im Zweierkomplement-Format mit einem Mindestwert von -2^63 und einem Höchstwert von 2^63-1.

- `double` – Eine signierte 64-Bit-Gleitkommazahl mit doppelter Genauigkeit. Der Bereich liegt zwischen 4,94065645841246544e-324d bis 1,79769313486231570e+308d, positiv oder negativ. `double` folgt dem IEEE-Standard für Gleitkommaarithmetik (IEEE 754).
- `float` – Eine signierte 32-Bit-Gleitkommazahl mit einfacher Genauigkeit. Der Bereich liegt zwischen 1,40129846432481707e-45 bis 3,40282346638528860e+38, positiv oder negativ. `float` folgt dem IEEE-Standard für Gleitkommaarithmetik (IEEE 754). Entspricht dem `real` in Presto. Verwenden Sie in Athena `float` in DDL-Anweisungen wie `CREATE TABLE` und `real` in SQL-Funktionen wie `SELECT CAST`. Der AWS Glue Crawler gibt Werte in zurück `float`und Athena übersetzt die `float` Typen `real` und intern (siehe 5. Juni 2018 Versionshinweise).
- `decimal [ (`*`precision`*`, `*`scale`*`) ]`, wobei *`precision`* die Gesamtanzahl der Stellen und *`scale`* (optional) die Anzahl der Nachkommastellen ist. Der Standardwert ist 0. Verwenden Sie z. B. diese Definitionen: `decimal(11,5)`, `decimal(15)`. Die maximale Wert der *`Genauigkeit`* ist 38 und der maximale Wert für die *`Skalierung`* beträgt 38.

  Um Dezimalwerte als Literale anzugeben, z. B. bei der Auswahl von Zeilen mit einem bestimmten Dezimalwert im DDL-Ausdruck einer Abfrage, legen Sie als Typdefinition `decimal` fest und listen Sie die Dezimalwerte als Literalwert (in einfachen Anführungszeichen) in Ihrer Abfrage auf, wie in diesem Beispiel: `decimal_value = decimal '0.12'`.
- `char` – Zeichendaten mit fester Länge, die zwischen 1 und 255 Zeichen liegen muss, z. B `char(10)`. Weitere Informationen finden Sie unter CHAR-Hive-Datentyp.
- `varchar` – Zeichendaten mit variabler Länge, die zwischen 1 und 65535 Zeichen liegen muss, z. B `varchar(10)`. Weitere Informationen finden Sie unter VARCHAR-Hive-Datentyp.
- `string` – Ein Zeichenfolgenliteral, das in einfache oder doppelte Anführungszeichen eingeschlossen ist.

  > **Note**
  >
  > Nicht-Zeichenfolgen-Datentypen können nicht zu `string` in Athena umgewandelt werden. Wandeln Sie sie stattdessen zu `varchar` um.

- `binary` – (für Daten in Parquet)
- `date` – Ein Datum im ISO-Format, z. B. *`YYYY`*`-`*`MM`*`-`*`DD`*. Beispiel: `date '2008-09-15'` Eine Ausnahme ist OpenCSV,SerDe das die Anzahl der Tage verwendet, die seit dem 1. Januar 1970 verstrichen sind. Weitere Informationen finden Sie unter OpenCSVSerDe für CSV-Verarbeitung.

- `timestamp`– Datum und Uhrzeit in einem `java.sql.Timestamp`-kompatiblen Format bis zu einer maximalen Auflösung von Millisekunden, wie *`yyyy`*`-`*`MM`*`-`*`dd HH`*`:`*`mm`*`:`*`ss`*`[.f...]`. Beispiel: `timestamp '2008-09-15 03:04:05.324'` Eine Ausnahme ist OpenCSV,SerDe das `TIMESTAMP` Daten im numerischen UNIX-Format verwendet (z. B. `1579059880000`). Weitere Informationen finden Sie unter OpenCSVSerDe für CSV-Verarbeitung.
- `array` < data_type >
- `map` < primitive_type, data_type >
- `struct` < col_name : data_type [COMMENT col_comment] [, ...] >

[COMMENT table_comment]

Erstellt die Tabelleneigenschaft `comment` und füllt diese mit dem von Ihnen angegebenen `table_comment`.

[PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]

Erstellt eine partitionierte Tabelle mit einer oder mehreren Partitionsspalten, bei denen `col_name`, `data_type` und `col_comment` angegeben sind. Eine Tabelle kann eine oder mehrere Partitionen umfassen. Diese bestehen aus einem eindeutigen Spaltennamen und einer Wertekombination. Für jede angegebene Kombination wird ein eigenes Datenverzeichnis erstellt. Dies kann die Abfrageleistung in einigen Fällen verbessern. In den Tabellendaten selbst sind keine partitionierten Spalten vorhanden. Wenn Sie einen Wert für `col_name` verwenden, der mit einer Tabellenspalte identisch ist, erhalten Sie einen Fehler. Weitere Informationen finden Sie auf der Seite zum Partitionieren von Daten.

> **Note**
>
> Nach dem Erstellen einer Tabelle mit Partitionen führen Sie eine weitere Abfrage aus, die aus der Klausel MSCK REPAIR TABLE besteht, um die Partitionsmetadaten aufzufrischen, beispielsweise `MSCK REPAIR TABLE cloudfront_logs;`. Verwenden Sie für Partitionen, die nicht Hive-kompatibel sind, ALTER TABLE ADD PARTITION zum Laden von Partitionen, sodass Sie die Daten abfragen können.

[CLUSTERED BY (col_name, col_name, ...) INTO num_buckets BUCKETS]

Unterteilt die Daten in den angegebenen `col_name`-Spalten mit oder ohne Partitionierung in Datenteilmengen, die als Buckets bezeichnet werden. Der `num_buckets`-Parameter gibt die

Anzahl der zu erstellenden Buckets an. Bucketing kann die Leistung einiger Abfragen auf großen Datensätzen verbessern.

[ROW FORMAT row_format]

Gibt das Zeilenformat der Tabelle und deren zugrunde liegenden Quelldaten an, wenn zutreffend. Für `row_format` können Sie mit der Klausel `DELIMITED` ein oder mehrere Trennzeichen angeben, oder alternativ die Klausel `SERDE` verwenden, wie unten beschrieben. Wenn `ROW FORMAT` ausgelassen oder angegeben `ROW FORMAT DELIMITED` wird, wird ein SerDe nativer verwendet.

- [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]

  Nur mit Hive 0.13 verfügbar und wenn das STORED AS-Dateiformat `TEXTFILE` ist.

--ODER--

- SERDE 'serde_name' [WITH SERDEPROPERTIES ("property_name" = "property_value", "property_name" = "property_value" [, ...])]

  Die `serde_name` gibt die SerDe zu verwendende an. Mit der -`WITH SERDEPROPERTIES`Klausel können Sie eine oder mehrere benutzerdefinierte Eigenschaften angeben, die von der zugelassen werden SerDe.

[STORED AS Dateiformat]

Gibt das Dateiformat für Tabellendaten an. Wenn nichts angegeben ist, wird standardmäßig `TEXTFILE` verwendet. Optionen für `file_format` sind:

- SEQUENCEFILE
- TEXTFILE
- RCFILE
- ORC
- PARQUET
- AVRO

- ION
- INPUTFORMAT input_format_classname OUTPUTFORMAT output_format_classname

[LOCATION 's3://bucket_name/[folder]/']

Gibt den Speicherort der zugrunde liegenden Daten in Amazon S3 an, aus denen die Tabelle erstellt wurde. Der Speicherpfad muss ein Bucket-Name oder ein Bucket-Name sowie ein oder mehrere Ordner sein. Wenn Sie Partitionen verwenden, geben Sie den Stamm der partitionierten Daten an. Weitere Informationen zum Speicherort der Tabelle finden Sie unter Tabellenspeicherort in Amazon S3. Weitere Informationen zum Datenformat und zu den Berechtigungen finden Sie unter Voraussetzungen für Tabellen in Athena und Daten in Amazon S3.

Verwenden Sie einen abschließenden Schrägstrich für Ihren Ordner oder Bucket. Verwenden Sie keine Dateinamen oder glob-Zeichen.

Verwenden:

`s3://mybucket/`

`s3://mybucket/folder/`

`s3://mybucket/folder/anotherfolder/`

Verwenden Sie nicht:

`s3://path_to_bucket`

`s3://path_to_bucket/*`

`s3://path_to-bucket/mydatafile.dat`

[TBLPROPERTIES (['has_encrypted_data'='true | false',] ['classification'='classification_value',] property_name=property_value [, ...])]

Gibt zusätzlich zu den vordefinierten Tabelleneigenschaften wie `"comment"` benutzerdefinierte Schlüssel/Wert-Paare in Form von Metadaten für die Tabellendefinition an.

has_encrypted_data – Athena verfügt über eine integrierte Eigenschaft, `has_encrypted_data`. Setzen Sie diese Eigenschaft auf `true`, um anzugeben, dass das zugrunde liegende Dataset, das durch `LOCATION` festgelegt ist, verschlüsselt ist. Wenn dies ausgelassen wird und die Einstellungen der Arbeitsgruppe clientseitige Einstellungen nicht überschreiben, wird `false` angenommen. Wenn nichts angegeben oder die Eigenschaft auf `false` gesetzt ist, schlägt die

Abfrage fehl, wenn die zugrunde liegenden Daten verschlüsselt sind. Weitere Informationen finden Sie unter Verschlüsselung im Ruhezustand.

Klassifizierung – Tabellen, die für Athena in der CloudTrail Konsole erstellt wurden, fügen `cloudtrail` als Wert für die -`classification`Eigenschaft hinzu. Um ETL-Aufträge auszuführen, AWS Glue müssen Sie eine Tabelle mit der `classification` Eigenschaft erstellen, um den Datentyp für AWS Glue als `csv`, `parquet`, `orcavro`, oder anzugeben`json`. Beispiel: `'classification'='csv'` ETL-Aufträge schlagen fehl, wenn Sie diese Eigenschaft nicht festlegen. Sie können diese nachträglich über AWS Glue -Konsole, die API oder die CLI angeben. Weitere Informationen finden Sie unter Verwenden von AWS Glue Aufträgen für ETL mit Athena und Autorisieren von Aufträgen in AWS Glue im AWS Glue -Entwicklerhandbuch.

compression_level – Die Eigenschaft `compression_level` gibt die zu verwendende Komprimierungsstufe an. Diese Eigenschaft gilt nur für die ZSTD-Komprimierung. Mögliche Werte liegen zwischen 1 und 22. Der Standardwert ist 3. Weitere Informationen finden Sie unter Verwendung von ZSTD-Komprimierungsstufen in Athena.

Weitere Informationen zu anderen Tabelleneigenschaften finden Sie unter ALTER TABLE SET TBLPROPERTIES.

Weitere Informationen zum Erstellen von Tabellen finden Sie unter Erstellen von Tabellen in Athena.

CREATE TABLE AS

Erstellt eine neue Tabelle mit den Ergebnissen einer SELECT-Abfrage. Um eine leere Tabelle zu erstellen, verwenden Sie CREATE TABLE. `CREATE TABLE AS` kombiniert eine `CREATE TABLE`-DDL-Anweisung mit einer `SELECT`-DML-Anweisung und enthält daher technisch gesehen sowohl DDL als auch DML. Beachten Sie, dass CTAS-Abfragen in Athena, obwohl `CREATE TABLE AS` sie hier mit anderen DDL-Anweisungen gruppiert ist, für Service Quotas als DML behandelt werden. Informationen zu Service Quotas für Athena finden Sie unter Service Quotas.

> **Note**
>
> Für CTAS-Anweisungen gilt die erwartete Bucket-Eigentümereinstellung nicht für den Speicherort der Zieltabelle in Amazon S3. Die erwartete Bucket-Eigentümereinstellung gilt nur für den Amazon-S3-Ausgabespeicherort, den Sie für Athena-Abfrageergebnisse angeben. Weitere Informationen finden Sie unter Angeben eines Speicherorts des Abfrageergebnisses mithilfe der Athena-Konsole.

Weitere Informationen zu `CREATE TABLE AS`, die über den Rahmen dieses Referenzthemas hinausgehen, finden Sie unter Erstellen einer Tabelle aus Abfrageergebnissen (CTAS).

Themen

Syntax

```
CREATE TABLE table_name
[ WITH ( property_name = expression [, ...] ) ]
AS query
[ WITH [ NO ] DATA ]
```

Wobei gilt:

WITH (property_name = expression [, ...])

Eine Liste der optionalen CTAS-Tabelleneigenschaften, von denen einige für das Datenspeicherformat spezifisch sind. Siehe CTAS-Tabelleneigenschaften.

query

Eine SELECT-Abfrage, die verwendet wird, um eine neue Tabelle zu erstellen.

> **Important**
>
> Wenn Sie eine Abfrage mit Partitionen erstellen möchten, geben Sie die Namen der partitionierten Spalten am Ende der Liste in der Liste der Spalten in der `SELECT` Anweisung an.

[WITH [NO] DATA]

Wenn `WITH NO DATA` verwendet wird, wird eine neue leere Tabelle mit demselben Schema wie die ursprüngliche Tabelle erstellt.

> **Note**
>
> Um Spaltenüberschriften in Ihre Abfrageergebnis-Ausgabe einzubeziehen, können Sie eine einfache `SELECT`-Abfrage statt einer CTAS-Abfrage verwenden. Sie können die Ergebnisse aus Ihrem Speicherort für Abfrageergebnisse abrufen oder die Ergebnisse direkt über die Athena-Konsole herunterladen. Weitere Informationen finden Sie unter Arbeiten mit Abfrageergebnissen, letzten Abfragen und Ausgabedateien.

CTAS-Tabelleneigenschaften

Jede CTAS-Tabelle in Athena verfügt über eine Liste der optionalen CTAS-Tabelleneigenschaften, die Sie mit `WITH (property_name = expression [, ...] )` angeben. Informationen zur Verwendung dieser Parameter finden Sie unter Beispiele für CTAS-Abfragen.

`WITH (property_name = expression [, ...], )`

`table_type = ['HIVE', 'ICEBERG']`

Optional. Der Standardwert ist `HIVE`. Gibt den Tabellentyp der resultierenden Tabelle an

Beispiel:

```
WITH (table_type ='ICEBERG')
```

`external_location = [location]`

> **Note**
>
> Da Iceberg-Tabellen nicht extern sind, gilt diese Eigenschaft nicht für Iceberg-Tabellen. Um den Stammspeicherort einer Iceberg-Tabelle in einer CTAS-Anweisung zu definieren, verwenden Sie die später in diesem Abschnitt beschriebene `location`-Eigenschaft.

Optional. Der Speicherort, an dem Athena Ihre CTAS-Abfrage in Amazon S3 speichert.

Beispiel:

```
WITH (external_location ='s3://DOC-EXAMPLE-BUCKET/tables/parquet_table/')
```

Athena verwendet nicht zweimal denselben Pfad für Abfrageergebnisse. Wenn Sie den Speicherort manuell angeben, müssen Sie sicherstellen, dass der von Ihnen angegebene Amazon-S3-Speicherort keine Daten enthält. Athena versucht nie, Ihre Daten zu löschen. Wenn Sie denselben Speicherort erneut verwenden möchten, müssen Sie die Daten manuell löschen. Andernfalls schlägt Ihre CTAS-Abfrage fehl.

Wenn Sie eine CTAS-Abfrage ausführen, die einen `external_location` in einer Arbeitsgruppe angibt, die einen Speicherort für Abfrageergebnisse erzwingt, schlägt die Abfrage mit einer Fehlermeldung fehl. Sie können den für die Arbeitsgruppe angegebenen Abfrageergebnisspeicherort in den Arbeitsgruppendetails anzeigen.

Wenn Ihre Arbeitsgruppe die clientseitige Einstellung für den Abfrageergebnisspeicherort überschreibt, erstellt Athena die Tabelle am folgenden Speicherort:

```
s3://workgroup-query-results-location/tables/query-id/
```

Wenn Sie nicht die Eigenschaft `external_location` verwenden, um einen Speicherort anzugeben, und Ihre Arbeitsgruppe clientseitige Einstellungen nicht überschreibt, verwendet Athena Ihre clientseitige Einstellung für den Abfrageergebnisspeicherort, um die Tabelle am folgenden Speicherort zu erstellen:

```
s3://query-results-location-setting/Unsaved-or-query-name/year/month/date/
tables/query-id/
```

`is_external = [boolean]`

Optional. Gibt an, ob es sich bei der Tabelle um eine externe Tabelle handelt. Der Standardwert ist "true". Für Iceberg-Tabellen muss dies auf „false“ gesetzt werden.

Beispiel:

```
WITH (is_external = false)
```

`location = [location]`

Erforderlich für Iceberg-Tabellen. Gibt den Stammspeicherort für die aus den Abfrageergebnissen zu erstellende Iceberg-Tabelle an.

Beispiel:

```
WITH (location ='s3://DOC-EXAMPLE-BUCKET/tables/iceberg_table/')
```

field_delimiter = [delimiter]

Optionale und speziell für Text-basierte Datenspeicherformate. Das einstellige Feldtrennzeichen für Dateien in CSV-, TSV- und Textdateien. Zum Beispiel `WITH (field_delimiter = ',')`. Derzeit werden mehrzeilige Feldtrennzeichen für CTAS-Abfragen nicht unterstützt. Wenn Sie kein Feldtrennzeichen angeben, wird `\001` standardmäßig verwendet.

format = [storage_format]

Das Speicherformat für die CTAS-Abfrageergebnisse, z. B. `ORC`, `PARQUET`, `AVRO`, `JSON`, `ION` oder `TEXTFILE`. Für Iceberg-Tabellen sind die zulässigen Formate `ORC`, `PARQUET` und `AVRO`. Wenn nichts angegeben ist, wird standardmäßig `PARQUET` verwendet. Der Name dieses Parameters, `format`, muss in Kleinbuchstaben angegeben werden oder Ihre CTAS-Abfrage schlägt fehl.

Beispiel:

```
WITH (format = 'PARQUET')
```

bucketed_by = ARRAY[column_name[,…], bucket_count = [int]]

> **Note**
>
> Diese Eigenschaft gilt nicht für Iceberg-Tabellen. Verwenden Sie für Iceberg-Tabellen die Partitionierung mit Bucket-Transformation.

Eine Array-Liste der Buckets zum Gruppieren von Daten. Wenn nicht angegeben, gruppiert Athena Ihre Daten nicht in dieser Abfrage.

bucket_count = [int]

> **Note**
>
> Diese Eigenschaft gilt nicht für Iceberg-Tabellen. Verwenden Sie für Iceberg-Tabellen die Partitionierung mit Bucket-Transformation.

Die Anzahl der Buckets für das Bucketing Ihrer Daten. Wenn nicht angegeben, gruppiert Athena Ihre Daten nicht. Beispiel:

```
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**
>
> Diese Eigenschaft gilt nicht für Iceberg-Tabellen. Verwenden Sie die später in diesem Abschnitt beschriebene `partitioning`-Eigenschaft, um Partitionstransformationen für Iceberg-Tabellen zu verwenden.

Optional. Eine Array-Liste der Spalten, nach denen die CTAS-Tabelle partitioniert wird. Stellen Sie sicher, dass die Namen der partitionierten Spalten am Ende der Liste in der Liste der Spalten in der `SELECT`-Anweisung aufgeführt werden.

partitioning = ARRAY[partition_transform, ...]

Optional. Gibt die Partitionierung der zu erstellenden Iceberg-Tabelle an. Iceberg unterstützt eine Vielzahl von Partitionstransformationen und Partitionsentwicklungen. Partitionstransformationen sind in der folgenden Tabelle zusammengefasst.

Transform	Beschreibung
`year(ts)`	Erstellt eine Partition für jedes Jahr. Der Partitionswert ist die ganzzahlige Differenz in Jahren zwischen `ts` und dem 1. Januar 1970.

Transform	Beschreibung
`month(ts)`	Erstellt eine Partition für jeden Monat jedes Jahres. Der Partitionswert ist die ganzzahlige Differenz in Monaten zwischen `ts` und dem 1. Januar 1970.
`day(ts)`	Erstellt eine Partition für jeden Tag jedes Jahres. Der Partitionswert ist die ganzzahlige Differenz in Tagen zwischen `ts` und dem 1. Januar 1970.
`hour(ts)`	Erstellt eine Partition für jede Stunde jeden Tages. Der Partitionswert ist ein Zeitstempel, bei dem die Minuten und Sekunden auf Null gesetzt werden.
`bucket(x, nbuckets)`	Hasht die Daten in die angegebene Anzahl an Buckets. Der Partitionswert ist ein ganzzahliger Hash von `x`, mit einem Wert zwischen 0 und einschließlich `nbuckets - 1`.
`truncate(s, nchars)`	Erstellt den Partitionswert aus den ersten `nchars`-Zeichen von `s`.

Beispiel:

```
WITH (partitioning = ARRAY['month(order_date)',
                           'bucket(account_number, 10)',
                           'country']))
```

`optimize_rewrite_min_data_file_size_bytes = [long]`

Optional. Spezifische Konfiguration zur Datenoptimierung. Dateien, die kleiner als der angegebene Wert sind, werden zur Optimierung eingeschlossen. Der Standardwert ist das 0,75-fache des Wertes von `write_target_data_file_size_bytes`. Diese Eigenschaft gilt nur für Iceberg-Tabellen. Weitere Informationen finden Sie unter Optimieren von Iceberg-Tabellen.

Beispiel:

```
WITH (optimize_rewrite_min_data_file_size_bytes = 402653184)
```

`optimize_rewrite_max_data_file_size_bytes = [long]`

Optional. Spezifische Konfiguration zur Datenoptimierung. Dateien, die größer als der angegebene Wert sind, werden zur Optimierung eingeschlossen. Der Standardwert ist das 1,8-fache des Wertes von `write_target_data_file_size_bytes`. Diese Eigenschaft gilt nur für Iceberg-Tabellen. Weitere Informationen finden Sie unter Optimieren von Iceberg-Tabellen.

Beispiel:

```
WITH (optimize_rewrite_max_data_file_size_bytes = 966367641)
```

`optimize_rewrite_data_file_threshold = [int]`

Optional. Spezifische Konfiguration zur Datenoptimierung. Wenn es weniger Datendateien gibt, die eine Optimierung erfordern als der angegebene Schwellenwert, werden die Dateien nicht neu geschrieben. Dies ermöglicht die Ansammlung von mehr Datendateien, um Dateien näher an der Zielgröße zu erzeugen und unnötige Berechnungen zur Kosteneinsparung zu überspringen. Der Standardwert für ist 5. Diese Eigenschaft gilt nur für Iceberg-Tabellen. Weitere Informationen finden Sie unter Optimieren von Iceberg-Tabellen.

Beispiel:

```
WITH (optimize_rewrite_data_file_threshold = 5)
```

`optimize_rewrite_delete_file_threshold = [int]`

Optional. Spezifische Konfiguration zur Datenoptimierung. Wenn weniger Löschdateien mit einer Datendatei verknüpft sind als der Schwellenwert, wird die Datendatei nicht neu geschrieben. Dies ermöglicht die Ansammlöung von mehr Löschdateien für jede Datendatei zur Kosteneinsparung. Der Standardwert ist 2. Diese Eigenschaft gilt nur für Iceberg-Tabellen. Weitere Informationen finden Sie unter Optimieren von Iceberg-Tabellen.

Beispiel:

```
WITH (optimize_rewrite_delete_file_threshold = 2)
```

`vacuum_min_snapshots_to_keep = [int]`

Optional. Vakuum-spezifische Konfiguration. Die Mindestanzahl der aktuellsten Snapshots, die beibehalten werden sollen. Der Standardwert ist 1. Diese Eigenschaft gilt nur für Iceberg-Tabellen. Weitere Informationen finden Sie unter VACUUM.

> **Note**
>
> Die `vacuum_min_snapshots_to_keep`-Eigenschaft erfordert die Athena-Engine-Version 3.

Beispiel:

```
WITH (vacuum_min_snapshots_to_keep = 1)
```

`vacuum_max_snapshot_age_seconds = [long]`

Optional. Vakuum-spezifische Konfiguration. Ein Zeitraum in Sekunden, der das Alter der beibehaltenden Snapshots darstellt. Der Standardwert beträgt 432 000 (5 Tage). Diese Eigenschaft gilt nur für Iceberg-Tabellen. Weitere Informationen finden Sie unter VACUUM.

> **Note**
>
> Die `vacuum_max_snapshot_age_seconds`-Eigenschaft erfordert die Athena-Engine-Version 3.

Beispiel:

```
WITH (vacuum_max_snapshot_age_seconds = 432000)
```

`write_compression = [compression_format]`

Der Komprimierungstyp, der für jedes Speicherformat verwendet werden soll, mit dem die Komprimierung angegeben werden kann. Der `compression_format`-Wert gibt die Komprimierung an, die verwendet werden soll, wenn die Daten in die Tabelle geschrieben werden. Sie können die Komprimierung für die `TEXTFILE`-, `JSON`-, `PARQUET`- und `ORC`-Dateiformate angeben.

Wenn die `format`-Eigenschaft beispielsweise `PARQUET` als Speicherformat angibt, gibt der Wert für `write_compression` das Komprimierungsformat für Parquet an. In diesem Fall entspricht die Angabe eines Wertes für `write_compression` der Angabe eines Wertes für `parquet_compression`.

Wenn die `format`-Eigenschaft `ORC` als Speicherformat angibt, gibt der Wert für `write_compression` das Komprimierungsformat für ORC an. In diesem Fall entspricht die Angabe eines Wertes für `write_compression` der Angabe eines Wertes für `orc_compression`.

Mehrere Eigenschaften von Komprimierungsformaten können in derselben CTAS-Abfrage nicht angegeben werden. Sie können beispielsweise nicht sowohl `write_compression` als auch `parquet_compression` in derselben Abfrage angeben. Dasselbe gilt für `write_compression` und `orc_compression`. Informationen zu den für jedes Dateiformat unterstützten Komprimierungstypen finden Sie unter Athena-Komprimierungs-Support.

`orc_compression = [compression_format]`

Der Komprimierungstyp verwendet das `ORC`-Dateiformat, wenn die `ORC`-Daten in die Tabelle geschrieben werden. Zum Beispiel `WITH (orc_compression = 'ZLIB')`. Teile innerhalb der `ORC`-Datei (mit Ausnahme von `ORC` Postscript) werden mit der von Ihnen angegebenen Komprimierung komprimiert. Wenn es weggelassen wird, wird die ZLIB-Komprimierung standardmäßig für `ORC` verwendet.

> **Note**
>
> Aus Gründen der Kontinuität empfehlen wir, die `write_compression`-Eigenschaft anstelle von `orc_compression` zu verwenden. Verwenden Sie die `format`-Eigenschaft, um das Speicherformat als `ORC` anzugeben, und verwenden Sie dann die `write_compression`-Eigenschaft, um das von `ORC` verwendete Komprimierungsformat anzugeben.

`parquet_compression = [compression_format]`

Der Komprimierungstyp verwendet das Parquet-Dateiformat, wenn die Parquet-Daten in die Tabelle geschrieben werden. Zum Beispiel `WITH (parquet_compression = 'SNAPPY')`. Diese Komprimierung wird auf Spaltenstücke innerhalb der Parquet-Dateien angewendet. Wenn dies weggelassen wird, wird die GZIP-Komprimierung standardmäßig für Parquet verwendet.

> **Note**
>
> Aus Gründen der Kontinuität empfehlen wir, die `write_compression`-Eigenschaft anstelle von `parquet_compression` zu verwenden. Verwenden Sie die `format`-

Eigenschaft, um das Speicherformat als PARQUET anzugeben, und verwenden Sie dann die `write_compression`-Eigenschaft, um das von PARQUET verwendete Komprimierungsformat anzugeben.

`compression_level = [compression_level]`

Die zu verwendende Komprimierungsstufe. Diese Eigenschaft gilt nur für die ZSTD-Komprimierung. Mögliche Werte liegen zwischen 1 und 22. Der Standardwert ist 3. Weitere Informationen finden Sie unter Verwendung von ZSTD-Komprimierungsstufen in Athena.

Beispiele

Beispiele für CTAS-Abfragen finden Sie in den folgenden Ressourcen.

- Beispiele für CTAS-Abfragen
- Verwenden von CTAS und INSERT INTO für ETL und Datenanalyse
- Verwendung von CTAS-Anweisungen mit Amazon Athena, um die Kosten zu senken und die Leistung zu verbessern
- Verwenden von CTAS und INSERT INTO zum Umgehen des Limits von 100 Partitionen

CREATE VIEW

Erstellt eine neue Ansicht aus einer angegebenen SELECT-Abfrage. Die Ansicht ist eine logische Tabelle, auf die in zukünftigen Abfragen verwiesen werden kann. Ansichten enthalten keine Daten und schreiben keine Daten. Stattdessen wird die in der Ansicht angegebene Abfrage jedes Mal ausgeführt, wenn Sie in einer anderen Abfrage auf die Ansicht verweisen.

Note

Dieses Thema enthält zusammenfassende Informationen zur Referenz. Weitere Informationen zur Verwendung von Ansichten in Athena finden Sie unter Arbeiten mit Ansichten. Weitere Informationen zu Einschränkungen finden Sie unter Einschränkungen für Ansichten.

Syntax

```
CREATE [ OR REPLACE ] VIEW view_name AS query
```

Mit der optionalen `OR REPLACE`-Klausel können Sie die vorhandene Ansicht aktualisieren, indem Sie sie ersetzen. Weitere Informationen finden Sie unter Erstellen von Ansichten.

Beispiele

Verwenden Sie zum Erstellen einer Ansicht `test` aus der Tabelle `orders` eine Abfrage wie die folgende:

```
CREATE VIEW test AS
SELECT
orderkey,
orderstatus,
totalprice / 2 AS half
FROM orders;
```

Verwenden Sie zum Erstellen einer Ansicht `orders_by_date` aus der Tabelle `orders` die folgende Abfrage:

```
CREATE VIEW orders_by_date AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate;
```

Verwenden Sie zum Aktualisieren einer vorhandenen Ansicht ein Beispiel wie das folgende:

```
CREATE OR REPLACE VIEW test AS
SELECT orderkey, orderstatus, totalprice / 4 AS quarter
FROM orders;
```

Siehe auch SHOW COLUMNS, SHOW CREATE VIEW, DESCRIBE VIEW und DROP VIEW.

DESCRIBE

Zeigt eine oder mehrere Spalten, einschließlich Partitionsspalten, für die angegebene Tabelle an. Dieser Befehl ist nützlich, um die Attribute komplexer Spalten zu untersuchen.

Syntax

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

> **Important**
>
> Die Syntax für diese Anweisung lautet `DESCRIBE table_name`, nicht `DESCRIBE TABLE table_name`. Die Verwendung der letztgenannten Syntax führt zu der Fehlermeldung FAILED: SemanticException [Error 10001]: Tabelle hat Tabelle nicht gefunden.

Parameter

[EXTENDED | FORMATTED]

Bestimmt das Format der Ausgabe. Wenn Sie diese Parameter weglassen, werden Spaltennamen und ihre entsprechenden Datentypen, einschließlich Partitionsspalten, im Tabellenformat angezeigt. Die Angabe von `FORMATTED` zeigt nicht nur Spaltennamen und Datentypen in tabellarischer Form, sondern auch detaillierte Tabellen- und Speicherinformationen. `EXTENDED` zeigt Spalten- und Datentypinformationen im Tabellenformat und detaillierte Metadaten für die Tabelle in serialisierter Thrift-Form. Dieses Format ist weniger lesbar und eignet sich in erster Linie für das Debugging.

[PARTITION partition_spec]

Wenn diese Option enthalten ist, werden die von `partition_spec` angegebenen Metadaten für die Partition aufgeführt, wobei `partition_spec` das Format `(partition_column = partition_col_value, partition_column = partition_col_value, ...)` aufweist.

[col_name ([.field_name] | [.'$elem$'] | [.'key'] | [.'$value$'])*]

Gibt die zu untersuchende Spalte und die zu untersuchenden Attribute an. Sie können `.field_name` für ein Element einer Struktur, `'$elem$'` für ein Array-Element, `'$key$'` für einen Zuordnungsschlüssel und `'$value$'` für einen Zuordnungswert angeben. Sie können dies rekursiv angeben, um die komplexe Spalte eingehender zu prüfen.

Beispiele

```
DESCRIBE orders
```

```
DESCRIBE FORMATTED mydatabase.mytable PARTITION (part_col = 100) columnA;
```

Die folgende Abfrage und Ausgabe zeigt Spalten- und Datentyp-Informationen aus einer `impressions`-Tabelle basierend auf Amazon-EMR-Beispieldaten.

```
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
```

Die folgende Beispielabfrage und -ausgabe zeigen das Ergebnis für dieselbe Tabelle, wenn die Option FORMATTED verwendet wird.

```
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
```

Die folgende Beispielabfrage und -ausgabe zeigen das Ergebnis für dieselbe Tabelle, wenn die Option `EXTENDED` verwendet wird. Die detaillierten Tabelleninformationen werden in einer einzigen Zeile ausgegeben, aber hier zur Lesbarkeit formatiert.

```
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

Zeigt die Liste der Spalten für die benannte Ansicht an. Auf diese Weise können Sie die Attribute einer komplexen Ansicht untersuchen.

Syntax

```
DESCRIBE [db_name.]view_name
```

Beispiel

```
DESCRIBE orders;
```

Siehe auch SHOW COLUMNS, SHOW CREATE VIEW, SHOW VIEWS und DROP VIEW.

DROP DATABASE

Entfernt die benannte Datenbank aus dem Katalog. Wenn die Datenbank Tabellen enthält, müssen Sie die Tabellen entweder vor dem Ausführen von `DROP DATABASE` löschen oder die `CASCADE`-

Klausel verwenden. Sie können `DATABASE` oder `SCHEMA` gleichwertig verwenden. Sie haben dieselbe Bedeutung.

Syntax

```
DROP {DATABASE | SCHEMA} [IF EXISTS] database_name [RESTRICT | CASCADE]
```

Parameter

[IF EXISTS]

Falls `database_name` nicht vorhanden ist, wird der Fehler unterdrückt.

[RESTRICT|CASCADE]

Legt fest, wie Tabellen in `database_name` während der `DROP`-Operation behandelt werden. Wenn Sie `RESTRICT` angeben, wird die Datenbank nicht gelöscht, wenn sie Tabellen enthält. Dies ist das Standardverhalten. Wenn Sie `CASCADE` angeben, werden die Datenbank und alle darin enthaltenen Tabellen gelöscht.

Beispiele

```
DROP DATABASE clickstreams;
```

```
DROP SCHEMA IF EXISTS clickstreams CASCADE;
```

> **Note**
>
> Wenn Sie versuchen, eine Datenbank zu löschen, deren Name Sonderzeichen enthält (z. B. `my-database`), wird möglicherweise eine Fehlermeldung angezeigt. Um dieses Problem zu beheben, versuchen Sie, den Namen der Datenbank in hintere Anführungszeichen (`) einzuschließen. Informationen zum Benennen von Datenbanken in Athena finden Sie unter Namen für Tabellen, Datenbanken und Spalten.

DROP TABLE

Entfernt die Metadatendefinition für die Tabelle `table_name`. Wenn Sie eine externe Tabelle löschen, bleiben die zugrunde liegenden Daten intakt.

Syntax

```
DROP TABLE [IF EXISTS] table_name
```

Parameter

[IF EXISTS]

Falls `table_name` nicht vorhanden ist, wird der Fehler unterdrückt.

Beispiele

```
DROP TABLE fulfilled_orders
```

```
DROP TABLE IF EXISTS fulfilled_orders
```

Wenn Sie den Athena-Konsolenabfrage-Editor verwenden, um eine Tabelle mit anderen Sonderzeichen als dem Unterstrich (_) zu löschen, verwenden Sie Backticks, wie im folgenden Beispiel.

```
DROP TABLE `my-athena-database-01.my-athena-table`
```

Wenn Sie den JDBC-Konnektor verwenden, um eine Tabelle mit Sonderzeichen zu löschen, sind Backtick-Zeichen nicht erforderlich.

```
DROP TABLE my-athena-database-01.my-athena-table
```

DROP VIEW

Löscht eine vorhandene Ansicht. Mit der optionalen `IF EXISTS`-Klausel wird der Fehler unterdrückt, falls die Ansicht nicht existiert.

Weitere Informationen finden Sie unter Arbeiten mit Ansichten.

Syntax

```
DROP VIEW [ IF EXISTS ] view_name
```

Beispiele

```
DROP VIEW orders_by_date
```

```
DROP VIEW IF EXISTS orders_by_date
```

Siehe auch CREATE VIEW, SHOW COLUMNS, SHOW CREATE VIEW, SHOW VIEWS und DESCRIBE VIEW.

MSCK REPAIR TABLE

Verwenden Sie den `MSCK REPAIR TABLE`-Befehl, um die Metadaten im Katalog zu aktualisieren, nachdem Sie mit Hive kompatible Partitionen hinzugefügt haben.

Der `MSCK REPAIR TABLE`-Befehl durchsucht ein Dateisystem, z. B., Amazon S3, nach mit Hive kompatiblen Partitionen, die dem Dateisystem hinzugefügt wurden, nachdem die Tabelle erstellt wurde. `MSCK REPAIR TABLE` vergleicht die Partitionen in den Tabellenmetadaten und die Partitionen in S3. Wenn neue Partitionen am S3-Speicherort vorhanden sind, den Sie beim Erstellen der Tabelle angegeben haben, werden diese Partitionen den Metadaten und der Athena-Tabelle hinzugefügt.

Wenn Sie physische Partitionen hinzufügen, werden die Metadaten im Katalog mit dem Layout der Daten im Dateisystem inkonsistent und Informationen über die neuen Partitionen müssen dem Katalog hinzugefügt werden. Um die Metadaten zu aktualisieren, führen Sie `MSCK REPAIR TABLE` aus, damit Sie die Daten in den neuen Partitionen von Athena abfragen können.

> **Note**
>
> `MSCK REPAIR TABLE` fügt nur Partitionen zu Metadaten hinzu; sie werden nicht entfernt. Um Partitionen aus den Metadaten zu entfernen, nachdem die Partitionen in Amazon S3 manuell gelöscht wurden, führen Sie den Befehl `ALTER TABLE` *`table-name`* `DROP PARTITION` aus. Weitere Informationen finden Sie unter ALTER TABLE DROP PARTITION.

Überlegungen und Einschränkungen

Beachten Sie bei der Verwendung von `MSCK REPAIR TABLE` die folgenden Punkte:

- Möglicherweise dauert es etwas, bis alle Partitionen hinzugefügt wurden. Wird das Zeitlimit für diese Operation überschritten, ist der Status "unvollständig", da nur einige Partitionen zum Katalog

hinzugefügt wurden. Führen Sie `MSCK REPAIR TABLE` für dieselbe Tabelle so lange aus, bis alle Partitionen hinzugefügt wurden. Weitere Informationen finden Sie unter Daten in Athena partitionieren.

- Verwenden Sie für nicht mit Hive kompatible Partitionen ALTER TABLE ADD PARTITION, um sie zu laden und ihre Daten abzufragen.
- Partitionsspeicherorte zur Verwendung mit Athena müssen das `s3`-Protokoll verwenden (z. B. `s3://bucket/folder/`). In Athena führen Speicherorte, die andere Protokolle verwenden (z. B.`s3a://bucket/folder/`), zu Abfragefehlern, wenn `MSCK REPAIR TABLE`-Abfragen für die enthaltenen Tabellen ausgeführt werden.
- Da es sich bei `MSCK REPAIR TABLE` durchsucht sowohl einen Ordner als auch seine Unterordner, um ein übereinstimmendes Partitionsschema zu finden. Achten Sie darauf, dass die Daten für separate Tabellen in separaten Ordnerhierarchien aufbewahrt werden. Angenommen, Sie haben Daten für Tabelle A in `s3://table-a-data` und Daten für Tabelle B in `s3://table-a-data/table-b-data`. Wenn beide Tabellen nach Zeichenfolge partitioniert sind, fügt `MSCK REPAIR TABLE` die Partitionen für Tabelle B zu Tabelle A hinzu. Um dies zu vermeiden, verwenden Sie stattdessen separate Ordnerstrukturen wie `s3://table-a-data` und `s3://table-b-data`. Beachten Sie, dass dieses Verhalten mit Amazon EMR und Apache Hive konsistent ist.
- Aufgrund eines bekannten Problems schlägt `MSCK REPAIR TABLE` ohne Fehlermeldung fehl, wenn Partitionswerte einen Doppelpunkt (`:`) enthalten (wenn der Partitionswert z. B. ein Zeitstempel ist). Als Problemumgehung verwenden Sie ALTER TABLE ADD PARTITION.
- `MSCK REPAIR TABLE` fügt keine Partitionsspaltennamen hinzu, die mit einem Unterstrich (_) beginnen. Verwenden Sie ALTER TABLE ADD PARTITION, um dieses Limit zu umgehen.

Syntax

```
MSCK REPAIR TABLE table_name
```

Beispiele

```
MSCK REPAIR TABLE orders;
```

Fehlerbehebung

Nachdem Sie `MSCK REPAIR TABLE` ausgeführt haben, überprüfen Sie Folgendes, wenn Athena die Partitionen nicht der Tabelle in AWS Glue Data Catalog hinzugefügt hat:

- AWS Glue-Zugriff – Stellen Sie sicher, dass die AWS Identity and Access Management (IAM)-Rolle über eine Richtlinie verfügt, die die `glue:BatchCreatePartition`-Aktion zulässt. Weitere Informationen finden Sie unter glue:BatchCreatePartition in der IAM-Richtlinie zulassen an späterer Stelle in diesem Dokument.
- Amazon-S3-Zugriff – Stellen Sie sicher, dass die Rolle über eine Richtlinie mit ausreichenden Berechtigungen für den Zugriff auf Amazon S3 verfügt, einschließlich der `s3:DescribeJob`-Aktion. Ein Beispiel dafür, welche Amazon-S3-Aktionen zulässig sind, finden Sie in der Beispiel-Bucket-Richtlinie in Kontoübergreifender Zugriff auf Amazon-S3-Buckets in Athena.
- Amazon-S3-Objektschlüssel-Case-Schreibweise – Achten Sie darauf, dass der Amazon-S3-Pfad in Kleinbuchstaben und nicht mit der Camel-Case-Schreibweise geschrieben ist (z. B. `userid` statt `userId`), oder verwenden Sie `ALTER TABLE ADD PARTITION`, um die Objektschlüsselnamen anzugeben. Weitere Informationen finden Sie unter Den Amazon-S3-Pfad ändern oder neu definieren an späterer Stelle in diesem Dokument.
- Zeitüberschreitungen bei Abfragen – `MSCK REPAIR TABLE` wird am besten verwendet, wenn eine Tabelle zum ersten Mal erstellt wird oder wenn Unsicherheit bezüglich der Parität zwischen Daten und Partitionsmetadaten besteht. Wenn Sie `MSCK REPAIR TABLE` verwenden, um häufig neue Partitionen hinzuzufügen (z. B. täglich) und Abfragezeitüberschreitungen auftreten, sollten Sie ALTER TABLE ADD PARTITION verwenden.
- Partitionen, die im Dateisystem fehlen – Wenn Sie eine Partition manuell in Amazon S3 löschen und dann `MSCK REPAIR TABLE` ausführen, erhalten Sie möglicherweise die Fehlermeldung Partitionen fehlen im Dateisystem. Dies tritt auf, weil `MSCK REPAIR TABLE` keine veralteten Partitionen aus den Tabellenmetadaten entfernt. Um die gelöschten Partitionen aus den Tabellenmetadaten zu entfernen, führen Sie stattdessen ALTER TABLE DROP PARTITION aus. Beachten Sie, dass SHOW PARTITIONS ebenfalls nur die Partitionen in den Metadaten auflistet, nicht die Partitionen im Dateisystem.
- Fehler „NullPointerException-Name is null“

  Wenn Sie den AWS Glue-API-Vorgang CreateTable oder die AWS CloudFormation-Vorlage `AWS::Glue::Table` verwenden, um eine Tabelle zur Verwendung in Athena zu erstellen, ohne die `TableType`-Eigenschaft anzugeben und dann eine DDL-Abfrage wie `SHOW CREATE TABLE` oder `MSCK REPAIR TABLE` ausführen, können Sie die Fehlermeldung FEHLGESCHLAGEN: NullPointerException-Name ist Null anzeigen.

  Um den Fehler zu beheben, geben Sie einen Wert für das TableInput `TableType`-Attribut als Teil des AWS Glue-API-Aufrufs `CreateTable` oder der z-AWS CloudFormation Vorlage an. Mögliche Werte für `TableType` sind `EXTERNAL_TABLE` oder `VIRTUAL_VIEW`.

Diese Anforderung gilt nur, wenn Sie eine Tabelle mit dem AWS Glue-API-Vorgang `CreateTable` oder der `AWS::Glue::Table`-Vorlage erstellen. Wenn Sie eine Tabelle für Athena mithilfe einer DDL-Anweisung oder eines AWS Glue-Crawlers erstellen, wird die `TableType`-Eigenschaft automatisch für Sie definiert.

In den folgenden Abschnitten finden Sie zusätzliche Informationen.

glue:BatchCreatePartition in der IAM-Richtlinie zulassen

Überprüfen Sie die IAM-Richtlinien, die der Rolle zugeordnet sind, die Sie zum Ausführen von `MSCK REPAIR TABLE` verwenden. Wenn Sie den AWS Glue Data Catalog mit Athena verwenden, muss die IAM-Richtlinie die Aktion `glue:BatchCreatePartition` zulassen. Ein Beispiel für eine IAM-Richtlinie, die die Aktion `glue:BatchCreatePartition` zulässt, finden Sie unter AWS Verwaltete Richtlinie: AmazonAthenaFullAccess.

Den Amazon-S3-Pfad ändern oder neu definieren

Wenn ein oder mehrere Objektschlüssel im Amazon-S3-Pfad in Großbuchstaben statt in Kleinbuchstaben geschrieben sind, werden die `MSCK REPAIR TABLE` möglicherweise nicht zur AWS Glue Data Catalog hinzugefügt. Wenn Ihr Amazon-S3-Pfad beispielsweise den Objektschlüsselnamen `userId` enthält, werden die folgenden Partitionen möglicherweise nicht zur AWS Glue Data Catalog hinzugefügt:

```
s3://DOC-EXAMPLE-BUCKET/path/userId=1/

s3://DOC-EXAMPLE-BUCKET/path/userId=2/

s3://DOC-EXAMPLE-BUCKET/path/userId=3/
```

Sie lösen dieses Problem, indem Sie einen der folgenden Schritte ausführen:

- Verwenden Sie Kleinbuchstaben statt Großbuchstaben, wenn Sie Ihre Amazon-S3-Objektschlüssel erstellen:

  ```
  s3://DOC-EXAMPLE-BUCKET/path/userid=1/

  s3://DOC-EXAMPLE-BUCKET/path/userid=2/

  s3://DOC-EXAMPLE-BUCKET/path/userid=3/
  ```

- Verwenden Sie ALTER TABLE ADD PARTITION, um den Standort neu zu definieren, wie im folgenden Beispiel:

```
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/'
```

Beachten Sie, dass Amazon-S3-Objektschlüsselnamen zwar Großbuchstaben verwenden können, die Amazon-S3-Bucket-Namen selbst jedoch immer in Kleinbuchstaben geschrieben werden müssen. Weitere Informationen finden Sie unter Richtlinien zur Benennung von Objektschlüsseln und Regeln zur Benennung von Buckets im Amazon-S3-Benutzerhandbuch.

SHOW COLUMNS

Zeigt nur die Spaltennamen für eine bestimmte Tabelle oder Ansicht an. Um detailliertere Informationen zu erhalten, fragen Sie stattdessen den AWS Glue Data Catalog ab. Informationen und Beispiele finden Sie in den folgenden Abschnitten im Abfragen von AWS Glue Data Catalog-Thema:

- Zur Anzeige von Spaltenmetadaten, wie z. B. dem Datentyp, siehe Auflisten oder Durchsuchen von Spalten für eine angegebene Tabelle oder Ansicht.
- Um alle Spalten für alle Tabellen in einer bestimmten Datenbank in `AwsDataCatalog` anzuzeigen, siehe Auflisten oder Durchsuchen von Spalten für eine angegebene Tabelle oder Ansicht.
- Um alle Spalten für alle Tabellen in allen Datenbanken in `AwsDataCatalog` anzuzeigen, siehe Auflisten aller Spalten für alle Tabellen.
- Informationen zur Anzeige der Spalten, die bestimmte Tabellen in einer Datenbank gemeinsam haben, finden Sie unterAuflistung der Spalten, die bestimmte Tabellen gemeinsam haben.

Syntax

```
SHOW COLUMNS {FROM|IN} database_name.table_name
```

```
SHOW COLUMNS {FROM|IN} table_name [{FROM|IN} database_name]
```

Die FROM- und IN-Schlüsselwörter können synonym verwendet werden. Wenn *table_name* oder *database_name* Sonderzeichen wie Bindestriche enthalten, umgeben Sie den Namen in Anführungszeichen (z. B. `my-database`.`my-table`). Setzen Sie *table_name* oder *database_name* nicht in einfache oder doppelte Anführungszeichen. Derzeit wird die Verwendung von LIKE- und Mustervergleichsausdrücken nicht unterstützt.

Beispiele

Die folgenden äquivalenten Beispiele zeigen die Spalten aus der orders-Tabelle in der customers-Datenbank. Die ersten beiden Beispiele gehen davon aus, dass customers die aktuelle Datenbank ist.

```
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

Analysiert eine vorhandene Tabelle namens table_name, um die Abfrage zu erzeugen, mit der sie erstellt wurde.

Syntax

```
SHOW CREATE TABLE [db_name.]table_name
```

Parameter

TABLE [db_name.]table_name

Der Parameter db_name ist optional. Wird er nicht angegeben, wird als Standardkontext die aktuelle Datenbank verwendet.

> **Note**
>
> Der Tabellenname ist erforderlich.

Beispiele

```
SHOW CREATE TABLE orderclickstoday;
```

```
SHOW CREATE TABLE `salesdata.orderclickstoday`;
```

Fehlerbehebung

Wenn Sie den API-Vorgang AWS Glue CreateTable oder die Vorlage AWS CloudFormation `AWS::Glue::Table` verwenden, um eine Tabelle zur Verwendung in Athena zu erstellen, ohne die `TableType`-Eigenschaft anzugeben und dann eine DDL-Abfrage wie `SHOW CREATE TABLE` oder `MSCK REPAIR TABLE` ausführen, können Sie die Fehlermeldung FEHLGESCHLAGEN: NullPointerException-Name is null anzeigen.

Um den Fehler zu beheben, geben Sie einen Wert für das TableInput `TableType`-Attribut als Teil des AWS Glue-API-Aufrufs `CreateTable` oder der z-AWS CloudFormation Vorlage an. Mögliche Werte für `TableType` sind `EXTERNAL_TABLE` oder `VIRTUAL_VIEW`.

Diese Anforderung gilt nur, wenn Sie eine Tabelle mit dem AWS Glue-API-Vorgang `CreateTable` oder der `AWS::Glue::Table`-Vorlage erstellen. Wenn Sie eine Tabelle für Athena mithilfe einer DDL-Anweisung oder eines AWS Glue-Crawlers erstellen, wird die `TableType`-Eigenschaft automatisch für Sie definiert.

SHOW CREATE VIEW

Zeigt die SQL-Anweisung an, die die angegebene Ansicht erstellt.

Syntax

```
SHOW CREATE VIEW view_name
```

Beispiele

```
SHOW CREATE VIEW orders_by_date
```

Weitere Informationen finden Sie auch unter CREATE VIEW und DROP VIEW.

SHOW DATABASES

Listet alle Datenbanken auf, die im Metastore definiert sind. Sie können `DATABASES` oder `SCHEMAS` verwenden. Sie haben dieselbe Bedeutung.

Syntax

```
SHOW {DATABASES | SCHEMAS} [LIKE 'regular_expression']
```

Parameter

[LIKE '***regular_expression***']

Filtert die Liste der Datenbanken nach dem von Ihnen spezifizierten *`regular_expression`*. Für den Platzhalterzeichenabgleich können Sie die Kombination `.*` verwenden, die null Mal bis unbegrenzt oft einem beliebigen Zeichen entspricht.

Beispiele

```
SHOW SCHEMAS;
```

```
SHOW DATABASES LIKE '.*analytics';
```

SHOW PARTITIONS

Listet alle Partitionen in einer Athena-Tabelle in unsortierter Reihenfolge auf.

Syntax

```
SHOW PARTITIONS table_name
```

- Wie Sie Partitionen in einer Tabelle anzeigen und in einer bestimmten Reihenfolge auflisten, erfahren Sie im Abschnitt Auflisten von Partitionen für eine bestimmte Tabelle auf der Seite Abfragen von AWS Glue Data Catalog.
- Wie Sie den Inhalt einer Partition anzeigen, erfahren Sie im Abschnitt Abfragen der Daten auf der Seite Daten in Athena partitionieren.
- `SHOW PARTITIONS` führt keine Partitionen auf, die von Athena projiziert, aber nicht im AWS Glue-Katalog registriert sind. Weitere Informationen zur Partitionsprojektion finden Sie unter Partitionsprojektion mit Amazon Athena.
- `SHOW PARTITIONS` listet die Partitionen in Metadaten auf, nicht die Partitionen im eigentlichen Dateisystem. Um die Metadaten nach dem manuellen Löschen von Partitionen in Amazon S3 zu aktualisieren, führen Sie ALTER TABLE DROP PARTITION aus.

Beispiele

Die folgende Beispielabfrage zeigt die Partitionen für die `flight_delays_csv`-Tabelle, die Flugtabellendaten des US-Verkehrsministeriums anzeigt. Weitere Informationen zu dieser Beispieltabelle `flight_delays_csv` finden Sie unter LazySimpleSerDe für CSV- und TSV-Dateien sowie für benutzerdefinierte, durch Trennzeichen getrennte Dateien. Die Tabelle ist nach Jahr partitioniert.

```
SHOW PARTITIONS flight_delays_csv
```

Ergebnisse

```
year=2007
year=2015
year=1999
year=1993
year=1991
year=2003
year=1996
year=2014
year=2004
year=2011
...
```

Die folgende Beispielabfrage zeigt die Partitionen für die `impressions`-Tabelle, die Beispieldaten zum Surfen im Internet enthält. Weitere Informationen zu dieser Beispieltabelle `impressions` finden Sie unter Daten in Athena partitionieren. Die Tabelle wird durch die Spalte `dt` (datetime) partitioniert.

```
SHOW PARTITIONS impressions
```

Ergebnisse

```
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
...
```

Partitionen in sortierter Reihenfolge auflisten

Um die Partitionen in der Ergebnisliste zu ordnen, verwenden Sie die folgende `SELECT`-Syntax anstelle von `SHOW PARTITIONS`.

```
SELECT * FROM database_name."table_name$partitions" ORDER BY column_name
```

Die folgende Abfrage zeigt die Liste der Partitionen für das `flight_delays_csv`-Beispiel, jedoch in sortierter Reihenfolge.

```
SELECT * FROM "flight_delays_csv$partitions" ORDER BY year
```

Ergebnisse

```
year
1987
1988
1989
1990
1991
```

```
1992
1993
1994
1995
1996
1997
1998
1999
...
```

Weitere Informationen finden Sie im Auflisten von Partitionen für eine bestimmte Tabelle-Abschnitt auf der Abfragen von AWS Glue Data Catalog-Seite.

SHOW TABLES

Listet alle Basistabellen und Ansichten in einer Datenbank auf.

Syntax

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

Parameter

[IN database_name]

Gibt den `database_name` an, aus der die Tabellen aufgeführt werden. Wird dies nicht angegeben, wird die Datenbank des aktuellen Kontexts verwendet.

> **Note**
>
> `SHOW TABLES` kann scheitern, wenn `database_name` ein nicht unterstütztes Zeichen verwendet, wie etwa einen Bindestrich. Versuchen Sie zur Behebung, den Datenbanknamen in umgekehrten Anzeichen einzuschließen.

['regular_expression']

Filtert die Liste der Tabellen nach dem von Ihnen spezifizierten `regular_expression`. Um ein beliebiges Zeichen in `AWSDataCatalog`-Tabellen anzugeben, können Sie den `*`- oder `.*`-Platzhalterausdruck verwenden. Verwenden Sie für Apache Hive-Datenbanken den

Platzhalterausdruck `.*`. Um eine Auswahl zwischen Zeichen anzuzeigen, verwenden Sie das `|`-Zeichen.

Beispiele

Example – alle Tabellen in der Datenbank „**sampledb**“ anzeigen

```
SHOW TABLES IN sampledb
```

Results

```
alb_logs
cloudfront_logs
elb_logs
flights_2016
flights_parquet
view_2016_flights_dfw
```

Example – die Namen aller Tabellen in **sampledb** anzeigen, die das Wort „Flüge“ enthalten

```
SHOW TABLES IN sampledb '*flights*'
```

Results

```
flights_2016
flights_parquet
view_2016_flights_dfw
```

Example – die Namen aller Tabellen in **sampledb** anzeigen, die mit dem Wort „Protokolle“ enden

```
SHOW TABLES IN sampledb '*logs'
```

Results

```
alb_logs
cloudfront_logs
elb_logs
```

SHOW TBLPROPERTIES

Listet Tabelleneigenschaften für die benannte Tabelle auf.

Syntax

```
SHOW TBLPROPERTIES table_name [('property_name')]
```

Parameter

[('property_name')]

Falls angegeben, wird nur der Wert der Eigenschaft `property_name` aufgelistet.

Beispiele

```
SHOW TBLPROPERTIES orders;
```

```
SHOW TBLPROPERTIES orders('comment');
```

SHOW VIEWS

Listet die Ansichten in der angegebenen Datenbank oder in der aktuellen Datenbank auf, wenn Sie den Datenbanknamen weglassen. Verwenden Sie die optionale `LIKE`-Klausel mit einem regulären Ausdruck, um die Liste der Ansichtsnamen einzugrenzen.

Athena gibt eine Liste der Werte des Typs `STRING` zurück, wobei jeder Wert der Name einer Ansicht ist.

Syntax

```
SHOW VIEWS [IN database_name] [LIKE 'regular_expression']
```

Parameter

[IN database_name]

Gibt den `database_name` an, aus der die Ansichten aufgeführt werden. Wird dies nicht angegeben, wird die Datenbank des aktuellen Kontexts verwendet.

[LIKE 'regular_expression']

Filtert die Liste der Ansichten nach dem von Ihnen spezifizierten `regular_expression`. Lediglich der Platzhalter `*`, der für ein beliebiges Zeichen steht oder `|`, welches eine Auswahl zwischen Zeichen anzeigt, können verwendet werden.

Beispiele

```
SHOW VIEWS;
```

```
SHOW VIEWS IN marketing_analytics LIKE 'orders*'
```

Siehe auch SHOW COLUMNS, SHOW CREATE VIEW, DESCRIBE VIEW und DROP VIEW.

Überlegungen und Einschränkungen für SQL-Abfragen in Amazon Athena

Beachten Sie beim Ausführen von Abfragen in Athena die folgenden Überlegungen und Einschränkungen:

- Gespeicherte Verfahren – Gespeicherte Verfahren werden nicht unterstützt.
- Maximale Anzahl von Partitionen – Die maximale Anzahl von Partitionen, die Sie mit CTAS-Anweisungen (`CREATE TABLE AS SELECT`) erstellen können, beträgt 100. Weitere Informationen finden Sie unter CREATE TABLE AS. Eine Problemumgehung finden Sie unter Verwenden von CTAS und INSERT INTO zum Umgehen des Limits von 100 Partitionen.
- Nicht unterstützte Anweisungen – Die folgenden Anweisungen werden nicht unterstützt:
  - `CREATE TABLE LIKE` wird nicht unterstützt.
  - `DESCRIBE INPUT` und `DESCRIBE OUTPUT` werden nicht unterstützt.
  - Die `MERGE`-Anweisung wird nur für Transaktionstabellenformate unterstützt. Weitere Informationen finden Sie unter MERGE INTO.
  - `UPDATE`-Anweisungen werden nicht unterstützt.
- Trino- und Presto-Konnektoren – Weder Trino- noch Presto-Konnektoren werden unterstützt. Sie können mit Amazon Athena Federated Query verbinden. Weitere Informationen finden Sie unter Nutzung von Amazon-Athena-Verbundabfrage.
- Zeitüberschreitungen für Tabellen mit vielen Partitionen – Bei Athena können Zeitüberschreitungen bei Abfragen zu einer Tabelle auftreten, die Tausende von Partitionen enthält. Das kann vorkommen, wenn die Tabelle viele Partitionen hat, die nicht dem Typ `string` angehören. Wenn

Sie den Typ `string` verwenden, schneidet Athena Partitionen auf MetaStore-Ebene heraus. Verwenden Sie jedoch andere Datentypen, schneidet Athena Partitionen auf der Serverseite heraus. Je mehr Partitionen vorhanden sind, desto länger dauert dieser Vorgang und desto wahrscheinlicher ist es, dass eine Zeitüberschreitung für Ihre Abfragen eintritt. Um dieses Problem zu beheben, legen Sie den Partitionstyp mit `string` fest, sodass Athena Partitionen auf MetaStore-Ebene herauschneidet. Dies reduziert den Overhead und verhindert, dass Zeitüberschreitungen bei Abfragen auftreten.

- Unterstützung von S3 Glacier – Informationen zum Abfragen wiederhergestellter Amazon-S3-Glacier-Objekte finden Sie unter Wiederhergestellte Amazon-S3-Glacier-Objekte abfragen.
- Als augeblendet behandelte Dateien – Athena behandelt Quelldateien, die mit einem Unterstrich (_) oder einem Punkt (`.`) beginnen, als ausgeblendet. Benennen Sie die Dateien um, um diese Einschränkung zu umgehen.
- Begrenzung der Zeilen- oder Spaltengröße – Die Größe einer einzelnen Zeile oder ihrer Spalten darf 32 Megabyte nicht überschreiten. Dieser Grenzwert kann überschritten werden, wenn beispielsweise eine Zeile in einer CSV- oder JSON-Datei eine einzelne Spalte mit 300 Megabyte enthält. Eine Überschreitung dieser Grenze kann auch die Fehlermeldung Zeile zu lang in Textdatei erzeugen. Um diese Einschränkung zu umgehen, stellen Sie sicher, dass die Summe der Daten der Spalten in jeder Zeile kleiner als 32 MB ist.
- LIMIT-Klausel-Maximalwert – Die maximale Anzahl von Zeilen für die `LIMIT`-Klausel lautet

  9223372036854775807. Bei Verwendung von `ORDER BY` ist die maximale Anzahl unterstützter Zeilen für die LIMIT-Klausel 2 147 483 647. Ein Überschreiten dieser Grenze führt zur Fehlermeldung NOT_SUPPORTED: ORDER BY LIMIT > 2147483647 is not supported (NOT_SUPPORTED: ORDER BY LIMIT > 2147483647 wird nicht unterstützt).
- information_schema – Abfragen von `information_schema` sind am leistungsstärksten, wenn Sie eine kleine bis mittelgroße Menge an AWS Glue-Metadaten haben. Wenn Sie eine große Menge an Metadaten haben, können Fehler auftreten. Weitere Informationen zum Abfragen der `information_schema`-Datenbank für AWS Glue-Metadaten finden Sie unter Abfragen von AWS Glue Data Catalog.
- Array-Initialisierungen – Aufgrund einer Einschränkung in Java ist es nicht möglich, ein Array in Athena zu initialisieren, das mehr als 254 Argumente hat.
- Versteckte Metadatenspalten – Die versteckten Hive- oder Iceberg-Metadatenspalten `$bucket`, `$file_modified_time`, `$file_size` und `$partition` werden für Ansichten nicht unterstützt. Informationen zur Verwendung der `$path`-Metadatenspalte in Athena finden Sie unter Abrufen der Dateispeicherorte für Quelldaten in Amazon S3 .

Informationen zur maximalen Länge von Abfragezeichenfolgen, Kontingenten für Abfrage-Timeouts und Kontingenten für die aktive Anzahl von DML-Abfragen finden Sie unter Service Quotas.

Athena-Fehlerbehebung

Das Athena-Team hat die folgenden Informationen zur Fehlerbehebung von Kundenproblemen gesammelt. Obwohl sie nicht umfassend ist, enthält sie Ratschläge zu einigen häufigen Leistungs-, Timeout- und Speicherproblemen.

Themen

CREATE TABLE AS SELECT (CTAS)

Duplizierte Daten treten mit gleichzeitigen CTAS-Anweisungen auf

Athena führt keine gleichzeitige Validierung für CTAS durch. Stellen Sie sicher, dass für denselben Speicherort keine doppelte CTAS-Anweisung zur gleichen Zeit vorhanden ist. Selbst wenn

eine CTAS- oder INSERT-INTO-Anweisung fehlschlägt, können verwaiste Daten an dem in der Anweisung angegebenen Datenspeicherort verbleiben.

HIVE_TOO_MANY_OPEN_PARTITIONS

Wenn Sie eine CTAS-Anweisung verwenden, um eine Tabelle mit mehr als 100 Partitionen zu erstellen, wird möglicherweise der Fehler HIVE_TOO_MANY_OPEN_PARTITIONS: Exceeded limit of 100 open writers for partitions/buckets angezeigt. Um diese Einschränkung zu umgehen, können Sie eine CTAS-Anweisung und eine Reihe von `INSERT INTO`-Anweisungen verwenden, die jeweils bis zu 100 Partitionen erstellen oder einfügen. Weitere Informationen finden Sie unter Verwenden von CTAS und INSERT INTO zum Umgehen des Limits von 100 Partitionen.

Probleme mit Datendateien

Athena kann versteckte Dateien nicht lesen

Athena behandelt Quelldateien, die mit einem Unterstrich (_) oder einem Punkt (.) beginnen, als ausgeblendet. Benennen Sie die Dateien um, um diese Einschränkung zu umgehen.

Athena liest Dateien, die ich vom AWS Glue-Crawler ausgeschlossen habe

Athena erkennt keine Ausschlussmuster, die Sie einem AWS Glue-Crawler angeben. Wenn Sie beispielsweise über einen Amazon-S3-Bucket verfügen, der sowohl `.csv`- als auch `.json`-Dateien enthält und Sie die `.json`-Dateien vom Crawler ausschließen, fragt Athena beide Dateigruppen ab. Um dies zu vermeiden, platzieren Sie die Dateien, die Sie ausschließen möchten, an einem anderen Speicherort.

HIVE_BAD_DATA: Fehler beim Analysieren des Feldwerts

Dieser Fehler kann in folgenden Szenarien auftreten:

- Der in der Tabelle definierte Datentyp stimmt nicht mit den Quelldaten überein, oder ein einzelnes Feld enthält verschiedene Datentypen. Lösungsvorschläge finden Sie unter Meine Amazon-Athena-Abfrage schlägt mit dem Fehler „HIVE_BAD_DATA: Fehler beim Parsen des Feldwerts für das Feld X: Für Eingabezeichenfolge: „12312845691“ fehl“ im AWS-Wissenscenter.
- Nullwerte sind in einem ganzzahligen Feld vorhanden. Eine Problemumgehung besteht darin, die Spalte mit den Nullwerten als `string` zu erstellen und dann mit `CAST` das Feld in einer Abfrage zu konvertieren, wobei der Standardwert `0` für Nullwerte bereitgestellt wird. Weitere Informationen finden Sie unter Wenn ich CSV-Daten in Athena abfrage, erhalte ich den Fehler

„HIVE_BAD_DATA: Fehler beim Parsen des Feldwerts '' für Feld X: Für Eingabezeichenfolge: """ im AWS-Wissenscenter.

HIVE_CANNOT_OPEN_SPLIT: Fehler beim Öffnen von Hive split s3://*bucket-name*

Dieser Fehler kann auftreten, wenn Sie ein Amazon-S3-Bucket-Präfix abfragen, das eine große Anzahl von Objekten aufweist. Weitere Informationen finden Sie unter Wie behebe ich den Fehler „HIVE_CANNOT_OPEN_SPLIT: Fehler beim Öffnen von Hive split s3://awsdoc-example-bucket/: Langsam“ in Athena? im AWS-Wissenscenter.

HIVE_CURSOR_ERROR: com.amazonaws.services.s3.model.Amazons3Exception: Der angegebene Schlüssel ist nicht vorhanden

Dieser Fehler tritt normalerweise auf, wenn eine Datei entfernt wird, wenn eine Abfrage ausgeführt wird. Führen Sie entweder die Abfrage erneut aus, oder überprüfen Sie Ihren Workflow, um festzustellen, ob ein anderer Auftrag oder Prozess die Dateien ändert, wenn die Abfrage ausgeführt wird.

HIVE_CURSOR_ERROR: Unerwartetes Ende des Eingabedatenstreams

Diese Meldung zeigt an, dass die Datei beschädigt oder leer ist. Überprüfen Sie die Integrität der Datei und führen Sie die Abfrage erneut aus.

HIVE_FILESYSTEM_ERROR: Falsche FileSize *1234567* für Datei

Diese Meldung kann auftreten, wenn sich eine Datei zwischen Abfrageplanung und Abfrageausführung geändert hat. Es tritt normalerweise auf, wenn eine Datei auf Amazon S3 direkt ersetzt wird (z. B. wird ein PUT für einen Schlüssel ausgeführt, in dem bereits ein Objekt existiert). Athena unterstützt das Löschen oder Ersetzen des Inhalts einer Datei nicht, wenn eine Abfrage ausgeführt wird. Um diesen Fehler zu vermeiden, planen Sie Aufträge, die Dateien zu Zeiten überschreiben oder löschen, in denen Abfragen nicht ausgeführt werden, oder schreiben Sie nur Daten in neue Dateien oder Partitionen.

HIVE_UNKNOWN_ERROR: Eingabeformat kann nicht erstellt werden

Dieser Fehler kann auf Probleme wie die folgenden zurückzuführen sein:

- Der AWS Glue-Crawler konnte das Datenformat nicht klassifizieren
- Bestimmte AWS Glue-Tabellendefinitionseigenschaften sind leer

- Athena unterstützt das Datenformat der Dateien in Amazon S3 nicht

Weitere Informationen finden Sie unter Wie behebe ich den Fehler „Eingabeformat kann nicht erstellt werden“ in Athena? im AWS-Wissenscenter oder sehen Sie sich das Wissenscenter-Video an.

Der zum Speichern der Abfrageergebnisse bereitgestellte S3-Speicherort ist ungültig.

Stellen Sie sicher, dass Sie einen gültigen S3-Speicherort für Ihre Abfrageergebnisse angegeben haben. Weitere Informationen finden Sie unter Angeben eines Speicherorts des Abfrageergebnisses im Thema Arbeiten mit Abfrageergebnissen, letzten Abfragen und Ausgabedateien.

Delta-Lake-Tabellen von Linux Foundation

Das Delta-Lake-Tabellenschema ist nicht synchron

Wenn Sie eine Delta-Lake-Tabelle abfragen, die ein veraltetes Schema in AWS Glue enthält, kann die folgende Fehlermeldung angezeigt werden:

```
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.
```

Das Schema kann veraltet sein, wenn es nach dem Hinzufügen zu Athena in AWS Glue geändert wird. Führen Sie einen der folgenden Schritte aus, um das Schema zu aktualisieren:

- Führen Sie in AWS Glue den AWS Glue-Crawler aus.
- Löschen Sie in Athena die Tabelle und erstellen Sie sie erneut.
- Fügen Sie fehlende Spalten manuell hinzu, indem Sie entweder die ALTER TABLE ADD COLUMNS-Anweisung in Athena verwenden oder das Tabellenschema in AWS Glue bearbeiten.

Verbundabfragen

Timeout beim Aufrufen von ListTableMetadata

Ein Aufruf der API ListTableMetadata kann zu einem Timeout führen, wenn die Datenquelle viele Tabellen enthält, wenn die Datenquelle langsam ist oder wenn das Netzwerk langsam ist. Sie lösen dieses Problem, indem Sie die folgenden Schritte ausführen.

- Anzahl der Tabellen überprüfen – Wenn Sie mehr als 1 000 Tabellen haben, versuchen Sie, die Anzahl der Tabellen zu reduzieren. Für eine möglichst schnelle `ListTableMetadata`-Antwort empfehlen wir, weniger als 1 000 Tabellen pro Katalog zu haben.
- Überprüfen Sie die Lambda-Konfiguration – Die Überwachung des Verhaltens der Lambda-Funktion ist von entscheidender Bedeutung. Wenn Sie Verbundkataloge verwenden, sollten Sie unbedingt die Ausführungsprotokolle der Lambda-Funktion überprüfen. Passen Sie auf der Grundlage der Ergebnisse die Arbeitsspeicher- und Timeout-Werte entsprechend an. Um mögliche Probleme mit Timeouts zu identifizieren, überprüfen Sie Ihre Lambda-Konfiguration erneut. Weitere Informationen finden Sie unter Konfigurieren von Funktions-Timeout (Konsole) im AWS Lambda-Entwicklerhandbuch.
- Überprüfen Sie die Protokolle der Verbunddatenquellen – Untersuchen Sie die Protokolle und Fehlermeldungen der verbundenen Datenquelle, um festzustellen, ob Probleme oder Fehler vorliegen. Die Protokolle können wertvolle Einblicke in die Ursache des Timeouts liefern.
- **`StartQueryExecution`** zum Abrufen von Metadaten verwenden – Wenn Sie über mehr als 1 000 Tabellen verfügen, kann es länger als erwartet dauern, Metadaten über Ihren Verbundkonnektor abzurufen. Da die asynchrone Natur von StartQueryExecution sicherstellt, dass Athena die Abfrage optimal ausführt, sollten Sie die Verwendung von `StartQueryExecution` als Alternative zu `ListTableMetadata` in Betracht ziehen. Die folgenden AWS CLI-Beispiele zeigen, wie `StartQueryExecution` anstelle von `ListTableMetadata` zum Abrufen aller Metadaten von Tabellen in Ihrem Datenkatalog verwendet werden kann.

  Führen Sie zunächst eine Abfrage aus, die alle Tabellen abruft, wie im folgenden Beispiel.

```
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"
```

  Als Nächstes rufen Sie die Metadaten einer einzelnen Tabelle ab, wie im folgenden Beispiel.

```
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"
```

  Wie lange es dauert, bis die Ergebnisse abgerufen werden, hängt von der Anzahl der Tabellen in Ihrem Katalog ab.

Informationen zur Fehlerbehebung bei Verbundabfragen finden Sie unter Verbreitete Probleme im Abschnitt awslabs/aws-athena-query-federation von GitHub oder in der Dokumentation der individuellen Athena-Datenquellen-Konnektoren.

JSON-bezogene Fehler

NULL oder falsche Datenfehler beim Versuch, JSON-Daten zu lesen

NULL oder falsche Datenfehler, wenn Sie versuchen, JSON-Daten zu lesen, können auf eine Reihe von Ursachen zurückzuführen sein. Um Zeilen zu identifizieren, die Fehler verursachen, wenn Sie OpenX SerDe verwenden, setzen Sie `ignore.malformed.json` auf `true`. Fehlgeformte Datensätze werden als NULL zurückgegeben. Weitere Informationen finden Sie unter Ich erhalte Fehler, wenn ich versuche, JSON-Daten in Amazon Athena zu lesen im AWS-Wissenscenter oder sehen Sie sich das Wissenscenter-Video an.

HIVE_BAD_DATA: Fehler beim Parsen des Feldwerts für Feld 0: java.lang.String kann nicht in org.openx.data.jsonserde.json.JSONObject umgewandelt werden

Das OpenX JSON SerDe löst diesen Fehler aus, wenn eine Spalte in einer Athena-Abfrage nicht analysiert werden kann. Dies kann passieren, wenn Sie eine Spalte als `map` oder `struct` definieren, die zugrunde liegenden Daten jedoch tatsächlich ein `string`, `int` oder ein anderer primitiver Typ sind.

HIVE_CURSOR_ERROR: Zeile ist kein gültiges JSON-Objekt - JSONException: Doppelter Schlüssel

Dieser Fehler tritt auf, wenn Sie Athena verwenden, um AWS Config-Ressourcen abzufragen, die mehrere Tags mit demselben Namen in unterschiedlichen Fällen aufweisen. Die Lösung besteht darin, `CREATE TABLE` mit `WITH SERDEPROPERTIES 'case.insensitive'='false'` auszuführen und die Namen zuzuordnen. Weitere Informationen zu `case.insensitive` und Mapping finden Sie unter JSON-SerDe-Bibliotheken. Weitere Informationen finden Sie unter Wie behebe ich „HIVE_CURSOR_ERROR: Zeile ist kein gültiges JSON-Objekt - JSONException: Duplizierter Schlüssel“ beim Lesen von Dateien aus AWS Config in Athena? im AWS-Wissenscenter.

HIVE_CURSOR_ERROR Nachrichten mit hübsch gedrucktem JSON

Die Hive JSON SerDe- und OpenX JSON SerDe-Bibliotheken erwarten, dass sich jedes JSON-Dokument auf einer einzigen Textzeile befindet, ohne Zeilenabschlusszeichen, die die Felder

im Datensatz trennen. Wenn der JSON-Text im hübschen Druckformat vorliegt, erhalten Sie möglicherweise eine Fehlermeldung wie HIVE_CURSOR_ERROR: Zeile ist kein gültiges JSON-Objekt oder HIVE_CURSOR_ERROR: JsonParseException: Unerwartetes Ende der Eingabe: Erwartete Schließmarkierung für OBJEKT, wenn Sie versuchen, die Tabelle abzufragen, nachdem Sie dies erstellt haben. Weitere Informationen finden Sie unter JSON-Datendatei in der OpenX-SerDe-Dokumentation auf GitHub.

Mehrere JSON-Datensätze geben einen SELECT COUNT von 1 zurück

Wenn Sie das OpenX JSON SerDe verwenden, stellen Sie sicher, dass die Datensätze durch ein Zeilenumbruchzeichen getrennt sind. Weitere Informationen finden Sie unter Die SELECT-COUNT-Abfrage in Amazon Athena gibt nur einen Datensatz zurück, obwohl die Eingabe-JSON-Datei mehrere Datensätze enthält im AWS-Wissenscenter.

Eine Tabelle, die von einem AWS Glue-Crawler erstellt wurde, der einen benutzerdefinierten JSON-Klassifikator verwendet, kann nicht abgefragt werden

Die Athena-Engine unterstützt keine benutzerdefinierten JSON-Klassifikatoren. Um dieses Problem zu umgehen, erstellen Sie eine neue Tabelle ohne den benutzerdefinierten Klassifikator. Um den JSON zu transformieren, können Sie CTAS verwenden oder eine Ansicht erstellen. Wenn Sie beispielsweise mit Arrays arbeiten, können Sie die Option UNNEST verwenden, um den JSON zu reduzieren. Eine andere Möglichkeit besteht darin, einen AWS Glue-ETL-Auftrag zu verwenden, der den benutzerdefinierten Klassifikator unterstützt, die Daten in Amazon S3 in Parquet zu konvertieren und sie dann in Athena abzufragen.

MSCK REPAIR TABLE

Informationen zu Problemen im Zusammenhang mit MSCK REPAIR TABLE finden Sie in den Abschnitten Überlegungen und Einschränkungen und Fehlerbehebung der Seite MSCK REPAIR TABLE.

Probleme mit der Ausgabe

Ausgabe-Bucket konnte nicht verifiziert/erstellt werden

Dieser Fehler kann auftreten, wenn der angegebene Abfrageergebnisspeicherort nicht vorhanden ist oder wenn die richtigen Berechtigungen nicht vorhanden sind. Weitere Informationen finden Sie unter Wie behebe ich den Fehler „Ausgabe-Bucket kann nicht überprüft/erstellt werden“ in Amazon Athena? im AWS-Wissenscenter.

TIMESTAMP-Ergebnis ist leer

Athena benötigt das Java–TIMESTAMP-Format. Weitere Informationen finden Sie unter Wenn ich eine Tabelle in Amazon Athena abfrage, ist das TIMESTAMP-Ergebnis leer im AWS-Wissenscenter.

Speichern der Athena Abfrageausgabe in einem anderen Format als CSV

Standardmäßig gibt Athena Dateien nur im CSV-Format aus. Um die Ergebnisse einer `SELECT`-Abfrage in einem anderen Format auszugeben, können Sie die `UNLOAD`-Anweisung verwenden. Weitere Informationen finden Sie unter UNLOAD. Sie können auch eine CTAS-Abfrage verwenden, die die `format` Tabelleneigenschaft verwendet, um das Ausgabeformat zu konfigurieren.
Im Gegensatz zu `UNLOAD` erfordert die CTAS-Technik die Erstellung einer Tabelle. Weitere Informationen finden Sie unter Wie kann ich eine Athena-Abfrageausgabe in einem anderen Format als CSV speichern, z. B. einem komprimierten Format? im AWS-Wissenscenter.

Der zum Speichern der Abfrageergebnisse bereitgestellte S3-Speicherort ist ungültig

Sie können diese Fehlermeldung erhalten, wenn sich Ihr Ausgabe-Bucket-Speicherort nicht in derselben Region befindet wie die Region, in der Sie Ihre Abfrage ausführen. Um dies zu vermeiden, geben Sie einen Abfrageergebnisspeicherort in der Region an, in der Sie die Abfrage ausführen. Informationen zu den erforderlichen Schritten finden Sie unter Angeben eines Speicherorts des Abfrageergebnisses.

Probleme mit Parquet

org.apache.parquet.io.groupColumnio kann nicht in org.apache.parquet.io.primitiveColumnio umgewandelt werden

Dieser Fehler wird durch eine Nichtübereinstimmung des Parquet-Schemas verursacht. Eine Spalte mit einem nicht-primitiven Typ (z. B. `array`) wurde in AWS Glue als primitiver Typ (z. B. `string`) deklariert. Um dieses Problem zu beheben, überprüfen Sie das Datenschema in den Dateien und vergleichen Sie es mit dem in AWS Glue deklarierten Schema.

Probleme mit Statistiken in Parquet

Wenn Sie Parquet-Daten lesen, erhalten Sie möglicherweise Fehlermeldungen wie die folgenden:

```
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.
```

Um dieses Problem zu umgehen, verwenden Sie die CREATE TABLE- oder ALTER TABLE SET TBLPROPERTIES-Anweisung, um die Parquet-SerDe-Eigenschaft `parquet.ignore.statistics` auf `true` zu setzen, wie in den folgenden Beispielen.

Beispiel für CREATE TABLE

```
...
ROW FORMAT SERDE
'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
WITH SERDEPROPERTIES ('parquet.ignore.statistics'='true')
STORED AS PARQUET
...
```

Beispiel für ALTER TABLE

```
ALTER TABLE ... SET TBLPROPERTIES ('parquet.ignore.statistics'='true')
```

Weitere Informationen über Parquet Hive SerDe finden Sie unter Parquet SerDe.

Partitionierungsprobleme

MSCK REPAIR TABLE entfernt keine veralteten Partitionen

Wenn Sie eine Partition manuell in Amazon S3 löschen und dann MSCK REPAIR TABLE ausführen, erhalten Sie möglicherweise die Fehlermeldung Partitions missing from filesystem (Partitionen fehlen im Dateisystem). Dies tritt auf, weil MSCK REPAIR TABLE nicht veraltete Partitionen aus Tabellenmetadaten entfernt. Verwenden Sie ALTER TABLE DROP PARTITION, um die veralteten Partitionen manuell zu entfernen. Weitere Informationen finden Sie im Abschnitt „Fehlerbehebung“ des MSCK REPAIR TABLE-Themas.

Fehler MSCK REPAIR TABLE

Wenn einer bestimmten Tabelle eine große Anzahl von Partitionen (z. B. mehr als 100.000) zugeordnet ist, kann `MSCK REPAIR TABLE` aufgrund von Speicherbeschränkungen fehlschlagen. Verwenden Sie stattdessen ALTER TABLE ADD PARTITION, um dieses Limit zu umgehen.

MSCK REPAIR TABLE erkennt Partitionen, fügt sie jedoch nicht zuAWS Glue hinzu

Dieses Problem kann auftreten, wenn ein Amazon-S3-Pfad in Camel statt in Kleinbuchstaben angegeben ist oder eine IAM-Richtlinie die Aktion `glue:BatchCreatePartition` nicht zulässt. Weitere Informationen finden Sie unter MSCK REPAIR TABLE erkennt Partitionen in Athena, fügt sie jedoch nicht zum AWS Glue Data Catalog hinzu im AWS-Wissenscenter.

Partitionsprojektionsbereiche mit dem Datumsformat TT-MM-JJJ-HH-MM-SS oder JJJJ-MM-TT funktionieren nicht

Um korrekt zu funktionieren, muss das Datumsformat auf `yyyy-MM-dd HH:00:00` eingestellt sein. Weitere Informationen finden Sie im Stack-Überlauf-Beitrag Athena-Partitionsprojektion funktioniert nicht wie erwartet.

PARTITION BY unterstützt den BIGINT-Typ nicht

Konvertieren Sie den Datentyp in `string`und versuchen Sie es erneut.

Keine aussagekräftigen Partitionen verfügbar

Diese Fehlermeldung bedeutet in der Regel, dass die Partitionseinstellungen beschädigt wurden. Um dieses Problem zu beheben, löschen Sie die Tabelle und erstellen Sie eine Tabelle mit neuen Partitionen.

Partitionsprojektion funktioniert nicht in Verbindung mit Bereichspartitionen

ÜStellen Sie sicher, dass die Zeitbereichseinheitprojection.*<columnName>*.interval.unit mit dem Trennzeichen für die Partitionen übereinstimmt. Wenn Partitionen beispielsweise durch Tage getrennt sind, funktioniert eine Bereichseinheit von Stunden nicht.

Fehler bei der Partitionsprojektion, wenn der Bereich mit einem Bindestrich angegeben wird

Die Angabe der `range`-Tabelleneigenschaft mit einem Bindestrich anstelle eines Kommas erzeugt einen Fehler wie INVALID_TABLE_PROPERTY: Für die Eingabezeichenfolge: „*Zahl* - *Zahl*“. Beachten Sie, dass die Werte durch Kommata voneinander getrennt werden müssen, nicht durch einen Bindestrich. Weitere Informationen finden Sie unter Ganzzahl-Typ.

HIVE_UNKNOWN_ERROR: Eingabeformat kann nicht erstellt werden

Eine oder mehrere der Glue-Partitionen werden in einem anderen Format deklariert, da jede Glue-Partition unabhängig über ihr eigenes spezifisches Eingabeformat verfügt. Bitte überprüfen Sie, wie Ihre Partitionen in AWS Glue definiert sind.

HIVE_PARTITION_SCHEMA_MISMATCH

Unterscheidet sich das Schema einer Partition vom Schema der Tabelle, kann eine Abfrage mit der Fehlermeldung HIVE_PARTITION_SCHEMA_MISMATCH fehlschlagen. Weitere Informationen finden Sie unter Synchronisieren von Partitionsschemata zur Vermeidung von "HIVE_PARTITION_SCHEMA_MISMATCH".

SemanticException-Tabelle ist nicht partitioniert, aber die Partitionsspezifikation existiert

Dieser Fehler kann auftreten, wenn in der `CREATE TABLE`-Anweisung keine Partitionen definiert wurden. Weitere Informationen finden Sie unter Wie kann ich den Fehler „FEHLGESCHLAGEN: SemanticException-Tabelle ist nicht partitioniert, aber Partitionsspezifikation existiert“ in Athena beheben? im AWS-Wissenscenter.

Null Byte **`_$folder$`**-Datei

Wenn du eine `ALTER TABLE ADD PARTITION`-Anweisung ausführst und fälschlicherweise eine Partition angibts, die bereits vorhanden ist, und einen falschen Ort für Simple Storage Service (Amazon S3), Null-Byte-Platzhalterdateien des Formats *`partition_value`*`_$folder$` werden in Simple Storage Service (Amazon S3) erstellt. Sie müssen diese Dateien manuell entfernen.

Um dies zu verhindern, verwenden Sie die `ADD IF NOT EXISTS`-Syntax in Ihrer `ALTER TABLE ADD PARTITION`-Aussage, wie folgt:

```
ALTER TABLE table_name ADD IF NOT EXISTS PARTITIION […]
```

Aus partitionierten Daten zurückgegebene Datensätze

Dieses Problem kann aus einer Vielzahl von Gründen auftreten. Mögliche Ursachen und Lösungen finden Sie unter Ich habe eine Tabelle in Amazon Athena mit definierten Partitionen erstellt, aber wenn ich die Tabelle abfrage, werden null Datensätze zurückgegeben im AWS-Wissenscenter.

Weitere Informationen finden Sie auch unter HIVE_TOO_MANY_OPEN_PARTITIONS.

Berechtigungen

Zugriffsverweigerter Fehler beim Abfragen von Amazon S3

Dies kann auftreten, wenn Sie nicht über die Berechtigung zum Lesen der Daten im Bucket oder zum Schreiben in den Ergebnis-Bucket verfügen oder der Amazon-S3-Pfad einen Regionsendpunkt wie `us-east-1.amazonaws.com` enthält. Weitere Informationen finden Sie unter Wenn ich eine Athena-Abfrage ausführe, erhalte ich die Fehlermeldung „Zugriff verweigert“ im AWS-Wissenscenter.

Zugriff verweigert mit Statuscode: 403 Fehler beim Ausführen von DDL-Abfragen für verschlüsselte Daten in Amazon S3

Wenn Sie möglicherweise die Fehlermeldung Zugriff verweigert (Service: Amazon S3; Statuscode: 403; Fehlercode: AccessDenied; Anforderungs-ID: *<request_id>*) erhalten, wenn die folgenden Bedingungen zutreffen:

1. Sie führen eine DDL-Abfrage wie `ALTER TABLE ADD PARTITIO`Noder `MSCK REPAIR TABLE` aus.
2. Sie haben einen Bucket, dessen Standardverschlüsselung für die Verwendung von `SSE-S3` konfiguriert ist.
3. Der Bucket verfügt auch über eine Bucket-Richtlinie wie die folgende, die `PutObject`-Anforderungen erzwingt, die PUT-Header `"s3:x-amz-server-side-encryption": "true"` und `"s3:x-amz-server-side-encryption": "AES256"` anzugeben.

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

In einem solchen Fall besteht die empfohlene Lösung darin, die Bucket-Richtlinie wie die oben genannte zu entfernen, da die Standardverschlüsselung des Buckets bereits vorhanden ist.

Zugriff verweigert mit Statuscode: 403 beim Abfragen eines Amazon-S3-Buckets in einem anderen Konto

Dieser Fehler kann auftreten, wenn Sie versuchen, Protokolle abzufragen, die von einem anderen AWS-Service geschrieben wurden und das zweite Konto der Bucket-Eigentümer ist, aber nicht der Eigentümer der Objekte im Bucket. Weitere Informationen finden Sie unter Ich erhalte die Amazon-S3-Ausnahme „Zugriff verweigert mit Statuscode: 403“ in Amazon Athena, wenn ich einen Bucket in einem anderen Konto abfrage im AWS-Wissenscenter oder sehen Sie sich das Wissenscenter-Video an.

IAM-Rollenanmeldeinformationen zum Herstellen einer Verbindung mit dem Athena-JDBC-Treiber

Sie können die temporären Anmeldeinformationen einer Rolle abrufen, um die JDBC-Verbindung zu Athena zu authentifizieren. Temporäre Anmeldeinformationen haben eine maximale Lebensdauer von 12 Stunden. Weitere Informationen finden Sie unter Wie kann ich meine IAM-Rollenanmeldeinformationen verwenden oder zu einer anderen IAM-Rolle wechseln, wenn ich über den JDBC-Treiber eine Verbindung zu Athena herstelle? im AWS Wissenscenter.

Probleme mit der Abfragesyntax

Fehlgeschlagen: NullPointerException-Name ist Null

Wenn Sie den API-Vorgang AWS Glue CreateTable oder die Vorlage AWS CloudFormation `AWS::Glue::Table` verwenden, um eine Tabelle zur Verwendung in Athena zu erstellen, ohne

die `TableType`-Eigenschaft anzugeben und dann eine DDL-Abfrage wie `SHOW CREATE TABLE` oder `MSCK REPAIR TABLE` ausführen, können Sie die Fehlermeldung FEHLGESCHLAGEN: NullPointerException-Name is null anzeigen.

Um den Fehler zu beheben, geben Sie einen Wert für das TableInput `TableType`-Attribut als Teil des AWS Glue-API-Aufrufs `CreateTable` oder der z-AWS CloudFormation Vorlage an. Mögliche Werte für `TableType` sind `EXTERNAL_TABLE` oder `VIRTUAL_VIEW`.

Diese Anforderung gilt nur, wenn Sie eine Tabelle mit dem AWS Glue-API-Vorgang `CreateTable` oder der `AWS::Glue::Table`-Vorlage erstellen. Wenn Sie eine Tabelle für Athena mithilfe einer DDL-Anweisung oder eines AWS Glue-Crawlers erstellen, wird die `TableType`-Eigenschaft automatisch für Sie definiert.

Funktion nicht registriert

Dieser Fehler tritt auf, wenn Sie versuchen, eine Funktion zu verwenden, die Athena nicht unterstützt. Eine Liste der von Athena unterstützten Funktionen finden Sie unter Funktionen in Amazon Athena oder führen Sie die `SHOW FUNCTIONS`-Anweisung im Abfrage-Editor aus. Sie können auch Ihre eigene benutzerdefinierte Funktion (UDF) schreiben. Weitere Informationen finden Sie unter Wie behebe ich den Syntaxfehler „Funktion nicht registriert“ in Athena? im AWS-Wissenscenter.

GENERIC_INTERNAL_ERROR Ausnahmen

`GENERIC_INTERNAL_ERROR`-Ausnahmen können verschiedene Ursachen haben, z. B. die folgenden:

- GENERIC_INTERNAL_ERROR: null – Sie können diese Ausnahme unter den folgenden Bedingungen sehen:
  - Sie haben eine Schema-Nichtübereinstimmung zwischen dem Datentyp einer Spalte in der Tabellendefinition und dem tatsächlichen Datentyp des Datensatzes.
  - So leiten Sie eine `CREATE TABLE AS SELECT`(CTAS)-Abfrage mit ungenauer Syntax ein.
- GENERIC_INTERNAL_ERROR: übergeordneter Builder ist null – Diese Ausnahme wird möglicherweise angezeigt, wenn Sie eine Tabelle mit Spalten vom Datentyp `array` abfragen und Sie die OpenCSVSerDe-Bibliothek verwenden. Das OpenCSVSerde-Format unterstützt den `array`-Datentyp nicht.
- GENERIC_INTERNAL_ERROR: Wert überschreitet MAX_INT – Diese Ausnahme wird möglicherweise angezeigt, wenn die Quelldatenspalte mit dem Datentyp `INT` definiert ist und einen numerischen Wert größer als 2.147.483.647 hat.

- GENERIC_INTERNAL_ERROR: Wert überschreitet MAX_BYTE – Diese Ausnahme wird möglicherweise angezeigt, wenn die Quelldatenspalte einen numerischen Wert aufweist, der die zulässige Größe für den Datentyp `BYTE` überschreitet. Der Datentyp `BYTE` ist gleich `TINYINT`. `TINYINT` ist eine 8-Bit-Vorzeichen-Ganzzahl im Zweierkomplement-Format mit einem Mindestwert von -128 und einem Höchstwert von 127.
- GENERIC_INTERNAL_ERROR: Die Anzahl der Partitionswerte stimmt nicht mit der Anzahl der Filter überein – Diese Ausnahme wird möglicherweise angezeigt, wenn Sie inkonsistente Partitionen für Amazon Simple Storage Service (Amazon S3)-Daten haben. Unter den folgenden Bedingungen haben Sie möglicherweise inkonsistente Partitionen:
  - Partitionen auf Amazon S3 haben sich geändert (Beispiel: Neue Partitionen wurden hinzugefügt).
  - Die Anzahl der Partitionsspalten in der Tabelle stimmt nicht mit denen in den Partitionsmetadaten überein.

Weitere Informationen zu den einzelnen Fehlern finden Sie unter Wie behebe ich den Fehler „GENERIC_INTERNAL_ERROR“, wenn ich eine Tabelle in Amazon Athena abfrage? im AWS-Wissenscenter.

Anzahl übereinstimmender Gruppen stimmt nicht mit der Anzahl der Spalten überein

Dieser Fehler tritt auf, wenn Sie das Regex SerDe in einer CREATE-TABLE-Anweisung verwenden und die Anzahl der Regex-Übereinstimmungsgruppen nicht mit der Anzahl der Spalten übereinstimmt, die Sie für die Tabelle angegeben haben. Weitere Informationen finden Sie unter Wie behebe ich den RegexSerDe-Fehler „Anzahl übereinstimmender Gruppen stimmt nicht mit der Anzahl der Spalten überein“ in Amazon Athena? im AWS-Wissenscenter.

queryString konnte Einschränkung nicht erfüllen: Mitglied muss eine Länge von höchstens 262144 haben

Die maximale Länge der Abfragezeichenfolge in Athena (262.144 Byte) ist kein einstellbares Kontingent. AWS Support kann das Kontingent für Sie nicht erhöhen, aber Sie können das Problem umgehen, indem Sie lange Abfragen in kleinere aufteilen. Weitere Informationen finden Sie unter Wie kann ich die maximale Abfragezeichenfolgenlänge in Athena erhöhen? im AWS-Wissenscenter.

SYNTAX_ERROR: Spalte kann nicht gelöst werden

Dieser Fehler kann auftreten, wenn Sie eine Tabelle abfragen, die von einem AWS Glue-Crawler aus einer UTF-8-codierten CSV-Datei mit einer Bytereihenfolgemarkierung (BOM) erstellt wurde. AWS

Glue erkennt die Stücklisten nicht und ändert sie in Fragezeichen, die Amazon Athena nicht erkennt. Die Lösung besteht darin, das Fragezeichen in Athena oder in AWS Glue zu entfernen.

Zu viele Argumente für den Funktionsaufruf

In Athena-Engine-Version 3 können Funktionen nicht mehr als 127 Argumente akzeptieren. Diese Einschränkung ist konstruktionsbedingt. Wenn Sie eine Funktion mit mehr als 127 Parametern verwenden, wird eine Fehlermeldung wie die folgende angezeigt:

TOO_MANY_ARGUMENTS: Zeile *nnn*: *nn*: Zu viele Argumente für den Funktionsaufruf *function_name*().

Sie lösen dieses Problem, indem Sie weniger Parameter pro Funktionsaufruf verwenden.

Probleme mit dem Abfrage-Timeout

Wenn bei Ihren Athena-Abfragen Timeout-Fehler auftreten, überprüfen Sie Ihre CloudTrail-Protokolle. Bei Abfragen kann es aufgrund der Drosselung von AWS Glue oder Lake Formation APIs zu Timeouts kommen. Wenn diese Fehler auftreten, können die entsprechenden Fehlermeldungen eher auf ein Problem mit dem Abfrage-Timeout als auf ein Drosselungsproblem hinweisen. Um das Problem zu beheben, können Sie Ihre CloudTrail-Protokolle überprüfen, bevor Sie Kontakt mit AWS Support aufnehmen. Weitere Informationen erhalten Sie unter Abfragen von AWS CloudTrail-Protokollen und Protokollieren von Amazon-Athena-API-Aufrufen mit AWS CloudTrail.

Informationen zu Problemen mit dem Abfrage-Timeout bei Verbundabfragen beim Aufrufen der `ListTableMetadata`-API finden Sie unter Timeout beim Aufrufen von ListTableMetadata.

Probleme mit Drosselung

Wenn Ihre Abfragen die Grenzen abhängiger Services wie Amazon S3, AWS KMS, AWS Glue, oder AWS Lambda überschreiten, sind die folgenden Meldungen zu erwarten. Um diese Probleme zu beheben, reduzieren Sie die Anzahl der gleichzeitigen Anrufe, die von demselben Konto stammen.

Service	Fehlermeldung
AWS Glue	AWSGlueException: Rate überschritten.
AWS KMS	Sie haben die Rate überschritten, mit der Sie KMS aufrufen können. Verringern Sie die Häufigkeit Ihrer Aufrufe.

Service	Fehlermeldung
AWS Lambda	Quote überschritten. TooManyRequestsException
Amazon S3	Amazons3Exception: Bitte reduzieren Sie Ihre Anfragequote.

Informationen darüber, wie Sie die Drosselung von Amazon S3 bei der Verwendung von Athena verhindern können, finden Sie unter Drosselung durch Amazon S3 verhindern.

Ansichten

Ansichten, die in der Apache-Hive-Shell erstellt wurden, funktionieren in Athena nicht

Aufgrund ihrer grundlegend unterschiedlichen Implementierungen sind Ansichten, die in der Apache-Hive-Shell erstellt wurden, nicht mit Athena kompatibel. Um dieses Problem zu beheben, erstellen Sie die Ansichten in Athena erneut.

Ansicht ist veraltet; sie muss neu erstellt werden

Sie können diesen Fehler erhalten, wenn die Tabelle, die einer Ansicht zugrunde liegt, geändert oder gelöscht wurde. Die Auflösung besteht darin, die Ansicht neu zu erstellen. Weitere Informationen finden Sie unter Wie kann ich den Fehler „Ansicht ist veraltet; sie muss neu erstellt werden“ in Athena beheben? im AWS-Wissenscenter.

Arbeitsgruppen

Informationen zum Beheben von Problemen mit Arbeitsgruppen finden Sie unter Fehlerbehebung bei Arbeitsgruppen.

Weitere Ressourcen

Auf den folgenden Seiten finden Sie zusätzliche Informationen zur Behebung von Problemen mit Amazon Athena.

- Athena-Fehlerkatalog

- Service Quotas
- Überlegungen und Einschränkungen für SQL-Abfragen in Amazon Athena
- Nicht unterstützte DDLs
- Namen für Tabellen, Datenbanken und Spalten
- Datentypen in Amazon Athena
- Unterstützte SerDes- und Daten-Formate
- Athena-Komprimierungs-Support
- Reservierte Schlüsselwörter
- Fehlerbehebung bei Arbeitsgruppen

Folgende AWS-Ressourcen können auch hilfreich sein:

- Athena-Themen im AWS-Wissenscenter
- Amazon-Athena-Fragen zu AWS-re:Post
- Athena-Beiträge im AWS-Big-Data-Blog

Die Fehlerbehebung erfordert oft eine iterative Abfrage und Erkennung durch einen Experten oder eine Community von Helfern. Wenn Sie nach dem Ausprobieren der Vorschläge auf dieser Seite weiterhin Probleme haben, wenden Sie sich an AWS Support (in der AWS Management Console klicken Sie auf Support, Support Center (Supportcenter)) oder stellen Sie eine Frage zu AWS-re:Post unter Verwendung des Amazon-Athena-Tags.

Athena-Fehlerkatalog

Athena stellt standardisierte Fehlerinformationen bereit, mit denen Sie fehlgeschlagene Abfragen verstehen und Schritte unternehmen können, nachdem ein Abfragefehler aufgetreten ist. Das `AthenaError`-Feature beinhaltet ein `ErrorCategory`-Feld und `ErrorType`-Feld. `ErrorCategory` gibt an, ob die Ursache der fehlgeschlagenen Abfrage auf einen Systemfehler, einen Benutzerfehler oder einen anderen Fehler zurückzuführen ist. `ErrorType` liefert detailliertere Informationen über die Ursache des Fehlers. Durch die Kombination der beiden Felder können Sie die Umstände und Ursachen für den aufgetretenen spezifischen Fehler besser verstehen.

Fehlerkategorie

In der folgenden Tabelle finden Sie die Werte der Athena-Fehlerkategorie und ihre Bedeutungen.

Fehlerkategorie	Quelle
1	SYSTEM
2	USER
3	OTHER

Fehlertypreferenz

In der folgenden Tabelle finden Sie die Werte des Athena-Fehlertyps und ihre Bedeutungen.

Fehlertyp	Beschreibung
0	Abfrage erschöpfte Ressourcen in diesem Skalierungsfaktor
1	Abfrage erschöpfte Ressourcen in diesem Skalierungsfaktor
2	Abfrage erschöpfte Ressourcen in diesem Skalierungsfaktor
3	Abfrage erschöpfte Ressourcen in diesem Skalierungsfaktor
4	Abfrage erschöpfte Ressourcen in diesem Skalierungsfaktor
5	Abfrage erschöpfte Ressourcen in diesem Skalierungsfaktor
6	Abfrage erschöpfte Ressourcen in diesem Skalierungsfaktor
7	Abfrage erschöpfte Ressourcen in diesem Skalierungsfaktor
8	Abfrage erschöpfte Ressourcen in diesem Skalierungsfaktor
100	Interner Servicefehler
200	Bei der Abfrage-Engine trat ein interner Fehler auf
201	Bei der Abfrage-Engine trat ein interner Fehler auf
202	Bei der Abfrage-Engine trat ein interner Fehler auf

Fehlertyp	Beschreibung
203	Treiberfehler
204	Beim Metastore trat ein Fehler auf
205	Bei der Abfrage-Engine trat ein interner Fehler auf
206	Zeitlimit für Abfrage überschritten
207	Bei der Abfrage-Engine trat ein interner Fehler auf
208	Bei der Abfrage-Engine trat ein interner Fehler auf
209	Abfrage konnte nicht abgebrochen werden
210	Zeitlimit für Abfrage überschritten
211	Bei der Abfrage-Engine trat ein interner Fehler auf
212	Bei der Abfrage-Engine trat ein interner Fehler auf
213	Bei der Abfrage-Engine trat ein interner Fehler auf
214	Bei der Abfrage-Engine trat ein interner Fehler auf
215	Bei der Abfrage-Engine trat ein interner Fehler auf
216	Bei der Abfrage-Engine trat ein interner Fehler auf
217	Bei der Abfrage-Engine trat ein interner Fehler auf
218	Bei der Abfrage-Engine trat ein interner Fehler auf
219	Bei der Abfrage-Engine trat ein interner Fehler auf
220	Bei der Abfrage-Engine trat ein interner Fehler auf
221	Bei der Abfrage-Engine trat ein interner Fehler auf
222	Bei der Abfrage-Engine trat ein interner Fehler auf

Fehlertyp	Beschreibung
223	Bei der Abfrage-Engine trat ein interner Fehler auf
224	Bei der Abfrage-Engine trat ein interner Fehler auf
225	Bei der Abfrage-Engine trat ein interner Fehler auf
226	Bei der Abfrage-Engine trat ein interner Fehler auf
227	Bei der Abfrage-Engine trat ein interner Fehler auf
228	Bei der Abfrage-Engine trat ein interner Fehler auf
229	Bei der Abfrage-Engine trat ein interner Fehler auf
230	Bei der Abfrage-Engine trat ein interner Fehler auf
231	Bei der Abfrage-Engine trat ein interner Fehler auf
232	Bei der Abfrage-Engine trat ein interner Fehler auf
233	Iceberg-Fehler
234	Fehler bei der Lake Formation
235	Bei der Abfrage-Engine trat ein interner Fehler auf
236	Bei der Abfrage-Engine trat ein interner Fehler auf
237	Serialisierungsfehler
238	Metadaten konnten nicht in Amazon S3 hochgeladen werden
239	Allgemeiner Persistenzfehler
240	Abfrage konnte nicht gesendet werden
300	Interner Servicefehler
301	Interner Servicefehler

Fehlertyp	Beschreibung
302	Interner Servicefehler
303	Interner Servicefehler
400	Interner Servicefehler
401	Abfrageergebnisse konnten nicht in Amazon S3 geschrieben werden
402	Abfrageergebnisse konnten nicht in Amazon S3 geschrieben werden
1000	Benutzerfehler
1001	Datenfehler
1.002	Datenfehler
1003	DDL-Aufgabe fehlgeschlagen
1004	Schemafehler
1005	Serialisierungsfehler
1006	Syntaxfehler
1007	Datenfehler
1008	Abfrage abgelehnt
1009	Abfrage fehlgeschlagen
1010	Interner Servicefehler
1011	Abfrage vom Benutzer abgebrochen
1012	Bei der Abfrage-Engine trat ein interner Fehler auf
1013	Bei der Abfrage-Engine trat ein interner Fehler auf
1014	Abfrage vom Benutzer abgebrochen

Fehlertyp	Beschreibung
1100	Ungültiges Argument angegeben
1101	Ungültige Eigenschaft angegeben
1102	Bei der Abfrage-Engine trat ein interner Fehler auf
1103	Ungültige Tabelleneigenschaft angegeben
1104	Bei der Abfrage-Engine trat ein interner Fehler auf
1105	Bei der Abfrage-Engine trat ein interner Fehler auf
1106	Ungültiges Funktionsargument angegeben
1107	Ungültige Ansicht
1108	Funktion konnte nicht registriert werden
1109	Angegebener Amazon-S3-Pfad nicht gefunden
1110	Angegebene Tabelle oder Ansicht existiert nicht
1200	Abfrage nicht unterstützt
1201	Vorgesehener Decoder nicht unterstützt
1202	Abfragetyp nicht unterstützt
1300	Allgemeiner „Nicht gefunden“-Fehler
1301	Allgemeine Einheit nicht gefunden
1302	Datei nicht gefunden
1303	Bereitgestellte Funktion oder Funktionsimplementierung nicht gefunden
1304	Bei der Abfrage-Engine trat ein interner Fehler auf
1305	Bei der Abfrage-Engine trat ein interner Fehler auf

Fehlertyp	Beschreibung
1306	Amazon-S3-Bucket nicht gefunden
1307	Ausgewählte Engine nicht gefunden
1308	Bei der Abfrage-Engine trat ein interner Fehler auf
1400	Drosselungsfehler
1401	Abfrage aufgrund von AWS Glue-Drosselung fehlgeschlagen
1402	Abfrage aufgrund zu vieler Tabellenversionen in AWS Glue fehlgeschlagen
1403	Abfrage aufgrund von Amazon-S3-Drosselung fehlgeschlagen
1404	Abfrage aufgrund von Amazon-Athena-Drosselung fehlgeschlagen
1405	Abfrage aufgrund von Amazon-Athena-Drosselung fehlgeschlagen
1406	Abfrage aufgrund von Amazon-Athena-Drosselung fehlgeschlagen
1500	Berechtigungsfehler
1501	Amazon-S3-Berechtigungsfehler
1602	Das Limit der reservierten Kapazität wurde überschritten. Nicht genügend Kapazität, um diese Abfrage auszuführen.
9999	Interner Servicefehler

Codebeispiele

In den Beispielen in diesem Thema wird das SDK für Java 2.x als Ausgangspunkt für das Schreiben von Athena-Anwendungen verwendet.

> **Note**
>
> Informationen zum Programmieren von Athena mit anderen sprachspezifischen AWS-SDKs finden Sie in den folgenden Ressourcen:

- 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 v3 (`AthenaClient`)
- AWS SDK for PHP 3.x (`Aws\Athena`)
- AWS SDK for Python (Boto3) (`Athena.Client`)
- AWS SDK for Ruby v3 (`Aws::Athena::Client`)

Weitere Informationen zum Ausführen der Java-Codebeispiele in diesem Abschnitt finden Sie in der Amazon Athena-Java-Readme im AWS Codebeispiel-Repository auf GitHub. Die Java-Programmierreferenz für Athena finden Sie unter AthenaClient im AWS SDK for Java 2.x.

- Java-Codebeispiele
  - Konstanten
  - Erstellen eines Clients für den Zugriff auf Athena
  - Verwenden von Abfrageausführungen
    - Starten der Abfrageausführung
    - Anhalten der Abfrageausführung
    - Auflisten von Abfrageausführungen
  - Arbeiten mit benannten Abfragen
    - Erstellen einer benannten Abfrage
    - Löschen einer benannten Abfrage
    - Auflisten von Abfrageausführungen

> **Note**
>
> In diesen Beispielen werden Konstanten (wie z. B. `ATHENA_SAMPLE_QUERY`) für Zeichenfolgen verwendet, die in einer Klassendeklaration `ExampleConstants.java` definiert sind. Ersetzen Sie diese Konstanten durch eigene Zeichenfolgen oder definierte Konstanten.

Konstanten

Die Klasse `ExampleConstants.java` demonstriert, wie man eine Tabelle abfragt, die mit dem Tutorial Erste Schritte in Athena erstellt wurde.

```
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

}
```

Erstellen eines Clients für den Zugriff auf Athena

Die Klasse `AthenaClientFactory.java` zeigt, wie Sie einen Amazon-Athena-Client erstellen und konfigurieren.

```
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();
    }
}
```

Starten der Abfrageausführung

Das `StartQueryExample` zeigt, wie Sie eine Anfrage an Athena absenden, auf die Ergebnisse warten und dann die Ergebnisse verarbeiten.

```
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());
            }
        }
    }
}
```

Anhalten der Abfrageausführung

Das `StopQueryExecutionExample` führt eine Beispielabfrage aus, stoppt die Abfrage sofort und überprüft den Status der Abfrage, um sicherzustellen, dass sie abgebrochen wurde.

```
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;

    }
}
```

Auflisten von Abfrageausführungen

Das `ListQueryExecutionsExample` zeigt, wie Sie eine Liste von Abfrageausführungs-IDs abrufen.

```
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);
        }
    }
}
```

Erstellen einer benannten Abfrage

Das `CreateNamedQueryExample` zeigt, wie Sie eine benannte Abfrage erstellen.

```
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);
        }
    }
}
```

Löschen einer benannten Abfrage

Das `DeleteNamedQueryExample` zeigt, wie Sie eine benannte Abfrage mithilfe der ID der benannten Abfrage löschen.

```
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;
    }
}
```

Auflisten benannter Abfragen

Das `ListNamedQueryExample` zeigt, wie Sie eine Liste von IDs benannter Abfragen abrufen.

```
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);
        }
    }
}
```

Verwenden von Apache Spark in Amazon Athena

Amazon Athena vereinfacht die interaktive Ausführung von Datenanalysen und -erkundungen mithilfe von Apache Spark, ohne dass Sie Ressourcen planen, konfigurieren oder verwalten müssen. Das Ausführen von Apache-Spark-Anwendungen auf Athena bedeutet, dass Spark-Code zur Verarbeitung übermittelt und die Ergebnisse direkt empfangen werden, ohne dass eine zusätzliche Konfiguration erforderlich ist. Sie können die vereinfachte Notebook-Erfahrung in der Amazon-Athena-Konsole verwenden, um Apache-Spark-Anwendungen mit Python oder Athena-Notebook-APIs zu entwickeln. Apache Spark auf Amazon Athena ist Serverless und bietet eine automatische, bedarfsgerechte Skalierung, die sofortige Rechenleistung für wechselnde Daten-Volumes und Verarbeitungsanforderungen bereitstellt.

Amazon Athena bietet die folgenden Features:

- Verwendung der Konsole – Übermitteln Sie Ihre Spark-Anwendungen über die Amazon-Athena-Konsole.
- Scripting – Erstellen und debuggen Sie schnell und interaktiv Apache-Spark-Anwendungen in Python.
- Dynamische Skalierung – Amazon Athena bestimmt automatisch die Rechen- und Arbeitsspeicherressourcen, die zum Ausführen eines Auftrags erforderlich sind, und skaliert diese Ressourcen fortlaufend entsprechend bis zu den von Ihnen angegebenen Höchstwerten. Diese dynamische Skalierung reduziert die Kosten, ohne die Geschwindigkeit zu beeinträchtigen.
- Notebook-Erlebnis – Verwenden Sie den Athena-Notebook-Editor, um Berechnungen über eine vertraute Benutzeroberfläche zu erstellen, zu bearbeiten und auszuführen. Athena-Notebooks sind mit Jupyter Notebooks kompatibel und enthalten eine Liste von Zellen, die der Reihe nach als Berechnungen ausgeführt werden. Zelleninhalte können Code, Text, Markdown, Mathematik, Diagramme und Multimedia enthalten.

Weitere Informationen finden Sie unter Erkunden Sie Ihren Data Lake mit Amazon Athena für Apache Spark im AWS-Big-Data-Blog.

Überlegungen und Einschränkungen

- Derzeit ist Amazon Athena für Apache Spark in den folgenden AWS-Regionen verfügbar:
  - Asien-Pazifik (Mumbai)

  - Asien-Pazifik (Singapur)
  - Asien-Pazifik (Sydney)
  - Asien-Pazifik (Tokio)
  - Europe (Frankfurt)
  - Europa (Irland)
  - USA Ost (Nord-Virginia)
  - USA Ost (Ohio)
  - USA West (Oregon)
- AWS Lake Formation wird nicht unterstützt.
- Tabellen, die Partitionsprojektion verwenden, werden nicht unterstützt.
- Apache-Spark-fähige Arbeitsgruppen können den Athena-Notebook-Editor verwenden, aber nicht den Athena-Abfrage-Editor. Nur Athena-SQL-Arbeitsgruppen können den Athena-Abfrageeditor verwenden.
- Engine-übergreifende Ansichten-Abfragen werden nicht unterstützt. Mit Athena SQL erstellte Ansichten können von Athena für Spark nicht abgefragt werden. Da die Ansichten für die beiden Engines unterschiedlich implementiert sind, sind sie nicht für die Engine-übergreifende Verwendung kompatibel.
- MLlib (Apache Spark Machine Learning Library) und das -`pyspark.ml`Paket werden nicht unterstützt. Eine Liste der unterstützten Python-Bibliotheken finden Sie unter Liste der vorinstallierten Python-Bibliotheken.
- Derzeit `pip install` wird in Athena für Spark-Sitzungen nicht unterstützt.
- Pro Notebook ist nur eine aktive Sitzung zulässig.
- Wenn mehrere Benutzer die Konsole zum Öffnen einer vorhandenen Sitzung in einer Arbeitsgruppe verwenden, greifen diese auf dasselbe Notebook zu. Um Verwirrung zu vermeiden, öffnen Sie nur Sitzungen, die Sie selbst erstellt haben.
- Die Hosting-Domains für Apache Spark-Anwendungen, die Sie möglicherweise mit Amazon Athena verwenden (z. B. `analytics-gateway.us-east-1.amazonaws.com`), sind in der Internet Public Suffix List (PSL) registriert. Falls Sie jemals sensible Cookies in Ihren Domains einrichten müssen, empfehlen wir Ihnen, Cookies mit einem `__Host-`-Präfix zu verwenden, um Ihre Domain vor CSRF-Versuchen (Cross-Site Request Forgery) zu schützen. Weitere Informationen finden Sie auf der Set-Cookie-Seite in der Entwicklerdokumentation von Mozilla.org.
- Informationen zur Fehlerbehebung bei Spark-Notebooks, -Sitzungen und -Arbeitsgruppen in Athena finden Sie unter Fehlerbehebung in Athena für Spark.

Erste Schritte mit Apache Spark auf Amazon Athena

Um mit Apache Spark auf Amazon Athena beginnen zu können, müssen Sie zunächst eine Spark-fähige Arbeitsgruppe erstellen. Nachdem Sie zur Arbeitsgruppe gewechselt sind, können Sie ein Notebook erstellen oder ein vorhandenes Notebook öffnen. Wenn Sie ein Notebook in Athena öffnen, wird automatisch eine neue Sitzung dafür gestartet und Sie können direkt im Athena-Notebook-Editor damit arbeiten.

> **Note**
>
> Stellen Sie sicher, dass Sie eine Spark-fähige Arbeitsgruppe erstellen, bevor Sie versuchen, ein Notebook zu erstellen.

Erstellen einer Spark-fähigen Arbeitsgruppe in Athena

Sie können Arbeitsgruppen in Athena verwenden, um Benutzer, Teams, Anwendungen oder Workloads zu gruppieren und Kosten zu verfolgen. Um Apache Spark in Amazon Athena zu verwenden, erstellen Sie eine Amazon-Athena-Arbeitsgruppe, die eine Spark-Engine verwendet.

> **Note**
>
> Apache-Spark-fähige Arbeitsgruppen können den Athena-Notebook-Editor verwenden, aber nicht den Athena-Abfrage-Editor. Nur Athena-SQL-Arbeitsgruppen können den Athena-Abfrageeditor verwenden.

So erstellen Sie eine Spark-fähige Arbeitsgruppe in Athena

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.

3. Wählen Sie im Navigationsbereich die Option Arbeitsgruppen aus.
4. Wählen Sie auf der Seite Workgroups (Arbeitsgruppen) die Option Create workgroup (Arbeitsgruppe erstellen) aus.
5. Geben Sie als Workgroup name (Arbeitsgruppenname) einen Namen für Ihre Apache-Spark-Arbeitsgruppe ein.
6. (Optional) Geben Sie im Feld Description (Beschreibung) eine Beschreibung für Ihre Arbeitsgruppe ein.
7. Wählen Sie als Analytics engine (Analytik-Engine) die Option Apache Spark aus.

> **Note**
>
> Nachdem Sie eine Arbeitsgruppe erstellt haben, kann der Analytik-Engine-Typ der Arbeitsgruppe nicht mehr geändert werden. Beispielsweise kann eine Arbeitsgruppe der Athena-Engine-Version 3 nicht in eine Arbeitsgruppe der PySpark-Engine-Version 3 geändert werden.

8. Wählen Sie für die Zwecke dieses Tutorials Turn on example notebook (Beispiel-Notebook aktivieren) aus. Dieses optionale Feature fügt Ihrer Arbeitsgruppe ein Beispiel-Notebook mit dem Namen `example-notebook-random_string` hinzu und fügt AWS Glue-bezogene Berechtigungen, die das Notebook verwendet, um bestimmte Datenbanken und Tabellen in Ihrem Konto zu erstellen, anzuzeigen und zu löschen, sowie Leseberechtigungen in Amazon S3 für den Beispieldatensatz hinzu. Um die hinzugefügten Berechtigungen anzuzeigen, wählen Sie View additional permissions details (Details zu zusätzlichen Berechtigungen anzeigen) aus.

> **Note**
>
> Für den Betrieb des Beispiel-Notebooks können zusätzliche Kosten anfallen.

9. Führen Sie für Additional configurations (Zusätzliche Konfigurationen) einen der folgenden Schritte aus:

   - Verwenden Sie die Einstellung Use defaults (Standardwerte verwenden). Diese Option ist die Standardoption und hilft Ihnen beim Einstieg in Ihre Spark-fähige Arbeitsgruppe. Mit dieser Option erstellt Athena für Sie eine IAM-Rolle und einen Speicherort für Berechnungsergebnisse in Amazon S3. Der Name der IAM-Rolle und der zu erstellende S3-Bucket-Speicherort werden in dem Feld unter der Überschrift Additional configurations (Zusätzliche Konfigurationen) angezeigt.
   - Deaktivieren Sie die Einstellung Use defaults (Standardwerte verwenden) und fahren Sie anschließend mit den Schritten im Festlegen Ihrer eigenen Arbeitsgruppenkonfigurationen-Abschnitt fort, um Ihre Arbeitsgruppe manuell zu konfigurieren.

10. (Optional) Tags – Verwenden Sie diese Option, um Ihrer Arbeitsgruppe Tags hinzuzufügen. Weitere Informationen finden Sie unter Markieren von Athena-Ressourcen.
11. Wählen Sie Create workgroup (Arbeitsgruppe erstellen) aus. Eine Meldung informiert Sie darüber, dass die Arbeitsgruppe erfolgreich erstellt wurde, und die Arbeitsgruppe wird in der Liste der Arbeitsgruppen angezeigt.

Festlegen Ihrer eigenen Arbeitsgruppenkonfigurationen

Wenn Sie Ihre eigene IAM-Rolle und den Speicherort der Berechnungsergebnisse für Ihr Notebook festlegen möchten, folgen Sie den Schritten in diesem Abschnitt. Wenn Sie Use defaults (Standardwerte verwenden) für die Option Additional configurations (Zusätzliche Konfigurationen) ausgewählt haben, überspringen Sie diesen Abschnitt und gehen Sie direkt zu Öffnen des Notebook-Explorers und Wechseln der Arbeitsgruppen .

Das folgende Verfahren setzt voraus, dass Sie die Schritte 1 bis 9 des Verfahrens To create a Spark enabled workgroup in Athena (So erstellen Sie eine Spark-fähige Arbeitsgruppe in Athena) im vorherigen Abschnitt abgeschlossen haben.

So legen Sie Ihre eigenen Arbeitsgruppenkonfigurationen fest

1. Wenn Sie Ihre eigene IAM-Rolle erstellen oder verwenden oder die Notebook-Verschlüsselung konfigurieren möchten, erweitern Sie die IAM role configuration (IAM-Rollenkonfiguration).

   - Wählen Sie für Service Role (Servicerolle) einen der folgenden Schritte aus:
     - Eine Servicerolle erstellen – Wählen Sie diese Option aus, damit Athena eine Servicerolle für Sie erstellt. Um die Berechtigungen anzuzeigen, die die Rolle gewährt, wählen Sie View permission details (Berechtigungsdetails anzeigen).
     - Vorhandene Servicerolle auswählen – Wählen Sie aus dem Dropdown-Menü eine vorhandene Rolle aus. Die von Ihnen gewählte Rolle muss die Berechtigungen aus der ersten Option enthalten. Weitere Informationen über Berechtigungen für Notebooks finden Sie unter Fehlerbehebung bei Spark-fähigen Arbeitsgruppen.
   - Wählen Sie für die Notebook and calculation code encryption key management (Verwaltung von Notebook- und Berechnungscode-Verschlüsselungsschlüsseln) eine der folgenden Optionen aus:
     - Im Besitz von Amazon Athena – Der AWS KMS-Schlüssel ist im Besitz von Amazon Athena und wird von Amazon Athena verwaltet. Für die Verwendung dieses Schlüssels wird Ihnen keine zusätzliche Gebühr berechnet.
     - Ein in Ihrem Konto gespeicherter symmetrischer Schlüssel, der Ihnen gehört und von Ihnen verwaltet wird – Führen Sie für diese Option einen der folgenden Schritte aus:
       - Um einen vorhandenen Schlüssel zu verwenden, wählen Sie im Suchfeld ein AWS KMS aus oder geben Sie einen Schlüssel-ARN ein.
       - Um einen Schlüssel in der AWS KMS-Konsole zu erstellen, wählen Sie Einen AWS KMS-Schlüssel erstellen aus. Ihre Ausführungsrolle muss über die Berechtigung verfügen, den von Ihnen erstellten Schlüssel verwenden zu können.

   > **Important**
   >
   > Wenn Sie den AWS KMS key für eine Arbeitsgruppe ändern, verweisen Notebooks, die vor dem Update bearbeitet wurden, weiterhin auf den alten KMS-Schlüssel. Notebooks, die nach dem Update bearbeitet werden, verwenden den neuen KMS-Schlüssel. Um die alten Notebooks so zu aktualisieren, dass sie auf den neuen KMS-Schlüssel verweisen, exportieren und importieren Sie jedes der alten Notebooks. Wenn Sie den alten KMS-Schlüssel löschen, bevor Sie die alten Notebook-Referenzen

auf den neuen KMS-Schlüssel aktualisieren, sind die alten Notebooks nicht mehr entschlüsselbar und können nicht wiederhergestellt werden.
Dieses Verhalten gilt auch für Aktualisierungen von Aliassen, bei denen es sich um Anzeigenamen für KMS-Schlüssel handelt. Wenn Sie einen KMS-Schlüssel-Alias aktualisieren, um auf einen neuen KMS-Schlüssel zu verweisen, verweisen die vor der Alias-Aktualisierung verwalteten Notebooks weiterhin auf den alten KMS-Schlüssel und die nach der Alias-Aktualisierung verwalteten Notebooks verwenden den neuen KMS-Schlüssel. Berücksichtigen Sie diese Punkte, bevor Sie Ihre KMS-Schlüssel oder -Aliase aktualisieren.

2. Wenn Sie Ihre eigenen Einstellungen für Berechnungsergebnisse angeben möchten, erweitern Sie die Option Calculation result settings (Einstellungen für Berechnungsergebnisse) und wählen Sie dann eine der folgenden Optionen aus.

   - Neuen S3-Bucket erstellen – Mit dieser Option wird in Ihrem Konto ein Amazon-S3-Bucket für Ihre Berechnungsergebnisse erstellt. Der Bucket-Name besitzt das Format *account_id*-*region*-athena-results-bucket-*alphanumeric_id* und verwendet die Einstellungen ACLs deaktiviert, öffentlicher Zugriff gesperrt, Versionsverwaltung deaktiviert und Bucket-Eigentümer erzwungen.
   - Einen vorhandenen S3-Speicherort auswählen – Gehen Sie bei dieser Option wie folgt vor:
     - Geben Sie den S3-Pfad zu einem vorhandenen Speicherort in das Suchfeld ein oder wählen Sie Browse S3 (S3 durchsuchen) aus, um einen Bucket aus einer Liste auszuwählen.

       > **Note**
       >
       > Wenn Sie einen bereits vorhandenen Speicherort in Amazon S3 auswählen, fügen Sie dem Speicherort keinen nachgestellten Schrägstrich (/) hinzu. Dies führt dazu, dass der Link zum Speicherort der Berechnungsergebnisse auf der Seite mit den Berechnungsdetails auf das falsche Verzeichnis verweist. Bearbeiten Sie in diesem Fall den Speicherort der Ergebnisse der Arbeitsgruppe, um den nachgestellten Schrägstrich zu entfernen.

     - (Optional) Wählen Sie View (Anzeigen) aus, um die Seite Buckets der Amazon-S3-Konsole zu öffnen. Hier finden Sie weitere Informationen über den vorhandenen Bucket, den Sie ausgewählt haben.
     - (Optional) Geben Sie für Expected bucket owner (Erwarteter Bucket-Eigentümer) die AWS-Konto-ID ein, von der Sie erwarten, dass sie der Eigentümer des Buckets für den

Ausgabestandort Ihrer Abfrageergebnisse ist. Wir empfehlen Ihnen, diese Option nach Möglichkeit als zusätzliche Sicherheitsmaßnahme zu wählen. Wenn die Konto-ID des Bucket-Eigentümers nicht mit der von Ihnen angegebenen ID übereinstimmt, schlagen Versuche zur Ausgabe an den Bucket fehl. Ausführliche Informationen finden Sie unter Überprüfen der Bucket-Eigentümerschaft mit Bucket-Eigentümer-Bedingung im Amazon-S3-Benutzerhandbuch.

- (Optional) Wählen Sie Assign bucket owner full control over query results (Bucket-Eigetümer die volle Kontrolle über Abfrageergebnisse zuweisen) aus, wenn der Speicherort Ihrer Berechnungsergebnisse einem anderen Konto gehört und Sie dem anderen Konto die volle Kontrolle über Ihre Abfrageergebnisse geben möchten.

3. (Optional) Wählen Sie Encrypt calculation results (Berechnungsergebnisse verschlüsseln) aus, und wählen Sie dann eine der folgenden Optionen aus:

   - SSE_S3 – Dies ist ein von S3 verwalteter serverseitiger Verschlüsselungsschlüssel.
   - SSE_KMS – Ein von Ihnen bereitgestellter Schlüssel. Für Einen AWS KMS-Schlüssel auswählen können Sie eine der folgenden Optionen auswählen:
     - AWS-eigenen Schlüssel verwenden – Verwenden Sie einen Schlüssel, den AWS besitzt und für Sie verwaltet.
     - Einen anderen AWS KMS-Schlüssel auswählen (erweitert) – Wählen oder erstellen Sie einen Schlüssel.
       - Um einen vorhandenen Schlüssel zu verwenden, wählen Sie im Suchfeld ein AWS KMS aus oder geben Sie einen Schlüssel-ARN ein.
       - Um einen Schlüssel in der KMS-Konsole zu erstellen, wählen Sie Einen AWS KMS-Schlüssel erstellen aus. Nachdem Sie den Schlüssel in der KMS-Konsole erstellt haben, kehren Sie zur Seite Arbeitsgruppe erstellen in der Athena-Konsole zurück und verwenden Sie dann das Suchfeld Einen AWS KMS-Schlüssel auswählen oder einen ARN eingeben, um den soeben erstellten Schlüssel auszuwählen.

4. (Optional) Other settings (Weitere Einstellungen) – Erweitern Sie diese Option, um die Option Publish CloudWatch metrics (CloudWatch-Metriken veröffentlichen) für die Arbeitsgruppe zu aktivieren oder zu deaktivieren. Dieses Feld ist standardmäßig ausgewählt. Weitere Informationen finden Sie unter Überwachen von Apache-Spark-Berechnungen mit CloudWatch-Metriken.

5. (Optional) Tags – Verwenden Sie diese Option, um Ihrer Arbeitsgruppe Tags hinzuzufügen. Weitere Informationen finden Sie unter Markieren von Athena-Ressourcen.

6. Wählen Sie Create workgroup (Arbeitsgruppe erstellen) aus. Eine Meldung informiert Sie darüber, dass die Arbeitsgruppe erfolgreich erstellt wurde, und die Arbeitsgruppe wird in der Liste der Arbeitsgruppen angezeigt.

Öffnen des Notebook-Explorers und Wechseln der Arbeitsgruppen

Bevor Sie die soeben erstellte Spark-fähige Arbeitsgruppe verwenden können, müssen Sie zur Arbeitsgruppe wechseln. Um zwischen Spark-fähigen Arbeitsgruppen zu wechseln, können Sie die Option Workgroup (Arbeitsgruppe) im Notebook-Explorer oder Notebook-Editor verwenden.

> **Note**
>
> Vergewissern Sie sich bevor Sie anfangen, dass Ihr Browser Cookies von Drittanbietern nicht blockiert. Jeder Browser, der Cookies von Drittanbietern standardmäßig oder per Benutzereinstellung blockiert, verhindert das Starten von Notebooks. Weitere Informationen zum Verwalten von Cookies finden Sie unter:
>
> - Chrome
> - Firefox
> - Safari

So öffnen Sie den Notebook-Explorer und wechseln die Arbeitsgruppen

1. Wählen Sie im Navigationsbereich Notebook explorer (Notebook-Explorer) aus.
2. Verwenden Sie die Option Workgroup (Arbeitsgruppe) oben rechts in der Konsole, um die Spark-fähige Arbeitsgruppe auszuwählen, die Sie erstellt haben. Das Beispiel-Notebook wird in der Liste der Notebooks angezeigt.

   Sie können den Notebook-Explorer auf folgende Weise verwenden:

   - Wählen Sie den verknüpften Namen eines Notebooks, um das Notebook in einer neuen Sitzung zu öffnen.
   - Verwenden Sie das Menü Actions (Aktionen), um Ihr Notebook umzubenennen, zu löschen oder zu exportieren.
   - Um eine Notebook-Datei zu importieren, wählen Sie Import file (Datei importieren).
   - Um ein Notebook zu erstellen, wählen Sie Create notebook (Notebook erstellen) aus.

Ausführen des Beispiel-Notebooks

Das Beispiel-Notebook fragt Daten aus einem öffentlich zugänglichen Datensatz für Taxifahrten in New York City ab. Das Notebook enthält Beispiele, die zeigen, wie Sie mit Spark DataFrames, Spark SQL und AWS Glue Data Catalog arbeiten.

So führen Sie das Beispiel-Notebook aus

1. Wählen Sie im Notebook-Explorer den verknüpften Namen des Beispiel-Notebooks aus.

   Dadurch wird eine Notebook-Sitzung mit Standardparametern gestartet und das Notebook im Notebook-Editor geöffnet. Eine Meldung informiert Sie darüber, dass eine neue Apache-Spark-Sitzung mit Standardparametern (maximal 20 DPUs) gestartet wurde.
2. Um die Zellen der Reihe nach auszuführen und die Ergebnisse zu überwachen, wählen Sie einmal für jede Zelle des Notebooks die Schaltfläche Run (Ausführen) aus.

   - Scrollen Sie nach unten, um die Ergebnisse anzuzeigen und neue Zellen einzublenden.
   - Für die Zellen, die eine Berechnung enthalten, zeigt ein Fortschrittsbalken den abgeschlossenen Prozentsatz, die verstrichene Zeit und die verbleibende Zeit an.
   - Das Beispiel-Notebook erstellt eine Beispieldatenbank und -tabelle in Ihrem Konto. Die letzte Zelle entfernt diese in einem Schritt der Datenbereinigung.

> **Note**
>
> Wenn Sie Ordner-, Tabellen- oder Datenbanknamen im Beispiel-Notebook ändern, stellen Sie sicher, dass diese Änderungen in den von Ihnen verwendeten IAM-Rollen widergespiegelt werden. Andernfalls kann das Notebook aufgrund unzureichender Berechtigungen nicht ausgeführt werden.

Bearbeiten von Sitzungsdetails

Nachdem Sie eine Notebook-Sitzung gestartet haben, können Sie Sitzungsdetails wie Tabellenformat, Verschlüsselung, Leerlaufzeit der Sitzung und die maximale Anzahl der Datenverarbeitungseinheiten (DPUs), die Sie gleichzeitig verwenden möchten, bearbeiten. Bei einer DPU handelt es sich um ein relatives Maß der Rechenleistung, die aus 4 vCPUs Rechenkapazität und 16 GB Arbeitsspeicher besteht.

So bearbeiten Sie Sitzungsdetails

1. Wählen Sie im Notebook-Editor aus dem Menü Session (Sitzung) oben rechts die Option Edit session (Sitzung bearbeiten) aus.
2. Wählen Sie im Dialogfeld Sitzungsdetails bearbeiten im Abschnitt Spark-Parameter Werte für die folgenden Optionen aus oder geben Sie sie ein:

   - Zusätzliches Tabellenformat – Wählen Sie Linux Foundation Delta Lake, Apache Hudi, Apache Iceberg oder Benutzerdefiniert.
     - Für die Tabellenoptionen Delta, Hudi oder Iceberg werden Ihnen die erforderlichen Tabelleneigenschaften für das entsprechende Tabellenformat automatisch in den Optionen In Tabelle bearbeiten und In JSON bearbeiten zur Verfügung gestellt. Weitere Informationen zum Verwenden dieser Tabellenformate finden Sie unter Nicht-Hive-Tabellenformaten in Amazon Athena für Apache Spark verwenden.
     - Um Tabelleneigenschaften für die benutzerdefinierte oder andere Tabellenarten hinzuzufügen oder zu entfernen, verwenden Sie die Optionen In Tabelle bearbeiten und In JSON bearbeiten.
     - Wählen Sie für die Option In Tabelle bearbeiten die Option Eigenschaft hinzufügen aus, um eine Eigenschaft hinzuzufügen, oder wählen Sie Entfernen, um eine Eigenschaft zu entfernen. Verwenden Sie die Felder Schlüssel und Wert, um Eigenschaftsnamen und ihre Werte einzugeben.
     - Verwenden Sie für die Option In JSON bearbeiten den JSON-Texteditor, um die Konfiguration direkt zu bearbeiten.
       - Wählen Sie zum Kopieren des JSON-Textes in die Zwischenablage Kopieren aus.
       - Wählen Sie Löschen, um den gesamten Text aus dem JSON-Editor zu entfernen.
       - Wählen Sie das Einstellungssymbol (Zahnrad), um den Zeilenumbruch zu konfigurieren, oder wählen Sie ein Farbdesign für den JSON-Editor.
   - Spark-Verschlüsselung aktivieren – Wählen Sie diese Option, um Daten zu verschlüsseln, die auf die Festplatte geschrieben und über Spark-Netzwerkknoten gesendet werden. Weitere Informationen finden Sie unter Apache-Spark-Verschlüsselung aktivieren.
3. Wählen Sie im Abschnitt Sitzungsparameter die Werte für die folgenden Optionen aus oder geben Sie sie ein:

   - Session idle timeout (Zeitüberschreitung bei Sitzungsleerlauf) – Wählen Sie einen Wert zwischen 1 und 480 Minuten aus oder geben Sie ihn ein. Der Standardwert ist 20.

- Coordinator size (Größe des Koordinators) – Ein Koordinator ist ein spezieller Executor, der die Verarbeitungsarbeit orchestriert und andere Executors in einer Notebook-Sitzung verwaltet. Derzeit ist 1 DPU der Standardwert und der einzig mögliche Wert.
- Executor size (Größe des Executors) – Ein Executor ist die kleinste Recheneinheit, die eine Notebook-Sitzung von Athena anfragen kann. Derzeit ist 1 DPU der Standardwert und der einzig mögliche Wert.
- Max concurrent value (Maximaler gleichzeitiger Wert) – Die maximale Anzahl von DPUs, die gleichzeitig ausgeführt werden können. Der Standardwert ist 20, der Mindestwert ist 3 und der Höchstwert ist 60. Wenn Sie diesen Wert erhöhen, werden zusätzliche Ressourcen nicht automatisch zugewiesen. Stattdessen wird Athena versuchen, die Ressourcen bis zum angegebenen Höchstwert zuzuweisen, sofern die Rechenlast dies erfordert und die Ressourcen verfügbar sind.

4. Wählen Sie Save (Speichern).
5. Wählen Sie bei der Aufforderung zur Confirm edit (Änderung bestätigen) die Option Confirm (Bestätigen) aus.

   Athena speichert Ihr Notebook und startet eine neue Sitzung mit den von Ihnen angegebenen Parametern. Ein Banner im Notebook-Editor informiert Sie darüber, dass eine neue Sitzung mit den geänderten Parametern gestartet wurde.

   > **Note**
   >
   > Athena merkt sich Ihre Sitzungseinstellungen für dieses Notebook. Wenn Sie die Parameter einer Sitzung bearbeiten und dann die Sitzung beenden, verwendet Athena die Sitzungsparameter, die Sie beim nächsten Start einer Sitzung für das Notebook konfiguriert haben.

Anzeigen von Sitzungs- und Berechnungsdetails

Nachdem Sie das Notebook ausgeführt haben, können Sie Ihre Sitzungs- und Berechnungsdetails anzeigen.

So zeigen Sie Sitzungs- und Berechnungsdetails an

1. Wählen Sie im Menü Session (Sitzung) oben rechts die Option View details (Details anzeigen) aus.

- Auf der Registerkarte Current session (Aktuelle Sitzung) werden Informationen zur aktuellen Sitzung angezeigt, einschließlich Sitzungs-ID, Erstellungszeit, Status und Arbeitsgruppe.
- Auf der Registerkarte History (Verlauf) werden die Sitzungs-IDs früherer Sitzungen aufgelistet. Um die Details einer vorherigen Sitzung anzuzeigen, wählen Sie die Registerkarte History (Verlauf) und wählen Sie dann eine Sitzungs-ID aus der Liste aus.
- Der Abschnitt Calculations (Berechnungen) zeigt eine Liste der Berechnungen, die in der Sitzung ausgeführt wurden.

2. Um die Details einer Berechnung anzuzeigen, wählen Sie die Berechnungs-ID aus.
3. Auf der Seite Calculation details (Berechnungsdetails) können Sie Folgendes tun:

- Den Code für die Berechnung finden Sie im Abschnitt Code.
- Um die Ergebnisse der Berechnung anzuzeigen, wählen Sie die Registerkarte Results (Ergebnisse).
- Um die angezeigten Ergebnisse im Textformat herunterzuladen, wählen Sie Download results (Ergebnisse herunterladen) aus.
- Um Informationen zu den Berechnungsergebnissen in Amazon S3 anzuzeigen, wählen Sie View in S3 (In S3 anzeigen) aus.

Beenden einer Sitzung

So beenden Sie eine Notebook-Sitzung

1. Wählen Sie im Notebook-Editor im Menü Session (Sitzung) oben rechts die Option Terminate (Beenden) aus.
2. Wählen Sie bei der Aufforderung Confirm session termination (Beenden der Sitzung bestätigen) die Option Confirm (Bestätigen) aus. Ihr Notebook wird gespeichert und Sie kehren zum Notebook-Editor zurück.

> **Note**
>
> Mit dem Schließen der Notebook-Registerkarte im Notebook-Editor wird die Sitzung für ein aktives Notebook nicht automatisch beendet. Wenn Sie sicherstellen möchten, dass die Sitzung beendet wird, verwenden Sie die Optionen Session (Sitzung) und Terminate (Beenden).

Erstellen Ihres eigenen Notebooks

Nachdem Sie eine Spark-fähige Athena-Arbeitsgruppe erstellt haben, können Sie Ihr eigenes Notebook erstellen.

So erstellen Sie ein Notebook

1. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.
2. Wählen Sie im Navigationsbereich der Athena-Konsole den Notebook explorer (Notebook-Explorer) oder den Notebook editor (Notebook-Editor) aus.
3. Führen Sie eine der folgenden Aktionen aus:

   - Wählen Sie im Notebook explorer (Notebook-Explorer) die Option Create notebook (Notebook erstellen) aus.
   - Wählen Sie im Notebook explorer (Notebook-Editor) die Option Create notebook (Notebook erstellen) oder klicken Sie auf das Plussymbol (+), um ein Notebook hinzuzufügen.
4. Geben Sie im Dialogfeld Create notebook (Notebook erstellen) unter Notebook name (Notebook-Name) einen Namen ein.
5. (Optional) Erweitern Sie die Sitzungsparameter und wählen Sie dann Werte für die folgenden Optionen aus, oder geben Sie diese ein:

   - Zusätzliches Tabellenformat – Wählen Sie Linux Foundation Delta Lake, Apache Hudi, Apache Iceberg oder Benutzerdefiniert.
     - Für die Tabellenoptionen Delta, Hudi oder Iceberg werden Ihnen die erforderlichen Tabelleneigenschaften für das entsprechende Tabellenformat automatisch in den Optionen In Tabelle bearbeiten und In JSON bearbeiten zur Verfügung gestellt. Weitere Informationen zum Verwenden dieser Tabellenformate finden Sie unter Nicht-Hive-Tabellenformaten in Amazon Athena für Apache Spark verwenden.
     - Um Tabelleneigenschaften für die benutzerdefinierte oder andere Tabellenarten hinzuzufügen oder zu entfernen, verwenden Sie die Optionen In Tabelle bearbeiten und In JSON bearbeiten.
     - Wählen Sie für die Option In Tabelle bearbeiten die Option Eigenschaft hinzufügen aus, um eine Eigenschaft hinzuzufügen, oder wählen Sie Entfernen, um eine Eigenschaft zu entfernen. Verwenden Sie die Felder Schlüssel und Wert, um Eigenschaftsnamen und ihre Werte einzugeben.

- Verwenden Sie für die Option In JSON bearbeiten den JSON-Texteditor, um die Konfiguration direkt zu bearbeiten.
  - Wählen Sie zum Kopieren des JSON-Textes in die Zwischenablage Kopieren aus.
  - Wählen Sie Löschen, um den gesamten Text aus dem JSON-Editor zu entfernen.
  - Wählen Sie das Einstellungssymbol (Zahnrad), um den Zeilenumbruch zu konfigurieren, oder wählen Sie ein Farbdesign für den JSON-Editor.
- Spark-Verschlüsselung aktivieren – Wählen Sie diese Option, um Daten zu verschlüsseln, die auf die Festplatte geschrieben und über Spark-Netzwerkknoten gesendet werden. Weitere Informationen finden Sie unter Apache-Spark-Verschlüsselung aktivieren.

6. (Optional) Erweitern Sie die Session parameters (Sitzungsparameter) und wählen Sie dann Werte für die folgenden Optionen aus, oder geben Sie diese ein:

   - Session idle timeout (Zeitüberschreitung bei Sitzungsleerlauf) – wählen Sie einen Wert zwischen 1 und 480 Minuten aus oder geben Sie diesen ein. Der Standardwert ist 20.
   - Coordinator size (Größe des Koordinators) – Ein Koordinator ist ein spezieller Executor, der die Verarbeitungsarbeit orchestriert und andere Executors in einer Notebook-Sitzung verwaltet. Derzeit ist 1 DPU der Standardwert und der einzig mögliche Wert. Eine DPU (Data Processing Unit) ist ein relatives Maß für die Rechenleistung, die aus 4 vCPUs Rechenkapazität und 16 GB Arbeitsspeicher besteht.
   - Executor size (Größe des Executor) – Ein Executor ist die kleinste Recheneinheit, die eine Notebook-Sitzung von Athena anfragen kann. Derzeit ist 1 DPU der Standardwert und der einzig mögliche Wert.
   - Max concurrent value (Maximaler gleichzeitiger Wert) – Die maximale Anzahl von DPUs, die gleichzeitig ausgeführt werden können. Der Standardwert ist 20 und der Höchstwert ist 60. Wenn Sie diesen Wert erhöhen, werden zusätzliche Ressourcen nicht automatisch zugewiesen. Stattdessen wird Athena versuchen, die Ressourcen bis zum angegebenen Höchstwert zuzuweisen, sofern die Rechenlast dies erfordert und die Ressourcen verfügbar sind.

7. Wählen Sie Erstellen aus. Ihr Notebook wird in einer neuen Sitzung im Notebook-Editor geöffnet.

Öffnen eines zuvor erstellten Notebooks

So öffnen Sie ein zuvor erstelltes Notebook

1. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.
2. Wählen Sie im Navigationsbereich der Athena-Konsole den Notebook editor (Notebook-Editor) oder den Notebook explorer (Notebook-Explorer) aus.
3. Führen Sie eine der folgenden Aktionen aus:

   - Wählen Sie im Notebook editor (Notebook-Editor) ein Notebook aus der Liste Recent notebooks (Zuletzt verwendete Notebooks) oder Saved notebooks (Gespeicherte Notebooks) aus. Das Notebook öffnet sich in einer neuen Sitzung.
   - Wählen Sie im Notebook explorer (Notebook-Explorer) den Namen eines Notebooks aus der Liste aus. Das Notebook öffnet sich in einer neuen Sitzung.

Weitere Informationen zur Verwaltung Ihrer Notebook-Dateien finden Sie unter Verwalten von Notebook-Dateien.

Arbeiten mit Notebooks

Sie verwalten Ihre Notebooks im Athena-Notebook-Explorer und bearbeiten und führen sie in Sitzungen mit dem Athena-Notebook-Editor aus. Sie können die DPU-Nutzung für Ihre Notebook-Sitzungen gemäß Ihren Anforderungen konfigurieren.

Wenn Sie ein Notebook anhalten, beenden Sie die zugehörige Sitzung. Alle Dateien werden gespeichert, aber laufende Änderungen an deklarierten Variablen, Funktionen und Klassen gehen verloren. Wenn Sie das Notebook neu starten, lädt Athena die Notebook-Datei neu und Sie können Ihren Code erneut ausführen.

Sitzungen und Berechnungen

Jedes Notebook ist einem einzelnen Python-Kernel zugeordnet und führt Python-Code aus. Ein Notebook kann eine oder mehrere Zellen beinhalten, die Befehle enthalten. Um die Zellen in einem Notebook auszuführen, erstellen Sie zunächst eine Sitzung für das Notebook. In Sitzungen werden die Variablen und der Status der Notebooks nachverfolgt.

Das Ausführen einer Zelle in einem Notebook bedeutet das Ausführen einer Berechnung in der aktuellen Sitzung. Berechnungen verbessern den Status des Notebooks und können Aufgaben wie das Lesen aus Amazon S3 oder das Schreiben in andere Datenspeicher ausführen. Solange eine Sitzung ausgeführt wird, verwenden und ändern Berechnungen den Status, der für das Notebook verwaltet wird.

Wenn Sie den Status nicht mehr benötigen, können Sie eine Sitzung beenden. Wenn Sie eine Sitzung beenden, bleibt das Notebook erhalten, aber die Variablen und andere Statusinformationen werden zerstört. Wenn Sie an mehreren Projekten gleichzeitig arbeiten müssen, können Sie für jedes Projekt eine Sitzung erstellen, und die Sitzungen sind voneinander unabhängig.

Sitzungen verfügen über eine dedizierte Rechenkapazität, gemessen in DPU. Beim Erstellen einer Sitzung können Sie der Sitzung eine Reihe von DPUs zuweisen. Unterschiedliche Sitzungen können abhängig von den Anforderungen der Aufgabe unterschiedliche Kapazitäten haben.

Verwenden des Athena-Notebook-Editors

Der Athena-Notebook-Editor ist eine interaktive Umgebung für das Schreiben und Ausführen von Code. Die folgenden Abschnitte beschreiben die Features der Umgebung.

Befehlsmodus im Vergleich zu Bearbeitungsmodus

Der Notebook-Editor verfügt über eine modale Benutzeroberfläche: einen Bearbeitungsmodus für die Eingabe von Text in eine Zelle und einen Befehlsmodus für die Ausgabe von Befehlen an den Editor selbst wie Kopieren, Einfügen oder Ausführen.

Um den Bearbeitungsmodus und den Befehlsmodus zu verwenden, können Sie die folgenden Aufgaben ausführen:

- Um in den Bearbeitungsmodus zu wechseln, drücken Sie **ENTER** oder wählen Sie eine Zelle aus. Wenn sich eine Zelle im Bearbeitungsmodus befindet, hat die Zelle einen grünen linken Rand.
- Um in den Befehlsmodus zu wechseln, drücken Sie **ESC** oder klicken Sie außerhalb einer Zelle. Beachten Sie, dass Befehle normalerweise nur für die aktuell ausgewählte Zelle gelten, nicht für alle Zellen. Wenn sich der Editor im Befehlsmodus befindet, hat die Zelle einen blauen linken Rand.
- Im Befehlsmodus können Sie Tastenkombinationen und das Menü über dem Editor verwenden, aber keinen Text in einzelne Zellen eingeben.
- Um eine Zelle auszuwählen, klicken Sie auf die Zelle.
- Um alle Zellen auszuwählen, drücken Sie **Ctrl+A** (Windows) oder **Cmd+A** (Mac).

Menü des Notebook-Editors

Die Symbole im oberen Menü des Notebook-Editors bieten die folgenden Optionen:

- Speichern – Speichert den aktuellen Status des Notebooks.
- Zelle unterhalb einfügen – Fügt eine neue (leere) Zelle unterhalb der aktuell ausgewählten ein.
- Ausgewählte Zellen ausschneiden – Entfernt die ausgewählte Zelle von ihrem aktuellen Standort und kopiert die Zelle in den Speicher.
- Ausgewählte Zellen kopieren – Kopiert die ausgewählte Zelle in den Speicher.
- Zellen unterhalb einfügen – Fügt die kopierte Zelle unterhalb der aktuellen Zelle ein.
- Ausgewählte Zellen nach oben verschieben – Verschiebt die aktuelle Zelle über die darüber liegende Zelle.
- Ausgewählte Zellen nach unten verschieben – Verschiebt die aktuelle Zelle unter die darunter liegende Zelle.
- Ausführen – Führt die aktuelle (ausgewählte) Zelle aus. Die Ausgabe wird unmittelbar unter der aktuellen Zelle angezeigt.
- Alle ausführen – Führt alle Zellen im Notebook aus. Die Ausgabe für jede Zelle wird unmittelbar unter der aktuellen Zelle angezeigt.
- Stoppen (Kernel unterbrechen) – Stoppt das aktuelle Notebook durch Unterbrechen des Kernels.
- Formatierungsoption – Wählt das Zellenformat aus, das eines der folgenden sein kann:
  - Code – Wird für Python-Code (die Voreinstellung) verwendet.
  - Markdown – Zur Eingabe von Text im GitHub-ähnlichen Markdown-Format. Um den Markdown zu rendern, führen Sie die Zelle aus.
  - Raw NbConvert – Wird für die Eingabe von Text in unveränderter Form verwendet. Zellen, die als Raw NBConvert gekennzeichnet sind, können mit dem Jupyter-Befehlszeilentool nbconvert in ein anderes Format wie HTML konvertiert werden.
- Überschrift – Dient zum Ändern der Überschriftenebene der Zelle.
- Befehlspalette – Enthält Jupyter-Notebook-Befehle und ihre Tastenkombinationen. Weitere Informationen zu den Tastenkombinationen finden Sie in den Abschnitten weiter unten in diesem Dokument.
- Sitzung – Verwenden Sie die Optionen in diesem Menü, um die Details einer Sitzung anzuzeigen, Sitzungsparameter zu bearbeiten oder die Sitzung zu beenden.

Tastenkombinationen für den Befehlsmodus

Im Folgenden finden Sie einige gängige Tastenkombinationen für den Befehlsmodus des Notebook-Editors. Diese Tastenkombinationen sind verfügbar, nachdem Sie **ESC** gedrückt haben, um in den Befehlsmodus zu wechseln. Um eine vollständige Liste der im Editor verfügbaren Befehle anzuzeigen, drücken Sie **ESC + H**.

Schlüssel	Action
1 - 6	Ändern Sie den Zellentyp in Markdown und stellen Sie die Überschriftsebene auf die eingegebene Zahl ein
a	Erstellt eine Zelle über der aktuellen Zelle
b	Erstellt eine Zelle unter der aktuellen Zelle
c	Kopiert die aktuelle Zelle in den Speicher
d d	Löscht die aktuelle Zelle
h	Zeigt den Hilfebildschirm für Tastaturkürzel an
j	Eine Zelle nach unten
k	Eine Zelle nach oben
m	Ändert das aktuelle Zellenformat zu Markdown
r	Ändert das aktuelle Zellenformat zu Raw
s	Speichert das Notebook
v	Fügt Speicherinhalte unter die aktuelle Zelle ein
x	Schneidet die ausgewählte(n) Zelle oder Zellen aus
y	Ändert das Zellenformat zu Code
z	Rückgängig

Schlüssel	Action
Ctrl+Enter	Führt die aktuelle Zelle aus und ruft den Befehlsmodus auf
Shift+Enter oder **Alt+Enter**	Führt die aktuelle Zelle aus, erstellt eine neue Zelle unterhalb der Ausgabe und ruft die neue Zelle im Bearbeitungsmodus auf
Space	Seite nach unten
Shift+Space	Seite nach oben
Shift + L	Schaltet die Sichtbarkeit von Zeilennummern in Zellen ein oder aus

Bearbeiten von Tastenkürzeln im Befehlsmodus

Der Notebook-Editor verfügt über eine Option zum Anpassen der Tastenkombinationen für den Befehlsmodus.

So bearbeiten Sie Tastenkombinationen im Befehlsmodus

1. Wählen Sie im Menü des Notebook-Editors die Command palette (Befehlspalette) aus.
2. Wählen Sie in der Befehlspalette den Befehl Edit command mode keyboard shortcuts (Tastenkombinationen für den Befehlsmodus bearbeiten) aus.
3. Verwenden Sie die Schnittstelle Edit command mode shortcuts (Befehlsmodus-Kurzbefehle bearbeiten), um der Tastatur Befehle zuzuordnen oder neu zuzuordnen.

   Um Anweisungen zum Bearbeiten von Tastenkombinationen für den Befehlsmodus anzuzeigen, scrollen Sie zum unteren Rand des Bildschirms Edit command mode shortcuts (Tastenkombinationen für den Befehlsmodus bearbeiten).

Hinweise zur Verwendung von Magic-Befehlen in Athena für Apache Spark finden Sie unter Verwenden von magischen Befehlen.

Verwenden von magischen Befehlen

Magische Befehle oder Magics sind spezielle Befehle, die Sie in einer Notebook-Zelle ausführen können. Beispielsweise zeigt `%env` die Umgebungsvariablen in einer Notebook-Sitzung an. Athena unterstützt die Magic-Funktionen in IPython 6.0.3.

Dieser Abschnitt zeigt einige wichtige Magics-Befehle in Athena für Apache Spark.

- Um eine Liste von Magics-Befehlen in Athena anzuzeigen, führen Sie den Befehl **`%lsmagic`** in einer Notebook-Zelle aus.
- Informationen zur Verwendung von Magics zur Erstellung von Diagrammen in Athena-Notebooks finden Sie unter Magics zum Erstellen von Datendiagrammen.
- Weitere Informationen zu den integrierten Magics-Befehlen finden Sie unter Integrierte Magics-Befehle in der IPython-Dokumentation.

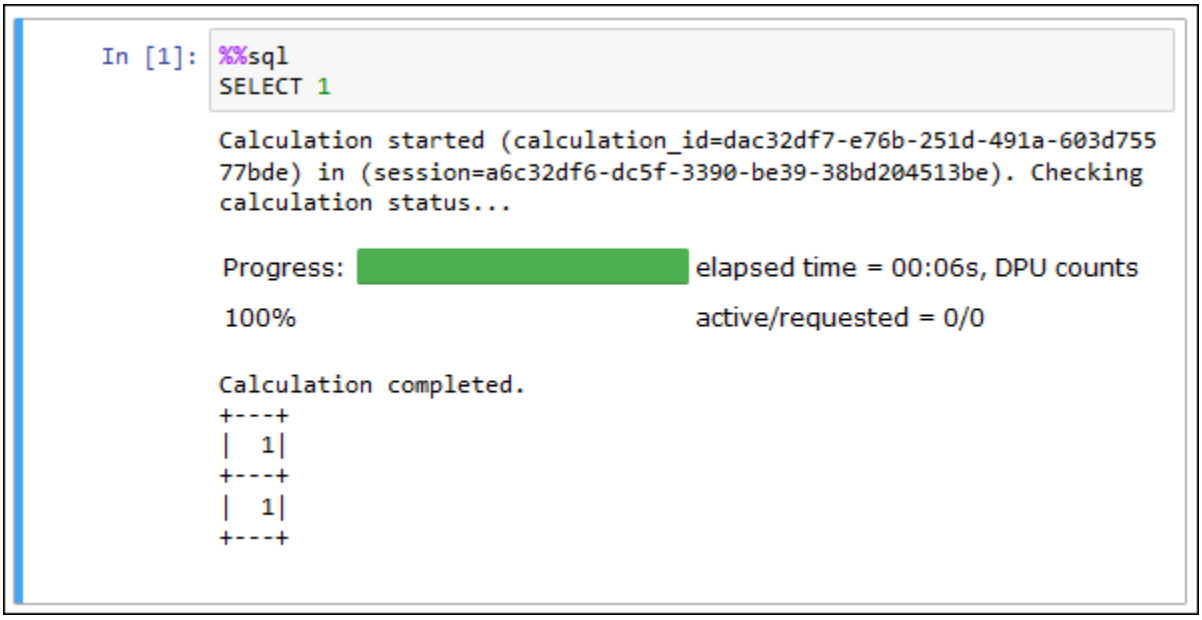
> **Note**
>
> Derzeit schlägt der `%pip`-Befehl bei der Ausführung fehl. Dies ist ein bekanntes Problem.

Zellen-Magics

Magics, die sich über mehrere Zeilen verteilen, wird ein doppeltes Prozentzeichen (`%%`) vorangestellt und sie werden als Zellen-Magic-Funktionen oder Zellen-Magics bezeichnet.

%%sql

Diese Zellen-Magics ermöglicht es, SQL-Anweisungen direkt auszuführen, ohne sie mit einer Spark-SQL-Anweisung verzieren zu müssen. Der Befehl zeigt auch die Ausgabe an, indem er implizit den zurückgegebenen Datenrahmen `.show()` aufruft.



Der `%%sql`-Befehl kürzt Spaltenausgaben automatisch auf eine Breite von 20 Zeichen. Diese Einstellung ist derzeit nicht konfigurierbar. Um diese Einschränkung zu umgehen, verwenden Sie die folgende vollständige Syntax und ändern Sie die Parameter der `show`-Methode entsprechend.

```
spark.sql("""YOUR_SQL""").show(n=number, truncate=number, vertical=bool)
```

- n `int`, optional. Die Anzahl der anzuzeigenden Zeilen.
- kürzt – `bool` oder `int`, optional – wenn `true`, kürzt Zeichenketten, die länger als 20 Zeichen sind. Wenn dieser Wert auf eine Zahl größer als 1 gesetzt ist, werden lange Zeichenketten auf die angegebene Länge gekürzt und die Zellen werden rechtsbündig ausgerichtet.
- vertikal – `bool`, optional. Wenn `true`, werden die Ausgabezeilen vertikal ausgedruckt (eine Zeile pro Spaltenwert).

Linien-Magics

Magics, die sich auf einer einzelnen Zeile befinden, wird ein Prozentzeichen (%) vorangestellt und sie werden als Zeilen-Magic-Funktionen oder Zeilen-Magics bezeichnet.

%help

Zeigt Beschreibungen der verfügbaren Magic-Befehle an.

```
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

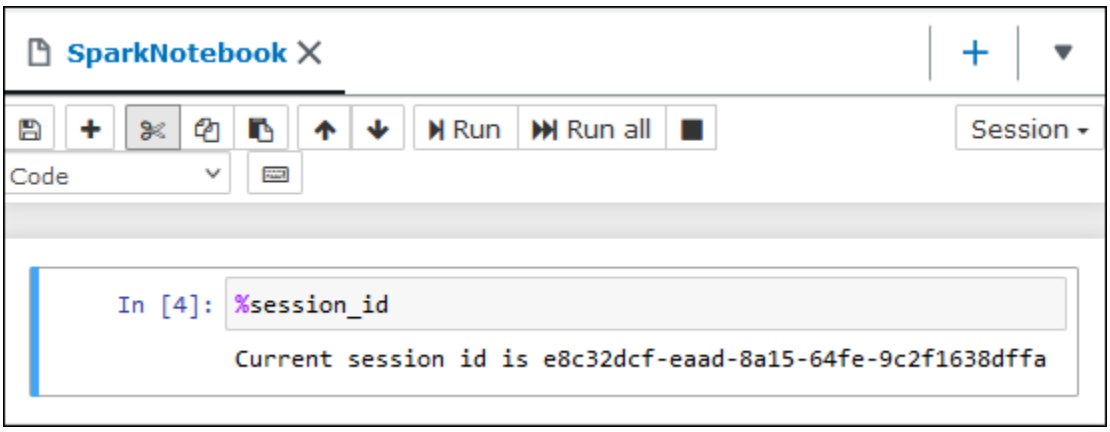
Listet die mit dem Noebook verknüpften Sitzungen auf. Zu den Informationen für jede Sitzung gehören die Sitzungs-ID, der Sitzungsstatus sowie Datum und Uhrzeit des Beginns und Endes der Sitzung.

```
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

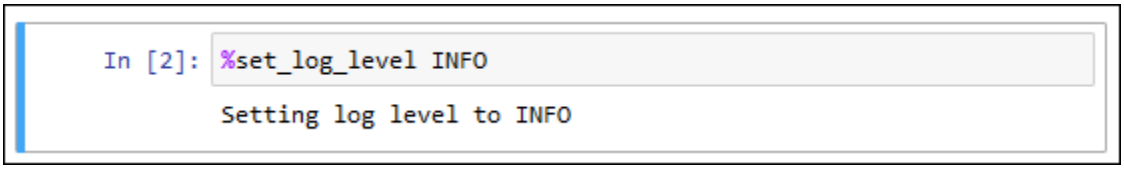
%session_id

Ruft die aktuelle Sitzungs-ID ab.



%set_log_level

Setzt den Logger auf die angegebene Protokollstufe oder setzt ihn zurück. Die möglichen Werte sind `DEBUG`, `ERROR`, `FATAL`, `INFO` und `WARN` oder `WARNING`. Werte müssen in Großbuchstaben geschrieben sein und dürfen nicht in einfache oder doppelte Anführungszeichen eingeschlossen sein.



%status

Beschreibt die aktuelle Sitzung. Die Ausgabe umfasst die Sitzungs-ID, den Sitzungsstatus, den Arbeitsgruppennamen, die PySpark-Engine-Version und die Startzeit der Sitzung. Dieser Magic-Befehl erfordert eine aktive Sitzung, um Sitzungsdetails abzurufen.

Folgende Werte sind für Status möglich:

ERSTELLEN – Die Sitzung wird gestartet, einschließlich des Erwerbs von Ressourcen.

ERSTELLT – Die Sitzung wurde gestartet.

UNTÄTIG – Die Sitzung kann eine Berechnung akzeptieren.

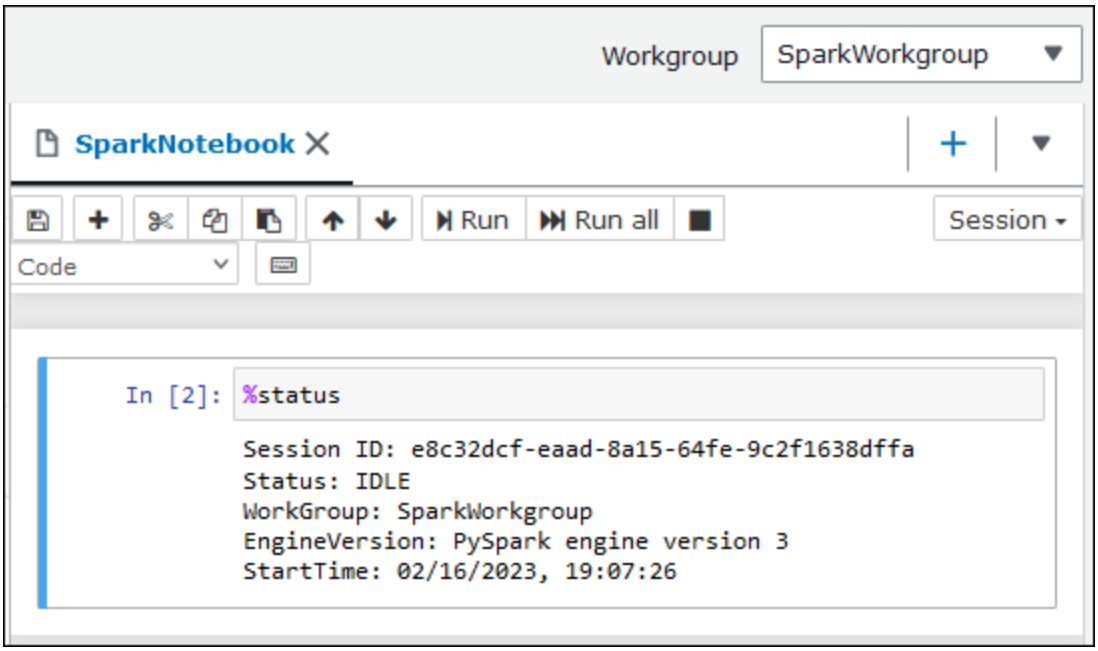
BESCHÄFTIGT – Die Sitzung verarbeitet gerade eine andere Aufgabe und kann keine Berechnung akzeptieren.

BEENDE – Der Service wird gerade heruntergefahren.

BEENDET – Die Sitzung und ihre Ressourcen werden nicht mehr ausgeführt.

HERABGESTUFT – Die Sitzung hat keine funktionierenden Koordinatoren.

FEHLGESCHLAGEN – Aufgrund eines Fehlers werden die Sitzung und ihre Ressourcen nicht mehr ausgeführt.



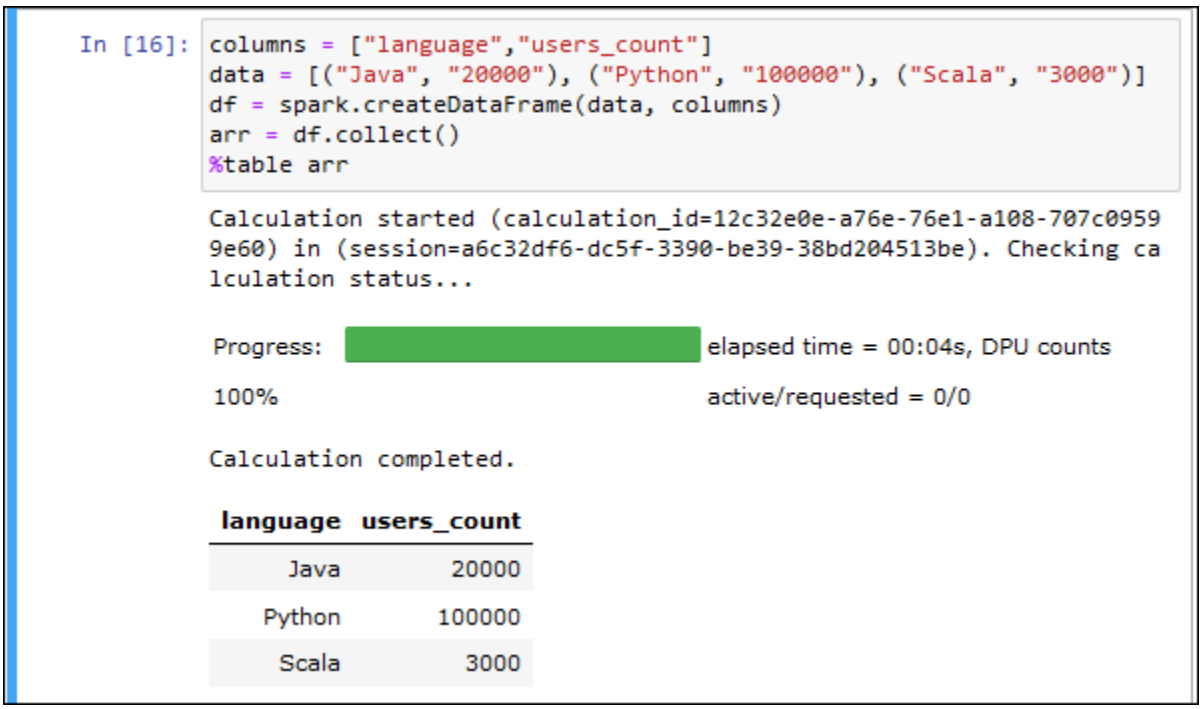
Magics zum Erstellen von Datendiagrammen

Die Linien-Magics in diesem Abschnitt spezialisieren sich auf das Rendern von Daten für bestimmte Datentypen oder in Verbindung mit Grafikbibliotheken.

%table

Sie können den `%table`-Magics-Befehl verwenden, um Dataframe-Daten im Tabellenformat anzuzeigen.

Im folgenden Beispiel wird ein Datenrahmen mit zwei Spalten und drei Datenzeilen erstellt und die Daten anschließend im Tabellenformat angezeigt.



%matplot

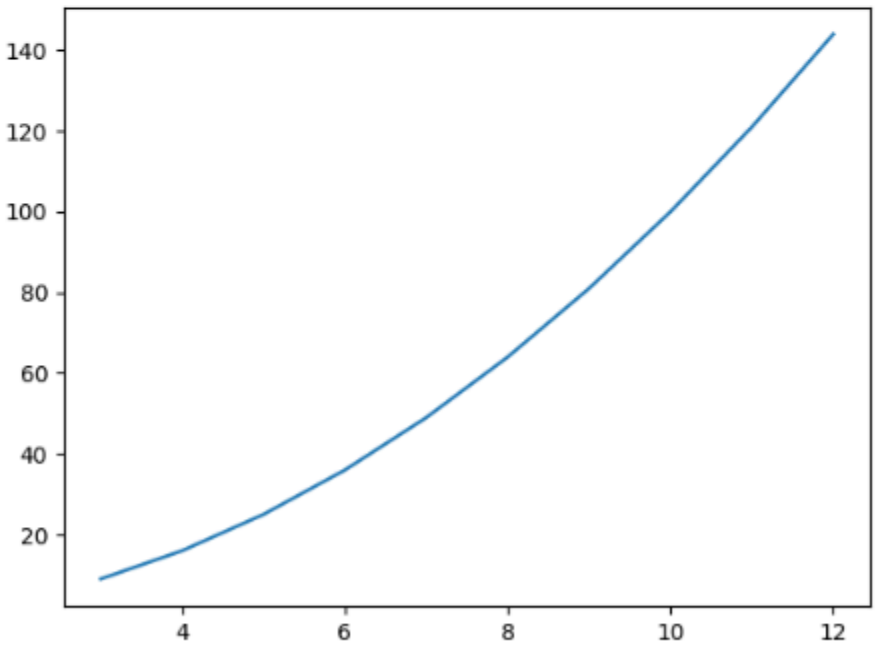
Matplotlib ist eine umfassende Bibliothek zur Erstellung statischer, animierter und interaktiver Visualisierungen in Python. Sie können den `%matplot`-Magics-Befehl verwenden, um ein Diagramm zu erstellen, nachdem Sie die Matplotlib-Bibliothek in eine Notebook-Zelle importiert haben.

Das folgende Beispiel importiert die Matplotlib-Bibliothek, erstellt einen Satz von X- und Y-Koordinaten und verwendet dann den Befehl use `%matplot` magic, um ein Diagramm der Punkte zu erstellen.

```
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>]
```

Die Bibliotheken matplotlib und seaborn zusammen verwenden

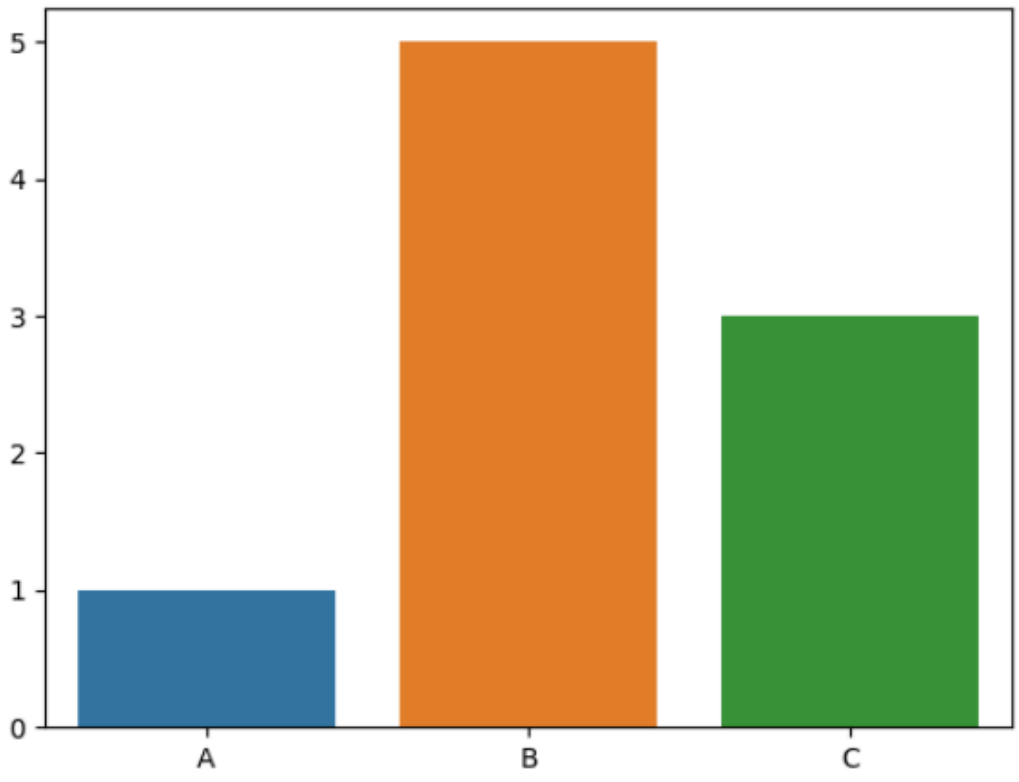
Seaborn ist eine Bibliothek zur Erstellung statistischer Grafiken in Python. Es baut auf Matplotlib auf und ist eng in die Datenstrukturen von Pandas (Python-Datenanalyse) integriert. Sie können auch den `%matplot`-Magics-Befehl verwenden, um Seaborn-Daten zu rendern.

Das folgende Beispiel verwendet sowohl die Bibliotheken matplotlib als auch seaborn, um ein einfaches Balkendiagramm zu erstellen.

```
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.
```

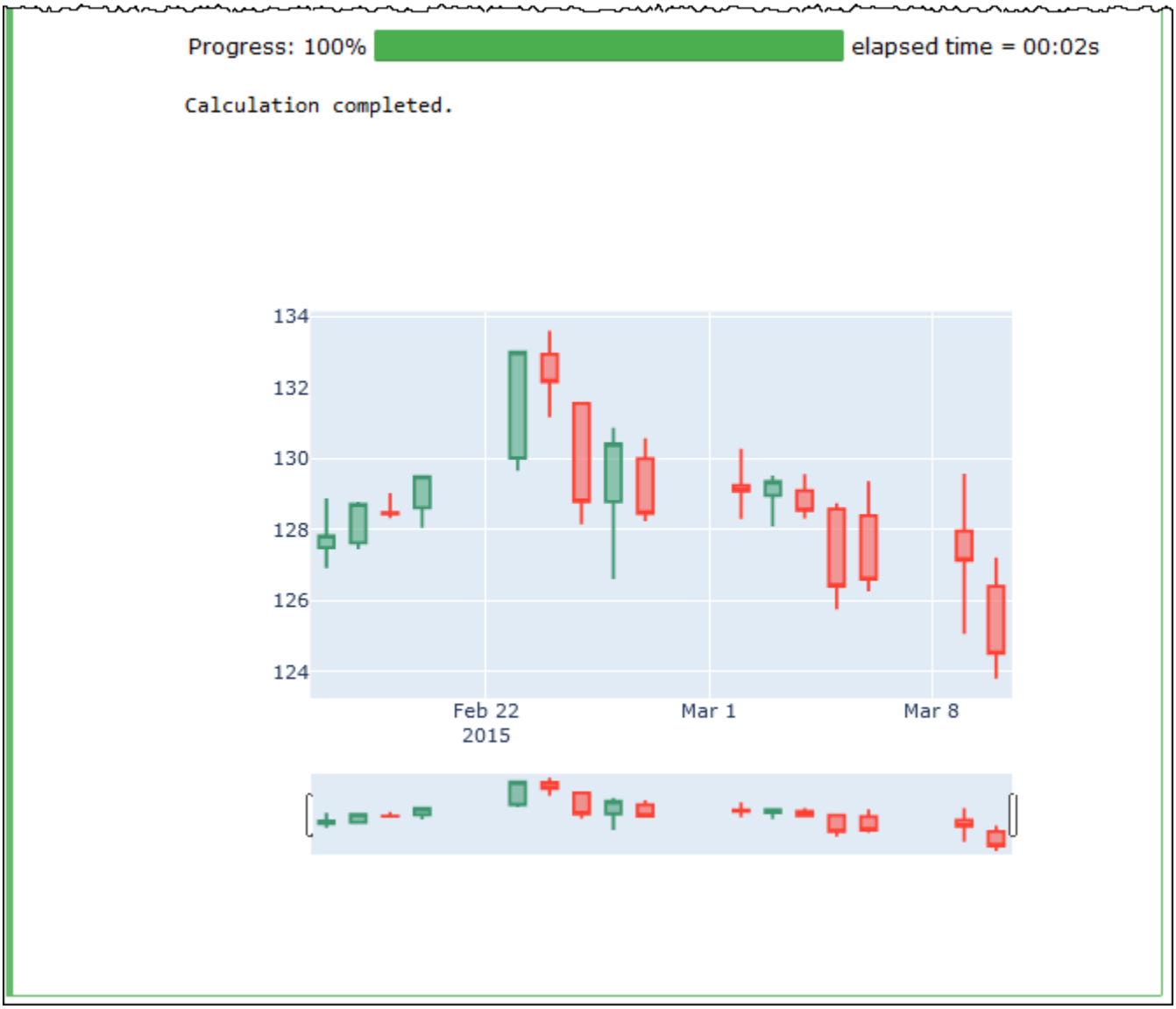


%plotly

Plotly ist eine Open-Source-Grafikbibliothek für Python, mit der Sie interaktive Grafiken erstellen können. Sie können auch den `%ploty`-Magics-Befehl verwenden, um Plotly-Daten zu rendern.

Im folgenden Beispiel werden die Bibliotheken StringIO, Plotly und Pandas für Aktienkursdaten verwendet, um ein Diagramm der Aktienaktivitäten im Februar und März 2015 zu erstellen.

```
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
```



Verwalten von Notebook-Dateien

Sie können den Notebook-Explorer nicht nur zum Erstellen und Öffnen von Notebooks verwenden, sondern auch zum Umbenennen, Löschen, Exportieren oder Importieren von Notebooks oder zum Anzeigen des Sitzungsverlaufs für ein bestimmtes Notebook.

So benennen Sie ein Notebook um

1. Beenden Sie alle aktiven Sitzungen für das Notebook, das Sie umbenennen möchten. Die aktiven Sitzungen des Notebooks müssen beendet werden, bevor Sie das Notebook umbenennen können.
2. Öffnen Sie den Notebook explorer (Notebook-Explorer).

3. Wählen Sie in der Liste Notebooks das Optionsfeld für das Notebook, das Sie umbenennen möchten.
4. Wählen Sie im Menü Actions (Aktionen) die Option Rename (Umbenennen).
5. Geben Sie bei der Aufforderung Rename notebook (Notebook umbenennen) den neuen Namen ein, und wählen Sie anschließend Save (Speichern) aus. Der neue Notebook-Name wird in der Notebook-Liste angezeigt.

So löschen Sie ein Notebook

1. Beenden Sie alle aktiven Sitzungen für das Notebook, das Sie löschen möchten. Die aktiven Sitzungen des Notebooks müssen beendet werden, bevor Sie das Notebook löschen können.
2. Öffnen Sie den Notebook explorer (Notebook-Explorer).
3. Wählen Sie in der Liste Notebooks das Optionsfeld für das Notebook, das Sie löschen möchten.
4. Wählen Sie im Menü Actions (Aktionen) die Option Delete (Löschen) aus.
5. Geben Sie bei der Eingabeaufforderung Delete notebook? (Notebook löschen?) den Namen des Notebooks ein und wählen Sie dann Delete (Löschen), um den Löschvorgang zu bestätigen. Der Name des Notebooks wird aus der Notebook-Liste entfernt.

So exportieren Sie ein Notebook

1. Öffnen Sie den Notebook explorer (Notebook-Explorer).
2. Wählen Sie in der Liste Notebooks das Optionsfeld für das Notebook, das Sie exportieren möchten.
3. Wählen Sie im Menü Actions (Aktionen) die Option Export file (Datei exportieren).

So importieren Sie ein Notebook

1. Öffnen Sie den Notebook explorer (Notebook-Explorer).
2. Wählen Sie Import file (Datei importieren).
3. Navigieren Sie auf Ihrem lokalen Computer zum Speicherort der Datei, die Sie importieren möchten, und wählen Sie dann Open (Öffnen) aus. Das importierte Notebook wird in der Notebook-Liste angezeigt.

So zeigen Sie den Sitzungsverlauf für ein Notebook an

1. Öffnen Sie den Notebook explorer (Notebook-Explorer).
2. Wählen Sie in der Liste Notebooks das Optionsfeld für das Notebook, dessen Sitzungsverlauf Sie anzeigen möchten.
3. Wählen Sie im Menü Actions (Aktionen) die Option Session history (Sitzungsverlauf) aus.
4. Wählen Sie auf der Registerkarte History (Verlauf) eine Session ID (Sitzungs-ID) aus, um Informationen über die Sitzung und ihre Berechnungen anzuzeigen.

Nicht-Hive-Tabellenformaten in Amazon Athena für Apache Spark verwenden

Wenn Sie in Athena für Spark mit Sessions und Notebooks arbeiten, können Sie neben Apache-Hive-Tabellen auch Linux-Foundation-Delta-Lake-, Apache-Hudi- und Apache-Iceberg-Tabellen verwenden.

Überlegungen und Einschränkungen

Wenn Sie andere Tabellenformate als Apache Hive mit Athena für Spark verwenden, sollten Sie die folgenden Punkte berücksichtigen:

- Zusätzlich zu Apache Hive wird nur ein Tabellenformat pro Notebook unterstützt. Um mehrere Tabellenformate in Athena für Spark zu verwenden, erstellen Sie für jedes Tabellenformat ein separates Notizbuch. Informationen zum Erstellen von Notebooks in Athena für Spark finden Sie unter Erstellen Ihres eigenen Notebooks.
- Die Tabellenformate Delta Lake, Hudi und Iceberg wurden auf Athena für Spark getestet, indem AWS Glue als Metastore verwendet wurde. Möglicherweise können Sie andere Metastores verwenden, aber eine solche Verwendung wird derzeit nicht unterstützt.
- Um die zusätzlichen Tabellenformate zu verwenden, überschreiben Sie die `spark_catalog`-Standardeigenschaft, wie in der Athena-Konsole und in dieser Dokumentation angegeben. Diese Nicht-Hive-Kataloge können zusätzlich zu ihren eigenen Tabellenformaten Hive-Tabellen lesen.

Tabellenversionen

Die folgende Tabelle zeigt die unterstützten Nicht-Hive-Tabellenversionen in Amazon Athena für Apache Spark.

Tabellenformat	Unterstützte Version
Apache Iceberg	1.2.1
Apache Hudi	0,13
Linux Foundation Delta Lake	2.0.2

In Athena für Spark werden diese `.jar`-Dateien im Tabellenformat und ihre Abhängigkeiten in den Klassenpfad für Spark-Treiber und -Ausführern geladen.

Themen

- Apache Iceberg
- Apache Hudi
- Linux Foundation Delta Lake

Apache Iceberg

Apache Iceberg ist ein offenes Tabellenformat für große Datensätze in Amazon Simple Storage Service (Amazon S3). Es bietet Ihnen schnelle Abfrageleistung bei großen Tabellen, atomare Commits, gleichzeitige Schreibvorgänge und eine SQL-kompatible Tabellenentwicklung.

Um Apache-Iceberg-Tabellen in Athena für Spark zu verwenden, konfigurieren Sie die folgenden Spark-Eigenschaften. Diese Eigenschaften werden standardmäßig in der Athena für Spark-Konsole für Sie konfiguriert, wenn Sie Apache-Iceberg als Tabellenformat wählen. Die Schritte finden Sie in Bearbeiten von Sitzungsdetails oder Erstellen Ihres eigenen Notebooks.

```
"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"
```

Das folgende Verfahren zeigt Ihnen, wie Sie eine Apache-Iceberg-Tabelle in einem Notebook von Athena für Spark verwenden. Führen Sie jeden Schritt in einer neuen Zelle im Notebook aus.

Wie Sie Apache-Iceberg-Tabellen in Amazon Athena für Apache Spark verwenden

1. Definieren Sie die Konstanten, die im Notebook verwendet werden sollen.

```
DB_NAME = "NEW_DB_NAME"
TABLE_NAME = "NEW_TABLE_NAME"
TABLE_S3_LOCATION = "s3://example_path"
```

2. Erstellen Sie einen Apache Spark DataFrame.

```
columns = ["language","users_count"]
data = [("Golang", 3000)]
df = spark.createDataFrame(data, columns)
```

3. Erstellen Sie eine Datenbank.

```
spark.sql("CREATE DATABASE {} LOCATION '{}'".format(DB_NAME, TABLE_S3_LOCATION))
```

4. Erstellen Sie eine leere Apache-Iceberg-Tabelle.

```
spark.sql("""
CREATE TABLE {}.{} (
language string,
users_count int
) USING ICEBERG
""".format(DB_NAME, TABLE_NAME))
```

5. Fügt eine Datenzeile in die Tabelle ein.

```
spark.sql("""INSERT INTO {}.{} VALUES ('Golang',
 3000)""".format(DB_NAME, TABLE_NAME))
```

6. Vergewissern Sie sich, dass Sie die neue Tabelle abfragen können.

```
spark.sql("SELECT * FROM {}.{}".format(DB_NAME, TABLE_NAME)).show()
```

Weitere Informationen und Beispiele für die Arbeit mit Spark- DataFrames und Iceberg-Tabellen finden Sie unter Spark-Abfragen in der Apache-Iceberg-Dokumentation.

Apache Hudi

Apache Hudi ist ein Open-Source-Datenmanagement-Framework, das die inkrementelle Datenverarbeitung vereinfacht. Aktionen zum Einfügen, Aktualisieren, Upsert und Löschen auf Datensatzebene werden mit höherer Präzision verarbeitet, wodurch der Overhead reduziert wird.

Um Apache-Hudi-Tabellen in Athena für Spark zu verwenden, konfigurieren Sie die folgenden Spark-Eigenschaften. Diese Eigenschaften werden standardmäßig in der Athena für Spark-Konsole für Sie konfiguriert, wenn Sie Apache-Hudi als Tabellenformat wählen. Die Schritte finden Sie in Bearbeiten von Sitzungsdetails oder Erstellen Ihres eigenen Notebooks.

```
"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"
```

Das folgende Verfahren zeigt Ihnen, wie Sie eine Apache-Hudi-Tabelle in einem Athena für Spark-Notebook verwenden. Führen Sie jeden Schritt in einer neuen Zelle im Notebook aus.

Wie Sie Apache-Hudi-Tabellen in Amazon Athena für Apache Spark verwenden

1. Definieren Sie die Konstanten, die im Notebook verwendet werden sollen.

   ```
   DB_NAME = "NEW_DB_NAME"
   TABLE_NAME = "NEW_TABLE_NAME"
   TABLE_S3_LOCATION = "s3://example_path"
   ```

2. Erstellen Sie einen Apache Spark DataFrame.

   ```
   columns = ["language","users_count"]
   data = [("Golang", 3000)]
   df = spark.createDataFrame(data, columns)
   ```

3. Erstellen Sie eine Datenbank.

   ```
   spark.sql("CREATE DATABASE {} LOCATION '{}'".format(DB_NAME, TABLE_S3_LOCATION))
   ```

4. Erstellen Sie eine leere Apache-Hudi-Tabelle.

   ```
   spark.sql("""
   CREATE TABLE {}.{} (
   language string,
   ```

```
users_count int
) USING HUDI
TBLPROPERTIES (
primaryKey = 'language',
type = 'mor'
);
""".format(DB_NAME, TABLE_NAME))
```

5. Fügt eine Datenzeile in die Tabelle ein.

```
spark.sql("""INSERT INTO {}.{} VALUES ('Golang',
 3000)""".format(DB_NAME,TABLE_NAME))
```

6. Vergewissern Sie sich, dass Sie die neue Tabelle abfragen können.

```
spark.sql("SELECT * FROM {}.{}".format(DB_NAME, TABLE_NAME)).show()
```

Linux Foundation Delta Lake

Linux Foundation Delta Lake ist ein Tabellenformat für Big-Data-Analytik. Sie können Athena für Spark verwenden, um in Amazon S3 gespeicherte Delta-Lake-Tabellen direkt zu lesen.

Um Apache-Delta-Lake-Tabellen in Athena für Spark zu verwenden, konfigurieren Sie die folgenden Spark-Eigenschaften. Diese Eigenschaften werden standardmäßig in der Athena für Spark-Konsole für Sie konfiguriert, wenn Sie Delta Lake als Tabellenformat wählen. Die Schritte finden Sie in Bearbeiten von Sitzungsdetails oder Erstellen Ihres eigenen Notebooks.

```
"spark.sql.catalog.spark_catalog" : "org.apache.spark.sql.delta.catalog.DeltaCatalog",
"spark.sql.extensions" : "io.delta.sql.DeltaSparkSessionExtension"
```

Das folgende Verfahren zeigt Ihnen, wie Sie eine Delta-Lake-Tabelle in einem Athena für Spark-Notebook verwenden. Führen Sie jeden Schritt in einer neuen Zelle im Notebook aus.

Wie Sie eine Delta-Lake-Tabelle in Athena für Spark verwenden

1. Definieren Sie die Konstanten, die im Notebook verwendet werden sollen.

```
DB_NAME = "NEW_DB_NAME"
TABLE_NAME = "NEW_TABLE_NAME"
TABLE_S3_LOCATION = "s3://example_path"
```

2. Erstellen Sie einen Apache Spark DataFrame.

```
columns = ["language","users_count"]
data = [("Golang", 3000)]
df = spark.createDataFrame(data, columns)
```

3. Erstellen Sie eine Datenbank.

```
spark.sql("CREATE DATABASE {} LOCATION '{}'".format(DB_NAME, TABLE_S3_LOCATION))
```

4. Erstellen Sie eine leere Delta-Lake-Tabelle.

```
spark.sql("""
CREATE TABLE {}.{} (
  language string,
  users_count int
) USING DELTA
""".format(DB_NAME, TABLE_NAME))
```

5. Fügt eine Datenzeile in die Tabelle ein.

```
spark.sql("""INSERT INTO {}.{} VALUES ('Golang',
 3000)""".format(DB_NAME, TABLE_NAME))
```

6. Vergewissern Sie sich, dass Sie die neue Tabelle abfragen können.

```
spark.sql("SELECT * FROM {}.{}".format(DB_NAME, TABLE_NAME)).show()
```

Unterstützung der Python-Bibliothek in Amazon Athena für Apache Spark

Auf dieser Seite werden die verwendete Terminologie und das befolgte Lebenszyklusmanagement für die Laufzeiten, Bibliotheken und Pakete beschrieben, die in Amazon Athena für Apache Spark verwendet werden.

Definitionen

- Amazon Athena for Apache Spark (Amazon Athena für Apache Spark) ist eine benutzerdefinierte Version von Open Source Apache Spark. Um die aktuelle Version zu sehen, führen Sie den Befehl `print(f'{spark.version}')` in einer Notebookzelle aus.
- Die Athena runtime (Athena-Laufzeit) ist die Umgebung, in der Ihr Code ausgeführt wird. Die Umgebung beinhaltet einen Python-Interpreter und PySpark-Bibliotheken.
- Eine external library or package (externe Bibliothek oder ein Paket) ist eine Java- oder Scala-JAR- oder Python-Bibliothek, die nicht Teil der Athena-Laufzeit ist, aber in Athena-für-Spark-Aufträge enthalten sein kann. Externe Pakete können von Amazon oder von Ihnen erstellt werden.
- Ein convenience package (Convenience-Paket) ist eine Sammlung externer Pakete, die von Athena ausgewählt wurden und die Sie in Ihre Spark-Anwendungen aufnehmen können.
- Ein bundle (Bundle) kombiniert die Athena-Laufzeit und ein Convenience-Paket.
- Eine user library (Benutzerbibliothek) ist eine externe Bibliothek oder ein externes Paket, das Sie Ihrem Athena-für-Spark-Auftrag explizit hinzufügen.
  - Eine Benutzerbibliothek ist ein externes Paket, das nicht Teil eines Convenience-Pakets ist. Eine Benutzerbibliothek muss geladen und installiert werden. Wenn Sie beispielsweise einige `.py`-Dateien schreiben, komprimieren Sie sie und fügen Sie die `.zip`-Datei dann zu Ihrer Anwendung hinzu.
- Eine Athena for Spark application (Athena-für-Spark-Anwendung) ist ein Auftrag oder eine Abfrage, die Sie an Athena für Spark senden.

Verwaltung des Lebenszyklus

Versionsverwaltung und Veraltung zur Laufzeit

Die Hauptkomponente in der Athena-Laufzeit ist der Python-Interpreter. Da Python eine sich entwickelnde Sprache ist, werden regelmäßig neue Versionen veröffentlicht und die Unterstützung für ältere Versionen entfernt. Athena rät davon ab, Programme mit veralteten Versionen des Python-Interpreters auszuführen, und empfiehlt dringend, nach Möglichkeit die neueste Athena-Laufzeitumgebung zu verwenden.

Der Zeitplan für die Veraltung der Athena-Laufzeit sieht wie folgt aus:

1. Nachdem Athena eine neue Laufzeit bereitgestellt hat, unterstützt Athena die vorherige Laufzeit weiterhin für 6 Monate. Während dieser Zeit wendet Athena Sicherheitspatches und Updates auf die vorherige Laufzeit an.
2. Nach 6 Monaten beendet Athena die Unterstützung für die vorherige Laufzeit. Athena wendet keine Sicherheitspatches und andere Updates mehr auf die vorherige Laufzeit an. Spark-Anwendungen, die die vorherige Laufzeit verwenden, haben keinen Anspruch mehr auf technischen Support.
3. Nach 12 Monaten können Sie Spark-Anwendungen in einer Arbeitsgruppe, die die vorherige Laufzeit verwendet, nicht mehr aktualisieren oder bearbeiten. Wir empfehlen Ihnen, Ihre Spark-Anwendungen vor Ablauf dieses Zeitraums zu aktualisieren. Nach Ablauf des Zeitraums können Sie vorhandene Notebooks weiterhin ausführen, aber alle Notebooks, die noch die vorherige Laufzeit verwenden, protokollieren eine entsprechende Warnung.
4. Nach 18 Monaten können Sie in der Arbeitsgruppe keine Aufträge mehr mit der vorherigen Laufzeit ausführen.

Versionsverwaltung und Veraltung von Convenience-Paketen

Der Inhalt von Convenience-Paketen ändert sich im Laufe der Zeit. Athena fügt diese Convenience-Pakete gelegentlich hinzu, entfernt oder aktualisiert sie.

Athena verwendet die folgenden Richtlinien für Convenience-Pakete:

- Convenience-Pakete verfügen über ein einfaches Versionsverwaltungsschema wie 1, 2, 3.
- Jede Version des Convenience-Pakets enthält spezifische Versionen externer Pakete. Nachdem Athena ein Convenience-Paket erstellt hat, ändern sich die externen Pakete des Convenience-Pakets und ihre entsprechenden Versionen nicht.
- Athena erstellt eine neue Convenience-Paketversion, wenn es ein neues externes Paket enthält, ein externes Paket entfernt oder die Version eines oder mehrerer externer Pakete aktualisiert.

Athena markiert ein Convenience-Paket als veraltet, wenn es die vom Paket verwendete Athena-Laufzeit als veraltet markiert. Athena kann Pakete früher als veraltet markieren, um die Anzahl der unterstützten Bundles zu begrenzen.

Der Zeitplan für die Verwerfung von Convenience-Paketen folgt dem Zeitplan für die Verwerfung der Athena-Laufzeit.

Liste der vorinstallierten Python-Bibliotheken

Zu den vorinstallierten Python-Bibliotheken gehören die folgenden.

```
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
```

Hinweise

- MLlib (Apache Spark Machine Learning Library) und das -`pyspark.ml`Paket werden nicht unterstützt.
- Derzeit `pip install` wird in Athena für Spark-Sitzungen nicht unterstützt.

Informationen zum Importieren von Python-Bibliotheken in Amazon Athena für Apache Spark finden Sie unter Importieren von Dateien und Python-Bibliotheken in Amazon Athena für Apache Spark.

Importieren von Dateien und Python-Bibliotheken in Amazon Athena für Apache Spark

Dieses Dokument enthält Beispiele für den Import von Dateien und Python-Bibliotheken in Amazon Athena für Apache Spark.

Überlegungen und Einschränkungen

- Python-Version – Derzeit verwendet Athena für Spark die Python-Version 3.9.16. Beachten Sie, dass Python-Pakete empfindlich auf Python-Nebenversionen reagieren.
- Architektur von Athena für Spark – Athena für Spark verwendet Amazon Linux 2 auf der ARM64-Architektur. Beachten Sie, dass einige Python-Bibliotheken keine Binärdateien für diese Architektur verteilen.
- Binary Shared Objects (SOs) – Da die SparkContext-AddPyFile-Methode keine binären gemeinsamen Objekte erkennt, kann sie in Athena für Spark nicht verwendet werden, um Python-Pakete hinzuzufügen, die von gemeinsamen Objekten abhängen.
- Resilient Distributed Datasets (RDDs) – RDDs werden nicht unterstützt.
- Dataframe.foreach – Die PySpark-Methode DataFrame.foreach wird nicht unterstützt.

Beispiele

In den Beispielen werden die folgenden Konventionen verwendet.

- Der Platzhalter Amazon-S3-Speicherort `s3://DOC-EXAMPLE-BUCKET`. Ersetzen Sie dies durch Ihren eigenen S3-Bucket-Speicherort.

- Alle Codeblöcke, die von einer Unix-Shell aus ausgeführt werden, werden als *directory_name* $ angezeigt. Beispielsweise werden der Befehl `ls` im Verzeichnis `/tmp` und seine Ausgabe wie folgt angezeigt:

  ```
  /tmp $ ls
  ```

  Ausgabe

  ```
  file1 file2
  ```

- Hinzufügen einer Datei zu einem Notebook nach dem Schreiben in das lokale temporäre Verzeichnis
- Importieren einer Datei aus Amazon S3
- Hinzufügen von Python-Dateien und Registrieren einer UDF
- Importieren einer Python-ZIP-Datei
- Importieren von zwei Versionen einer Python-Bibliothek als separate Module
- Importieren einer Python-ZIP-Datei aus PyPI
- Importieren einer Python-ZIP-Datei aus PyPI mit Abhängigkeiten

Importieren von Textdateien zur Verwendung in Berechnungen

Die Beispiele in diesem Abschnitt veranschaulichen, wie Sie Textarchive für Berechnungen in Ihren Notebooks in Athena für Spark importieren.

Hinzufügen einer Datei zu einem Notebook nach dem Schreiben in das lokale temporäre Verzeichnis

Das folgende Beispiel zeigt, wie eine Datei in ein lokales temporäres Verzeichnis geschrieben, einem Notebook hinzugefügt und getestet wird.

```
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()
```

Ausgabe

```
Calculation completed.
+---+---+-------+
| _1| _2|    col|
+---+---+-------+
|  1|  a|[aaaaa]|
|  2|  b|[bbbbb]|
+---+---+-------+
```

Importieren einer Datei aus Amazon S3

Das folgende Beispiel zeigt, wie Sie eine Datei aus Amazon S3 in ein Notebook importieren und testen.

Importieren einer Datei aus Amazon S3 in ein Notebook

1. Erstellen Sie eine Datei mit dem Namen `test.txt`, die eine einzelne Zeile mit dem Wert `5` enthält.
2. Fügen Sie die Datei einem Bucket in Amazon S3 hinzu. In diesem Beispiel wird der Speicherort `s3://DOC-EXAMPLE-BUCKET` verwendet.
3. Verwenden Sie den folgenden Code, um die Datei in Ihr Notebook zu importieren und die Datei zu testen.

```
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()
```

Ausgabe

```
Calculation completed.
+---+---+-------+
| _1| _2|    col|
+---+---+-------+
|  1|  a|[aaaaa]|
|  2|  b|[bbbbb]|
+---+---+-------+
```

Hinzufügen von Python-Dateien

Die Beispiele in diesem Abschnitt zeigen, wie Sie Ihren Spark-Notebooks in Athena-Python-Dateien und -Bibliotheken hinzufügen.

Hinzufügen von Python-Dateien und Registrieren einer UDF

Das folgende Beispiel zeigt, wie Sie Python-Dateien aus Amazon S3 zu Ihrem Notebook hinzufügen und eine UDF registrieren.

So fügen Sie Ihrem Notebook Python-Dateien hinzu und registrieren eine UDF

1. Verwenden Sie Ihren eigenen Amazon-S3-Speicherort und erstellen Sie die Datei `s3://DOC-EXAMPLE-BUCKET/file1.py` mit dem folgenden Inhalt:

```
def xyz(input):
    return 'xyz  - udf ' + str(input);
```

2. Erstellen Sie am selben S3-Speicherort die Datei `s3://DOC-EXAMPLE-BUCKET/file2.py` mit dem folgenden Inhalt:

```
from file1 import xyz
def uvw(input):
    return 'uvw -> ' + xyz(input);
```

3. Führen Sie in Ihrem Athena-for-Spark-Notebook die folgenden Befehle aus.

```
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)
```

Ausgabe

```
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... |
+---+---+--------------------+
```

Importieren einer Python-ZIP-Datei

Sie können die Python-Methoden `addPyFile` und `import` verwenden, um eine Python-ZIP-Datei in Ihr Notebook zu importieren.

> **Note**
>
> Die `.zip`-Dateien, die Sie in Athena Spark importieren, enthalten möglicherweise nur Python-Pakete. Beispielsweise wird das Einbeziehen von Paketen mit C-basierten Dateien nicht unterstützt.

So importieren Sie eine **`.zip`**-Python-Datei in Ihr Notebook

1. Erstellen Sie auf Ihrem lokalen Computer in einem Desktop-Verzeichnis wie z. B. `\tmp` ein Verzeichnis mit dem Namen `moduletest`.
2. Erstellen Sie im Verzeichnis `moduletest` eine Datei namens `hello.py` mit folgendem Inhalt:

```
def hi(input):
    return 'hi ' + str(input);
```

3. Fügen Sie im selben Verzeichnis eine leere Datei mit dem Namen `__init__.py` hinzu.

   Wenn Sie die Verzeichnisinhalte auflisten, sollten diese jetzt wie folgt aussehen.

```
/tmp $ ls moduletest
__init__.py       hello.py
```

4. Verwenden Sie den `zip`-Befehl, um die beiden Moduldateien in einer Datei mit dem Namen `moduletest.zip` zu platzieren.

```
moduletest $ zip -r9 ../moduletest.zip *
```

5. Laden Sie die `.zip`-Datei in Ihren Bucket in Amazon S3 hoch.
6. Verwenden Sie den folgenden Code, um die `.zip`-Python-Datei in Ihr Notebook zu importieren.

```
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()
```

Ausgabe

```
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|
+---+---+----+
```

Importieren von zwei Versionen einer Python-Bibliothek als separate Module

Die folgenden Codebeispiele zeigen, wie Sie zwei verschiedene Versionen einer Python-Bibliothek von einem Speicherort in Amazon S3 als zwei separate Module hinzufügen und importieren. Der Code fügt jeweils die Bibliotheksdatei aus S3 hinzu, importiert sie und gibt dann die Bibliotheksversion aus, um den Import zu überprüfen.

```
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__)
```

Ausgabe

```
3.15.0
```

```
import simplejson_v3_17_6
print(simplejson_v3_17_6.__version__)
```

Ausgabe

```
3.17.6
```

Importieren einer Python-ZIP-Datei aus PyPI

In diesem Beispiel wird der `pip`-Befehl verwendet, um eine Python-ZIP-Datei des Projekts bpabel/piglatin aus dem Python Package Index (PyPI) herunterzuladen.

So importieren Sie eine Python-ZIP-Datei aus PyPI

1. Verwenden Sie auf Ihrem lokalen Desktop die folgenden Befehle, um ein Verzeichnis mit dem Namen `testpiglatin` zu erstellen und eine virtuelle Umgebung zu erstellen.

   ```
   /tmp $ mkdir testpiglatin
   /tmp $ cd testpiglatin
   testpiglatin $ virtualenv .
   ```

   Ausgabe

   ```
   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. Erstellen Sie ein Unterverzeichnis mit dem Namen `unpacked`, um das Projekt zu speichern.

   ```
   testpiglatin $ mkdir unpacked
   ```

3. Verwenden Sie den `pip`-Befehl, um das Projekt in das `unpacked`-Verzeichnis zu installieren.

   ```
   testpiglatin $ bin/pip install -t $PWD/unpacked piglatin
   ```

   Ausgabe

   ```
   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. Überprüfen Sie den Inhalt des Verzeichnisses.

```
testpiglatin $ ls
```

Ausgabe

```
bin lib pyvenv.cfg unpacked
```

5. Wechseln Sie in das `unpacked`-Verzeichnis und zeigen Sie den Inhalt an.

```
testpiglatin $ cd unpacked
unpacked $ ls
```

Ausgabe

```
piglatin piglatin-1.0.6.dist-info
```

6. Verwenden Sie den `zip`-Befehl, um den Inhalt des Piglatin-Projekts in einer Datei mit dem Namen `library.zip` zu platzieren.

```
unpacked $ zip -r9 ../library.zip *
```

Ausgabe

```
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. (Optional) Verwenden Sie die folgenden Befehle, um den Import lokal zu testen.

a. Legen Sie den Python-Pfad auf den Speicherort der `library.zip`-Datei fest und starten Sie Python.

```
/home $ PYTHONPATH=/tmp/testpiglatin/library.zip
/home $ python3
```

Ausgabe

```
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. Importieren Sie die Bibliothek und führen Sie einen Testbefehl aus.

```
>>> import piglatin
>>> piglatin.translate('hello')
```

Ausgabe

```
'ello-hay'
```

8. Verwenden Sie Befehle wie die folgenden, um die `.zip`-Datei aus Amazon S3 hinzuzufügen, sie in Ihr Notebook in Athena zu importieren und zu testen.

```
sc.addPyFile('s3://user1-athena-output/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()
```

Ausgabe

```
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|
+---+-----+--------+
```

Importieren einer Python-ZIP-Datei aus PyPI mit Abhängigkeiten

In diesem Beispiel wird das md2gemini-Paket, das Text im Markdown in das Gemini-Textformat konvertiert, aus PyPI importiert. Das Paket weist folgende Abhängigkeiten auf:

```
cjkwrap
mistune
wcwidth
```

So importieren Sie eine Python-ZIP-Datei mit Abhängigkeiten

1. Verwenden Sie auf Ihrem lokalen Computer die folgenden Befehle, um ein Verzeichnis mit dem Namen `testmd2gemini` zu erstellen und eine virtuelle Umgebung zu erstellen.

   ```
   /tmp $ mkdir testmd2gemini
   /tmp $ cd testmd2gemini
   testmd2gemini$ virtualenv .
   ```

2. Erstellen Sie ein Unterverzeichnis mit dem Namen `unpacked`, um das Projekt zu speichern.

   ```
   testmd2gemini $ mkdir unpacked
   ```

3. Verwenden Sie den `pip`-Befehl, um das Projekt in das `unpacked`-Verzeichnis zu installieren.

   ```
   /testmd2gemini $ bin/pip install -t $PWD/unpacked md2gemini
   ```

   Ausgabe

   ```
   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. Wechseln Sie in das `unpacked`-Verzeichnis und überprüfen Sie den Inhalt.

```
testmd2gemini $ cd unpacked
unpacked $ ls -lah
```

Ausgabe

```
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. Verwenden Sie den `zip`-Befehl, um den Inhalt des md2gemini-Projekts in einer Datei mit dem Namen `md2gemini.zip` zu platzieren.

```
unpacked $ zip -r9 ../md2gemini *
```

Ausgabe

```
  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. (Optional) Verwenden Sie die folgenden Befehle, um zu testen, ob die Bibliothek auf Ihrem lokalen Computer funktioniert.

   a. Legen Sie den Python-Pfad auf den Speicherort der `md2gemini.zip`-Datei fest und starten Sie Python.

   ```
   /home $ PYTHONPATH=/tmp/testmd2gemini/md2gemini.zip
   /home python3
   ```

   b. Importieren Sie die Bibliothek und führen Sie einen Test durch.

   ```
   >>> from md2gemini import md2gemini
   >>> print(md2gemini('[abc](https://abc.def)'))
   ```

   Ausgabe

   ```
   https://abc.def abc
   ```

7. Verwenden Sie die folgenden Befehle, um die `.zip`-Datei aus Amazon S3 hinzuzufügen, diese in Ihr Notebook in Athena zu importieren und einen Nicht-UDF-Test durchzuführen.

   ```
   # (non udf test)
   sc.addPyFile('s3://DOC-EXAMPLE-BUCKET/md2gemini.zip')
   from md2gemini import md2gemini
   print(md2gemini('[abc](https://abc.def)'))
   ```

   Ausgabe

   ```
   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. Verwenden Sie die folgenden Befehle, um einen UDF-Test durchzuführen.

   ```
   # (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()
```

Ausgabe

```
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...|
+---+--------------------+--------------------+
```

Hinzufügen von JAR-Dateien und benutzerdefinierte Spark-Konfiguration

Wenn Sie eine Sitzung in Amazon Athena für Apache Spark erstellen oder bearbeiten, können Sie Spark-Eigenschaften verwenden, um `.jar`-Dateien, Pakete oder eine andere benutzerdefinierte Konfiguration für die Sitzung anzugeben. Um Ihre Spark-Eigenschaften anzugeben, können Sie die Athena-Konsole, die AWS CLI oder die Athena-API verwenden.

Verwenden der Athena-Konsole zur Angabe von Spark-Eigenschaften

In der Athena-Konsole können Sie Ihre Spark-Eigenschaften angeben, wenn Sie ein Notebook erstellen oder eine aktuelle Sitzung bearbeiten.

Um Eigenschaften im Dialogfeld Notebook erstellen oder Sitzungsdetails bearbeiten hinzuzufügen

1. Erweitern Sie Spark-Eigenschaften.

2. Um Ihre Eigenschaften hinzuzufügen, verwenden Sie die Option In Tabelle bearbeiten oder In JSON bearbeiten.

   - Wählen Sie für die Option In Tabelle bearbeiten die Option Eigenschaft hinzufügen aus, um eine Eigenschaft hinzuzufügen, oder wählen Sie Entfernen, um eine Eigenschaft zu entfernen. Verwenden Sie die Felder Schlüssel und Wert, um Eigenschaftsnamen und ihre Werte einzugeben.
     - Verwenden Sie die `spark.jars`-Eigenschaft, um eine benutzerdefinierte `.jar`-Datei hinzuzufügen.
     - Um eine Paketdatei anzugeben, verwenden Sie die `spark.jars.packages`-Eigenschaft.
   - Um Ihre Konfiguration direkt einzugeben und zu bearbeiten, wählen Sie die Option In JSON bearbeiten. Im JSON-Texteditor können Sie die folgenden Aufgaben durchführen:
     - Wählen Sie zum Kopieren der URL in die Zwischenablage Kopieren aus.
     - Wählen Sie Löschen, um den gesamten Text aus dem JSON-Editor zu entfernen.
     - Wählen Sie das Einstellungssymbol (Zahnrad), um den Zeilenumbruch zu konfigurieren, oder wählen Sie ein Farbdesign für den JSON-Editor.

Hinweise

- Sie können Eigenschaften in Athena für Spark festlegen, was dem direkten Einstellen von Spark-Eigenschaften für ein SparkConf-Objekt entspricht.
- Starten Sie alle Spark-Eigenschaften mit dem `spark.`-Präfix. Eigenschaften mit anderen Präfixen werden ignoriert.
- Nicht alle Spark-Eigenschaften sind für die benutzerdefinierte Konfiguration auf Athena verfügbar. Wenn Sie eine `StartSession`-Anfrage mit einer eingeschränkten Konfiguration einreichen, kann die Sitzung nicht gestartet werden.
  - Sie können das `spark.athena.`-Präfix nicht verwenden, da es reserviert ist.

Verwenden der AWS CLI oder Athena-API zur Bereitstellung einer benutzerdefinierten Konfiguration

Um die Sitzungskonfiguration über die AWS CLI oder Athena-API bereitzustellen, verwenden Sie die API-Aktion StartSession oder den CLI-Befehl start-session. Verwenden Sie in Ihrer `StartSession`-Anfrage das `SparkProperties`-Feld des EngineConfiguration-Objekts, um Ihre

Konfigurationsinformationen im JSON-Format zu übergeben. Dadurch wird eine Sitzung mit der von Ihnen angegebenen Konfiguration gestartet. Informationen zur Anforderungssyntax finden Sie unter StartSession in der Amazon-Athena-API-Referenz.

Behebung von Fehlern beim Sitzungsstart

Wenn während eines Sitzungsstarts ein benutzerdefinierter Konfigurationsfehler auftritt, zeigt die Konsole von Athena für Spark ein Fehlermeldungsbanner an. Um Fehler beim Sitzungsstart zu beheben, können Sie die Änderung des Sitzungsstatus oder die Protokollierungsinformationen überprüfen.

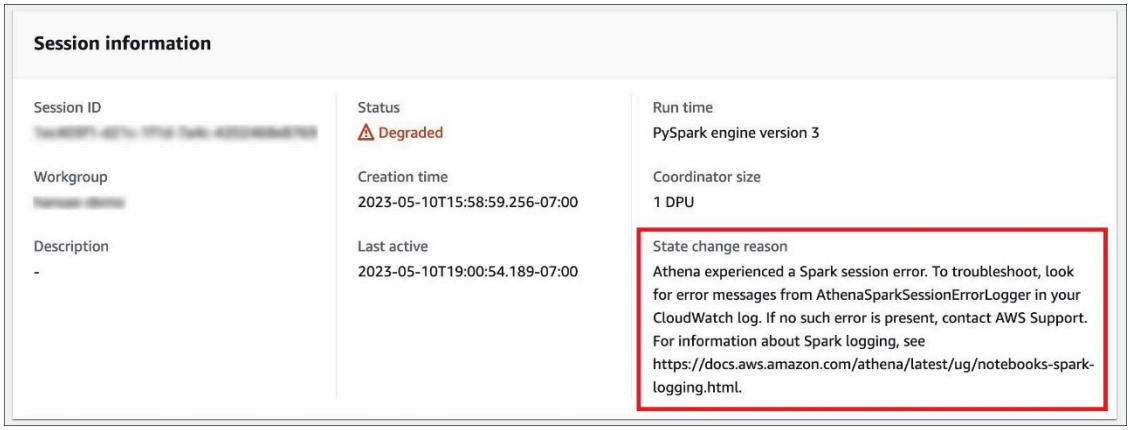
Informationen zur Änderung des Sitzungsstatus anzeigen

Einzelheiten zu einer Änderung des Sitzungsstatus erhalten Sie im Athena-Notebook-Editor oder in der Athena-API.

So zeigen Sie Informationen zum Sitzungsstatus in der Athena-Konsole an

1. Wählen Sie im Notebook-Editor im Menü Sitzung oben rechts die Option Details anzeigen aus.
2. Sehen Sie sich die Registerkarte Aktuelle Sitzung an. Im Abschnitt Sitzungsinformationen werden Informationen wie Sitzungs-ID, Arbeitsgruppe, Status und Grund für die Statusänderung angezeigt.

   Das folgende Beispiel für eine Bildschirmaufnahme zeigt Informationen im Abschnitt Grund der Statusänderung des Dialogfelds Sitzungsinformationen für einen Spark-Sitzungsfehler in Athena.



So zeigen Sie Sitzungsstatusinformationen mithilfe der Athena-API an

- In der Athena-API finden Sie Informationen zur Änderung des Sitzungsstatus im `StateChangeReason`-Feld des SessionStatus-Objekts.

> **Note**
>
> Nachdem Sie eine Sitzung manuell beendet haben oder wenn die Sitzung nach einem Leerlauf-Timeout beendet wird (die Standardeinstellung ist 20 Minuten), ändert sich der Wert von StateChangeReason in Session wurde per Anforderung beendet.

Verwenden der Protokollierung zur Behebung von Sitzungsstartfehlern

Benutzerdefinierte Konfigurationsfehler, die während eines Sitzungsstarts auftreten, werden von Amazon CloudWatch protokolliert. Suchen Sie in Ihren CloudWatch-Protokollen nach Fehlermeldungen von `AthenaSparkSessionErrorLogger`, um einen fehlgeschlagenen Sitzungsstart zu beheben.

Weitere Informationen zur Spark-Protokollierung finden Sie unter Protokollierung von Spark-Anwendungsereignissen in Athena.

Weitere Informationen zu Problembehandlungssitzungen in Athena für Spark finden Sie unter Fehlerbehebung bei Sitzungen.

Unterstützte Daten- und Speicherformate

Die folgende Tabelle zeigt Formate, die nativ in Athena für Apache Spark unterstützt werden.

Data format (Datenformat)	Read (Lesen)	Write (Schreiben)	Write compression (Schreibkomprimierung)
parquet	Ja	Ja	none, uncompressed, snappy, gzip
orc	Ja	Ja	none, snappy, zlib, lzo

Data format (Datenformat)	Read (Lesen)	Write (Schreiben)	Write compression (Schreibkomprimierung)
json	Ja	Ja	bzip2, gzip, deflate
csv	Ja	Ja	bzip2, gzip, deflate
text	Ja	Ja	none, bzip2, gzip, deflate
Binärdatei	Ja	–	–

Überwachen von Apache-Spark-Berechnungen mit CloudWatch-Metriken

Athena veröffentlicht berechnungsbezogene Metriken auf Amazon CloudWatch, wenn die Publish CloudWatch metrics-Option für Ihre Spark-fähige Arbeitsgruppe ausgewählt ist. Sie können in der CloudWatch-Konsole benutzerdefinierte Dashboards erstellen, Warnungen und Auslöser für Metriken festlegen.

Athena veröffentlicht die folgende Metrik in der CloudWatch-Konsole unter dem `AmazonAthenaForApacheSpark`-Namespace:

- `DPUCount` – Anzahl der DPUs, die während der Sitzung zur Ausführung der Berechnungen verbraucht wurden.

Diese Metrik besitzt die folgenden Dimensionen:

- `SessionId` – Die ID der Sitzung, in der die Berechnungen übermittelt werden.
- `WorkGroup` – Name der Arbeitsgruppe.

So zeigen Sie Metriken für Spark-fähige Arbeitsgruppen in der Amazon CloudWatch-Konsole an

1. Öffnen Sie die CloudWatch-Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im Navigationsbereich Metrics (Metriken) All metrics (Alle Metriken) aus.

3. Wählen Sie den AmazonAthenaForApacheSpark-Namespace aus.

So zeigen Sie Metriken im CLI-Dashboard an

- Führen Sie eine der folgenden Aktionen aus:
  - Um die Metriken für Athena-Spark-fähige Arbeitsgruppen aufzulisten, öffnen Sie eine Eingabeaufforderung und verwenden Sie den folgenden Befehl:

```
aws cloudwatch list-metrics --namespace "AmazonAthenaForApacheSpark"
```

  - Verwenden Sie den folgenden Befehl, um alle verfügbaren Metriken aufzulisten:

```
aws cloudwatch list-metrics
```

Liste der CloudWatch-Metriken und -Dimensionen für Apache Spark-Berechnungen in Athena

Wenn Sie CloudWatch-Metriken in Ihrer Spark-fähigen Athena-Arbeitsgruppe aktiviert haben, sendet Athena die folgende Metrik pro Arbeitsgruppe an CloudWatch. Die Metrik verwendet den `AmazonAthenaForApacheSpark`-Namespace.

Metrikname	Beschreibung
DPUCount	Anzahl der DPUs (Datenverarbeitungseinheiten), die während der Sitzung zur Ausführung der Berechnungen verbraucht werden. Bei einer DPU handelt es sich um ein relatives Maß der Rechenleistung, die aus 4 vCPUs Rechenkap azität und 16 GB Arbeitsspeicher besteht.

Diese Metrik besitzt die folgenden Dimensionen.

Dimension	Beschreibung
SitzungsID	Die ID der Sitzung, in der die Berechnungen übermittelt werden.
WorkGroup	Der Name der Arbeitsgruppe.

Durch die Aktivierung des Antragstellers werden Amazon-S3-Buckets in Athena für Spark bezahlt

Wenn ein Amazon-S3-Bucket mit Zahlung durch den Anforderer konfiguriert ist, werden die mit der Abfrage verbundenen Datenzugriffs- und Datenübertragungsgebühren dem Konto des Benutzers, der die Abfrage ausführt, belastet. Weitere Informationen finden Sie unter Verwendung von Anforderer zahlt Buckets für Speicherübertragungen und -nutzung im Amazon-S3-Benutzerhandbuch.

In Athena für Spark sind kostenpflichtige Buckets pro Sitzung aktiviert, nicht pro Arbeitsgruppe. Auf einer höheren Ebene umfasst die Aktivierung von Anforderer zahlt Buckets die folgenden Schritte:

1. Aktivieren Sie in der Amazon-S3-Konsole, dass der Antragsteller für die Eigenschaften des Buckets bezahlt, und fügen Sie eine Bucket-Richtlinie hinzu, um den Zugriff festzulegen.
2. Erstellen Sie in der IAM-Konsole eine IAM-Richtlinie, die den Zugriff auf den Bucket ermöglicht, und fügen Sie die Richtlinie dann der IAM-Rolle hinzu, die für den Zugriff auf den Bucket mit Zahlung durch den Anforderer verwendet wird.
3. Fügen Sie in Athena für Spark eine Sitzungseigenschaft hinzu, um das Feature Anforderer zahlt Buckets zu aktivieren.

1. Aktivieren Sie Zahlungen durch den Anforderer für einen Amazon-S3-Bucket und fügen Sie eine Bucket-Richtlinie hinzu

So aktivieren Sie die Zahlung durch den Anforderer für einen Amazon-S3-Bucket

1. Öffnen Sie die Amazon S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie in der Liste Buckets den Link des Buckets aus, für den Sie die Zahlung durch den Anforderer aktivieren möchten.
3. Wählen Sie auf der Seite für Ihren Bucket die Registerkarte Eigenschaften aus.
4. Scrollen Sie nach unten zum Abschnitt Anforderer zahlt und wählen Sie anschließend Bearbeiten aus.
5. Wählen Sie auf der Seite Anforderer zahlt die Option Aktivieren und anschließend Änderungen speichern aus.
6. Wählen Sie die Registerkarte Permissions (Berechtigungen).
7. Wählen Sie im Abschnitt Bucket-Richtlinie die Option Bearbeiten aus.

8. Wenden Sie auf der Seite Bucket-Richtlinie bearbeiten die gewünschte Bucket-Richtlinie auf den Quell-Bucket an. Die folgende Beispielrichtlinie gewährt Zugriff auf alle AWS-Prinzipale (`"AWS": "*"`), Ihr Zugriff kann jedoch detaillierter sein. Beispielsweise möchten Sie möglicherweise nur eine bestimmte IAM-Rolle in einem anderen Konto angeben.

```
{
    "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. Erstellen Sie eine IAM-Richtlinie und fügen Sie ihr die IAM-Rolle an

Als nächstes erstellen Sie eine IAM-Richtlinie, die Zugriff auf den Bucket gewährt. Dann fügen Sie die Richtlinie der Rolle zu, die für den Zugriff auf den Anforderer-zahlt-Bucket verwendet werden soll.

Wie Sie eine IAM-Richtlinie für den Anforderer-zahlt-Bucket erstellen und die Richtlinie einer Rolle hinzufügen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich der IAM-Konsole die Option Richtlinien.
3. Wählen Sie Create Policy (Richtlinie erstellen) aus.
4. Wählen Sie JSON.
5. Fügen Sie im Bucket-Richtlinieneditor eine Richtlinie, wie die folgende hinzu:

```
{
    "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. Wählen Sie Next (Weiter).
7. Geben Sie auf der Seite Überprüfen und erstellen einen Namen und eine optionale Beschreibung für die Richtlinie ein und wählen Sie anschließend Richtlinie erstellen aus.
8. Wählen Sie im Navigationsbereich Roles (Rollen) aus.
9. Suchen Sie auf der Seite Rollen die Rolle, die Sie verwenden möchten, und wählen Sie dann den Link mit dem Rollennamen.
10. Unter Berechtigungsrichtlinien im Abschnitt Berechtigungen hinzufügen wählen Sie Richtlinien anfügen aus.
11. Aktivieren Sie im Abschnitt Andere Berechtigungsrichtlinien das Kontrollkästchen für die Richtlinie, die Sie erstellt haben, und wählen Sie dann Berechtigungen hinzufügen aus.

3. Fügen Sie eine Sitzungseigenschaft von Athena für Spark hinzu

Nachdem Sie den Amazon-S3-Bucket und die zugehörigen Berechtigungen für Zahlungen durch den Antragsteller konfiguriert haben, können Sie das Feature in einer Athena-für-Spark-Sitzung aktivieren.

So aktivieren Sie Anforderer-zahlt-Buckets in einer Athena-für-Spark-Sitzung

1. Wählen Sie im Notebook-Editor aus dem Menü Session (Sitzung) oben rechts die Option Edit session (Sitzung bearbeiten) aus.

2. Erweitern Sie Spark-Eigenschaften.
3. Wählen Sie In JSON bearbeiten aus.
4. Geben Sie im JSON-Texteditor Folgendes ein:

```
{
  "spark.hadoop.fs.s3.useRequesterPaysHeader":"true"
}
```

5. Wählen Sie Save (Speichern).

Apache-Spark-Verschlüsselung aktivieren

Sie können die Apache-Spark-Verschlüsselung in Athena aktivieren. Dadurch werden Daten während der Übertragung zwischen Spark-Knoten, sowie Daten im Ruhezustand, die lokal von Spark gespeichert werden, verschlüsselt. Um die Sicherheit dieser Daten zu erhöhen, verwendet Athena die folgende Verschlüsselungskonfiguration:

```
spark.io.encryption.keySizeBits="256"
spark.io.encryption.keygen.algorithm="HmacSHA384"
```

Um die Spark-Verschlüsselung zu aktivieren, können Sie die Athena-Konsole, die AWS CLI oder die Athena-API verwenden.

So aktivieren Sie die Verschlüsselung über die Athena-Konsole

Um ein neues Notebook zu erstellen, für das die Spark-Verschlüsselung aktiviert ist

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.
3. Führen Sie eine der folgenden Aktionen aus:

   - Wählen Sie im Notebook explorer (Notebook-Explorer) die Option Create notebook (Notebook erstellen) aus.
   - Wählen Sie im Notebook explorer (Notebook-Editor) die Option Create notebook (Notebook erstellen) oder klicken Sie auf das Plussymbol (+), um ein Notebook hinzuzufügen.
4. Geben Sie unter Notebookname einen Namen für das Notebook ein.

5. Erweitern Sie die Option Spark-Eigenschaften.
6. Wählen Sie Spark-Verschlüsselung aktivieren aus.
7. Wählen Sie Erstellen aus.

Die von Ihnen erstellte Notebook-Sitzung ist verschlüsselt. Verwenden Sie das neue Notebook wie gewohnt. Wenn Sie später neue Sitzungen starten, die das Notebook verwenden, werden die neuen Sitzungen ebenfalls verschlüsselt.

Sie können auch die Athena-Konsole verwenden, um die Spark-Verschlüsselung für ein vorhandenes Notebook zu aktivieren.

Um die Verschlüsselung für ein vorhandenes Notebook zu aktivieren

1. Öffnen Sie eine neue Sitzung für ein zuvor erstelltes Notebook.
2. Wählen Sie im Notebook-Editor aus dem Menü Session (Sitzung) oben rechts die Option Edit session (Sitzung bearbeiten) aus.
3. Erweitern Sie im Dialogfeld Sitzungsdetails bearbeiten die Spark-Eigenschaften.
4. Wählen Sie Spark-Verschlüsselung aktivieren aus.
5. Wählen Sie Save (Speichern).

Die Konsole startet eine neue Sitzung, für die Verschlüsselung aktiviert ist. In späteren Sitzungen, die Sie für dieses Notebook erstellen, ist auch die Verschlüsselung aktiviert.

Spark-Verschlüsselung mit der AWS CLI aktivieren

Sie können die AWS CLI verwenden, um die Verschlüsselung zu aktivieren, wenn Sie eine Sitzung starten, indem Sie die entsprechenden Spark-Eigenschaften angeben.

Um die Spark-Verschlüsselung mit der AWS CLI zu aktivieren

1. Verwenden Sie einen Befehl wie den folgenden, um ein JSON-Objekt für die Engine-Konfiguration zu erstellen, das die Spark-Verschlüsselungseigenschaften spezifiziert.

```
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. Verwenden Sie in der AWS CLI den `athena start-session`-Befehl und übergeben Sie das JSON-Objekt, das Sie erstellt haben, an das `--engine-configuration`-Argument, wie im folgenden Beispiel:

```
aws athena start-session \
   --region "region" \
   --work-group "your-work-group" \
   --engine-configuration "$ENGINE_CONFIGURATION_JSON"
```

So aktivieren Sie die Spark-Verschlüsselung über die Athena-API

Um die Spark-Verschlüsselung mit der Athena-API zu aktivieren, verwenden Sie die Aktion StartSession und ihren `SparkProperties`-Parameter EngineConfiguration, um die Verschlüsselungskonfiguration in Ihrer `StartSession`-Anforderung anzugeben.

Konfigurieren des kontoübergreifenden AWS Glue-Zugriffs in Athena für Spark

In diesem Thema wird gezeigt, wie das Verbraucherkonto *666666666666* und das Besitzerkonto *999999999999* für den kontoübergreifenden AWS Glue-Zugriff konfiguriert werden können. Wenn die Konten konfiguriert sind, kann das Verbraucherkonto Abfragen von Athena für Spark in den AWS Glue-Datenbanken und Tabellen des Besitzers ausführen.

1. Stellen Sie in AWS Glue Zugriff auf Verbraucherrollen bereit

In AWS Glue erstellt der Eigentümer eine Richtlinie, die den Rollen des Verbrauchers Zugriff auf den AWS Glue-Datenkatalog des Besitzers gewährt.

Wie Sie eine AWS Glue-Richtlinie hinzufügen, die einer Verbraucherrolle Zugriff auf den Datenkatalog des Besitzers gewährt

1. Melden Sie sich mit dem Konto des Katalogbesitzers bei der AWS Management Console an.
2. Öffnen Sie die AWS Glue-Konsole unter https://console.aws.amazon.com/glue/.
3. Erweitern Sie im Navigationsbereich den Eintrag Data Catalog und wählen Sie dann Katalogeinstellungen aus.
4. Fügen Sie auf der Seite mit den Datenkatalogeinstellungen im Abschnitt Berechtigungen eine Richtlinie wie die folgende hinzu. Diese Richtlinie bietet Rollen für das Verbraucherkonto *666666666666*, Zugriff auf den Datenkatalog im Besitzerkonto *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. Konfigurieren Sie das Verbraucherkonto für den Zugriff

Erstellen Sie im Verbraucherkonto eine Richtlinie, die den Zugriff auf die Datenbanken und Tabellen des AWS Glue Data Catalog-Besitzers ermöglicht, und weisen Sie die Richtlinie einer Rolle zu. Im folgenden Beispiel wird das Verbraucherkonto *666666666666* verwendet.

Wie Sie eine AWS Glue-Richtlinie für den Zugriff auf die AWS Glue Data Catalog des Besitzers zu erstellen

1. Melden Sie sich mit dem Verbraucherkonto bei der AWS Management Console an.
2. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
3. Erweitern Sie im Navigationsbereich die Option Zugriffsverwaltung, und wählen Sie dann Richtlinien aus.
4. Wählen Sie Create Policy (Richtlinie erstellen) aus.
5. Wählen Sie auf der Seite Berechtigungen angeben die Option JSON aus.
6. Geben Sie im Richtlinien-Editor eine JSON-Anweisung wie die folgende ein, die AWS Glue-Aktionen im Datenkatalog des Eigentümerkontos ermöglicht.

```
{
    "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. Wählen Sie Next (Weiter).
8. Geben Sie auf der Seite Richtlinie prüfen im Feld Name einen Namen für die Richtlinie ein.
9. Wählen Sie Create Policy (Richtlinie erstellen) aus.

Als Nächstes verwenden Sie die IAM-Konsole im Verbraucherkonto, um die Richtlinie, die Sie gerade erstellt haben, der IAM-Rolle oder den Rollen zuzuordnen, die das Verbraucherkonto für den Zugriff auf den Datenkatalog des Besitzers verwendet.

Wie Sie die AWS Glue-Richtlinie den Rollen im Verbraucherkonto zuordnen

1. Wählen Sie im Navigationsbereich der IAM-Konsole Rollen aus.

2. Suchen Sie auf der Seite Rollen nach der Rolle, der Sie die Richtlinie zuordnen möchten.
3. Wählen Sie Berechtigungen hinzufügen aus und wählen Sie dann Richtlinien direkt anhängen aus.
4. Suchen Sie die Richtlinie, die Sie gerade erstellt haben.
5. Aktivieren Sie das Kontrollkästchen für die Richtlinie und wählen Sie dann Berechtigungen hinzufügen aus.
6. Wiederholen Sie die Schritte, um die Richtlinie zu anderen Rollen hinzuzufügen, die Sie verwenden möchten.

3. Konfigurieren Sie eine Sitzung und erstellen Sie eine Abfrage

Erstellen Sie in Athena Spark im Anfordererkonto mit der angegebenen Rolle eine Sitzung, um den Zugriff zu testen, indem Sie ein Notebook erstellen oder eine aktuelle Sitzung bearbeiten. Wenn Sie die Sitzungseigenschaften konfigurieren, geben Sie eine der folgenden an:

- Der Glue-Katalog-Separator – Bei diesem Ansatz schließen Sie die Konto-ID des Besitzers in Ihre Abfragen ein. Verwenden Sie diese Methode, wenn Sie die Sitzung verwenden möchten, um Datenkataloge von verschiedenen Besitzern abzufragen.
- Die Glue-Katalog-ID – Bei diesem Ansatz fragen Sie die Datenbank direkt ab. Diese Methode ist praktischer, wenn Sie die Sitzung verwenden möchten, um nur den Datenkatalog eines einzelnen Besitzers abzufragen.

Den Ansatz zur Trennung von AWS Glue-Katalogen verwenden

Wenn Sie die Sitzungseigenschaften bearbeiten, fügen Sie Folgendes hinzu:

```
{
    "spark.hadoop.aws.glue.catalog.separator": "/"
}
```

Wenn Sie eine Abfrage in einer Zelle ausführen, verwenden Sie eine Syntax wie im folgenden Beispiel. Beachten Sie, dass in der FROM-Klausel die Katalog-ID und das Trennzeichen vor dem Datenbanknamen angegeben werden müssen.

```
df = spark.sql('SELECT requestip, uri, method, status FROM `999999999999/
mydatabase`.cloudfront_logs LIMIT 5')
```

```
df.show()
```

Den AWS Glue-Katalog-ID-Ansatz verwenden

Wenn Sie die Sitzungseigenschaften bearbeiten, fügen Sie folgende Eigenschaft hinzu. Ersetzen Sie *999999999999* durch die Konto-ID des Besitzers.

```
{
    "spark.hadoop.hive.metastore.glue.catalogid": "999999999999"
}
```

Wenn Sie eine Abfrage in einer Zelle ausführen, verwenden Sie eine Syntax wie die folgende. Beachten Sie, dass in der FROM-Klausel die Katalog-ID und das Trennzeichen vor dem Datenbanknamen nicht angegeben werden müssen.

```
df = spark.sql('SELECT * FROM mydatabase.cloudfront_logs LIMIT 10')
df.show()
```

Weitere Informationen finden Sie unter:

Kontoübergreifender Zugriff auf AWS Glue -Datenkataloge

Verwalten von kontenübergreifenden Berechtigungen mithilfe von AWS Glue und Lake Formation im AWS Lake Formation-Entwicklerhandbuch.

Konfigurieren Sie den kontoübergreifenden Zugriff auf eine gemeinsame AWS Glue Data Catalog mit Amazon Athena in der Anleitung für Muster in der AWS Prescriptive Guidance.

Servicekontingente für Amazon Athena für Apache Spark

Servicekontingente, auch als Limits bezeichnet, sind die maximale Anzahl von Service-Ressourcen oder Vorgängen, die Ihr AWS-Konto nutzen kann. Weitere Informationen über Servicekontingente bei anderen AWS-Services, die Sie mit Amazon Athena für Apache Spark nutzen können, finden Sie unter AWS-Servicekontingente im Allgemeine Amazon Web Services-Referenz.

> **Note**
>
> Neue AWS-Konten haben möglicherweise niedrigere anfängliche Kontingente, die sich im Laufe der Zeit erhöhen können. Amazon Athena für Apache Spark überwacht ständig die

Kontonutzung in jeder AWS-Region und erhöht dann automatisch die Kontingente, basierend auf Ihrer Nutzung. Wenn Ihre Anforderungen die angegebenen Grenzwerte überschreiten, wenden Sie sich an den Kundensupport.

In der folgenden Tabelle sind die Servicekontingente für Amazon Athena für Apache-Spark aufgeführt.

Name	Standard	Anpassbar	Beschreibung
Parallelität von Apache Spark DPU	160	Nein	Die maximale Anzahl von Datenverarbeitungseinheiten (DPUs), die Sie gleichzeitig für Apache-Spark-Berechnungen für ein einzelnes Konto in der derzeitigen AWS-Region verwenden können. Bei einer DPU handelt es sich um ein relatives Maß der Rechenleistung, die aus 4 vCPUs Rechenkapazität und 16 GB Arbeitsspeicher besteht.
Sitzungs-Parallelität von Apache Spark DPU	60	Nein	Die maximale Anzahl von DPUs, die Sie gleichzeitig für Apache-Spark-Berechnungen verwenden können.

Athena-Notebooks-APIs

Die folgende Liste enthält Referenzlinks zu den Athena-Notebook-API-Aktionen. Datenstrukturen und andere Athena-API-Aktionen finden Sie in der Amazon-Athena-API-Referenz.

- CreateNotebook
- CreatePresignedNotebookUrl
- DeleteNotebook
- ExportNotebook
- GetCalculationExecution
- GetCalculationExecutionCode
- GetCalculationExecutionStatus

- GetNotebookMetadata
- GetSession
- GetSessionStatus
- ImportNotebook
- ListApplicationDPUSizes
- ListCalculationExecutions
- ListExecutors
- ListNotebookMetadata
- ListNotebookSessions
- ListSessions
- StartCalculationExecution
- StartSession
- StopCalculationExecution
- TerminateSession
- UpdateNotebook
- UpdateNotebookMetadata

Bekannte Probleme in Athena für Spark

Diese Seite dokumentiert einige der bekannten Probleme in Athena für Apache Spark.

Unzulässige Argumentausnahme beim Erstellen einer Tabelle

Obwohl Spark nicht zulässt, dass Datenbanken mit einer leeren Standorteigenschaft erstellt werden, können Datenbanken in AWS Glue eine leere `LOCATION`-Eigenschaft haben, wenn sie außerhalb von Spark erstellt werden.

Wenn Sie eine Tabelle erstellen und eine AWS Glue-Datenbank mit einem leeren `LOCATION`-Feld angeben, kann eine Ausnahme wie die folgende auftreten: IllegalArgumentException: Kann keinen Pfad aus einer leeren Zeichenfolge erstellen.

Der folgende Befehl löst beispielsweise eine Ausnahme aus, wenn die Standarddatenbank in AWS Glue ein leeres `LOCATION`-Feld enthält:

```
spark.sql("create table testTable (firstName STRING)")
```

Lösungsvorschlag A – Verwenden Sie AWS Glue, um der Datenbank, die Sie verwenden, einen Speicherort hinzuzufügen.

So fügen Sie einen Speicherort zu einer AWS Glue-Datenbank hinzu

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die AWS Glue-Konsole unter https://console.aws.amazon.com/glue/.
2. Wählen Sie im Navigationsbereich Databases (Datenbanken) aus.
3. Wählen Sie in der Liste der Datenbanken die Datenbank aus, die Sie bearbeiten möchten.
4. Wählen Sie auf der Detailseite für die Datenbank Edit (Bearbeiten) aus.
5. Geben Sie auf der Seite Update a database (Datenbank aktualisieren) für Location (Speicherort) einen Amazon-S3-Speicherort ein.
6. Wählen Sie Update Database (Datenbank aktualisieren) aus.

Lösungsvorschlag B – Verwenden Sie eine andere AWS Glue-Datenbank, die über einen vorhandenen, gültigen Speicherort in Amazon S3 verfügt. Wenn Sie beispielsweise eine Datenbank mit dem Namen `dbWithLocation` haben, verwenden Sie den Befehl `spark.sql("use dbWithLocation")`, um zu dieser Datenbank zu wechseln.

Lösungsvorschlag C – Wenn Sie Spark SQL zum Erstellen der Tabelle verwenden, geben Sie einen Wert für `location` an, wie im folgenden Beispiel.

```
spark.sql("create table testTable (firstName STRING)
       location 's3://DOC-EXAMPLE-BUCKET/'").
```

Lösungsvorschlag D – Wenn Sie bei der Erstellung der Tabelle einen Speicherort angegeben haben, das Problem jedoch weiterhin auftritt, stellen Sie sicher, dass der von Ihnen angegebene Amazon-S3-Pfad einen abschließenden Schrägstrich enthält. Der folgende Befehl löst beispielsweise eine Ausnahme für ein unzulässiges Argument aus:

```
spark.sql("create table testTable (firstName STRING)
       location 's3://DOC-EXAMPLE-BUCKET'")
```

Um dies zu korrigieren, fügen Sie dem Speicherort einen abschließenden Schrägstrich hinzu (z. B. `'s3:// DOC-EXAMPLE-BUCKET/'`).

An einem Arbeitsgruppenspeicherort erstellte Datenbank

Wenn Sie einen Befehl wie `spark.sql('create database db')` zum Erstellen einer Datenbank verwenden und keinen Speicherort für die Datenbank angeben, erstellt Athena ein Unterverzeichnis in Ihrem Arbeitsgruppenverzeichnis und verwendet diesen Speicherort für die neu erstellte Datenbank.

Probleme mit von Hive verwalteten Tabellen in der AWS Glue-Standarddatenbank

Wenn die `Location`-Eigenschaft Ihrer Standarddatenbank in AWS Glue nicht leer ist und einen gültigen Speicherort in Amazon S3 angibt und Sie Athena für Spark verwenden, um eine von Hive verwaltete Tabelle in Ihrer AWS Glue-Standarddatenbank zu erstellen, werden Daten in den Amazon-S3-Standort geschrieben, der in Ihrer Athena-Spark-Arbeitsgruppe angegeben ist, anstatt in den von der AWS Glue-Datenbank angegebenen Speicherort.

Dieses Problem tritt aufgrund der Art und Weise auf, wie Apache Hive seine Standarddatenbank behandelt. Apache Hive erstellt Tabellendaten im Stammverzeichnis des Hive-Warehouses, das sich vom tatsächlichen Standardspeicherort der Datenbank unterscheiden kann.

Wenn Sie Athena für Spark verwenden, um eine von Hive verwaltete Tabelle unter der Standarddatenbank in AWS Glue zu erstellen, können die AWS Glue-Tabellenmetadaten auf zwei verschiedene Speicherorte verweisen. Dies kann zu unerwartetem Verhalten führen, wenn Sie versuchen, einen `INSERT`- oder `DROP TABLE`-Vorgang durchzuführen.

Gehen Sie wie folgt vor, um das Problem zu reproduzieren:

1. In Athena für Spark verwenden Sie eine der folgenden Methoden zum Erstellen oder Speichern einer verwalteten Hive-Tabelle:
   - Eine SQL-Anweisung wie `CREATE TABLE $tableName`
   - Ein PySpark-Befehl wie `df.write.mode("overwrite").saveAsTable($tableName)` spezifiziert die `path` Option in der DataFrame-API nicht.

   Zu diesem Zeitpunkt zeigt die AWS Glue-Konsole möglicherweise einen falschen Speicherort für die Tabelle in Amazon S3 an.
2. In Athena für Spark verwenden Sie die `DROP TABLE $table_name`-Anweisung, um die von Ihnen erstellte Tabelle zu entfernen.

3. Nachdem Sie die `DROP TABLE`-Anweisung ausgeführt haben, stellen Sie fest, dass die zugrunde liegenden Dateien in Amazon S3 immer noch vorhanden sind.

Sie lösen dieses Problem, indem Sie einen der folgenden Schritte ausführen:

Lösung A – Verwenden Sie eine andere AWS Glue-Datenbank, wenn Sie verwaltete Hive-Tabellen erstellen.

Lösung B – Geben Sie einen leeren Speicherort für die Standarddatenbank in AWS Glue an. Erstellen Sie dann Ihre verwalteten Tabellen in der Standarddatenbank.

Inkompatibilität der CSV- und JSON-Dateiformate zwischen Athena für Spark und Athena SQL

Aufgrund eines bekannten Problems mit Open-Source-Spark ist die Tabelle, wenn Sie in Athena für Spark mit CSV- oder JSON-Daten erstellen, möglicherweise nicht von Athena SQL aus lesbar und umgekehrt.

Für das Erstellen einer Tabelle in Athena für Spark steht Ihnen beispielsweise eine der folgenden Methoden zur Verfügung:

- Mit der folgenden `USING csv`-Syntax:

```
spark.sql('''CREATE EXTERNAL TABLE $tableName (
$colName1 $colType1,
$colName2 $colType2,
$colName3 $colType3)
USING csv
PARTITIONED BY ($colName1)
LOCATION $s3_location''')
```

- Mit der folgenden DataFrame-API-Syntax:

```
df.write.format('csv').saveAsTable($table_name)
```

Aufgrund des bekannten Problems mit Open Source Spark sind Abfragen von Athena SQL für die resultierenden Tabellen möglicherweise nicht erfolgreich.

Lösungsvorschlag – Versuchen Sie, die Tabelle in Athena für Spark mit der Apache-Hive-Syntax zu erstellen. Weitere Informationen finden Sie unter HIVEFORMAT-TABELLE ERSTELLEN in der Apache-Spark-Dokumentation.

Fehlerbehebung in Athena für Spark

Verwenden Sie die folgenden Informationen, um Probleme zu beheben, die bei der Verwendung von Notebooks und Sitzungen auf Athena auftreten können.

Themen

Fehlerbehebung bei Spark-fähigen Arbeitsgruppen

Verwenden Sie die folgenden Informationen für die Fehlerbehebung in Spark-fähigen Arbeitsgruppen in Athena.

Die Sitzung reagiert nicht mehr, wenn eine vorhandene IAM-Rolle verwendet wird

Wenn Sie kein neues `AWSAthenaSparkExecutionRole` für Ihre Spark-fähige Arbeitsgruppe erstellt und stattdessen eine vorhandene IAM-Rolle aktualisiert oder ausgewählt haben, reagiert Ihre Sitzung möglicherweise nicht mehr. In diesem Fall müssen Sie möglicherweise die folgenden Vertrauens- und Berechtigungsrichtlinien zu Ihrer Spark-fähigen Arbeitsgruppen-Ausführungsrolle hinzufügen.

Fügen Sie die folgende Beispiel-Vertrauensrichtlinie hinzu. Die Richtlinie enthält eine verwechselte Stellvertreterprüfung für die Ausführungsrolle. Ersetzen Sie die Werte für *`111122223333`*, *`aws-region`* und *`workgroup-name`* durch die AWS-Konto-ID, AWS-Region und die Arbeitsgruppe, die Sie verwenden.

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

Fügen Sie eine Berechtigungsrichtlinie wie die folgende Standardrichtlinie für Notebook-fähige Arbeitsgruppen hinzu. Ändern Sie die Amazon-S3-Platzhalterstandorte und AWS-Konto-IDs so, dass sie den von Ihnen verwendeten entsprechen. Ersetzen Sie die Werte für *DOC-EXAMPLE-BUCKET*, *aws-region*, *111122223333* und *workgroup-name* durch den Amazon-S3-Bucket, AWS-Region, AWS-Konto-ID und die Arbeitsgruppe, die Sie verwenden.

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

Verwenden der Spark-EXPLAIN-Anweisung zur Fehlerbehebung in Spark SQL

Sie können die EXPLAIN-Anweisung von Spark mit Spark SQL verwenden, um Fehler in Ihrem Spark-Code zu beheben. Die folgenden Code- und Ausgabebeispiele veranschaulichen diese Verwendung.

Example – Spark-SELECT-Anweisung

```
spark.sql("select * from select_taxi_table").explain(True)
```

Ausgabe

```
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://123456789012-us-east-1-athena-results-bucket-om0yj71w5l/select_taxi],
PartitionFilters: [], PushedFilters: [],
ReadSchema: struct<VendorID:bigint,passenger_count:bigint,count:bigint>
```

Example – Spark-Datenrahmen

Das folgende Beispiel zeigt die Verwendung von `EXPLAIN` mit einem Spark-Datenrahmen.

```
taxi1_df=taxi_df.groupBy("VendorID", "passenger_count").count()
taxi1_df.explain("extended")
```

Ausgabe

```
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://athena-examples-us-east-1/
            notebooks/yellow_tripdata_2016-01.parquet], PartitionFilters: [],
            PushedFilters: [],
            ReadSchema: struct<VendorID:bigint,passenger_count:bigint>
```

Protokollieren von Spark-Anwendungsereignissen in Athena

Der Athena-Notebook-Editor ermöglicht die Standardprotokollierung von Jupyter, Spark und Python. Sie können `df.show()` verwenden, um PySpark-DataFrame-Inhalte anzuzeigen, oder `print("Output")` verwenden, um Werte in der Zellausgabe anzuzeigen. Die `stdout`-, `stderr`-, und `results`-Ausgaben für Ihre Berechnungen werden in den Bucket-Speicherort Ihrer Abfrageergebnisse in Amazon S3 geschrieben.

Protokollieren von Spark-Anwendungsereignissen mit Amazon CloudWatch

Ihre Athena-Sitzungen können auch Protokolle für Amazon CloudWatch in dem von Ihnen verwendeten Konto aufzeichnen.

Grundlegendes zu Protokollstreams und Protokollgruppen

CloudWatch organisiert Protokollaktivitäten in Protokollstreams und Protokollgruppen.

Protokollstreams – Ein CloudWatch-Protokoll-Stream ist eine Abfolge von Protokollereignissen, die dieselbe Quelle nutzen. Jede separate Quelle für Protokolle in CloudWatch Logs bildet einen separaten Protokollstream.

Protokollgruppen – In CloudWatch Logs ist eine Protokollgruppe eine Gruppe von Protokollstreams, die die gleichen Aufbewahrungs-, Überwachungs- und Zugriffssteuerungseinstellungen verwenden.

Es gibt keine Begrenzung dazu, wie viele Protokoll-Streams zu einer Protokollgruppe gehören können.

Wenn Sie in Athena zum ersten Mal eine Notebook-Sitzung starten, erstellt Athena eine Protokollgruppe in CloudWatch, die den Namen Ihrer Spark-fähigen Arbeitsgruppe verwendet, wie im folgenden Beispiel.

```
/aws-athena/workgroup-name
```

Diese Protokollgruppe erhält einen Protokollstream für jeden Executor in Ihrer Sitzung, der mindestens ein Protokollereignis erzeugt. Ein Executor ist die kleinste Recheneinheit, die eine Notebook-Sitzung von Athena anfragen kann. In CloudWatch beginnt der Name des Protokollstreams mit der Sitzungs-ID und der Executor-ID.

Weitere Informationen über CloudWatch-Protokollgruppen und Protokollstreams finden Sie unter Arbeiten mit Protokollgruppen und Protokollstreams im Benutzerhandbuch zu Amazon CloudWatch Logs.

Verwenden von Standard-Protokollierungsobjekten in Athena für Spark

In einer Athena-für-Spark-Sitzung können Sie die folgenden zwei globalen Standard-Protokollierungsobjekte verwenden, um Protokolle in Amazon CloudWatch zu schreiben:

- athena_user_logger – Sendet Protokolle ausschließlich an CloudWatch. Verwenden Sie dieses Objekt, wenn Sie Informationen zu Ihren Spark-Anwendungen direkt in CloudWatch protokollieren möchten, wie im folgenden Beispiel.

  ```
  athena_user_logger.info("CloudWatch log line.")
  ```

Das Beispiel schreibt ein Protokollereignis wie das folgende in CloudWatch:

```
AthenaForApacheSpark: 2022-01-01 12:00:00,000 INFO builtins: CloudWatch log line.
```

- athena_shared_logger – Sendet das gleiche Protokoll sowohl an CloudWatch als auch an AWS für Unterstützungszwecke. Sie können dieses Objekt verwenden, um Protokolle zur Fehlerbehebung für AWS-Service-Teams freizugeben, wie im folgenden Beispiel.

```
athena_shared_logger.info("Customer debug line.")
var = [...some variable holding customer data...]
athena_shared_logger.info(var)
```

Das Beispiel protokolliert die `debug`-Zeile und den Wert der `var`-Variablen in CloudWatch Logs und sendet eine Kopie jeder Zeile an AWS Support.

> **Note**
>
> Aus Datenschutzgründen werden Ihr Berechnungscode und Ihre Ergebnisse nicht an AWS weitergegeben. Stellen Sie sicher, dass Ihre Aufrufe an `athena_shared_logger` nur die Informationen schreiben, die Sie für AWS Support sichtbar machen möchten.

Die bereitgestellten Protokollierer schreiben Ereignisse über Apache Log4j und erben die Protokollierungsebenen dieser Schnittstelle. Mögliche Werte für die Protokollebene sind `DEBUG`, `ERROR`, `FATAL`, `INFO` und `WARN` oder `WARNING`. Sie können die entsprechend benannte Funktion auf dem Protokollierer verwenden, um diese Werte zu erzeugen.

> **Note**
>
> Verknüpfen Sie die Namen `athena_user_logger` oder `athena_shared_logger` nicht neu. Dadurch können die Protokollierungsobjekte für den Rest der Sitzung nicht mehr in CloudWatch schreiben.

Beispiel: Protokollierung von Notebook-Ereignissen in CloudWatch

Das folgende Verfahren zeigt, wie Sie Athena-Notebook-Ereignisse in Amazon CloudWatch Logs protokollieren.

So protokollieren Sie Athena-Notebook-Ereignisse in Amazon CloudWatch Logs

1. Befolgen Sie Erste Schritte mit Apache Spark auf Amazon Athena, um eine Spark-fähige Arbeitsgruppe in Athena mit einem eindeutigen Namen zu erstellen. In diesem Tutorial wird der Arbeitsgruppenname `athena-spark-example` verwendet.
2. Folgen Sie den Schritten unter Erstellen Ihres eigenen Notebooks, um ein Notebook zu erstellen und eine neue Sitzung zu starten.
3. Geben Sie im Athena-Notebook-Editor in einer neuen Notebook-Zelle den folgenden Befehl ein:

   ```
   athena_user_logger.info("Hello world.")
   ```

4. Führen Sie die Zelle aus.
5. Rufen Sie die aktuelle Sitzungs-ID ab, indem Sie eine der folgenden Aktionen ausführen:

   - Zeigen Sie die Zellenausgabe an (z. B `... session=72c24e73-2c24-8b22-14bd-443bdcd72de4`).
   - Führen Sie in einer neuen Zelle den Magic-Befehl `%session_id` aus.
6. Speichern Sie die Sitzungs-ID.
7. Öffnen Sie mit der gleichen AWS-Konto, das Sie zum Ausführen der Notebook-Sitzung verwenden, die CloudWatch-Konsole unter https://console.aws.amazon.com/cloudwatch/.
8. Wählen Sie im Navigationsbereich der CloudWatch-Konsole Protokollgruppen aus.
9. Wählen Sie in der Liste von Protokollgruppen die Protokollgruppe aus, die den Namen Ihrer Spark-fähigen Athena-Arbeitsgruppe trägt, wie im folgenden Beispiel gezeigt.

   ```
   /aws-athena/athena-spark-example
   ```

   Der Abschnitt Protokollstreams enthält eine Liste mit einem oder mehreren Protokollstream-Links für die Arbeitsgruppe. Jeder Protokollstream-Name enthält die Sitzungs-ID, die Executor-ID und die eindeutige UUID, getrennt durch Schrägstriche.

   Wenn die Sitzungs-ID beispielsweise `5ac22d11-9fd8-ded7-6542-0412133d3177` und die Executor-ID `f8c22d11-9fd8-ab13-8aba-c4100bfba7e2` lautet, ähnelt der Name des Protokollstreams dem folgenden Beispiel.

   ```
   5ac22d11-9fd8-ded7-6542-0412133d3177/f8c22d11-9fd8-ab13-8aba-c4100bfba7e2/f012d7cb-cefd-40b1-90b9-67358f003d0b
   ```

10. Wählen Sie den Protokollstream-Link für Ihre Sitzung aus.
11. Rufen Sie auf der Seite Log events (Protokollereignisse) die Spalte Message (Nachricht) auf.

    Das Protokollereignis für die Zelle, die Sie ausgeführt haben, sieht folgendermaßen aus:

    ```
    AthenaForApacheSpark: 2022-01-01 12:00:00,000 INFO builtins: Hello world.
    ```

12. Kehren Sie zum Athena-Notebook-Editor zurück.
13. Geben Sie in einer neuen Zelle den folgenden Code ein. Der Code protokolliert eine Variable in CloudWatch:

    ```
    x = 6
    athena_user_logger.warn(x)
    ```

14. Führen Sie die Zelle aus.
15. Kehren Sie zur Seite Log events (Protokollereignisse) der CloudWatch-Konsole für denselben Protokollstream zurück.
16. Der Protokollstream enthält nun einen Eintrag für ein Protokollereignis mit einer Meldung wie der folgenden:

    ```
    AthenaForApacheSpark: 2022-01-01 12:00:00,000 WARN builtins: 6
    ```

Verwenden von CloudTrail zur Fehlerbehebung bei Athena-Notebook-API-Aufrufen

Zur Fehlerbehebung bei Notebook-API-Aufrufen können Sie Athena-CloudTrail-Protokolle untersuchen, um Anomalien zu untersuchen oder von Benutzern initiierte Aktionen zu entdecken. Ausführliche Informationen zur Verwendung von CloudTrail mit Athena finden Sie unter Protokollieren von Amazon-Athena-API-Aufrufen mit AWS CloudTrail.

Die folgenden Beispiele veranschaulichen CloudTrail-Protokolleinträge für Athena-Notebook-APIs:

- StartSession
- TerminateSession
- ImportNotebook
- UpdateNotebook

- StartCalculationExecution

StartSession

Das folgende Beispiel zeigt das CloudTrail-Protokoll für ein Notebook-StartSession-Ereignis.


```
{
    "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

Das folgende Beispiel zeigt das CloudTrail-Protokoll für ein Notebook-TerminateSession-Ereignis.

```
{
    "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

Das folgende Beispiel zeigt das CloudTrail-Protokoll für ein Notebook-ImportNotebook-Ereignis. Aus Sicherheitsgründen werden einige Inhalte ausgeblendet.

```
{
    "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

Das folgende Beispiel zeigt das CloudTrail-Protokoll für ein Notebook-UpdateNotebook-Ereignis. Aus Sicherheitsgründen werden einige Inhalte ausgeblendet.

```
{
    "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

Das folgende Beispiel zeigt das CloudTrail-Protokoll für ein Notebook-StartCalculationExecution-Ereignis. Aus Sicherheitsgründen werden einige Inhalte ausgeblendet.

```
{
    "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",
```

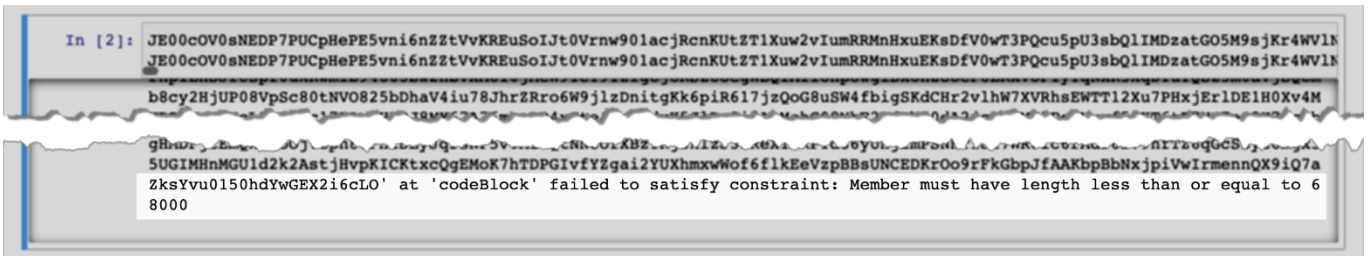
```
        "clientProvidedHostHeader": "athena.us-east-1.amazonaws.com"
    }
}
```

Überwindung der Größenbeschränkung von 68k für Codeblöcke

Athena für Spark hat eine bekannte Blockgrößenbeschränkung für Berechnungscodes von 68 000 Zeichen. Wenn Sie eine Berechnung mit einem Codeblock ausführen, der diesen Grenzwert überschreitet, kann die folgende Fehlermeldung angezeigt werden:

'...' bei 'codeBlock' konnte Einschränkung nicht erfüllen: Mitglied muss eine Länge von höchstens 68 000 haben

Die folgende Abbildung zeigt diesen Fehler im Notebook-Editor der Athena-Konsole.



Derselbe Fehler kann auftreten, wenn Sie die AWS CLI verwenden, um eine Berechnung auszuführen, die einen großen Codeblock enthält, wie im folgenden Beispiel.

```
aws athena start-calculation-execution \
    --session-id "{SESSION_ID}" \
    --description "{SESSION_DESCRIPTION}" \
    --code-block "{LARGE_CODE_BLOCK}"
```

Der Befehl gibt die folgende Fehlermeldung aus:

{LARGE_CODE_BLOCK} at 'codeBlock' konnte Einschränkung nicht erfüllen: Mitglied muss eine Länge von höchstens 68 000 haben

Workaround

Um dieses Problem zu umgehen, laden Sie die Datei mit Ihrem Anfrage- oder Berechnungscode auf Amazon S3 hoch. Verwenden Sie dann boto3, um die Datei zu lesen und Ihr SQL oder Ihren Code auszuführen.

In den folgenden Beispielen wird davon ausgegangen, dass Sie die Datei mit Ihrer SQL-Abfrage oder Ihrem Python-Code bereits auf Amazon S3 hochgeladen haben.

SQL-Beispiel

Der folgende Beispielcode liest die `large_sql_query.sql` Datei aus einem Amazon-S3-Bucket und führt dann die umfangreiche Abfrage aus, die die Datei enthält.

```
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)
```

PySpark-Beispiel

Der folgende Beispielcode liest die `large_py_spark.py`-Datei aus einem Amazon-S3-Bucket und führt dann die umfangreiche Abfrage aus, die die Datei enthält.

```
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)
```

Fehlerbehebung bei Sitzungen

Verwenden Sie die Informationen in diesem Thema, um Sitzungsprobleme zu beheben.

Sitzung in fehlerhaftem Zustand

Wenn Sie die Fehlermeldung Sitzung im fehlerhaftem Zustand erhalten. Erstellen Sie eine neue Sitzung, beenden Sie Ihre bestehende Sitzung und erstellen Sie eine neue.

Es konnte keine Verbindung zum Notebook-Server hergestellt werden

Wenn Sie ein Notebook öffnen, finden Sie möglicherweise die folgende Fehlermeldung:

```
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.
```

Ursache

Wenn Athena ein Notebook öffnet, erstellt Athena eine Sitzung und stellt mithilfe einer vorsignierten Notebook-URL eine Verbindung zum Notebook her. Die Verbindung zum Notebook verwendet das WSS-Protokoll (WebSocket Secure).

Der Fehler kann aus folgenden Gründen auftreten:

- Eine lokale Firewall (z. B. eine unternehmensweite Firewall) blockiert den WSS-Datenverkehr.
- Proxy- oder Antivirensoftware auf Ihrem lokalen Computer blockiert die WSS-Verbindung.

Lösung

Angenommen, Sie haben eine WSS-Verbindung in der `us-east-1`-Region wie die Folgende:

```
wss://94c2bcdf-66f9-4d17-9da6-7e7338060183.analytics-gateway.us-east-1.amazonaws.com/
api/kernels/33c78c82-b8d2-4631-bd22-1565dc6ec152/channels?session_id=
7f96a3a048ab4917b6376895ea8d7535
```

Wenden Sie eine der folgenden Strategien an, um den Fehler zu beheben.

- Verwenden Sie die Syntax für Platzhaltermuster, um den Listen-WSS-Verkehr an Anschluss `443` über AWS-Regionen und AWS-Konten zuzulassen.

  ```
  wss://*amazonaws.com
  ```

- Verwenden Sie die Syntax für Platzhaltermuster, um Listen-WSS-Verkehr an Anschluss `443` in einem AWS-Region und über AWS-Konten in dem von Ihnen angegebenen AWS-Region zuzulassen. Im folgenden Beispiel wird verwende `us-east-1`.

  ```
  wss://*analytics-gateway.us-east-1.amazonaws.com
  ```

Fehlerbehebung bei Tabellen

Beim Erstellen einer Tabelle kann kein Pfadfehler erstellt werden

Fehlermeldung: IllegalArgumentException: Kann keinen Pfad aus einer leeren Zeichenfolge erstellen.

Ursache: Dieser Fehler kann auftreten, wenn Sie Apache Spark in Athena zum Erstellen einer Tabelle in einer AWS Glue-Datenbank verwenden und die Datenbank eine leere `LOCATION`-Eigenschaft hat.

Lösungsvorschlag: Weitere Informationen und Lösungen finden Sie unter Unzulässige Argumentausnahme beim Erstellen einer Tabelle.

AccessDeniedException beim Abfragen von AWS Glue-Tabellen

Fehlermeldung: pyspark.sql.utils.AnalysisException: Das Vorhandensein der Standarddatenbank kann nicht überprüft werden: com.amazonaws.services.glue.model.accessDeniedException: Benutzer: arn:aws:sts::*aws-account-id*:assumed-role/awsathenasParkExecutionRole-*unique-identifier*/athenaExecutor-*unique-identifier* ist nicht autorisiert, Folgendes auszuführen: glue:GetDatabase auf der Ressource: arn:aws:glue:*aws-region*:*aws-account-id*:catalog, weil keine identitätsbasierte Richtlinie die Aktion glue:getDatabase zulässt (Service: AWSGlue; Statuscode: 400; Fehlercode: AccessDeniedException; Anforderungs-ID: *request-id*; Proxy: null)

Ursache: Der Ausführungsrolle für Ihre Spark-fähige Arbeitsgruppe fehlen die Berechtigungen für den Zugriff auf AWS Glue-Ressourcen.

Lösungsvorschlag: Um dieses Problem zu beheben, gewähren Sie Ihrer Ausführungsrolle Zugriff auf AWS Glue-Ressourcen und bearbeiten Sie dann Ihre Amazon-S3-Bucket-Richtlinie, um Ihrer Ausführungsrolle Zugriff zu gewähren.

Das folgende Verfahren beschreibt diese Schritte im Detail.

So gewähren Sie Ihrer Ausführungsrolle Zugriff auf AWS Glue-Ressourcen

1. Öffnen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.
2. Wenn der Navigationsbereich in der Konsole nicht sichtbar ist, wählen Sie das Erweiterungsmenü auf der linken Seite.

3. Wählen Sie im Navigationsbereich der Athena-Konsole Workgroups (Arbeitsgruppen) aus.
4. Wählen Sie auf der Seite der Workgroups (Arbeitsgruppen) den Link der Arbeitsgruppe aus, die Sie anzeigen möchten.
5. Wählen Sie auf der Seite Overview Details (Übersichtsdetails) für die Arbeitsgruppe den Link Role ARN (Rollen-ARN) aus. Über den Link wird die Spark-Ausführungsrolle in der IAM-Konsole geöffnet.
6. Wählen Sie im Abschnitt Permissions policies (Berechtigungsrichtlinien) den Namen der verknüpften Rollenrichtlinie.
7. Wählen Sie Edit policy (Richtlinie bearbeiten) und anschließend JSON.
8. Fügen Sie der Rolle AWS Glue-Zugriff hinzu. In der Regel fügen Sie Berechtigungen für die Aktionen `glue:GetDatabase` und `glue:GetTable` hinzu. Weitere Informationen zum Konfigurieren von IAM-Rollen finden Sie unter Hinzufügen und Entfernen von IAM-Identitätsberechtigungen im IAM-Benutzerhandbuch.
9. Wählen Sie Review policy (Richtlinie überprüfen) aus und klicken Sie anschließend auf Save changes (Änderungen speichern).
10. Bearbeiten Sie Ihre Amazon-S3-Bucket-Richtlinie, um Zugriff auf die Ausführungsrolle zu gewähren. Beachten Sie, dass Sie der Rolle sowohl Zugriff auf den Bucket als auch auf die Objekte im Bucket gewähren müssen. Weitere Schritte finden Sie unter Hinzufügen einer Bucket-Richtlinie mit der Amazon-S3-Konsole im Benutzerhandbuch zu Amazon Simple Storage Service.

Supportanfragen

Um Unterstützung von AWS zu erhalten, wählen Sie Support, Support Center aus dem AWS Management Console aus. Um Ihre Erfahrung zu erleichtern, halten Sie bitte die folgenden Informationen bereit:

- Athena-Abfragen-ID
- Sitzungs-ID
- Berechnungs-ID

Versionshinweise

Beschreibt Features, Verbesserungen und Fehlerbehebungen von Amazon Athena nach Veröffentlichungsdatum.

Themen

Athena-Versionshinweise für 2024

15. März 2024

Veröffentlicht am 18.03.2024

Amazon Athena kündigt die Verfügbarkeit von Athena SQL in der Region Kanada West (Calgary) an.

Eine vollständige Liste der in den einzelnen AWS-Region Ländern AWS-Services verfügbaren AWS Dienste finden Sie unter Services nach Regionen.

15. Februar 2024

Veröffentlicht am 15.02.2021

Athena veröffentlicht die JDBC-Treiberversion 3.1.0.

Die Amazon Athena JDBC-Treiberversion 3.1.0 bietet Unterstützung für Microsoft Active Directory Federation Services (AD FS) Windows Integrated Authentication und formularbasierte Authentifizierung. Die Version 3.1.0 enthält auch weitere kleinere Verbesserungen und Fehlerkorrekturen.

Informationen zum Herunterladen des JDBC v3-Treibers finden Sie unter. JDBC-3.x-Treiber-Download

31. Januar 2024

Veröffentlicht am 31.01.2021

Athena kündigt die folgenden Features und Verbesserungen an.

- Hudi-Upgrade — Sie können jetzt Athena SQL verwenden, um Hudi 0.14.0-Tabellen abzufragen. Hinweise zur Verwendung von Athena SQL zur Abfrage von Hudi-Tabellen finden Sie unter. Verwenden von Athena zum Abfragen von Apache-Hudi-Datensätzen

Athena-Versionshinweise für 2023

14. Dezember 2023

Veröffentlicht am 14.12.2023

Athena kündigt die folgenden Korrekturen und Verbesserungen an.

Athena veröffentlicht JDBC-Treiber-Version 2.1.3. Der Treiber behebt die folgenden Probleme:

- Die Protokollierung wurde verbessert, um Konflikte mit der Anwendungsprotokollierung von Spring Boot und Gradle zu vermeiden.
- Bei Verwendung der `executeBatch()`-JDBC-Methode zum Einfügen von Datensätzen hat der Treiber fälschlicherweise nur einen Datensatz eingefügt. Da Athena keine Batch-Ausführung von Abfragen unterstützt, meldet der Treiber bei Verwendung von `executeBatch()` einen Fehler. Um die Einschränkung zu umgehen, können Sie einzelne Abfragen in einer Schleife übermitteln.

Informationen zum Herunterladen der neuen JDBC-Treiber, Versionshinweise und Dokumentation finden Sie unter Athena-JDBC-2.x-Treiber.

09. Dezember 2023

Veröffentlicht am 09.12.2023

Veröffentlichung des ODBC-1.2.1.1000-Treibers für Athena.

Features und Verbesserungen:

- Aktualisierte RStudio-Unterstützung – Der ODBC-Treiber unterstützt jetzt RStudio unter macOS.
- Unterstützung für einzelne Kataloge und Schemas – Der Konnektor kann jetzt einen einzelnen Katalog und ein einzelnes Schema zurückgeben. Weitere Informationen finden Sie im herunterladbaren Installations- und Konfigurationsleitfaden.

Behobene Probleme:

- Vorbereitete Anweisungen – Bei der Ausführung vorbereiteter Anweisungen mit einem Array von Parametern unter Verwendung eines spaltenweisen Schemas gab der Konnektor ein falsches Abfrageergebnis zurück.
- Spaltengröße – Bei Auswahl der Systemspalte `$file_modified_time` hat der Konnektor eine falsche Spaltengröße zurückgegeben.
- SQLPrepare – Beim Binden von `SQLPrepare`-bezogenen Parametern in `SELECT`-Abfragen gab der Konnektor einen Fehler zurück.

Weitere Informationen und das Herunterladen des neuen Treibers, der Versionshinweise und der Dokumentation finden Sie unter Athena-ODBC-1.x-Treiber.

07. Dezember 2023

Veröffentlicht am 07.12.2023

Athena kündigt ODBC-Treiber-Version 2.0.2.1 an. Weitere Informationen finden Sie in den 2.0.2.1-Versionshinweisen. Informationen zum Herunterladen des neuesten ODBC-v2-Treibers finden Sie unter ODBC-2.x-Treiber-Download. Verbindungsinformationen finden Sie unter ODBC-2.x-Verbindungen von Amazon Athena konfigurieren.

05. Dezember 2023

Veröffentlicht am 05.12.2023

Sie können jetzt Athena SQL-Arbeitsgruppen erstellen, die den AWS IAM Identity Center Authentifizierungsmodus verwenden. Diese Arbeitsgruppen unterstützen das Feature der Weitergabe vertrauenswürdiger Identitäten von IAM Identity Center. Trusted Identity Propagation ermöglicht die Verwendung von Identitäten in AWS Analysediensten wie Amazon Athena und Amazon EMR Studio.

Weitere Informationen finden Sie unter Verwendung von für IAM Identity Center aktivierten Athena-Arbeitsgruppen.

28. November 2023

Veröffentlicht am 28.11.2023

Sie können jetzt Daten in der Speicherklasse Amazon S3 Express One Zone abfragen, um schnelle Abfrageergebnisse zu erhalten. S3 Express One Zone ist ein hochleistungsfähiger Speicher mit einer einzelnen Availability Zone, der entwickelt wurde, um einen konsistenten Datenzugriff im einstelligen Millisekundenbereich für Ihre Daten, auf die am häufigsten zugegriffen wird, und für latenzempfindliche Anwendungen zu gewährleisten. Verschieben Sie zunächst Ihre Daten in den Speicher der S3 Express One Zone und katalogisieren Sie die Daten mit AWS Glue Data Catalog für ein nahtloses Abfrageerlebnis in Athena.

Weitere Informationen finden Sie unter Abfragen von Daten der S3 Express One Zone.

8. November 2023

Veröffentlicht am 27.11.2023

Athena kündigt die folgenden Features und Verbesserungen an.

- Glue-Datenkatalogansichten — Glue-Datenkatalogansichten bieten eine einzige gemeinsame Ansicht für AWS Dienste wie Amazon Athena und Amazon Redshift. In Glue-Data-Catalog-Ansichten werden Zugriffsberechtigungen durch den Benutzer definiert, der die Ansicht erstellt hat, und nicht durch den Benutzer, der die Ansicht abfragt. Diese Ansichten bieten eine bessere Zugriffskontrolle, tragen zur Gewährleistung vollständiger Datensätze bei, bieten erhöhte Sicherheit und können den Zugriff auf zugrunde liegende Tabellen verhindern.

  Weitere Informationen finden Sie unter AWS Glue Data Catalog Ansichten verwenden.
- CloudTrail Lake-Unterstützung — Sie können jetzt Amazon Athena verwenden, um Daten in AWS CloudTrail Lake zu analysieren. AWS CloudTrail Lake ist ein verwalteter Data Lake CloudTrail , mit dem Sie Aktivitätsprotokolle für Audit-, Sicherheits- und Betriebsuntersuchungen aggregieren, unveränderlich speichern und analysieren können. Um Ihre CloudTrail Lake-Aktivitätsprotokolle von Athena abzufragen, müssen Sie keine Daten verschieben oder separate Datenverarbeitungspipelines erstellen. Es sind keine ETL-Vorgänge erforderlich.

  Aktivieren Sie zunächst den Datenverbund in CloudTrail Lake. Wenn Sie die Metadaten Ihres CloudTrail Lake-Ereignisdatenspeichers mit teilen AWS Glue Data Catalog, werden die

erforderlichen AWS Glue Data Catalog Ressourcen CloudTrail erstellt und die Daten mit registriert AWS Lake Formation. In Lake Formation können Sie die Benutzer und Rollen angeben, die Athena zum Abfragen Ihres Ereignisdatenspeichers verwenden können.

Weitere Informationen finden Sie unter Aktivieren der Lake-Abfrageverbund im AWS CloudTrail - Benutzerhandbuch.

17. November 2023

Veröffentlicht am 17.11.2023

Athena kündigt die folgenden Features und Verbesserungen an.

Features

- Kostenbasierter Optimierer — Athena kündigt die allgemeine Verfügbarkeit der kostenbasierten Optimierung unter Verwendung von Statistiken von an. AWS Glue Um Ihre Abfragen in Athena SQL zu optimieren, können Sie anfordern, dass Athena Statistiken auf Tabellen- oder Spaltenebene für Ihre Tabellen in AWS Glue sammelt. Wenn alle Tabellen in Ihrer Abfrage Statistiken enthalten, verwendet Athena die Statistiken, um alternative Ausführungspläne zu untersuchen und den Plan auszuwählen, der am wahrscheinlichsten am schnellsten ist.

  Weitere Informationen finden Sie unter Verwenden des kostenbasierten Optimierers.
- Amazon-EMR-Studio-Integration – Sie können Athena jetzt in einem Amazon EMR Studio verwenden, ohne die Athena-Konsole direkt verwenden zu müssen. Mit der Athena-Integration in Amazon EMR können Sie die folgenden Aufgaben ausführen:
  - Athena-SQL-Abfragen ausführen
  - Abfrageergebnisse anzeigen
  - Abfrageverlauf anzeigen
  - Gespeicherte Abfragen anzeigen
  - Parametrisierten Abfragen durchführen
  - Datenbanken, Tabellen und Ansichten für einen Datenkatalog anzeigen

  Weitere Informationen finden Sie unter Amazon EMR Studio im Thema AWS-Service Integrationen mit Athena.
- Verschachtelte Zugriffskontrolle – Athena kündigt Unterstützung für Lake-Formation-Zugriffskontrolle für verschachtelte Daten an. In Lake Formation können Sie Datenfilter für

verschachtelte Spalten mit `struct`-Datentypen definieren und anwenden. Sie können die Datenfilterung verwenden, um den Benutzerzugriff auf Unterstrukturen verschachtelter Spalten einzuschränken. Weitere Informationen zum Erstellen von Datenfiltern finden Sie unter Erstellen von Datenfiltern in Lake Formation im AWS Lake Formation -Entwicklerhandbuch.

- Metriken zur bereitgestellten Kapazitätsnutzung — Athena kündigt neue CloudWatch Metriken für Kapazitätsreservierungen an. Sie können die neuen Metriken verwenden, um die Anzahl der von Ihnen bereitgestellten DPUs und die Anzahl der von Ihren Abfragen verwendeten DPUs zu verfolgen. Wenn die Abfragen abgeschlossen sind, können Sie auch die Anzahl der DPUs anzeigen, die von der Abfrage konsumiert wurden.

  Weitere Informationen finden Sie unter Überwachung von Athena-Abfragen mit CloudWatch Metriken.

Verbesserungen

- Änderung der Fehlermeldung – Die Fehlermeldung `Insufficient Lake Formation permissions` lautet jetzt `Table not found` oder `Schema not found`. Diese Änderung wurde vorgenommen, um zu verhindern, dass böswillige Akteure aus der Fehlermeldung auf die Existenz von Tabellen- oder Datenbankressourcen schließen.

16. November 2023

Veröffentlicht am 16.11.2023

Athena veröffentlicht einen neuen JDBC-Treiber, der die Verbindung, Abfrage und Visualisierung von Daten aus kompatiblen SQL-Entwicklungs- und Business-Intelligence-Anwendungen verbessert. Der neue Treiber ist einfach zu aktualisieren. Der Treiber kann Abfrageergebnisse direkt aus Amazon S3 lesen, wodurch Ihnen Abfrageergebnisse schneller bereitgestellt werden.

Weitere Informationen finden Sie unter Athena-JDBC-3.x-Treiber.

31. Oktober 2023

Veröffentlicht am 31.10.2023

Amazon Athena kündigt einstündige Reservierungen für bereitgestellte Kapazität an. Ab heute können Sie bereitgestellte Kapazitäten reservieren und nach einer Stunde wieder freigeben. Diese

Änderung macht es einfacher, die Kosten für Workloads zu optimieren, deren Bedarf sich im Laufe der Zeit ändert.

Bereitgestellte Kapazität ist ein Feature von Athena, das Workload-Management-Funktionen bereitstellt, mit denen Sie Ihre wichtigsten interaktiven Workloads priorisieren, kontrollieren und skalieren können. Sie können jederzeit Kapazität hinzufügen, um die Anzahl der Abfragen zu erhöhen, die Sie gleichzeitig ausführen können, zu kontrollieren, welche Workloads die Kapazität nutzen können, und die Kapazität auf mehrere Workloads verteilen.

Weitere Informationen finden Sie unter Kapazität zur Abfrageverarbeitung verwalten. Informationen zur Preisgestaltung finden Sie auf der Preisseite von Amazon Athena.

25. Oktober 2023

Veröffentlicht am 26.10.2023

Athena kündigt die folgenden Korrekturen und Verbesserungen an.

Jackson-Core-Paket – JSON-Text mit einem numerischen Wert von mehr als 1 000 Zeichen schlägt jetzt fehl. Diese Korrektur behebt das Sicherheitsproblem sonatype-2022-6438.

17. Oktober 2023

Veröffentlicht am 17.10.2023

Athena kündigt die ODBC-Treiberversion 2.0.2.0 an. Weitere Informationen finden Sie in den 2.0.2.0-Versionshinweisen. Informationen zum Herunterladen des neuesten ODBC-v2-Treibers finden Sie unter ODBC-2.x-Treiber-Download. Verbindungsinformationen finden Sie unter ODBC-2.x-Verbindungen von Amazon Athena konfigurieren.

26. September 2023

Veröffentlicht am 26.09.2023

Athena kündigt die folgenden Features und Verbesserungen an.

- Leseunterstützung bei Lake Formation für Delta-Lake-Tabellen. Weitere Informationen zur Verwendung von Delta-Lake-Tabellen in Athena finden Sie unter Delta-Lake-Tabellen von Linux Foundation abfragen.

23. August 2023

Veröffentlicht am 23.08.2023

Amazon Athena gibt die Verfügbarkeit von Athena SQL in der Region Israel (Tel Aviv) bekannt.

Eine vollständige Liste der in den einzelnen Ländern AWS-Services verfügbaren AWS Dienste finden Sie AWS-Region unter Services nach Regionen.

10. August 2023

Veröffentlicht am 10.08.2023

Athena kündigt die folgenden Korrekturen und Verbesserungen an.

ODBC-Treiberversion 2.0.1.1

Athena kündigt die ODBC-Treiberversion 2.0.1.1 an. Weitere Informationen finden Sie in den 2.0.1.1-Versionshinweisen. Informationen zum Herunterladen des neuesten ODBC-v2-Treibers finden Sie unter ODBC-2.x-Treiber-Download. Verbindungsinformationen finden Sie unter ODBC-2.x-Verbindungen von Amazon Athena konfigurieren.

JDBC-Treiberversion 2.1.1

Athena veröffentlicht JDBC-Treiberversion 2.1.1. Der Treiber behebt die folgenden Probleme:

- Ein Fehler, der auftrat, als eine Tabelle mit einer Anweisung erstellt wurde, die einen regulären Ausdruck enthielt.
- Ein Problem, das dazu führte, dass der `ApplicationName`-Verbindungsparameter falsch angewendet wurde.

Informationen zum Herunterladen der neuen JDBC-Treiber, Versionshinweise und Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC.

31. Juli 2023

Veröffentlicht am 31.07.2023

Amazon Athena kündigt die Verfügbarkeit von Athena SQL in zusätzlichen AWS-Regionen an.

Diese Version erweitert die Verfügbarkeit von Athena SQL um Asien-Pazifik (Hyderabad), Asien-Pazifik (Melbourne), Europa (Spanien) und Europa (Zürich).

Eine vollständige Liste der in den einzelnen AWS-Services AWS-Region Ländern verfügbaren AWS Dienste finden Sie unter Dienste nach Regionen.

27. Juli 2023

Veröffentlicht am 27.07.2023

Athena veröffentlicht die BigQuery Google-Connector-Version 2023.30.1. Diese Version des Connectors reduziert die Ausführungszeit von Abfragen und bietet Unterstützung für Abfragen an privaten Endpunkten. BigQuery

Informationen zum BigQuery Google-Connector finden Sie unter. Amazon Athena Google-Konnektor BigQuery Weitere Hinweise zur Aktualisierung Ihrer Datenquellen-Konnektoren finden Sie unter Schreiben eines Datenquellen-Konnektors.

24. Juli 2023

Veröffentlicht am 24.07.2023

Athena kündigt die folgenden Korrekturen und Verbesserungen an.

- Abfragen mit Unions – Die Leistung bestimmter Abfragen mit Unions wurde verbessert.
- Verknüpfungen mit Typvergleichen – Es wurde ein potenzieller Abfragefehler für `JOIN`-Anweisungen behoben, die einen Vergleich zwischen zwei verschiedenen Typen beinhalteten.
- Unterabfragen für verschachtelte Spalten – Es wurde ein Problem behoben, das mit Abfragefehlern zusammenhing, wenn Unterabfragen in verschachtelten Spalten korreliert wurden.
- Iceberg-Ansichten – Ein Kompatibilitätsproblem mit der Genauigkeit von Zeitstempelspalten in Apache-Iceberg-Ansichten wurde behoben. Iceberg-Ansichten mit Zeitstempelspalten sind jetzt lesbar, unabhängig davon, ob die Spalten mit Athena-Engine-Version 2 oder Athena-Engine-Version 3 erstellt wurden.

20. Juli 2023

Veröffentlicht am 20.07.2023

Athena veröffentlicht JDBC-Treiberversion 2.1.0. Der Treiber enthält neue Verbesserungen und hat ein Problem behoben.

Verbesserungen

Die folgenden Jackson-JSON-Parser-Bibliotheken wurden aktualisiert:

- jackson-annotations 2.15.2 (zuvor 2.14.0)
- jackson-core 2.15.2 (zuvor 2.14.0)
- jackson-databind 2.15.2 (zuvor 2.14.0)

Gelöste Probleme

- Es wurde ein Problem mit der Übergabe von Array-Parametern behoben, wenn die Bibliothek sql2o verwendet wurde.

Weitere Informationen und das Herunterladen des neuen Treibers, der Versionshinweise und der Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC.

13. Juli 2023

Veröffentlicht am 19.09.2023

Athena kündigt die folgenden Features und Verbesserungen an.

- EXPLAIN ANALYZE – Unterstützung für Warteschlangen, Analyse, Planung und Ausführungszeit wurde zur Ausgabe von `EXPLAIN ANALYZE` hinzugefügt.
- EXPLAIN – Die `EXPLAIN`-Ausgabe zeigt jetzt Statistiken an, wenn die Abfrage Aggregationen enthält.
- Parquet Hive SerDe — Die `parquet.ignore.statistics` Eigenschaft wurde hinzugefügt, mit der Verarbeitungsstatistiken beim Lesen von Parquet-Daten ignoriert werden können. Weitere Informationen finden Sie unter Parquet-Statistiken ignorieren.

Weitere Informationen zu `EXPLAIN` und `EXPLAIN ANALYZE` finden Sie unter Verwenden von EXPLAIN und EXPLAIN ANALYZE in Athena. Weitere Informationen zum Parquet Hive finden Sie SerDe unter. Parquet SerDe

03. Juli 2023

Veröffentlicht am 25.07.2023

Am 3. Juli 2023 begann Athena, die Abfragezeichenfolgen aus CloudTrail den Protokollen zu redigieren. Die Abfragezeichenfolge hat jetzt einen Wert von `***OMITTED***`. Diese Änderung wurde vorgenommen, um die unbeabsichtigte Offenlegung von Tabellennamen oder Filterwerten zu verhindern, die vertrauliche Informationen enthalten könnten. Wenn Sie sich bisher auf CloudTrail Protokolle verlassen haben, um auf vollständige Abfragezeichenfolgen zuzugreifen, empfehlen wir, die `Athena::GetQueryExecution` API zu verwenden und den Wert von `responseElements.queryExecutionId` aus dem CloudTrail Protokoll zu übergeben. Weitere Informationen finden Sie unter der GetQueryExecutionAktion in der Amazon Athena API-Referenz.

30. Juni 2023

Veröffentlicht am 30.06.2023

Der Athena-Abfrage-Editor unterstützt jetzt Typeahead-Code-Vorschläge für eine schnellere Abfrageerstellung. Mithilfe der folgenden Feature können Sie jetzt SQL-Abfragen mit verbesserter Genauigkeit und Effizienz schreiben:

- Während der Eingabe werden in Echtzeit Vorschläge für Schlüsselwörter, lokale Variablen, Codefragmente und Katalogelemente angezeigt.
- Wenn Sie einen Datenbank- oder Tabellennamen gefolgt von einem Punkt eingeben, zeigt der Editor komfortabel eine Liste von Tabellen oder Spalten an, aus denen Sie wählen können.
- Wenn Sie den Mauszeiger über einen Codefragmentvorschlag bewegen, wird in einer Zusammenfassung ein kurzer Überblick über die Syntax und Verwendung des Codefragments angezeigt.
- Um die Lesbarkeit des Codes zu verbessern, wurden auch die Keywords und ihre Hervorhebungsregeln aktualisiert, sodass sie der neuesten Syntax von Trino und Hive entsprechen.

Dieses Feature ist standardmäßig aktiviert. Sie können das Feature in den Einstellungen des Code-Editors aktivieren oder deaktivieren.

Um die Typeahehead-Codevorschläge im Athena-Abfrage-Editor auszuprobieren, besuchen Sie die Athena-Konsole unter https://console.aws.amazon.com/athena/.

29. Juni 2023

Veröffentlicht am 29.06.2023

- Athena kündigt die ODBC-Treiberversion 2.0.1.0 an. Weitere Informationen finden Sie in den 2.0.1.0-Versionshinweisen. Informationen zum Herunterladen des neuesten ODBC-v2-Treibers finden Sie unter ODBC-2.x-Treiber-Download. Verbindungsinformationen finden Sie unter ODBC-2.x-Verbindungen von Amazon Athena konfigurieren.
- Athena und seine Feature sind jetzt in der Region Naher Osten (VAE) verfügbar. Eine vollständige Liste der jeweils AWS-Services verfügbaren AWS Services finden Sie AWS-Region unter Services nach Regionen.

28. Juni 2023

Veröffentlicht am 28.06.2023

Sie können Amazon Athena verwenden, um wiederhergestellte Objekte aus den Amazon-S3-Speicherklassen S3 Glacier Flexible Retrieval (früher Glacier) und S3 Glacier Deep Archive abzufragen. Sie konfigurieren diese Funktion für jede Tabelle einzeln. Das Feature wird nur für Apache-Hive-Tabellen auf der Athena-Engine-Version 3 unterstützt.

Weitere Informationen finden Sie unter Wiederhergestellte Amazon-S3-Glacier-Objekte abfragen.

12. Juni 2023

Veröffentlicht am 12.06.2023

Athena kündigt die folgenden Korrekturen und Verbesserungen an.

- Parquet Reader-Zeitstempel – Unterstützung für das Lesen von Zeitstempeln als `bigint` (Millisekunden) für Parquet Reader hinzugefügt. Dieses Update bietet Parität mit der Unterstützung in Athena-Engine-Version 2.
- EXPLAIN ANALYZE – Der Abfragestatistik und der Ausgabe von `EXPLAIN ANALYZE` wurde die Lesezeit für physische Eingaben hinzugefügt. Weitere Informationen zu `EXPLAIN ANALYZE` finden Sie unter Verwenden von EXPLAIN und EXPLAIN ANALYZE in Athena.
- INSERT – Verbesserte Abfrageleistung bei Tabellen, in die mit `INSERT` geschrieben wurde. Weitere Informationen zu `INSERT` finden Sie unter INSERT INTO.

- Delta-Lake-Tabellen – Es wurde ein Problem mit `DROP TABLE`-Delta-Lake-Tabellen behoben, das verhinderte, dass sie vollständig gelöscht wurden, wenn sie gleichzeitig geändert wurden.

08. Juni 2023

Veröffentlicht am 08.06.2023

Amazon Athena für Apache Spark kündigt die folgenden neuen Features an.

- Support für benutzerdefinierte Java-Bibliotheken und Konfigurationen – Sie können jetzt Ihre eigenen Java-Pakete und benutzerdefinierte Konfigurationen für Ihre Apache-Spark-Sitzungen in Athena verwenden. Verwenden Sie Spark-Eigenschaften, um `.jar` Dateien, Pakete oder andere benutzerdefinierte Konfigurationen mit der Athena-Konsole AWS CLI, der oder der Athena-API anzugeben. Weitere Informationen finden Sie unter Hinzufügen von JAR-Dateien und benutzerdefinierte Spark-Konfiguration.
- Support für Apache-Hudi-, Apache-Iceberg- und Delta-Lake-Tabellen – Athena für Spark unterstützt jetzt die Open-Source-Data-Lake-Speichertabellenformate Apache Iceberg, Apache Hudi und Delta Lake der Linux Foundation Delta Lake. Weitere Informationen finden Sie unter Nicht-Hive-Tabellenformaten in Amazon Athena für Apache Spark verwenden und in den einzelnen Themen zur Verwendung von Apache Iceberg, Apache Hudi und Linux Foundation Delta Lake-Tabellen in Athena für Spark.
- Verschlüsselungsunterstützung für Apache Spark – In Athena für Spark können Sie jetzt die Verschlüsselung für Daten während der Übertragung aktivieren, die zwischen Spark-Knoten übertragen werden, und für lokale Daten im Ruhezustand, die von Spark auf der Festplatte gespeichert werden. Um die Spark-Verschlüsselung zu aktivieren, können Sie die Athena-Konsole AWS CLI, die oder die Athena-API verwenden. Weitere Informationen finden Sie unter Apache-Spark-Verschlüsselung aktivieren.

Weitere Informationen zu Amazon Athena für Apache Spark finden Sie unterVerwenden von Apache Spark in Amazon Athena.

02. Juni 2023

Veröffentlicht am 06.02.2023

Sie können jetzt Kapazitätsreservierungen in Athena löschen und AWS CloudFormation Vorlagen verwenden, um Athena-Kapazitätsreservierungen anzugeben.

- Kapazitätsreservierungen löschen – Sie können jetzt stornierte Kapazitätsreservierungen in Athena löschen. Eine Reservierung muss storniert werden, bevor sie gelöscht werden kann. Durch das Löschen einer Kapazitätsreservierung wird die Reservierung sofort aus Ihrem Konto entfernt. Auf die gelöschte Reservierung kann nicht mehr verwiesen werden, auch nicht anhand ihres ARN. Um eine Reservierung zu löschen, können Sie die Athena-Konsole oder die Athena-API verwenden. Weitere Informationen finden Sie Eine Kapazitätsreservierung löschen im Amazon Athena Athena-Benutzerhandbuch und DeleteCapacityReservationin der Amazon Athena Athena-API-Referenz.
- AWS CloudFormation Vorlagen für Kapazitätsreservierungen verwenden — Sie können jetzt AWS CloudFormation Vorlagen verwenden, um Athena-Kapazitätsreservierungen mithilfe der `AWS::Athena::CapacityReservation` Ressource anzugeben. Weitere Informationen finden Sie unter AWS: :Athena:: CapacityReservation im AWS CloudFormation Benutzerhandbuch.

Weitere Informationen zur Nutzung von Kapazitätsreservierungen zur Bereitstellung Ihrer Kapazität in Athena finden Sie unter Kapazität zur Abfrageverarbeitung verwalten.

25. Mai 2023

Veröffentlicht am 25.05.2023

Athena hat Updates für Datenquellenkonnektoren veröffentlicht, die die Leistung von Verbundabfragen verbessern. Dank neuer Push-down-Optimierungen und dynamischer Filterung können mehr Operationen in der Quelldatenbank als in Athena ausgeführt werden. Diese Optimierungen reduzieren die Laufzeit von Abfragen und die Menge der gescannten Daten. Diese Verbesserungen erfordern die Athena-Engine-Version 3.

Die folgenden Konnektoren wurden aktualisiert:

- Azure Data Lake Storage
- Azure Synapse
- Cloudera Hive
- Cloudera Impala
- Db2
- DynamoDB
- Google BigQuery
- Hortonworks
- MySQL

- Oracle
- PostgreSQL
- Redshift
- SAP HANA
- Snowflake
- SQL Server
- Teradata

Weitere Hinweise zu Datenquellen-Konnektoren finden Sie unter Schreiben eines Datenquellen-Konnektors.

18. Mai 2023

Veröffentlicht am 18.05.2023

Sie können es jetzt AWS PrivateLink für eingehende IPv6-Verbindungen zu Amazon Athena verwenden.

Amazon Athena hat seine Unterstützung für eingehende Verbindungen über IPv6-Endpunkte (Internet Protocol Version 6) um AWS PrivateLink erweitert. Ab heute können Sie sich über Ihre Amazon Virtual Private Cloud (Amazon VPC) sicher und privat mit AWS PrivateLink Athena verbinden, zusätzlich zu den öffentlichen IPv6-Endpunkten, die zuvor verfügbar waren.

Das schnelle Wachstum des Internets erschöpft die Verfügbarkeit von IPv4-Adressen (Internet Protocol Version 4). IPv6 erhöht die Anzahl der verfügbaren Adressen um ein Vielfaches, sodass Sie sich nicht mehr mit überlappenden Adressräumen in Ihren VPCs befassen müssen. Mit dieser Version können Sie nun die Vorteile der IPv6-Adressierung mit den Sicherheits- und Leistungsvorteilen von AWS PrivateLink kombinieren.

Um programmgesteuert eine Verbindung zu einem AWS Service herzustellen, können Sie das AWS CLIoder AWS SDK verwenden, um einen Endpunkt anzugeben. Weitere Informationen zu Service-Endpunkten und Athena-Service-Endpunkten finden Sie unter AWS -Service-Endpunkte und Amazon-Athena-Endpunkte und Kontingente in der Allgemeine Amazon Web Services-Referenz.

15. Mai 2023

Veröffentlicht am 15.05.2023

Athena kündigt die Veröffentlichung von Apache Spark DataSource V2 (DSV2) -Konnektoren für DynamoDB, CloudWatch Logs, CloudWatch Metrics und CMDB an. AWS Verwenden Sie die neuen DSV2-Konnektoren, um diese Datenquellen mit Spark abzufragen. DSV2-Konnektoren verwenden dieselben Parameter wie ihre entsprechenden Athena-Verbundkonnektoren. Die DSV2-Konnektoren werden direkt auf Spark-Workern ausgeführt und erfordern keine Bereitstellung einer Lambda-Funktion, um sie zu verwenden.

Weitere Informationen finden Sie unter Athena-Datenquellenkonnektoren für Apache Spark.

10. Mai 2023

Veröffentlicht am 10.05.2023

Veröffentlichung des ODBC-1.1.20-Treibers für Athena.

Features und Verbesserungen:

- Unterstützung für das Überschreiben von Endpunkten in Lake Formation.
- Das ADFS-Authentifizierungs-Plugin verfügt über einen neuen Parameter für die Einstellung des Werts Relying Party (`LoginToRP`).
- AWS Aktualisierungen der Bibliothek.

Fehlerbehebungen:

- Fehler bei der Freigabe der vorbereiteten Anweisung, wenn die `SQLPrepare()`-Methode nicht gesendet werden konnte.
- Fehler beim Binden der Parameter für vorbereitete Anweisungen bei der Konvertierung eines C-Typs in einen SQL-Typ.
- Fehler beim Zurückgeben von Daten, wenn `EXPLAIN`- und `EXPLAIN ANALYZE`-Abfragen `SQLPrepare()` und `SQLExecute()` verwendeten.

Weitere Informationen und das Herunterladen des neuen Treibers, der Versionshinweise und der Dokumentation finden Sie unter Herstellen einer Verbindung mit Amazon Athena mit ODBC.

8. Mai 2023

Veröffentlicht am 08.05.2023

Athena kündigt die folgenden Korrekturen und Verbesserungen an.

- Aktualisierte Hudi-Integration – Athena hat seine Integration mit Apache Hudi aktualisiert. Sie können jetzt Athena verwenden, um Hudi-0.12.2-Tabellen abzufragen, und die Auflistung von Hudi-Metadaten für Hudi-Tabellen wird jetzt unterstützt. Weitere Informationen finden Sie unter Verwenden von Athena zum Abfragen von Apache-Hudi-Datensätzen und Liste der Hudi-Metadaten.
- Korrektur der Zeitstempelkonvertierung – Die Behandlung von Zeitstempelkonvertierungen in einen Datentyp mit niedrigerer Genauigkeit wurde korrigiert. Zuvor wurde in Athena-Engine-Version 3 der Wert fälschlicherweise auf den Zieltyp gerundet, anstatt ihn beim Casting zu kürzen.

  Die folgenden Beispiele veranschaulichen die falsche Handhabung vor dem Fix.

  Beispiel 1: Umwandlung von einem Zeitstempel in Mikrosekunden in Millisekunden

  Beispieldaten

```
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
```

  Die folgende Abfrage versucht, die Zeitstempel abzurufen, die einem bestimmten Wert entsprechen.

```
SELECT *
FROM table
WHERE timestamps.col = timestamp'2020-06-10 15:55:23.383'
```

  Die Abfrage gibt die folgenden Ergebnisse zurück.

```
A, 2020-06-10 15:55:23.383
C, 2020-06-10 15:55:23.383
E, 2020-06-10 15:55:23.383
```

  Vor dem Update hat Athena die Werte `2020-06-10 15:55:23.383945` oder `2020-06-10 15:55:23.383945278` nicht aufgenommen, weil sie auf `2020-06-10 15:55:23.384` gerundet wurden.

Beispiel 2: Umwandlung von einem Zeitstempel in ein Datum

Die folgende Abfrage hat ein fehlerhaftes Ergebnis zurückgegeben.

```
SELECT date(timestamp '2020-12-31 23:59:59.999')
```

Ergebnis

```
2021-01-01
```

Vor der Korrektur hat Athena den Wert aufgerundet und damit den Tag vorverlegt. Solche Werte werden jetzt eher gekürzt als aufgerundet.

28. April 2023

Veröffentlicht am 28.04.2023

Sie können jetzt Kapazitätsreservierungen auf Amazon Athena verwenden, um SQL-Abfragen auf vollständig verwalteter Rechenkapazität auszuführen.

Bereitgestellte Kapazität bietet Workload-Management-Funktionen, mit denen Sie Ihre wichtigsten interaktiven Workloads priorisieren, kontrollieren und skalieren können. Sie können jederzeit Kapazität hinzufügen, um die Anzahl der Abfragen zu erhöhen, die Sie gleichzeitig ausführen können, zu kontrollieren, welche Workloads die Kapazität nutzen können, und die Kapazität auf mehrere Workloads verteilen.

Weitere Informationen finden Sie unter Kapazität zur Abfrageverarbeitung verwalten. Informationen zur Preisgestaltung finden Sie auf der Preisseite von Amazon Athena.

17. April 2023

Veröffentlicht am 17.04.2023

Athena veröffentlicht JDBC-Treiberversion 2.0.36. Der Treiber enthält neue Features und hat ein Problem behoben.

Neue Features

- Sie können jetzt anpassbare Kennungen der vertrauenden Partei mit der AD-FS-Authentifizierung verwenden.

- Sie können jetzt den Namen der Anwendung, die den Konnektor verwendet, zur Zeichenfolge des Benutzeragenten hinzufügen.

Gelöste Probleme

- Es wurde ein Fehler behoben, der auftrat, wenn `getSchema()` verwendet wurde, um ein nicht vorhandenes Schema abzurufen.

Weitere Informationen und das Herunterladen des neuen Treibers, der Versionshinweise und der Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC.

14. April 2023

Veröffentlicht am 20.06.2023

Athena kündigt die folgenden Korrekturen und Verbesserungen an.

- Wenn Sie eine Zeichenfolge in einen Zeitstempel umwandeln, ist ein Leerzeichen zwischen dem Tag und der Uhrzeit oder Zeitzone erforderlich. Weitere Informationen finden Sie unter Bei der Umwandlung von einer Zeichenfolge in einen Zeitstempel ist Platz zwischen Datums- und Uhrzeitwerten erforderlich.
- Eine grundlegende Änderung in der Art und Weise, wie mit der Genauigkeit von Zeitstempeln umgegangen wurde, wurde entfernt. Um die Konsistenz zwischen Athena-Engine-Version 2 und Athena-Engine-Version 3 zu gewährleisten, ist die Zeitstempelgenauigkeit jetzt standardmäßig auf Millisekunden statt auf Mikrosekunden festgelegt.
- Athena erzwingt jetzt konsistent den Zugriff auf den Abfrageausgabe-Bucket, wenn Abfragen ausgeführt werden. Bitte stellen Sie sicher, dass alle IAM-Prinzipale, die die StartQueryExecutionAktion ausführen, über die S3: GetBucketLocation -Berechtigung für den Abfrageausgabe-Bucket verfügen.

4. April 2023

Veröffentlicht am 04.04.2023

Sie können jetzt Amazon Athena verwenden, um verbundene Abfragen für 10 neue Datenquellen auszuführen. Verwenden Sie eine einzelne Verbundansicht, um mehrere externe Tabellen oder Teilmengen von Daten abzufragen. Dies vereinfacht die erforderliche SQL-Anweisung und bietet

Ihnen die Flexibilität, Datenquellen von Endbenutzern zu verschleiern, die SQL zur Datenabfrage verwenden müssen.

Weitere Informationen finden Sie unter Arbeiten mit Ansichten und Verbundabfragen ausführen.

30. März 2023

Veröffentlicht am 30.03.2023

Amazon Athena kündigt die Verfügbarkeit von Amazon Athena für Apache Spark in zusätzlichen AWS-Regionen an.

Diese Version erweitert die Verfügbarkeit von Amazon Athena für Apache Spark um Asien-Pazifik (Mumbai), Asien-Pazifik (Singapur), Asien-Pazifik (Sydney) und Europa (Frankfurt).

Weitere Informationen zu Amazon Athena für Apache Spark finden Sie unterVerwenden von Apache Spark in Amazon Athena.

28. März 2023

Veröffentlicht am 28.03.2023

Athena kündigt die folgenden Korrekturen und Verbesserungen an.

- In den Antworten auf die API-Aktionen `GetQueryExecution` und die `BatchGetQueryExecution`-Athena-API-Aktionen zeigt das neue `subStatementType`-Feld den Typ der Abfrage an, die ausgeführt wurde (z. B. `SELECT`, `INSERT`, `UNLOAD`, `CREATE_TABLE` oder `CREATE_TABLE_AS_SELECT`).
- Es wurde ein Fehler behoben, durch den Manifestdateien für Apache-Hive-Schreiboperationen nicht korrekt verschlüsselt wurden.
- Athena-Engine-Version 3 verarbeitet `NaN`- und `Infinity`-Werte in der `approx_percentile`-Funktion jetzt korrekt. Die `approx_percentile`-Funktion gibt das ungefähre Perzentil für einen Datensatz zum angegebenen Prozentsatz zurück.

  Athena-Engine-Version 2 behandelt `NaN` fälschlicherweise einen Wert größer als `Infinity`. Die Athena-Engine-Version 3 behandelt `NaN` und `Infinity` nun entsprechend der Behandlung in anderen analytischen und statistischen Funktionen. In den folgenden Punkten wird das neue Verhalten ausführlicher beschrieben.

  - Wenn `NaN` im Datensatz vorhanden ist, gibt Athena `NaN` zurück.

- Wenn `NaN` nicht anwesend, aber `Infinity` präsent ist, behandelt Athena `Infinity` als eine sehr große Zahl.
- Wenn mehrere `Infinity`-Werte vorhanden sind, behandelt Athena sie als dieselbe sehr große Zahl. Falls erforderlich, gibt Athena `Infinity` aus.
- Wenn ein einzelner Datensatz sowohl `Infinity` als auch `-Double.MAX_VALUE` enthält und ein Perzentilergebnis `-Double.MAX_VALUE` ist, gibt Athena `-Infinity` zurück.
- Wenn ein einzelner Datensatz sowohl `Infinity` als auch `Double.MAX_VALUE` enthält und ein Perzentilergebnis `Double.MAX_VALUE` ist, gibt Athena `Infinity` zurück.
- Verwenden Sie die `is_finite()`-Funktion, um `Infinity` und `NaN` aus einer Berechnung auszuschließen, wie im folgenden Beispiel gezeigt.

```
approx_percentile(x, 0.5) FILTER (WHERE is_finite(x))
```

27. März 2023

Veröffentlicht am 27.03.2023

Sie können jetzt eine Mindestverschlüsselungsstufe für Athena-SQL-Arbeitsgruppen in Amazon Athena angeben. Dieses Feature stellt sicher, dass die Ergebnisse aller Abfragen in der Athena-SQL-Arbeitsgruppe auf der von Ihnen angegebenen Verschlüsselungsebene oder höher verschlüsselt werden. Sie können zwischen verschiedenen Verschlüsselungsstufen wählen, um Ihre Daten zu schützen. Um die gewünschte Mindestverschlüsselungsstufe zu konfigurieren, können Sie die Athena-Konsole AWS CLI, die API oder das SDK verwenden.

Das Mindestverschlüsselungs-Feature ist für Apache-Spark-fähige Arbeitsgruppen nicht verfügbar. Weitere Informationen finden Sie unter Konfiguration der Mindestverschlüsselung für eine Arbeitsgruppe.

17. März 2023

Veröffentlicht am 17.03.2023

Athena kündigt die folgenden Korrekturen und Verbesserungen an.

- Es wurde ein Problem mit dem Amazon Athena DynamoDB-Connector behoben, das dazu führte, dass Abfragen fehlschlugen und die Fehlermeldung nur eine Bedingung pro Schlüssel enthalten KeyConditionExpressions darf.

Dieses Problem tritt auf, weil Athena-Engine-Version 3 die Möglichkeit erkennt, mehr Arten von Prädikaten herunterzufahren als Athena-Engine-Version 2. In Athena-Engine-Version 3 werden Klauseln wie `some_column LIKE 'someprefix%` als Filterprädikate nach unten verschoben, die eine Unter- und Obergrenze auf eine bestimmte Spalte anwenden. Athena-Engine-Version 2 hat diese Prädikate nicht nach unten gedrückt. Wenn in der Athena-Engine-Version 3 die `some_column`-Spalte ein Sortierschlüssel ist, gibt die Engine das Filterprädikat an den DynamoDB-Konnektor weiter. Das Filterprädikat wird dann weiter nach unten an den DynamoDB-Service weitergegeben. Da DynamoDB nicht mehr als eine Filterbedingung für einen Sortierschlüssel unterstützt, gibt DynamoDB den Fehler zurück.

Um dieses Problem zu beheben, aktualisieren Sie Ihren Amazon-Athena-DynamoDB-Konnektor auf Version 2023.11.1. Anweisungen zum Aktualisieren des Konnektors finden Sie unter Schreiben eines Datenquellen-Konnektors.

08. März 2023

Veröffentlicht am 08.03.2023

Athena kündigt die folgenden Korrekturen und Verbesserungen an.

- Es wurde ein Problem mit Verbundabfragen behoben, das dazu führte, dass Zeitstempel-Prädikatwerte in Mikrosekunden statt in Millisekunden gesendet wurden.

15. Februar 2023

Veröffentlicht am 15.02.2023

Athena kündigt die folgenden Korrekturen und Verbesserungen an.

- Sie können jetzt die clientseitige Verschlüsselung verwenden, um Daten in Amazon S3 für Iceberg-Schreibvorgänge zu verschlüsseln.
- Es wurde ein Problem behoben, das die serverseitige Verschlüsselung in Amazon S3 für Iceberg-Schreibvorgänge beeinträchtigte.

31. Januar 2023

Veröffentlicht am 31.01.2023

Sie können jetzt Amazon Athena zum Abfragen von Daten in Google Cloud Storage verwenden. Wie Amazon S3 ist Google Cloud Storage ein verwalteter Service, der Daten in Buckets speichert. Verwenden Sie den Athena-Konnektor für Google Cloud Storage, um interaktive Verbundabfragen für Ihre externen Daten auszuführen.

Weitere Informationen finden Sie unter Google-Cloud-Storage-Konnektor für Amazon Athena.

20. Januar 2023

Veröffentlicht am 20.01.2023

Sie können jetzt eine erweiterte Dokumentation für die Unterstützung der Athena-Komprimierung anzeigen. Es wurden einzelne Themen für Komprimierung von Hive-Tabellen, Komprimierung von Iceberg-Tabellen und ZSTD-Komprimierungsstufen hinzugefügt.

Weitere Informationen finden Sie unter Athena-Komprimierungs-Support.

3. Januar 2023

Veröffentlicht am 03.01.2023

Athena kündigt die folgenden Aktualisierungen an:

- Zusätzliche Befehle für Hive-Metastores – Sie können Athena verwenden, um eine Verbindung zu Ihrem selbstverwalteten Apache Hive Metastore als Metadatenkatalog herzustellen und in Amazon S3 gespeicherte Daten abzufragen. Mit dieser Version können Sie `CREATE TABLE AS` (CTAS), `INSERT INTO` und 12 zusätzliche DDL-Befehle (Data Definition Language) verwenden, um mit dem Apache Hive Metastore zu interagieren. Mit diesen erweiterten SQL-Funktionen können Sie Ihre Hive-Metastore-Schemas direkt von Athena aus verwalten.

  Weitere Informationen finden Sie unter Verwenden von Athena-Daten-Connector für externen Hive-Metastore.
- JDBC-Treiberversion 2.0.35 – Athena veröffentlicht JDBC-Treiberversion 2.0.35. Der JDBC-Treiber 2.0.35 enthält die folgenden Aktualisierungen:
  - Der Treiber verwendet jetzt die folgenden Bibliotheken für den Jackson-JSON-Parser.
    - jackson-annotations 2.14.0 (zuvor 2.13.2)
    - jackson-core 2.14.0 (zuvor 2.13.2)
    - jackson-databind 2.14.0 (zuvor 2.13.2.2)
  - Die Unterstützung für JDBC-Version 4.1 wurde eingestellt.

Weitere Informationen und das Herunterladen des neuen Treibers, der Versionshinweise und der Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC.

Athena-Versionshinweise für 2022

14. Dezember 2022

Veröffentlicht am 14.12.2022

Sie können jetzt den Amazon-Athena-Konnektor für Kafka verwenden, um SQL-Abfragen für Streaming-Daten auszuführen. Sie können beispielsweise analytische Abfragen für Echtzeit-Streaming-Daten in Amazon Managed Streaming für Apache Kafka (Amazon MSK) ausführen und diese mit historischen Daten in Ihrem Data Lake in Amazon S3 verknüpfen.

Der Amazon-Athena-Konnektor für Kafka unterstützt Abfragen auf mehreren Streaming-Engines. Sie können Athena verwenden, um SQL-Abfragen auf von Amazon MSK bereitgestellten und Serverless-Clustern, auf selbstverwalteten Kafka-Bereitstellungen und auf Streaming-Daten in Confluent Cloud auszuführen.

Weitere Informationen finden Sie unter Amazon-Athena-MSK-Konnektor.

2. Dezember 2022

Veröffentlicht am 02.12.2022

Athena veröffentlicht JDBC-Treiberversion 2.0.34. Der JDBC-Treiber 2.0.34 enthält die folgenden neuen Features und behobenen Probleme:

- Unterstützung für die Wiederverwendung von Abfrageergebnissen – Sie können jetzt die Ergebnisse zuvor ausgeführter Abfragen bis zu einem von Ihnen festgelegten Zeitlimit wiederverwenden, anstatt Athena die Ergebnisse jedes Mal neu berechnen zu lassen, wenn die Abfrage ausgeführt wird. Weitere Informationen finden Sie im Installations- und Konfigurationshandbuch (das auf der JDBC-Downloadseite verfügbar ist) und unter Wiederverwenden von Abfrageergebnissen.
- InstanceMetadata Ec2-Unterstützung — Der JDBC-Treiber unterstützt jetzt die InstanceMetadataEc2-Authentifizierungsmethode mithilfe von IAM-Instanzprofilen.
- Behebung zeichenbasierter Ausnahmen – Eine Ausnahme, die bei Abfragen auftrat, die bestimmte Sprachzeichen enthielten, wurde behoben.

- Behebung einer Sicherheitslücke — Es wurde eine Sicherheitslücke im Zusammenhang mit AWS Abhängigkeiten behoben, die im Konnektor enthalten sind.

Weitere Informationen und das Herunterladen des neuen Treibers, der Versionshinweise und der Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC.

30. November 2022

Veröffentlicht am 30.11.2022

Sie können jetzt interaktiv Apache-Spark-Anwendungen und Jupyter-kompatible Notebooks auf Athena erstellen und ausführen. Führen Sie Datenanalysen auf Athena mit Spark durch, ohne Ressourcen einplanen, konfigurieren oder verwalten zu müssen. Senden Sie den Spark-Code zur Verarbeitung und erhalten Sie die Ergebnisse unmittelbar. Verwenden Sie die vereinfachte Notebook-Erfahrung in der Amazon-Athena-Konsole, um Apache Spark-Anwendungen mit Python oder Athena-Notebooks-APIs zu entwickeln.

Apache Spark auf Amazon Athena ist Serverless und bietet eine automatische, bedarfsgerechte Skalierung, die sofortige Rechenleistung für sich ändernde Daten-Volumes und Verarbeitungsanforderungen bereitstellt.

Weitere Informationen finden Sie unter Verwenden von Apache Spark in Amazon Athena.

18. November 2022

Veröffentlicht am 18.11.2022

Sie können jetzt den Amazon Athena-Konnektor für IBM Db2 verwenden, um Db2 von Athena abzufragen. Sie können beispielsweise analytische Abfragen über ein Data Warehouse in Db2 und einen Data Lake in Amazon S3 ausführen.

Der Amazon-Athena-Db2-Konnektor stellt mehrere Konfigurationsoptionen durch Lambda-Umgebungsvariablen bereit. Informationen zu Konfigurationsoptionen, Parametern, Verbindungszeichenfolgen, Bereitstellung und Einschränkungen finden Sie unter Amazon-Athena-IBM-Db2-Konnektor.

17. November 2022

Veröffentlicht am 17. November 2022

Die Apache-Iceberg-Unterstützung in der Athena-Engine-Version 3 bietet jetzt die folgenden erweiterten ACID-Transaktionsfeatures:

- ORC- und Avro-Unterstützung – Erstellen Sie Iceberg-Tabellen mit den zeilen- und spaltenbasierten Dateiformaten Apache Avro und Apache ORC. Die Unterstützung für diese Formate erfolgt zusätzlich zur vorhandenen Unterstützung für Parquet.
- MERGE INTO – Verwenden Sie den `MERGE INTO`-Befehl, um Daten in großem Umfang effizient zusammenzuführen. `MERGE INTO` kombiniert die Vorgänge `INSERT`, `UPDATE` und `DELETE` zu einer Transaktion. Dies reduziert den Verarbeitungsaufwand in Ihrer Datenpipeline und benötigt weniger SQL zum Schreiben. Weitere Informationen finden Sie unter Aktualisieren von Iceberg-Datentabellen und MERGE INTO.
- CTAS- und VIEW-Unterstützung – Verwenden Sie `CREATE TABLE AS SELECT`-(CTAS) und `CREATE VIEW`-Anweisungen mit Iceberg-Tabellen. Weitere Informationen finden Sie unter CREATE TABLE AS und CREATE VIEW.
- VACUUM-Unterstützung – Mit der `VACUUM`-Anweisung können Sie Ihren Data Lake optimieren, indem Sie nicht mehr benötigte Snapshots und Daten löschen. Sie können dieses Feature verwenden, um die Leseleistung zu verbessern und gesetzliche Anforderungen wie die DSGVO zu erfüllen. Weitere Informationen finden Sie unter Optimieren von Iceberg-Tabellen und VACUUM.

Diese neuen Features erfordern die Athena-Engine-Version 3 und sind in allen Regionen verfügbar, in denen Athena unterstützt wird. Sie können diese mit der Athena-Konsole, den Treibern oder der API verwenden.

Informationen zur Verwendung von Iceberg in Athena finden Sie unter Apache-Iceberg-Tabellen verwenden.

14. November 2022

Veröffentlicht am 14.11.2022

Amazon Athena unterstützt nun IPv6-Endpunkte für eingehende Verbindungen, die Sie zum Aufrufen von Athena-Funktionen über IPv6 verwenden können. Sie können dieses Feature verwenden, um IPv6-Compliance-Anforderungen zu erfüllen. Außerdem entfällt der Bedarf an zusätzlicher Netzwerkausrüstung für die Adressübersetzung zwischen IPv4 und IPv6.

Um dieses Feature zu verwenden, konfigurieren Sie Ihre Anwendungen so, dass sie die neuen Athena-Dual-Stack-Endpunkte verwenden, die sowohl IPv4 als auch IPv6 unterstützen. Dual-Stack-

Endpunkte verwenden das Format `athena.region.api.aws`. Der Dual-Stack-Endpunkt in der Region USA Ost (Nord-Virginia) ist beispielsweise `athena.us-east-1.api.aws`.

Wenn Sie eine Anfrage an einen Dual-Stack Athena-Endpunkt stellen, löst der Endpunkt eine IPv6- oder eine IPv4-Adresse auf, je nachdem, welches Protokoll Ihr Netzwerk und Ihr Client verwendet. Um programmgesteuert eine Verbindung zu einem AWS Dienst herzustellen, können Sie das AWS CLIoder AWS SDK verwenden, um einen Endpunkt anzugeben.

Weitere Informationen über Service-Endpunkte finden Sie unter AWS -Service-Endpunkte. Weitere Informationen zu den Service-Endpunkten von Athena finden Sie in der AWS -Dokumentation unter Amazon-Athena-Endpunkte und Kontingente.

Sie können die neuen Athena-Dual-Stack-Endpunkte ohne zusätzliche Kosten für eingehende Verbindungen verwenden. Dual-Stack-Endpunkte sind generell in allen AWS-Regionen verfügbar.

11. November 2022

Veröffentlicht am 11.11.2022

Athena kündigt die folgenden Korrekturen und Verbesserungen an.

- Erweiterte differenzierte Zugriffskontrolle in Lake Formation – Sie können jetzt AWS Lake Formation-differenzierte Zugriffskontrollrichtlinien in Athena-Abfragen für Daten verwenden, die in jedem unterstützten Datei- oder Tabellenformat gespeichert sind. Sie können eine differenzierte Zugriffskontrolle in Lake Formation verwenden, um den Zugriff auf Daten in Abfrageergebnissen mit Datenfiltern einzuschränken, um Sicherheit auf Spalten-, Zeilen- und Zellenebene zu erreichen. Zu den unterstützten Tabellenformaten in Athena gehören Apache Iceberg, Apache Hudi und Apache Hive. Eine erweiterte, differenzierte Zugriffskontrolle ist in allen von Athena unterstützten Regionen verfügbar. Die erweiterte Tabellen- und Dateiformatunterstützung erfordert Athena-Engine-Version 3, das neue Features und eine verbesserte Abfrageleistung bietet. Es ändert jedoch nichts daran, wie Sie differenzierte Zugriffskontrollrichtlinien in Lake Formation einrichten.

  Bei der Verwendung dieser erweiterten, differenzierten Zugriffskontrolle in Athena sind die folgenden Überlegungen zu berücksichtigen:

  - ERLÄUTERN – In Lake Formation definierte Zeilen- oder Zellenfilterinformationen und Statistikinformationen für Abfragen werden in der Ausgabe von `EXPLAIN` und `EXPLAIN ANALYZE` nicht angezeigt. Informationen zu `EXPLAIN` in Athena finden Sie unter Verwenden von EXPLAIN und EXPLAIN ANALYZE in Athena.

- Externe Hive-Metastores – Ausgeblendete Apache-Hive-Spalten können nicht für eine differenzierte Filterung der Zugriffskontrolle verwendet werden, und versteckte Apache-Hive-Systemtabellen werden von der detaillierten Zugriffskontrolle nicht unterstützt. Weitere Informationen finden Sie unter Überlegungen und Einschränkungen im Thema Verwenden von Athena-Daten-Connector für externen Hive-Metastore.
- Abfragestatistiken – Die Zeilenebenen-Informationen über die Anzahl der Eingabe- und Ausgabezeilen und die Datengröße werden in den Athena-Abfragestatistiken nicht angezeigt, wenn für eine Abfrage in Lake Formation Filter auf Zeilenebene definiert wurden. Hinweise zum Anzeigen von Statistiken für Athena-Abfragen finden Sie unter Anzeigen von Statistiken und Ausführungsdetails für abgeschlossene Abfragen und GetQueryRuntimeStatistics.
- Arbeitsgruppen – Benutzer in derselben Athena-Arbeitsgruppe können die Daten anzeigen, die die differenzierte Zugriffskontrolle von Lake Formation so konfiguriert hat, dass sie für die Arbeitsgruppe zugänglich sind. Informationen zur Verwendung von Athena zum Abfragen von bei Lake Formation registrierten Daten finden Sie unter Verwenden von Athena zum Abfragen von Daten, die in AWS Lake Formationregistriert sind.

  Informationen zur Verwendung der differenzierten Zugriffskontrolle in Lake Formation finden Sie unter Verwaltung der differenzierten Zugriffskontrolle mithilfe von AWS Lake Formation im AWS - Big-Data-Blog.

- Athena-Verbundabfrage – Athena-Verbundabfragen behalten jetzt die ursprüngliche Groß- und Kleinschreibung von Feldnamen in `struct`-Objekten bei. Bisher wurden `struct`-Feldnamen automatisch in Kleinbuchstaben geschrieben.

8. November 2022

Veröffentlicht am 11.08.2022

Sie können jetzt das Feature zur Wiederverwendung von Abfrageergebnissen nutzen, um wiederholte Abfragen in Athena zu beschleunigen. Eine wiederholte Abfrage ist eine SQL-Abfrage, die mit einer kürzlich gestellten Abfrage identisch ist und dieselben Ergebnisse liefert. Wenn Sie mehrere identische Abfragen ausführen müssen, kann das Zwischenspeichern der Wiederverwendung von Ergebnissen die zum Erzeugen von Ergebnissen erforderliche Zeit verkürzen. Das Zwischenspeichern von Ergebnissen senkt auch die Kosten, indem es die Anzahl der gescannten Bytes reduziert.

Weitere Informationen finden Sie unter Wiederverwenden von Abfrageergebnissen.

13. Oktober 2022

Veröffentlicht am 13.10.2022

Athena kündigt Athena-Engine-Version 3 an.

Athena hat seine SQL-Abfrage-Engine aktualisiert, um die neuesten Features des Open-Source-Projekts Trino einzuschließen. Neben der Unterstützung aller Features der Athena-Engine-Version 2 bietet die Athena-Engine-Version 3 über 50 neue SQL-Features, 30 neue Features und mehr als 90 Verbesserungen der Abfrageleistung. Mit dem heutigen Start führt Athena auch einen kontinuierlichen Integrationsansatz für das Open-Source-Softwaremanagement ein, der die Aktualität mit den Projekten Trino und Presto verbessert, sodass Sie schnelleren Zugriff auf Community-Verbesserungen erhalten, die in die Athena-Engine integriert und optimiert sind.

Weitere Informationen finden Sie unter Athena-Engine-Version 3.

10. Oktober 2022

Veröffentlicht am 10.10.2022

Athena veröffentlicht JDBC-Treiberversion 2.0.33. Der JDBC-Treiber 2.0.33 enthält die folgenden Änderungen:

- Neue Eigenschaften für Treiberversion, JDBC-Version und Plugin-Name wurden der Benutzer-Kundendienstmitarbeiter-Zeichenfolge in der Klasse des Anmeldeinformationsanbieters hinzugefügt.
- Fehlermeldungen wurden korrigiert und notwendige Informationen hinzugefügt.
- Vorbereitete Anweisungen werden jetzt freigegeben, wenn die Verbindung geschlossen wird oder die Ausführung der vorbereiteten Athena-Anweisung fehlschlägt.

Weitere Informationen und das Herunterladen des neuen Treibers, der Versionshinweise und der Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC.

23. September 2022

Veröffentlicht am 26.09.2022

Der Neptune-Konnektor von Amazon Athena unterstützt jetzt den Abgleich von Spalten- und Tabellennamen ohne Berücksichtigung der Groß- und Kleinschreibung.

- Der Neptune-Datenquellenkonnektor kann Spaltennamen in Neptune-Tabellen, die Groß- und Kleinschreibung verwenden, auflösen, auch wenn die Spaltennamen in der Tabelle in AWS Glue alle in Kleinbuchstaben geschrieben sind. Um dieses Verhalten zu aktivieren, setzen Sie die Umgebungsvariable `enable_caseinsensitivematch` in der Lambda-Funktion des Neptune-Konnektors auf `true`.
- Da nur Tabellennamen in Kleinbuchstaben AWS Glue unterstützt werden, geben Sie beim Erstellen einer AWS Glue Tabelle für Neptune den AWS Glue Tabellenparameter an. `"glabel" = table_name`

Weitere Informationen zum Neptune-Konnektor finden Sie unter Amazon Athena Neptune Konnektor.

13. September 2022

Veröffentlicht am 13.09.2022

Athena kündigt die folgenden Korrekturen und Verbesserungen an.

- External Hive metastore – Athena gibt jetzt `NULL` aus, anstatt eine Ausnahme zu erzeugen, wenn eine `WHERE`-Klausel eine Partition enthält, die nicht in einem externer Hive-Metastores (EHMS) existiert. Das neue Verhalten entspricht dem von AWS Glue Data Catalog.
- Parameterized queries – Werte in Parametrisierte Abfragen können jetzt auf den `DOUBLE`-Datentyp übertragen werden.
- Apache Iceberg – Schreibvorgänge in Iceberg-Tabellen sind jetzt erfolgreich, wenn Object Lock in einem Amazon-S3-Bucket aktiviert ist.

31. August 2022

Veröffentlicht am 31.08.2022

Amazon Athena gibt die Verfügbarkeit von Athena und seinen Features in der Region Asien-Pazifik (Jakarta) bekannt.

Diese Version erweitert die Verfügbarkeit von Athena in Asien-Pazifik (Hongkong), Asien-Pazifik (Jakarta), Asien-Pazifik (Mumbai), Asien-Pazifik (Osaka), Asien-Pazifik (Seoul), Asien-Pazifik (Singapur), Asien-Pazifik (Sydney) und Asien-Pazifik (Tokio). Eine vollständige Liste der in diesen und anderen Regionen verfügbaren AWS-Services finden Sie in der Liste der al AWS-Region-Services.

23. August 2022

Veröffentlicht am 23.08.2022

Release v2022.32.1 des Athena Query Federation SDKs enthält die folgenden Änderungen:

- Der Amazon Athena-Datenquellen-Konnektor für Oracle unterstützt jetzt SSL-basierte Verbindungen zu Amazon RDS-Instances. Die Unterstützung ist auf das Transport Layer Security (TLS)-Protokoll und auf die Authentifizierung des Servers durch den Client beschränkt. Weil die gegenseitige Authentifizierung in Amazon RDS nicht unterstützt wird, umfasst das Update keine Unterstützung für die gegenseitige Authentifizierung.

  Weitere Informationen finden Sie unter Amazon Athena Oracle Konnektor.

03. August 2022

Veröffentlicht am 03.08.2022

Athena veröffentlicht JDBC-Treiberversion 2.0.32. Der JDBC-Treiber 2.0.32 enthält die folgenden Änderungen:

- Die an das Athena SDK gesendete `User-Agent`-Zeichenfolge wurde um die Treiberversion, die JDBC-Spezifikationsversion und den Namen des Authentifizierungs-Plug-Ins erweitert.
- Eine `NullPointerException` wurde behoben, die ausgelöst wurde, wenn für den `CheckNonProxyHost`-Parameter kein Wert angegeben wurde.
- Ein Problem beim `login_url` Parsen im Authentifizierungs-Plugin wurde behoben. BrowserSaml
- Ein Proxy-Host-Problem wurde behoben, das auftrat, wenn der `UseProxyforIdp`-Parameter auf `true` gesetzt wurde.

Weitere Informationen und das Herunterladen des neuen Treibers, der Versionshinweise und der Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC.

1. August 2022

Veröffentlicht am 01.08.2022

Athena kündigt Verbesserungen am Athena Query Federation SDK und den von Athena vorgefertigten Datenquellkonnektoren an. Es wurden u. a. folgende -Verbesserungen vorgenommen:

- Struktur-Analyse – Ein Problem mit `GlueFieldLexer` Parsing im Athena Query Federation SDK, das verhinderte, dass bestimmte komplizierte Strukturen alle ihre Daten anzeigen konnten, wurde behoben. Dieses Problem betraf Konnektoren im Athena Query Federation SDK.
- AWS Glue Tabellen — Zusätzliche Unterstützung für die `decimal` Spaltentypen `set` und in AWS Glue Tabellen hinzugefügt.
- DynamoDB-Konnektor – Möglichkeit hinzugefügt, Groß-/Kleinschreibung bei DynamoDB-Attributnamen zu ignorieren. Weitere Informationen finden Sie unter `disable_projection_and_casing` im Parameter-Abschnitt auf der Amazon Athena DynamoDB Konnektor-Seite.

Weitere Informationen finden Sie in Version v2022.30.2 von Athena Query Federation am. GitHub

21. Juli 2022

Veröffentlicht am 21.07.2022

Sie können Ihre Abfragen jetzt mithilfe von Leistungsmetriken und interaktiven, visuellen Abfrageanalyse-Tools in der Athena-Konsole analysieren und debuggen. Die Daten zur Abfrageleistung und zu den Ausführungsdetails können Ihnen helfen, Engpässe in Abfragen zu identifizieren, die Operatoren und Statistiken für jede Phase einer Abfrage zu überprüfen, das Datenvolumen zwischen den Phasen zu verfolgen und die Auswirkungen von Abfrageprädikaten zu überprüfen. Sie können jetzt:

- Greifen Sie mit einem einzigen Klick auf den verteilten und logischen Ausführungsplan für Ihre Abfrage zu.
- Erkunden Sie die Abläufe in jeder Phase, bevor die Phase ausgeführt wird.
- Visualisieren Sie die Leistung abgeschlossener Abfragen mit Metriken für die Zeit, die sie in der Warteschlange-, Planungs- und Ausführungsphase verbracht haben.
- Informieren Sie sich über die Anzahl der Zeilen und die Menge der von Ihrer Abfrage verarbeiteten und ausgegebenen Quelldaten.
- Sehen Sie sich detaillierte Ausführungsdetails für Ihre Abfragen an, die im Kontext dargestellt und als interaktives Diagramm formatiert werden.
- Verwenden Sie genaue Ausführungsdetails auf Stufenebene, um den Datenfluss durch Ihre Abfrage zu verstehen.
- Analysieren Sie Abfrage-Leistungsdaten programmgesteuert mithilfe neuer APIs, um Abfragelaufzeitstatistiken zu erhalten, ebenfalls heute veröffentlicht.

Um zu erfahren, wie Sie diese Funktionen für Ihre Abfragen verwenden können, schauen Sie sich das Video-Tutorial Optimieren von Amazon Athena Athena-Abfragen mit neuen Abfrageanalysetools auf dem AWS YouTube Kanal an.

Eine Dokumentation finden Sie unter Anzeigen von Ausführungsplänen für SQL-Abfragen und Anzeigen von Statistiken und Ausführungsdetails für abgeschlossene Abfragen.

11. Juli 2022

Veröffentlicht am 11.07.2022.

Sie können jetzt parametrisierte Abfragen direkt von der Athena-Konsole oder API aus ausführen, ohne zuvor SQL-Anweisungen vorzubereiten.

Wenn Sie in der Athena-Konsole Abfragen ausführen, die Parameter in Form von Fragezeichen haben, werden Sie nun von der Benutzeroberfläche aufgefordert, Werte für die Parameter direkt einzugeben. Dadurch entfällt die Notwendigkeit, Literalwerte im Abfrage-Editor jedes Mal zu ändern, wenn Sie die Abfrage ausführen möchten.

Wenn Sie die erweiterte Abfrageausführungs-API verwenden, können Sie jetzt die Ausführungsparameter und ihre Werte in einem einzigen Aufruf bereitstellen.

Weitere Informationen finden Sie unter Verwenden von parametrisierten Abfragen in diesem Benutzerhandbuch und in dem Post Use Amazon Athena parameterized queries to provide data as a service des AWS Big Data Blog.

8. Juli 2022

Veröffentlicht am 08.07.2022

Athena kündigt die folgenden Korrekturen und Verbesserungen an.

- Es wurde ein Problem mit der Behandlung von `DATE` Spaltenkonvertierungen für SageMaker Endpunkte (UDF) behoben, das zu Abfragefehlern führte.

6. Juni 2022

Veröffentlicht am 06.06.2022

Athena veröffentlicht JDBC-Treiberversion 2.0.31. Der JDBC-Treiber 2.0.31 enthält die folgenden Änderungen:

- Log4j-Abhängigkeitsproblem – Die Fehlermeldung Cannot find driver class (Treiberklasse kann nicht gefunden werden), die durch eine Log4j-Abhängigkeit verursacht wird, wurde behoben.

Weitere Informationen und das Herunterladen des neuen Treibers, der Versionshinweise und der Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC.

25. Mai 2022

Veröffentlicht am 25.05.2022

Athena kündigt die folgenden Korrekturen und Verbesserungen an.

- Iceberg-Support
  - Einführung von Support für regionsübergreifende Abfragen Jetzt können Sie Iceberg-Tabellen in einer AWS-Region anderen Version abfragen als der AWS-Region , die Sie verwenden.
  - Einführung der Unterstützung für serverseitige Verschlüsselungskonfiguration. Jetzt können Sie SSE-S3/SSE-KMS verwenden, um Daten aus Iceberg-Schreibvorgängen in Amazon S3 zu verschlüsseln.

  Weitere Informationen zur Verwendung von Apache Iceberg in Athena finden Sie unter Apache-Iceberg-Tabellen verwenden.
- JDBC-2.0.30-Treiber-Release

  Der JDBC-2.0.30-Treiber für Athena hat die folgenden Verbesserungen:
  - Behebt ein Data-Race-Problem, das sich auf parametrisierte vorbereitete Anweisungen auswirkte.
  - Behebt ein Problem beim Starten der Anwendung, das in Gradle-entwickelten Umgebungen aufgetreten ist.

  Informationen zum Herunterladen des JDBC-2.0.30-Treiber, Versionshinweise und Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC.

6. Mai 2022

Veröffentlicht am 06.05.2022

Veröffentlicht die JDBC-2.0.29- und ODBC-1.1.17-Treiber für Athena.

Diese Treiber umfassen die folgenden Änderungen:

- Der Startvorgang des SAML-Plug-In-Browsers wurde aktualisiert.

Weitere Informationen zu diesen Änderungen sowie zum Herunterladen der neuen Treiber, Versionshinweise und Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC und Herstellen einer Verbindung mit Amazon Athena mit ODBC.

22. April 2022

Veröffentlicht am 22.04.2022

Athena kündigt die folgenden Korrekturen und Verbesserungen an.

- Es wurde ein Problem in den Partitionsindizes und dem Filterfeature mit dem Partitions-Cache behoben, das auftrat, wenn die folgenden Bedingungen erfüllt waren:
  - Der `partition_filtering.enabled` Schlüssel wurde `true` in den AWS Glue Tabelleneigenschaften für eine Tabelle auf festgelegt.
  - Die gleiche Tabelle wurde mehrfach mit unterschiedlichen Partitions-Filterwerten verwendet.

21. April 2022

Veröffentlicht am 21.04.2022

Sie können jetzt Amazon Athena verwenden, um Verbundabfragen für neue Datenquellen wie Google BigQuery, Azure Synapse und Snowflake auszuführen. Neue Datenquellen-Konnektors umfassen:

- Azure Data Lake Storage (ADLS) Gen2
- Azure Synapse
- Cloudera Hive
- Cloudera Impala
- Google BigQuery
- Hortonworks
- Microsoft SQL Server
- Oracle

- SAP HANA (Express-Edition)
- Snowflake
- Teradata

Eine vollständige Liste der von Athena unterstützten Datenquellen finden Sie unter Verfügbare Datenquellenkonnektoren.

Um das Durchsuchen der verfügbaren Quellen und das Herstellen einer Verbindung zu Ihren Daten zu vereinfachen, können Sie jetzt die verfügbaren Konnektors über einen aktualisierten Bildschirm Data Sources (Datenquellen) in der Athena-Konsole suchen, sortieren und filtern.

Weitere Informationen zum Abfragen von föderierten Quellen finden Sie unter Nutzung von Amazon-Athena-Verbundabfrage und Verbundabfragen ausführen.

13. April 2022

Veröffentlicht am 13.04.2022

Athena veröffentlicht JDBC-Treiberversion 2.0.28. Der JDBC-2.0.28-Treiber enthält die folgenden Änderungen:

- JWT-Unterstützung – Der Treiber unterstützt jetzt JSON-Web-Tokens (JSON-Web-Tokens) für die Authentifizierung. Informationen zur Verwendung von JWT mit dem JDBC-Treiber finden Sie im Installations- und Konfigurationsleitfaden, der von der JDBC-Treiberseite heruntergeladen werden kann.
- Log4j-Bibliotheken wurden aktualisiert – Der JDBC-Treiber verwendet jetzt die folgenden Log4J-Bibliotheken:
  - Log4J-API 2.17.1 (zuvor 2.17.0)
  - Log4J-API 2.17.1 (zuvor 2.17.0)
  - Log4j-JCl 2.17.2
- Weitere Verbesserungen – Der neue Treiber enthält auch die folgenden Verbesserungen und Bugfixes:
  - Das Feature für vorbereitete Anweisungen von Athena ist jetzt über JDBC verfügbar. Weitere Informationen zu vorbereiteten Anweisungen finden Sie unter Verwenden von parametrisierten Abfragen.
  - Der Athena-JDBC-SAML-Verband ist jetzt für die China-Regionen funktionsfähig.

- Weitere kleinere Verbesserungen.

Weitere Informationen und das Herunterladen des neuen Treibers, der Versionshinweise und der Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC.

30. März 2022

Veröffentlicht am 30.03.2022

Athena kündigt die folgenden Korrekturen und Verbesserungen an.

- Regionsübergreifende Abfragen — Sie können Athena jetzt verwenden, um Daten abzufragen, die sich in einem Amazon S3 S3-Bucket befinden, AWS-Regionen einschließlich Asien-Pazifik (Hongkong), Naher Osten (Bahrain), Afrika (Kapstadt) und Europa (Mailand).
  - Eine Liste der Länder, AWS-Regionen in denen Athena verfügbar ist, finden Sie unter Amazon Athena Athena-Endpunkte und Kontingente.
  - Informationen zur Aktivierung einer AWS-Region , die standardmäßig deaktiviert ist, finden Sie unter Region aktivieren.
  - Informationen zur Abfrage zwischen Regionen finden Sie unter Abfragen über Regionen hinweg.

18. März 2022

Veröffentlicht am 18.03.2022

Athena kündigt die folgenden Korrekturen und Verbesserungen an.

- Dynamic filtern–Dynamic filtern wurde für ganzzahlige Spalten verbessert, indem der Filter effizient auf jeden Datensatz einer entsprechenden Tabelle angewendet wurde.
- Iceberg – Es wurde ein Problem behoben, das beim Schreiben von Iceberg Parquet-Dateien mit mehr als 2 GB Fehler verursachte.
- Unkomprimierte Ausgabe – CREATE TABLE-Anweisungen unterstützen jetzt das Schreiben unkomprimierter Dateien. Verwenden Sie die folgende Syntax, um unkomprimierte Dateien zu schreiben:
  - CREATE TABLE (Textdatei oder JSON) — In `TBLPROPERTIES`, spezifizieren `write.compression = NONE`.
  - CREATE TABLE (Parquet) — In `TBLPROPERTIES`, spezifizieren `parquet.compression = UNCOMPRESSED`.

  - CREATE TABLE (ORC) — In `TBLPROPERTIES`, spezifizieren `orc.compress = NONE`.
- Komprimierung — Es wurde ein Problem mit Einfügungen für Textdateitabellen behoben, bei denen Dateien in einem Format komprimiert wurden, aber eine andere Erweiterung des Komprimierungsformats verwendeten, wenn nicht standardmäßige Komprimierungsmethoden verwendet wurden.
- Avro — Es wurden Probleme behoben, die beim Lesen von Dezimalzahlen des festen Typs aus Avro-Dateien auftraten.

2. März 2022

Veröffentlicht am 02.03.2022

Athena kündigt die folgenden Features und Verbesserungen an.

- Sie können dem Eigentümer des Amazon-S3-Bucket die volle Kontrolle über Abfrageergebnisse gewähren, wenn für den Speicherort des Abfrageergebnis-Bucket ACLs aktiviert sind. Weitere Informationen finden Sie unter Angeben eines Speicherorts des Abfrageergebnisses.
- Sie können jetzt vorhandene benannte Abfragen aktualisieren. Weitere Informationen finden Sie unter Verwenden von gespeicherten Abfragen.

23. Februar 2022

Veröffentlicht am 23.02.2022

Athena kündigt die folgenden Korrekturen und Leistungsverbesserungen an.

- Verbesserungen bei der Speicherverwaltung, um die Leistung zu verbessern und Speicherfehler zu reduzieren.
- Athena liest jetzt ORC-Zeitstempelspalten mit Zeitzoneninformationen, die in Stripe-Fußzeilen gespeichert sind, und schreibt ORC-Dateien mit Zeitzone (UTC) in Fußzeilen. Dies wirkt sich nur auf das Verhalten des ORC-Zeitstempel-Lesens aus, wenn die zu lesende ORC-Datei in einer Nicht-UTC-Zeitzonenumgebung erstellt wurde.
- Falsche Schätzungen der Symlink-Tabellengröße, die zu suboptimalen Abfrageplänen führten, wurden behoben.
- Seitliche Explosionsansichten können jetzt in der Athena-Konsole aus Hive-Metastore-Datenquellen abgefragt werden.

- Amazon-S3-Lesefehlermeldungen wurden verbessert, um detailliertere Amazon-S3-Fehlercode-Informationen einzubeziehen.
- Es wurde ein Problem behoben, durch das Ausgabedateien im ORC-Format mit Apache Hive 3.1 nicht kompatibel waren.
- Es wurde ein Problem behoben, das dazu führte, dass Tabellennamen mit Anführungszeichen in bestimmten DML- und DDL-Abfragen fehlschlugen.

15. Februar 2022

Veröffentlicht am 15.02.2022

Amazon Athena hat das Kontingent für aktive DML-Abfragen in allen AWS Regionen erhöht. Zu den aktiven Abfragen gehören sowohl laufende Abfragen als auch Abfragen in der Warteschlange. Mit dieser Änderung können Sie jetzt mehr DML-Abfragen in einem aktiven Status haben als zuvor.

Informationen zu Athena-Servicekontingenten finden Sie unter Service Quotas. Informationen zu den Abfragekontingenten in der Region, in der Sie Athena verwenden, finden Sie unter Endpunkte und Kontingente von Amazon Athena in der Allgemeine AWS-Referenz.

Um Ihre Kontingentnutzung zu überwachen, können Sie Nutzungsmetriken verwenden CloudWatch . Athena veröffentlicht die `ActiveQueryCount`-Metrik im `AWS/Usage`-Namespace. Weitere Informationen finden Sie unter Überwachung der Athena-Nutzungsmetriken.

Nachdem Sie Ihre Nutzung überprüft haben, können Sie die Service-Quotas-Konsole verwenden, um eine Kontingenterhöhung zu beantragen. Wenn Sie zuvor eine Kontingenterhöhung für Ihr Konto beantragt haben, gilt Ihr angefordertes Kontingent weiterhin, wenn es das neue standardmäßige aktive DML-Abfragekontingent überschreitet. Andernfalls verwenden alle Konten den neuen Standardwert.

14. Februar 2022

Veröffentlicht am 14.02.2022

Diese Version fügt das `ErrorType` Unterfeld zum AthenaErrorAntwortobjekt in der GetQueryExecutionAthena-API-Aktion hinzu.

Während das vorhandene `ErrorCategory`-Feld die allgemeine Quelle einer fehlgeschlagenen Abfrage (System, Benutzer oder andere) angibt, enthält das neue `ErrorType`-Feld detailliertere

Informationen zu dem aufgetretenen Fehler. Kombinieren Sie die Informationen aus beiden Feldern, um einen Einblick in die Ursachen des Abfragefehlers zu erhalten.

Weitere Informationen finden Sie unter Athena-Fehlerkatalog.

9. Februar 2022

Veröffentlicht am 9.2.2022

Die alte Athena-Konsole ist nicht mehr verfügbar. Die neue Konsole von Athena unterstützt alle Features der früheren Konsole, verfügt jedoch über eine benutzerfreundlichere, moderne Benutzeroberfläche und enthält neue Features, die die Erfahrung beim Entwickeln von Abfragen, der Analyse von Daten und der Verwaltung Ihrer Nutzung verbessern. Um die neue Athena-Konsole zu verwenden, besuchen Sie https://console.aws.amazon.com/athena/.

8. Februar 2022

Veröffentlicht am 8.2.2022

Erwarteter Bucket-Besitzer — Als zusätzliche Sicherheitsmaßnahme können Sie jetzt optional die AWS-Konto ID angeben, von der Sie erwarten, dass sie der Besitzer Ihres Buckets für die Ausgabe der Abfrageergebnisse in Athena sein wird. Wenn die Konto-ID des Bucket-Eigentümers der Abfrageergebnisse nicht mit der von Ihnen angegebenen Konto-ID übereinstimmt, schlagen Versuche, in den Bucket auszugeben, mit einem Amazon-S3-Berechtigungsfehler fehl. Sie können diese Einstellung auf Client- oder Arbeitsgruppenebene vornehmen.

Weitere Informationen finden Sie unter Angeben eines Speicherorts des Abfrageergebnisses.

28. Januar 2022

Veröffentlicht am 28.01.2022

Athena kündigt die folgenden Verbesserungen der Engine-Features an.

- Apache Hudi – Snapshot-Abfragen zu Hudi-Merge-on-Read(MoR)-Tabellen können jetzt Zeitstempelspalten lesen, die den `INT64`-Datentyp haben.
- UNION-Abfragen – Leistungsverbesserung und Datenscanreduzierung für bestimmte `UNION`-Abfragen, die dieselbe Tabelle mehrmals scannen.
- Disjunkte Abfragen – Leistungsverbesserung für Abfragen, die nur disjunkte Werte für jede Partitionsspalte im Filter aufweisen.

- Verbesserungen bei Partitionsprojektionen
  - Mehrere disjunkte Werte sind jetzt unter der Filterbedingung für Spalten vom Typ `injected` zulässig. Weitere Informationen finden Sie unter Injizierter Typ.
  - Leistungsverbesserung für Spalten zeichenfolgenbasierter Typen wie `CHAR` oder `VARCHAR`, die nur disjunkte Werte im Filter haben.

13. Januar 2022

Veröffentlicht am 13.1.2022

Veröffentlicht die JDBC-2.0.27- und ODBC-1.1.15-Treiber für Athena.

Der JDBC-2.0.27-Treiber enthält die folgenden Änderungen:

- Der Treiber wurde aktualisiert, um externe Kataloge abzurufen.
- Die erweiterte Treiberversionsnummer ist jetzt in der `user-agent`-Zeichenfolge als Teil des Athena-API-Aufrufs enthalten.

Der ODBC-1.1.15-Treiber enthält die folgenden Änderungen:

- Behebt ein Problem mit zweiten Aufrufen an `SQLParamData()`.

Weitere Informationen zu diesen Änderungen sowie zum Herunterladen der neuen Treiber, Versionshinweise und Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC und Herstellen einer Verbindung mit Amazon Athena mit ODBC.

Athena-Versionshinweise für 2021

26. November 2021

Veröffentlicht am 26.11.2021

Athena kündigt die öffentliche Vorschau von Athena-ACID-Transaktionen an, die Schreib-, Lösch-, Aktualisierungs- und Zeitreisevorgänge zu Athenas-SQL-Datenbearbeitungssprache (DML) hinzufügen. Athena-ACID-Transaktionen ermöglichen es mehreren gleichzeitigen Benutzern, zuverlässige Änderungen an Amazon-S3-Daten auf Zeilenebene vorzunehmen. Basierend auf dem Apache-Iceberg-Tabellenformat sind Athena-ACID-Transaktionen mit anderen Services und

Engines wie Amazon EMR und Apache Spark kompatibel, die ebenfalls die Iceberg-Tabellenformate unterstützen.

Athena-ACID-Transaktionen und vertraute SQL-Syntax vereinfachen Aktualisierungen Ihrer Geschäfts- und regulatorischen Daten. Um beispielsweise auf eine Datenlöschanforderung zu antworten, können Sie einen `DELETE`-SQL-Vorgang ausführen. Um manuelle Datensatzkorrekturen vorzunehmen, können Sie eine einzelne `UPDATE`-Anweisung verwenden. Um kürzlich gelöschte Daten wiederherzustellen, können Sie Zeitreiseabfragen mit einer `SELECT`-Anweisung verwenden. Athena-Transaktionen sind über die Athena-Konsole, API-Vorgänge sowie ODBC- und JDBC-Treiber verfügbar.

Weitere Informationen finden Sie unter Athena-ACID-Transaktionen verwenden.

24. November 2021

Veröffentlicht am 24.11.2021

Athena kündigt Unterstützung beim Lesen und Schreiben ZStandard-komprimierte ORC-, Parquet- und Textfile-Daten an. Athena verwendet die ZStandard-Komprimierungsstufe 3 beim Schreiben von komprimierten ZStandard-Daten.

Weitere Informationen zur Datenkomprimierung in Athena finden Sie unter Athena-Komprimierungs-Support.

22. November 2021

Veröffentlicht am 22.11.2021

Sie können jetzt AWS Step Functions Workflows von der Amazon Athena Athena-Konsole aus verwalten, was es einfacher macht, skalierbare Datenverarbeitungspipelines zu erstellen, Abfragen auf der Grundlage benutzerdefinierter Geschäftslogik auszuführen, Verwaltungs- und Warnaufgaben zu automatisieren und vieles mehr.

Step Functions ist jetzt in Athenas aktualisierter Konsole integriert und Sie können damit ein interaktives Workflow-Diagramm Ihrer Zustandsmaschinen anzeigen, die Athena aufrufen. Wählen Sie für die ersten Schritte Workflows aus dem linken Navigationsbereich. Wenn Sie über vorhandene Zustandsmaschinen mit Athena-Abfragen verfügen, wählen Sie einen Zustandscomputer aus, um ein interaktives Diagramm des Workflows anzuzeigen. Wenn Sie neu bei Step Functions sind, können Sie beginnen, indem Sie ein Beispielprojekt von der Athena-Konsole aus starten und es an Ihre Anwendungsfälle anpassen.

Weitere Informationen finden Sie unter Erstellen und Orchestrieren von ETL-Pipelines mit Amazon Athena und AWS Step Functions oder in der Dokumentation zu Step Functions.

18. November 2021

Veröffentlicht am 18. November 2021

Athena kündigt neue Features und Verbesserungen an.

- Support spill-to-disk für Aggregationsabfragen `DISTINCTORDER BY`, die oder beides enthalten, wie im folgenden Beispiel:

```
SELECT array_agg(orderstatus ORDER BY orderstatus)
FROM orders
GROUP BY orderpriority, custkey
```

- Um Probleme mit der Speicherbehandlung für Abfragen zu beheben, verwenden Sie `DISTINCT`. Um Fehlermeldungen wie Abfrage erschöpfte Ressourcen bei diesem Skalierungsfaktor zu vermeiden, wenn Sie `DISTINCT`-Abfragen verwenden, wählen Sie Spalten aus, die eine niedrige Kardinalität für `DISTINCT` haben, oder reduzieren Sie die Datengröße der Abfrage.
- In `SELECT COUNT(*)`-Abfragen, die keine bestimmte Spalte angeben, verbesserten sich die Leistung und die Speicherauslastung, indem nur die Anzahl ohne Zeilenpufferung beibehalten wurde.
- Führt die folgenden Zeichenfolgenfunktionen ein.
  - `translate(source, from, to)` – Gibt die `source`-Zeichenfolge zurück, wobei die in der `from`-Zeichenfolge gefundenen Zeichen durch die entsprechenden Zeichen in der `to`-Zeichenfolge ersetzt wurden. Wenn die `from`-Zeichenfolge Duplikate enthält, wird nur die erste verwendet. Wenn das Zeichen `source` nicht in der `from`-Zeichenfolge existiert, wird das `source`-Zeichen ohne Übersetzung kopiert. Wenn der Index des übereinstimmenden Zeichens in der `from`-Zeichenfolge größer als die Länge der `to`-Zeichenfolge ist, wird das Zeichen aus der resultierenden Zeichenfolge weggelassen.
  - `concat_ws(string0, array(varchar))` – Gibt die Verkettung von Elementen im Array mit `string0` als Trennzeichen an. Wenn `string0` null ist, ist der Rückgabewert null. Alle Nullwerte im Array werden übersprungen.
- Es wurde ein Fehler behoben, bei dem Abfragen fehlschlugen, wenn versucht wurde, auf ein fehlendes Unterfeld in einem `struct` zuzugreifen. Abfragen geben jetzt eine Null für das fehlende Unterfeld zurück.

- Es wurde ein Problem mit inkonsistentem Hashing für den Datentyp „decimal“ behoben.
- Es wurde ein Problem behoben, das erschöpfte Ressourcen verursachte, wenn zu viele Spalten in einer Partition vorhanden waren.

17. November 2021

Veröffentlicht am 17. November 2021

Amazon Athena unterstützt jetzt die Partitionsindexierung, um Abfragen für partitionierte Tabellen im AWS Glue Data Catalog zu beschleunigen.

Beim Abfragen partitionierter Tabellen ruft Athena die verfügbaren Tabellenpartitionen ab und filtert sie in die für Ihre Abfrage relevante Teilmenge. Wenn neue Daten und Partitionen hinzugefügt werden, ist mehr Zeit für die Verarbeitung der Partitionen erforderlich, und die Abfragelaufzeit kann sich erhöhen. Um die Partitionsverarbeitung zu optimieren und die Abfrageleistung für hochpartitionierte Tabellen zu verbessern, unterstützt Athena jetzt AWS Glue -Partitions-Indizes.

Weitere Informationen finden Sie unter AWS Glue Indizierung und Filterung von Partitionen.

16. November 2021

Veröffentlicht am 16.11.2021

Die neue und verbesserte Amazon Athena Athena-Konsole ist jetzt allgemein im AWS Handel und in den GovCloud Regionen erhältlich, in denen Athena erhältlich ist. Die neue Konsole von Athena unterstützt alle Features der früheren Konsole, verfügt jedoch über eine benutzerfreundlichere, moderne Benutzeroberfläche und enthält neue Features, die die Erfahrung beim Entwickeln von Abfragen, der Analyse von Daten und der Verwaltung Ihrer Nutzung verbessern. Sie können jetzt:

- Mehrere Abfrage-Registerkarten in einer neu gestalteten Abfrage-Registerkarte neu anordnen, navigieren oder sie schließen.
- Abfragen einfacher mit verbesserter SQL- und Textformatierung lesen und bearbeiten.
- Abfrageergebnisse zusätzlich zum Herunterladen der vollständigen Ergebnismenge in Ihre Zwischenablage kopieren.
- Abfrageverlauf, gespeicherte Abfragen und Arbeitsgruppen sortieren und auswählen, welche Spalten ein- oder ausgeblendet werden sollen.
- Eine vereinfachte Schnittstelle verwenden, um Datenquellen und Arbeitsgruppen mit weniger Klicks zu konfigurieren.

- Einstellungen für die Anzeige von Abfrageergebnissen, Abfrageverlauf, Zeilenumbruch und mehr festlegen.
- Steigern Sie Ihre Produktivität mit neuen und verbesserten Tastenkombinationen und eingebetteter Produktdokumentation.

Mit der heutigen Ankündigung wurde die Umgestaltete Konsole jetzt zum Standard. Um uns von Ihrer Erfahrung zu erzählen, wählen Sie Feedback in der linken unteren Ecke der Konsole.

Falls gewünscht, können Sie die frühere Konsole verwenden, indem Sie sich bei Ihrer anmelden AWS-Konto, Amazon Athena auswählen und im Navigationsbereich auf der linken Seite die Option Neues Athena-Erlebnis deaktivieren.

12. November 2021

Veröffentlicht am 12.11.2021

Sie können jetzt Amazon Athena verwenden, um Verbundabfragen für Datenquellen auszuführen, die sich in einem anderen AWS -Konto als Ihrem eigenen befinden. Bis heute mussten für die Abfrage dieser Daten die Datenquelle und ihr Konnektor dasselbe verwenden AWS-Konto wie der Benutzer, der die Daten abgefragt hat.

Als Datenadministrator können Sie kontoübergreifende Verbundabfragen aktivieren, indem Sie Ihren Daten-Konnektor mit dem Konto eines Datenanalysten teilen. Als Datenanalyst können Sie Ihrem Konto einen Daten-Konnektor hinzufügen, den ein Datenadministrator mit Ihnen geteilt hat. Konfigurationsänderungen am Konnektor im Ursprungskonto gelten automatisch für den freigegebenen Konnektor.

Informationen zum Aktivieren kontoübergreifender Verbundabfragen finden Sie unter Aktivieren von kontoübergreifenden Verbundabfragen. Weitere Informationen zum Abfragen von föderierten Quellen finden Sie unter Nutzung von Amazon-Athena-Verbundabfrage und Verbundabfragen ausführen.

2. November 2021

Veröffentlicht am 2. November 2021

Sie können jetzt die `EXPLAIN ANALYZE`-Anweisung in Athena verwenden, um den verteilten Ausführungsplan und die Kosten jedes Vorgangs für Ihre SQL-Abfragen anzuzeigen.

Weitere Informationen finden Sie unter Verwenden von EXPLAIN und EXPLAIN ANALYZE in Athena.

29. Oktober 2021

Veröffentlicht am 29.10.2021

Athena veröffentlicht JDBC-2.0.25- und ODBC-1.1.13-Treiber und kündigt Features und Verbesserungen an.

JDBC- und ODBC-Treiber

Veröffentlicht JDBC 2.0.25 und ODBC 1.1.13 Treiber für Athena. Beide Treiber bieten Unterstützung für die SAML-Multi-Faktor-Authentifizierung des Browsers, die für die Zusammenarbeit mit jedem SAML-2.0-Anbieter konfiguriert werden kann.

Der JDBC-2.0.25-Treiber enthält die folgenden Änderungen:

- Support für die SAML-Authentifizierung des Browsers. Der Treiber enthält ein Browser-SAML-Plug-In, das für die Zusammenarbeit mit jedem SAML-2.0-Anbieter konfiguriert werden kann.
- Support für AWS Glue API-Aufrufe. Sie können den `GlueEndpointOverride`-Parameter zum Überschreiben des AWS Glue -Endpunkts verwenden.
- `com.simba.athena.amazonaws`-Klassenpfad auf `com.amazonaws` geändert.

Der ODBC-1.1.13-Treiber enthält die folgenden Änderungen:

- Support für die SAML-Authentifizierung des Browsers. Der Treiber enthält ein Browser-SAML-Plug-In, das für die Zusammenarbeit mit jedem SAML-2.0-Anbieter konfiguriert werden kann. Ein Beispiel für die Verwendung des SAML-Plug-Ins des Browsers mit dem ODBC-Treiber finden Sie unter Konfigurieren von Single Sign-On mit ODBC, SAML 2.0 und dem Okta-Identitätsanbieter.
- Sie können jetzt die Dauer der Rollensitzung konfigurieren, wenn Sie ADFS, Azure AD oder Browser Azure AD zur Authentifizierung verwenden.

Weitere Informationen zu diesen und anderen Änderungen sowie zum Herunterladen der neuen Treiber, Versionshinweise und Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC und Herstellen einer Verbindung mit Amazon Athena mit ODBC.

Features und Verbesserungen

Athena kündigt die folgenden Features und Verbesserungen an.

- Eine neue Optimierungsregel wurde eingeführt, um in bestimmten Fällen doppelte Tabellenscans zu vermeiden.

4. Oktober 2021

Veröffentlicht am 4. Oktober 2021

Athena kündigt die folgenden Features und Verbesserungen an.

- SQL-OFFSET – Die SQL-`OFFSET`-Klausel wird jetzt in `SELECT`-Anweisungen unterstützt. Weitere Informationen finden Sie unter SELECT.
- CloudWatch Nutzungsmetriken — Athena veröffentlicht die `ActiveQueryCount` Metrik jetzt im `AWS/Usage` Namespace. Weitere Informationen finden Sie unter Überwachung der Athena-Nutzungsmetriken.
- Abfrageplanung – Es wurde ein Fehler behoben, der in seltenen Fällen Timeouts für die Abfrageplanung verursachen konnte.

16. September 2021

Veröffentlicht am 16.09.2021

Athena kündigt die folgenden neuen Features und Verbesserungen an.

Features

- Unterstützung für die Angabe von Textdateien und JSON-Komprimierung in CTAS mithilfe der `write_compression`-Tabelleneigenschaft hinzugefügt. Sie können auch die `write_compression`-Eigenschaft in CTAS für die Formate Parquet und ORC angeben. Weitere Informationen finden Sie unter CTAS-Tabelleneigenschaften.
- Das BZIP2-Komprimierungsformat wird jetzt zum Schreiben von Textdateien und JSON-Dateien unterstützt. Weitere Informationen zu den Komprimierungsformaten in Athena finden Sie unter Athena-Komprimierungs-Support.

Verbesserungen

- Es wurde ein Fehler behoben, bei dem Identitätsinformationen nicht an die UDF-Lambda-Funktion gesendet wurden.

- Ein Prädikat-Pushdown-Problem mit disjunkten Filterbedingungen wurde behoben.
- Ein Hashing-Problem für Dezimaltypen wurde behoben.
- Ein unnötiges Problem der Statistikerfassung wurde behoben.
- Eine inkonsistente Fehlermeldung wurde entfernt.
- Verbesserte Broadcast-Beitrittleistung durch Anwenden des dynamischen Partitionsschnitts im Worker-Knoten.
- Für Verbundabfragen:
  - Die Konfiguration wurde geändert, um das Auftreten von `CONSTRAINT_VIOLATION`-Fehlern in Verbundabfragen zu reduzieren.

15. September 2021

Veröffentlicht am 15.09.2021

Sie können jetzt eine neu gestaltete Amazon Athena-Konsole verwenden (Vorschau). Ein neuer Athena-JDBC-Treiber wurde freigegeben.

Athena-Konsolen-Vorschau

Sie können jetzt eine neu gestaltete Amazon Athena Athena-Konsole (Vorversion) von jedem AWS-Region Ort aus verwenden, an dem Athena verfügbar ist. Die neue Konsole unterstützt alle Features der vorhandenen Konsole, jedoch von einer benutzerfreundlicheren, modernen Benutzeroberfläche.

Um zur neuen Konsole zu wechseln, melden Sie sich bei Ihrer an AWS-Konto und wählen Sie Amazon Athena. Wählen Sie in der Navigationsleiste der AWS Konsole die Option Zur neuen Konsole wechseln aus. Um zur Standardkonsole zurückzukehren, deaktivieren Sie Neue Athena-Erfahrung aus dem Navigationsbereich auf der linken Seite.

Starten Sie noch heute mit der neuen Konsole. Wählen Sie Feedback in der unteren linken Ecke, um uns von Ihren Erfahrungen zu erzählen.

Athena-JDBC-Treiber 2.0.24

Athena kündigt die Verfügbarkeit von JDBC-Treiberversion 2.0.24 für Athena an. Diese Version aktualisiert die Proxy-Unterstützung für alle Anbieter von Anmeldeinformationen. Der Treiber unterstützt jetzt die Proxy-Authentifizierung für alle Hosts, die von der `NonProxyHosts`-Verbindungseigenschaft nicht unterstützt werden.

Der Einfachheit halber enthält diese Version Downloads des JDBC-Treibers sowohl mit als auch ohne SDK. AWS Mit dieser JDBC-Treiberversion können Sie sowohl das AWS-SDK als auch den Athena-JDBC-Treiber in das Projekt einbetten.

Weitere Informationen und das Herunterladen des neuen Treibers, der Versionshinweise und der Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC.

31. August 2021

Veröffentlicht am 31.08.2021

Athena kündigt die folgenden Featureerweiterungen und Fehlerbehebungen an.

- Athena-Verbundverbesserungen – Athena hat als Teil des Athena Query Federation SDK Unterstützung für Kartentypen und eine bessere Unterstützung für komplexe Typen hinzugefügt. Diese Version enthält auch einige Speicherverbesserungen und Leistungsoptimierungen.
- Neue Fehlerkategorien – Einführung der Fehlerkategorien `USER` und `SYSTEM` in Fehlermeldungen. Diese Kategorien helfen Ihnen, zwischen Fehlern zu unterscheiden, die Sie selbst beheben können (`USER`) und Fehlern, die möglicherweise Unterstützung durch den Athena-Support erfordern (`SYSTEM`).
- Fehler-Messaging für die Verbundabfrage – `USER_ERROR`-Kategorisierungen für Fehler im Zusammenhang mit Verbundabfragen aktualisiert.
- JOIN — Damit spill-to-disk verbundene Fehler und Speicherprobleme wurden behoben, um die Leistung zu verbessern und Speicherfehler bei `JOIN` Vorgängen zu reduzieren.

12. August 2021

Veröffentlicht am 12.08.2021

Veröffentlicht den ODBC-1.1.12-Treiber für Athena. Diese Version behebt Probleme im Zusammenhang mit `SQLPrepare()`, `SQLGetInfo()` und `EndpointOverride`.

Informationen zum Herunterladen der neuen Treiber, Versionshinweise und Dokumentation finden Sie unter Herstellen einer Verbindung mit Amazon Athena mit ODBC.

6. August 2021

Veröffentlicht am 06.08.2021

Amazon Athena gibt die Verfügbarkeit von Athena und seinen Features in der Region Asien-Pazifik (Osaka) bekannt.

Diese Version erweitert die Verfügbarkeit von Athena in Asien-Pazifik (Hongkong), Asien-Pazifik (Mumbai), Asien-Pazifik (Osaka), Asien-Pazifik (Seoul), Asien-Pazifik (Singapur), Asien-Pazifik (Sydney) und Asien-Pazifik (Tokio). Eine vollständige Liste der in diesen und anderen Regionen AWS-Services verfügbaren Dienste finden Sie in der Liste AWS-Region aller Dienste.

5. August 2021

Veröffentlicht am 05.08.2021

Sie können die `UNLOAD`-Anweisung verwenden, um die Ausgabe einer `SELECT`-Abfrage in die Formate PARQUET, ORC, AVRO und JSON zu schreiben.

Weitere Informationen finden Sie unter UNLOAD.

30. Juli 2021

Veröffentlicht am 30.07.2021

Athena gibt die folgenden Featureerweiterungen und Fehlerbehebungen bekannt.

- Dynamische Filterung und Partitionsbereinigung – Verbesserungen erhöhen die Leistung und reduzieren die bei bestimmten Abfragen gescannte Datenmenge, wie im folgenden Beispiel.

  In diesem Beispiel wird davon ausgegangen, dass `Table_B` eine nicht partitionierte Tabelle ist, deren Dateigrößen sich auf weniger als 20 MB summieren. Bei solchen Abfragen werden weniger Daten von `Table_A` gelesen und die Abfrage wird schneller abgeschlossen.

  ```
  SELECT *
  FROM Table_A
  JOIN Table_B ON Table_A.date = Table_B.date
  WHERE Table_B.column_A = "value"
  ```

- ORDER BY mit LIMIT, DISTINCT mit LIMIT – Leistungsverbesserungen bei Abfragen, die `ORDER BY` oder `DISTINCT` gefolgt von einer `LIMIT`-Klausel verwenden.
- S3-Glacier-Deep-Archive-Dateien – Wenn Athena eine Tabelle abfragt, die eine Mischung aus S3-Glacier-Deep-Archive-Dateien und Nicht-S3-Glacier-Dateien enthält, überspringt Athena jetzt die S3-Glacier-Deep-Archive-Dateien für Sie. Zuvor mussten Sie diese Dateien manuell vom Abfragespeicherort verschieben, da die Abfrage sonst fehlschlägt. Wenn Sie Athena

verwenden möchten, um Objekte im S3-Glacier-Deep-Archive-Speicher abzufragen, müssen Sie sie wiederherstellen. Weitere Informationen finden Sie unter Wiederherstellen eines archivierten Objekts im Amazon-S3-Benutzerhandbuch.

- Es wurde ein Fehler behoben, bei dem leere Dateien, die von der CTAS-`bucketed_by`-Tabelleneigenschaft erstellt wurden, nicht korrekt verschlüsselt wurden.

21. Juli 2021

Veröffentlicht am 21.07.2021

Mit der Version Juli 2021 von Microsoft Power BI Desktop können Sie Berichte und Dashboards mit einem nativen Datenquellen-Konnektor für Amazon Athena erstellen. Der Connector für Amazon Athena ist als Standardkonnektor in Power BI verfügbar. Er unterstützt DirectQueryund ermöglicht die Analyse großer Datenmengen und die Aktualisierung von Inhalten über Power BI Gateway.

Da der Konnektor Ihren vorhandenen ODBC-Datenquellennamen (DSN) verwendet, um eine Verbindung mit Athena herzustellen und Abfragen auszuführen, benötigt er den Athena-ODBC-Treiber. Informationen zum Herunterladen des neuesten ODBC-Treibers finden Sie unter Herstellen einer Verbindung mit Amazon Athena mit ODBC.

Weitere Informationen finden Sie unter Verwenden des Amazon-Athena-Power-BI-Connectors.

16. Juli 2021

Veröffentlicht am 16.07.2021

Amazon Athena hat seine Integration mit Apache Hudi aktualisiert. Hudi ist ein Open-Source-Datenmanagement-Framework zur Vereinfachung der inkrementellen Datenverarbeitung in Amazon-S3-Data-Lakes. Die aktualisierte Integration ermöglicht es Ihnen, mithilfe von Athena-Hudi 0.8.0 Tabellen abzufragen, die über Amazon EMR, Apache Spark, Apache Hive oder andere kompatible Services verwaltet werden. Darüber hinaus unterstützt Athena jetzt zwei zusätzliche Features: Snapshot-Abfragen auf Merge-on-Read (MoR)-Tabellen und Lese-Unterstützung für Bootstrap-Tabellen.

Apache Hudi bietet Datenverarbeitung auf Datensatzebene, mit der Sie die Entwicklung von CDC-Pipelines (Change Data Capture) vereinfachen, GDPR-gesteuerte Aktualisierungen und Löschungen einhalten und Streaming-Daten von Sensoren oder Geräten besser verwalten können, die Dateneinfügung und Ereignisaktualisierungen erfordern. Mit der Version 0.8.0 ist es einfacher, große Parquet-Tabellen nach Hudi zu migrieren, ohne Daten zu kopieren, damit Sie sie über Athena

abfragen und analysieren können. Sie können die neue Unterstützung von Athena für Snapshot-Abfragen verwenden, um nahezu Echtzeitansichten Ihrer Streaming-Tabellenaktualisierungen zu erhalten.

Weitere Informationen zur Verwendung von Hudi mit Athena finden Sie unter Verwenden von Athena zum Abfragen von Apache-Hudi-Datensätzen.

8. Juli 2021

Veröffentlicht am 08.07.2021

Veröffentlicht den ODBC 1.1.11-Treiber für Athena. Der ODBC-Treiber kann die Verbindung jetzt mit einem JSON-Web-Token (JWT) authentifizieren. Unter Linux wurde der Standardwert für die Arbeitsgruppeneigenschaft auf Primär festgelegt.

Weitere Informationen und das Herunterladen des neuen Treibers, der Versionshinweise und der Dokumentation finden Sie unter Herstellen einer Verbindung mit Amazon Athena mit ODBC.

1. Juli 2021

Veröffentlicht am 01.07.2021

Am 1. Juli 2021 endete die spezielle Handhabung von Vorschau-Arbeitsgruppen. Während `AmazonAthenaPreviewFunctionality`-Arbeitsgruppen ihren Namen behalten, haben sie keinen Sonderstatus mehr. Sie können weiterhin `AmazonAthenaPreviewFunctionality`-Arbeitsgruppen verwenden, um Abfragen anzuzeigen, zu ändern, zu organisieren und auszuführen. Abfragen, die Features verwenden, die sich zuvor in der Vorschau befanden, unterliegen jetzt jedoch den standardmäßigen Abrechnungsbedingungen von Athena. Weitere Informationen zur Fakturierung finden Sie unter Preise zu Amazon Athena.

23. Juni 2021

Veröffentlicht am 23.06.2021

Veröffentlicht JDBC 2.0.23 und ODBC 1.1.10 Treiber für Athena. Beide Treiber bieten eine verbesserte Leseleistung und unterstützen EXPLAIN-Anweisungen und parametrisierte Abfragen.

`EXPLAIN`-Anweisungen zeigen den logischen oder verteilten Ausführungsplan einer SQL-Abfrage. Parametrisierte Abfragen ermöglichen die mehrfache Verwendung derselben Abfrage mit unterschiedlichen Werten, die zur Laufzeit bereitgestellt werden.

Die JDBC-Version bietet außerdem Unterstützung für Active Directory Federation Services 2019 und eine benutzerdefinierte Endpunktüberschreibungsoption für AWS STS. Die ODBC-Version behebt ein Problem mit den IAM-Profilanmeldeinformationen.

Weitere Informationen und das Herunterladen des neuen Treibers, der Versionshinweise und der Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC und Herstellen einer Verbindung mit Amazon Athena mit ODBC.

12. Mai 2021

Veröffentlicht am 12.05.2021

Sie können jetzt Amazon Athena verwenden, um einen AWS Glue Katalog von einem anderen Konto als Ihrem eigenen zu registrieren. Nachdem Sie die erforderlichen IAM-Berechtigungen für konfiguriert haben AWS Glue, können Sie Athena verwenden, um kontenübergreifende Abfragen auszuführen.

Weitere Informationen finden Sie unter Registrieren eines AWS Glue Data Catalog von einem anderen Konto und Kontoübergreifender Zugriff auf AWS Glue -Datenkataloge.

10. Mai 2021

Veröffentlicht am 10.05.2021

ODBC-Treiberversion 1.1.9.1001 für Athena veröffentlicht. Diese Version behebt ein Problem mit dem `BrowserAzureAD`-Authentifizierungstyp bei Verwendung von Azure Active Directory (AD).

Informationen zum Herunterladen der neuen Treiber, Versionshinweise und Dokumentation finden Sie unter Herstellen einer Verbindung mit Amazon Athena mit ODBC.

5. Mai 2021

Veröffentlicht am 05.05.2021

Sie können jetzt den Amazon-Athena-Vertica-Connector in Verbundabfragen verwenden, um Vertica-Datenquellen von Athena abzufragen. Sie können beispielsweise analytische Abfragen über ein Data Warehouse in Vertica und einen Data Lake in Amazon S3 ausführen.

Um den Athena Vertica-Konnektor bereitzustellen, besuchen Sie die AthenaVerticaConnectorSeite im. AWS Serverless Application Repository

Der Amazon Athena Vertica Konnektor stellt mehrere Konfigurationsoptionen über Lambda-Umgebungsvariablen zur Verfügung. Informationen zu Konfigurationsoptionen, Parametern, Verbindungszeichenfolgen, Bereitstellung und Einschränkungen finden Sie unter Amazon Athena Vertica Konnektor.

Ausführlichere Informationen zur Verwendung des Vertica-Konnektor finden Sie unter Querying a Vertica data source in Amazon Athena using the Athena Federated Query SDK (Abfragen einer Vertica-Datenquelle in Amazon Athena mithilfe des Athena-Federated-Query-SDK) im AWS -Big-Data-Blog.

30. April 2021

Veröffentlicht am 30.04.2021

Veröffentlicht Treiber JDBC 2.0.21 und ODBC 1.1.9 für Athena. Beide Versionen unterstützen die SAML-Authentifizierung mit Azure Active Directory (AD) und die SAML-Authentifizierung mit. PingFederate Die JDBC-Version unterstützt auch parametrisierte Abfragen. Informationen zu parametrisierten Abfragen in Athena finden Sie unter Verwenden von parametrisierten Abfragen.

Informationen zum Herunterladen der neuen Treiber, Versionshinweise und Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC und Herstellen einer Verbindung mit Amazon Athena mit ODBC.

29. April 2021

Veröffentlicht am 29.04.2021

Amazon Athena kündigt die Verfügbarkeit von Athena Engine Version 2 in den Regionen China (Peking) und China (Ningxia) an.

Informationen über Athena-Engine-Version 2 finden Sie unter Athena-Engine-Version 2.

26. April 2021

Veröffentlicht am 26.04.2021

Fensterwert-Funktionen in Athena-Engine-Version 2 unterstützen jetzt `IGNORE NULLS` und `RESPECT NULLS`.

Weitere Informationen finden Sie unter Wertfunktionen in der Presto-Dokumentation.

21. April 2021

Veröffentlicht am 21.04.2021

Amazon Athena kündigt die Verfügbarkeit von Athena Engine Version 2 in den Regionen Europa (Mailand) und Afrika (Kapstadt) an.

Informationen über Athena-Engine-Version 2 finden Sie unter Athena-Engine-Version 2.

05. April 2021

Veröffentlicht am 05.04.2021

EXPLAIN-Anweisung

Sie können jetzt die `EXPLAIN`-Anweisung in Athena verwenden, um die Ausführungspläne für Ihre SQL-Abfragen anzuzeigen.

Weitere Informationen finden Sie unter Verwenden von EXPLAIN und EXPLAIN ANALYZE in Athena und Die Ergebnisse der Athena-EXPLAIN-Anweisung verstehen.

SageMaker Modelle für Machine Learning in SQL-Abfragen

Die Inferenz von Modellen für maschinelles Lernen mit Amazon SageMaker ist jetzt allgemein für Amazon Athena verfügbar. Verwenden Sie Machine-Learning-Modelle in SQL-Abfragen, um komplexe Aufgaben wie Anomalieerkennung, Kundenkohortenanalyse und Zeitreihenvorhersagen zu vereinfachen, indem Sie eine Funktion in einer SQL-Abfrage aufrufen.

Weitere Informationen finden Sie unter Verwendung von Machine Learning (ML) mit Amazon Athena.

Benutzerdefinierte Funktionen (User Defined Functions, UDFs)

Benutzerdefinierte Funktionen (UDFs) sind jetzt allgemein für Athena verfügbar. Verwenden Sie UDFs, um benutzerdefinierte Funktionen zu nutzen, die Datensätze oder Datensatzgruppen in einer einzelnen SQL-Abfrage verarbeiten.

Weitere Informationen finden Sie unter Abfragen mit benutzerdefinierten Funktionen (User Defined Functions, UDFs).

30. März 2021

Veröffentlicht am 30.03.2021

Amazon Athena kündigt die Verfügbarkeit von Athena Engine Version 2 in den Regionen Asien-Pazifik (Hongkong) und Naher Osten (Bahrain) an.

Informationen über Athena-Engine-Version 2 finden Sie unter Athena-Engine-Version 2.

25. März 2021

Veröffentlicht am 25.03.2021

Amazon Athena kündigt die Verfügbarkeit von Athena Engine Version 2 in der Region Europa (Stockholm) an.

Informationen über Athena-Engine-Version 2 finden Sie unter Athena-Engine-Version 2.

5. März 2021

Veröffentlicht am 05.03.2021

Amazon Athena kündigt die Verfügbarkeit von Athena Engine Version 2 in den Regionen Kanada (Zentral), Europa (Frankfurt) und Südamerika (São Paulo) an.

Informationen über Athena-Engine-Version 2 finden Sie unter Athena-Engine-Version 2.

25. Februar 2021

Veröffentlicht am 25.02.2021

Amazon Athena kündigt allgemeine Verfügbarkeit von Athena-Engine-Version 2 in den Regionen Asien-Pazifik (Singapur), Asien-Pazifik (Sydney), Europa (London) und Europa (Paris) an.

Informationen über Athena-Engine-Version 2 finden Sie unter Athena-Engine-Version 2.

Athena-Versionshinweise für 2020

16. Dezember 2020

Veröffentlicht am 16.12.2020

Amazon Athena kündigt die Verfügbarkeit von Athena Engine Version 2, Athena Federated Query und AWS PrivateLink in weiteren Regionen an.

Athena-Engine-Version 2 und Athena Federated Query

Amazon Athena kündigt allgemeine Verfügbarkeit von Athena-Engine-Version 2 und Athena Federated Query in den Regionen Asien-Pazifik (Mumbai), Asien-Pazifik (Tokio), Europa (Irland) und USA West (Nordkalifornien) an. Die Athena Engine Version 2 und Verbundabfragen sind bereits in den Regionen USA Ost (Nord-Virginia), USA Ost (Ohio) und USA West (Oregon) verfügbar.

Weitere Informationen finden Sie unter Athena-Engine-Version 2 und Nutzung von Amazon-Athena-Verbundabfrage.

AWS PrivateLink

AWS PrivateLink for Athena wird jetzt in der Region Europa (Stockholm) unterstützt. Informationen zu AWS PrivateLink For Athena finden Sie unterHerstellen einer Verbindung mit Amazon Athena über einen Schnittstellen-VPC-Endpunkt.

24. November 2020

Veröffentlicht am 24.11.2020

Veröffentlicht Treiber JDBC 2.0.16 und ODBC 1.1.6 für Athena. Diese Versionen unterstützen auf Kontoebene die Okta Verify Multi-Faktor-Authentifizierung (MFA). Sie können Okta MFA auch verwenden, um die SMS-Authentifizierung und die Google-Authenticator-Authentifizierung als Faktoren zu konfigurieren.

Informationen zum Herunterladen der neuen Treiber, Versionshinweise und Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC und Herstellen einer Verbindung mit Amazon Athena mit ODBC.

11. November 2020

Veröffentlicht am 11.11.2020

Amazon Athena gibt die allgemeine Verfügbarkeit in den Regionen USA Ost (Nord-Virginia), USA Ost (Ohio) und USA West (Oregon) für Athena-Engine-Version 2 und Verbundabfragen bekannt.

Athena-Engine-Version 2

Amazon Athena kündigt allgemeine Verfügbarkeit einer neuen Abfrage-Engine-Version Athena-Engine-Version 2 in den Regionen USA Ost (Nord-Virginia), USA Ost (Ohio) und USA West (Oregon) an.

Athena-Engine-Version 2 umfasst Leistungsverbesserungen und neue Features wie Unterstützung für die Schema-Evolution für Parkett-Formatdaten, zusätzliche räumliche Funktionen, Unterstützung für das Lesen verschachtelter Schemas zur Kostensenkung und Leistungsverbesserungen bei JOIN- und AGGREGATE-Vorgänge.

- Weitere Informationen zu Verbesserungen, wichtigen Änderungen und Fehlerbehebungen finden Sie unter Athena-Engine-Version 2.
- Informationen zum Upgrade finden Sie unter Ändern von Athena-Engine-Versionen.
- Informationen zum Testen von Abfragen finden Sie unter Testen von Abfragen im Voraus eines Engine-Versions-Upgrades.

Verbund-SQL-Abfragen

Sie können die Verbundabfrage von Athena jetzt in den Regionen USA Ost (Nord-Virginia), USA Ost (Ohio) und USA West (Oregon) verwenden, ohne die `AmazonAthenaPreviewFunctionality`-Arbeitsgruppe zu verwenden.

Verwenden Sie Verbund-SQL-Abfragen, um SQL-Abfragen über relationale, nicht relationale, objektbezogene und benutzerdefinierte Datenquellen hinweg auszuführen. Mit der Verbundabfrage können Sie eine einzelne SQL-Abfrage senden, mit der Daten aus mehreren Quellen gescannt werden, die On-Premises ausgeführt oder in der Cloud gehostet werden.

Die Ausführung von Analysen für über Anwendungen verteilte Daten kann aus folgenden Gründen komplex und zeitaufwändig sein:

- Die für die Analyse erforderlichen Daten sind häufig auf relationale, Schlüsselwert-, Dokument-, In-Memory-, Such-, Grafik-, Objekt-, Zeitreihen- und Hauptbuchdatenspeicher verteilt.
- Um Daten über diese Quellen hinweg zu analysieren, erstellen Analysten komplexe Pipelines, um sie zu extrahieren, zu transformieren und in ein Data Warehouse zu laden, sodass die Daten abgefragt werden können.
- Der Zugriff auf Daten aus verschiedenen Quellen erfordert das Erlernen neuer Programmiersprachen und Datenzugriffskonstrukte.

Verbund-SQL-Abfragen in Athena beseitigen diese Komplexität, indem Benutzer die Daten direkt von jedem Ort aus abfragen können. Analysten können vertraute SQL-Konstrukte verwenden, um Daten mit `JOIN` für eine schnelle Analyse aus mehreren Datenquellen zusammenzuführen und die Ergebnisse in Amazon S3 für die spätere Verwendung zu speichern.

Datenquellen-Konnektors

Um Verbundabfragen zu verarbeiten, verwendet Athena Athena-Datenquellen-Konnektor, die auf AWS Lambda ausgeführt werden. Die folgenden quelloffenen, vorgefertigten Konnektors wurden von Athena geschrieben und getestet. Verwenden Sie sie, um SQL-Abfragen in Athena auf ihren entsprechenden Datenquellen auszuführen.

- CloudWatch
- CloudWatch -Metriken
- DocumentDB
- DynamoDB
- OpenSearch
- HBase
- Neptune
- Redis
- Timestream
- TPC Benchmark DS (TPC-DS)

Benutzerdefinierte Datenquellen-Konnektors

Mithilfe des Athena Query Federation SDK können Entwickler außerdem Konnektors zu jeder Datenquelle erstellen, damit Athena SQL-Abfragen für diese Datenquelle ausführen kann. Athena Query Federation Connector erweitert die Vorteile von Verbundabfragen über die bereitgestellten Konnektoren hinaus AWS . Da Connectors auf dem System laufen AWS Lambda, müssen Sie sich nicht um die Infrastruktur kümmern und auch keine Skalierung für Spitzenanforderungen planen.

Nächste Schritte

- Weitere Informationen zum Verbundabfragefeature finden Sie unter Nutzung von Amazon-Athena-Verbundabfrage.
- Informationen zu den ersten Schritten bei der Verwendung eines vorhandenen Konnektors finden Sie unter Bereitstellen von Konnektors und Herstellen von Verbindungen mit Datenquellen.
- Informationen zum Erstellen eines eigenen Datenquellen-Connectors mit dem Athena Query Federation SDK finden Sie unter Beispiel für einen Athena Connector unter. GitHub

22. Oktober 2020

Veröffentlicht am 22.10.2020

Sie können Athena jetzt mit AWS Step Functions anrufen. AWS Step Functions kann bestimmte AWS-Services direkt über die Sprache der Amazon-Staaten steuern. Sie können Step Functions mit Athena verwenden, um die Abfrageausführung zu starten und zu stoppen, Abfrageergebnisse abzurufen, Ad-hoc- oder geplante Datenabfragen auszuführen und Ergebnisse aus Data Lakes in Amazon S3 abzurufen.

Weitere Informationen finden Sie unter Aufrufen von Athena mit Step Functions im AWS Step Functions -Entwicklerhandbuch.

29. Juli 2020

Veröffentlicht am 29.07.2020

JDBC-Treiberversion 2.0.13 wurde freigegeben. Diese Version unterstützt die Verwendung mehrerer bei Athena registrierter Datenkataloge, des Okta-Services zur Authentifizierung und Verbindungen zu VPC-Endpunkten.

Um die neue Version des Treibers herunterzuladen und zu verwenden, siehe Herstellen einer Verbindung zu Amazon Athena mit JDBC.

9. Juli 2020

Veröffentlicht am 09.07.2020

Amazon Athena bietet Unterstützung für die Abfrage komprimierter Hudi-Datensätze und fügt die AWS CloudFormation `AWS::Athena::DataCatalog` Ressource zum Erstellen, Aktualisieren oder Löschen von Datenkatalogen hinzu, die Sie in Athena registrieren.

Abfragen von Apache-Hudi-Datensätzen

Apache Hudi ist ein Open-Source-Datenmanagement-Framework, das die inkrementelle Datenverarbeitung vereinfacht. Amazon Athena unterstützt jetzt das Abfragen der leseoptimierten Ansicht eines Apache-Hudi-Datensatzes in Ihrem Amazon-S3-basierten Data Lake.

Weitere Informationen finden Sie unter Verwenden von Athena zum Abfragen von Apache-Hudi-Datensätzen.

AWS CloudFormation Datenkatalog-Ressource

Um das Verbundabfragefeature von Amazon Athena zum Abfragen einer beliebigen Datenquelle zu verwenden, müssen Sie zuerst Ihren Datenkatalog in Athena registrieren. Sie können die AWS CloudFormation `AWS::Athena::DataCatalog` Ressource jetzt verwenden, um Datenkataloge zu erstellen, zu aktualisieren oder zu löschen, die Sie in Athena registrieren.

Weitere Informationen finden Sie AWS::Athena::DataCatalogim AWS CloudFormation Benutzerhandbuch.

1. Juni 2020

Veröffentlicht am 01.06.2020

Verwenden von Apache Hive Metastore als Metakatalog mit Amazon Athena

Sie können Athena nun mit einem oder mehreren Apache-Hive-Metastores verbinden, zusätzlich zu AWS Glue Data Catalog mit Athena.

Um eine Verbindung zu einem selbst gehosteten Hive Metastore herzustellen, benötigen Sie einen Athena Hive-Metastore-Konnektor. Athena bietet einen Referenz-Implementierungs-Konnektor, den Sie verwenden können. Der Konnektor wird in Ihrem Konto als AWS Lambda -Funktion ausgeführt.

Weitere Informationen finden Sie unter Verwenden von Athena-Daten-Connector für externen Hive-Metastore.

21. Mai 2020

Veröffentlicht am 21.05.2020

Amazon Athena fügt Unterstützung für Partitionsprojektion hinzu. Verwenden Sie die Partitionsprojektion, um die Abfrageverarbeitung von hochpartitionierten Tabellen zu beschleunigen und das Partitionsmanagement zu automatisieren. Weitere Informationen finden Sie unter Partitionsprojektion mit Amazon Athena.

01. April 2020

Veröffentlicht am 01.04.2020

Neben der Region USA Ost (Nord-Virginia) sind die Amazon-Athena-Verbundabfrage, benutzerdefinierte Funktionen (UDFs), Machine-Learning-Inferenz und externe Hive-Metastore-

Features jetzt in der Vorschau in den Regionen Asien-Pazifik (Mumbai), Europa (Irland) und USA West (Oregon) verfügbar.

11. März 2020

Veröffentlicht am 11.03.2020

Amazon Athena veröffentlicht jetzt EventBridge Amazon-Ereignisse für Abfragestatusübergänge. Wenn eine Abfrage zwischen Zuständen wechselt — zum Beispiel vom Status Wird ausgeführt in einen Terminalstatus wie Erfolgreich oder Abgebrochen — veröffentlicht Athena ein Ereignis zur EventBridge Änderung des Abfragestatus für. Das Ereignis enthält Informationen zum Abfragezustandsübergang. Weitere Informationen finden Sie unter Überwachung von Athena-Abfragen mit Amazon EventBridge-Ereignissen.

6. März 2020

Veröffentlicht am 06.03.2020

Sie können jetzt Amazon Athena Athena-Arbeitsgruppen mithilfe der AWS CloudFormation `AWS::Athena::WorkGroup` Ressource erstellen und aktualisieren. Weitere Informationen finden Sie AWS::Athena::WorkGroupim AWS CloudFormation Benutzerhandbuch.

Athena-Versionshinweise für 2019

26. November 2019

Veröffentlicht am 12.17.2019

Amazon Athena unterstützt die Ausführung von SQL-Abfragen über relationale, nicht relationale, objekt- und benutzerdefinierte Datenquellen hinweg, das Aufrufen von Machine-Learning-Modellen in SQL-Abfragen, benutzerdefinierten Funktionen (User Defined Functions, UDFs) (Vorversion), die Verwendung von Apache Hive Metastore als Metadatenkatalog mit Amazon Athena (Vorversion) und vier zusätzliche abfragebezogene Metriken.

Verbund-SQL-Abfragen

Verwenden Sie Verbund-SQL-Abfragen, um SQL-Abfragen über relationale, nicht relationale, objektbezogene und benutzerdefinierte Datenquellen hinweg auszuführen.

Sie können mit einer Verbundabfrage von Athena Daten scannen, die in relationalen, nicht relationalen, objektbezogenen und benutzerdefinierten Datenquellen gespeichert sind. Mit der

Verbundabfrage können Sie eine einzelne SQL-Abfrage senden, mit der Daten aus mehreren Quellen gescannt werden, die On-Premises ausgeführt oder in der Cloud gehostet werden.

Die Ausführung von Analysen für über Anwendungen verteilte Daten kann aus folgenden Gründen komplex und zeitaufwändig sein:

- Die für die Analyse erforderlichen Daten sind häufig auf relationale, Schlüsselwert-, Dokument-, In-Memory-, Such-, Grafik-, Objekt-, Zeitreihen- und Hauptbuchdatenspeicher verteilt.
- Um Daten über diese Quellen hinweg zu analysieren, erstellen Analysten komplexe Pipelines, um sie zu extrahieren, zu transformieren und in ein Data Warehouse zu laden, sodass die Daten abgefragt werden können.
- Der Zugriff auf Daten aus verschiedenen Quellen erfordert das Erlernen neuer Programmiersprachen und Datenzugriffskonstrukte.

Verbund-SQL-Abfragen in Athena beseitigen diese Komplexität, indem Benutzer die Daten direkt von jedem Ort aus abfragen können. Analysten können vertraute SQL-Konstrukte verwenden, um Daten mit `JOIN` für eine schnelle Analyse aus mehreren Datenquellen zusammenzuführen und die Ergebnisse in Amazon S3 für die spätere Verwendung zu speichern.

Datenquellen-Konnektors

Athena verarbeitet Verbundabfragen mithilfe von Athena-Datenquellen-Konnektors, die auf AWS Lambda ausgeführt werden. Verwenden Sie diese Open-Source-Datenquellen-Konnektoren, um föderierte SQL-Abfragen in Athena über Amazon DynamoDB, Apache HBase, Amazon Document DB, Amazon CloudWatch, Amazon CloudWatch Metrics und JDBC-konforme relationale Datenbanken wie MySQL und PostgreSQL unter der Apache 2.0-Lizenz auszuführen.

Benutzerdefinierte Datenquellen-Konnektors

Mithilfe des Athena Query Federation SDK können Entwickler außerdem Konnektors zu jeder Datenquelle erstellen, damit Athena SQL-Abfragen für diese Datenquelle ausführen kann. Athena Query Federation Connector erweitert die Vorteile von Verbundabfragen über die bereitgestellten Konnektoren hinaus AWS . Da Connectors auf dem System laufen AWS Lambda, müssen Sie sich nicht um die Infrastruktur kümmern und auch keine Skalierung für Spitzenanforderungen planen.

Verfügbarkeit der Vorversion

Die Athena-Verbundabfrage ist in der Region USA Ost (Nord-Virginia) als Vorversion verfügbar.

Nächste Schritte

- Befolgen Sie zum Ausprobieren der Vorversion die Anweisungen in Häufig gestellten Fragen zu Vorschaufeatures in Athena.
- Weitere Informationen zum Verbundabfragefeature finden Sie unter Verwenden der Amazon-Athena-Verbundabfrage (Vorversion).
- Informationen zu den ersten Schritten bei der Verwendung eines vorhandenen Konnektors finden Sie unter Bereitstellen von Konnektors und Herstellen von Verbindungen mit Datenquellen.
- Informationen zum Erstellen eines eigenen Datenquellen-Connectors mit dem Athena Query Federation SDK finden Sie unter Beispiel für einen Athena Connector unter. GitHub

Aufrufen von Machine Learning-Modellen in SQL-Abfragen

Sie können Machine-Learning-Modelle für die Inferenz nun direkt in Ihren Athena-Abfragen aufrufen. Dank der Möglichkeit, Machine Learning-Modelle in SQL-Abfragen zu verwenden, gestalten sich komplexe Aufgaben wie die Anomalieerkennung, eine Kohortenanalyse beim Kunden und Vertriebsprognosen so einfach wie das Aufrufen einer Funktion in einer SQL-Abfrage.

ML-Modelle

Sie können mehr als ein Dutzend von Amazon SageMaker bereitgestellte integrierte Algorithmen für maschinelles Lernen verwenden, Ihre eigenen Modelle trainieren oder Modellpakete von Amazon SageMaker Hosting Services finden AWS Marketplaceund abonnieren und dort bereitstellen. Es ist keine zusätzliche Einrichtung erforderlich. Sie können diese ML-Modelle in Ihren SQL-Abfragen über die Athena-Konsole, Athena-APIs und über den Vorschau-JDBC-Treiber von Athena aufrufen.

Verfügbarkeit der Vorversion

Die ML-Funktionalität von Athena ist heute als Vorversion in der Region USA Ost (Nord-Virginia) verfügbar.

Nächste Schritte

- Befolgen Sie zum Ausprobieren der Vorversion die Anweisungen in Häufig gestellten Fragen zu Vorschaufeatures in Athena.
- Weitere Informationen zum Feature von Machine Learning finden Sie unter Verwendung von Machine Learning (ML) mit Amazon Athena (Vorversion).

Benutzerdefinierte Funktionen (User Defined Functions, UDFs) (Vorversion)

Sie können nun benutzerdefinierte skalare Funktionen schreiben und sie in Ihren Athena-Abfragen aufrufen. Sie können Ihre UDFs mit dem Athena Query Federation SDK in Java schreiben. Wenn eine UDF in einer an Athena übermittelten SQL-Abfrage verwendet wird, wird sie auf AWS Lambda aufgerufen und ausgeführt. UDFs können in sowohl in `SELECT`- als auch in `FILTER`-Klauseln einer SQL-Abfrage verwendet werden. Sie können mehrere UDFs in derselben Abfrage aufrufen.

Verfügbarkeit der Vorversion

Die Athena-UDF-Funktionalität ist im Vorschaumodus in der Region USA Ost (Nord-Virginia) verfügbar.

Nächste Schritte

- Befolgen Sie zum Ausprobieren der Vorversion die Anweisungen in Häufig gestellten Fragen zu Vorschaufeatures in Athena.
- Weitere Informationen finden Sie unter Abfragen mit benutzerdefinierten Funktionen (User Defined Functions, UDFs) (Vorversion).
- Beispiele für UDF-Implementierungen finden Sie unter Amazon Athena UDF Connector. GitHub
- Informationen zum Schreiben eigener Funktionen mit dem Athena Query Federation SDK finden Sie unter Erstellen und Bereitstellen eines UDF mit Lambda.

Verwenden von Apache Hive Metastore als Metakatalog mit Amazon Athena (Vorversion)

Sie können Athena nun mit einem oder mehreren Apache-Hive-Metastores verbinden, zusätzlich zu AWS Glue Data Catalog mit Athena.

Metastore Konnektor

Um eine Verbindung zu einem selbst gehosteten Hive Metastore herzustellen, benötigen Sie einen Athena-Hive-Metastore-Konnektor. Athena bietet einen Referenz-Implementierungs-Konnektor, den Sie verwenden können. Der Connector wird als AWS Lambda Funktion in Ihrem Konto ausgeführt. Weitere Informationen finden Sie unter Verwenden des Athena Data Konnektor für den externen Hive-Metastore (Vorversion).

Verfügbarkeit der Vorversion

Das Feature „Hive Metastore“ ist in der Region USA Ost (Nord-Virginia) als Vorversion verfügbar.

Nächste Schritte

- Befolgen Sie zum Ausprobieren der Vorversion die Anweisungen in Häufig gestellten Fragen zu Vorschaufeatures in Athena.
- Weitere Informationen zu diesem Feature finden Sie unter Verwenden des Athena Data Konnektor für den externen Hive-Metastore (Vorversion).

Neue abfragebezogene Metriken

Athena veröffentlicht jetzt zusätzliche Abfragemetriken, die Ihnen helfen können, die Leistung von Amazon Athena zu verstehen. Athena veröffentlicht abfragebezogene Metriken auf Amazon. CloudWatch In dieser Version veröffentlicht Athena die folgenden zusätzlichen Abfragemetriken:

- Query Planning Time (Abfrageplanungszeit) – Die Zeit, die zum Planen der Abfrage benötigt wird. Dies enthält die Zeit, die zum Abrufen von Tabellenpartitionen aus der Datenquelle benötigt wurde.
- Query Queuing Time (Abfragewartezeit) – Die Zeit, zu der sich die Abfrage in einer Warteschlange befand und auf Ressourcen wartete.
- Service Processing Time (Serviceverarbeitungszeit) – Die Zeit, die zum Schreiben der Ergebnisse benötigt wird, nachdem die Abfrage-Engine ihre Verarbeitung beendet hat.
- Total Execution Time (Gesamtausführungszeit) – Die Zeit, die Athena zum Ausführen der Abfrage gebraucht hat.

Um diese neuen Abfragemetriken zu nutzen, können Sie benutzerdefinierte Dashboards erstellen, Alarme und Auslöser für Metriken einrichten oder vorausgefüllte Dashboards direkt von der Athena-Konsole aus verwenden. CloudWatch

Nächste Schritte

Weitere Informationen finden Sie unter Überwachen von Athena-Abfragen mit CloudWatch Metriken.

12. November 2019

Veröffentlicht am 12.17.2019

Amazon Athena ist nun in der Region Naher Osten (Bahrain) verfügbar.

8. November 2019

Veröffentlicht am 12.17.2019

Amazon Athena ist jetzt in der Region USA West (Nordkalifornien) und Europa (Paris) verfügbar.

8. Oktober 2019

Veröffentlicht am 12.17.2019

Amazon Athena ermöglicht es Ihnen jetzt, über einen VPC-Schnittstellenendpunkt in Ihrer Virtual Private Cloud (VPC) eine direkte Verbindung zu Athena herzustellen. Mit diesem Feature können Sie Ihre Abfragen sicher an Athena senden und benötigen dazu keinen Internet-Gateway in Ihrer VPC.

Um einen VPC-Schnittstellen-Endpunkt für die Verbindung mit Athena zu erstellen, können Sie AWS Management Console oder AWS Command Line Interface ()AWS CLI verwenden. Informationen zum Erstellen eines Schnittstellendpunkts finden Sie unter Erstellen eines Schnittstellenendpunkts.

Wenn Sie einen VPC-Endpunkt mit Schnittstelle verwenden, ist die Kommunikation zwischen Ihrer VPC und den Athena-APIs sicher und verbleibt im Netzwerk. AWS Für die Nutzung dieses Features fallen keine zusätzlichen Athena-Kosten an. Es fallen Gebühren für VPC-Schnittstellenendpunkte an.

Weitere Informationen zu diesem Feature finden Sie unter Herstellen einer Verbindung mit Amazon Athena über einen Schnittstellen-VPC-Endpunkt.

19. September 2019

Veröffentlicht am 12.17.2019

Amazon Athena fügt Unterstützung für das Einfügen neuer Daten in eine vorhandene Tabelle mithilfe der `INSERT INTO`-Anweisung hinzu. Sie können auf Basis einer `SELECT`-Abfrageanweisung, die für eine Quelltabelle ausgeführt wird, oder auf Basis einer Reihe von Werten, die im Rahmen der Abfrageanweisung bereitgestellt werden, neue Zeilen in eine Zieltabelle einfügen. Die folgenden Datenformate werden unterstützt: Avro, JSON, ORC, Parquet und Textdateien.

`INSERT INTO`-Anweisungen können Sie außerdem dabei unterstützen, Ihren ETL-Prozess zu vereinfachen. So können Sie beispielsweise `INSERT INTO` in einer einzelnen Abfrage verwenden, um Daten aus einer Quelldatei im JSON-Format auszuwählen und in eine Zieltabelle im Parquet-Format zu schreiben.

`INSERT INTO`-Anweisungen werden basierend auf der Anzahl der Bytes berechnet, die in der `SELECT`-Phase gescannt werden, ähnlich wie Athena Gebühren für `SELECT`-Abfragen erhebt. Weitere Informationen hierzu finden Sie unter Preise zu Amazon Athena.

Weitere Informationen zur Verwendung`INSERT INTO`, einschließlich unterstützter Formate SerDes und Beispiele, finden Sie unter INSERT INTO im Athena-Benutzerhandbuch.

12. September 2019

Veröffentlicht am 12.17.2019

Amazon Athena ist jetzt in der Region Asien-Pazifik (Hongkong) verfügbar.

16. August 2019

Veröffentlicht am 12.17.2019

Amazon Athena unterstützt die Abfrage von Daten in Amazon-S3-Buckets mit Zahlung durch den Anforderer.

Wenn ein Amazon-S3-Bucket mit Zahlung durch den Anforderer konfiguriert ist, zahlt der Anforderer, nicht der Bucket-Eigentümer, die Kosten für die Amazon-S3-Anforderung und die Datenübertragung. In Athena können Arbeitsgruppenadministratoren nun Arbeitsgruppeneinstellungen konfigurieren, damit Arbeitsgruppenmitglieder S3-Buckets mit Zahlung durch den Anforderer abfragen können.

Informationen zum Konfigurieren der Einstellung der Zahlung durch den Anforderer (Requester Pays) für Ihre Arbeitsgruppe finden Sie unter Create a Workgroup (Erstellen einer Arbeitsgruppe) im Benutzerhandbuch von Amazon Athena. Weitere Informationen zu Buckets mit Zahlung durch den Anforderer finden Sie unter Buckets mit Zahlung durch Auftraggeber im Entwicklerhandbuch von Amazon Simple Storage Service.

9. August 2019

Veröffentlicht am 12.17.2019

Amazon Athena unterstützt jetzt die Durchsetzung von AWS Lake Formation-Richtlinien für eine detaillierte Zugriffskontrolle auf neue oder bestehende Datenbanken, Tabellen und Spalten, die im AWS Glue Data Catalog für in Amazon S3 gespeicherte Daten definiert sind.

Sie können diese Funktion in den folgenden Bereichen verwenden AWS-Regionen: USA Ost (Ohio), USA Ost (Nord-Virginia), USA West (Oregon), Asien-Pazifik (Tokio) und Europa (Irland). Für die Verwendung dieses Features fallen keine zusätzlichen Gebühren an.

Weitere Informationen zur Verwendung dieses Features finden Sie unter Verwenden von Athena zum Abfragen von Daten, die in AWS Lake Formationregistriert sind. Mehr über AWS Lake Formation erfahren Sie unter AWS Lake Formation.

26. Juni 2019

Amazon Athena ist jetzt in der Region Europa (Stockholm) erhältlich. Eine Liste der unterstützten Regionen finden Sie unter AWS-Regionen und Endpunkte.

24. Mai 2019

Veröffentlicht am 24.05.2019

Amazon Athena ist jetzt in den Regionen AWS GovCloud (USA Ost) und AWS GovCloud (USA West) verfügbar. Eine Liste der unterstützten Regionen finden Sie unter AWS-Regionen und Endpunkte.

5. März 2019

Veröffentlicht am 05.03.2019

Amazon Athena jetzt in der Region Kanada (Zentral) verfügbar Eine Liste der unterstützten Regionen finden Sie unter AWS-Regionen und Endpunkte. Veröffentlichung der neuen Version des ODBC-Treibers mit Unterstützung für Athena-Arbeitsgruppen. Weitere Informationen finden Sie in den Versionshinweisen für ODBC-Treiber.

Informationen zum Herunterladen der ODBC-Treiberversion 1.0.5 und der zugehörigen Dokumentation finden Sie unter Herstellen einer Verbindung mit Amazon Athena mit ODBC. Informationen zu dieser Version finden Sie unter Versionshinweise für ODBC-Treiber.

Stellen Sie zur Verwendung von Arbeitsgruppen mit dem ODBC-Treiber die neue Verbindungseigenschaft, `Workgroup`, wie im folgenden Beispiel gezeigt in der Verbindungszeichenfolge ein:

```
Driver=Simba Athena ODBC
 Driver;AwsRegion=[Region];S3OutputLocation=[S3Path];AuthenticationType=IAM
 Credentials;UID=[YourAccessKey];PWD=[YourSecretKey];Workgroup=[WorkgroupName]
```

Suchen Sie für weitere Informationen nach „Arbeitsgruppe“ im Installations- und Konfigurationshandbuch für ODBC-Treiber Version 1.0.5. Es werden keine Änderungen an der Verbindungszeichenfolge des ODBC-Treibers vorgenommen, wenn Sie Tags in Arbeitsgruppen verwenden. Um Tags zu verwenden, aktualisieren Sie auf die neueste Version des ODBC-Treibers (die aktuelle Version).

Diese Treiberversion ermöglicht Ihnen die Verwendung von Athena-API-Arbeitsgruppen-Aktionen zum Erstellen und Verwalten von Arbeitsgruppen und Athena-API-Tag-Aktionen zum Hinzufügen,

Auflisten oder Entfernen von Tags auf Arbeitsgruppen. Stellen vor Beginn sicher, dass Sie über Berechtigungen auf Ressourcenebene in IAM für Aktionen auf Arbeitsgruppen und Tags verfügen.

Weitere Informationen finden Sie unter:

- Verwenden von Arbeitsgruppen zum Ausführen von Abfragen und Beispiel-Arbeitsgruppenrichtlinien.
- Markieren von Athena-Ressourcen und Tagbasierte IAM-Zugriffssteuerungsrichtlinien.

Wenn Sie den JDBC-Treiber oder das AWS SDK verwenden, aktualisieren Sie auf die neueste Version des Treibers und des SDK, die beide bereits Unterstützung für Arbeitsgruppen und Tags in Athena bieten. Weitere Informationen finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC.

22. Februar 2019

Veröffentlicht am 22.02.2019

Hinzufügung der Tag-Unterstützung für Arbeitsgruppen in Amazon Athena. Ein Tag besteht aus einem Schlüssel und einem Wert, die Sie beide selbst definieren können. Wenn Sie eine Arbeitsgruppe markieren, weisen Sie ihr benutzerdefinierte Metadaten zu. Sie können Arbeitsgruppen Stichwörter hinzufügen, um sie anhand von bewährten Methoden besser kategorisieren zu können. AWS Sie können mit Tags den Zugriff auf Arbeitsgruppen einschränken und Kosten nachverfolgen. Erstellen Sie beispielsweise eine Arbeitsgruppe für jede Kostenstelle. Wenn Sie diesen Arbeitsgruppen dann Tags hinzufügen, können Sie Ihre Athena-Ausgaben für jede Kostenstelle nachverfolgen. Weitere Informationen finden Sie unter Verwendung von Tags für die Fakturierung im AWS Billing and Cost Management -Benutzerhandbuch.

Sie können mit Tags arbeiten, indem Sie die Athena-Konsole oder die API-Vorgänge verwenden. Weitere Informationen finden Sie unter Markieren von Athena-Ressourcen.

In der Athena-Konsole können Sie jeder Ihrer Arbeitsgruppen einen oder mehrere Tags hinzufügen und dann nach Tags suchen. Arbeitsgruppen sind eine von IAM kontrollierte Ressource in Athena. Sie können in IAM einschränken, wer Tags auf einer von Ihnen erstellten Arbeitsgruppe hinzufügen, entfernen oder auflisten kann. Sie können auch den API-Vorgang `CreateWorkGroup` verwenden, die den optionalen Tag-Parameter für die Hinzufügung eines oder mehrerer Tags zu der Arbeitsgruppe enthält. Verwenden Sie zum Hinzufügen, Entfernen oder Auflisten von Tags `TagResource`, `UntagResource` und `ListTagsForResource`. Weitere Informationen finden Sie unter Verwenden von Tag-Operationen.

Um Benutzern zu erlauben, beim Erstellen von Arbeitsgruppen tags hinzuzufügen, stellen Sie sicher, dass Sie jedem Benutzer IAM-Berechtigungen für die API-Aktionen `TagResource` und `CreateWorkGroup` gewähren. Weitere Informationen und Beispiele finden Sie unter Tagbasierte IAM-Zugriffssteuerungsrichtlinien.

Wenn Sie Tags für Arbeitsgruppen verwenden, gibt es keine Änderungen am JDBC-Treiber. Wenn Sie neue Arbeitsgruppen erstellen und den JDBC-Treiber oder das SDK verwenden, führen Sie ein Upgrade auf die neueste Version des Treibers und des AWS SDK durch. Weitere Informationen finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC.

18. Februar 2019

Veröffentlicht am 18.02.2019

Hinzufügung der Möglichkeit zur Kontrolle der Abfragekosten durch die Ausführung von Abfragen in Arbeitsgruppen. Weitere Informationen finden Sie unter Verwendung von Arbeitsgruppen zur Kontrolle des Abfragenzugriffs und der Kosten. Das in Athena SerDe verwendete JSON OpenX wurde verbessert, ein Problem behoben, bei dem Athena Objekte, die in die `GLACIER` Speicherklasse umgestellt wurden, nicht ignorierte, und es wurden Beispiele für die Abfrage von Network Load Balancer Balancer-Protokollen hinzugefügt.

Durchführung der folgenden Änderungen:

- Hinzufügung der Unterstützung für Arbeitsgruppen. Verwenden Sie Arbeitsgruppen zum Trennen von Benutzern, Teams, Anwendungen oder Workloads sowie zur Einrichtung von Grenzwerten für die Datenmenge, die jede Abfrage oder die gesamte Arbeitsgruppe verarbeiten kann. Da Arbeitsgruppen als IAM-Ressourcen fungieren, können Sie Berechtigungen auf Ressourcenebene verwenden, um den Zugriff auf eine bestimmte Arbeitsgruppe zu steuern. Sie können auch abfragebezogene Metriken in Amazon anzeigen CloudWatch, die Abfragekosten kontrollieren, indem Sie Grenzwerte für die Menge der gescannten Daten konfigurieren, Schwellenwerte erstellen und Aktionen wie Amazon SNS SNS-Alarme auslösen, wenn diese Schwellenwerte überschritten werden. Weitere Informationen finden Sie unter Verwenden von Arbeitsgruppen zum Ausführen von Abfragen und Steuern von Kosten und Überwachen von Abfragen mit CloudWatch Metriken und Ereignissen.

  Arbeitsgruppen sind IAM-Ressourcen. Eine vollständige Liste der arbeitsgruppenbezogenen Aktionen, Ressourcen und Bedingungen in IAM finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Athena in der Serviceautorisierungsreferenz. Bevor Sie neue

Arbeitsgruppen erstellen, stellen Sie sicher, dass Sie Arbeitsgruppen-IAM-Richtlinien und die AWS Verwaltete Richtlinie: AmazonAthenaFullAccess verwenden.

Sie können mit den Arbeitsgruppen-API-Vorgängen oder mit dem JDBC-Treiber beginnen, Arbeitsgruppen zu verwenden. Ein fortgeschrittenes Verfahren finden Sie unter Einrichten von Arbeitsgruppen. Für den Download des JDBC-Treibers mit Unterstützung für Arbeitsgruppen vgl. Herstellen einer Verbindung zu Amazon Athena mit JDBC.

Wenn Sie Arbeitsgruppen mit dem JDBC-Treiber verwenden, müssen Sie den Arbeitsgruppennamen in der Verbindungszeichenfolge mithilfe des `Workgroup`-Konfigurationsparameters wie im folgenden Beispiel dargestellt einrichten:

```
jdbc:awsathena://AwsRegion=<AWSREGION>;UID=<ACCESSKEY>;
PWD=<SECRETKEY>;S3OutputLocation=s3://<athena-output>-<AWSREGION>/;
Workgroup=<WORKGROUPNAME>;
```

Es gibt keine Änderungen bei der Art und Weise, wie Sie SQL-Anweisungen ausführen oder JDBC-API-Aufrufe an den Treiber durchführen. Der Treiber übergibt den Namen der Arbeitsgruppe an Athena.

Informationen zu den bei Arbeitsgruppen eingeführten Unterschieden finden Sie unter Athena-Arbeitsgruppen-APIs und Fehlerbehebung bei Arbeitsgruppen.

- Das in Athena SerDe verwendete JSON OpenX wurde verbessert. Die Verbesserungen beinhalten u. a. Folgendes:
  - Unterstützung für die Eigenschaft `ConvertDotsInJsonKeysToUnderscores`. Wenn diese Option auf gesetzt ist`TRUE`, können SerDe die Punkte in Schlüsselnamen durch Unterstriche ersetzt werden. Wenn der JSON-Datensatz beispielsweise einen Schlüssel mit dem Namen `"a.b"` enthält, können Sie diese Eigenschaft verwenden, um den Spaltennamen als `"a_b"` in Athena zu definieren. Der Standardwert ist `FALSE`. Standardmäßig erlaubt Athena keine Punkte in Spaltennamen.
  - Unterstützung für die Eigenschaft `case.insensitive`. Standardmäßig erfordert Athena, dass alle Schlüssel in Ihrem JSON-Datensatz Kleinbuchstaben verwenden. Mit `WITH SERDE PROPERTIES ("case.insensitive"= FALSE;)` können Sie Schlüsselnamen verwenden, bei denen Groß-/Kleinschreibung bedeutsam ist. Der Standardwert ist `TRUE`. Bei Einstellung auf werden `TRUE` alle Großbuchstabenspalten in Kleinbuchstaben SerDe umgewandelt.

  Weitere Informationen finden Sie unter OpenX JSON SerDe.

- Es wurde ein Problem behoben, bei dem Athena `"access denied"`-Fehlermeldungen zurückgab, wenn es Amazon-S3-Objekte verarbeitete, die durch Amazon-S3-Lebenszyklusrichtlinien in Glacier archiviert wurden. Als Ergebnis der Behebung dieses Problems ignoriert Athena Objekte, die in die `GLACIER`-Speicherklasse überführt wurden. Athena unterstützt das Abfragen von Daten aus der `GLACIER`-Speicherklasse nicht.

  Weitere Informationen finden Sie unter the section called “Voraussetzungen für Tabellen in Athena und Daten in Amazon S3” und Übergang zur GLACIER-Speicherklasse (Objektarchivierung) im Benutzerhandbuch für Amazon Simple Storage Service.
- Beispiele für das Abfragen von Network-Load-Balancer-Zugriffsprotokollen, die Informationen zu den Anforderungen von Transport Layer Security (TLS) erhalten, wurden hinzugefügt. Weitere Informationen finden Sie unter the section called “Network Load Balancer”.

Athena-Versionshinweise für 2018

20. November 2018

Veröffentlicht am 20.11.2018

Die neuen Versionen der JDBC- und ODBC-Treiber mit Unterstützung für Verbundzugriff auf die Athena-API mit AD FS und SAML 2.0 (Security Assertion Markup Language 2.0) wurden veröffentlicht. Einzelheiten können Sie den Versionshinweisen für JDBC-Treiber und den Versionshinweisen für ODBC-Treiber entnehmen.

Ab dieser Version wird der Verbundzugriff auf Athena für Active Directory Federation Service (AD FS 3.0) unterstützt. Der Zugriff wird durch die Versionen von JDBC- bzw. ODBC-Treibern hergestellt, die SAML 2.0 unterstützen. Informationen zur Konfiguration des Verbundzugriffs auf die Athena-API finden Sie unter the section called “Aktivieren des föderierten Zugriffs auf die Athena-API”.

Informationen zum Herunterladen der JDBC-Treiberversion 2.0.6 und der zugehörigen Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC. Informationen zu dieser Version finden Sie unter Versionshinweise für JDBC-Treiber.

Informationen zum Herunterladen der ODBC-Treiberversion 1.0.4 und der zugehörigen Dokumentation finden Sie unter Herstellen einer Verbindung mit Amazon Athena mit ODBC. Informationen zu dieser Version finden Sie unter Versionshinweise für ODBC-Treiber.

Weitere Informationen zur SAML 2.0-Unterstützung finden Sie unter AWSAbout SAML 2.0 Federation im IAM-Benutzerhandbuch.

15. Oktober 2018

Veröffentlicht am 15.10.2018

Wenn Sie ein Upgrade auf die durchgeführt haben AWS Glue Data Catalog, gibt es zwei neue Funktionen, die Unterstützung bieten für:

- Verschlüsselung der Datenkatalog-Metadaten. Wenn Sie Metadaten im Datenkatalog verschlüsseln möchten, müssen Sie Athena bestimmte Richtlinien hinzufügen. Weitere Informationen finden Sie unter Zugriff auf verschlüsselte Metadaten in AWS Glue Data Catalog.
- Detaillierte Berechtigungen für den Zugriff auf Ressourcen in der. AWS Glue Data Catalog
  Sie können jetzt identitätsbasierte (IAM)-Richtlinien definieren, um den Zugriff auf spezifische Datenbanken und Tabellen über den in Athena verwendeten Datenkatalog einzuschränken oder zuzulassen. Weitere Informationen finden Sie unter Differenzierter Zugriff auf Datenbanken und Tabellen in AWS Glue Data Catalog.

  > **Note**
  >
  > Die Daten befinden sich in den Amazon-S3-Buckets, und der Zugriff darauf wird von Zugriff auf Amazon S3 gesteuert. Für den Zugriff auf Datenbanken und Tabellen verwenden Sie weiterhin Zugriffskontrollrichtlinien für Amazon-S3-Buckets, in denen die Daten gespeichert sind.

10. Oktober 2018

Veröffentlicht am 10.10.2018

Athena unterstützt `CREATE TABLE AS SELECT`, womit eine Tabelle aus dem Ergebnis einer `SELECT`-Abfrageanweisung erstellt wird. Weitere Informationen finden Sie unter Erstellen einer Tabelle aus Abfrageergebnissen (CTAS).

Bevor Sie CTAS-Abfragen erstellen, ist es wichtig, mehr über ihr Verhalten in der Athena-Dokumentation zu erfahren. Sie enthält Informationen über den Speicherort für Abfrageergebnisse in Amazon S3, der Liste der unterstützten Formate für das Speichern von CTAS-Abfrageergebnissen,

die Anzahl der Partitionen, die Sie erstellen können, und unterstützte Komprimierungsformate. Weitere Informationen finden Sie unter Überlegungen und Einschränkungen für CTAS-Abfragen.

Verwenden Sie CTAS-Abfragen, um:

- Erstellen Sie eine Tabelle von Abfrageergebnisse in einem Schritt.
- Erstellen Sie CTAS-Abfragen in der Athena-Konsole unter Verwendung von Beispielen. Weitere Informationen zur Syntax finden Sie unter CREATE TABLE AS.
- Transformieren Sie Abfrageergebnisse in andere Dateiformate wie PARQUET, ORC, AVRO, JSON und TEXTFILE. Weitere Informationen finden Sie unter Überlegungen und Einschränkungen für CTAS-Abfragen und Spaltenbasierte Speicherformate.

6. September 2018

Veröffentlicht am 06.09.2018

Neue Version des ODBC-Treibers (Version 1.0.3) wurde freigegeben. In der neuen Version des ODBC-Treibers werden Ergebnisse standardmäßig gestreamt, anstatt sie zu paginieren. Auf diese Weise können Business Intelligence-Tools schneller große Datensätze abrufen. Diese Version enthält auch Verbesserungen, Fehlerbehebungen und eine aktualisierte Dokumentation für "Verwenden von SSL mit einem Proxy-Server". Weitere Informationen finden Sie in den Versionshinweisen zum Treiber.

Informationen zum Herunterladen der ODBC-Treiberversion 1.0.3 und der zugehörigen Dokumentation finden Sie unter Herstellen einer Verbindung mit Amazon Athena mit ODBC.

Das Streamingfeature für Ergebnisse ist mit dieser neuen Version des ODBC-Treibers verfügbar. Sie ist auch mit dem JDBC-Treiber verfügbar. Informationen zu Streaming-Ergebnissen finden Sie im ODBC-Treiberinstallations- und Konfigurationshandbuch. Suchen Sie dort nach. UseResultsetStreaming

Die ODBC-Treiberversion 1.0.3 ist ein Drop-In-Ersatz für die vorherige Version des Treibers. Es wird empfohlen, zum aktuellen Treiber zu migrieren.

> **Important**
>
> Um den ODBC-Treiber der Version 1.0.3 verwenden zu können, müssen Sie die folgenden Anforderungen erfüllen:

- Behalten Sie den Port 444 für ausgehenden Datenverkehr offen.
- Fügen Sie die `athena:GetQueryResultsStream`-Richtlinienaktion zur Liste der Richtlinien für Athena hinzu. Diese Richtlinienaktion wird nicht direkt mit der API verfügbar gemacht und wird nur mit den ODBC- und JDBC-Treibern im Rahmen der Unterstützung für das Streaming von Ergebnissen verwendet. Eine Beispielrichtlinie finden Sie unter AWS Verwaltete Richtlinie: AWSQuicksightAthenaAccess.

23. August 2018

Veröffentlicht am 23.08.2018

Unterstützung für diese DDL-bezogenen Features hinzugefügt und folgende Fehler behoben:

- Unterstützung für die Datentypen `BINARY` und `DATE` für Daten in Parquet sowie für die Datentypen `DATE` und `TIMESTAMP` für Daten in Avro hinzugefügt.
- Unterstützung für `INT` und `DOUBLE` in DDL-Abfragen hinzugefügt. `INTEGER` ist ein Alias für `INT` und `DOUBLE PRECISION` ist ein Alias für `DOUBLE`.
- Verbesserung der Leistung von `DROP TABLE`- und `DROP DATABASE`-Anfragen.
- Die `_$folder$`-Objekterstellung in Amazon S3 wurde entfernt, wenn ein Daten-Bucket leer ist.
- Behebung eines Problems, bei dem `ALTER TABLE ADD PARTITION` eine Fehlermeldung ausgab, wenn kein Partitionswert angegeben wurde.
- Behebung eines Problems, bei dem `DROP TABLE` den Datenbanknamen bei der Überprüfung von Partitionen ignorierte, nachdem der qualifizierte Name in der Anweisung angegeben wurde.

Weitere Informationen zu den in Athena unterstützten Datentypen finden Sie unter Datentypen in Amazon Athena.

Informationen zu den unterstützten Datentyp-Mappings zwischen den Typen in Athena, dem JDBC-Treiber und den Java-Datentypen finden Sie im Abschnitt Datentypen im Installations- und Konfigurationsleitfaden für JDBC-Treiber.

16. August 2018

Veröffentlicht am 16.08.2018

JDBC-Treiberversion 2.0.5 wurde freigegeben. In der neuen Version des JDBC-Treibers werden Ergebnisse standardmäßig gestreamt, anstatt sie zu paginieren. Auf diese Weise können Business Intelligence-Tools schneller große Datensätze abrufen. Im Vergleich mit der vorherigen Version des JDBC-Treibers gibt es die folgenden Leistungsverbesserungen:

- Leistungssteigerung um ungefähr das 2-Fache beim Abrufen von weniger als 10.000 Zeilen
- Leistungssteigerung um ungefähr das 5- bis 6-Fache beim Abrufen von mehr als 10.000 Zeilen

Das Streamingfeature für Ergebnisse ist nur für den JDBC-Treiber verfügbar. Für den ODBC-Treiber ist sie nicht verfügbar. Eine Verwendung mit der Athena-API ist nicht möglich. Informationen zu Streaming-Ergebnissen finden Sie im Installations- und Konfigurationshandbuch für JDBC-Treiber. Suchen Sie dort nach. UseResultsetStreaming

Informationen zum Herunterladen der JDBC-Treiberversion 2.0.5 und der zugehörigen Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC.

Die JDBC-Treiberversion 2.0.5 ist ein Drop-In-Ersatz für die vorherige Version des Treibers (2.0.2). Um sicherzustellen, dass Sie die JDBC-Treiberversion 2.0.5 verwenden können, fügen Sie die `athena:GetQueryResultsStream`-Richtlinienaktion zur Liste der Richtlinien für Athena hinzu. Diese Richtlinienaktion wird nicht direkt mit der API verfügbar gemacht und wird nur mit dem JDBC-Treiber verwendet, im Rahmen des Supports für das Streaming von Ergebnissen. Eine Beispielrichtlinie finden Sie unter AWS Verwaltete Richtlinie: AWSQuicksightAthenaAccess. Weitere Informationen zur Migration von Version 2.0.2 zu Version 2.0.5 des Treibers finden Sie unter Migrationshandbuch für den JDBC-Treiber.

Wenn Sie von einem 1.x-Treiber zu einem 2.x Treiber migrieren, müssen Sie Ihre bestehenden Konfigurationen zur neuen Konfiguration migrieren. Es wird dringend empfohlen, zur aktuellen Treiberversion zu migrieren. Weitere Informationen finden Sie im Migrationshandbuch für JDBC-Treiber.

7. August 2018

Veröffentlicht am 07.08.2018

Sie können jetzt Flow-Protokolle von Amazon Virtual Private Cloud in einem GZIP-Format direkt in Amazon S3 speichern. Hier können Sie die in Athena abfragen. Weitere Informationen finden Sie unter Abfragen von Amazon-VPC-Flow-Protokollen und Amazon VPC Flow Logs können nun an S3 übermittelt werden.

5. Juni 2018

Veröffentlicht am 05.06.2018

Themen

Unterstützung für Ansichten

Unterstützung für Ansichten hinzugefügt. Sie können jetzt CREATE VIEW, DESCRIBE VIEW, DROP VIEW, SHOW CREATE VIEW und SHOW VIEWS in Athena verwenden. Die Abfrage, die die Ansicht definiert, wird jedes Mal ausgeführt, wenn Sie in einer Abfrage auf die Ansicht verweisen. Weitere Informationen finden Sie unter Arbeiten mit Ansichten.

Verbesserungen und Aktualisierungen bei Fehlermeldungen

- Es wurde eine GSON 2.8.0-Bibliothek hinzugefügt CloudTrail SerDe, um ein Problem mit der zu lösen CloudTrail SerDe und das Parsen von JSON-Zeichenketten zu ermöglichen.
- Verbesserte Validierung des Partitionsschemas in Athena für Parquet und in einigen Fällen für ORC, indem die Umsortierung von Spalten ermöglicht wurde. Dadurch kann Athena besser mit Änderungen in der Schemaentwicklung im Laufe der Zeit und mit Tabellen umgehen, die vom AWS Glue Crawler hinzugefügt wurden. Weitere Informationen finden Sie unter Verarbeiten von Schema-Updates.
- Unterstützung zur Verarbeitung für `SHOW VIEWS` hinzugefügt.
- Folgende Verbesserungen an den häufigsten Fehlermeldungen wurden vorgenommen:
  - Eine interne Fehlermeldung wurde durch eine beschreibende Fehlermeldung ersetzt, wenn die Spalte in SerDe einer Athena-Abfrage nicht analysiert werden kann. Zuvor löste Athena bei Analysefehlern einen internen Fehler aus. Die neue Fehlermeldung lautet: „HIVE_BAD_DATA: Fehler beim Verarbeiten des Feldwerts für das Feld 0: java.lang.String kann nicht nach org.openx.data.jsonserde.json.JSONObject umgewandelt werden“.
  - Verbesserte Fehlermeldungen über unzureichende Berechtigungen durch Hinzufügen weiterer Details.

Fehlerbehebungen

Folgende Fehler wurden behoben:

- Es wurde ein Problem behoben, das dazu führte, dass die interne Übersetzung der Datentypen `REAL` nach `FLOAT` ermöglichte. Dies verbessert die Integration mit dem AWS Glue Crawler, der `FLOAT`-Datentypen zurückgibt.
- Behebung eines Problems, durch das Athena AVRO `DECIMAL` (einen logischen Typ) nicht in einen `DECIMAL`-Typ umwandelte.
- Behebung eines Problems, bei dem Athena keine Ergebnisse für Abfragen für Parquet-Daten mit `WHERE`-Klauseln zurückgab, die auf Werte im `TIMESTAMP`-Datentyp verwiesen.

17. Mai 2018

Veröffentlicht am 17.05.2018

Das Kontingent für die Gleichzeitigkeit von Abfragen in Athena wurde von fünf auf zwanzig erhöht. Das bedeutet: Sie können bis zu 20 `DDL`-Abfragen und 20 `SELECT`-Abfragen gleichzeitig absenden und ausführen. Hinweis: Die Kontingente für gleichzeitige Abfragen gelten separat für `DDL`- und `SELECT`-Abfragen.

Kontingente für gleichzeitige Abfragen in Athena sind definiert als die Anzahl der Abfragen, die gleichzeitig an den Service übermittelt werden können. Sie können bis zu 20 Abfragen des gleichen Typs (`DDL` oder `SELECT`) gleichzeitig absenden. Wenn Sie eine Abfrage übermitteln, die das Kontingent für gleichzeitige Abfragen überschreitet, zeigt die Athena-API eine Fehlermeldung an.

Nachdem Sie die Abfragen an Athena übermittelt haben, werden diese verarbeitet, indem Ressourcen je nach der Gesamtauslastung des Services sowie der Anzahl eingehender Anfragen zugewiesen werden. Wir überwachen den Service fortlaufend und passen ihn an, damit Ihre Abfragen so schnell wie möglich verarbeitet werden.

Weitere Informationen finden Sie unter Service Quotas. Dies ist ein anpassbares Kontingent. Sie können die Service-Quotas-Konsole verwenden, um eine Kontingenterhöhung für gleichzeitige Abfragen anzufordern.

19. April 2018

Veröffentlicht am 19.04.2018

Es wurde die neue Version des JDBC-Treibers (Version 2.0.2) mit Unterstützung für die Rückgabe der `ResultSet`-Daten als Array-Datentyp sowie mit Verbesserungen und Fehlerbehebungen freigegeben. Weitere Informationen finden Sie in den Versionshinweisen zum Treiber.

Informationen zum Herunterladen der neuen JDBC-Treiberversion 2.0.2 und der zugehörigen Dokumentation finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC.

Die aktuelle Version des JDBC-Treibers ist 2.0.2. Wenn Sie von einem 1.x-Treiber zu einem 2.x Treiber migrieren, müssen Sie Ihre bestehenden Konfigurationen zur neuen Konfiguration migrieren. Es wird dringend empfohlen, zum aktuellen Treiber zu migrieren.

Informationen zu den Änderungen an der neuen Treiberversion, zu den Versionsunterschieden sowie Beispiele finden Sie im Migrationsleitfaden für den JDBC-Treiber.

6. April 2018

Veröffentlicht am 06.04.2018

Verwenden Sie die automatische Vervollständigung, um Abfragen in die Athena-Konsole einzugeben.

15. März 2018

Veröffentlicht am 15.03.2018

Es wurde die Möglichkeit hinzugefügt, automatisch Athena-Tabellen für CloudTrail Protokolldateien direkt von der CloudTrail Konsole aus zu erstellen. Weitere Informationen finden Sie unter Verwenden der CloudTrail Konsole zum Erstellen einer Athena-Tabelle für CloudTrail Protokolle .

2. Februar 2018

Veröffentlicht am 12.02.2018

Es wurde eine Möglichkeit für das Auslagern von Zwischendaten auf Festplatte bei arbeitsspeicherintensiven Abfragen, die die `GROUP BY`-Klausel verwenden, hinzugefügt. Dadurch wird die Zuverlässigkeit solcher Abfragen verbessert und die Fehlermeldung "Query resource exhausted" (Abfrage-Ressource aufgebraucht) vermieden.

19. Januar 2018

Veröffentlicht am 19.01.2018

Athena verwendet Presto, eine Open-Source-basierte verteilte Abfrage-Engine, zum Ausführen von Abfragen.

Bei Athena sind keine Versionen zu verwalten. Wir haben für die zugrunde liegende Engine in Athena ein transparentes Upgrade auf eine Version ausgeführt, die auf Presto-Version 0.172 basiert. Von Ihrer Seite aus ist keine Aktion erforderlich.

Mit dem Upgrade können Sie nun Funktionen und Operatoren von Presto 0.172 einschließlich der Lambda-Ausdrücke von Presto 0.172 in Athena nutzen.

Zu den wichtigen Aktualisierungen dieser Version, darunter auch von der Community entwickelte Fehlerbehebungen, zählen folgende:

- Unterstützung für das Ignorieren von Headern. Mithilfe der Eigenschaft `skip.header.line.count` können Sie im Rahmen der Tabellendefinition festlegen, dass Athena die Header ignorieren darf. Dies wird für Abfragen unterstützt, die OpenCSV LazySimpleSerDeund nicht für Grok SerDe oder Regex verwenden. SerDes
- Unterstützung für den Datentyp `CHAR(n)` in `STRING`-Funktionen. Der Bereich für `CHAR(n)` ist `[1.255]`, wohingegen `VARCHAR(n)` den Bereich `[1,65535]` aufweist.
- Unterstützung für korrelierte Unterabfragen.
- Unterstützung für Lambda-Ausdrücke und Funktionen von Presto.
- Verbesserte Leistung von `DECIMAL`-Datentypen und -Operatoren.
- Unterstützung für gefilterte Aggregationen, wie z. B. `SELECT sum(col_name) FILTER` mit `id > 0`.
- Prädikatweitergabe (Predicate Pushdown) für die Datentypen `DECIMAL`, `TINYINT`, `SMALLINT` und `REAL`.
- Unterstützung für quantifizierte Vergleichsprädikate: `ALL`, `ANY` und `SOME`.
- Hinzugefügte Funktionen: `arrays_overlap()`, `array_except()`, `levenshtein_distance()`, `codepoint()`, `skewness()`, `kurtosis()` und `typeof()`.
- Variante der Funktion `from_unixtime()` hinzugefügt, die ein Zeitzonen-Argument annimmt.
- Aggregationsfunktionen `bitwise_and_agg()` und `bitwise_or_agg()` hinzugefügt.
- Funktionen `xxhash64()` und `to_big_endian_64()` hinzugefügt.
- Unterstützung für die Funktionen `json_extract()` und `json_extract_scalar()` hinzugefügt, damit doppelte Anführungszeichen oder umgekehrte Schrägstriche unter Verwendung eines umgekehrten Schrägstrichs mit einem JSON-Pfadindex als Escape-Zeichen verwendet werden

können. Dadurch wird die Semantik jedes Aufruf mit einem umgekehrten Schrägstrich geändert, da dieser zuvor als reguläres Zeichen galt.

Weitere Informationen zu Funktionen und Operatoren finden Sie unter DML-Abfragen, -Funktionen und -Operatoren in diesem Handbuch sowie unter Funktionen und Operatoren in der Presto-Dokumentation.

Nicht alle Presto-Features werden von Athena unterstützt. Weitere Informationen finden Sie unter Limitations.

Athena-Versionshinweise für 2017

13. November 2017

Veröffentlicht am 13.11.2017

Unterstützung für die Verbindungserstellung von Athena zum ODBC-Treiber hinzugefügt. Weitere Informationen finden Sie unter Herstellen einer Verbindung mit Amazon Athena mit ODBC.

1. November 2017

Veröffentlicht am 01.11.2017

Unterstützung für die Abfrage von koordinatenbasierten Daten und für die Regionen Asien-Pazifik (Seoul), Asien-Pazifik (Mumbai) und EU (London) hinzugefügt. Weitere Informationen finden Sie unter Abfragen von koordinatenbasierten Daten und AWS-Regionen und Endpunkte.

19. Oktober 2017

Veröffentlicht am 19.10.2017

Unterstützung für EU (Frankfurt) hinzugefügt. Eine Liste der unterstützten Regionen finden Sie unter AWS-Regionen und Endpunkte.

3. Oktober 2017

Veröffentlicht am 03.10.2017

Erstellen Sie benannte Athena-Abfragen mit AWS CloudFormation. Weitere Informationen finden Sie AWS::Athena::NamedQueryim AWS CloudFormation Benutzerhandbuch.

25. September 2017

Veröffentlicht am 25.09.2017

Unterstützung für die Region Asien-Pazifik (Sydney) hinzugefügt Eine Liste der unterstützten Regionen finden Sie unter AWS-Regionen und Endpunkte.

14. August 2017

Veröffentlicht am 14.08.2017

Integration mit dem AWS Glue Data Catalog und ein Migrationsassistent für die Aktualisierung vom verwalteten Athena-Datenkatalog auf den AWS Glue Data Catalog hinzugefügt. Weitere Informationen finden Sie unter Integration in AWS Glue.

4. August 2017

Veröffentlicht am 04.08.2017

Unterstützung für Grok wurde hinzugefügt SerDe, wodurch der Musterabgleich für Datensätze in unstrukturierten Textdateien wie Protokollen erleichtert wird. Weitere Informationen finden Sie unter Grok SerDe. Tastenkombinationen zum Durchblättern der Abfrageverläufe mithilfe der Konsole hinzugefügt (STRG+⇧/⇩ bei Windows, CMD+⇧/⇩ bei Mac).

22. Juni 2017

Veröffentlicht am 22.06.2017

Unterstützung für die Regionen Asien-Pazifik (Tokio) und Asien-Pazifik (Singapur) hinzugefügt. Eine Liste der unterstützten Regionen finden Sie unter AWS-Regionen und Endpunkte.

8. Juni 2017

Veröffentlicht am 08.06.2017

Es wurde Unterstützung für Europa (Irland) hinzugefügt. Weitere Informationen finden Sie unter AWS-Regionen und Endpunkte.

19. Mai 2017

Veröffentlicht am 19.05.2017

Amazon Athena-API und AWS CLI Unterstützung für Athena hinzugefügt; JDBC-Treiber auf Version 1.1.0 aktualisiert; verschiedene Probleme behoben.

- Amazon Athena ermöglicht die Anwendungsprogrammierung für Athena. Weitere Informationen finden Sie in der Amazon Athena-API-Referenz. Die neuesten AWS SDKs unterstützen die Athena-API. Links zu Dokumentationen und Downloads finden Sie unter SDKs in den Tools für Amazon Web Services.
- Das AWS CLI beinhaltet neue Befehle für Athena. Weitere Informationen finden Sie in der Amazon-Athena-API-Referenz.
- Der neue JDBC-Treiber 1.1.0 ist verfügbar. Er unterstützt die neue Athena-API sowie die neuesten Features und Fehlerbehebungen. Laden Sie den Treiber unter https://downloads.athena.us-east-1.amazonaws.com/drivers/AthenaJDBC41-1.1.0.jar herunter. Es wird empfohlen, auf die neueste JDBC-Treiberversion für Athena zu aktualisieren, Sie können jedoch auch die vorherige Treiberversion verwenden. Vorherige Treiberversionen bieten keine Unterstützung für die Athena-API. Weitere Informationen finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC.
- Die in früheren Athena-Versionen vorhandenen speziellen Aktionen für Richtlinienanweisungen sind veraltet. Wenn Sie auf die JDBC-Treiberversion 1.1.0 aktualisieren und kundenverwaltete oder eingebundene IAM-Richtlinien mit den JDBC-Benutzern verknüpft haben, müssen Sie die IAM-Richtlinien ebenfalls aktualisieren. Im Gegensatz dazu wird die Athena-API von früheren JDBC-Treiberversionen nicht unterstützt. Daher können Sie in Richtlinien, die mit Benutzern der vorherigen JDBC-Treiberversion verknüpft sind, nur veraltete Aktionen angeben. Aus diesem Grund sollte keine Aktualisierung der kundenverwalteten oder eingebundenen IAM-Richtlinien erfolgen.
- Diese richtlinienspezifischen Aktionen wurden in Athena vor Einführung der Athena-API verwendet. Verwenden Sie diese veralteten Aktionen in Richtlinien nur mit JDBC-Treibern vor der Version 1.1.0. Wenn Sie die JDBC-Treiberversion aktualisieren, müssen Sie Richtlinienanweisungen, die veraltete Aktionen zulassen oder verweigern, entsprechend durch die nachstehend aufgeführten API-Aktionen ersetzen, da andernfalls Fehler auftreten:

Veraltete richtlinienspezifische Aktion	Entsprechende Athena-API-Aktion
`athena:RunQuery`	`athena:StartQueryExecution`

Veraltete richtlinienspezifische Aktion	Entsprechende Athena-API-Aktion
`athena:CancelQueryExecution`	`athena:StopQueryExecution`
`athena:GetQueryExecutions`	`athena:ListQueryExecutions`

Verbesserungen

- Längenbegrenzung der Abfragezeichenfolge wurde auf 256 KB erhöht.

Fehlerbehebungen

- Der Fehler, aufgrund dessen die Abfrageergebnisse beim Blättern durch die Ergebnisse in der Konsole falsch formatiert wirkten, wurde behoben.
- Der Fehler, aufgrund dessen eine `\u0000`-Zeichenfolge in Amazon S3-Datendateien zu Fehlern führte, wurde behoben.
- Der Fehler, aufgrund dessen der angeforderte Abbruch einer Abfrage vom JDBC-Treiber fehlschlug, wurde behoben.
- Es wurde ein Problem behoben, das AWS CloudTrail SerDe dazu führte, dass Amazon S3 S3-Daten in USA Ost (Ohio) fehlschlug.
- Der Fehler, aufgrund dessen `DROP TABLE` für eine partitionierte Tabelle nicht ordnungsgemäß ausgeführt werden konnte, wurde behoben.

4. April 2017

Veröffentlicht am 04.04.2017

Unterstützung für die Amazon S3-Datenverschlüsselung und die veröffentlichte JDBC-Treiberaktualisierung (Version 1.0.1) mit Verschlüsselungsunterstützung sowie Verbesserungen und Fehlerbehebungen hinzugefügt.

Features

- Folgende Verschlüsselungsfeatures wurden hinzugefügt:

  - Unterstützung für Abfragen von verschlüsselten Daten in Amazon S3
  - Unterstützung für die Verschlüsselung von Athena-Abfrageergebnissen
- Eine neue Version des Treibers unterstützt neue Verschlüsselungsfeatures und enthält Verbesserungen sowie Fehlerbehebungen.
- Die Möglichkeit, Spalten mithilfe von `ALTER TABLE` hinzuzufügen, zu ersetzen und zu ändern, wurde hinzugefügt. Weitere Informationen finden Sie unter Alter Column in der Hive-Dokumentation.
- Unterstützung für die Abfrage von LZO-komprimierten Daten wurde hinzugefügt.

Weitere Informationen finden Sie unter Verschlüsselung im Ruhezustand.

Verbesserungen

- Bessere JDBC-Abfrageleistung mit verbesserter Seitengröße, sodass anstelle von 100 Zeilen nun 1.000 zurückgegeben werden.
- Möglichkeit, eine Abfrage über die JDBC-Treiberschnittstelle abzubrechen, hinzugefügt.
- Möglichkeit, JDBC-Optionen in der JDBC-Verbindungs-URL anzugeben, hinzugefügt. Siehe Herstellen einer Verbindung zu Amazon Athena mit JDBC für den aktuellsten JDBC-Treiber.
- PROXY-Einstellung im Treiber hinzugefügt, die jetzt ClientConfigurationim AWS SDK for Java festgelegt werden kann.

Fehlerbehebungen

Folgende Fehler wurden behoben:

- Im Falle mehrerer Abfragen über die JDBC-Treiberschnittstelle traten Ablehnungsfehler auf.
- Die Projektion des Datentyps „decimal“ führte zu einem Abbruch des JDBC-Treibers.
- Der JDBC-Treiber gab jeden Datentyp als Zeichenfolge zurück, und zwar unabhängig von der Definition des Datentyps in der Tabelle. Wenn Sie beispielsweise eine Spalte, für die der Datentyp `INT` definiert war, mit `resultSet.GetObject()` ausgewählt haben, wurde der Datentyp `STRING` anstelle von `INT` zurückgegeben.
- Vom JDBC-Treiber wurden die Anmeldeinformationen zum Zeitpunkt der Verbindungserstellung überprüft – und nicht zum Zeitpunkt der Abfrageausführung.
- Abfragen über den JDBC-Treiber waren fehlerhaft, sobald zusammen mit der URL ein Schema angegeben wurde.

24. März 2017

Veröffentlicht am 24.03.2017

Hinzugefügt AWS CloudTrail SerDe, verbesserte Leistung, behobene Partitionsprobleme.

Features

- Das wurde hinzugefügt AWS CloudTrail SerDe, das inzwischen durch das Hive JSON SerDe zum Lesen CloudTrail von Protokollen ersetzt wurde. Hinweise zum Abfragen von CloudTrail Protokollen finden Sie unter. Abfragen von AWS CloudTrail-Protokollen

Verbesserungen

- Verbesserte Leistung beim Scannen einer großen Partitionsanzahl.
- Verbesserte Leistung bei dem Vorgang `MSCK Repair Table`.
- Möglichkeit, nicht in der primären Region gespeicherte Amazon-S3-Daten abzufragen, wurde hinzugefügt. Neben den Athena-Standardgebühren fallen auch die Amazon S3-Standardgebühren für regionenübergreifende Datenübertragung an.

Fehlerbehebungen

- Der Fehler, aufgrund dessen ein "table not found error" auftrat, falls keine Partition geladen war, wurde behoben.
- Der Fehler, aufgrund dessen bei Abfragen des Typs `ALTER TABLE ADD PARTITION IF NOT EXISTS` eine Ausnahme auftrat, wurde behoben.
- Ein Fehler in `DROP PARTITIONS` wurde behoben.

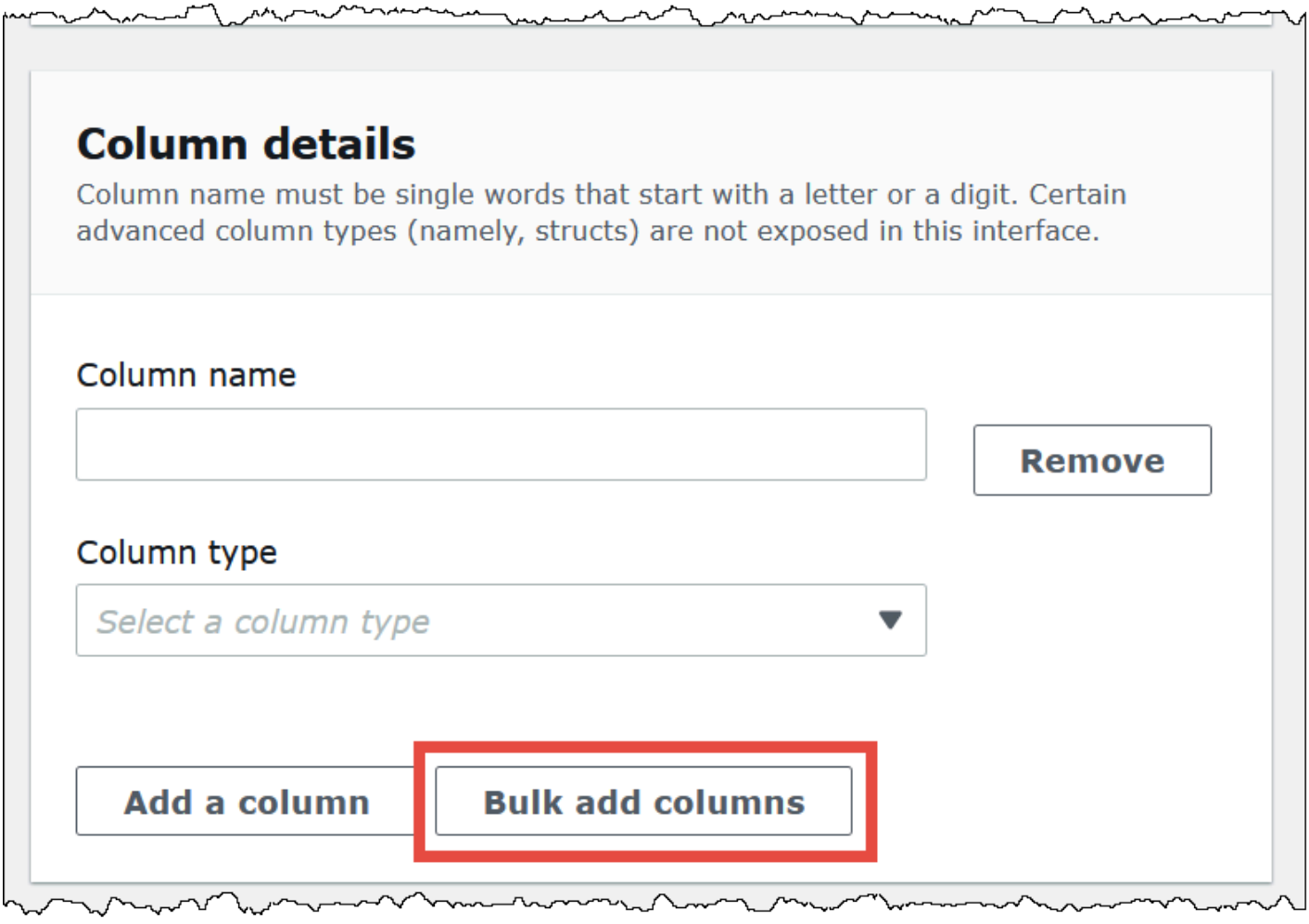
20. Februar 2017

Veröffentlicht am 20.02.2017

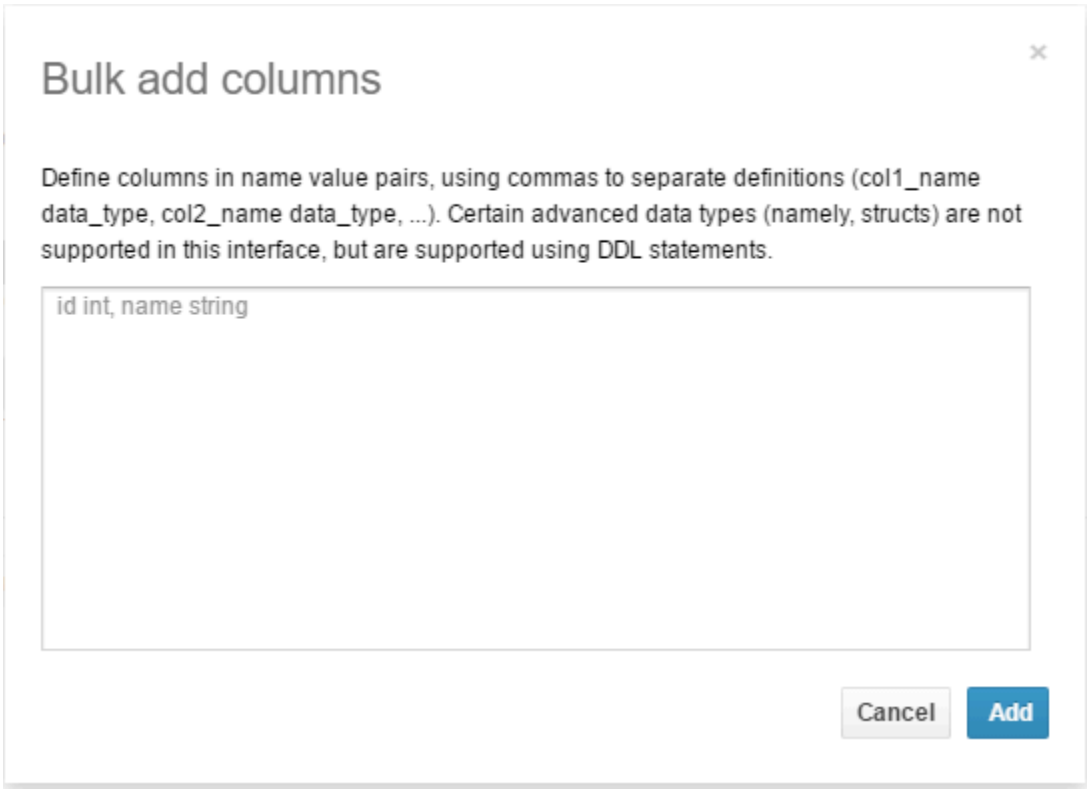
Unterstützung für AvroSerDe OpenCSVSerDe, Region USA Ost (Ohio) und Massenbearbeitung von Spalten im Konsolenassistenten hinzugefügt. Die Leistung bei großen Parquet-Tabellen wurde verbessert.

Features

- Unterstützung für neue Funktionen eingeführt: SerDes
  - Avro SerDe
  - OpenCSVSerDe für CSV-Verarbeitung
- Markteinführung der Region USA Ost (Ohio) (us-east-2). Sie können nun Abfragen in dieser Region ausführen.
- Sie können jetzt das Formular Tabelle aus S3-Bucket-Daten erstellen verwenden, um Tabellenschemata in großen Mengen zu definieren. Wählen Sie im Abfrage-Editor Create (Erstellen), S3 bucket data (S3-Bucket-Daten) aus und dann Bulk add columns (Spalten in großen Mengen hinzufügen) im Abschnitt Column details (Details der Spalte).



Geben Sie im Textfeld Name-Wert-Paare ein und wählen Sie Add (Hinzufügen).



Verbesserungen

- Die Leistung bei großen Parquet-Tabellen wurde verbessert.

Dokumentverlauf

Letzte Aktualisierung der Dokumentation: 27. März 2024.

Wir aktualisieren die Dokumentation regelmäßig, um Ihr Feedback zu berücksichtigen. In der folgenden Tabelle sind wichtige Ergänzungen zur Amazon-Athena-Dokumentation enthalten. Nicht alle Aktualisierungen werden dargestellt.

Änderung	Beschreibung	Datum der Veröffentlichung
Von `AmazonAthenaFullAccess` verwaltete Richtlinie aktualisiert.	Die `datazone:ListAccountEnvironments` Berechtigungen `datazone:ListDomains` `datazone:ListProjects` , und wurden der AmazonAthenaFullAccessverwalteten Richtlinie hinzugefügt. Die hinzugefügten Aktionen ermöglichen es Athena-Benutzern, mit DataZone Amazon-Domains, -Projekten und -Umgebungen zu arbeiten. Weitere Informationen finden Sie unter Verwenden von Amazon DataZone in Athena.	3. Januar 2024
Von `AmazonAthenaFullAccess` verwaltete Richtlinie aktualisiert.	Der AmazonAthenaFullAccessverwalteten Richtlinie wurden `glue:GetColumnStatisticsTaskRuns` Berechtigungen, und hinzugefügt`glue:StartColumnStatisticsTaskRun` . `glue:GetColumnStatisticsTaskRun` Die hinzugefügten Aktionen ermöglichen es Athena, aufzurufen AWS Glue , um Statistiken für die kostenbasierte Optimierungsfunktion abzurufen. Weitere Informationen finden Sie unter Verwenden des kostenbasierten Optimierers.	3. Januar 2024
Dokumentation für IAM Identity Center aktivierte Athena-Arbeitsgruppen hinzugefügt.	Sie können Athena-SQL-Arbeitsgruppen erstellen, die den Authentifizierungsmodus IAM Identity Center verwenden. Diese Arbeitsgruppen unterstützen die Verwendung derselben Identität für AWS Dienste wie Amazon Athena und Amazon EMR Studio. Weitere	05. Dezember 2023

Änderung	Beschreibung	Datum der Veröffentlichung
	Informationen finden Sie unter Verwendung von für IAM Identity Center aktivierten Athena-Arbeitsgruppen.	
Dokumentation zum Abfragen von Daten der S3 Express One Zone hinzugefügt	Sie können Athena zum Abfragen von Daten in der Speicherklasse Amazon S3 Express One Zone verwenden. Weitere Informationen finden Sie unter Abfragen von Daten der S3 Express One Zone.	28. November 2023
Dokumentation für Glue-Data-Catalog-Ansichten hinzugefügt.	Sie können Glue-Data-Catalog-Ansichten verwenden, um eine einzige gemeinsame Ansicht für AWS -Service wie Amazon Athena und Amazon Redshift bereitzustellen. Weitere Informationen finden Sie unter AWS Glue Data Catalog Ansichten verwenden.	8. November 2023
Die Dokumentation für das Feature der kostenbasierten Optimierung wurde hinzugefügt.	Sie können Statistiken von verwenden AWS Glue , um Ihre Abfragen in Athena SQL zu optimieren. Weitere Informationen finden Sie unter Verwenden des kostenbasierten Optimierers.	17. November 2023
Dokumentation für den Athena-JDBC-3.x-Treiber hinzugefügt	Sie können den Athena-JDBC-3.x-Treiber verwenden, um Abfrageergebnisse direkt aus Amazon S3 zu lesen. Der JDBC-3.x-Treiber unterstützt fast alle Authentifizierungsmethoden, die der JDBC-2.x-Treiber unterstützt. Weitere Informationen finden Sie unter Athena-JDBC-3.x-Treiber.	16. November 2023
Dokumentation für die Verwendung DataZone in Athena hinzugefügt.	Sie können DataZone es verwenden, um Ihre Erfahrung mit AWS Analysediensten wie Athena und Lake Formation zu vereinfachen. AWS Glue Weitere Informationen finden Sie unter Verwenden von Amazon DataZone in Athena.	04. Oktober 2023

Änderung	Beschreibung	Datum der Veröffentlichung
Dokumentation für Kapazitätsreservierungen hinzugefügt.	Sie können jetzt Kapazitätsreservierungen auf Amazon Athena verwenden, um SQL-Abfragen auf vollständig verwalteter Rechenkapazität auszuführen. Weitere Informationen finden Sie unter Kapazität zur Abfrageverarbeitung verwalten.	28. April 2023
Dokumentation für die Abfrage von Verbundansichten hinzugefügt.	Sie können jetzt Ansichten auf Verbunddatenquellen in Athena erstellen und abfragen. Weitere Informationen finden Sie unter Verbundansichten abfragen.	4. April 2023
Dokumentation zur Verhinderung von Drosselung in Amazon S3 hinzugefügt.	Weitere Informationen finden Sie unter Drosselung durch Amazon S3 verhindern.	24. März 2023
Von `AmazonAthenaFullAccess` verwaltete Richtlinie aktualisiert.	`pricing:GetProducts` Zur AmazonAthenaFullAccessverwalteten Richtlinie hinzugefügt. Die hinzugefügte Aktion bietet Zugriff auf AWS Billing and Cost Management. Weitere Informationen finden Sie GetProductsin der AWS Billing and Cost Management API-Referenz.	25. Januar 2023
Erweiterte Dokumentation für die Unterstützung der Athena-Komprimierung.	Einzelne Themen für Komprimierung von Hive-Tabellen, Komprimierung von Iceberg-Tabellen und ZSTD-Komprimierungsstufen hinzugefügt. Weitere Informationen finden Sie unter Athena-Komprimierungs-Support.	20. Januar 2023

Änderung	Beschreibung	Datum der Veröffentlichung
Dokumentation für Amazon Athena für Apache Spark wurde hinzugefügt.	Sie können jetzt interaktiv Apache-Spark-Anwendungen und Jupyter-kompatible Notebooks auf Amazon Athena erstellen und ausführen. Weitere Informationen finden Sie unter Verwenden von Apache Spark in Amazon Athena.	30. November 2022
Dokumentation für den Athena-IBM-Db2-Konnektor hinzugefügt.	Sie können den Amazon-Athena-Konnektor für IBM Db2 verwenden, um Db2 von Athena abzufragen. Weitere Informationen finden Sie unter Amazon-Athena-IBM-Db2-Konnektor.	18. November 2022
Dokumentation zur Wiederverwendung von Abfrageergebnissen hinzugefügt.	Wenn Sie eine Abfrage in Athena erneut ausführen, können Sie jetzt optional das zuletzt gespeicherte Abfrageergebnis wiederverwenden. Dies kann die Leistung erhöhen und die Kosten in Bezug auf die Anzahl der gescannten Bytes reduzieren. Weitere Informationen finden Sie unter Wiederverwenden von Abfrageergebnissen.	08. November 2022
Die Dokumentation für CloudTrail Protokolle wurde aktualisiert.	Die `CREATE TABLE` DDL für die Abfrage von CloudTrail Protokollen wurde aktualisiert und verwendet nun JSON SerDe anstelle von. CloudTrail SerDe Weitere Informationen finden Sie unter Abfragen von AWS CloudTrail-Protokollen.	03. November 2022
Dokumentation für die Athena-Engine-Version 3 hinzugefügt.	Weitere Informationen zur Athena-Engine-Version 3 finden Sie unter Athena-Engine-Version 3.	13. Oktober 2022

Änderung	Beschreibung	Datum der Veröffentlichung
Praktische Anleitung zur Konfiguration von SSO für ODBC mit dem Okta-Plug-In hinzugefügt.	Konfigurieren Sie den Amazon-Athena-ODBC-Treiber und das Okta-Plug-In, um mithilfe des Okta-Identitätsanbieters Funktionen für Single-Sign-On (SSO) hinzuzufügen. Weitere Informationen finden Sie unter Konfigurieren von SSO für ODBC mit dem Okta-Plug-In und einem Okta-Identitätsanbieter.	23. August 2022
Dokumentation für die Anzeige von Abfrageplänen und Statistiken in der Athena-Konsole hinzugefügt.	Sie können den Athena-Abfrage-Editor verwenden, um grafisch dargestellt zu bekommen, wie Ihre Abfragen ausgeführt werden, sowie Grafiken, Details und Statistiken darüber zu erhalten, wie abgeschlossene Abfragen ausgeführt wurden. Weitere Informationen finden Sie unter Anzeigen von Ausführungsplänen für SQL-Abfragen und Anzeigen von Statistiken und Ausführungsdetails für abgeschlossene Abfragen.	21. Juli 2022
Es wurde eine Dokumentation zum Abfragen von Apache-Hive-Ansichten in externen Hive-Metastores hinzugefügt.	Sie können Athena verwenden, um Apache-Ansichten abzufragen, die in externen Hive-Metastores erstellt wurden. Einige Hive-Funktionen werden nicht unterstützt oder erfordern eine spezielle Handhabung. Weitere Informationen finden Sie unter Arbeiten mit Hive-Ansichten.	22. April 2022
Es wurde Dokumentation für gespeicherte Abfragen hinzugefügt.	Sie können das Feature für gespeicherte Abfragen in Athena verwenden, um Ihre Abfragen zu speichern, abzurufen, zu bearbeiten und umzubenennen. Weitere Informationen finden Sie Verwenden von gespeicherten Abfragen in diesem Handbuch und UpdateNamedQueryin der Amazon Athena API-Referenz.	28. Februar 2022

Änderung	Beschreibung	Datum der Veröffentlichung
Vorschau-Dokumentation für Apache-Iceberg-Unterstützung hinzugefügt.	Athena unterstützt Lese-, Zeitreise- und Schreibabfragen für Apache Iceberg-Tabellen, die das Apache Parquet-Format für Daten und den AWS Glue Katalog für ihren Metastore verwenden. Weitere Informationen finden Sie unter Apache-Iceberg-Tabellen verwenden.	26. November 2021
Dokumentation für kontoübergreifende Verbundabfragen wurde hinzugefügt.	Sie können das kontoübergreifende Verbundabfragefeature verwenden, um Datenquellen in einem anderen Konto abzufragen. Weitere Informationen zum Einrichten von Berechtigungen zum Aktivieren dieses Features finden Sie unter Aktivieren von kontoübergreifenden Verbundabfragen.	12. November 2021
Dokumentation für die Athena-`UNLOAD`-Anweisung hinzugefügt.	Verwenden Sie die `UNLOAD`-Anweisung, um die Abfrageergebnisse einer `SELECT`-Anweisung in die Formate Apache Parquet, ORC, Apache Avro und JSON zu schreiben. Weitere Informationen finden Sie unter UNLOAD.	5. August 2021
Dokumentation für das Athena-`EXPLAIN`-Anweisungsfeature hinzugefügt.	Weitere Informationen finden Sie unter Verwenden von EXPLAIN und EXPLAIN ANALYZE in Athena und Die Ergebnisse der Athena-EXPLAIN-Anweisung verstehen.	05. April 2021
Seiten über Fehlerbehebung und Leistungsoptimierung in Athena hinzugefügt.	Weitere Informationen finden Sie unter Athena-Fehlerbehebung und Leistungsoptimierung in Athena.	30. Dezember 2020

Änderung	Beschreibung	Datum der Veröffentlichung
Dokumentation für die Athena-Engine-Versionierung und Athena-Engine-Version 2 hinzugefügt.	Weitere Informationen finden Sie unter Athena-Engine-Versionierung.	11. November 2020
Aktualisierte Dokumentation für die Verbundabfrage für allgemeine Verfügbarkeit.	Weitere Informationen finden Sie unter Nutzung von Amazon-Athena-Verbundabfrage und Athena mit CalledVia-Kontextschlüsseln verwenden.	11. November 2020
Dokumentation für die Verwendung des JDBC-Treibers mit Lake Formation für den Verbundzugriff auf Athena hinzugefügt.	Weitere Informationen finden Sie unter Verwenden von Lake Formation und den Athena-JDBC- und ODBC-Treibern für den Verbundzugriff auf Athena und Tutorial: Konfigurieren des Verbundzugriffs für Okta-Benutzer auf Athena mithilfe von Lake Formation und JDBC.	25. September 2020
Dokumentation für den Amazon Athena OpenSearch Athena-Datenkonnektor hinzugefügt.	Weitere Informationen finden Sie unter Amazon Athena OpenSearch Konnektor.	21. Juli 2020
Dokumentation zum Abfragen von Hudi-Datensätzen hinzugefügt.	Weitere Informationen finden Sie unter Verwenden von Athena zum Abfragen von Apache-Hudi-Datensätzen.	9. Juli 2020

Änderung	Beschreibung	Datum der Veröffentlichung
Dokumentation zum Abfragen von Apache-Webserver-Protokollen und IIS-Webserverprotokollen, die in Amazon S3 gespeichert sind, wurde hinzugefügt.	Weitere Informationen finden Sie unter Abfragen von Apache-Protokollen in Amazon S3 abfragen und Abfragen von Internetinformationsserver-Protokollen (IIS), die in Amazon S3 gespeichert sind.	8. Juli 2020
Dokumentation für die allgemeine Version des Athena Data Konnektor für den externen Hive Metastore wurde hinzugefügt.	Weitere Informationen finden Sie unter Verwenden von Athena-Daten-Connector für externen Hive-Metastore.	1. Juni 2020
Dokumentation zum Tagging von Datenkatalogressourcen hinzugefügt.	Weitere Informationen finden Sie unter Markieren von Athena-Ressourcen.	1. Juni 2020
Dokumentation zur Partitionsprojektion hinzugefügt.	Weitere Informationen finden Sie unter Partitionsprojektion mit Amazon Athena.	21. Mai 2020
Die Java-Codebeispiele für Athena wurden aktualisiert.	Weitere Informationen finden Sie unter Codebeispiele.	11. Mai 2020

Änderung	Beschreibung	Datum der Veröffentlichung
Es wurde ein Thema zur Abfrage von GuardDuty Amazon-Ergebnissen hinzugefügt.	Weitere Informationen finden Sie unter Abfragen von Amazon-GuardDuty-Ergebnissen.	19. März 2020
Es wurde ein Thema zur Verwendung von CloudWatch Ereignissen zur Überwachung von Statusübergängen bei Athena-Abfragen hinzugefügt.	Weitere Informationen finden Sie unter Überwachung von Athena-Abfragen mit Amazon EventBridge-Ereignissen.	11. März 2020
Es wurde ein Thema zur Abfrage von AWS Global Accelerator Flow-Logs mit Athena hinzugefügt.	Weitere Informationen finden Sie unter Abfragen von AWS Global Accelerator-Flow-Protokollen.	6. Februar 2020

Änderung	Beschreibung	Datum der Veröffentlichung
• Dokumentation zur Verwendung von CTAS mit INSERT INTO zum Hinzufügen von Daten aus einer nicht partitionierten Quelle zu einem partitionierten Ziel hinzugefügt. • Download-Links für die 1.1.0-Vorversion des ODBC-Treibers für Athena hinzugefügt. • Beschreibung für `SHOW DATABASES LIKE` Regex korrigiert. • Die `partitioned_by` -Syntax im CTA-Thema wurde korrigiert. • Weitere kleinere Korrekturen.	Zu den Aktualisierungen der Dokumentation gehören unter anderem die folgenden Themen: • Verwenden von CTAS und INSERT INTO für ETL und Datenanalyse • Herstellen einer Verbindung mit Amazon Athena mit ODBC (Die 1.1.0-Vorschaufeatures sind jetzt im 1.1.2-ODBC-Treiber enthalten.) • SHOW DATABASES • CREATE TABLE AS	4. Februar 2020

Änderung	Beschreibung	Datum der Veröffentlichung
Dokumentation zur Verwendung von CTAS mit INSERT INTO zum Hinzufügen von Daten aus einer partitionierten Quelle zu einem partitionierten Ziel hinzugefügt.	Weitere Informationen finden Sie unter Verwenden von CTAS und INSERT INTO zum Umgehen des Limits von 100 Partitionen.	22. Januar 2020
Angaben zum Speicherort der Abfrageergebnisse aktualisiert.	Athena erstellt keinen Standardspeicherort für Abfrageergebnisse mehr. Weitere Informationen finden Sie unter Angeben eines Speicherorts des Abfrageergebnisses.	20. Januar 2020
Thema zum Abfragen von AWS Glue Data Catalog hinzugefügt. Aktualisierte Informationen zu Service Quotas (ehemals „Service-Limits“) in Athena.	Weitere Informationen finden Sie unter den folgenden Themen: • Abfragen von AWS Glue Data Catalog • Service Quotas	17. Januar 2020

Änderung	Beschreibung	Datum der Veröffentlichung
Das Thema auf OpenCSV wurde SerDe dahingehend korrigiert, dass der `TIMESTAMP` Typ im numerischen UNIX-Format angegeben werden sollte.	Weitere Informationen finden Sie unter OpenCSVSerDe für CSV-Verarbeitung.	15. Januar 2020
Sicherheitsthema zur Verschlüsselung aktualisiert, um zu vermerken, dass Athena keine asymmetrischen Schlüssel unterstützt.	Athena unterstützt nur symmetrische Schlüssel zum Lesen und Schreiben von Daten. Weitere Informationen finden Sie unter Unterstützte Verschlüsselungsoptionen der Amazon S3.	8. Januar 2020
Es wurden Informationen zum kontoüber greifenden Zugriff auf Amazon S3 S3-Buckets hinzugefügt, die mit einem benutzerdefinierten AWS KMS Schlüssel verschlüsselt sind.	Weitere Informationen finden Sie unter Kontoüber greifender Zugriff auf einen Bucket, der mit einem benutzerdefinierten AWS KMS-Schlüssel verschlüsselt ist.	13. Dezember 2019

Änderung	Beschreibung	Datum der Veröffentlichung
Dokumentation für Verbundabfragen, externe Hive-Metastores, Machine Learning und benutzerdefinierte Funktionen hinzugefügt. Neue CloudWatch-Metriken hinzugefügt.	Weitere Informationen finden Sie unter den folgenden Themen: • Nutzung von Amazon-Athena-Verbundabfrage • Verfügbare Datenquellenkonnektoren • Verwenden von Athena-Daten-Connector für externen Hive-Metastore • Verwendung von Machine Learning (ML) mit Amazon Athena • Abfragen mit benutzerdefinierten Funktionen (User Defined Functions, UDFs) • Liste der CloudWatch Metriken und Dimensionen für Athena	26. November 2019
Abschnitt für den neuen INSERT INTO-Befehl und aktualisierte Standortinformationen für Abfrageergebnisse zur Unterstützung von Daten-Manifest-Dateien hinzugefügt.	Weitere Informationen finden Sie unter INSERT INTO und Arbeiten mit Abfrageergebnissen, letzten Abfragen und Ausgabedateien.	18. September 2019

Änderung	Beschreibung	Datum der Veröffentlichung
Abschnitt für die Unterstützung von Schnittstellen-VPC-Endpunkten (PrivateLink) hinzugefügt. Aktualisierte JDBC-Treiber. Aktualisierte Informationen zu erweiterten VPC-Flow-Protokollen.	Weitere Informationen finden Sie unter Herstellen einer Verbindung mit Amazon Athena über einen Schnittstellen-VPC-Endpunkt, Abfragen von Amazon-VPC-Flow-Protokollen und Herstellen einer Verbindung zu Amazon Athena mit JDBC.	11. September 2019
Abschnitt zur Integration mit AWS Lake Formation hinzugefügt.	Weitere Informationen finden Sie unter Verwenden von Athena zum Abfragen von Daten, die in AWS Lake Formationregistriert sind.	26. Juni 2019
Der Abschnitt "Sicherheit" wurde aus Konsistenzgründen mit anderen AWS-Services aktualisiert.	Weitere Informationen finden Sie unter Sicherheit von Amazon Athena.	26. Juni 2019
Abschnitt zum Abfragen von AWS WAF Logs hinzugefügt.	Weitere Informationen finden Sie unter Abfragen von AWS WAF Protokollen.	31. Mai 2019

Änderung	Beschreibung	Datum der Veröffentlichung
Veröffentlichung der neuen Version des ODBC-Treibers mit Unterstützung für Athena-Arbeitsgruppen.	Informationen zum Herunterladen der ODBC-Treiberversion 1.0.5 und der zugehörigen Dokumentation finden Sie unter Herstellen einer Verbindung mit Amazon Athena mit ODBC. Es werden keine Änderungen an der Verbindungszeichenfolge des ODBC-Treibers vorgenommen, wenn Sie Tags in Arbeitsgruppen verwenden. Um Tags zu verwenden, aktualisieren Sie auf die neueste Version des ODBC-Treibers (die aktuelle Version). Diese Treiberversion ermöglicht Ihnen die Verwendung von Athena-API-Arbeitsgruppen-Aktionen zum Erstellen und Verwalten von Arbeitsgruppen und Athena-API-Tag-Aktionen zum Hinzufügen, Auflisten oder Entfernen von Tags auf Arbeitsgruppen. Stellen vor Beginn sicher, dass Sie über Berechtigungen auf Ressourcenebene in IAM für Aktionen auf Arbeitsgruppen und Tags verfügen.	5. März 2019
Hinzufügung der Tag-Unterstützung für Arbeitsgruppen in Amazon Athena.	Ein Tag besteht aus einem Schlüssel und einem Wert, die Sie beide selbst definieren können. Wenn Sie eine Arbeitsgruppe markieren, weisen Sie ihr benutzerdefinierte Metadaten zu. Erstellen Sie beispielsweise eine Arbeitsgruppe für jede Kostenstelle. Wenn Sie diesen Arbeitsgruppen dann Tags hinzufügen, können Sie Ihre Athena-Ausgaben für jede Kostenstelle nachverfolgen. Weitere Informationen finden Sie unter Verwendung von Tags für die Fakturierung im AWS Billing and Cost Management -Benutzerhandbuch.	22. Februar 2019

Änderung	Beschreibung	Datum der Veröffentlichung
Das in Athena SerDe verwendete JSON OpenX wurde verbessert.	Die Verbesserungen beinhalten u. a. Folgendes: • Unterstützung für die Eigenschaft `ConvertDotsInJsonKeysToUnderscores`. Wenn diese Option auf gesetzt ist`TRUE`, können SerDe die Punkte in Schlüsselnamen durch Unterstriche ersetzt werden. Wenn der JSON-Datensatz beispielsweise einen Schlüssel mit dem Namen `"a.b"` enthält, können Sie diese Eigenschaft verwenden, um den Spaltennamen als `"a_b"` in Athena zu definieren. Der Standardwert ist `FALSE`. Standardmäßig erlaubt Athena keine Punkte in Spaltennamen. • Unterstützung für die Eigenschaft `case.insensitive`. Standardmäßig erfordert Athena, dass alle Schlüssel in Ihrem JSON-Datensatz Kleinbuchstaben verwenden. Mit `WITH SERDE PROPERTIES ("case.insensitive"= FALSE;)` können Sie Schlüsselnamen verwenden, bei denen Groß-/Kleinschreibung bedeutsam ist. Der Standardwert ist `TRUE`. Bei Einstellung auf werden `TRUE` alle Großbuchstabenspalten in Kleinbuchstaben SerDe umgewandelt. Weitere Informationen finden Sie unter OpenX JSON SerDe.	18. Februar 2019

Änderung	Beschreibung	Datum der Veröffentlichung
Hinzufügung der Unterstützung für Arbeitsgruppen.	Verwenden Sie Arbeitsgruppen zum Trennen von Benutzern, Teams, Anwendungen oder Workloads sowie zur Einrichtung von Grenzwerten für die Datenmenge, die jede Abfrage oder die gesamte Arbeitsgruppe verarbeiten kann. Da Arbeitsgruppen als IAM-Ressourcen fungieren, können Sie Berechtigungen auf Ressourcenebene verwenden, um den Zugriff auf eine bestimmte Arbeitsgruppe zu steuern. Sie können auch abfragebezogene Metriken in Amazon anzeigen CloudWatch, die Abfragekosten kontrollieren, indem Sie Grenzwerte für die Menge der gescannten Daten konfigurieren, Schwellenwerte erstellen und Aktionen wie Amazon SNS SNS-Alarme auslösen, wenn diese Schwellenwerte überschritten werden. Weitere Informationen finden Sie unter Verwenden von Arbeitsgruppen zum Ausführen von Abfragen und Steuern von Kosten und Überwachen von Abfragen mit CloudWatch Metriken und Ereignissen.	18. Februar 2019
Es wurde Support für das Analysieren von Protokollen aus dem Network Load Balancer hinzugefügt.	Es wurde Beispiel Athena-Abfragen für das Analysieren von Protokollen aus dem Network Load Balancer hinzugefügt. Diese Protokolle erhalten detaillierte Informationen über die Transport Layer Security(TLS)-Anfragen, die an Network Load Balancer gesendet werden. Sie können diese Zugriffsprotokolle für die Analyse von Datenverkehrsmustern und zur Problembehebung verwenden. Weitere Informationen finden Sie unter the section called “Network Load Balancer”.	24. Januar 2019

Änderung	Beschreibung	Datum der Veröffentlichung
Die neuen Versionen der JDBC- und ODBC-Treiber mit Unterstützung für Verbundzugriff auf die Athena-API mit AD FS und SAML 2.0 (Security Assertion Markup Language 2.0) wurden veröffentlicht.	Ab dieser Treiberversion wird der Verbundzugriff auf Athena für Active Directory Federation Service (AD FS 3.0) unterstützt. Der Zugriff wird durch die Versionen von JDBC- bzw. ODBC-Treibern hergestellt, die SAML 2.0 unterstützen. Informationen zur Konfiguration des Verbundzugriffs auf die Athena-API finden Sie unter the section called “Aktivieren des föderierten Zugriffs auf die Athena-API”.	10. November 2018
Zusätzliche Unterstützung für detaillierte Zugriffskontrolle auf Datenbanken und Tabellen in Athena. Darüber hinaus wurden Richtlinien zu Athena hinzugefügt, mit denen Sie Datenbank- und Tabellenmetadaten im Datenkatalog verschlüsseln können.	Es wurde Unterstützung für die Erstellung identitätsbasierter (IAM) -Richtlinien hinzugefügt, die eine differenzierte Zugriffskontrolle auf Ressourcen in der ermöglichen AWS Glue Data Catalog, z. B. auf Datenbanken und Tabellen, die in Athena verwendet werden. Außerdem können Sie Datenbank- und Tabellenmetadaten im Datenkatalog verschlüsseln, indem Sie spezifische Richtlinien zu Athena hinzufügen. Details hierzu finden Sie unter Differenzierter Zugriff auf Datenbanken und Tabellen in AWS Glue Data Catalog.	15. Oktober 2018

Änderung	Beschreibung	Datum der Veröffentlichung
Unterstützung für `CREATE TABLE AS SELECT`-Anweisungen hinzugefügt. Weitere Verbesserungen an der Dokumentation vorgenommen.	Unterstützung für `CREATE TABLE AS SELECT`-Anweisungen hinzugefügt. Informationen finden Sie unter Erstellen einer Tabelle aus Abfrageergebnissen (CTAS), Überlegungen und Einschränkungen für CTAS-Abfragen und Beispiele für CTAS-Abfragen.	10. Oktober 2018
ODBC-Treiber Version 1.0.3 freigegeben, mit Support für Streaming von Ergebnissen statt deren Abruf auf Seiten. Weitere Verbesserungen an der Dokumentation vorgenommen.	Der ODBC-Treiber Version 1.0.3 unterstützt Streaming-Ergebnisse und umfasst auch Verbesserungen, Fehlerbehebungen und eine aktualisierte Dokumentation für "Verwendung von SSL mit einem Proxy-Server". Informationen zum Herunterladen der ODBC-Treiberversion 1.0.3 und der zugehörigen Dokumentation finden Sie unter Herstellen einer Verbindung mit Amazon Athena mit ODBC.	6. September 2018

Änderung	Beschreibung	Datum der Veröffentlichung
JDBC-Treiber Version 2.0.5 freigegeben, mit Standard-Support für Streaming von Ergebnissen statt deren Abruf auf Seiten. Weitere Verbesserungen an der Dokumentation vorgenommen.	JDBC-Treiber 2.0.5 freigegeben, mit Standard-Support für Streaming von Ergebnissen statt deren Abruf auf Seiten. Weitere Informationen finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC.	16. August 2018
Dokumentation zum Abfragen von Amazon-Virtual-Private-Cloud-Flow-Protokollen, die direkt in Amazon S3 in einem GZIP-Format gespeichert werden können, aktualisiert. Beispiele für die Abfrage von ALB-Protokollen aktualisiert.	Dokumentation zum Abfragen von Amazon-Virtual-Private-Cloud-Flow-Protokollen, die direkt in Amazon S3 in einem GZIP-Format gespeichert werden können, aktualisiert. Weitere Informationen finden Sie unter Abfragen von Amazon-VPC-Flow-Protokollen. Beispiele für die Abfrage von ALB-Protokollen aktualisiert. Weitere Informationen finden Sie unter Abfragen von Application-Load-Balancer-Protokollen.	7. August 2018

Änderung	Beschreibung	Datum der Veröffentlichung
Unterstützung für Ansichten hinzugefügt. Zusätzliche Richtlinien für Schema-Manipulationen für verschiedene Datenspeicherformate.	Unterstützung für Ansichten hinzugefügt. Weitere Informationen finden Sie unter Arbeiten mit Ansichten. In diesem Abschnitt wurden Anleitungen zum Umgang mit Schema-Updates für verschiedene Datenspeicherformate ergänzt. Weitere Informationen finden Sie unter Verarbeiten von Schema-Updates.	5. Juni 2018
Die Standard-Obergrenze für gleichzeitige Abfragen wurde von fünf auf zwanzig angehoben.	Sie können bis zu 20 DDL-Abfragen und 20 SELECT-Abfragen gleichzeitig absenden und ausführen. Weitere Informationen finden Sie unter Service Quotas.	17. Mai 2018
Es wurden neue Registerkarten für Abfragen und eine Möglichkeit zur Konfiguration der automatischen Vervollständigung im Query Editor (Abfrage-Editor) hinzugefügt.	Es wurden neue Registerkarten für Abfragen und eine Möglichkeit zur Konfiguration der automatischen Vervollständigung im Query Editor (Abfrage-Editor) hinzugefügt. Weitere Informationen finden Sie unter Erste Schritte.	8. Mai 2018

Änderung	Beschreibung	Datum der Veröffentlichung
JDBC-Treiberversion 2.0.2 wurde freigegeben.	Neue Version des JDBC-Treibers (Version 2.0.2) wurde freigegeben. Weitere Informationen finden Sie unter Herstellen einer Verbindung zu Amazon Athena mit JDBC.	19. April 2018
Automatische Vervollständigung für die Eingabe von Abfragen in der Athena-Konsole hinzugefügt.	Automatische Vervollständigung für die Eingabe von Abfragen in der Athena-Konsole hinzugefügt.	6. April 2018
Es wurde die Möglichkeit hinzugefügt, Athena-Tabellen für CloudTrail Protokolldateien direkt von der CloudTrail Konsole aus zu erstellen.	Es wurde die Möglichkeit hinzugefügt, automatisch Athena-Tabellen für CloudTrail Protokolldateien direkt von der CloudTrail Konsole aus zu erstellen. Weitere Informationen finden Sie unter Verwenden der CloudTrail Konsole zum Erstellen einer Athena-Tabelle für CloudTrail Protokolle .	15. März 2018
Unterstützung für das sichere Auslagern von Zwischendaten auf Datenträger, um Abfragen mit `GROUP BY` auszuführen, hinzugefügt.	Es wurde eine Möglichkeit für das Auslagern von Zwischendaten auf Festplatte bei arbeitsspeicherintensiven Abfragen, die die `GROUP BY`-Klausel verwenden, hinzugefügt. Dadurch wird die Zuverlässigkeit solcher Abfragen verbessert und die Fehlermeldung "Query resource exhausted" (Abfrage-Ressource aufgebraucht) vermieden. Weitere Informationen finden Sie im Versionshinweis für 2. Februar 2018.	2. Februar 2018

Änderung	Beschreibung	Datum der Veröffentlichung
Unterstützung für Presto-Version 0.172 hinzugefügt.	Upgrade der zugrunde liegenden Engine in Amazon Athena auf eine Version, die auf Presto-Version 0.172 basiert. Weitere Informationen finden Sie im Versionshinweis für 19. Januar 2018.	19. Januar 2018
Unterstützung für den ODBC-Treiber hinzugefügt.	Unterstützung für die Verbindungserstellung von Athena zum ODBC-Treiber hinzugefügt. Weitere Informationen finden Sie unter Verbindung zu Amazon Athena mit ODBC.	13. November 2017
Unterstützung für die Regionen Asien-Pazifik (Seoul), Asien-Pazifik (Mumbai) und Europa (London) hinzugefügt. Unterstützung für die Abfrage von koordinatenbasierten Daten hinzugefügt.	Unterstützung für die Abfrage von koordinatenbasierten Daten und für die Regionen Asien-Pazifik (Seoul), Asien-Pazifik (Mumbai) und Europa (London) hinzugefügt. Weitere Informationen finden Sie unter Abfragen von koordinatenbasierten Daten und AWS-Regionen und Endpunkte.	1. November 2017
Unterstützung für Europa (Frankfurt) hinzugefügt.	Unterstützung für Europa (Frankfurt) hinzugefügt. Eine Liste der unterstützten Regionen finden Sie unter AWS-Regionen und Endpunkte.	19. Oktober 2017
Unterstützung für benannte Athena-Abfragen mit AWS CloudFormation hinzugefügt.	Unterstützung für die Erstellung benannter Athena-Abfragen mit AWS CloudFormation hinzugefügt. Weitere Informationen finden Sie AWS::Athena::NamedQueryim AWS CloudFormation Benutzerhandbuch.	3. Oktober 2017

Änderung	Beschreibung	Datum der Veröffentlichung
Unterstützung für die Region Asien-Pazifik (Sydney) hinzugefügt	Unterstützung für die Region Asien-Pazifik (Sydney) hinzugefügt Eine Liste der unterstützten Regionen finden Sie unter AWS-Regionen und Endpunkte.	25. September 2017
Diesem Handbuch wurde ein Abschnitt zum Abfragen von AWS-Service Protokollen und verschiedenen Datentypen hinzugefügt, darunter Maps, Arrays, verschachtelte Daten und Daten, die JSON enthalten.	Beispiele für Abfragen von AWS-Service-Protokollen und für Abfragen verschiedener Datentypen in Athena hinzugefügt. Weitere Informationen finden Sie unter Ausführen von SQL-Abfragen mit Amazon Athena.	5. September 2017
Unterstützung für hinzugefügt. AWS Glue Data Catalog	Integration mit dem AWS Glue Data Catalog und ein Migrationsassistent für die Aktualisierung vom verwalteten Athena-Datenkatalog auf den AWS Glue Data Catalog hinzugefügt. Weitere Informationen über die Integration mit AWS Glue finden Sie unter AWS Glue.	14. August 2017
Unterstützung für SerDe Grok hinzugefügt.	Unterstützung für Grok hinzugefügt SerDe, die einen einfacheren Musterabgleich für Datensätze in unstrukturierten Textdateien wie Protokollen ermöglicht. Weitere Informationen finden Sie unter Grok. SerDe Tastenkombinationen wurden hinzugefügt, um über die Konsole durch den Abfrageverlauf zu blättern.	4. August 2017

Änderung	Beschreibung	Datum der Veröffentlichung
Unterstützung für die Region Asien-Pazifik (Tokio) hinzugefügt	Unterstützung für die Regionen Asien-Pazifik (Tokio) und Asien-Pazifik (Singapur) hinzugefügt. Eine Liste der unterstützten Regionen finden Sie unter AWS-Regionen und Endpunkte.	22. Juni 2017
Es wurde Unterstützung für Europa (Irland) hinzugefügt.	Es wurde Unterstützung für Europa (Irland) hinzugefügt. Weitere Informationen finden Sie unter AWS-Regionen und Endpunkte.	8. Juni 2017
Amazon Athena-API und AWS CLI Support hinzugefügt.	Eine Amazon Athena Athena-API und AWS CLI Unterstützung für Athena wurden hinzugefügt. JDBC-Treiber auf Version 1.1.0 aktualisiert.	19. Mai 2017
Unterstützung für Amazon-S3-Datenverschlüsselung hinzugefügt.	Unterstützung für die Amazon-S3-Datenverschlüsselung und die veröffentlichte JDBC-Treiberaktualisierung (Version 1.0.1) mit Verschlüsselungsunterstützung sowie Verbesserungen und Fehlerbehebungen hinzugefügt. Weitere Informationen finden Sie unter Verschlüsselung im Ruhezustand.	4. April 2017

Änderung	Beschreibung	Datum der Veröffentlichung
Das wurde hinzugefügt. AWS CloudTrail SerDe	Hinzugefügt AWS CloudTrail SerDe, verbesserte Leistung, behobene Partitionsprobleme. • Das AWS CloudTrail SerDe wurde durch das Hive JSON SerDe zum Lesen CloudTrail von Protokollen ersetzt. Hinweise zum Abfragen von CloudTrail Protokollen finden Sie unter. Abfragen von AWS CloudTrail-Protokollen • Verbesserte Leistung beim Scannen einer großen Partitionsanzahl. • Verbesserte Leistung bei dem Vorgang `MSCK Repair Table`. • Möglichkeit, nicht in der primären Region gespeicherte Amazon S3-Daten abzufragen, wurde hinzugefügt. Neben den Athena-Standardgebühren fallen auch die Amazon S3-Standardgebühren für regionenübergreifende Datenübertragung an.	24. März 2017
Unterstützung für USA Ost (Ohio) wurde hinzugefügt.	Avro SerDe und OpenCSVSerDe für CSV-Verarbeitung, sowie die Massenbearbeitung von Spalten im Konsolenassistenten werden jetzt unterstützt. Die Leistung bei großen Parquet-Tabellen wurde verbessert.	20. Februar 2017
	Erstveröffentlichung: Amazon-Athena-Benutzerhandbuch.	November 2016

AWS-Glossar

Die neueste AWS-Terminologie finden Sie im AWS-Glossar in der AWS-Glossar-Referenz.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.