Panduan Pengguna

Amazon Athena

Amazon Athena: Panduan Pengguna

Table of Contents

Apa Itu Amazon Athena?

Amazon Athena adalah layanan kueri interaktif yang memudahkan untuk menganalisis data secara langsung di Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) menggunakan SQL standar. Dengan beberapa tindakan di dalamnya AWS Management Console, Anda dapat mengarahkan Athena ke data yang disimpan di Amazon S3 dan mulai menggunakan SQL standar untuk menjalankan kueri ad-hoc dan mendapatkan hasil dalam hitungan detik.

Untuk informasi selengkapnya, lihat Memulai.

Amazon Athena juga memudahkan untuk menjalankan analisis data secara interaktif menggunakan Apache Spark tanpa harus merencanakan, mengonfigurasi, atau mengelola sumber daya. Ketika Anda menjalankan aplikasi Apache Spark di Athena, Anda mengirimkan kode Spark untuk diproses dan menerima hasilnya secara langsung. Gunakan pengalaman notebook yang disederhanakan di konsol Amazon Athena untuk mengembangkan aplikasi Apache Spark menggunakan Python atau. API notebook Athena

Untuk informasi selengkapnya, lihat Memulai dengan Apache Spark di Amazon Athena.

Athena SQL dan Apache Spark di Amazon Athena tidak memiliki server, jadi tidak ada infrastruktur untuk mengatur atau mengelola, dan Anda hanya membayar untuk kueri yang Anda jalankan. Athena menskalakan secara otomatis—menjalankan kueri secara paralel—sehingga hasilnya cepat, bahkan dengan kumpulan data besar dan kueri yang kompleks.

Topik

Kapan saya harus menggunakan Athena?

Layanan kueri seperti Amazon Athena, gudang data seperti Amazon Redshift, dan kerangka kerja pemrosesan data canggih seperti Amazon EMR semuanya memenuhi kebutuhan dan kasus penggunaan yang berbeda. Panduan berikut dapat membantu Anda memilih satu atau lebih layanan berdasarkan kebutuhan Anda.

Amazon Athena

Athena membantu Anda menganalisis data tidak terstruktur, semi-terstruktur, dan terstruktur yang disimpan di Amazon S3. Contohnya termasuk format data CSV, JSON, atau kolumnar seperti Apache Parquet dan Apache ORC. Anda dapat menggunakan Athena untuk menjalankan kueri ad-hoc menggunakan ANSI SQL, tanpa perlu mengumpulkan atau memuat data ke Athena.

Athena terintegrasi dengan Amazon QuickSight untuk visualisasi data yang mudah. Anda dapat menggunakan Athena untuk menghasilkan laporan atau untuk mengeksplorasi data dengan alat kecerdasan bisnis atau klien SQL yang terhubung dengan driver JDBC atau ODBC. Untuk informasi selengkapnya, lihat Apa itu Amazon QuickSight di Panduan QuickSight Pengguna Amazon danMenghubungkan ke Amazon Athena dengan driver ODBC dan JDBC.

Athena terintegrasi dengan AWS Glue Data Catalog, yang menawarkan penyimpanan metadata persisten untuk data Anda di Amazon S3. Ini memungkinkan Anda untuk membuat tabel dan data kueri di Athena berdasarkan penyimpanan metadata pusat yang tersedia di seluruh akun Amazon Web Services Anda dan terintegrasi dengan ETL dan fitur penemuan data. AWS Glue Untuk informasi selengkapnya, lihat Integrasi dengan AWS Glue dan Apa yang dimaksud AWS Glue dalam AWS Glue Panduan Developer.

Amazon Athena memudahkan menjalankan kueri interaktif terhadap data langsung di Amazon S3 tanpa harus memformat data atau mengelola infrastruktur. Misalnya, Athena berguna jika Anda ingin menjalankan kueri cepat di log web untuk memecahkan masalah kinerja di situs Anda. Dengan Athena, Anda dapat memulai dengan cepat: Anda hanya menentukan tabel untuk data Anda dan mulai melakukan kueri menggunakan SQL standar.

Anda harus menggunakan Amazon Athena jika Anda ingin menjalankan kueri SQL ad hoc interaktif terhadap data di Amazon S3, tanpa harus mengelola infrastruktur atau cluster apa pun. Amazon Athena menyediakan cara termudah untuk menjalankan kueri ad hoc untuk data di Amazon S3 tanpa perlu mengatur atau mengelola server apa pun.

Untuk daftar yang dimanfaatkan atau diintegrasikan Athena, lihat. Layanan AWS the section called “Layanan AWS Integrasi dengan Athena”

Amazon EMR

Amazon EMR membuatnya sederhana dan hemat biaya untuk menjalankan kerangka kerja pemrosesan yang sangat terdistribusi seperti Hadoop, Spark, dan Presto jika dibandingkan

dengan penerapan lokal. Amazon EMR fleksibel — Anda dapat menjalankan aplikasi dan kode khusus, dan menentukan parameter komputasi, memori, penyimpanan, dan aplikasi tertentu untuk mengoptimalkan persyaratan analitik Anda.

Selain menjalankan kueri SQL, Amazon EMR dapat menjalankan berbagai tugas pemrosesan data skala untuk aplikasi seperti pembelajaran mesin, analisis grafik, transformasi data, streaming data, dan hampir semua hal yang dapat Anda kodekan. Anda harus menggunakan Amazon EMR jika Anda menggunakan kode khusus untuk memproses dan menganalisis kumpulan data yang sangat besar dengan kerangka kerja pemrosesan data besar terbaru seperti Spark, Hadoop, Presto, atau Hbase. Amazon EMR memberi Anda kontrol penuh atas konfigurasi cluster Anda dan perangkat lunak yang diinstal pada mereka.

Anda dapat menggunakan Amazon Athena untuk menanyakan data yang Anda proses menggunakan Amazon EMR. Amazon Athena mendukung banyak format data yang sama dengan Amazon EMR. Katalog data Athena kompatibel dengan Hive metastore. Jika Anda menggunakan EMR dan sudah memiliki metastore Hive, Anda dapat menjalankan pernyataan DDL Anda di Amazon Athena dan segera menanyakan data Anda tanpa memengaruhi pekerjaan EMR Amazon Anda.

Amazon Redshift

Gudang data seperti Amazon Redshift adalah pilihan terbaik Anda ketika Anda perlu mengumpulkan data dari berbagai sumber — seperti sistem inventaris, sistem keuangan, dan sistem penjualan ritel — ke dalam format umum, dan menyimpannya untuk jangka waktu yang lama. Jika Anda ingin membuat laporan bisnis canggih dari data historis, maka gudang data seperti Amazon Redshift adalah pilihan terbaik. Mesin kueri di Amazon Redshift telah dioptimalkan untuk berkinerja sangat baik dalam menjalankan kueri kompleks yang menggabungkan sejumlah besar tabel database yang sangat besar. Saat Anda perlu menjalankan kueri terhadap data yang sangat terstruktur dengan banyak gabungan di banyak tabel yang sangat besar, pilih Amazon Redshift.

Untuk informasi lebih lanjut tentang kapan harus menggunakan Athena, lihat sumber daya berikut:

- Panduan keputusan untuk layanan analitik AWS di Pusat Sumber Daya Memulai
- Kapan menggunakan Athena vs layanan data besar lainnya di FAQ Amazon Athena
- Ikhtisar Amazon Athena
- Fitur Amazon Athena
- Amazon Athena FAQ
- Postingan blog Amazon Athena

Layanan AWS Integrasi dengan Athena

Anda dapat menggunakan Athena untuk menanyakan data dari yang Layanan AWS tercantum di bagian ini. Untuk melihat Wilayah yang didukung oleh setiap layanan, lihat Wilayah dan titik akhir di. Referensi Umum Amazon Web Services

Layanan AWS Terintegrasi dengan Athena

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

Untuk informasi selengkapnya tentang setiap integrasi, lihat bagian berikut:

AWS CloudFormation

Pencadangan kapasitas

Topik referensi: AWS: :Athena:: CapacityReservation di Panduan Pengguna AWS CloudFormation

Menentukan reservasi kapasitas dengan nama yang disediakan dan jumlah unit pemrosesan data yang diminta. Untuk informasi selengkapnya, lihat Mengelola kapasitas pemrosesan kueri di Panduan Pengguna Amazon Athena dan CreateCapacityReservationReferensi API Amazon Athena.

Katalog data

Topik referensi: AWS: :Athena:: DataCatalog di Panduan Pengguna AWS CloudFormation

Tentukan katalog data Athena, termasuk nama, deskripsi, tipe, parameter, dan tanda. Untuk informasi selengkapnya, lihat Memahami tabel, database, dan katalog data di Panduan Pengguna Amazon Athena dan CreateDataCatalogReferensi API Amazon Athena.

Kueri bernama

Topik referensi: AWS: :Athena:: NamedQuery di Panduan Pengguna AWS CloudFormation

Tentukan kueri bernama dengan AWS CloudFormation dan jalankan di Athena. Kueri bernama memungkinkan Anda memetakan nama kueri untuk kueri kemudian menjalankannya sebagai kueri tersimpan dari konsol Athena. Untuk informasi selengkapnya, lihat Menggunakan kueri yang disimpan di Panduan Pengguna Amazon Athena dan CreateNamedQueryReferensi API Amazon Athena.

Pernyataan yang disiapkan

Topik referensi: AWS: :Athena:: PreparedStatement di Panduan Pengguna AWS CloudFormation

Menentukan pernyataan disiapkan untuk digunakan dengan query SQL di Athena. Sebuah pernyataan yang disiapkan berisi placeholder parameter yang nilainya disuplai pada waktu eksekusi. Untuk informasi selengkapnya, lihat Menggunakan kueri berparameter di Panduan Pengguna Amazon Athena dan CreatePreparedStatementReferensi API Amazon Athena.

Grup Kerja

Topik referensi: AWS: :Athena:: WorkGroup di Panduan Pengguna AWS CloudFormation

Tentukan kelompok kerja Athena menggunakan. AWS CloudFormation Gunakan grup kerja Athena untuk mengisolasi kueri untuk Anda atau grup Anda dari kueri lain di akun yang sama. Untuk informasi selengkapnya, lihat Menggunakan kelompok kerja untuk mengontrol akses kueri dan biaya di Panduan Pengguna Amazon Athena dan CreateWorkGroupReferensi API Amazon Athena.

Amazon CloudFront

Topik referensi: Menanyakan log Amazon CloudFront

Gunakan Athena untuk menanyakan log Amazon CloudFront . Untuk informasi selengkapnya tentang penggunaan CloudFront, lihat Panduan CloudFront Pengembang Amazon.

AWS CloudTrail

Topik referensi: Meminta log AWS CloudTrail

Menggunakan Athena dengan CloudTrail log adalah cara ampuh untuk meningkatkan analisis aktivitas AWS layanan Anda. Misalnya, Anda dapat menggunakan kueri untuk mengidentifikasi tren dan mengisolasi aktivitas selengkapnya berdasarkan atribut seperti alamat IP sumber atau pengguna. Anda dapat membuat tabel untuk menanyakan log langsung dari CloudTrail konsol, dan menggunakan tabel tersebut untuk menjalankan kueri di Athena. Untuk informasi selengkapnya, lihat Menggunakan CloudTrail konsol untuk membuat tabel Athena untuk log CloudTrail .

Amazon DataZone

Topik referensi: Menggunakan Amazon DataZone di Athena

Gunakan Amazon DataZone untuk berbagi, mencari, dan menemukan data dalam skala besar melintasi batas-batas organisasi. DataZonemenyederhanakan pengalaman Anda di seluruh layanan AWS analitik seperti Athena AWS Glue,, dan. AWS Lake Formation Jika Anda memiliki sejumlah besar data dalam sumber data yang berbeda, Anda dapat menggunakan Amazon DataZone untuk membangun pengelompokan orang, data, dan alat berbasis kasus penggunaan bisnis.

Di Athena, Anda dapat menggunakan editor kueri untuk mengakses dan menanyakan DataZone lingkungan. Untuk informasi selengkapnya, lihat Menggunakan Amazon DataZone di Athena.

Elastic Load Balancing

Topik referensi: Meminta log Application Load Balancer

Kueri Application Load Balancer log memungkinkan Anda melihat sumber lalu lintas, latency, dan byte yang ditransfer ke dan dari contoh Elastic Load Balancing dan aplikasi backend. Untuk informasi selengkapnya, lihat Meminta log Application Load Balancer.

Topik referensi: Meminta log Classic Load Balancer

Kueri log Classic Load Balancer untuk menganalisis dan memahami pola lalu lintas ke dan dari instans Elastic Load Balancing dan aplikasi backend. Anda dapat melihat sumber lalu lintas, latensi, dan bita yang ditransfer. Untuk informasi selengkapnya, lihat Membuat tabel untuk Log ELB.

Studio EMR Amazon

Topik referensi: Gunakan editor SQL Amazon Athena di EMR Studio

Anda dapat menggunakan Athena di EMR Studio untuk mengembangkan dan menjalankan kueri interaktif. Hal ini memungkinkan Anda untuk menggunakan EMR Studio untuk analisis SQL di Athena dari antarmuka EMR Amazon yang sama yang Anda gunakan untuk Spark, Scala, dan beban kerja lainnya. Dengan integrasi Athena di EMR Studio, Anda dapat melakukan tugas-tugas berikut:

- Lakukan kueri Athena SQL
- Lihat hasil kueri
- Lihat riwayat kueri
- Lihat kueri yang disimpan
- Lakukan kueri berparameter
- Melihat database, tabel, dan tampilan untuk katalog data

Fitur Athena berikut tidak tersedia di Amazon EMR Studio:

- Fitur admin seperti membuat atau memperbarui workgroup Athena, sumber data, atau reservasi kapasitas
- Athena untuk notebook Spark atau Spark
- DataZone integrasi
- Step Functions

Integrasi EMR Studio dengan Athena tersedia di semua tempat Wilayah AWS EMR Studio dan Athena tersedia. Untuk informasi selengkapnya tentang penggunaan Athena di EMR Studio, lihat Menggunakan editor SQL Amazon Athena di EMR Studio di Panduan Manajemen EMR Amazon.

AWS Glue Data Catalog

Topik referensi: Integrasi dengan AWS Glue

Athena terintegrasi dengan AWS Glue Data Catalog, yang menawarkan penyimpanan metadata persisten untuk data Anda di Amazon S3. Ini memungkinkan Anda untuk membuat tabel dan data kueri di Athena berdasarkan penyimpanan metadata pusat yang tersedia di seluruh akun Amazon Web Services Anda dan terintegrasi dengan ETL dan fitur penemuan data. AWS Glue Untuk informasi selengkapnya, lihat Integrasi dengan AWS Glue dan Apa yang ada AWS Glue di Panduan AWS Glue Pengembang.

AWS Identity and Access Management (IAM)

Topik referensi: Tindakan untuk Amazon Athena

Anda dapat menggunakan API Athena dalam kebijakan izin IAM. Untuk informasi selengkapnya, lihat Tindakan untuk Amazon Athena dan Manajemen identitas dan akses di Athena.

Amazon QuickSight

Topik referensi: Menghubungkan ke Amazon Athena dengan driver ODBC dan JDBC

Athena terintegrasi dengan Amazon QuickSight untuk visualisasi data yang mudah. Anda dapat menggunakan Athena untuk menghasilkan laporan atau untuk mengeksplorasi data dengan alat kecerdasan bisnis atau klien SQL yang terhubung dengan driver JDBC atau ODBC. Untuk informasi selengkapnya tentang Amazon QuickSight, lihat Apa itu Amazon QuickSight di Panduan QuickSight Pengguna Amazon. Untuk informasi tentang menggunakan driver JDBC dan ODBC dengan Athena, lihat Menghubungkan ke Amazon Athena dengan Driver ODBC dan JDBC.

Persediaan Amazon S3

Topik referensi: Menanyakan inventaris dengan Athena di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon

Anda dapat menggunakan Amazon Athena untuk melakukan kueri inventaris Amazon S3 menggunakan SQL standar. Gunakan inventaris Amazon S3 untuk mengaudit dan melaporkan replikasi dan status enkripsi objek Anda untuk kebutuhan bisnis, kepatuhan, dan regulasi. Untuk informasi selengkapnya, lihat inventaris Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

AWS Step Functions

Topik referensi: Panggil Athena dengan Step Functions dalam AWS Step Functions Panduan Developer

Hubungi Athena dengan. AWS Step Functions AWS Step Functions dapat mengontrol pilih Layanan AWS langsung menggunakan Amazon States Language. Anda dapat menggunakan Step Functions dengan Athena untuk memulai dan menghentikan eksekusi kueri, mendapatkan hasil kueri, menjalankan ad-hoc atau dijadwalkan permintaan data, dan mengambil hasil dari danau data di Amazon S3. Peran Step Functions harus memiliki izin untuk menggunakan Athena. Lihat informasi selengkapnya di Panduan Developer AWS Step Functions.

Video: Mengatur Kueri Amazon Athena menggunakan AWS Step Functions

Video berikut menunjukkan cara menggunakan Amazon Athena AWS Step Functions dan menjalankan kueri Athena yang dijadwalkan secara teratur dan menghasilkan laporan yang sesuai.

Mengatur kueri Amazon Athena menggunakan AWS Step Functions

Untuk contoh yang menggunakan Step Functions dan Amazon EventBridge untuk mengatur, AWS Glue DataBrew Athena, dan QuickSight Amazon, lihat AWS Glue DataBrew Mengatur pekerjaan dan kueri Amazon Athena di Big Data Blog. AWS Step Functions AWS

AWS Systems Manager Inventaris

Topik referensi: Menanyakan data inventaris dari beberapa wilayah dan akun di AWS Systems Manager Panduan Pengguna

AWS Systems Manager Inventaris terintegrasi dengan Amazon Athena untuk membantu Anda menanyakan data inventaris dari Wilayah AWS beberapa dan akun. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna AWS Systems Manager.

Amazon Virtual Private Cloud

Topik referensi: Menanyakan log aliran VPC Amazon

Log alur Amazon Virtual Private Cloud menangkap informasi tentang lalu lintas IP ke dan dari antarmuka jaringan di VPC Anda. Kueri log di Athena untuk menyelidiki pola lalu lintas jaringan dan mengidentifikasi ancaman dan risiko di seluruh jaringan VPC Amazon Anda. Untuk informasi tentang Amazon VPC selengkapnya, lihat Panduan Pengguna Amazon VPC.

Mengatur

Jika Anda sudah mendaftar untuk Amazon Web Services, Anda dapat segera mulai menggunakan Amazon Athena. Jika Anda belum mendaftar AWS atau membutuhkan bantuan untuk memulai, pastikan untuk menyelesaikan tugas-tugas berikut.

Mendaftar untuk Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran melibatkan tindakan menerima panggilan telepon dan memasukkan kode verifikasi di keypad telepon.

Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya di akun. Sebagai praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS mengirimi Anda email konfirmasi setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun Anda saat ini dan mengelola akun Anda dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

Buat pengguna dengan akses administratif

Setelah Anda mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Amankan Anda Pengguna root akun AWS

1. Masuk ke AWS Management Consolesebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat Akun AWS email Anda. Di laman berikutnya, masukkan kata sandi.

   Untuk bantuan masuk dengan menggunakan pengguna root, lihat Masuk sebagai pengguna root di AWS Sign-In Panduan Pengguna.
2. Mengaktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuk, lihat Mengaktifkan perangkat MFA virtual untuk pengguna Akun AWS root (konsol) Anda di Panduan Pengguna IAM.

Buat pengguna dengan akses administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center .
2. Di Pusat Identitas IAM, berikan akses administratif ke pengguna.

   Untuk tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, lihat Mengkonfigurasi akses pengguna dengan default Direktori Pusat Identitas IAM di Panduan AWS IAM Identity Center Pengguna.

Masuk sebagai pengguna dengan akses administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email saat Anda membuat pengguna Pusat Identitas IAM.

  Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal AWS akses di Panduan AWS Sign-In Pengguna.

Tetapkan akses ke pengguna tambahan

1. Di Pusat Identitas IAM, buat set izin yang mengikuti praktik terbaik menerapkan izin hak istimewa paling sedikit.

   Untuk petunjuknya, lihat Membuat set izin di Panduan AWS IAM Identity Center Pengguna.
2. Tetapkan pengguna ke grup, lalu tetapkan akses masuk tunggal ke grup.

   Untuk petunjuk, lihat Menambahkan grup di Panduan AWS IAM Identity Center Pengguna.

Memberikan akses programatis

Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS luar. AWS Management Console Cara untuk memberikan akses terprogram tergantung pada jenis pengguna yang mengakses AWS.

Untuk memberi pengguna akses programatis, pilih salah satu opsi berikut.

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
Identitas tenaga kerja (Pengguna yang dikelola di Pusat Identitas IAM)	Gunakan kredenal sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk AWS CLI, lihat Mengkonfigurasi yang akan AWS CLI digunakan AWS IAM Identity Center dalam

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
		Panduan AWS Command Line Interface Pengguna. • Untuk AWS SDK, alat, dan AWS API, lihat autentikasi Pusat Identitas IAM di Panduan Referensi AWS SDK dan Alat.
IAM	Gunakan kredenal sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS	Mengikuti petunjuk dalam Menggunakan kredensil sementara dengan AWS sumber daya di Panduan Pengguna IAM.

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
IAM	(Tidak direkomendasikan) Gunakan kredensil jangka panjang untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk mengetahui AWS CLI, lihat Mengautentikasi menggunakan kredensil pengguna IAM di Panduan Pengguna.AWS Command Line Interface • Untuk AWS SDK dan alat bantu, lihat Mengautentikasi menggunakan kredensil jangka panjang di Panduan Referensi AWS SDK dan Alat. • Untuk AWS API, lihat Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM.

Lampirkan kebijakan terkelola untuk Athena

Kebijakan terkelola Athena memberikan izin untuk menggunakan fitur Athena. Anda dapat melampirkan kebijakan terkelola ini ke satu atau beberapa peran IAM yang dapat diasumsikan pengguna untuk menggunakan Athena.

Sebuah peran IAM adalah identitas IAM yang dapat Anda buat di akun yang memiliki izin tertentu. Peran IAM mirip dengan pengguna IAM karena merupakan AWS identitas dengan kebijakan izin yang menentukan apa yang dapat dan tidak dapat dilakukan identitas. AWS Namun, alih-alih secara unik terkait dengan satu orang, peran dimaksudkan untuk menjadi dapat diambil oleh siapa pun yang membutuhkannya. Selain itu, peran tidak memiliki kredensial jangka panjang standar seperti kata

sandi atau kunci akses yang terkait dengannya. Sebagai gantinya, saat Anda mengambil peran, peran tersebut akan memberikan kredensial keamanan sementara untuk sesi peran.

Untuk informasi selengkapnya tentang peran, lihat peran IAM dan Membuat peran IAM di Panduan Pengguna IAM.

Untuk membuat peran yang memberikan akses ke Athena, Anda melampirkan kebijakan yang dikelola Athena ke peran tersebut. Ada dua kebijakan yang dikelola untuk Athena: `AmazonAthenaFullAccess` dan. `AWSQuicksightAthenaAccess` Kebijakan ini memberikan izin kepada Athena untuk menanyakan Amazon S3 dan menulis hasil kueri Anda ke bucket terpisah atas nama Anda. Untuk melihat isi kebijakan Athena ini, lihat. AWS kebijakan terkelola untuk Amazon Athena

Untuk langkah-langkah untuk melampirkan kebijakan terkelola Athena ke peran, ikuti Menambahkan izin identitas IAM (konsol) di Panduan Pengguna IAM dan tambahkan dan kebijakan `AWSQuicksightAthenaAccess` terkelola ke peran yang Anda buat. `AmazonAthenaFullAccess`

> **Note**
>
> Anda mungkin memerlukan izin tambahan untuk mengakses kumpulan data yang mendasarinya di Amazon S3. Jika Anda bukan pemilik akun atau membatasi akses ke bucket, hubungi pemilik bucket untuk memberikan akses menggunakan kebijakan bucket berbasis sumber daya, atau hubungi administrator akun Anda untuk memberikan akses menggunakan kebijakan berbasis peran. Untuk informasi selengkapnya, lihat Akses ke Amazon S3 dari Athena. Jika dataset atau hasil kueri Athena dienkripsi, Anda mungkin memerlukan izin tambahan. Untuk informasi selengkapnya, lihat Enkripsi diam.

Mengakses Athena

Anda dapat mengakses Athena menggunakan AWS Management Console, koneksi JDBC atau ODBC, Athena API, CLI Athena, SDK, atau. AWS AWS Tools for Windows PowerShell

- Untuk mulai menggunakan Athena SQL dengan konsol, lihat. Memulai
- Untuk mulai membuat notebook yang kompatibel dengan Jupyter dan aplikasi Apache Spark yang menggunakan Python, lihat. Menggunakan Apache Spark di Amazon Athena
- Untuk mempelajari cara menggunakan driver JDBC atau ODBC, lihat dan. Menghubungkan ke Amazon Athena dengan JDBC Menghubungkan ke Amazon Athena dengan ODBC

- Untuk menggunakan Athena API, lihat Referensi API Amazon Athena.
- Untuk menggunakan CLI, instal AWS CLI dan kemudian ketik `aws athena help` dari baris perintah untuk melihat perintah yang tersedia. Untuk informasi tentang perintah yang tersedia, lihat referensi baris perintah Amazon Athena.
- Untuk menggunakan AWS SDK for Java 2.x, lihat bagian Athena dari Referensi AWS SDK for Java 2.x API, contoh Athena Java V2 di GitHub .com, dan Panduan Pengembang.AWS SDK for Java 2.x
- Untuk menggunakan AWS SDK for .NET, lihat `Amazon.Athena` namespace di Referensi AWS SDK for .NET API, contoh.NET Athena di GitHub .com, dan Panduan Pengembang.AWS SDK for .NET
- Untuk menggunakan AWS Tools for Windows PowerShell, lihat referensi cmdlet AWS Tools for PowerShell - Amazon Athena, halaman AWS Tools for PowerShellportal, dan Panduan Pengguna.AWS Tools for Windows PowerShell
- Untuk informasi tentang titik akhir layanan Athena yang dapat Anda sambungkan secara terprogram, lihat titik akhir dan kuota Amazon Athena di. Referensi Umum Amazon Web Services

Menggunakan Athena SQL

Anda dapat menggunakan Athena SQL untuk menanyakan data Anda di tempat di Amazon S3 menggunakan metastore Hive eksternal AWS Glue Data Catalog, atau kueri gabungan menggunakan berbagai konektor bawaan ke sumber data lain.

Anda juga dapat:

- Connect ke alat intelijen bisnis dan aplikasi lain menggunakan driver JDBC dan ODBC Athena.
- Log AWS layanan kueri.
- Kueri tabel Apache Iceberg, termasuk kueri perjalanan waktu, dan kumpulan data Apache Hudi.
- Kueri data geospasial.
- Kueri menggunakan inferensi pembelajaran mesin dari Amazon SageMaker.
- Kueri menggunakan fungsi yang ditentukan pengguna Anda sendiri.
- Mempercepat pemrosesan kueri tabel yang sangat dipartisi dan mengotomatiskan manajemen partisi dengan menggunakan proyeksi partisi.

Topik

- Sampel Kode

Memahami tabel, database, dan katalog data

Di Athena, katalog, database, dan tabel adalah wadah untuk definisi metadata yang mendefinisikan skema untuk data sumber yang mendasarinya.

Athena menggunakan istilah berikut untuk merujuk ke hierarki objek data:

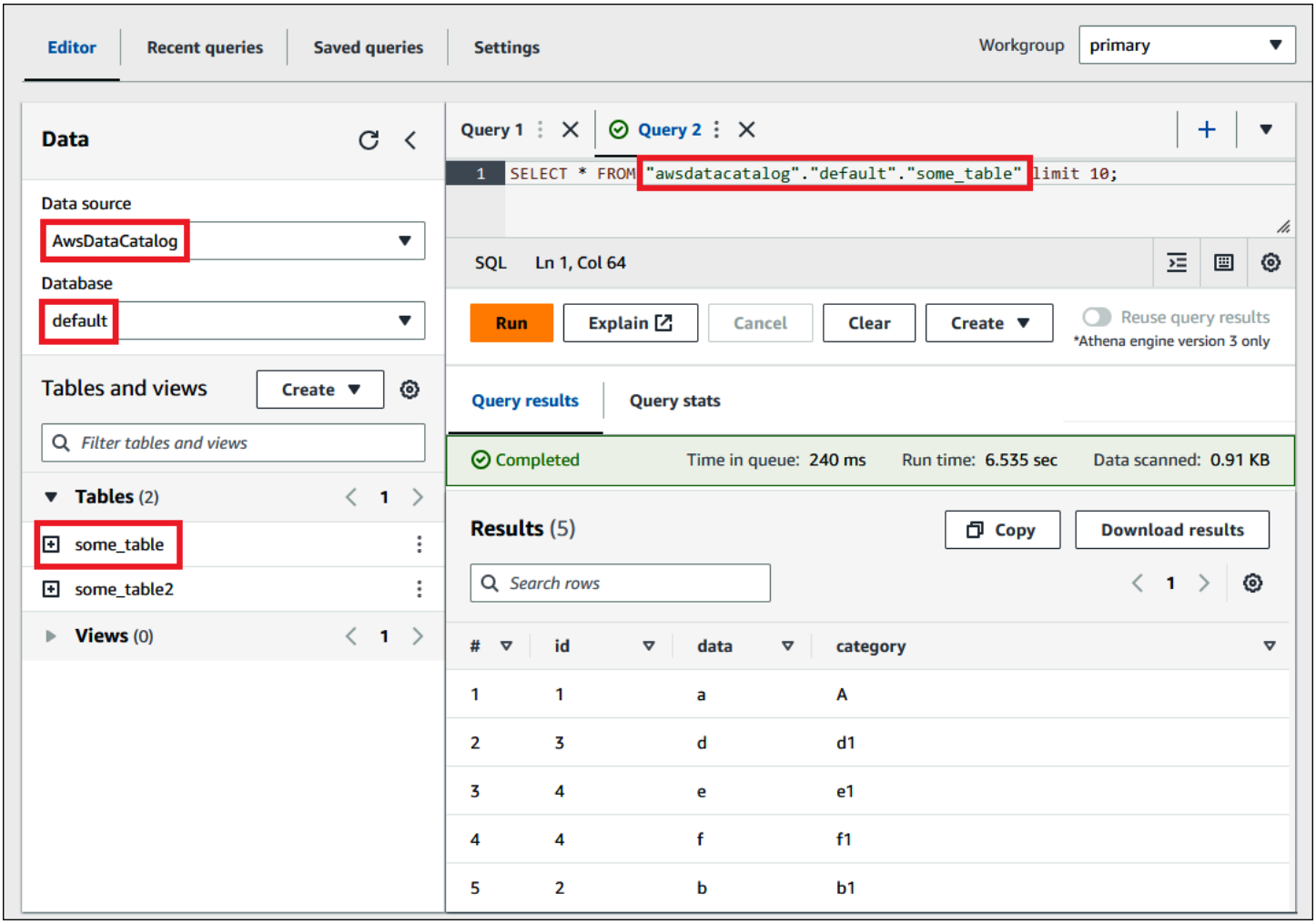
- Sumber data — sekelompok database
- Database — sekelompok tabel
- Tabel — data yang disusun sebagai sekelompok baris atau kolom

Terkadang benda-benda ini juga disebut dengan nama alternatif tetapi setara seperti berikut ini:

- Sumber data terkadang disebut sebagai katalog.
- Database kadang-kadang disebut sebagai skema.

> **Note**
>
> Terminologi ini dapat bervariasi dalam sumber data federasi yang Anda gunakan dengan Athena. Untuk informasi selengkapnya, lihat Athena dan kualifikasi nama tabel federasi.

Contoh query berikut di konsol Athena menggunakan sumber `awsdatacatalog` data, `default` database, dan tabel`some_table`.



Untuk setiap dataset, tabel harus ada di Athena. Metadata dalam tabel memberi tahu Athena tempat data berada di Amazon S3, dan menentukan struktur data, misalnya, nama kolom, tipe data, dan nama tabel. Database adalah pengelompokan tabel yang logis, dan juga hanya menyimpan informasi metadata dan skema untuk kumpulan data.

Untuk setiap kumpulan data yang ingin Anda kueri, Athena harus memiliki tabel dasar yang akan digunakan untuk mendapatkan dan mengembalikan hasil kueri. Karena itu, sebelum menanyakan data, tabel harus didaftarkan di Athena. Pendaftaran terjadi ketika Anda membuat tabel secara otomatis atau manual.

Anda dapat membuat tabel secara otomatis menggunakan AWS Glue crawler. Untuk informasi selengkapnya tentang AWS Glue dan crawler, lihat Integrasi dengan AWS Glue. Ketika AWS Glue membuat tabel, ia mendaftarkannya dalam Katalog AWS Glue Data sendiri. Athena menggunakan Katalog AWS Glue Data untuk menyimpan dan mengambil metadata ini, menggunakannya saat Anda menjalankan kueri untuk menganalisis kumpulan data yang mendasarinya.

Terlepas dari bagaimana tabel dibuat, proses pembuatan tabel mendaftarkan dataset dengan Athena. Pendaftaran ini terjadi di AWS Glue Data Catalog dan memungkinkan Athena untuk menjalankan kueri pada data. Di editor kueri Athena, katalog ini (atau sumber data) dirujuk dengan label. `AwsDataCatalog`

Setelah membuat tabel, Anda dapat menggunakan pernyataan SQL SELECT untuk menanyakannya, termasuk mendapatkan lokasi file tertentu untuk data sumber Anda. Hasil kueri Anda disimpan di Amazon S3 di lokasi hasil kueri yang Anda tentukan.

Katalog AWS Glue Data dapat diakses di seluruh akun Amazon Web Services Anda. Lainnya Layanan AWS dapat berbagi Katalog AWS Glue Data, sehingga Anda dapat melihat database dan tabel yang dibuat di seluruh organisasi Anda menggunakan Athena dan sebaliknya.

- Untuk membuat tabel secara manual:
  - Gunakan konsol Athena untuk menjalankan Create Table Wizard.
  - Gunakan konsol Athena untuk menulis pernyataan Hive DDL di Query Editor.
  - Gunakan Athena API atau CLI untuk menjalankan string kueri SQL dengan pernyataan DDL.
  - Gunakan driver Athena JDBC atau ODBC.

Saat Anda membuat tabel dan database secara manual, Athena menggunakan pernyataan HiveQL data definition language (DDL) `CREATE TABLE` seperti`CREATE DATABASE`,, `DROP TABLE` dan di bawah tenda untuk membuat tabel dan database di. AWS Glue Data Catalog

Untuk memulai, Anda dapat menggunakan tutorial di konsol Athena atau mengerjakan step-by-step panduan dalam dokumentasi Athena.

- Untuk menggunakan tutorial di konsol Athena, pilih ikon informasi di kanan atas konsol, lalu pilih tab Tutorial.
- Untuk step-by-step tutorial tentang membuat tabel dan menulis kueri di editor kueri Athena, lihat. Memulai

Memulai

Tutorial ini akan memandu Anda dalam menggunakan Amazon Athena untuk mengkueri data. Anda akan membuat tabel berdasarkan data sampel yang disimpan di Amazon Simple Storage Service, mengkueri tabel, dan memeriksa hasil kueri.

Tutorial menggunakan sumber daya langsung, sehingga Anda akan dikenai biaya untuk mengkueri yang Anda jalankan. Anda tidak dikenai biaya untuk data sampel di lokasi yang digunakan tutorial ini, tetapi jika Anda mengunggah file data Anda sendiri ke Amazon S3, biaya akan berlaku.

Prasyarat

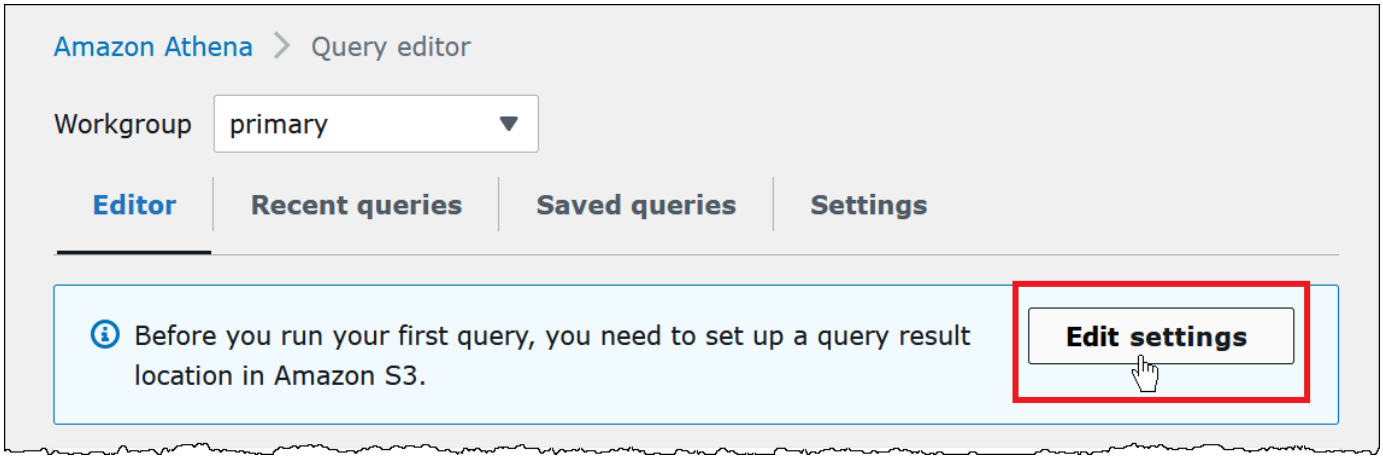
- Jika Anda belum melakukannya, daftar untuk Akun AWS.
- Menggunakan akun yang sama Wilayah AWS (misalnya, US West (Oregon)) dan akun yang Anda gunakan untuk Athena, ikuti langkah-langkah untuk membuat bucket di Amazon S3 untuk menyimpan hasil kueri Athena Anda. Anda akan mengonfigurasi bucket ini menjadi lokasi keluaran kueri Anda.

Langkah 1: Buat database

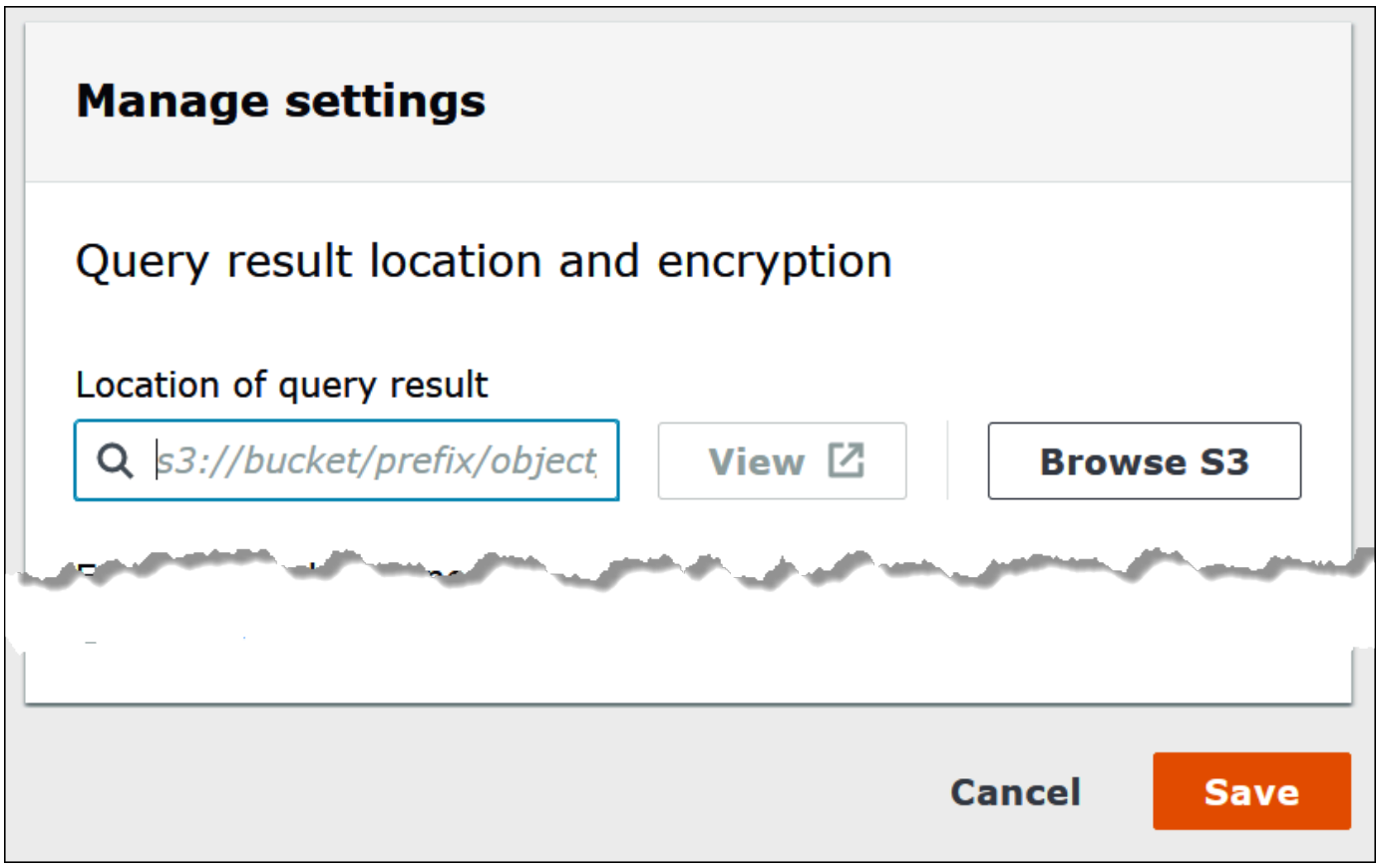
Anda harus terlebih dahulu membuat basis data di Athena.

Untuk membuat database Athena

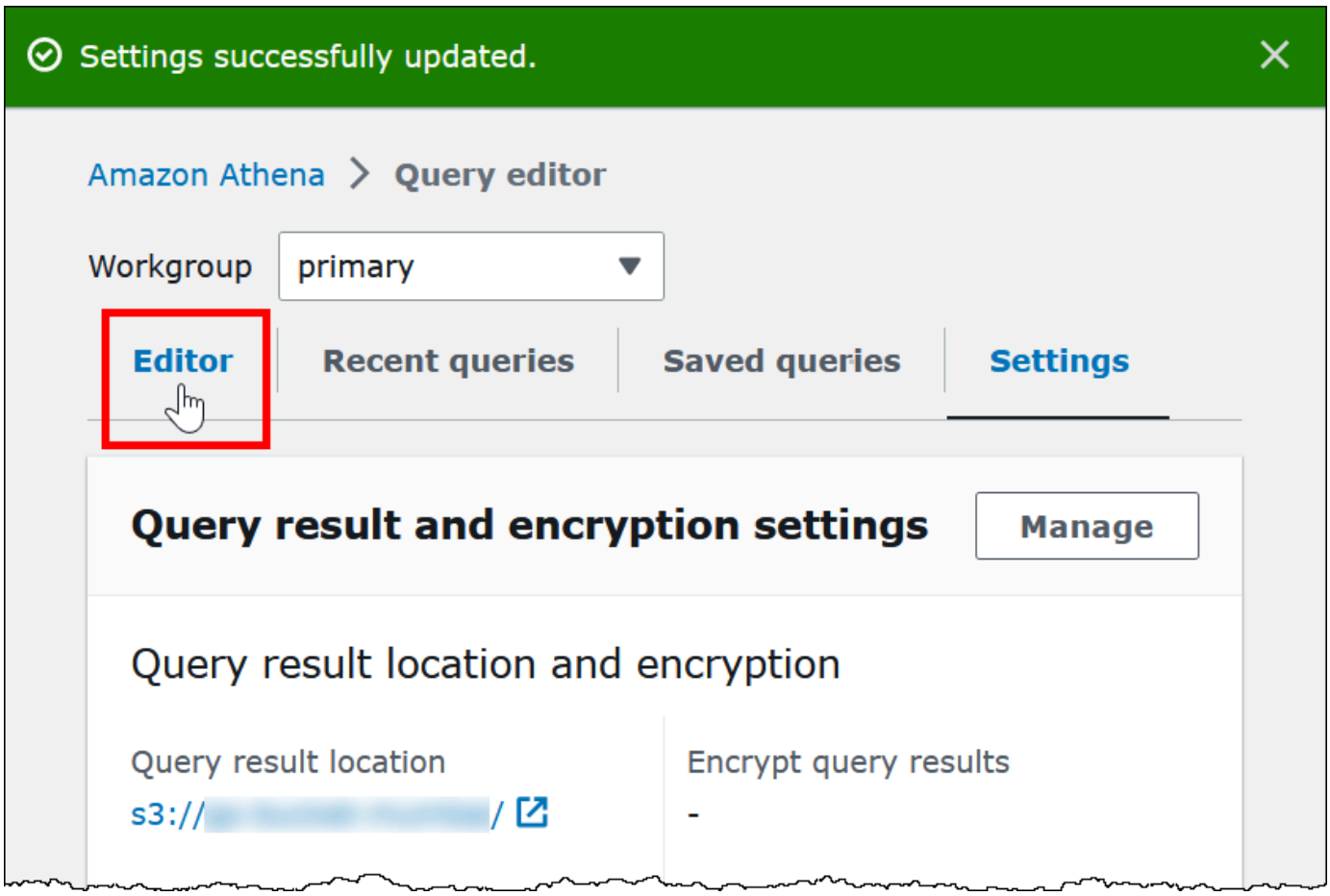
1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika ini adalah pertama kalinya Anda mengunjungi konsol Athena saat ini Wilayah AWS, pilih Jelajahi editor kueri untuk membuka editor kueri. Jika tidak, Athena terbuka di editor kueri.
3. Pilih Edit Pengaturan untuk mengatur lokasi hasil kueri di Amazon S3.



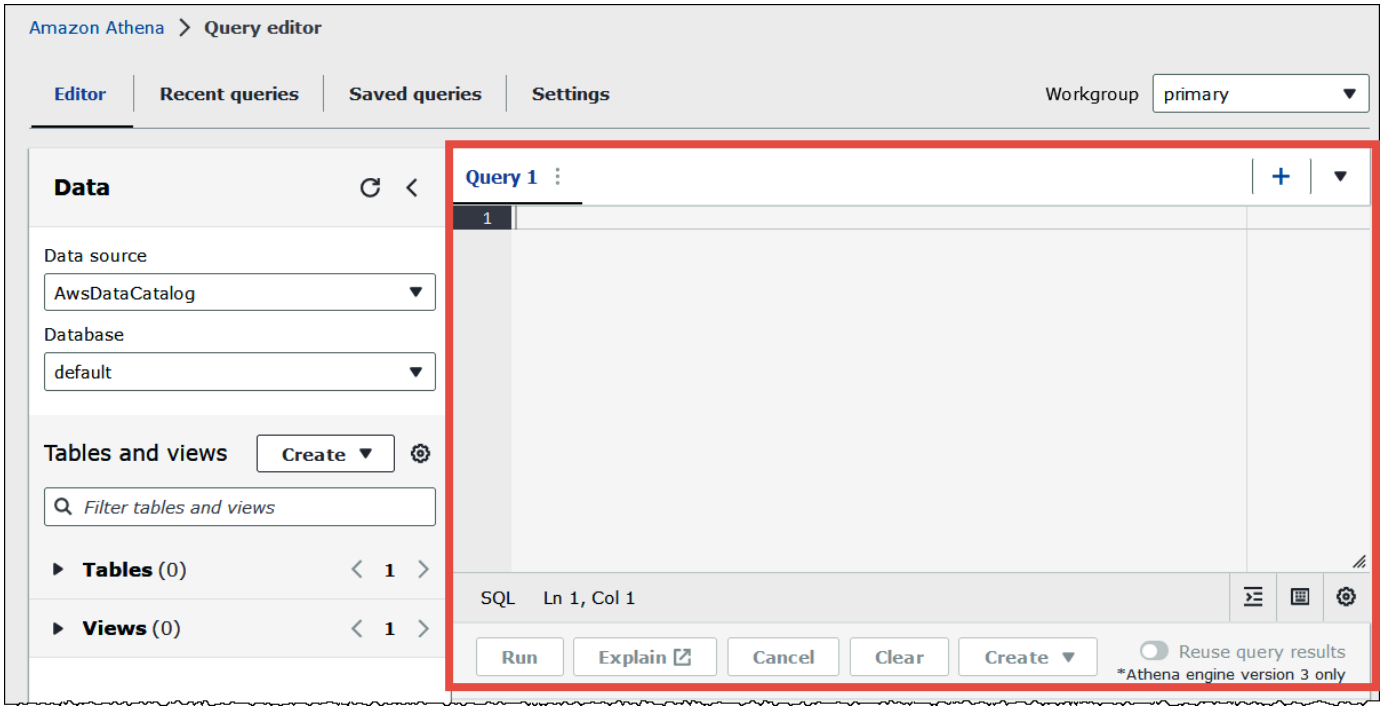
4. Untuk Mengelola pengaturan, lakukan salah satu hal berikut:
   - Di kotak Lokasi hasil kueri, masukkan jalur ke bucket yang Anda buat di Amazon S3 untuk hasil kueri. Awalan jalur dengan `s3://`.
   - Pilih Browse S3, pilih bucket Amazon S3 yang Anda buat untuk Wilayah saat ini, lalu pilih Pilih.



5. Pilih Simpan.
6. Pilih Editor untuk beralih ke editor kueri.



7. Di sebelah kanan panel navigasi, Anda dapat menggunakan editor kueri Athena untuk memasukkan dan menjalankan kueri dan pernyataan.

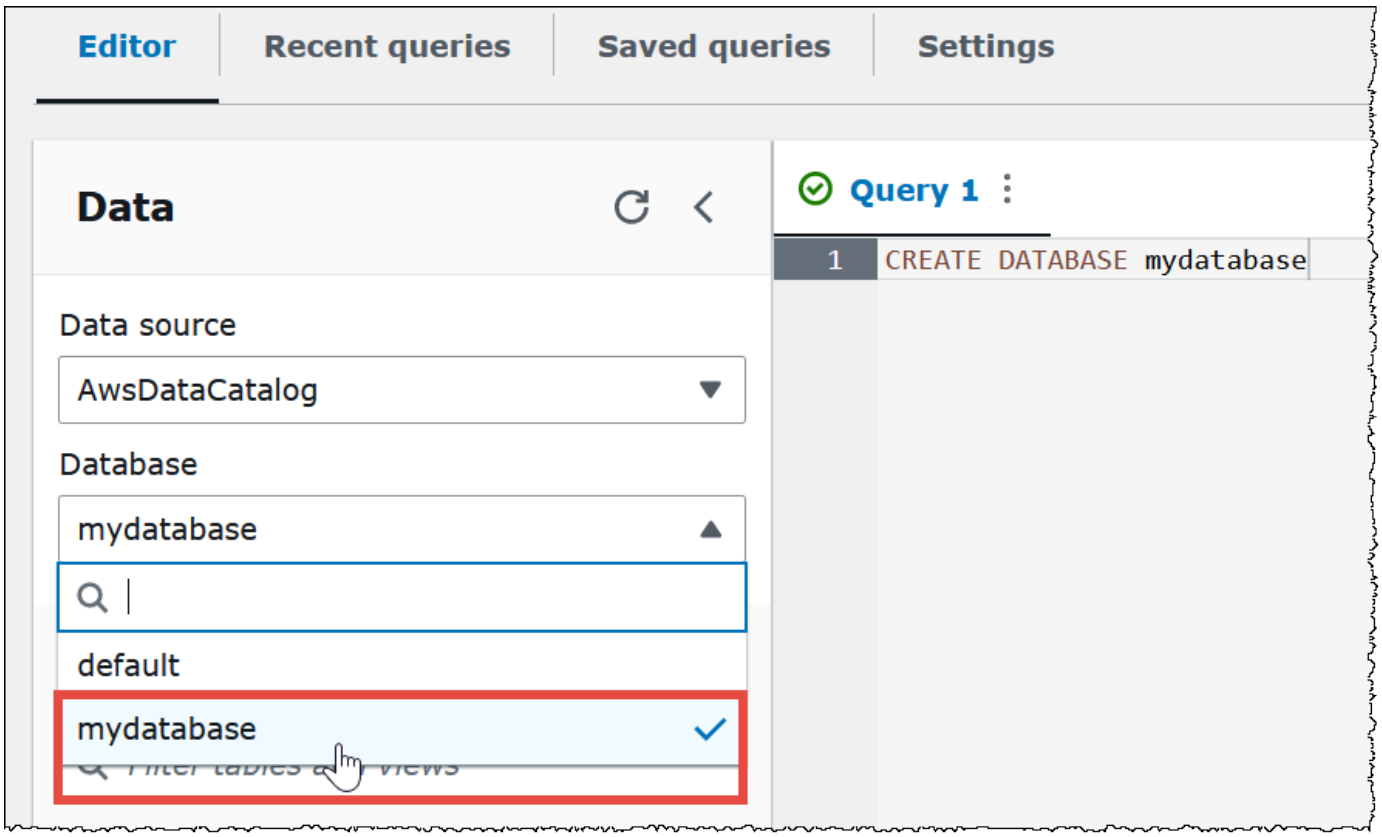


8. Untuk membuat basis data bernama `mydatabase`, masukkan pernyataan CREATE DATABASE berikut.

```
CREATE DATABASE mydatabase
```

9. Pilih Jalankan atau tekan**`Ctrl+ENTER`**.

10. Dari daftar Database di sebelah kiri, pilih `mydatabase` untuk menjadikannya database Anda saat ini.



Langkah 2: Buat Tabel

Setelah kini memiliki basis data, Anda dapat membuat tabel Athena untuk itu. Tabel yang Anda buat akan didasarkan pada contoh data CloudFront log Amazon di lokasi`s3://athena-examples-`*`myregion`*`/cloudfront/plaintext/`, di mana *`myregion`* adalah milik Anda saat ini Wilayah AWS.

Data log sampel dalam format tab-separated values (TSV), yang berarti bahwa karakter tab digunakan sebagai pembatas untuk memisahkan bidang. Data akan terlihat seperti berikut contoh berikut. Untuk keterbacaan, tab dalam kutipan telah dikonversi ke spasi dan bidang akhir dipersingkat.

```
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[...]
```

Untuk mengaktifkan Athena membaca data ini, Anda dapat membuat `CREATE EXTERNAL TABLE` pernyataan langsung seperti berikut ini. Pernyataan yang membuat tabel mendefinisikan kolom yang memetakan ke data, menentukan bagaimana data dibatasi, dan menentukan lokasi Amazon S3 yang berisi data sampel. Perhatikan bahwa karena Athena mengharapkan untuk memindai semua file dalam folder, `LOCATION` klausa menentukan lokasi folder Amazon S3, bukan file tertentu.

Jangan gunakan contoh ini dulu karena memiliki batasan penting yang akan segera dijelaskan.

```
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/';
```

Contoh membuat tabel yang disebut `cloudfront_logs` dan menentukan nama dan tipe data untuk setiap bidang. Bidang ini menjadi kolom dalam tabel. Karena `date` merupakan kata yang dicadangkan, itu lolos dengan karakter backtick (`). `ROW FORMAT DELIMITED`berarti Athena akan menggunakan pustaka default yang dipanggil LazySimpleSerDeuntuk melakukan pekerjaan sebenarnya dari parsing data. Contoh ini juga menetapkan bahwa bidang dipisahkan tab (`FIELDS TERMINATED BY '\t'`) dan bahwa setiap catatan dalam file berakhir dalam karakter baris baru (`LINES TERMINATED BY '\n`). Terakhir, klausa `LOCATION` menentukan jalur di Amazon S3, tempat data aktual yang akan dibaca berada.

Jika Anda memiliki tab sendiri atau data yang dipisahkan koma, Anda dapat menggunakan `CREATE TABLE` pernyataan seperti contoh yang baru saja disajikan—selama bidang Anda tidak berisi informasi bersarang. Namun, jika Anda memiliki kolom seperti `ClientInfo` itu berisi informasi bersarang yang menggunakan pembatas berbeda, diperlukan pendekatan yang berbeda.

Mengekstrak data dari lapangan ClientInfo

Melihat data sampel, berikut adalah contoh lengkap dari bidang akhir`ClientInfo`:

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

Seperti yang bisa Anda lihat, bidang ini memiliki beberapa nilai. Karena `CREATE TABLE` pernyataan contoh yang baru saja disajikan menentukan tab sebagai pembatas bidang, itu tidak dapat memecah komponen terpisah di dalam `ClientInfo` bidang menjadi kolom terpisah. Jadi, diperlukan `CREATE TABLE` pernyataan baru.

Untuk membuat kolom dari nilai di dalam `ClientInfo` bidang, Anda dapat menggunakan ekspresi reguler (regex) yang berisi grup regex. grup regex yang Anda tentukan menjadi kolom tabel terpisah. Untuk menggunakan regex di pernyataan `CREATE TABLE`, gunakan sintaks seperti berikut. Sintaks ini menginstruksikan Athena untuk menggunakan Regex SerDe dan ekspresi reguler yang Anda tentukan.

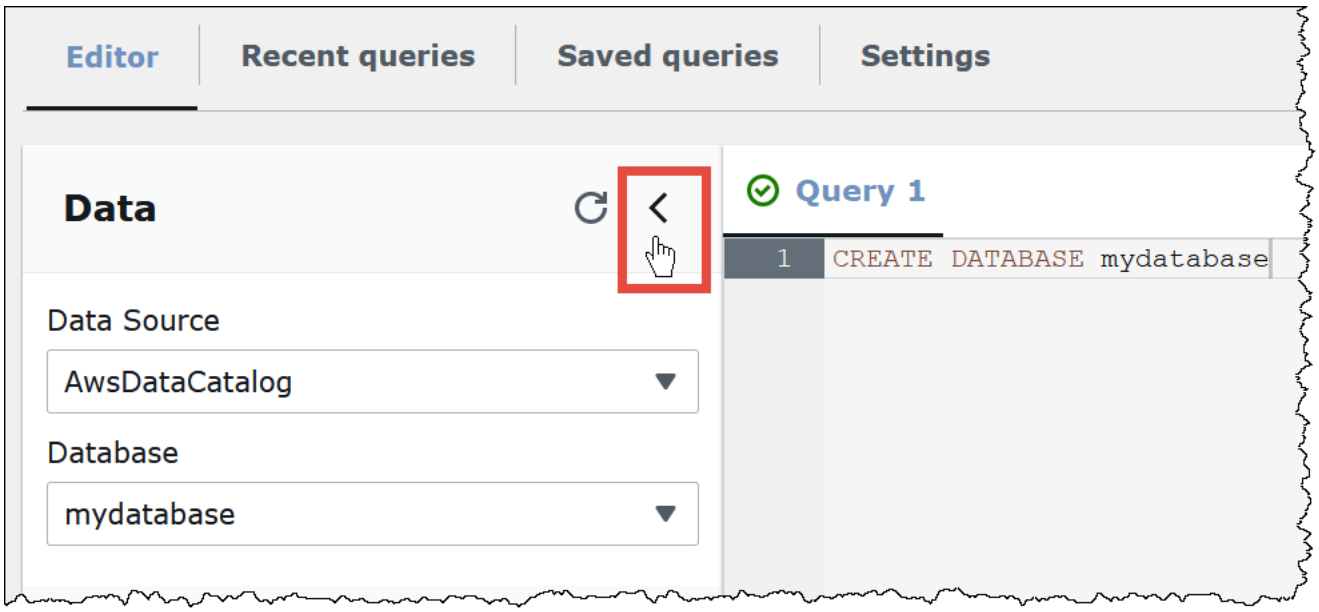
```
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
  WITH SERDEPROPERTIES ("input.regex" = "regular_expression")
```

Ekspresi reguler dapat berguna untuk membuat tabel dari data CSV atau TSV kompleks, tetapi bisa sulit untuk penulisan dan pemeliharaan. Untungnya, ada pustaka lain yang dapat Anda gunakan untuk format seperti JSON, Parquet, dan ORC. Untuk informasi selengkapnya, lihat Format yang didukung SerDes dan data.

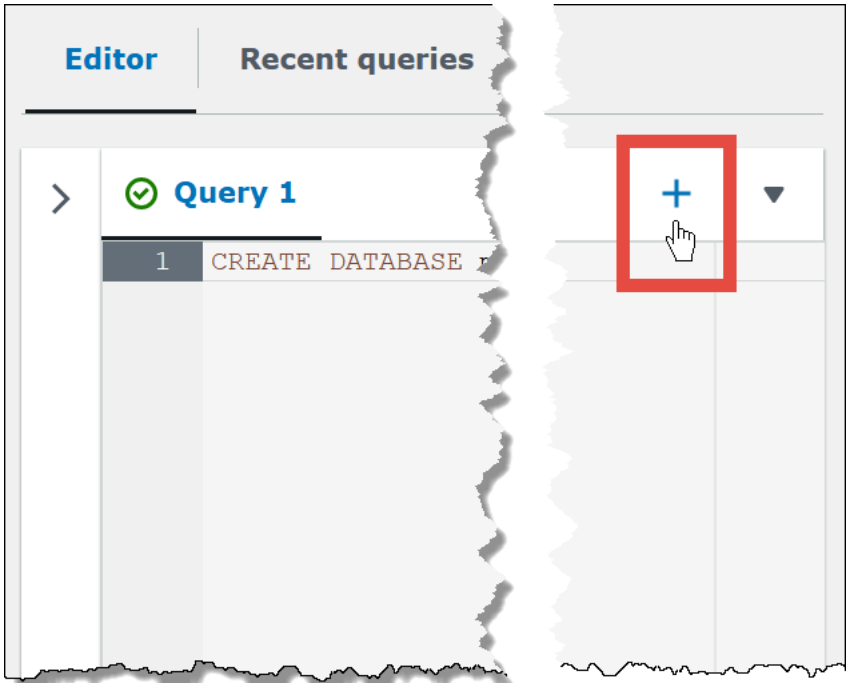
Sekarang Anda siap untuk membuat tabel di editor kueri Athena. Pernyataan dan regex `CREATE TABLE` disediakan untuk Anda.

Untuk membuat tabel di Athena

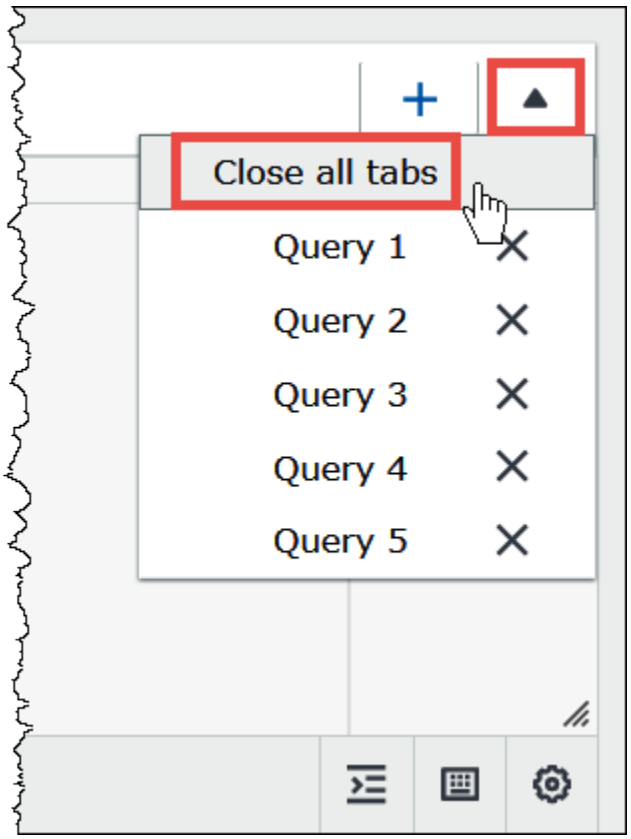
1. Di panel navigasi, untuk Database, pastikan yang `mydatabase` dipilih.
2. Untuk memberi diri Anda lebih banyak ruang di editor kueri, Anda dapat memilih ikon panah untuk menutup panel navigasi.



3. Untuk membuat tab untuk kueri baru, pilih tanda plus (+) di editor kueri. Anda dapat membuka hingga sepuluh tab kueri sekaligus.



4. Untuk menutup satu atau beberapa tab kueri, pilih panah di sebelah tanda plus. Untuk menutup semua tab sekaligus, pilih panah, lalu pilih Tutup semua tab.



5. Dalam panel kueri, masukkan persyaratan `CREATE EXTERNAL TABLE` berikut. Regex memecahkan sistem operasi, browser, dan informasi versi browser dari bidang `ClientInfo` di data log.

> **Note**
>
> Regex yang digunakan dalam contoh berikut dirancang untuk bekerja dengan data CloudFront log sampel yang tersedia untuk umum di lokasi `athena-examples` Amazon S3 dan hanya ilustratif. Untuk up-to-date regex lainnya yang menanyakan file CloudFront log standar dan real-time, lihat. Menanyakan log Amazon CloudFront

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

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

6. Dalam `LOCATION` pernyataan tersebut, ganti *`myregion`* dengan Wilayah AWS yang sedang Anda gunakan (misalnya,`us-west-1`).
7. Pilih Jalankan.

   Tabel `cloudfront_logs` dibuat dan muncul di bawah daftar Tabel untuk basis data `mydatabase`.

Langkah 3: Kueri data

Sekarang setelah Anda membuat tabel `cloudfront_logs` di Athena berdasarkan data di Amazon S3, Anda dapat menjalankan kueri SQL pada tabel dan melihat hasilnya di Athena. Untuk informasi selengkapnya tentang penggunaan SQL di Athena, lihat Referensi SQL untuk Athena.

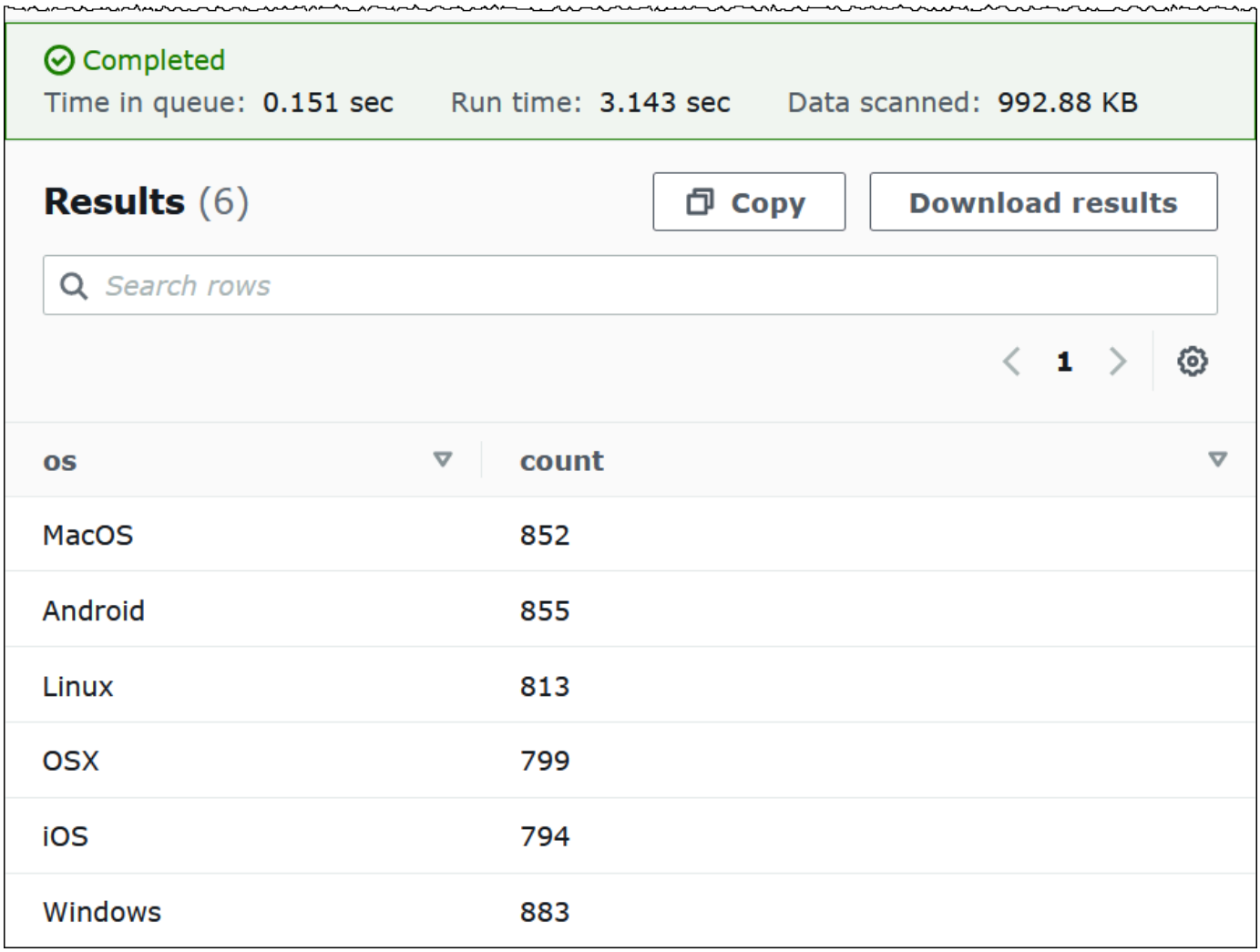
Untuk menjalankan kueri

1. Pilih tanda plus (+) untuk membuka tab kueri baru dan masukkan pernyataan SQL berikut di panel permintaan.

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

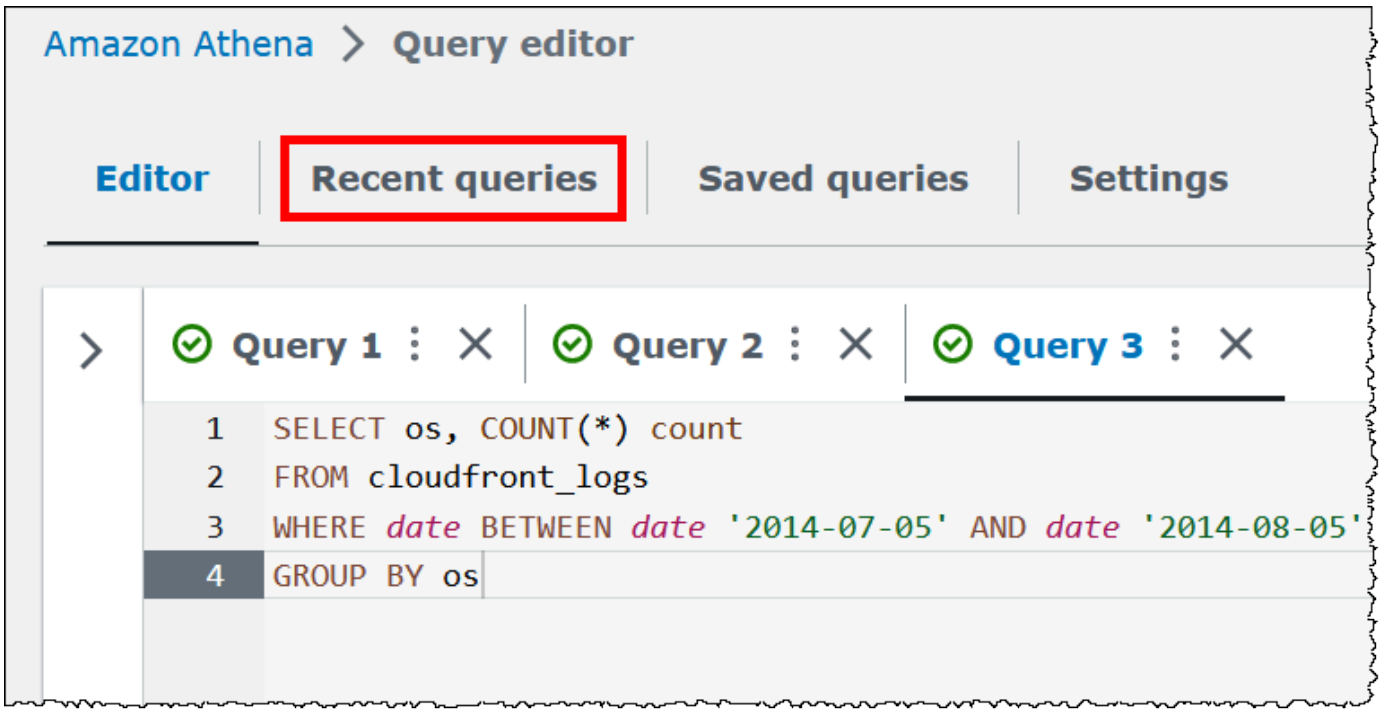
2. Pilih Jalankan.

   Hasilnya terlihat seperti berikut:

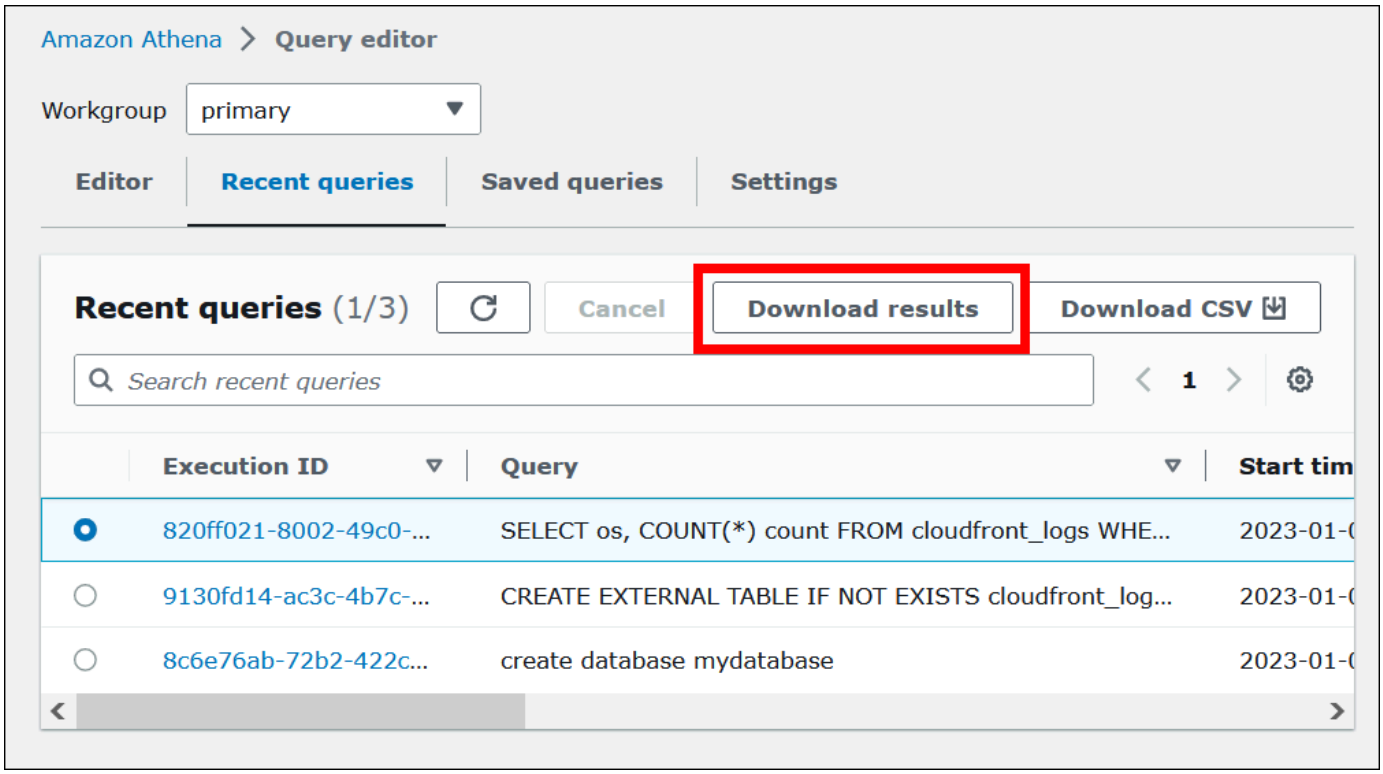


3. Untuk menyimpan hasil kueri ke `.csv` file, pilih Unduh hasil.

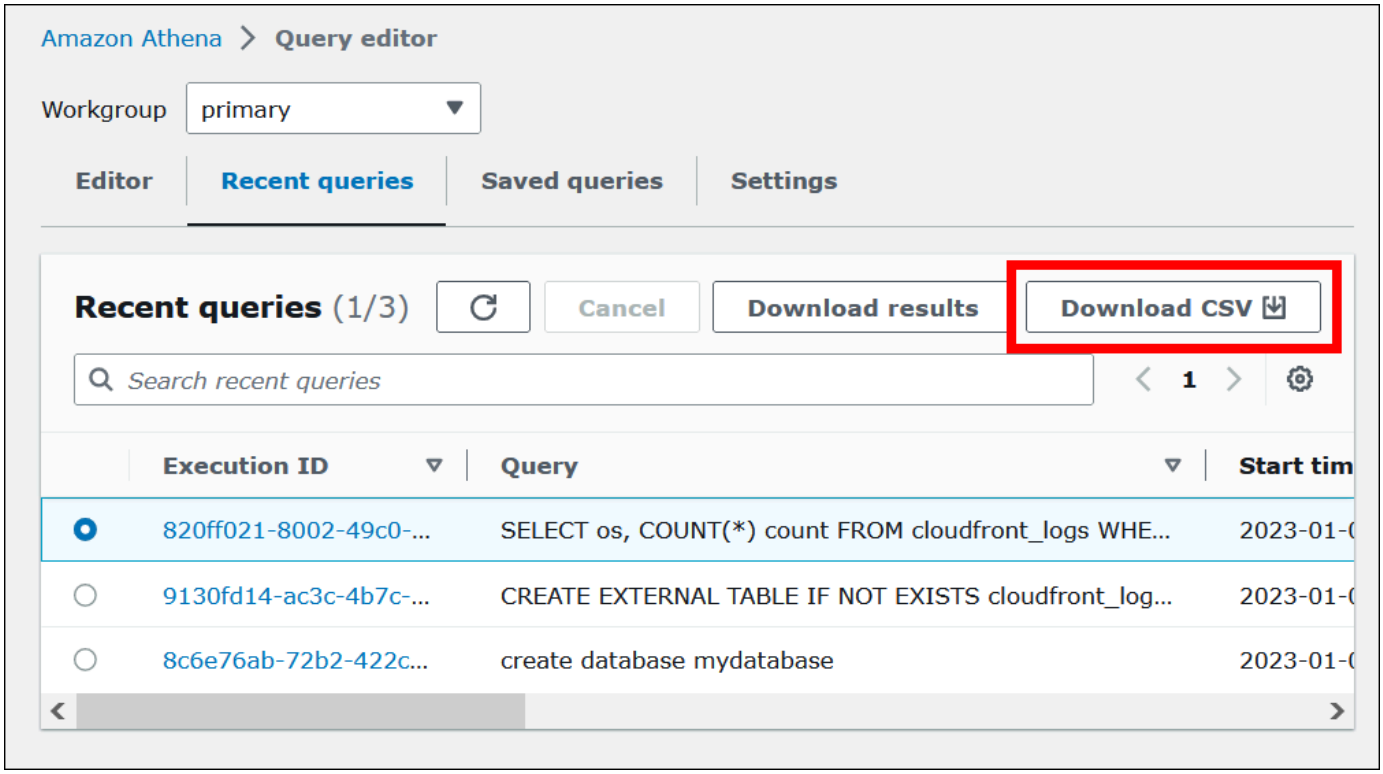
4. Untuk melihat atau menjalankan kueri sebelumnya, pilih tab Kueri terbaru.



5. Untuk mengunduh hasil kueri sebelumnya dari tab Kueri terbaru, pilih kueri, lalu pilih Unduh hasil. Kueri dipertahankan selama 45 hari.



6. Untuk mengunduh satu atau beberapa string kueri SQL terbaru ke file CSV, pilih Unduh CSV.



Untuk informasi selengkapnya, lihat Bekerja dengan hasil kueri, kueri terbaru, dan file keluaran.

Menyimpan kueri Anda

Anda dapat menyimpan kueri yang Anda buat atau edit di editor kueri dengan nama. Athena menyimpan kueri ini di tab Kueri tersimpan. Anda dapat menggunakan tab Kueri tersimpan untuk mengingat, menjalankan, mengganti nama, atau menghapus kueri yang disimpan. Untuk informasi selengkapnya, lihat Menggunakan kueri yang disimpan.

Pintasan keyboard dan saran typeahead

Editor kueri Athena menyediakan banyak pintasan keyboard untuk tindakan seperti menjalankan kueri, memformat kueri, operasi baris, dan menemukan dan mengganti. Untuk informasi selengkapnya dan daftar pintasan lengkap, lihat Meningkatkan produktivitas dengan menggunakan pintasan keyboard di editor kueri Amazon Athena di AWS Blog Big Data.

Editor kueri Athena mendukung saran kode typeahead untuk pengalaman penulisan kueri yang lebih cepat. Untuk membantu Anda menulis kueri SQL dengan akurasi yang ditingkatkan dan peningkatan efisiensi, ia menawarkan fitur-fitur berikut:

- Saat Anda mengetik, saran muncul secara real time untuk kata kunci, variabel lokal, cuplikan, dan item katalog.
- Saat Anda mengetik nama database atau nama tabel diikuti dengan titik, editor dengan mudah menampilkan daftar tabel atau kolom untuk dipilih.
- Saat Anda mengarahkan kursor ke saran cuplikan, sinopsis menunjukkan gambaran singkat tentang sintaks dan penggunaan cuplikan.
- Untuk meningkatkan keterbacaan kode, kata kunci dan aturan penyorotan mereka juga telah diperbarui untuk menyelaraskan dengan sintaks terbaru Trino dan Hive.

Fitur ini diaktifkan secara default. Untuk mengaktifkan atau menonaktifkan fitur, gunakan preferensi editor Kode (ikon roda gigi) di kanan bawah jendela editor kueri.

Menghubungkan ke sumber data lain

Tutorial ini menggunakan sumber data di Amazon S3 dalam format CSV. Untuk informasi tentang menggunakan Athena dengan AWS Glue, lihat. Menggunakan AWS Glue untuk terhubung ke sumber data di Amazon S3 Anda juga dapat menghubungkan Athena ke berbagai sumber data dengan menggunakan driver ODBC dan JDBC, metastore Hive eksternal, dan konektor sumber data Athena. Untuk informasi selengkapnya, lihat Menghubungkan ke sumber data.

Menghubungkan ke sumber data

Anda dapat menggunakan Amazon Athena untuk kueri data yang disimpan di lokasi dan format yang berbeda dalam set data. Set data ini mungkin dalam CSV, JSON, Avro, Parquet, atau beberapa format lainnya.

Tabel dan basis data yang Anda gunakan untuk bekerja di Athena untuk menjalankan kueri didasarkan pada metadata. Metadata adalah data tentang data yang mendasari dalam set data Anda. Bagaimana metadata tersebut menggambarkan set data Anda disebut skema. Sebagai contoh, nama tabel, nama kolom dalam tabel, dan tipe data dari setiap kolom adalah skema, disimpan sebagai metadata, yang menggambarkan set data yang mendasari. Di Athena, kita memanggil sistem untuk mengatur metadata katalog data atau metastore. Kombinasi dari set data dan katalog data yang menggambarkannya disebut sumber data.

Hubungan metadata ke set data yang mendasari tergantung tipe sumber data yang Anda gunakan. Sumber data relasional seperti MySQL, PostgreSQL, dan SQL Server erat mengintegrasikan metadata dengan set data. Dalam sistem ini, metadata paling sering ditulis saat data ditulis. Sumber

data lain, seperti yang dibuat menggunakan Hive, memungkinkan Anda menentukan metadata on-the-fly saat membaca kumpulan data. Set data dapat dalam berbagai format, misalnya, CSV, JSON, Parquet, atau Avro.

Athena secara asli mendukung. AWS Glue Data Catalog AWS Glue Data Catalog Ini adalah katalog data yang dibangun di atas kumpulan data dan sumber data lain seperti Amazon S3, Amazon Redshift, dan Amazon DynamoDB. Anda juga dapat menghubungkan Athena ke sumber data lain dengan menggunakan berbagai konektor.

Topik

Integrasi dengan AWS Glue

AWS Glueadalah ETL yang dikelola sepenuhnya (ekstrak, transformasi, dan muat) Layanan AWS. Salah satu kemampuan utamanya adalah menganalisis dan mengkategorikan data. Anda dapat menggunakan AWS Glue crawler untuk secara otomatis menyimpulkan skema database dan tabel dari data Anda di Amazon S3 dan menyimpan metadata terkait di. AWS Glue Data Catalog

Athena menggunakan metadata tabel AWS Glue Data Catalog untuk menyimpan dan mengambil data Amazon S3 di akun Amazon Web Services Anda. Metadata tabel memungkinkan mesin permintaan Athena tahu bagaimana menemukan, membaca, dan memproses data yang ingin Anda kueri.

Untuk membuat skema database dan tabel di AWS Glue Data Catalog, Anda dapat menjalankan AWS Glue crawler dari dalam Athena pada sumber data, atau Anda dapat menjalankan kueri Data Definition Language (DDL) langsung di Athena Query Editor. Kemudian, menggunakan basis data dan tabel skema yang Anda buat, Anda dapat menggunakan data manipulasi (DDLL) kueri di Athena untuk kueri data.

Anda dapat mendaftarkan akun AWS Glue Data Catalog dari akun selain milik Anda sendiri. Setelah mengonfigurasi izin IAM yang diperlukan AWS Glue, Anda dapat menggunakan Athena untuk

menjalankan kueri lintas akun. Untuk informasi selengkapnya, lihat Akses lintas akun ke katalog AWS Glue data.

Untuk informasi selengkapnya tentang AWS Glue Data Catalog, lihat Katalog Data dan crawler AWS Glue di Panduan AWS Glue Pengembang.

Untuk artikel ilustrasi yang menunjukkan cara menggunakan AWS Glue dan Athena untuk memproses data XHTML, lihat Memproses dan menganalisis file XHTML yang sangat bersarang dan besar menggunakan dan AWS Glue Amazon Athena di Big Data Blog. AWS

Biaya terpisah berlaku untuk AWS Glue. Untuk informasi lebih lanjut, lihat Harga AWS Glue.

Topik

- Menggunakan AWS Glue untuk terhubung ke sumber data di Amazon S3
- Mendaftarkan akun AWS Glue Data Catalog dari akun lain
- Praktik terbaik saat menggunakan Athena dengan AWS Glue
- Menggunakan AWS CLI untuk membuat ulang AWS Glue database dan tabelnya

Menggunakan AWS Glue untuk terhubung ke sumber data di Amazon S3

Athena dapat terhubung ke data Anda yang disimpan di Amazon S3 menggunakan AWS Glue Data Catalog metadata untuk menyimpan seperti nama tabel dan kolom. Setelah koneksi dibuat, basis data, tabel, dan tampilan Anda muncul di editor permintaan Athena.

Untuk menentukan informasi skema AWS Glue untuk digunakan, Anda dapat membuat AWS Glue crawler untuk mengambil informasi secara otomatis, atau Anda dapat menambahkan tabel secara manual dan memasukkan informasi skema.

Membuat AWS Glue crawler

Anda dapat membuat crawler dengan memulai di konsol Athena dan kemudian menggunakan konsol AWS Glue secara terintegrasi. Saat membuat crawler, Anda menentukan lokasi data di Amazon S3 untuk dirayapi.

Untuk membuat crawler di AWS Glue mulai dari konsol Athena

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Di editor kueri, di samping Tabel dan tampilan, pilih Buat, lalu pilih AWS Glue crawler.

3. PadaAWS GlueKonsolTambahkan crawler, ikuti langkah-langkah untuk membuat crawler. Untuk informasi selengkapnya, lihat Menggunakan AWS Glue Crawler dalam panduan ini dan Mengisi AWS Glue Data Catalog di Panduan AWS Glue Pengembang.

> **Note**
>
> Athena tidak mengenali pola pengecualian yang Anda tentukan untuk crawler. AWS Glue Misalnya, jika Anda memiliki bucket Amazon S3 yang berisi keduanya `.csv`dan `.json`file dan Anda mengecualikan `.json`file dari crawler, Athena mengkueri kedua grup file. Untuk menghindari hal ini, menempatkan file yang ingin Anda mengecualikan di lokasi yang berbeda.

Menambahkan tabel menggunakan formulir

Prosedur berikut menunjukkan cara menggunakan konsol Athena untuk menambahkan tabel menggunakan formulir data bucket Create Table From S3.

Untuk menambahkan tabel dan memasukkan informasi skema menggunakan formulir

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Di editor kueri, di samping Tabel dan tampilan, pilih Buat, lalu pilih data bucket S3.
3. Pada formulir data bucket Create Table From S3, untuk nama Table, masukkan nama untuk tabel.
4. Untuk konfigurasi Database, pilih database yang ada, atau buat yang baru.
5. UntukLokasi Set Input Data, tentukan jalur di Amazon S3 ke folder yang berisi kumpulan data yang ingin Anda proses. Jangan sertakan nama file di jalur. Athena memindai semua file di folder yang Anda tentukan. Jika data Anda sudah dipartisi (misalnya,

   s3://DOC-EXAMPLE-BUCKET/logs/year=2004/month=12/day=11/), masukkan jalur dasar saja (misalnya, s3://DOC-EXAMPLE-BUCKET/logs/).
6. Untuk Format Data, pilih di antara opsi berikut:

   - Untuk tipe Table, pilih Apache Hive, Apache Iceberg, atau Delta Lake. Athena menggunakan jenis tabel Apache Hive sebagai default. Untuk informasi tentang menanyakan tabel Apache Iceberg di Athena, lihat. Menggunakan tabel Apache Iceberg Untuk informasi tentang

penggunaan tabel Danau Delta di Athena, lihat. Menanyakan tabel Delta Lake Linux Foundation

- Untuk format File, pilih format file atau log tempat data Anda berada.
  - UntukBerkas Teks dengan Pembatas Kustompilihan, tentukanBidang terminator(yaitu pembatas kolom). Secara opsional, Anda dapat menentukan terminator Collection yang menandai akhir dari tipe array atau terminator Collection yang menandai akhir dari tipe data peta.
- SerDe library — Pustaka SerDe (serializer-deserializer) mem-parsing format data tertentu sehingga Athena dapat membuat tabel untuk itu. Untuk sebagian besar format, SerDe pustaka default dipilih untuk Anda. Untuk format berikut, pilih perpustakaan sesuai dengan kebutuhan Anda:
  - Apache Web Logs - Pilih salah satu RegexSerDeatau GrokSerDeperpustakaan. Untuk RegexSerDe, berikan ekspresi reguler di kotak definisi Regex. Untuk GrokSerDe, berikan serangkaian ekspresi reguler bernama untuk `input.format` SerDe properti. Ekspresi reguler bernama lebih mudah dibaca dan dipelihara daripada ekspresi reguler. Untuk informasi selengkapnya, lihat Menanyakan log Apache yang disimpan di Amazon S3.
  - CSV — Pilih LazySimpleSerDeapakah data yang dipisahkan koma Anda tidak berisi nilai yang diapit tanda kutip ganda atau jika menggunakan format. `java.sql.Timestamp` Pilih SerDeOpenCSV jika data Anda menyertakan tanda kutip atau menggunakan format numerik UNIX `TIMESTAMP` untuk (misalnya,). `1564610311` Untuk informasi selengkapnya, lihat LazySimpleSerDe untuk CSV, TSV, dan file yang dibatasi khusus dan OpenCSV untuk SerDe memproses CSV.
  - JSON - Pilih perpustakaan OpenX atau SerDe Hive JSON. Kedua format mengharapkan setiap dokumen JSON berada pada satu baris teks dan bidang itu tidak dipisahkan oleh karakter baris baru. OpenX SerDe menawarkan beberapa properti tambahan. Untuk informasi selengkapnya tentang properti ini, lihat OpenX JSON SerDe. Untuk informasi tentang Hive SerDe, lihatSarang JSON SerDe.

  Untuk informasi lebih lanjut tentang penggunaan SerDe perpustakaan di Athena, lihat. Format yang didukung SerDes dan data

7. Untuk SerDe properti, tambahkan, edit, atau hapus properti dan nilai sesuai dengan SerDe pustaka yang Anda gunakan dan kebutuhan Anda.

   - Untuk menambahkan SerDe properti, pilih Tambah SerDe properti.
   - Di bidang Nama, masukkan nama properti.

- Di bidang Nilai, masukkan nilai untuk properti.
- Untuk menghapus SerDe properti, pilih Hapus.

8. Untuk properti Tabel, pilih atau edit properti tabel sesuai dengan kebutuhan Anda.

   - Untuk kompresi Tulis, pilih opsi kompresi. Ketersediaan opsi kompresi tulis dan opsi kompresi yang tersedia tergantung pada format data. Untuk informasi selengkapnya, lihat Dukungan kompresi Athena.
   - Untuk Enkripsi, pilih Kumpulan data terenkripsi jika data yang mendasarinya dienkripsi di Amazon S3. Opsi ini menetapkan properti `has_encrypted_data` tabel ke true dalam `CREATE TABLE` pernyataan.

9. Untuk detail Kolom, masukkan nama dan tipe data kolom yang ingin Anda tambahkan ke tabel.

   - Untuk menambahkan lebih banyak kolom satu per satu, pilihTambahkan kolom.
   - Untuk menambahkan lebih banyak kolom dengan cepat, pilihTambahkan kolom. ***Di kotak teks, masukkan daftar kolom yang dipisahkan koma dalam format column_name data_type, column_name data_type [,...], lalu pilih Tambah.***

10. (Opsional) Untuk detail Partisi, tambahkan satu atau beberapa nama kolom dan tipe data. Partisi menyimpan data terkait bersama-sama berdasarkan nilai kolom dan dapat membantu mengurangi jumlah data yang dipindai per kueri. Untuk informasi tentang partisi, lihat Partisi data di Athena.
11. (Opsional) Untuk Bucketing, Anda dapat menentukan satu atau beberapa kolom yang memiliki baris yang ingin Anda kelompokkan bersama, lalu menempatkan baris tersebut ke dalam beberapa ember. Hal ini memungkinkan Anda untuk menanyakan hanya bucket yang ingin Anda baca saat nilai kolom berember ditentukan.

    - Untuk Bucket, pilih satu atau beberapa kolom yang memiliki sejumlah besar nilai unik (misalnya, kunci utama) dan yang sering digunakan untuk memfilter data dalam kueri Anda.
    - Untuk Jumlah ember, masukkan nomor yang memungkinkan file berukuran optimal. Untuk informasi selengkapnya, lihat 10 Tips Penyetelan Kinerja Terbaik untuk Amazon Athena di Blog Big AWS Data.
    - Untuk menentukan kolom bucketed Anda, `CREATE TABLE` pernyataan akan menggunakan sintaks berikut:

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

> **Note**
>
> Opsi Bucketing tidak tersedia untuk jenis tabel Iceberg.

12. Kotak kueri tabel pratinjau menunjukkan `CREATE TABLE` pernyataan yang dihasilkan oleh informasi yang Anda masukkan ke dalam formulir. Pernyataan pratinjau tidak dapat diedit secara langsung. Untuk mengubah pernyataan, ubah bidang formulir di atas pratinjau, atau buat pernyataan langsung di editor kueri alih-alih menggunakan formulir.
13. Pilih Buat tabel untuk menjalankan pernyataan yang dihasilkan di editor kueri dan buat tabel.

Mendaftarkan akun AWS Glue Data Catalog dari akun lain

Anda dapat menggunakan fitur AWS Glue katalog lintas akun Athena untuk mendaftarkan AWS Glue katalog dari akun selain milik Anda. Setelah Anda mengonfigurasi izin IAM yang diperlukan untuk AWS Glue dan daftarkan katalog sebagai Athena`DataCatalog`Sumber daya, Anda dapat menggunakan Athena untuk menjalankan kueri lintas akun. Untuk informasi tentang izin IAM yang diperlukan, lihat Akses lintas akun ke katalog AWS Glue data.

Prosedur berikut menunjukkan cara menggunakan konsol Athena untuk mengkonfigurasi AWS Glue Data Catalog di akun Amazon Web Services selain milik Anda sebagai sumber data.

Untuk mendaftarkan akun AWS Glue Data Catalog dari akun lain

1. Ikuti langkah-langkah diAkses lintas akun ke katalog AWS Glue dataUntuk memastikan bahwa Anda memiliki izin untuk meminta katalog data di akun lain.
2. Buka konsol Athena di https://console.aws.amazon.com/athena/.
3. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.

4. Pilih Sumber data.
5. Di kanan atas, pilih Buat sumber data.
6. Pada halaman Pilih sumber data, untuk Sumber data, pilih S3 - AWS Glue Data Catalog, lalu pilih Berikutnya.
7. Pada halaman Masukkan detail sumber data, di AWS Glue Data Catalogbagian, untuk Pilih AWS Glue Data Catalog, pilih AWS Glue Data Catalog di akun lain.
8. Untuk detail sumber data, masukkan informasi berikut:
   - Nama sumber data — Masukkan nama yang ingin Anda gunakan dalam kueri SQL Anda untuk merujuk ke katalog data di akun lain.
   - Deskripsi— (Opsional) Masukkan deskripsi katalog data di akun lainnya.
   - ID Katalog— Masukkan 12 digit Amazon Web Services akun ID dari akun tempat katalog data berada. Amazon Web Services akun ID adalah ID katalog.
9. (Opsional) Untuk Tag, masukkan pasangan nilai kunci yang ingin Anda kaitkan dengan sumber data. Untuk informasi selengkapnya tentang tag, lihat Menandai sumber daya Athena.
10. Pilih Berikutnya.
11. Pada halaman Tinjau dan buat, tinjau informasi yang Anda berikan, lalu pilih Buat sumber data. Halaman detail sumber data mencantumkan database dan tag untuk katalog data yang Anda daftarkan.
12. Pilih Sumber data. Katalog data yang Anda daftarkan tercantum di kolom Nama sumber data.
13. Untuk melihat atau mengedit informasi tentang katalog data, pilih katalog, lalu pilih Tindakan, Edit.

14. Untuk menghapus katalog data baru, pilih katalog, lalu pilih Tindakan, Hapus.

Untuk informasi selengkapnya, lihat Kueri lintas akun AWS Glue Data Catalog menggunakan Amazon Athena di AWS Blog Big Data.

Praktik terbaik saat menggunakan Athena dengan AWS Glue

Saat menggunakan Athena dengan AWS Glue Data Catalog, Anda dapat menggunakan AWS Glue untuk membuat database dan tabel (skema) untuk ditanyakan di Athena, atau Anda dapat menggunakan Athena untuk membuat skema dan kemudian menggunakannya dalam dan layanan terkait. AWS Glue Topik ini memberikan pertimbangan dan praktik terbaik saat menggunakan salah satu metode.

Di bawah tenda, Athena menggunakan Trino untuk memproses pernyataan DHTML dan Hive untuk memproses pernyataan DDL yang membuat dan memodifikasi skema. Dengan teknologi ini, ada beberapa konvensi yang harus diikuti sehingga Athena AWS Glue dan bekerja sama dengan baik.

Dalam topik ini

Nama database, tabel, dan kolom

Saat Anda membuat skema AWS Glue untuk kueri di Athena, pertimbangkan hal berikut:

- Karakter yang dapat diterima untuk nama database, nama tabel, dan nama kolom AWS Glue harus berupa string UTF-8 dan harus dalam huruf kecil. Perhatikan bahwa Athena secara otomatis menurunkan nama huruf besar dalam kueri DDL saat membuat database, tabel, atau kolom. Panjang string tidak boleh kurang dari 1 atau lebih dari 255 byte. Karakter yang dapat digunakan termasuk spasi.
- Saat ini, dimungkinkan untuk memiliki spasi terkemuka di awal nama. Karena ruang terdepan ini sulit dideteksi dan dapat menyebabkan masalah kegunaan setelah pembuatan, hindari secara tidak sengaja membuat nama objek dengan spasi terkemuka.
- Jika Anda menggunakan AWS::Glue::Database AWS CloudFormation template untuk membuat AWS Glue database dan tidak menentukan nama database, AWS Glue secara otomatis menghasilkan nama database dalam format *`resource_name-random_string`* yang tidak kompatibel dengan Athena.
- Anda dapat menggunakan Pengelola AWS Glue Katalog untuk mengganti nama kolom, tetapi bukan nama tabel atau nama database. Untuk mengatasi batasan ini, Anda harus menggunakan definisi database lama untuk membuat database dengan nama baru. Kemudian Anda menggunakan definisi tabel dari database lama untuk membuat ulang tabel dalam database baru. Untuk melakukan ini, Anda dapat menggunakan AWS CLI atau AWS Glue SDK. Untuk langkah, lihat Menggunakan AWS CLI untuk membuat ulang AWS Glue database dan tabelnya.

Untuk informasi selengkapnya tentang database dan tabel AWS Glue, lihat Database dan Tabel di Panduan AWS Glue Pengembang.

Menggunakan AWS Glue crawler

AWS Glue crawler membantu menemukan skema untuk kumpulan data dan mendaftarkannya sebagai tabel di Katalog Data. AWS Glue Crawler menelusuri data Anda dan menentukan skema. Selain itu, crawler dapat mendeteksi dan mendaftarkan partisi. Untuk informasi selengkapnya, lihat Mendefinisikan crawler di Panduan AWS Glue Pengembang. Tabel dari data yang berhasil dirayapi dapat ditanyakan dari Athena.

> **Note**
>
> Athena tidak mengenali pola pengecualian yang Anda tentukan untuk crawler. AWS Glue Misalnya, jika Anda memiliki bucket Amazon S3 yang berisi keduanya `.csv` dan `.json` file

dan Anda mengecualikan `.json`file dari crawler, Athena mengkueri kedua grup file. Untuk menghindari hal ini, menempatkan file yang ingin Anda mengecualikan di lokasi yang berbeda.

Menjadwalkan crawler agar tetap sinkron AWS Glue Data Catalog dan Amazon S3

AWS Glue crawler dapat diatur untuk berjalan sesuai jadwal atau sesuai permintaan. Untuk informasi selengkapnya, lihat Jadwal berbasis waktu untuk pekerjaan dan crawler di Panduan Pengembang.AWS Glue

Jika Anda memiliki data yang tiba untuk tabel yang dipartisi pada waktu yang tetap, Anda dapat mengatur AWS Glue crawler agar berjalan sesuai jadwal untuk mendeteksi dan memperbarui partisi tabel. Ini dapat menghilangkan kebutuhan untuk menjalankan berpotensi panjang dan mahal`MSCK REPAIR`perintah atau secara manual menjalankan`ALTER TABLE ADD PARTITION`Perintah. Untuk informasi selengkapnya, lihat Partisi tabel di Panduan AWS Glue Pengembang.

Menggunakan beberapa sumber data dengan crawler

Ketika AWS Glue crawler memindai Amazon S3 dan mendeteksi beberapa direktori, ia menggunakan heuristik untuk menentukan di mana root untuk tabel berada dalam struktur direktori, dan direktori mana yang merupakan partisi untuk tabel. Dalam beberapa kasus, tempat skema yang terdeteksi dalam dua atau lebih direktori serupa, crawler dapat memperlakukannya sebagai partisi dan bukan tabel terpisah. Salah satu cara untuk membantu crawler menemukan tabel individu adalah dengan menambahkan direktori root setiap tabel sebagai penyimpanan data untuk crawler.

Partisi berikut di Amazon S3 adalah contoh:

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

Jika skema untuk`table1`dan`table2`serupa, dan sumber data tunggal diatur ke`s3://DOC-EXAMPLE-BUCKET/folder1/`masuk AWS Glue, crawler dapat membuat tabel tunggal dengan dua kolom partisi: satu kolom partisi yang berisi`table1`dan`table2`, dan kolom partisi kedua yang berisi`partition1`melalui`partition5`.

Agar AWS Glue crawler membuat dua tabel terpisah, atur crawler untuk memiliki dua sumber data, `s3://DOC-EXAMPLE-BUCKET/folder1/table1/` dan`s3://DOC-EXAMPLE-BUCKET/folder1/table2`, seperti yang ditunjukkan dalam prosedur berikut.

Untuk menambahkan penyimpanan data S3 ke crawler yang ada di AWS Glue

1. Masuk ke AWS Management Console dan buka AWS Glue konsol di https://console.aws.amazon.com/glue/.
2. Di panel navigasi, pilih Perayap.
3. Pilih tautan ke crawler Anda, lalu pilih Edit.
4. Untuk Langkah 2: Pilih sumber data dan pengklasifikasi, pilih Edit.
5. Untuk Sumber data, pilih Tambahkan sumber data.
6. Dalam kotak dialog Tambahkan sumber data, untuk jalur S3, pilih Jelajahi.
7. Pilih bucket yang ingin Anda gunakan, lalu pilih Pilih.

   Sumber data yang Anda tambahkan muncul di daftar Sumber data.
8. Pilih Selanjutnya.
9. Pada halaman Konfigurasi pengaturan keamanan, buat atau pilih peran IAM untuk crawler, lalu pilih Berikutnya.
10. Pastikan bahwa jalur S3 berakhir dengan garis miring, lalu pilih Tambahkan sumber data S3.
11. Pada halaman Set output dan penjadwalan, untuk konfigurasi Output, pilih database target.
12. Pilih Selanjutnya.
13. Pada halaman Tinjau dan perbarui, tinjau pilihan yang Anda buat. Untuk mengedit langkah, pilih Edit.
14. Pilih Perbarui.

Menyinkronkan skema partisi untuk menghindari “HIVE_PARTITION_SCHEMA_MISMATCH”

Untuk setiap tabel dalam Katalog AWS Glue Data yang memiliki kolom partisi, skema disimpan pada tingkat tabel dan untuk setiap partisi individu dalam tabel. Skema untuk partisi diisi oleh AWS Glue crawler berdasarkan sampel data yang dibaca dalam partisi. Untuk informasi selengkapnya, lihat Menggunakan beberapa sumber data dengan crawler.

Saat Athena menjalankan kueri, itu memvalidasi skema tabel dan skema dari setiap partisi yang diperlukan untuk kueri. validasi membandingkan jenis data kolom dalam rangka dan memastikan bahwa mereka cocok untuk kolom yang tumpang tindih. Ini mencegah operasi tak terduga

seperti menambahkan atau menghapus kolom dari tengah tabel. Jika Athena mendeteksi bahwa skema partisi berbeda dari skema tabel, Athena mungkin tidak dapat memproses kueri dan gagal dengan`HIVE_PARTITION_SCHEMA_MISMATCH`.

Ada beberapa cara untuk memperbaiki masalah ini. Pertama, jika data tidak sengaja ditambahkan, Anda dapat menghapus file data yang menyebabkan perbedaan dalam skema, menjatuhkan partisi, dan re-crawl data. Kedua, Anda dapat drop partisi individu dan kemudian jalankan`MSCK REPAIR`dalam Athena untuk menciptakan kembali partisi menggunakan skema tabel ini. Opsi kedua ini bekerja hanya jika Anda yakin bahwa skema diterapkan akan terus membaca data dengan benar.

Memperbarui metadata tabel

Setelah crawl, AWS Glue crawler secara otomatis menetapkan metadata tabel tertentu untuk membantu membuatnya kompatibel dengan teknologi eksternal lainnya seperti Apache Hive, Presto, dan Spark. Kadang-kadang, crawler mungkin salah menetapkan properti metadata. Perbaiki properti secara manual AWS Glue sebelum menanyakan tabel menggunakan Athena. Untuk informasi selengkapnya, lihat Melihat dan mengedit detail tabel di Panduan AWS Glue Pengembang.

AWS Glue mungkin salah menetapkan metadata ketika file CSV memiliki tanda kutip di sekitar setiap bidang data, membuat properti salah. `serializationLib` Untuk informasi selengkapnya, lihat Data CSV terlampir dalam tanda kutip.

Bekerja dengan file CSV

File CSV terkadang memiliki tanda kutip di sekitar nilai data yang dimaksudkan untuk setiap kolom, dan mungkin ada nilai header yang disertakan dalam file CSV, yang bukan merupakan bagian dari data yang akan dianalisis. Bila Anda gunakan AWS Glue untuk membuat skema dari file-file ini, ikuti panduan di bagian ini.

Data CSV terlampir dalam tanda kutip

Anda mungkin memiliki file CSV yang memiliki bidang data tertutup dalam tanda kutip ganda seperti contoh berikut:

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

Untuk menjalankan kueri di Athena pada tabel yang dibuat dari file CSV yang memiliki nilai kutipan, Anda harus memodifikasi properti tabel untuk AWS Glue menggunakan OpenCSV. SerDe Untuk informasi selengkapnya tentang SerDe OpenCSV, lihat. OpenCSV untuk SerDe memproses CSV

Untuk mengedit properti tabel di AWS Glue konsol

1. Di panel navigasi AWS Glue konsol, pilih Tabel.
2. Pilih tautan untuk tabel yang ingin Anda edit, lalu pilih Tindakan, Edit tabel.
3. Pada halaman Edit tabel, buat perubahan berikut:
   - Untuk Serialisasi lib, masukkan. `org.apache.hadoop.hive.serde2.OpenCSVSerde`
   - UntukParameter serde, masukkan nilai berikut untuk kunci`escapeChar`,`quoteChar`, dan`separatorChar`:
     - Untuk`escapeChar`, masukkan garis miring terbalik (****).
     - Untuk`quoteChar`, masukkan kutipan ganda (**"**).
     - Untuk`separatorChar`, masukkan koma (**,**).
4. Pilih Simpan.

Untuk informasi selengkapnya, lihat Melihat dan mengedit detail tabel di Panduan AWS Glue Pengembang.

Memperbarui properti AWS Glue tabel secara terprogram

Anda dapat menggunakan operasi AWS Glue UpdateTableAPI atau perintah CLI update-table untuk memodifikasi `SerDeInfo` blok dalam definisi tabel, seperti pada contoh JSON berikut.

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

File CSV dengan header

Jika Anda menentukan tabel di Athena dengan`CREATE TABLE`pernyataan, Anda dapat menggunakan`skip.header.line.count`untuk mengabaikan header dalam data CSV Anda, seperti dalam contoh berikut.

```
...
```

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

Atau, Anda dapat menghapus header CSV terlebih dahulu sehingga informasi header tidak termasuk dalam hasil permintaan Athena. Salah satu cara untuk mencapai ini adalah dengan menggunakan AWS Glue pekerjaan, yang melakukan pekerjaan ekstrak, transformasi, dan beban (ETL). Anda dapat menulis skrip dalam AWS Glue menggunakan bahasa yang merupakan perpanjangan dari dialek PySpark Python. Untuk informasi selengkapnya, lihat Menulis Pekerjaan di AWS Glue di Panduan AWS Glue Pengembang.

Contoh berikut menunjukkan fungsi dalam AWS Glue skrip yang menulis frame dinamis menggunakan`from_options`, dan menetapkan opsi `writeHeader` format ke false, yang menghapus informasi header:

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

AWS Glue pengindeksan partisi dan penyaringan

Ketika Athena menanyakan tabel yang dipartisi, Athena mengambil dan memfilter partisi tabel yang tersedia ke subset yang relevan dengan kueri Anda. Saat data dan partisi baru ditambahkan, lebih banyak waktu diperlukan untuk memproses partisi, dan runtime kueri dapat meningkat. Jika Anda memiliki tabel dengan sejumlah besar partisi yang tumbuh seiring waktu, pertimbangkan untuk menggunakan pengindeksan dan penyaringan AWS Glue partisi. Pengindeksan partisi memungkinkan Athena untuk mengoptimalkan pemrosesan partisi dan meningkatkan kinerja kueri pada tabel yang sangat dipartisi. Menyiapkan pemfilteran partisi dalam properti tabel adalah proses dua langkah:

1. Membuat indeks partisi di AWS Glue.
2. Mengaktifkan pemfilteran partisi untuk tabel.

Membuat indeks partisi

Untuk langkah-langkah membuat indeks partisi AWS Glue, lihat Bekerja dengan indeks partisi di Panduan AWS Glue Pengembang. Untuk batasan indeks partisi di AWS Glue, lihat bagian Tentang indeks partisi di halaman itu.

Mengaktifkan pemfilteran partisi

Untuk mengaktifkan pemfilteran partisi untuk tabel, Anda harus mengatur properti tabel baru di AWS Glue. Untuk langkah-langkah tentang cara mengatur properti tabel AWS Glue, lihat halaman Menyiapkan proyeksi partisi. Saat Anda mengedit detail tabel AWS Glue, tambahkan pasangan kunci-nilai berikut ke bagian Properti tabel:

- Untuk Key, tambahkan `partition_filtering.enabled`
- Untuk Nilai, tambahkan `true`

Anda dapat menonaktifkan pemfilteran partisi pada tabel ini kapan saja dengan menyetel `partition_filtering.enabled` nilainya. `false`

Setelah Anda menyelesaikan langkah-langkah di atas, Anda dapat kembali ke konsol Athena untuk menanyakan data.

Untuk informasi selengkapnya tentang penggunaan pengindeksan dan pemfilteran partisi, lihat Meningkatkan kinerja kueri Amazon Athena AWS Glue Data Catalog menggunakan indeks partisi di AWS Blog Big Data.

Bekerja dengan data geospasial

AWS Glue tidak secara native mendukung Teks Terkenal (WKT), Biner Terkenal (WKB), atau tipe data PostGIS lainnya. AWS Glue Pengklasifikasi mem-parsing data geospasial dan mengklasifikasikannya menggunakan tipe data yang didukung untuk format, seperti untuk CSV. `varchar` Seperti AWS Glue tabel lainnya, Anda mungkin perlu memperbarui properti tabel yang dibuat dari data geospasial untuk memungkinkan Athena mengurai tipe data ini apa adanya. Untuk informasi selengkapnya, lihat Menggunakan AWS Glue crawler dan Bekerja dengan file CSV. Athena mungkin tidak dapat mengurai beberapa tipe data geospasial dalam tabel apa adanya. AWS Glue Untuk informasi selengkapnya tentang bekerja dengan data geospasial di Athena, lihatMenanyakan data geospasial.

Menggunakan AWS Glue pekerjaan untuk ETL dengan Athena

AWS Glue pekerjaan melakukan operasi ETL. AWS Glue Pekerjaan menjalankan skrip yang mengekstrak data dari sumber, mengubah data, dan memuatnya menjadi target. Untuk informasi selengkapnya, lihat Menulis Pekerjaan di AWS Glue di Panduan AWS Glue Pengembang.

Membuat tabel menggunakan Athena untuk AWS Glue pekerjaan ETL

Tabel yang Anda buat di Athena harus memiliki properti tabel ditambahkan ke mereka yang disebut `classification`, yang mengidentifikasi format data. Hal ini memungkinkan AWS Glue untuk menggunakan tabel untuk pekerjaan ETL. Nilai klasifikasi dapat berupa`avro`,`csv`,`json`,`orc`,`parquet`, atau`xml`. Contoh pernyataan `CREATE TABLE` di Athena berikut:

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

Jika properti tabel tidak ditambahkan saat tabel dibuat, Anda dapat menambahkannya menggunakan AWS Glue konsol.

Untuk menambahkan properti tabel klasifikasi menggunakan AWS Glue konsol

1. Masuk ke AWS Management Console dan buka AWS Glue konsol di https://console.aws.amazon.com/glue/.
2. Di panel navigasi konsol, pilih Tabel.
3. Pilih tautan untuk tabel yang ingin Anda edit, lalu pilih Tindakan, Edit tabel.
4. Gulir ke bawah ke bagian Properti tabel.
5. Pilih Tambahkan.
6. Untuk Kunci, masukkan **`classification`**.
7. Untuk Nilai, masukkan tipe data (misalnya,**`json`**).
8. Pilih Simpan.

   Di bagian Rincian tabel, tipe data yang Anda masukkan muncul di bidang Klasifikasi untuk tabel.

Untuk informasi selengkapnya, lihat Bekerja dengan tabel di Panduan AWS Glue Pengembang.

Menggunakan pekerjaan ETL untuk mengoptimalkan kinerja kueri

AWS Glue jobs dapat membantu Anda mengubah data ke format yang mengoptimalkan kinerja kueri di Athena. Format data memiliki dampak besar pada performa kueri dan biaya permintaan di Athena.

Kami merekomendasikan untuk menggunakan format data Parket dan ORC. AWS Glue mendukung penulisan ke kedua format data ini, yang dapat membuatnya lebih mudah dan lebih cepat bagi Anda untuk mengubah data ke format optimal untuk Athena. Untuk informasi selengkapnya tentang format ini dan cara lain untuk meningkatkan kinerja, lihat 10 kiat penyetelan kinerja teratas untuk Amazon Athena.

Mengonversi tipe data SMALLINT dan TINYINT ke INT saat mengonversi ke ORC

Untuk mengurangi kemungkinan bahwa Athena tidak dapat membaca `SMALLINT` dan tipe `TINYINT` data yang dihasilkan oleh pekerjaan AWS Glue ETL, konversi `SMALLINT` dan `TINYINT` ke `INT` saat menggunakan wizard atau menulis skrip untuk pekerjaan ETL.

Mengotomatiskan AWS Glue pekerjaan untuk ETL

Anda dapat mengonfigurasi pekerjaan AWS Glue ETL agar berjalan secara otomatis berdasarkan pemicu. Fitur ini sangat ideal ketika data dari luar AWS didorong ke bucket Amazon S3 dalam format suboptimal untuk kueri di Athena. Untuk informasi selengkapnya, lihat Memicu AWS Glue lowongan di Panduan AWS Glue Pengembang.

Menggunakan AWS CLI untuk membuat ulang AWS Glue database dan tabelnya

Mengganti nama AWS Glue database secara langsung tidak dimungkinkan, tetapi Anda dapat menyalin definisinya, memodifikasi definisi, dan menggunakan definisi untuk membuat ulang database dengan nama yang berbeda. Demikian pula, Anda dapat menyalin definisi tabel di database lama, memodifikasi definisi, dan menggunakan definisi yang dimodifikasi untuk membuat ulang tabel di database baru.

> **Note**
>
> Metode yang disajikan tidak menyalin partisi tabel.

Prosedur berikut untuk Windows mengasumsikan bahwa Anda AWS CLI dikonfigurasi untuk output JSON. Untuk mengubah format output default di AWS CLI, jalankan`aws configure`.

Untuk menyalin AWS Glue Database menggunakan AWS CLI

1. Pada prompt perintah, jalankan AWS CLI perintah berikut untuk mengambil definisi AWS Glue database yang ingin Anda salin.

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

Untuk informasi selengkapnya tentang `get-database` perintah, lihat get-database.

2. Simpan output JSON ke file dengan nama database baru (misalnya,*new_database_name*`.json`) ke desktop Anda.
3. Buka file *new_database_name*`.json` di editor teks.
4. Dalam file JSON, lakukan langkah-langkah berikut:
   a. Hapus `{ "Database":` entri luar dan penjepit penutup yang sesuai `}` di akhir file.
   b. Ubah `Name` entri ke nama database baru.
   c. Hapus `CatalogId` bidang.
5. Simpan file tersebut.
6. Pada prompt perintah, jalankan AWS CLI perintah berikut untuk menggunakan file definisi database yang dimodifikasi untuk membuat database dengan nama baru.

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

Untuk informasi selengkapnya tentang `create-database` perintah, lihat create-database. Untuk informasi tentang memuat AWS CLI parameter dari file, lihat Memuat AWS CLI parameter dari file di Panduan AWS Command Line Interface Pengguna.

7. Untuk memverifikasi bahwa database baru telah dibuat AWS Glue, jalankan perintah berikut:

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

Sekarang Anda siap untuk mendapatkan definisi untuk tabel yang ingin Anda salin ke database baru, memodifikasi definisi, dan menggunakan definisi yang dimodifikasi untuk membuat ulang tabel dalam database baru. Prosedur ini tidak mengubah nama tabel.

Untuk menyalin AWS Glue tabel menggunakan AWS CLI

1. Pada prompt perintah, jalankan AWS CLI perintah berikut.

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

Untuk informasi selengkapnya tentang `get-table` perintah, lihat get-table.

2. Simpan output JSON ke file dengan nama tabel (misalnya, *`table_name`* .json) ke desktop Windows Anda.
3. Buka file di editor teks.
4. Dalam file JSON, hapus `{"Table":` entri luar dan penjepit penutup yang sesuai `}` di akhir file.
5. Dalam file JSON, hapus entri berikut dan nilainya:

   - `DatabaseName`— Entri ini tidak diperlukan karena perintah `create-table` CLI menggunakan parameter. `--database-name`
   - `CreateTime`
   - `UpdateTime`
   - `CreatedBy`
   - `IsRegisteredWithLakeFormation`
   - `CatalogId`
   - `VersionId`
6. Simpan file definisi tabel.
7. Pada prompt perintah, jalankan AWS CLI perintah berikut untuk membuat ulang tabel di database baru:

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

   Untuk informasi selengkapnya tentang `create-table` perintah, lihat create-table.

   Tabel sekarang muncul di database baru di AWS Glue dan dapat ditanyakan dari Athena.
8. Ulangi langkah-langkah untuk menyalin setiap tabel tambahan ke database baru di AWS Glue.

Menggunakan Athena Data Connector untuk Eksternal Hive Metastore

Anda dapat menggunakan konektor data Amazon Athena untuk metastore Hive eksternal untuk kueri set data di Amazon S3 yang menggunakan metastore Apache Hive. Tidak diperlukan migrasi metadata ke metadata. AWS Glue Data Catalog Di konsol manajemen Athena, Anda mengonfigurasi fungsi Lambda untuk berkomunikasi dengan metastore Hive yang ada di VPC pribadi Anda dan kemudian menghubungkannya ke metastore. Sambungan dari Lambda ke metastore Hive Anda

dijamin dengan saluran Amazon VPC pribadi dan tidak menggunakan internet publik. Anda dapat memberikan kode fungsi Lambda Anda sendiri, atau Anda dapat menggunakan implementasi default dari konektor data Athena untuk metastore Hive eksternal.

Topik

Ikhtisar fitur

Dengan konektor data Athena untuk metastore Hive eksternal, Anda dapat melakukan tugas-tugas berikut:

- Gunakan konsol Athena untuk mendaftarkan katalog kustom dan menjalankan kueri menggunakan mereka.
- Mendefinisikan fungsi Lambda untuk metastores Hive eksternal yang berbeda dan bergabung dengan mereka dalam permintaan Athena.
- Gunakan metastor Hive AWS Glue Data Catalog dan eksternal Anda dalam kueri Athena yang sama.
- Menentukan katalog dalam konteks eksekusi kueri sebagai katalog default saat ini. Ini akan menghapus persyaratan untuk nama katalog awalan untuk nama basis data dalam pertanyaan Anda. Alih-alih menggunakan sintaks`catalog.database.table`, Anda dapat menggunakan`database.table`.

- Menggunakan berbagai alat untuk menjalankan kueri yang referensi metastores Hive eksternal. Anda dapat menggunakan konsol Athena, AWS SDK, Athena AWS CLI API, dan driver Athena JDBC dan ODBC yang diperbarui. Driver yang diperbarui memiliki dukungan untuk katalog kustom.

Dukungan API

Athena Data Connector untuk Eksternal Hive Metastore termasuk dukungan untuk operasi API pendaftaran katalog dan operasi metadata API.

- Pendaftaran katalog— Mendaftar katalog kustom untuk metastores Hive eksternal dan Sumber data gabungan.
- Metadata — Gunakan API metadata untuk menyediakan informasi database dan tabel untuk AWS Glue dan katalog apa pun yang Anda daftarkan di Athena.
- Klien SDK Athena JAVA— Gunakan API pendaftaran katalog, API metadata, dan dukungan untuk katalog di `StartQueryExecution` operasi di klien Athena Java SDK diperbarui.

Implementasi referensi

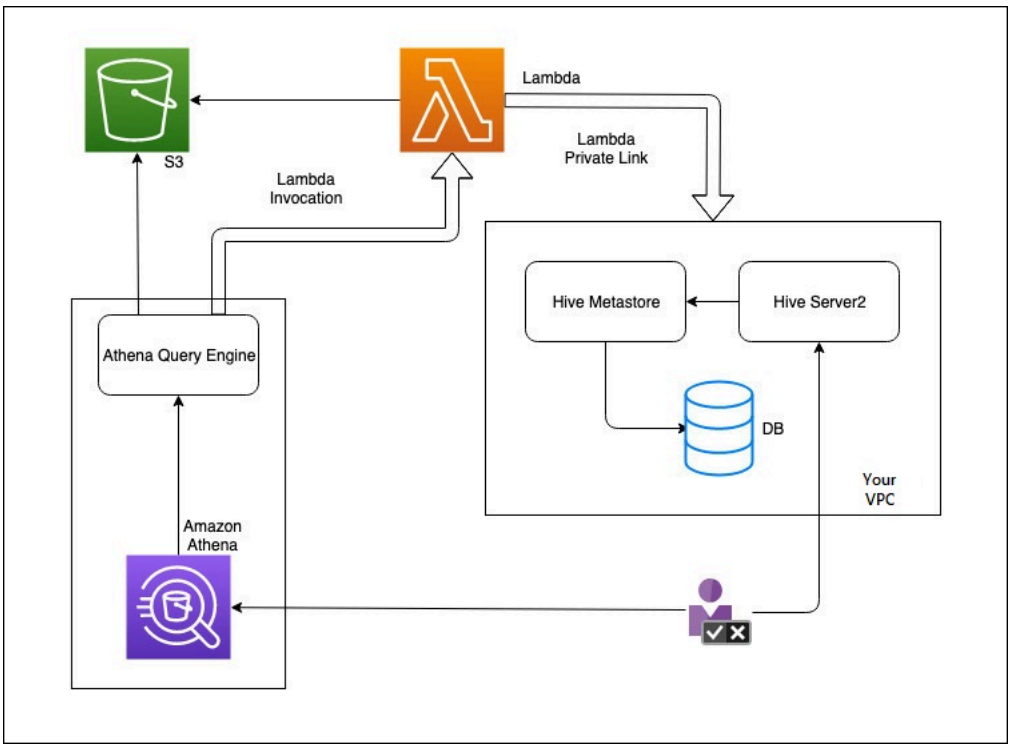
Athena menyediakan implementasi referensi untuk fungsi Lambda yang menghubungkan ke metastores Hive eksternal. Implementasi referensi disediakan GitHub sebagai proyek open source di Athena Hive metastore.

Implementasi referensi tersedia sebagai dua AWS SAM aplikasi berikut dalam AWS Serverless Application Repository (SAR). Anda dapat menggunakan salah satu dari aplikasi ini di SAR untuk membuat fungsi Lambda Anda sendiri.

- **`AthenaHiveMetastoreFunction`**— Fungsi Lambda `.jar` file. Sebuah “uber” JAR (juga dikenal sebagai JAR lemak atau JAR dengan dependensi) adalah `.jar` file yang berisi kedua program Java dan dependensi dalam satu file.
- **`AthenaHiveMetastoreFunctionWithLayer`**— Lapisan lambda dan fungsi Lambda tipis `.jar` file.

Alur kerja

Diagram berikut menunjukkan bagaimana Athena berinteraksi dengan metastore Hive eksternal Anda.



Dalam alur kerja ini, Anda basis data-terhubung Hive metastore adalah di dalam VPC Anda. Anda menggunakan Hive Server2 untuk mengelola metastore Hive Anda menggunakan CLI Hive.

Alur kerja untuk menggunakan metastores Hive eksternal dari Athena mencakup langkah-langkah berikut.

1. Anda membuat fungsi Lambda yang menghubungkan Athena ke metastore Hive yang ada di dalam VPC Anda.
2. Anda mendaftarkan nama katalog unik untuk metastore Hive Anda dan nama fungsi yang sesuai di akun Anda.
3. Saat Anda menjalankan permintaan Athena DDLL atau DDL yang menggunakan nama katalog, mesin permintaan Athena memanggil nama fungsi Lambda yang Anda terkait dengan nama katalog.
4. Menggunakan AWS PrivateLink, fungsi Lambda berkomunikasi dengan metastore Hive eksternal di VPC Anda dan menerima respons terhadap permintaan metadata. Athena menggunakan metadata dari metastore Hive eksternal Anda seperti menggunakan metadata dari default AWS Glue Data Catalog.

Pertimbangan dan batasan

Saat Anda menggunakan Athena Data konektor untuk Eksternal Hive Metastore, pertimbangkan hal-hal berikut:

- Anda dapat menggunakan CTAS untuk membuat tabel pada metastore Hive eksternal.
- Anda dapat menggunakan INSERT INTO untuk menyisipkan data ke metastore Hive eksternal.
- dukungan DDL untuk metastore Hive eksternal terbatas pada pernyataan berikut.
  - MENGUBAH DATABASE SET DBPROPERTIES
  - MENGUBAH TABEL TAMBAHKAN KOLOM
  - ALTER TABLE ADD PARTITION
  - UBAH PARTISI DROP TABEL
  - MENGUBAH NAMA TABEL PARTISI
  - MENGUBAH TABEL GANTI KOLOM
  - MENGUBAH LOKASI SET TABEL
  - MENGUBAH TABEL SET TBLPROPERTIES
  - BUAT BASIS DATA
  - CREATE TABLE
  - BUAT TABEL SEBAGAI
  - MENGGAMBARKAN TABEL
  - DROP DATABASE
  - MEJA DROP
  - TAMPILKAN KOLOM
  - TAMPILKAN TABEL BUAT
  - TAMPILKAN PARTISI
  - TAMPILKAN SKEMA
  - TAMPILKAN TABEL
  - TBLPROPERTIES
- Jumlah maksimum katalog terdaftar yang dapat Anda miliki adalah 1.000.
- Otentikasi Kerberos untuk Hive metastore tidak didukung.

- Untuk menggunakan driver JDBC dengan metastore Hive eksternal atauKueri gabungan, termasuk`MetadataRetrievalMethod=ProxyAPI`dalam string koneksi JDBC Anda. Untuk informasi tentang driver JDBC, lihatMenghubungkan ke Amazon Athena dengan JDBC.
- Kolom tersembunyi Hive`$path`,,`$bucket`, `$file_size` `$file_modified_time`, `$partition`, `$row_id` tidak dapat digunakan untuk pemfilteran kontrol akses berbutir halus.
- Sarang tabel sistem tersembunyi seperti *`example_table`*`$partitions` atau tidak *`example_table`*`$properties` didukung oleh kontrol akses berbutir halus.

Izin

Prebuilt dan konektor data kustom mungkin memerlukan akses ke sumber daya berikut untuk berfungsi dengan benar. Periksa informasi untuk konektor yang Anda gunakan untuk memastikan bahwa Anda telah mengonfigurasi VPC Anda dengan benar. Untuk informasi tentang izin IAM yang diperlukan untuk menjalankan kueri dan membuat konektor sumber data di Athena, lihatIzinkan akses ke Konektor Data Athena untuk Metastore Sarang EksternaldanIzinkan akses fungsi Lambda ke metastores Hive eksternal.

- Amazon S3— Selain menulis hasil kueri ke lokasi hasil kueri Athena di Amazon S3, konektor data juga menulis ke bucket tumpahan di Amazon S3. Konektivitas dan izin ke lokasi Amazon S3 ini diperlukan. Untuk informasi selengkapnya, lihat Lokasi tumpahan di Amazon S3 dalam topik ini.
- Athena— Akses diperlukan untuk memeriksa status kueri dan mencegah overscan.
- AWS Glue— Akses diperlukan jika konektor Anda menggunakan AWS Glue metadata tambahan atau primer.
- AWS Key Management Service
- Kebijakan— Hive metastore, Athena Kueri Federation, dan UDFS memerlukan kebijakan selainAWS kebijakan terkelola: AmazonAthenaFullAccess. Untuk informasi selengkapnya, lihat Manajemen identitas dan akses di Athena.

Lokasi tumpahan di Amazon S3

KarenaBatasanpada ukuran respon fungsi Lambda, tanggapan lebih besar dari tumpahan ambang ke lokasi Amazon S3 yang Anda tentukan saat Anda membuat fungsi Lambda Anda. Athena membaca tanggapan ini dari Amazon S3 langsung.

> **Note**
>
> Athena tidak menghapus file respon pada Amazon S3. Sebaiknya atur kebijakan penyimpanan untuk menghapus file respons secara otomatis.

Menghubungkan Athena ke metastore Apache Hive

Untuk menghubungkan Athena ke metastore Apache Hive, Anda harus membuat dan mengonfigurasi fungsi Lambda. Untuk implementasi dasar, Anda dapat melakukan semua langkah yang diperlukan mulai dari konsol manajemen Athena.

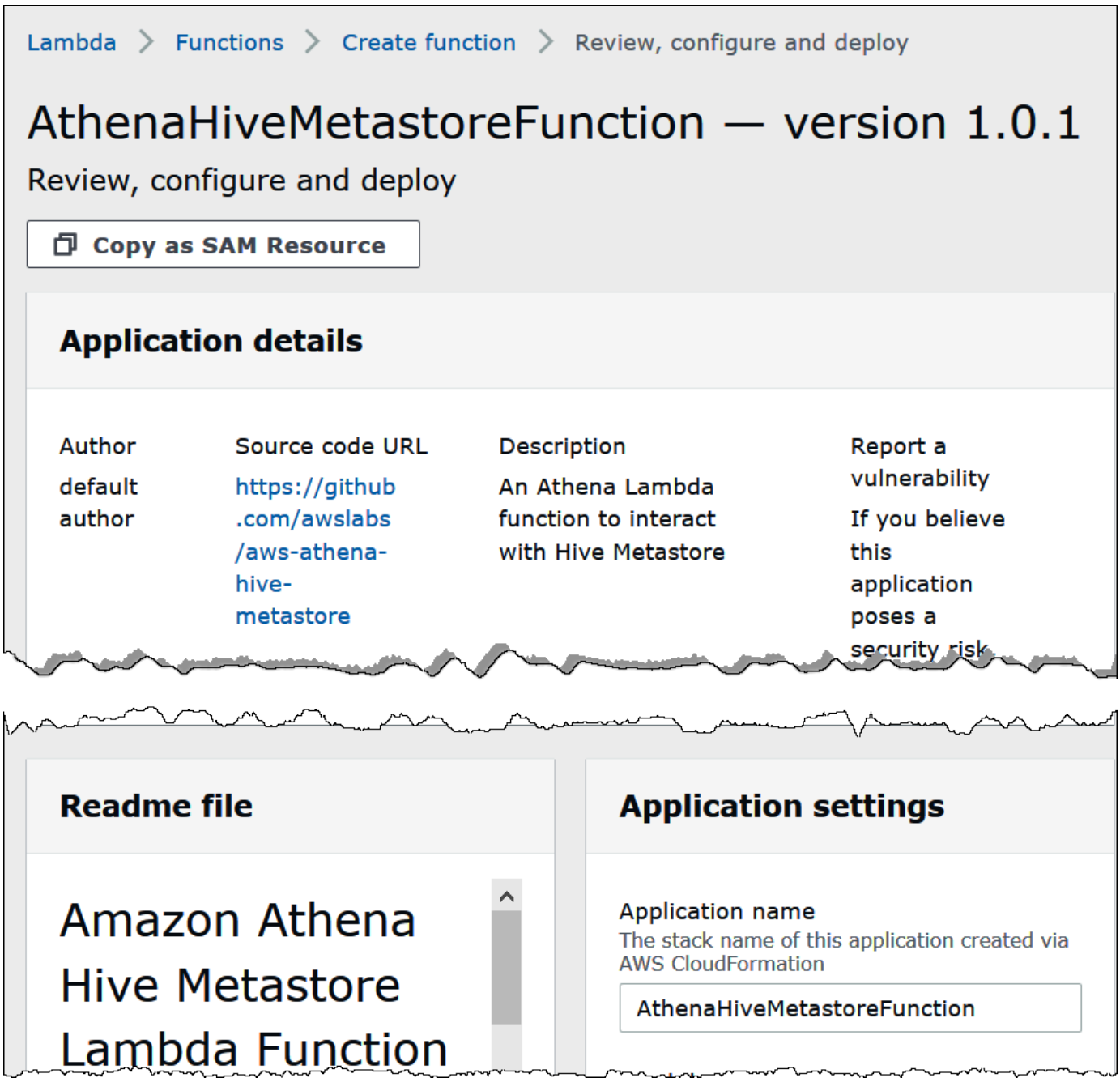
> **Note**
>
> Prosedur berikut mengharuskan Anda memiliki izin untuk membuat IAM role kustom untuk fungsi Lambda. Jika Anda tidak memiliki izin untuk membuat peran kustom, Anda dapat menggunakan implementasi referensi Athena untuk membuat fungsi Lambda secara terpisah, lalu menggunakan AWS Lambda konsol untuk memilih peran IAM yang ada untuk fungsi tersebut. Untuk informasi selengkapnya, lihat Menghubungkan Athena ke metastore Hive menggunakan peran eksekusi IAM yang ada.

Para conectar Athena a metastore a Hive

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.

3. Pilih Sumber data.
4. Di kanan atas konsol, pilih Buat sumber data.
5. Pada halaman Pilih sumber data, untuk Sumber data, pilih metastore S3 - Apache Hive.
6. Pilih Berikutnya.
7. Di bagian Rincian sumber data, untuk nama sumber data, masukkan nama yang ingin Anda gunakan dalam pernyataan SQL saat Anda menanyakan sumber data dari Athena. Nama bisa sampai 127 karakter dan harus unik di akun Anda. Itu tidak dapat diubah setelah Anda membuatnya. Karakter yang valid adalah a-z, A-Z, 0-9, _ (garis bawah), @ (pada tanda) dan - (tanda hubung). Nama`awsdatacatalog`,, `hivejmx`, dan dicadangkan `system` oleh Athena dan tidak dapat digunakan untuk nama sumber data.
8. Untuk fungsi Lambda, pilih Create Lambda function, lalu pilih Create a new Lambda function in AWS Lambda

   AthenaHiveMetastoreFunctionHalaman terbuka di AWS Lambda konsol. Halaman ini mencakup informasi rinci tentang konektor.

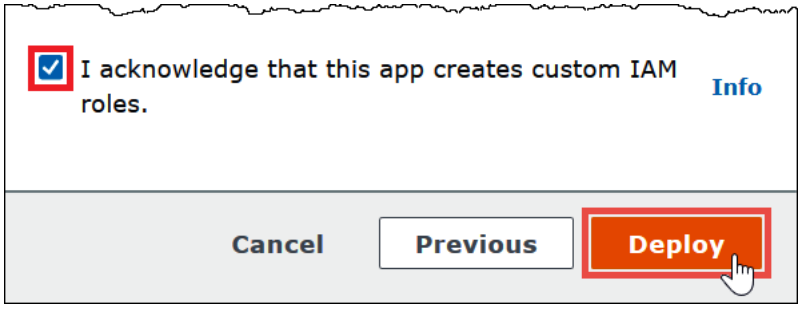


9. Di bawahPengaturan aplikasi, masukkan parameter untuk fungsi Lambda Anda.

   - LambdaFuncName— Berikan nama untuk fungsi tersebut. Misalnya, myHiveMetastore.
   - SpillLocation— Tentukan lokasi Amazon S3 di akun ini untuk menyimpan metadata spillover jika ukuran respons fungsi Lambda melebihi 4 MB.
   - HMSURIS— Masukkan URI host metastore Hive Anda yang menggunakan protokol Thrift di port 9083. Gunakan sintaks`thrift://<host_name>:9083`.

- LambdaMemory— Tentukan nilai dari 128 MB hingga 3008 MB. Fungsi Lambda dialokasikan siklus CPU sebanding dengan jumlah memori yang Anda mengonfigurasi. Defaultnya adalah 1.024.
- LambdaTimeout— Tentukan waktu pemanggilan Lambda maksimum yang diizinkan dalam hitungan detik dari 1 hingga 900 (900 detik adalah 15 menit). Default-nya adalah 300 detik (5 menit).
- VPC SecurityGroupIds — Masukkan daftar ID grup keamanan VPC yang dipisahkan koma untuk metastore Hive.
- VPC SubnetIds — Masukkan daftar ID subnet VPC yang dipisahkan koma untuk metastore Hive.

10. Pilih Saya mengakui bahwa aplikasi ini membuat peran IAM khusus, lalu pilih Terapkan.



Saat deployment selesai, fungsi Anda muncul dalam daftar aplikasi Lambda Anda. Sekarang bahwa fungsi metastore Hive telah dikerahkan ke akun Anda, Anda dapat mengonfigurasi Athena untuk menggunakannya.

11. Kembali ke halaman Masukkan detail sumber data di konsol Athena.
12. Di bagian fungsi Lambda, pilih ikon penyegaran di sebelah kotak pencarian fungsi Lambda. Menyegarkan daftar fungsi yang tersedia menyebabkan fungsi yang baru dibuat muncul dalam daftar.
13. Pilih nama fungsi yang baru saja Anda buat di konsol Lambda. ARN dari fungsi Lambda ditampilkan.
14. (Opsional) Untuk Tag, tambahkan pasangan nilai kunci untuk dikaitkan dengan sumber data ini. Untuk informasi selengkapnya tentang tag, lihat Menandai sumber daya Athena.
15. Pilih Berikutnya.
16. Pada halaman Tinjau dan buat, tinjau detail sumber data, lalu pilih Buat sumber data.
17. Bagian Detail sumber data pada halaman untuk sumber data Anda menunjukkan informasi tentang konektor baru Anda.

Anda sekarang dapat menggunakan nama sumber Data yang Anda tentukan untuk referensi metastore Hive dalam kueri SQL Anda di Athena. Dalam kueri SQL Anda, gunakan sintaks contoh berikut, menggantihms-catalog-1dengan nama katalog yang Anda tentukan sebelumnya.

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

18. Untuk informasi tentang melihat, mengedit, atau menghapus sumber data yang Anda buat, lihatMengelola sumber data.

Menggunakan AWS Serverless Application Repository untuk menyebarkan konektor sumber data Hive

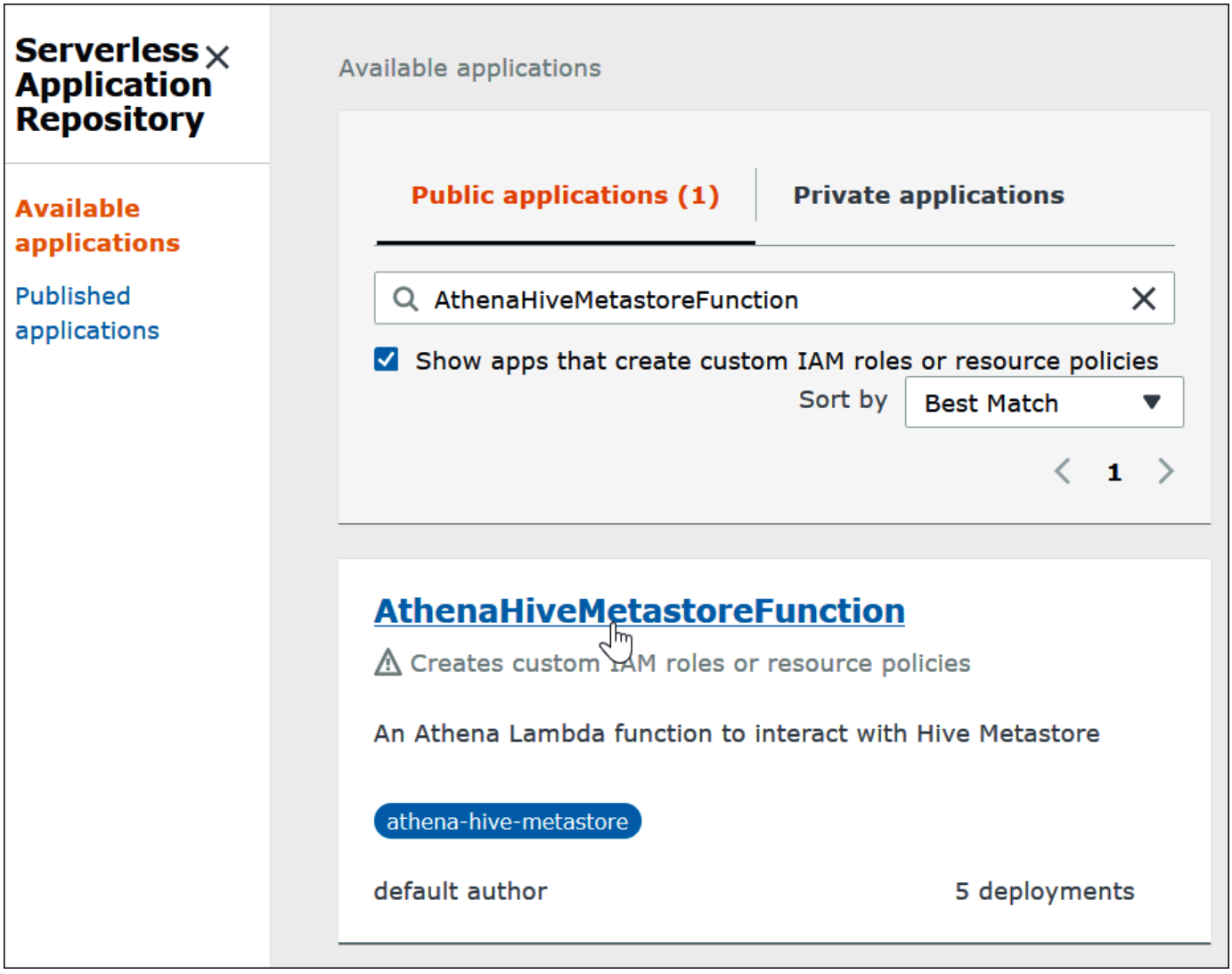
Untuk menggunakan konektor sumber data Athena untuk Hive, Anda dapat menggunakan AWS Serverless Application Repositoryalih-alih memulai dengan konsol Athena. Gunakan AWS Serverless Application Repository untuk menemukan konektor yang ingin Anda gunakan, berikan parameter yang dibutuhkan konektor, dan kemudian gunakan konektor ke akun Anda. Kemudian, setelah Anda menggunakan konektor, Anda menggunakan konsol Athena untuk membuat sumber data tersedia untuk Athena.

Untuk menggunakan AWS Serverless Application Repository untuk menyebarkan konektor sumber data untuk Hive ke akun Anda

1. Masuk ke AWS Management Console dan buka Repositori Aplikasi Tanpa Server.
2. Di panel navigasi, pilihAplikasi yang tersedia.
3. Pilih opsiMenampilkan aplikasi yang membuat IAM role khusus atau kebijakan sumber daya.
4. Dalam kotak pencarian, masukkan **`Hive`**. Konektor yang muncul meliputi dua berikut:

   - AthenaHiveMetastoreFunction— Fungsi Lambda`.jar`file.
   - AthenaHiveMetastoreFunctionWithLayer- Lapisan Lambda dan file fungsi Lambda tipis. `.jar`

   Kedua aplikasi memiliki fungsi yang sama dan hanya berbeda dalam implementasinya. Anda dapat menggunakan salah satu untuk membuat fungsi Lambda yang menghubungkan Athena ke metastore Hive Anda.
5. Pilih nama konektor yang ingin Anda gunakan. Tutorial ini menggunakan AthenaHiveMetastoreFunction.



6. Di bawahPengaturan aplikasi, masukkan parameter untuk fungsi Lambda Anda.

   - LambdaFuncName— Berikan nama untuk fungsi tersebut. Misalnya, myHiveMetastore.
   - SpillLocation— Tentukan lokasi Amazon S3 di akun ini untuk menyimpan metadata spillover jika ukuran respons fungsi Lambda melebihi 4 MB.
   - HMSURIS— Masukkan URI host metastore Hive Anda yang menggunakan protokol Thrift di port 9083. Gunakan sintaks`thrift://<host_name>:9083`.
   - LambdaMemory— Tentukan nilai dari 128 MB hingga 3008 MB. Fungsi Lambda dialokasikan siklus CPU sebanding dengan jumlah memori yang Anda mengonfigurasi. Defaultnya adalah 1.024.
   - LambdaTimeout— Tentukan waktu pemanggilan Lambda maksimum yang diizinkan dalam hitungan detik dari 1 hingga 900 (900 detik adalah 15 menit). Default-nya adalah 300 detik (5 menit).

- VPC SecurityGroupIds — Masukkan daftar ID grup keamanan VPC yang dipisahkan koma untuk metastore Hive.
- VPC SubnetIds — Masukkan daftar ID subnet VPC yang dipisahkan koma untuk metastore Hive.

7. Di bagian kanan bawahDetail aplikasihalaman, pilihSaya mengakui bahwa aplikasi ini menciptakan IAM role kustom, lalu pilihPenerapan.

Pada titik ini, Anda dapat mengonfigurasi Athena untuk menggunakan fungsi Lambda Anda untuk terhubung ke metastore Hive Anda. Untuk langkah, lihat Konfigurasikan Athena untuk menggunakan konektor metastore Hive yang digunakan.

Menghubungkan Athena ke metastore Hive menggunakan peran eksekusi IAM yang ada

Untuk menghubungkan metastore Hive eksternal Anda ke Athena dengan fungsi Lambda yang menggunakan IAM role yang ada, Anda dapat menggunakan implementasi referensi Athena dari konektor Athena untuk metastore Hive eksternal.

Tiga langkah utama adalah sebagai berikut:

1. Kloning dan bangun - Kloning implementasi referensi Athena dan buat file JAR yang berisi kode fungsi Lambda.
2. AWS Lambda konsol — Di AWS Lambda konsol, buat fungsi Lambda, tetapkan peran eksekusi IAM yang ada, dan unggah kode fungsi yang Anda buat.
3. Konsol Amazon Athena — Di konsol Amazon Athena, buat nama sumber data yang dapat Anda gunakan untuk merujuk ke metastore Hive eksternal di kueri Athena Anda.

Jika Anda sudah memiliki izin untuk membuat peran IAM kustom, Anda dapat menggunakan alur kerja sederhana yang menggunakan konsol Athena dan untuk membuat dan AWS Serverless Application Repository mengonfigurasi fungsi Lambda. Untuk informasi selengkapnya, lihat Menghubungkan Athena ke metastore Apache Hive.

Prasyarat

- Git harus diinstal pada sistem Anda.
- Anda harus memilikiApache Mavendiinstal.

- Anda memiliki peran eksekusi IAM yang dapat Anda tetapkan untuk fungsi Lambda. Untuk informasi selengkapnya, lihat Izinkan akses fungsi Lambda ke metastores Hive eksternal.

Kloning dan bangun fungsi Lambda

Kode fungsi untuk implementasi referensi Athena adalah proyek Maven yang terletak di awslabs/. GitHub aws-athena-hive-metastore Untuk informasi rinci tentang proyek, lihat file README yang sesuai pada GitHub atau Implementasi referensi topik dalam dokumentasi ini.

Untuk mengkloning dan membangun kode fungsi Lambda

1. Masukkan perintah berikut untuk mengkloning implementasi referensi Athena:

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

2. Jalankan perintah berikut untuk membangun `.jar`untuk fungsi Lambda:

   ```
   mvn clean install
   ```

   Setelah proyek berhasil dibangun, berikut `.jar`dibuat di folder target proyek Anda:

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

   Di bagian selanjutnya, Anda menggunakan AWS Lambda konsol untuk mengunggah file ini ke akun Amazon Web Services Anda.

Buat dan konfigurasikan fungsi Lambda di konsol AWS Lambda

Di bagian ini, Anda menggunakan AWS Lambda konsol untuk membuat fungsi yang menggunakan peran eksekusi IAM yang ada. Setelah Anda mengonfigurasi VPC untuk fungsi, Anda mengunggah kode fungsi dan mengonfigurasi variabel lingkungan untuk fungsi.

Buat fungsi Lambda

Pada langkah ini, Anda membuat fungsi di AWS Lambda konsol yang menggunakan peran IAM yang ada.

Untuk membuat fungsi Lambda yang menggunakan IAM role yang sudah ada

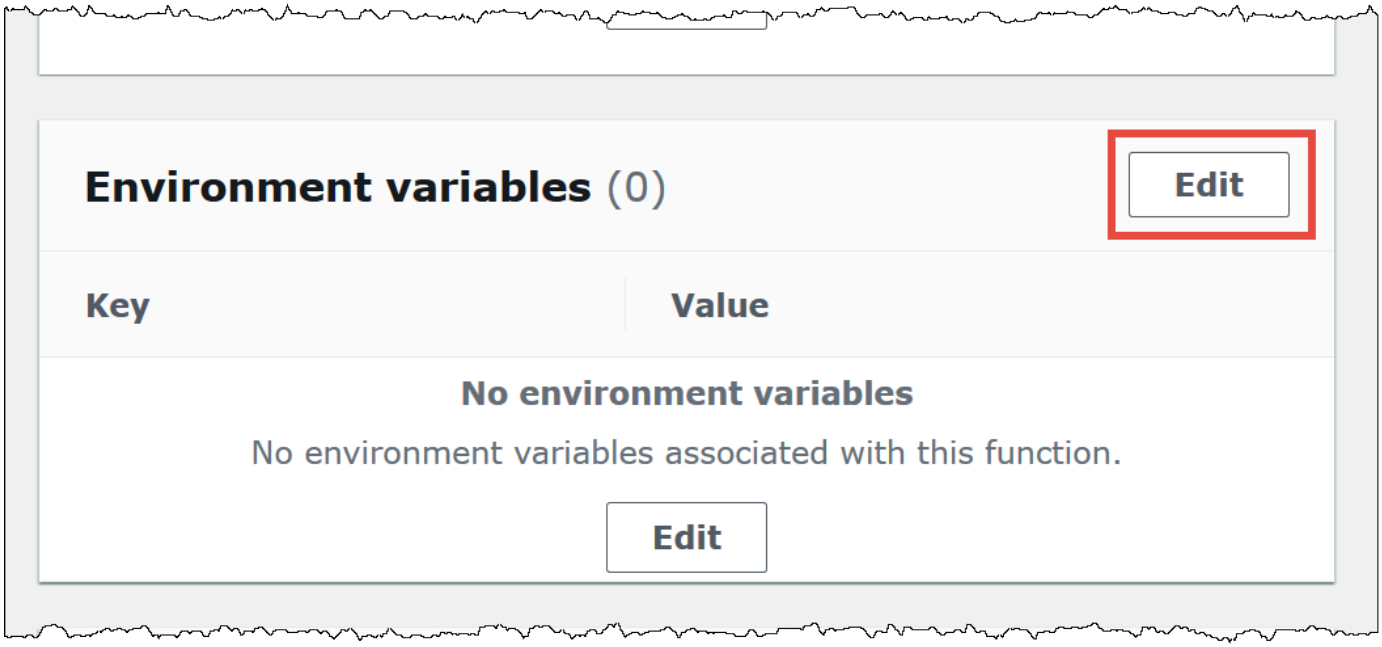
1. Masuk ke AWS Management Console dan buka AWS Lambda konsol di https://console.aws.amazon.com/lambda/.
2. Di panel navigasi, pilih Fungsi.
3. Pilih Buat fungsi.
4. Pilih Penulis dari scratch.
5. UntukNama fungsiMasukkan nama fungsi Lambda Anda (misalnya,**`EHMSBasedLambda`**).
6. UntukWaktu pengoperasianPilihJava 8.
7. (Opsional) Di bagian Izin, luaskan Ubah peran eksekusi default.
8. UntukPeran eksekusiPilihGunakan peran yang sudah ada.
9. UntukPeran yang ada, pilih peran eksekusi IAM yang fungsi Lambda Anda akan gunakan untuk Athena (contoh ini menggunakan peran yang disebut`AthenaLambdaExecutionRole`).
10. Perluas Pengaturan lanjutan.
11. Pilih Aktifkan Jaringan.
12. UntukVPC, pilih VPC yang fungsi Anda akan memiliki akses ke.
13. Untuk Subnet, pilih subnet VPV untuk digunakan oleh Lambda.
14. UntukGrup keamanan, pilih grup keamanan VPC untuk digunakan Lambda.
15. Pilih Buat fungsi. AWS Lambda Konsol dan membuka halaman konfigurasi untuk fungsi Anda dan mulai membuat fungsi Anda.

Unggah kode dan konfigurasikan fungsi Lambda

Saat konsol memberi tahu Anda bahwa fungsi Anda telah berhasil dibuat, Anda siap untuk mengunggah kode fungsi dan mengonfigurasi variabel lingkungannya.

Untuk mengunggah kode fungsi Lambda Anda dan mengonfigurasi variabel lingkungannya

1. Di konsol Lambda, pastikan Anda berada di tab Kode pada halaman fungsi yang Anda tentukan.
2. Untuk sumber Kode, pilih Unggah dari, lalu pilih file.zip atau.jar.
3. Unggah`hms-lambda-func-1.0-SNAPSHOT-withdep.jar`yang Anda buat sebelumnya.
4. Pada halaman fungsi Lambda Anda, pilih tab Konfigurasi.
5. Dari panel di sebelah kiri, pilih variabel Lingkungan.
6. Di bagian Variabel lingkungan, pilih Edit.

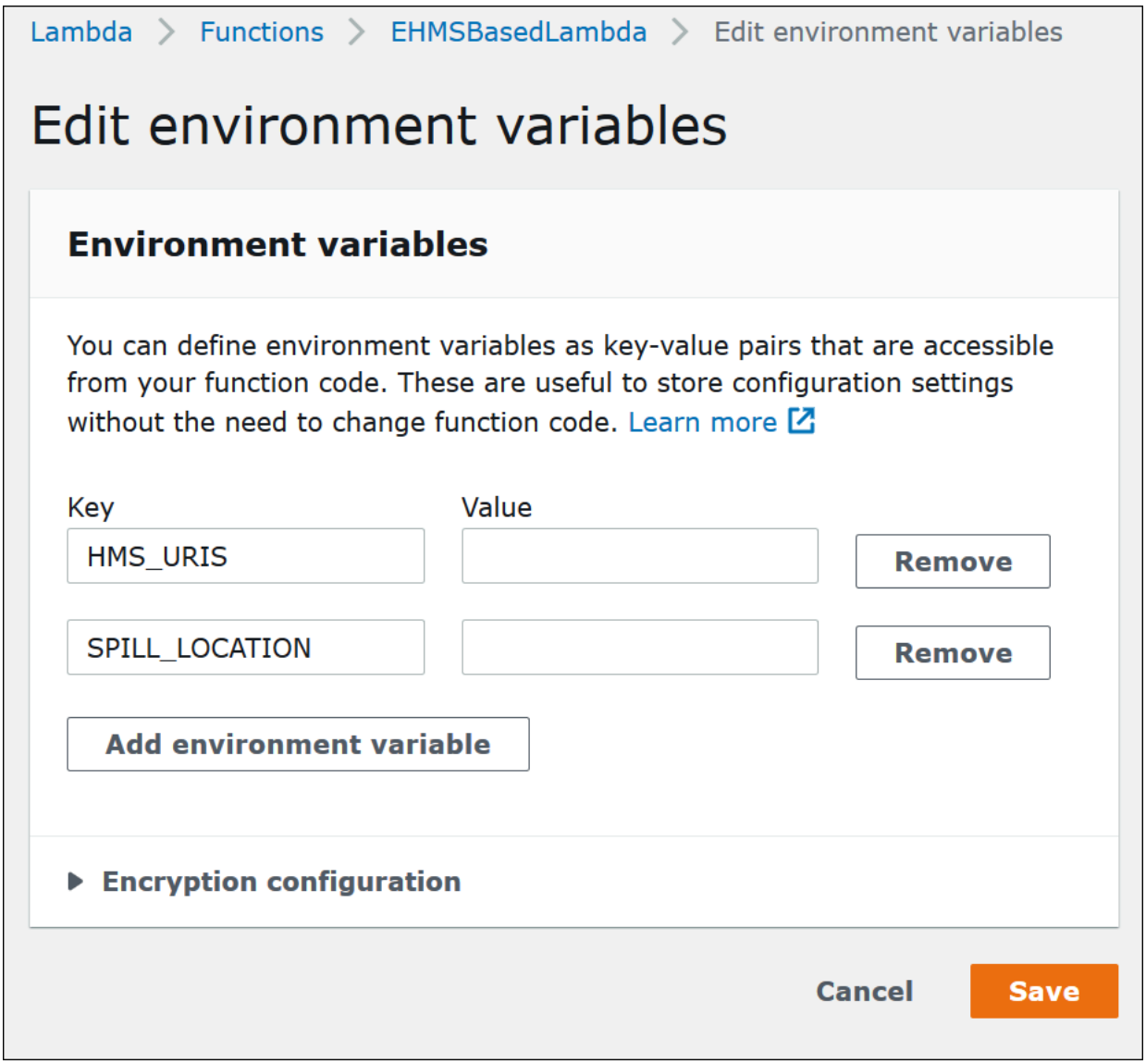


7. Pada halaman Edit variabel lingkungan, gunakan opsi Tambahkan variabel lingkungan untuk menambahkan kunci dan nilai variabel lingkungan berikut:

   - HMS_URIS— Gunakan sintaks berikut untuk memasukkan URI host metastore Hive Anda yang menggunakan protokol Thrift di port 9083.

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

   - SPILL_LOCATION - Tentukan lokasi Amazon S3 di akun Amazon Web Services Anda untuk menyimpan metadata spillover jika ukuran respons fungsi Lambda melebihi 4 MB.



8. Pilih Simpan.

Pada titik ini, Anda siap mengonfigurasi Athena untuk menggunakan fungsi Lambda Anda untuk terhubung ke metastore Hive Anda. Untuk langkah, lihat Konfigurasikan Athena untuk menggunakan konektor metastore Hive yang digunakan.

Konfigurasikan Athena untuk menggunakan konektor metastore Hive yang digunakan

Setelah Anda menerapkan konektor sumber data Lambda `AthenaHiveMetastoreFunction` seperti ke akun Anda, Anda dapat mengonfigurasi Athena untuk menggunakannya. Untuk melakukannya, Anda membuat nama sumber data yang merujuk ke metastore Hive eksternal untuk digunakan dalam kueri Athena Anda.

Untuk menghubungkan Athena ke metastore Hive Anda menggunakan fungsi Lambda yang ada

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.

3. Pilih Sumber data.
4. Pada halaman Sumber data, pilih Buat sumber data.
5. Pada halaman Pilih sumber data, untuk Sumber data, pilih metastore S3 - Apache Hive.
6. Pilih Berikutnya.
7. Di bagian Rincian sumber data, untuk nama sumber data, masukkan nama yang ingin Anda gunakan dalam pernyataan SQL saat Anda menanyakan sumber data dari Athena (misalnya`MyHiveMetastore`,). Nama bisa sampai 127 karakter dan harus unik di akun Anda. Itu tidak dapat diubah setelah Anda membuatnya. Karakter yang valid adalah a-z, A-Z, 0-9, _ (garis bawah), @ (pada tanda) dan - (tanda hubung). Nama`awsdatacatalog`,, `hivejmx`, dan dicadangkan `system` oleh Athena dan tidak dapat digunakan untuk nama sumber data.
8. Di bagian Detail koneksi, gunakan kotak Pilih atau masukkan fungsi Lambda untuk memilih nama fungsi yang baru saja Anda buat. ARN dari fungsi Lambda ditampilkan.
9. (Opsional) Untuk Tag, tambahkan pasangan nilai kunci untuk dikaitkan dengan sumber data ini. Untuk informasi selengkapnya tentang tag, lihat Menandai sumber daya Athena.
10. Pilih Berikutnya.
11. Pada halaman Tinjau dan buat, tinjau detail sumber data, lalu pilih Buat sumber data.
12. Bagian Detail sumber data pada halaman untuk sumber data Anda menunjukkan informasi tentang konektor baru Anda.

Anda sekarang dapat menggunakan nama sumber Data yang Anda tentukan untuk referensi metastore Hive dalam kueri SQL Anda di Athena.

Dalam kueri SQL Anda, gunakan sintaks contoh berikut, ganti `ehms-catalog` dengan nama sumber data yang Anda tentukan sebelumnya.

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

13. Untuk melihat, mengedit, atau menghapus sumber data yang Anda buat, lihatMengelola sumber data.

Menggunakan nama sumber data default dalam kueri metastore Hive eksternal

Saat Anda menjalankan DDLL dan DDL kueri pada metastores Hive eksternal, Anda dapat menyederhanakan sintaks kueri Anda dengan menghilangkan nama katalog jika nama yang dipilih dalam editor kueri. Pembatasan tertentu berlaku untuk fungsi ini.

Pernyataan DXML

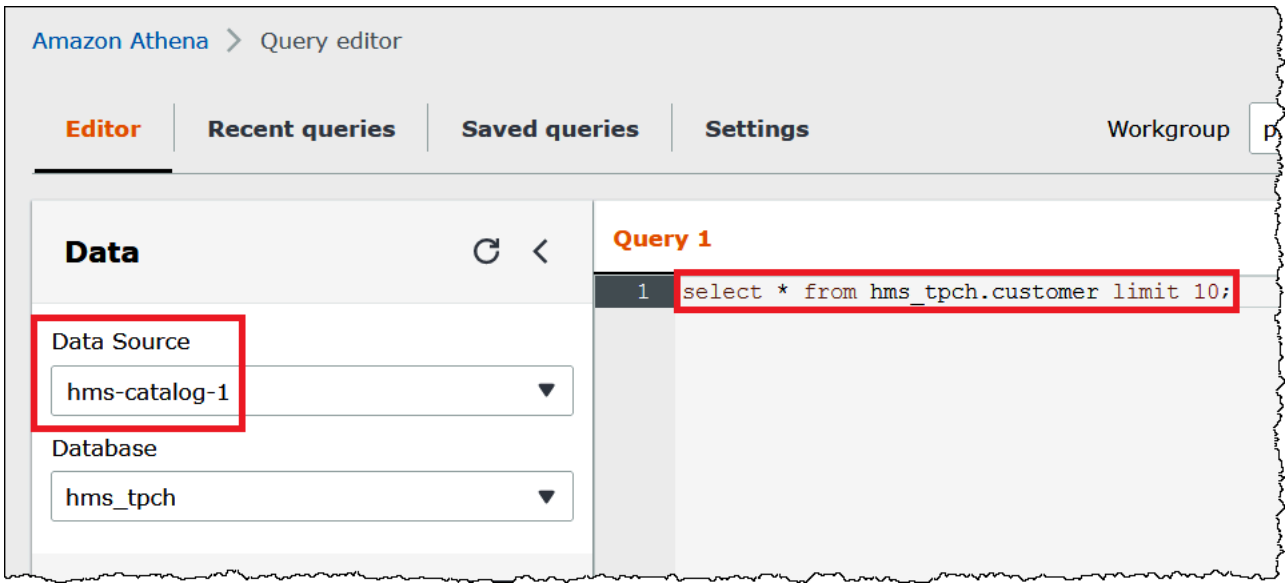
Untuk menjalankan kueri dengan katalog terdaftar

1. Anda dapat menempatkan nama sumber data sebelum database menggunakan sintaks`[[`*`data_source_name`*`].`*`database_name`*`].`*`table_name`*, seperti pada contoh berikut.

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

2. Ketika sumber data yang ingin Anda gunakan sudah dipilih di editor kueri, Anda dapat menghilangkan nama dari kueri, seperti pada contoh berikut.

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



3. Bila Anda menggunakan beberapa sumber data dalam kueri, Anda hanya dapat menghilangkan nama sumber data default, dan harus menentukan nama lengkap untuk sumber data non-default.

   Misalnya, misalkan `AwsDataCatalog` dipilih sebagai sumber data default di editor kueri. `FROM`Pernyataan dalam kutipan kueri berikut sepenuhnya memenuhi syarat dua nama sumber data pertama tetapi menghilangkan nama untuk sumber data ketiga karena ada di katalog data. AWS Glue

   ```
   ...
   FROM ehms01.hms_tpch.customer,
            "hms-catalog-1".hms_tpch.orders,
            hms_tpch.lineitem
   ...
   ```

Pernyataan DDL

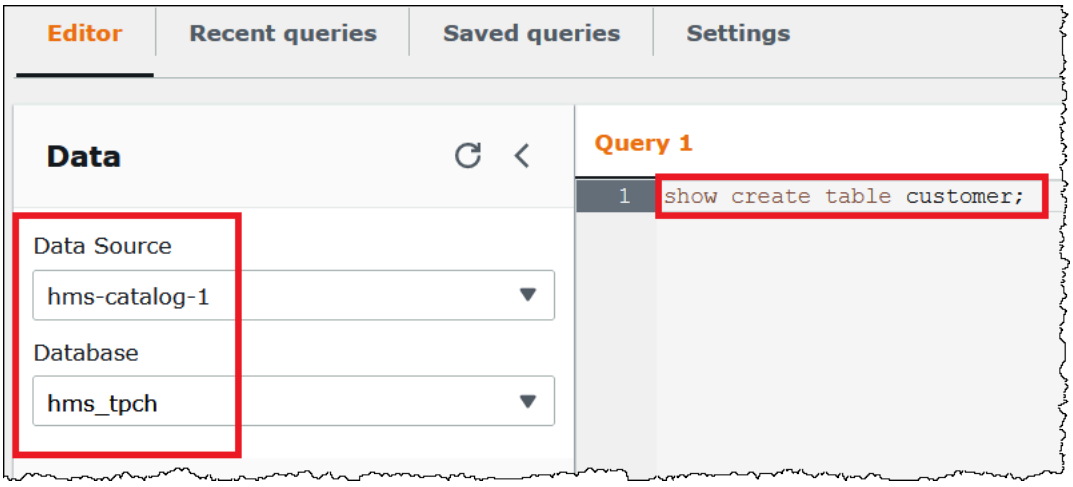
Pernyataan Athena DDL berikut mendukung awalan nama katalog. Nama katalog awalan dalam pernyataan DDL lainnya menyebabkan kesalahan sintaks.

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

Seperti halnya pernyataan DHTML, Anda dapat menghilangkan sumber data dan awalan database dari kueri saat sumber data dan database dipilih di editor kueri.

Pada gambar berikut, sumber `hms-catalog-1` data dan `hms_tpch` database dipilih di editor kueri. `show create table customer`Pernyataan berhasil meskipun `hms-catalog-1` awalan dan nama `hms_tpch` database dihilangkan dari kueri itu sendiri.



Menentukan sumber data default dalam string koneksi JDBC

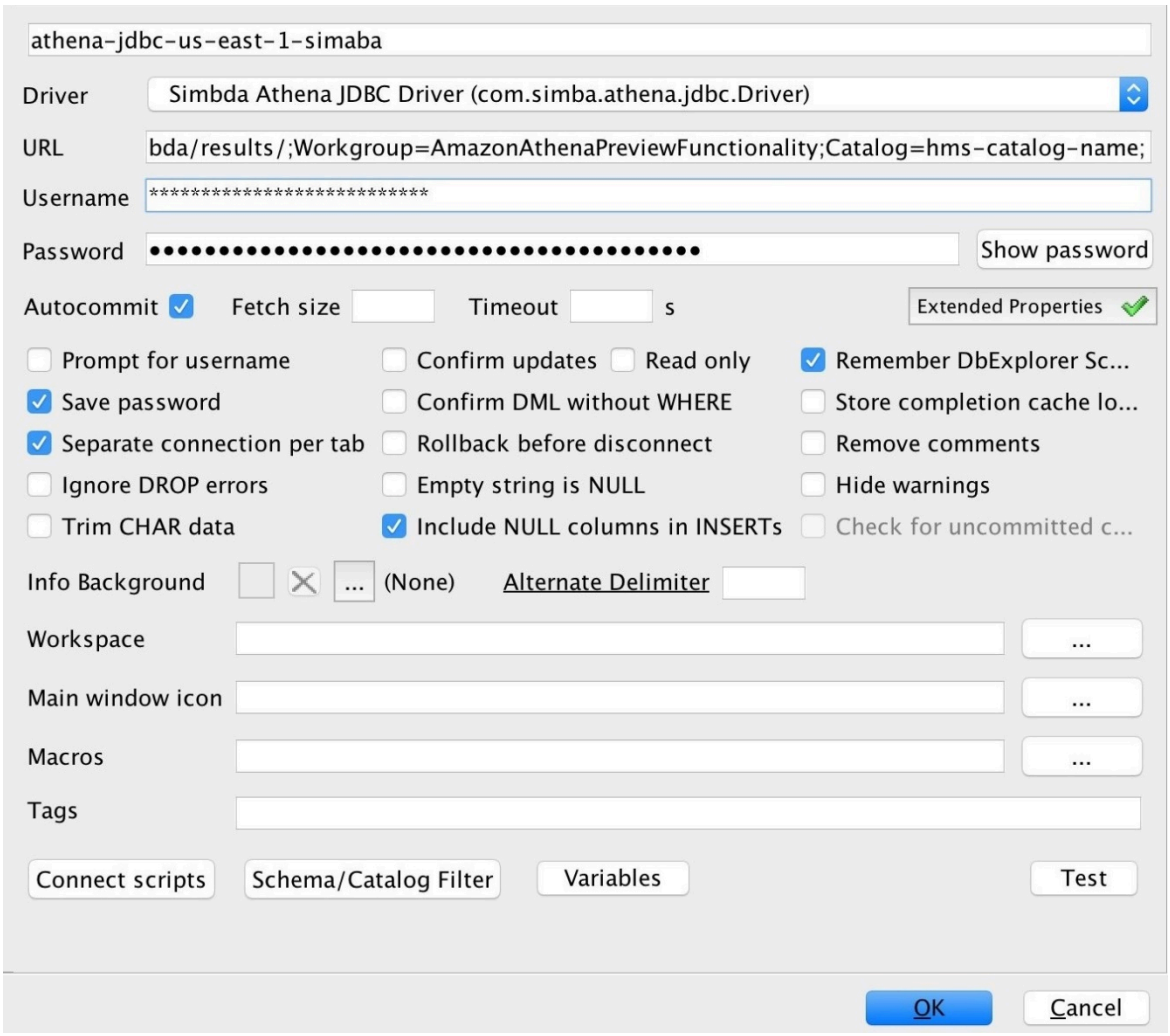
Saat Anda menggunakan Driver Athena JDBC untuk menghubungkan Athena ke metastore Hive eksternal, Anda dapat menggunakan `Catalog` parameter untuk menentukan nama sumber data default dalam string koneksi Anda di editor SQL seperti meja kerja SQL.

> **Note**
>
> Untuk mengunduh driver Athena JDBC terbaru, lihat Menggunakan Athena dengan driver JDBC.

String koneksi berikut menentukan sumber *`hms-catalog-name`*data default.

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

Gambar berikut menunjukkan contoh URL koneksi JDBC seperti yang dikonfigurasi dalam SQL Workbench.



Bekerja dengan tampilan Hive

Anda dapat menggunakan Athena untuk menanyakan tampilan yang ada di metastores Apache Hive eksternal Anda. Athena menerjemahkan pandangan Anda untuk Anda on-the-fly saat runtime tanpa mengubah tampilan asli atau menyimpan terjemahannya.

Misalnya, Anda memiliki tampilan Hive seperti berikut yang menggunakan sintaks yang tidak didukung di Athena seperti: `LATERAL VIEW explode()`

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

Athena menerjemahkan string kueri tampilan Hive menjadi pernyataan seperti berikut yang dapat dijalankan Athena:

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

Untuk informasi tentang menghubungkan metastore Hive eksternal ke Athena, lihat. Menggunakan Athena Data Connector untuk Eksternal Hive Metastore

Pertimbangan dan batasan

Saat menanyakan pandangan Hive dari Athena, pertimbangkan hal-hal berikut:

- Athena tidak mendukung pembuatan tampilan Hive. Anda dapat membuat tampilan Hive di metastore Hive eksternal Anda, yang kemudian dapat Anda kueri dari Athena.
- Athena tidak mendukung UDF khusus untuk tampilan Hive.
- Karena masalah yang diketahui di konsol Athena, tampilan Sarang muncul di bawah daftar tabel, bukan daftar tampilan.
- Meskipun proses penerjemahan otomatis, fungsi Hive tertentu tidak didukung untuk tampilan Hive atau memerlukan penanganan khusus. Untuk informasi selengkapnya, lihat bagian berikut.

Keterbatasan dukungan fungsi sarang

Bagian ini menyoroti fungsi Sarang yang tidak didukung Athena untuk tampilan Hive atau yang memerlukan perlakuan khusus. Saat ini, karena Athena terutama mendukung fungsi dari Hive 2.2.0, fungsi yang hanya tersedia dalam versi yang lebih tinggi (seperti Hive 4.0.0) tidak tersedia. Untuk daftar lengkap fungsi Hive, lihat Manual bahasa Hive UDF.

Fungsi agregat

Fungsi agregat yang membutuhkan penanganan khusus

Fungsi agregat berikut untuk tampilan Hive memerlukan penanganan khusus.

- Rata-rata — Alih-alih`avg(INT i)`, gunakan`avg(CAST(i AS DOUBLE))`.

Fungsi agregat tidak didukung

Fungsi agregat Hive berikut tidak didukung di Athena untuk tampilan Hive.

```
covar_pop
histogram_numeric
ntile
percentile
percentile_approx
```

Fungsi regresi seperti`regr_count`,`regr_r2`, dan tidak `regr_sxx` didukung di Athena untuk tampilan Hive.

Fungsi tanggal tidak didukung

Fungsi tanggal sarang berikut tidak didukung di Athena untuk tampilan Hive.

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

Fungsi masking tidak didukung

Fungsi penyembunyian sarang seperti`mask()`, dan tidak `mask_first_n()` didukung di Athena untuk tampilan Sarang.

Fungsi lain-lain

Fungsi lain-lain yang membutuhkan penanganan khusus

Fungsi lain-lain berikut untuk tampilan Hive memerlukan penanganan khusus.

- md5 — Athena `md5(binary)` mendukung tetapi tidak. `md5(varchar)`
- Meledak - Athena `explode` mendukung ketika digunakan dalam sintaks berikut:

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

- Posexplode - Athena mendukung `posexplode` ketika digunakan dalam sintaks berikut:

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

Dalam `(pos, val)` output, Athena memperlakukan `pos` kolom sebagai. `BIGINT` Karena itu, Anda mungkin perlu mentransmisikan `pos` kolom `BIGINT` untuk menghindari tampilan basi. Contoh berikut menggambarkan teknik ini.

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

Fungsi lain-lain tidak didukung

Fungsi Hive berikut tidak didukung di Athena untuk tampilan Hive.

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

Operator

Operator yang membutuhkan penanganan khusus

Operator berikut untuk tampilan Hive memerlukan penanganan khusus.

- Operator mod (%) - Karena `DOUBLE` tipe secara implisit melemparkan ke`DECIMAL(x,y)`, sintaks berikut dapat menyebabkan pesan kesalahan View is stale:

```
a_double % 1.0 AS column
```

Untuk mengatasi masalah ini, gunakanCAST, seperti pada contoh berikut.

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

- Operator divisi (/) — Di Hive, `int` dibagi dengan `int` menghasilkan a`double`. Di Athena, operasi yang sama menghasilkan terpotong. `int`

Operator tidak didukung

Athena tidak mendukung operator berikut untuk tampilan Hive.

~A — bitwise `NOT`

A ^ b — bitwise `XOR`

A & b — bitwise `AND`

A | b — bitwise `OR`

A **<=>** b — Mengembalikan hasil yang sama dengan operator sama dengan (=) untuk operan non-null. Mengembalikan `TRUE` `FALSE` jika keduanya`NULL`, jika salah satunya`NULL`.

Fungsi string

Fungsi string yang membutuhkan penanganan khusus

Fungsi string Hive berikut untuk tampilan Hive memerlukan penanganan khusus.

- chr (bigint|double a) — Hive memungkinkan argumen negatif; Athena tidak.
- instr (string str, string substr) - Karena pemetaan Athena untuk `instr` fungsi kembali `BIGINT` alih-alih, gunakan sintaks berikut: `INT`

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

  Tanpa langkah ini, tampilan akan dianggap basi.
- length (string a) - Karena pemetaan Athena untuk `length` fungsi kembali `BIGINT` alih-alih`INT`, gunakan sintaks berikut sehingga tampilan tidak akan dianggap basi:

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

Fungsi string tidak didukung

Fungsi string Hive berikut tidak didukung di Athena untuk tampilan Hive.

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

Fungsi XPath tidak didukung

Hive XPath berfungsi seperti`xpath`,`xpath_short`, dan tidak `xpath_int` didukung di Athena untuk tampilan Hive.

Pemecahan Masalah

Saat Anda menggunakan tampilan Hive di Athena, Anda mungkin mengalami masalah berikut:

- Tampilan basi <view name>— Pesan ini biasanya menunjukkan ketidakcocokan tipe antara tampilan di Hive dan Athena. Jika fungsi yang sama dalam fungsi Hive LanguageManual UDF dan Presto dan dokumentasi operator memiliki tanda tangan yang berbeda, coba casting tipe data yang tidak cocok.

- Fungsi tidak terdaftar - Athena saat ini tidak mendukung fungsi tersebut. Untuk detailnya, lihat informasi sebelumnya dalam dokumen ini.

Menggunakan AWS CLI metastores with Hive

Anda dapat menggunakan`aws athena`CLI perintah untuk mengelola katalog data metastore Hive yang Anda gunakan dengan Athena. Setelah Anda menentukan satu atau lebih katalog untuk digunakan dengan Athena, Anda dapat mereferensikan katalog tersebut di`aws athena`DDDL dan DDLL perintah.

Menggunakan AWS CLI untuk mengelola katalog metastore Hive

Mendaftarkan katalog: C reate-data-catalog

Untuk mendaftarkan katalog data, Anda menggunakan`create-data-catalog`Perintah. Gunakan`name`Parameter untuk menentukan nama yang ingin Anda gunakan sebagai katalog. Lulus ARN fungsi Lambda ke`metadata-function`opsi`parameters`argumen. Untuk membuat tanda untuk katalog baru, gunakan`tags`Parameter dengan satu atau beberapa spasi yang dipisahkan`Key=`*`key`*`,Value=`*`value`*pasangan argumen.

Contoh berikut register katalog metastore Hive bernama`hms-catalog-1`. Perintah telah diformat untuk dibaca.

```
$ 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
```

Menampilkan rincian katalog: G et-data-catalog

Untuk menampilkan detail katalog, lulus nama katalog untuk`get-data-catalog`seperti pada contoh berikut.

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

Hasil contoh berikut adalah dalam format JSON.

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

Daftar katalog terdaftar: L ist-data-catalogs

Untuk daftar katalog terdaftar, gunakan`list-data-catalogs`dan opsional menentukan wilayah seperti pada contoh berikut. Katalog yang terdaftar selalu mencakup AWS Glue.

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

Hasil contoh berikut adalah dalam format JSON.

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

Memperbarui katalog: U pdate-data-catalog

Untuk memperbarui katalog data, gunakan`update-data-catalog`seperti pada contoh berikut. Perintah telah diformat untuk dibaca.

```
$ 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
```

Menghapus katalog: D elete-data-catalog

Untuk menghapus katalog data, gunakan`delete-data-catalog`seperti pada contoh berikut.

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

Menampilkan rincian database: Get-Database

Untuk menampilkan detail basis data, lulus nama katalog dan basis data untuk`get-database`seperti pada contoh berikut.

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

Hasil contoh berikut adalah dalam format JSON.

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

Daftar database dalam katalog: Daftar-database

Untuk daftar basis data dalam katalog, gunakan`list-databases`dan opsional menentukan wilayah seperti pada contoh berikut.

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

Hasil contoh berikut adalah dalam format JSON.

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

Menampilkan detail tabel: G et-table-metadata

Untuk menampilkan metadata untuk tabel, termasuk nama kolom dan tipe data, lulus nama katalog, basis data, dan nama tabel untuk`get-table-metadata`seperti pada contoh berikut.

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

Hasil contoh berikut adalah dalam format JSON.

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

Menampilkan metadata untuk semua tabel dalam database: L ist-table-metadata

Untuk menampilkan metadata untuk semua tabel dalam basis data, lulus nama katalog dan basis data nama untuk`list-table-metadata`Perintah. Parameter`list-table-metadata`serupa

dengan`get-table-metadata`, kecuali bahwa Anda tidak menentukan nama tabel. Untuk membatasi jumlah hasil, Anda dapat menggunakan`--max-results`, seperti pada contoh berikut.

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

Hasil contoh berikut adalah dalam format JSON.

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

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

Menjalankan pernyataan DDL dan DML

Bila Anda menggunakan AWS CLI untuk menjalankan pernyataan DDL dan DHTML, Anda dapat meneruskan nama katalog metastore Hive dalam salah satu dari dua cara:

- Langsung ke pernyataan yang mendukungnya.
- Ke`--query-execution-context` `Catalog`parameter.

Pernyataan DDL

Contoh berikut melewati dalam nama katalog langsung sebagai bagian dari`show create table`Pernyataan DDL. Perintah telah diformat untuk dibaca.

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

Contoh berikut DDL`show create table`pernyataan menggunakan`Catalog`parameter`--query-execution-context`untuk lulus nama katalog Hive metastore`hms-catalog-1`. Perintah telah diformat untuk dibaca.

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

Pernyataan DXML

Contoh DDLL berikut`select`pernyataan melewati nama katalog ke dalam kueri secara langsung. Perintah telah diformat untuk dibaca.

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

Contoh DDLL berikut`select`pernyataan menggunakan`Catalog`parameter`--query-execution-context`untuk lulus dalam nama katalog Hive metastore`hms-catalog-1`. Perintah telah diformat untuk dibaca.

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

Implementasi referensi

Athena menyediakan implementasi referensi konektornya untuk metastore Hive eksternal di.com di https://github.com/awslabs/. GitHub aws-athena-hive-metastore

Implementasi referensi adalahApache Mavenproyek yang memiliki modul berikut:

- **`hms-service-api`**— Berisi operasi API antara fungsi Lambda dan klien layanan Athena. Operasi API ini didefinisikan dalam`HiveMetaStoreService`antarmuka. Karena ini adalah kontrak layanan, Anda tidak harus mengubah apa pun dalam modul ini.
- **`hms-lambda-handler`**- Satu set default Lambda penangan yang memproses semua Hive metastore API panggilan. Kelas`MetadataHandler`adalah petugas operator untuk semua panggilan API. Anda tidak perlu mengubah paket ini.
- **`hms-lambda-func`**— Contoh fungsi Lambda yang memiliki komponen-komponen berikut.
  - **`HiveMetaStoreLambdaFunc`**— Contoh fungsi Lambda yang meluas`MetadataHandler`.
  - **`ThriftHiveMetaStoreClient`**— Klien Thrift yang berkomunikasi dengan Hive metastore. Klien ini ditulis untuk Hive 2.3.0. Jika Anda menggunakan versi Hive yang berbeda, Anda mungkin perlu memperbarui kelas ini untuk memastikan bahwa objek respon yang kompatibel.
  - **`ThriftHiveMetaStoreClientFactory`**— Mengontrol perilaku fungsi Lambda. Sebagai contoh, Anda dapat menyediakan seperangkat penyedia handler Anda sendiri dengan mengesampingkan`getHandlerProvider()`metode.
  - `hms.properties`— Mengonfigurasi fungsi Lambda. Sebagian besar kasus memerlukan memperbarui dua properti berikut saja.
    - `hive.metastore.uris`— URI dari metastore Hive dalam format`thrift://<host_name>:9083`.
    - `hive.metastore.response.spill.location`: Lokasi Amazon S3 untuk menyimpan objek respons ketika ukurannya melebihi ambang batas tertentu (misalnya, 4 MB). Ambang batas didefinisikan dalam properti`hive.metastore.response.spill.threshold`. Mengubah nilai default tidak dianjurkan.

> **Note**
>
> Kedua properti ini dapat diganti olehVariabel lingkungan Lambda `HMS_URIS`dan`SPILL_LOCATION`. Gunakan variabel ini bukannya mengkompilasi ulang kode sumber untuk fungsi Lambda saat Anda ingin menggunakan fungsi dengan metastore Hive atau tumpahan lokasi yang berbeda.

- **`hms-lambda-layer`**- Sebuah proyek perakitan Maven yang menempatkan`hms-service-api`,`hms-lambda-handler`, dan dependensi mereka menjadi`.zip`file. Parameter`.zip`terdaftar sebagai lapisan Lambda untuk digunakan oleh beberapa fungsi Lambda.
- **`hms-lambda-rnp`**— Merekam respons dari fungsi Lambda dan kemudian menggunakannya untuk memutar ulang respons. Anda dapat menggunakan model ini untuk mensimulasikan respons Lambda untuk pengujian.

Membangun artefak sendiri

Sebagian besar kasus penggunaan tidak mengharuskan Anda untuk memodifikasi implementasi referensi. Namun, jika perlu, Anda dapat memodifikasi kode sumber, membangun artifact sendiri, dan mengunggahnya ke lokasi Amazon S3.

Sebelum Anda membangun artifact, memperbarui propertihive.metastore.urisdanhive.metastore.response.spill.locationdihms.properties dalamhms-lambda-funcmodul.

Untuk membangun artifact, Anda harus memiliki Apache Maven diinstal dan menjalankan perintah`mvn install`. Ini menghasilkan lapisan`.zip`file dalam folder output yang disebut`target`dalam modul`hms-lambda-layer`dan fungsi Lambda`.jar`berkas dalam modul`hms-lambd-func`.

Menggunakan Amazon Athena

Jika Anda memiliki data dalam sumber selain Amazon S3, Anda dapat menggunakan Kueri Gabungan Athena untuk mengkueri data di tempat atau membangun alur yang mengekstraksi data dari beberapa sumber data dan menyimpannya di Amazon S3. Dengan Kueri Gabungan Athena, Anda dapat menjalankan kueri SQL di seluruh data yang disimpan dalam sumber data relasional, non-relasional, objek, dan kustom.

Athena menggunakan konektor sumber data yang berjalan AWS Lambda untuk menjalankan kueri federasi. Konektor sumber data adalah bagian dari kode yang dapat menerjemahkan antara sumber data target Anda dan Athena. Anda bisa memikirkan konektor sebagai perpanjangan mesin permintaan Athena. Konektor sumber data Athena bawaan ada untuk sumber data seperti Amazon Logs, Amazon DynamoDB, CloudWatch Amazon DocumentDB, dan Amazon RDS, dan sumber data relasional yang sesuai dengan JDBC seperti MySQL, dan PostgreSQL di bawah lisensi Apache 2.0. Anda juga dapat menggunakan Athena Kueri Federation SDK untuk menulis konektor kustom. Untuk memilih, mengonfigurasi, dan menggunakan konektor sumber data ke akun Anda, Anda dapat

menggunakan konsol Athena dan Lambda atau AWS Serverless Application Repository. Setelah Anda men-deploy konektor sumber data, konektor dikaitkan dengan katalog yang dapat Anda tentukan dalam SQL kueri. Anda dapat menggabungkan pernyataan SQL dari beberapa katalog dan rentang beberapa sumber data dengan satu kueri.

Saat kueri diajukan terhadap sumber data, Athena memanggil konektor yang sesuai untuk mengidentifikasi bagian-bagian tabel yang perlu dibaca, mengelola paralelisme, dan menekan predikat filter. Berdasarkan pengguna mengirimkan kueri, konektor dapat menyediakan atau membatasi akses ke elemen data tertentu. Konektor menggunakan Apache Arrow sebagai format untuk mengembalikan data yang diminta dalam kueri, yang memungkinkan konektor untuk diimplementasikan dalam bahasa seperti C, C ++, Java, Python, dan Rust. Karena konektor diproses di Lambda, konektor dapat digunakan untuk mengakses data dari sumber data apa pun di cloud atau lokal yang dapat diakses dari Lambda.

Untuk menulis konektor sumber data Anda sendiri, Anda dapat menggunakan Athena Kueri Federation SDK untuk menyesuaikan salah satu konektor prebuilt yang disediakan dan dipelihara Amazon Athena. Anda dapat memodifikasi salinan kode sumber dari GitHub repositori dan kemudian menggunakan alat publikasi Connector untuk membuat paket Anda sendiri AWS Serverless Application Repository .

> **Note**
>
> Developer pihak ketiga mungkin telah menggunakan Athena Kueri Federation SDK untuk menulis konektor sumber data. Untuk masalah dukungan atau lisensi dengan konektor sumber data ini, silakan bekerja dengan penyedia konektor Anda. Konektor ini tidak diuji atau didukung oleh AWS.

Untuk daftar konektor sumber data yang ditulis dan diuji oleh Athena, lihat Konektor sumber data yang tersedia.

Untuk informasi tentang menulis konektor sumber data Anda sendiri, lihat Contoh konektor Athena aktif. GitHub

Pertimbangan dan batasan

- Versi mesin - Athena Federated Query hanya didukung pada mesin Athena versi 2 dan versi yang lebih baru. Untuk informasi selengkapnya tentang versi mesin Aurora, lihat Pembuatan versi mesin Athena.

- Tampilan — Anda dapat membuat dan menanyakan tampilan pada sumber data gabungan. Tampilan federasi disimpan di AWS Glue, bukan sumber data yang mendasarinya. Untuk informasi selengkapnya, lihat Menanyakan pandangan federasi.
- Operasi tulis - Operasi tulis seperti INSERT INTO tidak didukung. Mencoba melakukannya dapat mengakibatkan pesan kesalahan Operasi ini saat ini tidak didukung untuk katalog eksternal.
- Harga - Untuk informasi harga, lihat Harga Amazon Athena.

  Driver JDBC - Untuk menggunakan driver JDBC dengan kueri gabungan atau metastore Hive eksternal, termasuk`MetadataRetrievalMethod=ProxyAPI` dalam string koneksi JDBC Anda. Untuk informasi tentang driver JDBC, lihat Menghubungkan ke Amazon Athena dengan JDBC.
- Secrets Manager - Untuk menggunakan fitur Kueri Gabungan Athena dengan AWS Secrets Manager, Anda harus mengonfigurasi Vpc endpoint privat untuk Secrets Manager. Untuk informasi selengkapnya, lihat Membuat titik akhir pribadi VPC Secrets Manager di Panduan Pengguna.AWS Secrets Manager

Konektor sumber data mungkin memerlukan akses ke sumber daya berikut agar berfungsi dengan benar. Jika Anda menggunakan konektor prebuilt, periksa informasi untuk konektor untuk memastikan bahwa Anda telah mengonfigurasi VPC Anda dengan benar. Juga, memastikan bahwa IAM utama menjalankan kueri dan menciptakan konektor memiliki hak untuk tindakan yang diperlukan. Untuk informasi selengkapnya, lihat Contoh kebijakan izin IAM untuk mengizinkan Kueri Federasi Athena.

- Amazon S3— Selain menulis hasil kueri ke lokasi hasil kueri Athena di Amazon S3, konektor data juga menulis ke bucket tumpahan di Amazon S3. Konektivitas dan izin ke lokasi Amazon S3 ini diperlukan.
- Athena— Sumber data memerlukan konektivitas ke Athena dan sebaliknya untuk memeriksa status kueri dan mencegah overscan.
- AWS Glue Data Catalog— Konektivitas dan izin diperlukan jika konektor Anda menggunakan Katalog Data untuk metadata tambahan atau primer.

Video

Tonton video berikut untuk mempelajari selengkapnya tentang menggunakan Kueri Gabungan Athena.

Video: Analisis Hasil Kueri Federasi di Amazon Athena di Amazon QuickSight

Video berikut menunjukkan cara menganalisis hasil kueri federasi Athena di Amazon. QuickSight

Analisis hasil kueri federasi di Amazon Athena di Amazon QuickSight

Video: Alur Analitik Game

Video berikut menunjukkan cara men-deploy alur data tanpa server terukur untuk menelan, menyimpan, dan menganalisis data telemetri dari game dan layanan menggunakan kueri Gabungan Amazon Athena.

Pipa analitik game

Konektor sumber data yang tersedia

Bagian ini berisi daftar konektor sumber data Athena yang dapat Anda gunakan untuk mengkueri berbagai sumber data eksternal ke Amazon S3. Untuk menggunakan konektor dalam pertanyaan Athena Anda, konfigurasikan dan sebarkan ke akun Anda.

Pertimbangan dan batasan

- Beberapa konektor prebuilt mengharuskan Anda membuat VPC dan grup keamanan sebelum Anda dapat menggunakan konektor. Untuk informasi tentang membuat VPC, lihatMembuat VPC untuk konektor sumber data.
- Untuk menggunakan fitur Kueri Federasi Athena AWS Secrets Manager, Anda harus mengonfigurasi titik akhir pribadi Amazon VPC untuk Secrets Manager. Untuk informasi selengkapnya, lihat Membuat titik akhir pribadi VPC Secrets Manager di Panduan Pengguna.AWS Secrets Manager
- Untuk konektor yang tidak mendukung pushdown predikat, kueri yang menyertakan predikat membutuhkan waktu lebih lama untuk dieksekusi. Untuk kumpulan data kecil, sangat sedikit data yang dipindai, dan kueri membutuhkan waktu rata-rata sekitar 2 menit. Namun, untuk kumpulan data besar, banyak kueri dapat habis waktu.
- Beberapa sumber data federasi menggunakan terminologi untuk merujuk objek data yang berbeda dari Athena. Untuk informasi selengkapnya, lihat Athena dan kualifikasi nama tabel federasi.
- Untuk konektor yang tidak mendukung pagination ketika Anda daftar tabel, layanan web dapat time out jika database Anda memiliki banyak tabel dan metadata. Konektor berikut menyediakan dukungan pagination untuk daftar tabel:
  - DocumentDB
  - DynamoDB

- MySQL
- OpenSearch
- Oracle
- PostgreSQL
- Redshift
- SQL Server

Informasi tambahan

- Untuk informasi tentang penggunaan konektor sumber data Athena, lihat Menyebarkan konektor sumber data.
- Untuk informasi tentang kueri yang menggunakan konektor sumber data Athena, lihat. Menjalankan kueri federasi
- Untuk informasi mendalam tentang konektor sumber data Athena, lihat Konektor yang tersedia di. GitHub

Konektor sumber data Athena

- Konektor Gen2 Penyimpanan Data Lake Amazon Athena Azure (ADLS)
- Konektor Amazon Athena Azure Synapse
- Konektor Amazon Athena Cloudera Hive
- Konektor Amazon Athena Cloudera Impala
- Konektor Amazon Athena CloudWatch
- Konektor Metrik Amazon Athena CloudWatch
- Konektor CMDB Amazon Athena AWS
- Konektor Amazon Athena IBM Db2
- Konektor Amazon Athena IBM Db2 AS/400 (Db2 iSeries)
- Konektor Amazon Athena DocumentDB
- Konektor Amazon Athena DynamoDB
- Konektor Google Amazon Athena BigQuery
- Konektor Penyimpanan Awan Google Amazon Athena
- Konektor Amazon Athena HBase
- Konektor Amazon Athena Hortonworks

- Konektor Amazon Athena Apache Kafka
- Konektor MSK Amazon Athena
- Konektor MySQL Amazon Athena
- Konektor Amazon Athena Neptunus
- Konektor Amazon Athena OpenSearch
- Konektor Amazon Athena Oracle
- Konektor Amazon Athena PostgreSQL
- Konektor Amazon Athena Redis
- Konektor Pergeseran Merah Amazon Athena
- Konektor Amazon Athena SAP HANA
- Konektor Kepingan Salju Amazon Athena
- Konektor Amazon Athena Microsoft SQL Server
- Konektor Amazon Athena Teradata
- Konektor Timestream Amazon Athena
- Konektor DS (TPC-DS) patokan Amazon Athena TPC
- Konektor Amazon Athena Vertica

> **Note**
>
> AthenaJdbcConnector(versi terbaru 2022.4.1) telah usang. Sebagai gantinya, gunakan konektor khusus database seperti yang untuk MySQL, Redshift, atau PostgreSQL.

Konektor Gen2 Penyimpanan Data Lake Amazon Athena Azure (ADLS)

Konektor Amazon Athena untuk Azure Data Lake Storage (ADLS) Gen2 memungkinkan Amazon Athena menjalankan kueri SQL pada data yang disimpan di ADLS. Athena tidak dapat mengakses file yang disimpan di danau data secara langsung.

- Alur Kerja - Konektor mengimplementasikan antarmuka JDBC, yang menggunakan driver. `com.microsoft.sqlserver.jdbc.SQLServerDriver` Konektor meneruskan kueri ke mesin Azure Synapse, yang kemudian mengakses data lake.
- Penanganan data dan S3 - Biasanya, konektor Lambda menanyakan data secara langsung tanpa transfer ke Amazon S3. Namun, ketika data yang dikembalikan oleh fungsi Lambda melebihi batas

Lambda, data akan ditulis ke bucket tumpahan Amazon S3 yang Anda tentukan sehingga Athena dapat membaca kelebihannya.

- Otentikasi AAD — AAD dapat digunakan sebagai metode otentikasi untuk konektor Azure Synapse. Untuk menggunakan AAD, string koneksi JDBC yang digunakan konektor harus berisi parameter URL`authentication=ActiveDirectoryServicePrincipal`,, `AADSecurePrincipalId` dan. `AADSecurePrincipalSecret` Parameter ini dapat diteruskan secara langsung atau oleh Secrets Manager.

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Batasan

- Menulis operasi DDL tidak didukung.
- Dalam pengaturan multiplexer, bucket tumpahan dan awalan dibagikan di semua instance database.
- Batas Lambda yang relevan. Untuk informasi selengkapnya, lihat Kuota Lambda dalam Panduan Pengembang AWS Lambda .
- Tipe data tanggal dan stempel waktu dalam kondisi filter harus dilemparkan ke tipe data yang sesuai.

Ketentuan

Istilah-istilah berikut terkait dengan konektor Azure Data Lake Storage Gen2.

- Instans database — Instance apa pun dari database yang digunakan di tempat, di Amazon EC2, atau di Amazon RDS.
- Handler - Handler Lambda yang mengakses instance database Anda. Handler bisa untuk metadata atau untuk catatan data.
- Metadata handler — Penangan Lambda yang mengambil metadata dari instance database Anda.
- Record handler - Handler Lambda yang mengambil catatan data dari instance database Anda.

- Composite handler — Handler Lambda yang mengambil data metadata dan data dari instance database Anda.
- Properti atau parameter - Properti database yang digunakan oleh penangan untuk mengekstrak informasi database. Anda mengonfigurasi properti ini sebagai variabel lingkungan Lambda.
- Connection String — Sebuah string teks yang digunakan untuk membuat koneksi ke instance database.
- Katalog —AWS Glue Non-katalog yang terdaftar di Athena yang merupakan awalan yang diperlukan untuk properti. `connection_string`
- Multiplexing handler - Handler Lambda yang dapat menerima dan menggunakan beberapa koneksi database.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengonfigurasi konektor Azure Data Lake Storage Gen2.

String koneksi

Gunakan string koneksi JDBC dalam format berikut untuk terhubung ke instance database.

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

Menggunakan handler multiplexing

Anda dapat menggunakan multiplexer untuk terhubung ke beberapa instance database dengan satu fungsi Lambda. Permintaan dirutekan dengan nama katalog. Gunakan kelas-kelas berikut di Lambda.

Handler	Kelas
Handler komposit	`DataLakeGen2MuxCompositeHandler`
Penangan metadata	`DataLakeGen2MuxMetadataHandler`
Rekam handler	`DataLakeGen2MuxRecordHandler`

Parameter handler multiplexing

Parameter	Deskripsi
$*catalog*_connection_string	Wajib. Sebuah string koneksi instance database. Awalan variabel lingkungan dengan nama katalog yang digunakan di Athena. Misalnya, jika katalog terdaftar di Athena adalah`mydatalakegentwocatalog` , maka nama variabel lingkungan adalah. `mydatalakegentwocatalog_connection_string`
`default`	Wajib. String koneksi default. String ini digunakan ketika katalog `lambda:${` *`}AWS_LAMBDA_FUNCTION_NAME`* .

Contoh properti berikut adalah untuk fungsi DataLakeGen 2 MUX Lambda yang mendukung dua instance database`datalakegentwo1`: (default), dan. `datalakegentwo2`

Properti	Nilai
`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`* `}`

Memberikan kredensyal

Untuk memberikan nama pengguna dan kata sandi untuk database Anda dalam string koneksi JDBC Anda, Anda dapat menggunakan properti string koneksi atau. AWS Secrets Manager

- Connection String - Nama pengguna dan kata sandi dapat ditentukan sebagai properti dalam string koneksi JDBC.

> **Important**
>
> Sebagai praktik keamanan terbaik, jangan gunakan kredensyal hardcode dalam variabel lingkungan atau string koneksi Anda. Untuk informasi tentang memindahkan rahasia hardcode Anda AWS Secrets Manager, lihat Memindahkan rahasia hardcode ke AWS Secrets Manager dalam Panduan Pengguna.AWS Secrets Manager

- AWS Secrets Manager— Untuk menggunakan fitur Query Federasi Athena dengan, VPC AWS Secrets Manager yang terhubung ke fungsi Lambda Anda harus memiliki akses internet atau titik akhir VPC untuk terhubung ke Secrets Manager.

  Anda dapat memasukkan nama rahasia ke AWS Secrets Manager dalam string koneksi JDBC Anda. Konektor menggantikan nama rahasia dengan `password` nilai `username` dan dari Secrets Manager.

  Untuk instans database Amazon RDS, dukungan ini terintegrasi dengan erat. Jika Anda menggunakan Amazon RDS, kami sangat menyarankan penggunaan AWS Secrets Manager dan rotasi kredensyal. Jika database Anda tidak menggunakan Amazon RDS, simpan kredensialnya sebagai JSON dalam format berikut:

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

Contoh string koneksi dengan nama rahasia

String berikut memiliki nama rahasia`${secret1_name}`.

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

Konektor menggunakan nama rahasia untuk mengambil rahasia dan memberikan nama pengguna dan kata sandi, seperti pada contoh berikut.

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

Menggunakan handler koneksi tunggal

Anda dapat menggunakan metadata koneksi tunggal berikut dan penangan rekaman untuk terhubung ke satu instance Azure Data Lake Storage Gen2.

Jenis handler	Kelas
Handler komposit	`DataLakeGen2CompositeHandler`
Penangan metadata	`DataLakeGen2MetadataHandler`
Rekam handler	`DataLakeGen2RecordHandler`

Parameter handler koneksi tunggal

Parameter	Deskripsi
`default`	Wajib. String koneksi default.

Penangan koneksi tunggal mendukung satu instance database dan harus menyediakan parameter string `default` koneksi. Semua string koneksi lainnya diabaikan.

Properti contoh berikut adalah untuk satu instance Azure Data Lake Storage Gen2 yang didukung oleh fungsi Lambda.

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

Parameter tumpahan

Lambda SDK dapat menumpahkan data ke Amazon S3. Semua instance database yang diakses oleh fungsi Lambda yang sama tumpah ke lokasi yang sama.

Parameter	Deskripsi
`spill_bucket`	Wajib. Nama ember tumpahan.
`spill_prefix`	Wajib. Tumpahkan key prefix bucket.
`spill_put_request_headers`	(Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject {"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.

Dukungan tipe data

Tabel berikut menunjukkan tipe data yang sesuai untuk ADLS Gen2 dan Arrow.

ADLS Gen2	Panah
bit	TINYINT
tinyint	SMALLINT
smallint	SMALLINT
int	INT
bigint	BIGINT
desimal	DECIMAL
numerik	FLOAT8
uang kecil	FLOAT8
money	DECIMAL
mengapung [24]	FLOAT4

ADLS Gen2	Panah
mengapung [53]	FLOAT8
real	FLOAT4
datetime	Tanggal (MILLISECOND)
tanggal2	Tanggal (MILLISECOND)
smalldatetime	Tanggal (MILLISECOND)
date	Tanggal (HARI)
Waktu	VARCHAR
datetimeoffset	Tanggal (MILLISECOND)
arang [n]	VARCHAR
varchar [n/maks]	VARCHAR

Partisi dan split

Azure Data Lake Storage Gen2 menggunakan penyimpanan gumpalan Gen2 yang kompatibel dengan Hadoop untuk menyimpan file data. Data dari file-file ini ditanyakan dari mesin Azure Synapse. Mesin Azure Synapse memperlakukan data Gen2 yang disimpan dalam sistem file sebagai tabel eksternal. Partisi diimplementasikan berdasarkan jenis data. Jika data telah dipartisi dan didistribusikan dalam sistem penyimpanan Gen 2, konektor mengambil data sebagai split tunggal.

Kinerja

Konektor Azure Data Lake Storage Gen2 menunjukkan kinerja kueri yang lebih lambat saat menjalankan beberapa kueri sekaligus, dan tunduk pada pembatasan.

Konektor Athena Azure Data Lake Storage Gen2 melakukan pushdown predikat untuk mengurangi data yang dipindai oleh kueri. Predikat sederhana dan ekspresi kompleks didorong ke konektor untuk mengurangi jumlah data yang dipindai dan mengurangi waktu eksekusi kueri.

Predikat

Predikat adalah ekspresi dalam `WHERE` klausa kueri SQL yang mengevaluasi nilai Boolean dan menyaring baris berdasarkan beberapa kondisi. Konektor Athena Azure Data Lake Storage Gen2 dapat menggabungkan ekspresi ini dan mendorongnya langsung ke Azure Data Lake Storage Gen2 untuk meningkatkan fungsionalitas dan untuk mengurangi jumlah data yang dipindai.

Operator konektor Gen2 Athena Azure Data Lake Storage berikut mendukung pushdown predikat:

- Boolean: DAN, ATAU, TIDAK
- KESETARAAN: SAMA, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, NULL_IF, IS_NULL
- Aritmatika: TAMBAHKAN, KURANGI, KALIKAN, BAGI, MODULUS, MENIADAKAN
- Lainnya: LIKE_PATTERN, IN

Contoh pushdown gabungan

Untuk kemampuan kueri yang ditingkatkan, gabungkan jenis pushdown, seperti pada contoh berikut:

```
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%');
```

Kueri passthrough

Konektor Azure Data Lake Storage Gen2 mendukung kueri passthrough. Kueri passthrough menggunakan fungsi tabel untuk mendorong kueri lengkap Anda ke sumber data untuk dieksekusi.

Untuk menggunakan kueri passthrough dengan Azure Data Lake Storage Gen2, Anda dapat menggunakan sintaks berikut:

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

Contoh kueri berikut mendorong kueri ke sumber data di Azure Data Lake Storage Gen2. Kueri memilih semua kolom dalam `customer` tabel, membatasi hasilnya menjadi 10.

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

Informasi lisensi

Dengan menggunakan konektor ini, Anda mengakui penyertaan komponen pihak ketiga, daftar yang dapat ditemukan dalam file pom.xml untuk konektor ini, dan menyetujui persyaratan dalam masing-masing lisensi pihak ketiga yang disediakan dalam file LICENSE.txt di .com. GitHub

Sumber daya tambahan

Untuk informasi versi driver JDBC terbaru, lihat file pom.xml untuk konektor Azure Data Lake Storage Gen2 di.com. GitHub

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Amazon Athena Azure Synapse

Konektor Amazon Athena untuk analitik Azure Synapse memungkinkan Amazon Athena menjalankan kueri SQL pada database Azure Synapse Anda menggunakan JDBC.

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Batasan

- Menulis operasi DDL tidak didukung.
- Dalam pengaturan multiplexer, bucket tumpahan dan awalan dibagikan di semua instance database.
- Batas Lambda yang relevan. Untuk informasi selengkapnya, lihat Kuota Lambda dalam Panduan Pengembang AWS Lambda .
- Dalam kondisi filter, Anda harus mentransmisikan tipe `Date` dan `Timestamp` data ke tipe data yang sesuai.

- Untuk mencari nilai negatif dari jenis `Real` dan`Float`, gunakan >= operator <= atau.
- Tipe `rowversion` data `binary` `varbinaryimage`,, dan tidak didukung.

Ketentuan

Istilah-istilah berikut berhubungan dengan konektor Synapse.

- Instans database — Instance apa pun dari database yang digunakan di tempat, di Amazon EC2, atau di Amazon RDS.
- Handler - Handler Lambda yang mengakses instance database Anda. Handler bisa untuk metadata atau untuk catatan data.
- Metadata handler — Penangan Lambda yang mengambil metadata dari instance database Anda.
- Record handler - Handler Lambda yang mengambil catatan data dari instance database Anda.
- Composite handler — Handler Lambda yang mengambil data metadata dan data dari instance database Anda.
- Properti atau parameter - Properti database yang digunakan oleh penangan untuk mengekstrak informasi database. Anda mengonfigurasi properti ini sebagai variabel lingkungan Lambda.
- Connection String — Sebuah string teks yang digunakan untuk membuat koneksi ke instance database.
- Katalog —AWS Glue Non-katalog yang terdaftar di Athena yang merupakan awalan yang diperlukan untuk properti. `connection_string`
- Multiplexing handler - Handler Lambda yang dapat menerima dan menggunakan beberapa koneksi database.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengkonfigurasi konektor Synapse.

String koneksi

Gunakan string koneksi JDBC dalam format berikut untuk terhubung ke instance database.

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

Menggunakan handler multiplexing

Anda dapat menggunakan multiplexer untuk terhubung ke beberapa instance database dengan satu fungsi Lambda. Permintaan dirutekan dengan nama katalog. Gunakan kelas-kelas berikut di Lambda.

Handler	Kelas
Handler komposit	`SynapseMuxCompositeHandler`
Penangan metadata	`SynapseMuxMetadataHandler`
Rekam handler	`SynapseMuxRecordHandler`

Parameter handler multiplexing

Parameter	Deskripsi
`$`*`catalog`*`_connection_string`	Wajib. Sebuah string koneksi instance database. Awalan variabel lingkungan dengan nama katalog yang digunakan di Athena. Misalnya, jika katalog terdaftar di Athena adalah`mysynapse catalog` , maka nama variabel lingkungan adalah. `mysynapse catalog_connection_string`
`default`	Wajib. String koneksi default. String ini digunakan ketika katalog adalah `lambda:${` *`}AWS_LAMBDA_FUNCTION_NAME`* .

Contoh properti berikut adalah untuk fungsi Synapse MUX Lambda yang mendukung dua instance database`synapse1`: (default), dan. `synapse2`

Properti	Nilai
`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`*`}`

Properti	Nilai
onnection _string	
synapse_c atalog2_c onnection _string	synapse://jdbc:synapse://synapse2.hostname:po rt;databaseName= *<database_name>* ;${*secret2_name* }

Memberikan kredensi

Untuk memberikan nama pengguna dan kata sandi untuk database Anda dalam string koneksi JDBC Anda, Anda dapat menggunakan properti string koneksi atau. AWS Secrets Manager

- Connection String - Nama pengguna dan kata sandi dapat ditentukan sebagai properti dalam string koneksi JDBC.

  > **Important**
  >
  > Sebagai praktik keamanan terbaik, jangan gunakan kredensyal hardcode dalam variabel lingkungan atau string koneksi Anda. Untuk informasi tentang memindahkan rahasia hardcode Anda AWS Secrets Manager, lihat Memindahkan rahasia hardcode ke AWS Secrets Manager dalam Panduan Pengguna.AWS Secrets Manager

- AWS Secrets Manager— Untuk menggunakan fitur Query Federasi Athena dengan, VPC AWS Secrets Manager yang terhubung ke fungsi Lambda Anda harus memiliki akses internet atau titik akhir VPC untuk terhubung ke Secrets Manager.

  Anda dapat memasukkan nama rahasia ke AWS Secrets Manager dalam string koneksi JDBC Anda. Konektor menggantikan nama rahasia dengan `password` nilai `username` dan dari Secrets Manager.

  Untuk instans database Amazon RDS, dukungan ini terintegrasi dengan erat. Jika Anda menggunakan Amazon RDS, kami sangat menyarankan penggunaan AWS Secrets Manager dan rotasi kredensyal. Jika database Anda tidak menggunakan Amazon RDS, simpan kredensialnya sebagai JSON dalam format berikut:

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

Contoh string koneksi dengan nama rahasia

String berikut memiliki nama rahasia $ {secret_name}.

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

Konektor menggunakan nama rahasia untuk mengambil rahasia dan memberikan nama pengguna dan kata sandi, seperti pada contoh berikut.

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

Menggunakan handler koneksi tunggal

Anda dapat menggunakan metadata koneksi tunggal berikut dan penangan rekaman untuk terhubung ke satu instance Synapse.

Jenis handler	Kelas
Handler komposit	`SynapseCompositeHandler`
Penangan metadata	`SynapseMetadataHandler`
Rekam handler	`SynapseRecordHandler`

Parameter handler koneksi tunggal

Parameter	Deskripsi
`default`	Wajib. String koneksi default.

Penangan koneksi tunggal mendukung satu instance database dan harus menyediakan parameter string `default` koneksi. Semua string koneksi lainnya diabaikan.

Properti contoh berikut adalah untuk instance Synapse tunggal yang didukung oleh fungsi Lambda.

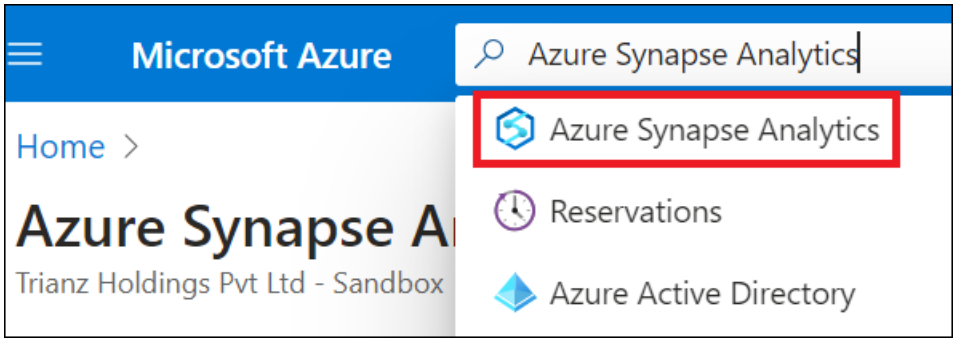
Prope	Nilai
defa	synapse://jdbc:sqlserver://hostname:port;databaseName= *<database_name>* ;${*secret_name* }

Mengkonfigurasi otentikasi Direktori Aktif

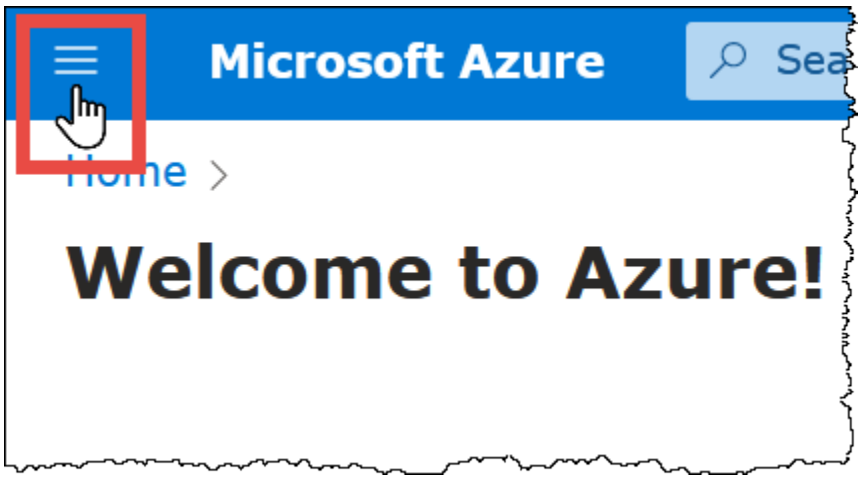
Konektor Amazon Athena Azure Synapse mendukung Otentikasi Direktori Aktif Microsoft. Sebelum Anda mulai, Anda harus mengkonfigurasi pengguna administratif di portal Microsoft Azure dan kemudian gunakan AWS Secrets Manager untuk membuat rahasia.

Untuk mengatur pengguna administratif Direktori Aktif

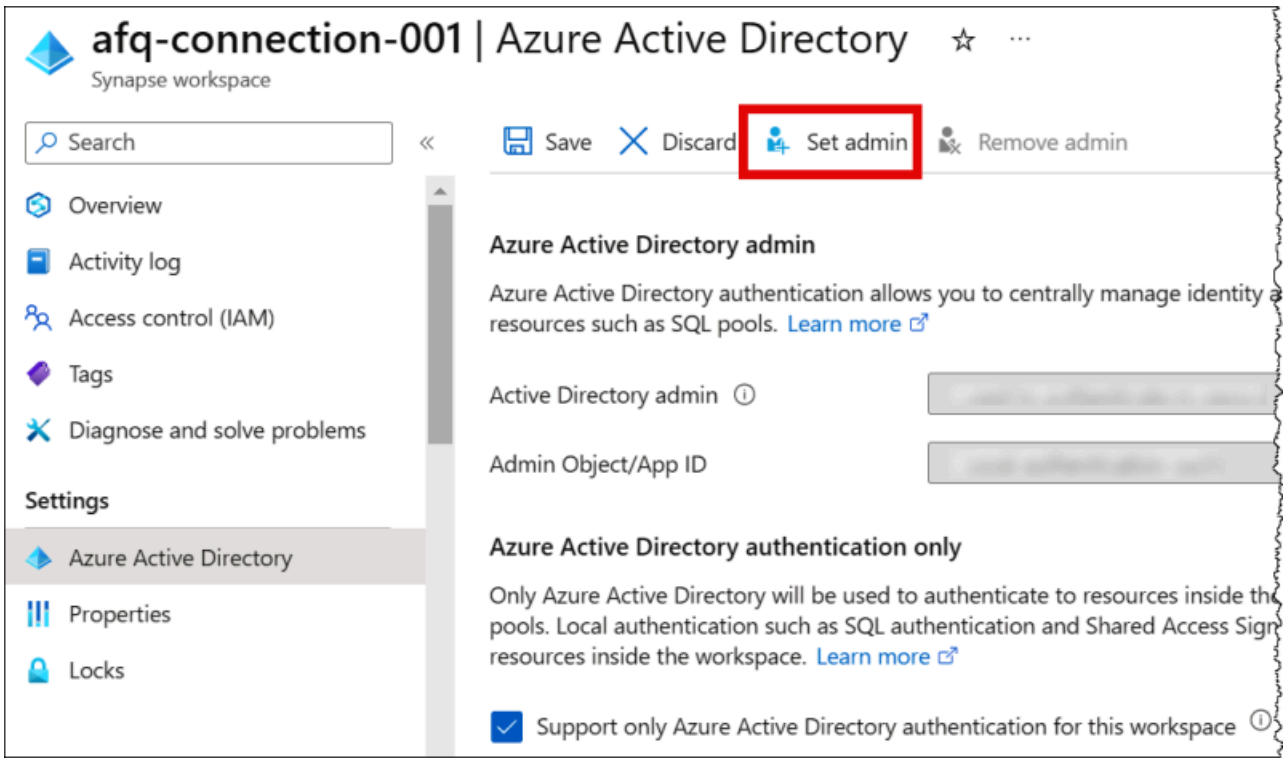
1. Menggunakan akun yang memiliki hak administratif, masuk ke portal Microsoft Azure di https://portal.azure.com/.
2. Di kotak pencarian, masukkan Azure Synapse Analytics, lalu pilih Azure Synapse Analytics.



3. Buka menu di sebelah kiri.



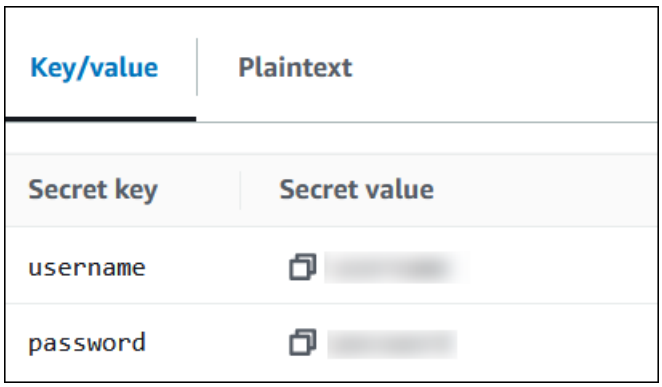
4. Di panel navigasi, pilih Azure Active Directory.
5. Pada tab Set admin, atur admin Active Directory ke pengguna baru atau yang sudah ada.



6. Di AWS Secrets Manager, simpan nama pengguna admin dan kredensyal kata sandi. Untuk informasi tentang cara membuat rahasia di Secrets Manager, lihat Membuat AWS Secrets Manager rahasia.

Untuk melihat rahasia Anda di Secrets Manager

1. Buka konsol Secrets Manager di https://console.aws.amazon.com/secretsmanager/.
2. Di panel navigasi, pilih Rahasia.
3. Pada halaman Rahasia, pilih tautan ke rahasia Anda.
4. Pada halaman detail untuk rahasia Anda, pilih Ambil nilai rahasia.



Memodifikasi string koneksi

Untuk mengaktifkan Otentikasi Direktori Aktif untuk konektor, ubah string koneksi menggunakan sintaks berikut:

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

Menggunakan ActiveDirectoryServicePrincipal

Konektor Amazon Athena Azure Synapse juga mendukung. `ActiveDirectoryServicePrincipal` Untuk mengaktifkan ini, ubah string koneksi sebagai berikut.

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

Untuk`secret_name`, tentukan aplikasi atau ID klien sebagai nama pengguna dan rahasia identitas utama layanan dalam kata sandi.

Parameter tumpahan

Lambda SDK dapat menumpahkan data ke Amazon S3. Semua instance database yang diakses oleh fungsi Lambda yang sama tumpah ke lokasi yang sama.

Parameter	Deskripsi
`spill_bucket`	Wajib. Nama ember tumpahan.

Parameter	Deskripsi
`spill_prefix`	Wajib. Tumpahkan key prefix bucket.
`spill_put_request_headers`	(Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject {"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.

Dukungan tipe data

Tabel berikut menunjukkan tipe data yang sesuai untuk Synapse dan Apache Arrow.

Sinaps	Panah
bit	TINYINT
tinyint	SMALLINT
smallint	SMALLINT
int	INT
bigint	BIGINT
desimal	DECIMAL
numerik	FLOAT8
uang kecil	FLOAT8
money	DECIMAL
mengapung [24]	FLOAT4
mengapung [53]	FLOAT8
real	FLOAT4

Sinaps	Panah
datetime	Tanggal (MILLISECOND)
tanggal2	Tanggal (MILLISECOND)
smalldatetime	Tanggal (MILLISECOND)
date	Tanggal (HARI)
Waktu	VARCHAR
datetimeoffset	Tanggal (MILLISECOND)
arang [n]	VARCHAR
varchar [n/maks]	VARCHAR
nchar [n]	VARCHAR
nvarchar [n/max]	VARCHAR

Partisi dan perpecahan

Partisi diwakili oleh kolom partisi tunggal tipe`varchar`. Synapse mendukung partisi rentang, sehingga partisi diimplementasikan dengan mengekstrak kolom partisi dan rentang partisi dari tabel metadata Synapse. Nilai rentang ini digunakan untuk membuat split.

Kinerja

Memilih subset kolom secara signifikan memperlambat runtime kueri. Konektor menunjukkan pelambatan yang signifikan karena konkurensi.

Konektor Athena Synapse melakukan pushdown predikat untuk mengurangi data yang dipindai oleh kueri. Predikat sederhana dan ekspresi kompleks didorong ke konektor untuk mengurangi jumlah data yang dipindai dan mengurangi waktu eksekusi kueri.

Predikat

Predikat adalah ekspresi dalam `WHERE` klausa kueri SQL yang mengevaluasi nilai Boolean dan menyaring baris berdasarkan beberapa kondisi. Konektor Athena Synapse dapat menggabungkan

ekspresi ini dan mendorongnya langsung ke Synapse untuk meningkatkan fungsionalitas dan untuk mengurangi jumlah data yang dipindai.

Operator konektor Athena Synapse berikut mendukung pushdown predikat:

- Boolean: DAN, ATAU, TIDAK
- KESETARAAN: SAMA, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, NULL_IF, IS_NULL
- Aritmatika: TAMBAHKAN, KURANGI, KALIKAN, BAGI, MODULUS, MENIADAKAN
- Lainnya: LIKE_PATTERN, IN

Contoh pushdown gabungan

Untuk kemampuan kueri yang ditingkatkan, gabungkan jenis pushdown, seperti pada contoh berikut:

```
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%');
```

Kueri passthrough

Konektor Synapse mendukung kueri passthrough. Kueri passthrough menggunakan fungsi tabel untuk mendorong kueri lengkap Anda ke sumber data untuk dieksekusi.

Untuk menggunakan kueri passthrough dengan Synapse, Anda dapat menggunakan sintaks berikut:

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

Contoh query berikut mendorong ke bawah query ke sumber data di Synapse. Kueri memilih semua kolom dalam `customer` tabel, membatasi hasilnya menjadi 10.

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

Informasi lisensi

Dengan menggunakan konektor ini, Anda mengakui penyertaan komponen pihak ketiga, daftar yang dapat ditemukan dalam file pom.xml untuk konektor ini, dan menyetujui persyaratan dalam masing-masing lisensi pihak ketiga yang disediakan dalam file LICENSE.txt di .com. GitHub

Sumber daya tambahan

- Untuk artikel yang menunjukkan cara menggunakan Kueri Federasi Amazon QuickSight dan Amazon Athena untuk membuat dasbor dan visualisasi pada data yang disimpan dalam database Microsoft Azure Synapse, lihat Melakukan analitik multi-cloud menggunakan Amazon, QuickSight Amazon Athena Federated Query, dan Microsoft Azure Synapse di Blog Big Data.AWS
- Untuk informasi versi driver JDBC terbaru, lihat file pom.xml untuk konektor Synapse di.com. GitHub
- Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Amazon Athena Cloudera Hive

Konektor Amazon Athena untuk Cloudera Hive memungkinkan Athena menjalankan kueri SQL pada distribusi Cloudera Hive Hadoop. Konektor mengubah kueri Athena SQL Anda menjadi sintaks HiveQL yang setara.

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.
- Siapkan VPC dan grup keamanan sebelum Anda menggunakan konektor ini. Untuk informasi selengkapnya, lihat Membuat VPC untuk konektor sumber data.

Batasan

- Menulis operasi DDL tidak didukung.
- Dalam pengaturan multiplexer, bucket tumpahan dan awalan dibagikan di semua instance database.
- Batas Lambda yang relevan. Untuk informasi selengkapnya, lihat Kuota Lambda dalam Panduan Pengembang AWS Lambda .

Ketentuan

Istilah-istilah berikut berhubungan dengan konektor Cloudera Hive.

- Instans database — Instance apa pun dari database yang digunakan di tempat, di Amazon EC2, atau di Amazon RDS.
- Handler - Handler Lambda yang mengakses instance database Anda. Handler bisa untuk metadata atau untuk catatan data.
- Metadata handler — Penangan Lambda yang mengambil metadata dari instance database Anda.
- Record handler - Handler Lambda yang mengambil catatan data dari instance database Anda.
- Composite handler — Handler Lambda yang mengambil data metadata dan data dari instance database Anda.
- Properti atau parameter - Properti database yang digunakan oleh penangan untuk mengekstrak informasi database. Anda mengonfigurasi properti ini sebagai variabel lingkungan Lambda.
- Connection String — Sebuah string teks yang digunakan untuk membuat koneksi ke instance database.
- Katalog —AWS Glue Non-katalog yang terdaftar di Athena yang merupakan awalan yang diperlukan untuk properti. `connection_string`
- Multiplexing handler - Handler Lambda yang dapat menerima dan menggunakan beberapa koneksi database.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengonfigurasi konektor Cloudera Hive.

String koneksi

Gunakan string koneksi JDBC dalam format berikut untuk terhubung ke instance database.

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

Menggunakan handler multiplexing

Anda dapat menggunakan multiplexer untuk terhubung ke beberapa instance database dengan satu fungsi Lambda. Permintaan dirutekan dengan nama katalog. Gunakan kelas berikut di Lambda.

Handler	Kelas
Pawang komposit	`HiveMuxCompositeHandler`
Penangan metadata	`HiveMuxMetadataHandler`
Rekam handler	`HiveMuxRecordHandler`

Parameter handler multiplexing

Parameter	Deskripsi
`$`*`catalog`*`_connection_string`	Wajib. Sebuah string koneksi instance database. Awalan variabel lingkungan dengan nama katalog yang digunakan di Athena. Misalnya, jika katalog terdaftar di Athena adalah`myhivecatalog` , maka nama variabel lingkungan adalah. `myhivecatalog_connection_string`
`default`	Wajib. String koneksi default. String ini digunakan ketika katalog `lambda:${` *`}AWS_LAMBDA_FUNCTION_NAME`* .

Contoh properti berikut adalah untuk fungsi Hive MUX Lambda yang mendukung dua instance database: `hive1` (default), dan. `hive2`

Properti	Nilai
`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`

Memberikan kredensi

Untuk memberikan nama pengguna dan kata sandi untuk database Anda dalam string koneksi JDBC Anda, Anda dapat menggunakan properti string koneksi atau. AWS Secrets Manager

- Connection String - Nama pengguna dan kata sandi dapat ditentukan sebagai properti dalam string koneksi JDBC.

  > **Important**
  >
  > Sebagai praktik keamanan terbaik, jangan gunakan kredensyal hardcode dalam variabel lingkungan atau string koneksi Anda. Untuk informasi tentang memindahkan rahasia hardcode Anda AWS Secrets Manager, lihat Memindahkan rahasia hardcode ke AWS Secrets Manager dalam Panduan Pengguna.AWS Secrets Manager

- AWS Secrets Manager— Untuk menggunakan fitur Query Federasi Athena dengan, VPC AWS Secrets Manager yang terhubung ke fungsi Lambda Anda harus memiliki akses internet atau titik akhir VPC untuk terhubung ke Secrets Manager.

  Anda dapat memasukkan nama rahasia ke AWS Secrets Manager dalam string koneksi JDBC Anda. Konektor menggantikan nama rahasia dengan `password` nilai `username` dan dari Secrets Manager.

  Untuk instans database Amazon RDS, dukungan ini terintegrasi dengan erat. Jika Anda menggunakan Amazon RDS, kami sangat menyarankan penggunaan AWS Secrets Manager dan rotasi kredensyal. Jika database Anda tidak menggunakan Amazon RDS, simpan kredensialnya sebagai JSON dalam format berikut:

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

Contoh string koneksi dengan nama rahasia

String berikut memiliki nama rahasia`${Test/RDS/hive1}`.

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

Konektor menggunakan nama rahasia untuk mengambil rahasia dan memberikan nama pengguna dan kata sandi, seperti pada contoh berikut.

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

Saat ini, konektor Cloudera Hive mengenali properti `UID` dan `PWD` JDBC.

Menggunakan handler koneksi tunggal

Anda dapat menggunakan metadata koneksi tunggal berikut dan penangan rekaman untuk terhubung ke satu instance Cloudera Hive.

Jenis handler	Kelas
Pawang komposit	`HiveCompositeHandler`
Penangan metadata	`HiveMetadataHandler`
Rekam handler	`HiveRecordHandler`

Parameter handler koneksi tunggal

Parameter	Deskripsi
`default`	Wajib. String koneksi default.

Penangan koneksi tunggal mendukung satu instance database dan harus menyediakan parameter string `default` koneksi. Semua string koneksi lainnya diabaikan.

Properti contoh berikut adalah untuk instance Cloudera Hive tunggal yang didukung oleh fungsi Lambda.

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

Parameter tumpahan

Lambda SDK dapat menumpahkan data ke Amazon S3. Semua instance database yang diakses oleh fungsi Lambda yang sama tumpah ke lokasi yang sama.

Parameter	Deskripsi
`spill_bucket`	Wajib. Nama ember tumpahan.
`spill_prefix`	Wajib. Tumpahkan key prefix bucket.
`spill_put_request_headers`	(Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject {"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.

Dukungan tipe data

Tabel berikut menunjukkan tipe data yang sesuai untuk JDBC, Cloudera Hive, dan Arrow.

JDBC	Sarang Cloudera	Panah
Boolean	Boolean	Bit
Bilangan Bulat	TINYINT	Kecil
Pendek	SMALLINT	Kecil
Bilangan Bulat	INT	Int
Long	BIGINT	Bigint
float	mengapung4	Mengapung4
Ganda	mengapung8	Mengapung8
Tanggal	tanggal	DateDay

JDBC	Sarang Cloudera	Panah
Stempel Waktu	timestamp	DateMilli
String	VARCHAR	Varchar
Byte	byte	Varbiner
BigDecimal	Decimal	Decimal
ARRAY	N/A (lihat catatan)	Daftar

> **Note**
>
> Saat ini, Cloudera Hive tidak mendukung jenis agregat`ARRAY`,,`MAP`, `STRUCT` atau. `UNIONTYPE` Kolom tipe agregat diperlakukan sebagai `VARCHAR` kolom dalam SQL.

Partisi dan split

Partisi digunakan untuk menentukan cara menghasilkan split untuk konektor. Athena membangun kolom sintetis tipe `varchar` yang mewakili skema partisi untuk tabel untuk membantu konektor menghasilkan split. Konektor tidak mengubah definisi tabel yang sebenarnya.

Kinerja

Cloudera Hive mendukung partisi statis. Konektor Athena Cloudera Hive dapat mengambil data dari partisi ini secara paralel. Jika Anda ingin menanyakan kumpulan data yang sangat besar dengan distribusi partisi yang seragam, partisi statis sangat disarankan. Konektor Cloudera Hive tahan terhadap pelambatan karena konkurensi.

Konektor Athena Cloudera Hive melakukan pushdown predikat untuk mengurangi data yang dipindai oleh kueri. `LIMIT`klausa, predikat sederhana, dan ekspresi kompleks didorong ke konektor untuk mengurangi jumlah data yang dipindai dan mengurangi waktu eksekusi kueri.

Klausul LIMIT

`LIMIT N`Pernyataan mengurangi data yang dipindai oleh kueri. Dengan `LIMIT N` pushdown, konektor hanya mengembalikan `N` baris ke Athena.

Predikat

Predikat adalah ekspresi dalam `WHERE` klausa kueri SQL yang mengevaluasi nilai Boolean dan menyaring baris berdasarkan beberapa kondisi. Konektor Athena Cloudera Hive dapat menggabungkan ekspresi ini dan mendorongnya langsung ke Cloudera Hive untuk meningkatkan fungsionalitas dan untuk mengurangi jumlah data yang dipindai.

Operator konektor Athena Cloudera Hive berikut mendukung pushdown predikat:

- Boolean: DAN, ATAU, TIDAK
- KESETARAAN: SAMA, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_NULL
- Aritmatika: TAMBAHKAN, KURANGI, KALIKAN, BAGI, MODULUS, MENIADAKAN
- Lainnya: LIKE_PATTERN, IN

Contoh pushdown gabungan

Untuk kemampuan kueri yang ditingkatkan, gabungkan jenis pushdown, seperti pada contoh berikut:

```
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;
```

Kueri passthrough

Konektor Cloudera Hive mendukung kueri passthrough. Kueri passthrough menggunakan fungsi tabel untuk mendorong kueri lengkap Anda ke sumber data untuk dieksekusi.

Untuk menggunakan kueri passthrough dengan Cloudera Hive, Anda dapat menggunakan sintaks berikut:

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

Contoh query berikut mendorong ke bawah query ke sumber data di Cloudera Hive. Kueri memilih semua kolom dalam `customer` tabel, membatasi hasilnya menjadi 10.

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

Informasi lisensi

Dengan menggunakan konektor ini, Anda mengakui penyertaan komponen pihak ketiga, daftar yang dapat ditemukan dalam file pom.xml untuk konektor ini, dan menyetujui persyaratan dalam masing-masing lisensi pihak ketiga yang disediakan dalam file LICENSE.txt di .com. GitHub

Sumber daya tambahan

Untuk informasi versi driver JDBC terbaru, lihat file pom.xml untuk konektor Cloudera Hive di.com. GitHub

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Amazon Athena Cloudera Impala

Konektor Amazon Athena Cloudera Impala memungkinkan Athena menjalankan kueri SQL pada distribusi Cloudera Impala. Konektor mengubah kueri Athena SQL Anda ke sintaks Impala yang setara.

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.
- Siapkan VPC dan grup keamanan sebelum Anda menggunakan konektor ini. Untuk informasi selengkapnya, lihat Membuat VPC untuk konektor sumber data.

Batasan

- Menulis operasi DDL tidak didukung.

- Dalam pengaturan multiplexer, bucket tumpahan dan awalan dibagikan di semua instance database.
- Batas Lambda yang relevan. Untuk informasi selengkapnya, lihat Kuota Lambda dalam Panduan Pengembang AWS Lambda .

Ketentuan

Istilah-istilah berikut berhubungan dengan konektor Cloudera Impala.

- Instans database — Instance apa pun dari database yang digunakan di tempat, di Amazon EC2, atau di Amazon RDS.
- Handler - Handler Lambda yang mengakses instance database Anda. Handler bisa untuk metadata atau untuk catatan data.
- Metadata handler — Penangan Lambda yang mengambil metadata dari instance database Anda.
- Record handler - Handler Lambda yang mengambil catatan data dari instance database Anda.
- Composite handler — Handler Lambda yang mengambil data metadata dan data dari instance database Anda.
- Properti atau parameter - Properti database yang digunakan oleh penangan untuk mengekstrak informasi database. Anda mengonfigurasi properti ini sebagai variabel lingkungan Lambda.
- Connection String — Sebuah string teks yang digunakan untuk membuat koneksi ke instance database.
- Katalog —AWS Glue Non-katalog yang terdaftar di Athena yang merupakan awalan yang diperlukan untuk properti. `connection_string`
- Multiplexing handler - Handler Lambda yang dapat menerima dan menggunakan beberapa koneksi database.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengkonfigurasi konektor Cloudera Impala.

String koneksi

Gunakan string koneksi JDBC dalam format berikut untuk terhubung ke cluster Impala.

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

Menggunakan handler multiplexing

Anda dapat menggunakan multiplexer untuk terhubung ke beberapa instance database dengan satu fungsi Lambda. Permintaan dirutekan dengan nama katalog. Gunakan kelas berikut di Lambda.

Handler	Kelas
Pawang komposit	`ImpalaMuxCompositeHandler`
Penangan metadata	`ImpalaMuxMetadataHandler`
Rekam handler	`ImpalaMuxRecordHandler`

Parameter handler multiplexing

Parameter	Deskripsi
`$`*`catalog`*`_connection_string`	Wajib. String koneksi cluster Impala untuk katalog Athena. Awalan variabel lingkungan dengan nama katalog yang digunakan di Athena. Misalnya, jika katalog terdaftar di Athena adalah`myimpalacatalog` , maka nama variabel lingkungan adalah. `myimpalacatalog_connection_string`
`default`	Wajib. String koneksi default. String ini digunakan ketika katalog adalah `lambda:${` *`}AWS_LAMBDA_FUNCTION_NAME`* .

Contoh properti berikut adalah untuk fungsi Impala MUX Lambda yang mendukung dua instance database: `impala1` (default), dan. `impala2`

Properti	Nilai
`default`	`impala://jdbc:impala://some.impala.host.name:21050/?${Test/impala1}`
`impala_catalog1_connection_string`	`impala://jdbc:impala://someother.impala.host.name:21050/?${Test/impala1}`

Properti	Nilai
`impala_catalog2_connection_string`	`impala://jdbc:impala://another.impala.host.name:21050/?UID=sample&PWD=sample`

Memberikan kredensi

Untuk memberikan nama pengguna dan kata sandi untuk database Anda dalam string koneksi JDBC Anda, Anda dapat menggunakan properti string koneksi atau. AWS Secrets Manager

- Connection String - Nama pengguna dan kata sandi dapat ditentukan sebagai properti dalam string koneksi JDBC.

  > **Important**
  >
  > Sebagai praktik keamanan terbaik, jangan gunakan kredensyal hardcode dalam variabel lingkungan atau string koneksi Anda. Untuk informasi tentang memindahkan rahasia hardcode Anda AWS Secrets Manager, lihat Memindahkan rahasia hardcode ke AWS Secrets Manager dalam Panduan Pengguna.AWS Secrets Manager

- AWS Secrets Manager— Untuk menggunakan fitur Query Federasi Athena dengan, VPC AWS Secrets Manager yang terhubung ke fungsi Lambda Anda harus memiliki akses internet atau titik akhir VPC untuk terhubung ke Secrets Manager.

  Anda dapat memasukkan nama rahasia ke AWS Secrets Manager dalam string koneksi JDBC Anda. Konektor menggantikan nama rahasia dengan `password` nilai `username` dan dari Secrets Manager.

  Untuk instans database Amazon RDS, dukungan ini terintegrasi erat. Jika Anda menggunakan Amazon RDS, kami sangat menyarankan penggunaan AWS Secrets Manager dan rotasi kredensyal. Jika database Anda tidak menggunakan Amazon RDS, simpan kredensialnya sebagai JSON dalam format berikut:

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

Contoh string koneksi dengan nama rahasia

String berikut memiliki nama rahasia`${Test/impala1host}`.

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

Konektor menggunakan nama rahasia untuk mengambil rahasia dan memberikan nama pengguna dan kata sandi, seperti pada contoh berikut.

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

Saat ini, Cloudera Impala mengenali properti `UID` dan `PWD` JDBC.

Menggunakan handler koneksi tunggal

Anda dapat menggunakan metadata koneksi tunggal berikut dan penangan rekaman untuk terhubung ke satu instance Cloudera Impala.

Jenis handler	Kelas
Pawang komposit	`ImpalaCompositeHandler`
Penangan metadata	`ImpalaMetadataHandler`
Rekam handler	`ImpalaRecordHandler`

Parameter handler koneksi tunggal

Parameter	Deskripsi
`default`	Wajib. String koneksi default.

Penangan koneksi tunggal mendukung satu instance database dan harus menyediakan parameter string `default` koneksi. Semua string koneksi lainnya diabaikan.

Properti contoh berikut adalah untuk instance Cloudera Impala tunggal yang didukung oleh fungsi Lambda.

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

Parameter tumpahan

Lambda SDK dapat menumpahkan data ke Amazon S3. Semua instance database yang diakses oleh fungsi Lambda yang sama tumpah ke lokasi yang sama.

Parameter	Deskripsi
`spill_bucket`	Wajib. Nama ember tumpahan.
`spill_prefix`	Wajib. Tumpahkan key prefix bucket.
`spill_put_request_headers`	(Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject {"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.

Dukungan tipe data

Tabel berikut menunjukkan tipe data yang sesuai untuk JDBC, Cloudera Impala, dan Arrow.

JDBC	Cloudera Impala	Panah
Boolean	Boolean	Bit
Bilangan Bulat	TINYINT	Mungil
Pendek	SMALLINT	Orang kecil
Bilangan Bulat	INT	Int

JDBC	Cloudera Impala	Panah
Long	BIGINT	Bigint
float	mengapung4	Mengapung4
Ganda	mengapung8	Mengapung8
Tanggal	tanggal	DateDay
Stempel Waktu	timestamp	DateMilli
String	VARCHAR	Varchar
Byte	byte	Varbiner
BigDecimal	Decimal	Decimal
ARRAY	N/A (lihat catatan)	Daftar

Note

Saat ini, Cloudera Impala tidak mendukung jenis agregat`ARRAY`,,`MAP`, `STRUCT` atau. `UNIONTYPE` Kolom tipe agregat diperlakukan sebagai `VARCHAR` kolom dalam SQL.

Partisi dan split

Partisi digunakan untuk menentukan cara menghasilkan split untuk konektor. Athena membangun kolom sintetis tipe `varchar` yang mewakili skema partisi untuk tabel untuk membantu konektor menghasilkan split. Konektor tidak mengubah definisi tabel yang sebenarnya.

Kinerja

Cloudera Impala mendukung partisi statis. Konektor Athena Cloudera Impala dapat mengambil data dari partisi ini secara paralel. Jika Anda ingin menanyakan kumpulan data yang sangat besar dengan distribusi partisi yang seragam, partisi statis sangat disarankan. Konektor Cloudera Impala tahan terhadap pelambatan karena konkurensi.

Konektor Athena Cloudera Impala melakukan pushdown predikat untuk mengurangi data yang dipindai oleh kueri. `LIMIT`klausa, predikat sederhana, dan ekspresi kompleks didorong ke konektor untuk mengurangi jumlah data yang dipindai dan mengurangi waktu eksekusi kueri.

Klausul LIMIT

`LIMIT N`Pernyataan mengurangi data yang dipindai oleh kueri. Dengan `LIMIT N` pushdown, konektor hanya mengembalikan `N` baris ke Athena.

Predikat

Predikat adalah ekspresi dalam `WHERE` klausa kueri SQL yang mengevaluasi nilai Boolean dan menyaring baris berdasarkan beberapa kondisi. Konektor Athena Cloudera Impala dapat menggabungkan ekspresi ini dan mendorongnya langsung ke Cloudera Impala untuk meningkatkan fungsionalitas dan untuk mengurangi jumlah data yang dipindai.

Operator konektor Athena Cloudera Impala berikut mendukung pushdown predikat:

- Boolean: DAN, ATAU, TIDAK
- KESETARAAN: SAMA, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Aritmatika: TAMBAHKAN, KURANGI, KALIKAN, BAGI, MODULUS, MENIADAKAN
- Lainnya: LIKE_PATTERN, IN

Contoh pushdown gabungan

Untuk kemampuan kueri yang ditingkatkan, gabungkan jenis pushdown, seperti pada contoh berikut:

```
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;
```

Kueri passthrough

Konektor Cloudera Impala mendukung kueri passthrough. Kueri passthrough menggunakan fungsi tabel untuk mendorong kueri lengkap Anda ke sumber data untuk dieksekusi.

Untuk menggunakan kueri passthrough dengan Cloudera Impala, Anda dapat menggunakan sintaks berikut:

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

Contoh query berikut mendorong ke bawah query ke sumber data di Cloudera Impala. Kueri memilih semua kolom dalam `customer` tabel, membatasi hasilnya menjadi 10.

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

Informasi lisensi

Dengan menggunakan konektor ini, Anda mengakui penyertaan komponen pihak ketiga, daftar yang dapat ditemukan dalam file pom.xml untuk konektor ini, dan menyetujui persyaratan dalam masing-masing lisensi pihak ketiga yang disediakan dalam file LICENSE.txt di .com. GitHub

Sumber daya tambahan

Untuk informasi versi driver JDBC terbaru, lihat file pom.xml untuk konektor Cloudera Impala di.com. GitHub

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Amazon Athena CloudWatch

CloudWatch Konektor Amazon Athena memungkinkan Amazon Athena untuk berkomunikasi sehingga Anda dapat menanyakan data log Anda CloudWatch dengan SQL.

Konektor memetakan skema LogGroups sebagai Anda dan masing-masing LogStream sebagai tabel. Konektor juga memetakan `all_log_streams` tampilan khusus yang berisi semua yang LogStreams ada di file LogGroup. Tampilan ini memungkinkan Anda untuk menanyakan semua log LogGroup sekaligus alih-alih mencari melalui masing-masing LogStream secara individual.

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengonfigurasi konektor. CloudWatch

- spill_bucket - Menentukan bucket Amazon S3 untuk data yang melebihi batas fungsi Lambda.
- spill_prefix — (Opsional) Default ke subfolder dalam nama yang ditentukan. `spill_bucket` `athena-federation-spill` Kami menyarankan Anda mengonfigurasi siklus hidup penyimpanan Amazon S3 di lokasi ini untuk menghapus tumpahan yang lebih lama dari jumlah hari atau jam yang telah ditentukan sebelumnya.
- spill_put_request_headers — (Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject` `{"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.
- kms_key_id — (Opsional) Secara default, data apa pun yang tumpah ke Amazon S3 dienkripsi menggunakan mode enkripsi yang diautentikasi AES-GCM dan kunci yang dihasilkan secara acak. Agar fungsi Lambda Anda menggunakan kunci enkripsi yang lebih kuat yang dihasilkan oleh KMS sepertia`7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, Anda dapat menentukan ID kunci KMS.
- disable_spill_encryption — (Opsional) Ketika diatur ke, menonaktifkan enkripsi tumpahan. `True` Defaultnya `False` sehingga data yang tumpah ke S3 dienkripsi menggunakan AES-GCM — baik menggunakan kunci yang dihasilkan secara acak atau KMS untuk menghasilkan kunci. Menonaktifkan enkripsi tumpahan dapat meningkatkan kinerja, terutama jika lokasi tumpahan Anda menggunakan enkripsi sisi server.

Konektor ini juga mendukung kontrol kemacetan AIMD untuk menangani peristiwa pelambatan dari melalui konstruksi CloudWatch Amazon Athena Query Federation SDK. `ThrottlingInvoker` Anda dapat mengubah perilaku pelambatan default dengan menyetel salah satu variabel lingkungan opsional berikut:

- throttle_initial_delay_ms - Penundaan panggilan awal diterapkan setelah peristiwa kemacetan pertama. Defaultnya adalah 10 milidetik.
- throttle_max_delay_ms — Penundaan maksimum antara panggilan. Anda dapat memperoleh TPS dengan membaginya menjadi 1000ms. Defaultnya adalah 1000 milidetik.
- throttle_decrease_factor — Faktor dimana Athena mengurangi tingkat panggilan. Defaultnya adalah 0,5
- throttle_increase_ms — Tingkat di mana Athena mengurangi penundaan panggilan. Defaultnya adalah 10 milidetik.

Database dan tabel

CloudWatch Konektor Athena memetakan skema LogGroups sebagai Anda (yaitu, database) dan masing-masing LogStream sebagai tabel. Konektor juga memetakan `all_log_streams` tampilan khusus yang berisi semua yang LogStreams ada di file LogGroup. Tampilan ini memungkinkan Anda untuk menanyakan semua log LogGroup sekaligus alih-alih mencari melalui masing-masing LogStream secara individual.

Setiap tabel yang dipetakan oleh konektor CloudWatch Athena memiliki skema berikut. Skema ini cocok dengan bidang yang disediakan oleh CloudWatch Log.

- log_stream — Sebuah `VARCHAR` yang berisi nama dari LogStream mana baris berasal.
- waktu — An `INT64` yang berisi waktu epoch ketika baris log dihasilkan.
- message — A `VARCHAR` yang berisi pesan log.

Contoh

Contoh berikut menunjukkan bagaimana melakukan `SELECT` query pada tertentu LogStream.

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

Contoh berikut menunjukkan bagaimana menggunakan `all_log_streams` tampilan untuk melakukan query pada semua LogStreams dalam tertentu LogGroup.

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

```
LIMIT 100
```

Izin yang Diperlukan

Untuk detail lengkap tentang kebijakan IAM yang diperlukan konektor ini, tinjau `Policies` bagian file athena-cloudwatch.yaml. Daftar berikut merangkum izin yang diperlukan.

- Akses tulis Amazon S3 - Konektor memerlukan akses tulis ke lokasi di Amazon S3 untuk menumpahkan hasil dari kueri besar.
- Athena GetQueryExecution — Konektor menggunakan izin ini untuk gagal cepat ketika kueri Athena hulu telah dihentikan.
- CloudWatch Log Baca/Tulis — Konektor menggunakan izin ini untuk membaca data log Anda dan menulis log diagnostiknya.

Kinerja

CloudWatch Konektor Athena mencoba mengoptimalkan kueri CloudWatch dengan memparalelkan pemindaian aliran log yang diperlukan untuk kueri Anda. Untuk filter periode waktu tertentu, pushdown predikat dilakukan baik di dalam fungsi Lambda maupun di dalam Log. CloudWatch

Untuk performa terbaik, gunakan hanya huruf kecil untuk nama grup log dan nama aliran log Anda. Menggunakan casing campuran menyebabkan konektor melakukan pencarian case insensitive yang lebih intensif secara komputasi.

Kueri passthrough

CloudWatch Konektor mendukung kueri passthrough yang menggunakan sintaks kueri CloudWatch Logs Insights. Untuk informasi selengkapnya tentang Wawasan CloudWatch Log, lihat Menganalisis data CloudWatch log dengan Wawasan Log di Panduan Pengguna CloudWatch Log Amazon.

Untuk membuat kueri passthrough dengan CloudWatch, gunakan sintaks berikut:

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

Berikut contoh CloudWatch passthrough query filter untuk `duration` bidang ketika tidak sama dengan 1000.

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

Informasi lisensi

Proyek CloudWatch konektor Amazon Athena dilisensikan di bawah Lisensi Apache-2.0.

Sumber daya tambahan

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Metrik Amazon Athena CloudWatch

Konektor CloudWatch Metrik Amazon Athena memungkinkan Amazon Athena untuk menanyakan CloudWatch data Metrik dengan SQL.

Untuk informasi tentang memublikasikan metrik kueri CloudWatch dari Athena itu sendiri, lihat. Mengontrol biaya dan memantau kueri dengan CloudWatch metrik dan peristiwa

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengonfigurasi konektor CloudWatch Metrik.

- spill_bucket - Menentukan bucket Amazon S3 untuk data yang melebihi batas fungsi Lambda.

- spill_prefix — (Opsional) Default ke subfolder dalam nama yang ditentukan. `spill_bucket athena-federation-spill` Kami menyarankan Anda mengonfigurasi siklus hidup penyimpanan Amazon S3 di lokasi ini untuk menghapus tumpahan yang lebih lama dari jumlah hari atau jam yang telah ditentukan sebelumnya.
- spill_put_request_headers — (Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject {"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.
- kms_key_id — (Opsional) Secara default, data apa pun yang tumpah ke Amazon S3 dienkripsi menggunakan mode enkripsi yang diautentikasi AES-GCM dan kunci yang dihasilkan secara acak. Agar fungsi Lambda Anda menggunakan kunci enkripsi yang lebih kuat yang dihasilkan oleh KMS sepertia`7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, Anda dapat menentukan ID kunci KMS.
- disable_spill_encryption — (Opsional) Ketika diatur ke, menonaktifkan enkripsi tumpahan. `True` Defaultnya `False` sehingga data yang tumpah ke S3 dienkripsi menggunakan AES-GCM — baik menggunakan kunci yang dihasilkan secara acak atau KMS untuk menghasilkan kunci. Menonaktifkan enkripsi tumpahan dapat meningkatkan kinerja, terutama jika lokasi tumpahan Anda menggunakan enkripsi sisi server.

Konektor ini juga mendukung kontrol kemacetan AIMD untuk menangani peristiwa pelambatan dari melalui konstruksi CloudWatch Amazon Athena Query Federation SDK. `ThrottlingInvoker` Anda dapat mengubah perilaku pelambatan default dengan menyetel salah satu variabel lingkungan opsional berikut:

- throttle_initial_delay_ms - Penundaan panggilan awal diterapkan setelah peristiwa kemacetan pertama. Defaultnya adalah 10 milidetik.
- throttle_max_delay_ms — Penundaan maksimum antara panggilan. Anda dapat memperoleh TPS dengan membaginya menjadi 1000ms. Defaultnya adalah 1000 milidetik.
- throttle_decrease_factor — Faktor dimana Athena mengurangi tingkat panggilan. Defaultnya adalah 0,5
- throttle_increase_ms — Tingkat di mana Athena mengurangi penundaan panggilan. Defaultnya adalah 10 milidetik.

Database dan tabel

Konektor Athena CloudWatch Metrics memetakan ruang nama, dimensi, metrik, dan nilai metrik Anda ke dalam dua tabel dalam satu skema yang disebut. `default`

Tabel metrik

`metrics`Tabel berisi metrik yang tersedia sebagaimana didefinisikan secara unik oleh kombinasi namespace, set, dan nama. `metrics`Tabel berisi kolom berikut.

- namespace — A yang `VARCHAR` berisi namespace.
- metric_name — A yang `VARCHAR` berisi nama metrik.
- dimensi — Sebuah `LIST` `STRUCT` objek yang terdiri dari `dim_name (VARCHAR)` dan`dim_value (VARCHAR)`.
- statistik — Sebuah `LIST` `VARCH` statistik (misalnya,, `p90AVERAGE`,...) tersedia untuk metrik.

Tabel metric_samples

`metric_samples`Tabel berisi sampel metrik yang tersedia untuk setiap metrik dalam `metrics` tabel. `metric_samples`Tabel berisi kolom berikut.

- namespace — A `VARCHAR` yang berisi namespace.
- metric_name — A `VARCHAR` yang berisi nama metrik.
- dimensi — Sebuah `LIST` `STRUCT` objek yang terdiri dari `dim_name (VARCHAR)` dan`dim_value (VARCHAR)`.
- dim_name - Bidang `VARCHAR` kenyamanan yang dapat Anda gunakan untuk memfilter dengan mudah pada satu nama dimensi.
- dim_value — Bidang `VARCHAR` kenyamanan yang dapat Anda gunakan untuk memfilter dengan mudah pada nilai dimensi tunggal.
- Periode — `INT` Bidang yang mewakili “periode” metrik dalam detik (misalnya, metrik 60 detik).
- stempel waktu — `BIGINT` Bidang yang mewakili waktu epoch dalam detik untuk sampel metrik.
- value — `FLOAT8` Bidang yang berisi nilai sampel.
- statistik — A `VARCHAR` yang berisi tipe statistik sampel (misalnya, `AVERAGE` ataup`90`).

Izin yang Diperlukan

Untuk detail selengkapnya tentang kebijakan IAM yang diperlukan konektor ini, tinjau `Policies` bagian athena-cloudwatch-metricsfile.yaml. Daftar berikut merangkum izin yang diperlukan.

- Akses tulis Amazon S3 - Konektor memerlukan akses tulis ke lokasi di Amazon S3 untuk menumpahkan hasil dari kueri besar.
- Athena GetQueryExecution — Konektor menggunakan izin ini untuk gagal cepat ketika kueri Athena hulu telah dihentikan.
- CloudWatch Metrik ReadOnly — Konektor menggunakan izin ini untuk menanyakan data metrik Anda.
- CloudWatch Log Tulis — Konektor menggunakan akses ini untuk menulis log diagnostiknya.

Kinerja

Konektor CloudWatch Metrik Athena mencoba mengoptimalkan kueri terhadap CloudWatch Metrik dengan memparalelkan pemindaian aliran log yang diperlukan untuk kueri Anda. Untuk periode waktu tertentu, metrik, namespace, dan filter dimensi, pushdown predikat dilakukan baik di dalam fungsi Lambda maupun di dalam Log. CloudWatch

Informasi lisensi

Proyek konektor CloudWatch Metrik Amazon Athena dilisensikan di bawah Lisensi Apache-2.0.

Sumber daya tambahan

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor CMDB Amazon Athena AWS

AWS Konektor CMDB Amazon Athena memungkinkan Athena berkomunikasi dengan berbagai AWS layanan sehingga Anda dapat menanyakannya dengan SQL.

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengonfigurasi konektor AWS CMDB.

- spill_bucket - Menentukan bucket Amazon S3 untuk data yang melebihi batas fungsi Lambda.
- spill_prefix — (Opsional) Default ke subfolder dalam nama yang ditentukan. `spill_bucket` `athena-federation-spill` Kami menyarankan Anda mengonfigurasi siklus hidup penyimpanan Amazon S3 di lokasi ini untuk menghapus tumpahan yang lebih lama dari jumlah hari atau jam yang telah ditentukan sebelumnya.
- spill_put_request_headers — (Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject` `{"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.
- kms_key_id — (Opsional) Secara default, data apa pun yang tumpah ke Amazon S3 dienkripsi menggunakan mode enkripsi yang diautentikasi AES-GCM dan kunci yang dihasilkan secara acak. Agar fungsi Lambda Anda menggunakan kunci enkripsi yang lebih kuat yang dihasilkan oleh KMS sepertia`7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, Anda dapat menentukan ID kunci KMS.
- disable_spill_encryption — (Opsional) Ketika diatur ke, menonaktifkan enkripsi tumpahan. `True` Defaultnya `False` sehingga data yang tumpah ke S3 dienkripsi menggunakan AES-GCM — baik menggunakan kunci yang dihasilkan secara acak atau KMS untuk menghasilkan kunci. Menonaktifkan enkripsi tumpahan dapat meningkatkan kinerja, terutama jika lokasi tumpahan Anda menggunakan enkripsi sisi server.
- default_ec2_image_owner — (Opsional) Saat disetel, mengontrol pemilik gambar Amazon EC2 default yang memfilter Gambar Mesin Amazon (AMI). Jika Anda tidak menetapkan nilai ini dan kueri Anda terhadap tabel gambar EC2 tidak menyertakan filter untuk pemilik, hasil Anda akan mencakup semua gambar publik.

Database dan tabel

Konektor AWS CMDB Athena membuat database dan tabel berikut tersedia untuk menanyakan inventaris sumber daya Anda. AWS Untuk informasi selengkapnya tentang kolom yang tersedia di setiap tabel, jalankan `DESCRIBE` *`database`*`.`*`table`* pernyataan menggunakan konsol Athena atau API.

- ec2 — Database ini berisi sumber daya terkait Amazon EC2, termasuk yang berikut ini.

- ebs_volumes - Berisi detail volume Amazon EBS Anda.
- ec2_instances - Berisi rincian Instans EC2 Anda.
- ec2_images - Berisi rincian gambar Instans EC2 Anda.
- routing_tables - Berisi rincian Tabel Routing VPC Anda.
- security_groups - Berisi rincian grup keamanan Anda.
- subnet - Berisi detail Subnet VPC Anda.
- vpcs - Berisi detail VPC Anda.

- emr - Database ini berisi sumber daya terkait Amazon EMR, termasuk yang berikut ini.

- emr_clusters - Berisi rincian Cluster EMR Anda.

- rds - Database ini berisi sumber daya terkait Amazon RDS, termasuk yang berikut ini.

- rds_instances - Berisi rincian Instans RDS Anda.

- s3 — Database ini berisi sumber daya terkait RDS, termasuk yang berikut ini.

- ember - Berisi detail ember Amazon S3 Anda.
- objek - Berisi detail objek Amazon S3 Anda, tidak termasuk isinya.

Izin yang Diperlukan

Untuk detail selengkapnya tentang kebijakan IAM yang diperlukan konektor ini, tinjau `Policies` bagian athena-aws-cmdbfile.yaml. Daftar berikut merangkum izin yang diperlukan.

- Akses tulis Amazon S3 - Konektor memerlukan akses tulis ke lokasi di Amazon S3 untuk menumpahkan hasil dari kueri besar.
- Athena GetQueryExecution — Konektor menggunakan izin ini untuk gagal cepat ketika kueri Athena hulu telah dihentikan.
- Daftar S3 - Konektor menggunakan izin ini untuk mencantumkan bucket dan objek Amazon S3 Anda.

- Jelaskan EC2 — Konektor menggunakan izin ini untuk menjelaskan sumber daya seperti instans Amazon EC2, grup keamanan, VPC, dan volume Amazon EBS.
- EMR Deskripsi/Daftar — Konektor menggunakan izin ini untuk menggambarkan kluster EMR Anda.
- RDS Deskripsikan — Konektor menggunakan izin ini untuk menggambarkan Instans RDS Anda.

Kinerja

Saat ini, konektor AWS CMDB Athena tidak mendukung pemindaian paralel. Predikat pushdown dilakukan dalam fungsi Lambda. Jika memungkinkan, predikat sebagian didorong ke layanan yang ditanyakan. Misalnya, kueri untuk detail instans Amazon EC2 tertentu memanggil EC2 API dengan ID instans tertentu untuk menjalankan operasi deskripsi yang ditargetkan.

Informasi lisensi

Proyek konektor AWS CMDB Amazon Athena dilisensikan di bawah Lisensi Apache-2.0.

Sumber daya tambahan

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Amazon Athena IBM Db2

Konektor Amazon Athena untuk Db2 memungkinkan Amazon Athena menjalankan kueri SQL pada database IBM Db2 Anda menggunakan JDBC.

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.
- Siapkan VPC dan grup keamanan sebelum Anda menggunakan konektor ini. Untuk informasi selengkapnya, lihat Membuat VPC untuk konektor sumber data.

Batasan

- Menulis operasi DDL tidak didukung.
- Dalam pengaturan multiplexer, bucket tumpahan dan awalan dibagikan di semua instance database.

- Batas Lambda yang relevan. Untuk informasi selengkapnya, lihat Kuota Lambda dalam Panduan Pengembang AWS Lambda .
- Tipe data tanggal dan stempel waktu dalam kondisi filter harus dilemparkan ke tipe data yang sesuai.

Ketentuan

Istilah-istilah berikut berhubungan dengan konektor Db2.

- Instans database — Instance apa pun dari database yang digunakan di tempat, di Amazon EC2, atau di Amazon RDS.
- Handler - Handler Lambda yang mengakses instance database Anda. Handler bisa untuk metadata atau untuk catatan data.
- Metadata handler — Penangan Lambda yang mengambil metadata dari instance database Anda.
- Record handler - Handler Lambda yang mengambil catatan data dari instance database Anda.
- Composite handler — Handler Lambda yang mengambil data metadata dan data dari instance database Anda.
- Properti atau parameter - Properti database yang digunakan oleh penangan untuk mengekstrak informasi database. Anda mengonfigurasi properti ini sebagai variabel lingkungan Lambda.
- Connection String — Sebuah string teks yang digunakan untuk membuat koneksi ke instance database.
- Katalog —AWS Glue Non-katalog yang terdaftar di Athena yang merupakan awalan yang diperlukan untuk properti. `connection_string`
- Multiplexing handler - Handler Lambda yang dapat menerima dan menggunakan beberapa koneksi database.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengkonfigurasi konektor Db2.

String koneksi

Gunakan string koneksi JDBC dalam format berikut untuk terhubung ke instance database.

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

Menggunakan handler multiplexing

Anda dapat menggunakan multiplexer untuk terhubung ke beberapa instance database dengan satu fungsi Lambda. Permintaan dirutekan dengan nama katalog. Gunakan kelas berikut di Lambda.

Handler	Kelas
Pawang komposit	`Db2MuxCompositeHandler`
Penangan metadata	`Db2MuxMetadataHandler`
Rekam handler	`Db2MuxRecordHandler`

Parameter handler multiplexing

Parameter	Deskripsi
`$`*`catalog`*`_connection_string`	Wajib. Sebuah string koneksi instance database. Awalan variabel lingkungan dengan nama katalog yang digunakan di Athena. Misalnya, jika katalog terdaftar di Athena adalah`mydbtwocatalog` , maka nama variabel lingkungan adalah. `mydbtwocatalog_connection_string`
`default`	Wajib. String koneksi default. String ini digunakan ketika katalog `lambda:${`*`AWS_LAMBDA_FUNCTION_NAME`*`}` .

Contoh properti berikut adalah untuk fungsi Db2 MUX Lambda yang mendukung dua instance database`dbtwo1`: (default), dan. `dbtwo2`

Properti	Nilai
`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`*`}`

Properti	Nilai
`dbtwo_catalog2_connection_string`	`dbtwo://jdbc:db2://dbtwo2.  hostname: port/ database_name :${secret2_name }`

Memberikan kredensi

Untuk memberikan nama pengguna dan kata sandi untuk database Anda dalam string koneksi JDBC Anda, Anda dapat menggunakan properti string koneksi atau. AWS Secrets Manager

- Connection String - Nama pengguna dan kata sandi dapat ditentukan sebagai properti dalam string koneksi JDBC.

  > **Important**
  >
  > Sebagai praktik keamanan terbaik, jangan gunakan kredensyal hardcode dalam variabel lingkungan atau string koneksi Anda. Untuk informasi tentang memindahkan rahasia hardcode Anda AWS Secrets Manager, lihat Memindahkan rahasia hardcode ke AWS Secrets Manager dalam Panduan Pengguna.AWS Secrets Manager

- AWS Secrets Manager— Untuk menggunakan fitur Query Federasi Athena dengan, VPC AWS Secrets Manager yang terhubung ke fungsi Lambda Anda harus memiliki akses internet atau titik akhir VPC untuk terhubung ke Secrets Manager.

  Anda dapat memasukkan nama rahasia ke AWS Secrets Manager dalam string koneksi JDBC Anda. Konektor menggantikan nama rahasia dengan `password` nilai `username` dan dari Secrets Manager.

  Untuk instans database Amazon RDS, dukungan ini terintegrasi erat. Jika Anda menggunakan Amazon RDS, kami sangat menyarankan penggunaan AWS Secrets Manager dan rotasi kredensyal. Jika database Anda tidak menggunakan Amazon RDS, simpan kredensialnya sebagai JSON dalam format berikut:

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

Contoh string koneksi dengan nama rahasia

String berikut memiliki nama rahasia`${secret_name}`.

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

Konektor menggunakan nama rahasia untuk mengambil rahasia dan memberikan nama pengguna dan kata sandi, seperti pada contoh berikut.

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

Menggunakan handler koneksi tunggal

Anda dapat menggunakan metadata koneksi tunggal berikut dan penangan rekaman untuk terhubung ke satu instance Db2.

Jenis handler	Kelas
Pawang komposit	`Db2CompositeHandler`
Penangan metadata	`Db2MetadataHandler`
Rekam handler	`Db2RecordHandler`

Parameter handler koneksi tunggal

Parameter	Deskripsi
`default`	Wajib. String koneksi default.

Penangan koneksi tunggal mendukung satu instance database dan harus menyediakan parameter string `default` koneksi. Semua string koneksi lainnya diabaikan.

Properti contoh berikut adalah untuk contoh Db2 tunggal didukung oleh fungsi Lambda.

Properti	Nilai
`default`	*dbtwo: //jdbc:db2: //hostname:port/ database_name: ${secret_name}*

Parameter tumpahan

Lambda SDK dapat menumpahkan data ke Amazon S3. Semua instance database yang diakses oleh fungsi Lambda yang sama tumpah ke lokasi yang sama.

Parameter	Deskripsi
`spill_bucket`	Wajib. Nama ember tumpahan.
`spill_prefix`	Wajib. Tumpahkan key prefix bucket.
`spill_put_request_headers`	(Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject {"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.

Dukungan tipe data

Tabel berikut menunjukkan tipe data yang sesuai untuk JDBC dan Arrow.

Db2	Panah
CHAR	VARCHAR
VARCHAR	VARCHAR
DATE	TANGGAL
TIME	VARCHAR
TIMESTAMP	DATEMILLI
DATETIME	DATEMILLI
BOOLEAN	BOOL
SMALLINT	SMALLINT

Db2	Panah
INTEGER	INT
BIGINT	BIGINT
DECIMAL	DECIMAL
REAL	FLOAT8
DOUBLE	FLOAT8
DECFLOAT	FLOAT8

Partisi dan split

Partisi diwakili oleh satu atau lebih kolom partisi tipe`varchar`. Konektor Db2 membuat partisi menggunakan skema organisasi berikut.

- Distribusikan dengan hash
- Partisi berdasarkan jangkauan
- Atur berdasarkan dimensi

Konektor mengambil rincian partisi seperti jumlah partisi dan nama kolom dari satu atau lebih tabel metadata Db2. Pemisahan dibuat berdasarkan jumlah partisi yang diidentifikasi.

Kinerja

Konektor Athena Db2 melakukan pushdown predikat untuk mengurangi data yang dipindai oleh kueri. `LIMIT`klausa, predikat sederhana, dan ekspresi kompleks didorong ke konektor untuk mengurangi jumlah data yang dipindai dan mengurangi waktu eksekusi kueri.

Klausul LIMIT

`LIMIT N`Pernyataan mengurangi data yang dipindai oleh kueri. Dengan `LIMIT N` pushdown, konektor hanya mengembalikan `N` baris ke Athena.

Predikat

Predikat adalah ekspresi dalam `WHERE` klausa kueri SQL yang mengevaluasi nilai Boolean dan menyaring baris berdasarkan beberapa kondisi. Konektor Athena Db2 dapat menggabungkan ekspresi ini dan mendorongnya langsung ke Db2 untuk meningkatkan fungsionalitas dan untuk mengurangi jumlah data yang dipindai.

Operator konektor Athena Db2 berikut mendukung pushdown predikat:

- Boolean: DAN, ATAU, TIDAK
- KESETARAAN: SAMA, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, IS_NULL
- Aritmatika: TAMBAHKAN, KURANGI, KALIKAN, BAGI, MODULUS, MENIADAKAN
- Lainnya: LIKE_PATTERN, IN

Contoh pushdown gabungan

Untuk kemampuan kueri yang ditingkatkan, gabungkan jenis pushdown, seperti pada contoh berikut:

```
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;
```

Kueri passthrough

Konektor Db2 mendukung kueri passthrough. Kueri passthrough menggunakan fungsi tabel untuk mendorong kueri lengkap Anda ke sumber data untuk dieksekusi.

Untuk menggunakan kueri passthrough dengan Db2, Anda dapat menggunakan sintaks berikut:

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

Contoh query berikut mendorong ke bawah query ke sumber data di Db2. Kueri memilih semua kolom dalam `customer` tabel, membatasi hasilnya menjadi 10.

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

Informasi lisensi

Dengan menggunakan konektor ini, Anda mengakui penyertaan komponen pihak ketiga, daftar yang dapat ditemukan dalam file pom.xml untuk konektor ini, dan menyetujui persyaratan dalam masing-masing lisensi pihak ketiga yang disediakan dalam file LICENSE.txt di .com. GitHub

Sumber daya tambahan

Untuk informasi versi driver JDBC terbaru, lihat file pom.xml untuk konektor Db2 di.com. GitHub

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Amazon Athena IBM Db2 AS/400 (Db2 iSeries)

Konektor Amazon Athena untuk Db2 AS/400 memungkinkan Amazon Athena menjalankan kueri SQL pada database IBM Db2 AS/400 (Db2 iSeries) Anda menggunakan JDBC.

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.
- Siapkan VPC dan grup keamanan sebelum Anda menggunakan konektor ini. Untuk informasi selengkapnya, lihat Membuat VPC untuk konektor sumber data.

Batasan

- Menulis operasi DDL tidak didukung.
- Dalam pengaturan multiplexer, bucket tumpahan dan awalan dibagikan di semua instance database.
- Batas Lambda yang relevan. Untuk informasi selengkapnya, lihat Kuota Lambda dalam Panduan Pengembang AWS Lambda .

- Tipe data tanggal dan stempel waktu dalam kondisi filter harus dilemparkan ke tipe data yang sesuai.

Ketentuan

Istilah-istilah berikut berhubungan dengan konektor Db2 AS/400.

- Instans database — Instance apa pun dari database yang digunakan di tempat, di Amazon EC2, atau di Amazon RDS.
- Handler - Handler Lambda yang mengakses instance database Anda. Handler bisa untuk metadata atau untuk catatan data.
- Metadata handler — Penangan Lambda yang mengambil metadata dari instance database Anda.
- Record handler - Handler Lambda yang mengambil catatan data dari instance database Anda.
- Composite handler — Handler Lambda yang mengambil data metadata dan data dari instance database Anda.
- Properti atau parameter - Properti database yang digunakan oleh penangan untuk mengekstrak informasi database. Anda mengonfigurasi properti ini sebagai variabel lingkungan Lambda.
- Connection String — Sebuah string teks yang digunakan untuk membuat koneksi ke instance database.
- Katalog —AWS Glue Non-katalog yang terdaftar di Athena yang merupakan awalan yang diperlukan untuk properti. `connection_string`
- Multiplexing handler - Handler Lambda yang dapat menerima dan menggunakan beberapa koneksi database.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengkonfigurasi konektor Db2 AS/400.

String koneksi

Gunakan string koneksi JDBC dalam format berikut untuk terhubung ke instance database.

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

Menggunakan handler multiplexing

Anda dapat menggunakan multiplexer untuk terhubung ke beberapa instance database dengan satu fungsi Lambda. Permintaan dirutekan dengan nama katalog. Gunakan kelas-kelas berikut di Lambda.

Handler	Kelas
Handler komposit	`Db2MuxCompositeHandler`
Penangan metadata	`Db2MuxMetadataHandler`
Rekam handler	`Db2MuxRecordHandler`

Parameter handler multiplexing

Parameter	Deskripsi
`$`*`catalog`*`_connection_string`	Wajib. Sebuah string koneksi instance database. Awalan variabel lingkungan dengan nama katalog yang digunakan di Athena. Misalnya, jika katalog terdaftar di Athena adalah`mydb2as400catalog` , maka nama variabel lingkungan adalah. `mydb2as400catalog_connection_string`
`default`	Wajib. String koneksi default. String ini digunakan ketika katalog `lambda:${` *`}AWS_LAMBDA_FUNCTION_NAME`* `.`

Contoh properti berikut adalah untuk fungsi Db2 MUX Lambda yang mendukung dua instance database`db2as4001`: (default), dan. `db2as4002`

Properti	Nilai
`default`	`db2as400://jdbc:as400://` *`<ip_address>`* `;`*`<properties>`* `;:${`*`<secret name>`*`};`
`db2as400_catalog1_connection_string`	`db2as400://jdbc:as400://db2as4001. hostname/:${` *`secret1_name`* `}`

Properti	Nilai
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>* ;

Memberikan kredensi

Untuk memberikan nama pengguna dan kata sandi untuk database Anda dalam string koneksi JDBC Anda, Anda dapat menggunakan properti string koneksi atau. AWS Secrets Manager

- Connection String - Nama pengguna dan kata sandi dapat ditentukan sebagai properti dalam string koneksi JDBC.

  > **Important**
  >
  > Sebagai praktik keamanan terbaik, jangan gunakan kredensyal hardcode dalam variabel lingkungan atau string koneksi Anda. Untuk informasi tentang memindahkan rahasia hardcode Anda AWS Secrets Manager, lihat Memindahkan rahasia hardcode ke AWS Secrets Manager dalam Panduan Pengguna.AWS Secrets Manager

- AWS Secrets Manager— Untuk menggunakan fitur Query Federasi Athena dengan, VPC AWS Secrets Manager yang terhubung ke fungsi Lambda Anda harus memiliki akses internet atau titik akhir VPC untuk terhubung ke Secrets Manager.

  Anda dapat memasukkan nama rahasia ke AWS Secrets Manager dalam string koneksi JDBC Anda. Konektor menggantikan nama rahasia dengan `password` nilai `username` dan dari Secrets Manager.

  Untuk instans database Amazon RDS, dukungan ini terintegrasi dengan erat. Jika Anda menggunakan Amazon RDS, kami sangat menyarankan penggunaan AWS Secrets Manager dan rotasi kredenal. Jika database Anda tidak menggunakan Amazon RDS, simpan kredensialnya sebagai JSON dalam format berikut:

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

Contoh string koneksi dengan nama rahasia

String berikut memiliki nama rahasia`${secret_name}`.

```
db2as400://jdbc:as400://<ip_address>;<properties>;:${<secret_name>};
```

Konektor menggunakan nama rahasia untuk mengambil rahasia dan memberikan nama pengguna dan kata sandi, seperti pada contoh berikut.

```
db2as400://jdbc:as400://<ip_address>;user=<username>;password=<password>;<properties>;
```

Menggunakan handler koneksi tunggal

Anda dapat menggunakan metadata koneksi tunggal berikut dan penangan rekaman untuk terhubung ke satu instans Db2 AS/400.

Jenis handler	Kelas
Handler komposit	`Db2CompositeHandler`
Penangan metadata	`Db2MetadataHandler`
Rekam handler	`Db2RecordHandler`

Parameter handler koneksi tunggal

Parameter	Deskripsi
`default`	Wajib. String koneksi default.

Penangan koneksi tunggal mendukung satu instance database dan harus menyediakan parameter string `default` koneksi. Semua string koneksi lainnya diabaikan.

Properti contoh berikut adalah untuk satu contoh Db2 AS/400 didukung oleh fungsi Lambda.

Properti	Nilai
default	db2as400://jdbc:as400:// *<ip_address>* ;*<properties>* ;: ${*<secret_name>* };

Parameter tumpahan

Lambda SDK dapat menumpahkan data ke Amazon S3. Semua instance database yang diakses oleh fungsi Lambda yang sama tumpah ke lokasi yang sama.

Parameter	Deskripsi
spill_bucket	Wajib. Nama ember tumpahan.
spill_prefix	Wajib. Tumpahkan key prefix bucket.
spill_put_request_headers	(Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). putObject {"x-amz-server-side-encryption" : "AES256"} Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.

Dukungan tipe data

Tabel berikut menunjukkan tipe data yang sesuai untuk JDBC dan Apache Arrow.

Db2 AS/400	Panah
CHAR	VARCHAR
VARCHAR	VARCHAR
DATE	TANGGAL
TIME	VARCHAR

Db2 AS/400	Panah
TIMESTAMP	DATEMILLI
DATETIME	DATEMILLI
BOOLEAN	BOOL
SMALLINT	SMALLINT
INTEGER	INT
BIGINT	BIGINT
DECIMAL	DECIMAL
REAL	FLOAT8
DOUBLE	FLOAT8
DECFLOAT	FLOAT8

Partisi dan perpecahan

Partisi diwakili oleh satu atau lebih kolom partisi tipe`varchar`. Konektor Db2 AS/400 membuat partisi menggunakan skema organisasi berikut.

- Distribusikan dengan hash
- Partisi berdasarkan jangkauan
- Atur berdasarkan dimensi

Konektor mengambil rincian partisi seperti jumlah partisi dan nama kolom dari satu atau lebih tabel metadata Db2 AS/400. Pemisahan dibuat berdasarkan jumlah partisi yang diidentifikasi.

Kinerja

Untuk meningkatkan kinerja, gunakan predikat pushdown untuk kueri dari Athena, seperti pada contoh berikut.

```
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'
```

Kueri passthrough

Konektor Db2 AS/400 mendukung kueri passthrough. Kueri passthrough menggunakan fungsi tabel untuk mendorong kueri lengkap Anda ke sumber data untuk dieksekusi.

Untuk menggunakan kueri passthrough dengan Db2 AS/400, Anda dapat menggunakan sintaks berikut:

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

Contoh query berikut mendorong ke bawah query ke sumber data di Db2 AS/400. Kueri memilih semua kolom dalam `customer` tabel, membatasi hasilnya menjadi 10.

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

Informasi lisensi

Dengan menggunakan konektor ini, Anda mengakui penyertaan komponen pihak ketiga, daftar yang dapat ditemukan dalam file pom.xml untuk konektor ini, dan menyetujui persyaratan dalam masing-masing lisensi pihak ketiga yang disediakan dalam file LICENSE.txt di .com. GitHub

Sumber daya tambahan

Untuk informasi versi driver JDBC terbaru, lihat file pom.xml untuk konektor Db2 AS/400 di.com. GitHub

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Amazon Athena DocumentDB

Konektor Amazon Athena DocumentDB memungkinkan Athena berkomunikasi dengan instans DocumentDB Anda sehingga Anda dapat menanyakan data DocumentDB Anda dengan SQL. Konektor juga bekerja dengan endpoint yang kompatibel dengan MongoDB.

Tidak seperti penyimpanan data relasional tradisional, koleksi Amazon DocumentDB tidak memiliki skema yang ditetapkan. DocumentDB tidak memiliki toko metadata. Setiap entri dalam koleksi DocumentDB dapat memiliki bidang dan tipe data yang berbeda.

Konektor DocumentDB mendukung dua mekanisme untuk menghasilkan informasi skema tabel: inferensi skema dasar dan metadata. AWS Glue Data Catalog

Inferensi skema adalah default. Opsi ini memindai sejumlah kecil dokumen dalam koleksi Anda, membentuk gabungan semua bidang, dan memaksa bidang yang memiliki tipe data yang tidak tumpang tindih. Opsi ini berfungsi dengan baik untuk koleksi yang sebagian besar memiliki entri seragam.

Untuk koleksi dengan variasi tipe data yang lebih besar, konektor mendukung pengambilan metadata dari file. AWS Glue Data Catalog Jika konektor melihat AWS Glue database dan tabel yang cocok dengan database DocumentDB dan nama koleksi Anda, ia mendapatkan informasi skema dari tabel yang sesuai. AWS Glue Saat Anda membuat AWS Glue tabel, kami sarankan Anda menjadikannya superset dari semua bidang yang mungkin ingin Anda akses dari koleksi DocumentDB Anda.

Jika Anda mengaktifkan Lake Formation di akun Anda, peran IAM untuk konektor Lambda federasi Athena yang Anda gunakan di harus memiliki akses baca di AWS Serverless Application Repository Lake Formation ke. AWS Glue Data Catalog

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengkonfigurasi konektor DocumentDB.

- spill_bucket - Menentukan bucket Amazon S3 untuk data yang melebihi batas fungsi Lambda.

- spill_prefix — (Opsional) Default ke subfolder dalam nama yang ditentukan. `spill_bucket athena-federation-spill` Kami menyarankan Anda mengonfigurasi siklus hidup penyimpanan Amazon S3 di lokasi ini untuk menghapus tumpahan yang lebih lama dari jumlah hari atau jam yang telah ditentukan sebelumnya.
- spill_put_request_headers — (Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject {"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.
- kms_key_id — (Opsional) Secara default, data apa pun yang tumpah ke Amazon S3 dienkripsi menggunakan mode enkripsi yang diautentikasi AES-GCM dan kunci yang dihasilkan secara acak. Agar fungsi Lambda Anda menggunakan kunci enkripsi yang lebih kuat yang dihasilkan oleh KMS sepertia`7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, Anda dapat menentukan ID kunci KMS.
- disable_spill_encryption — (Opsional) Ketika diatur ke, menonaktifkan enkripsi tumpahan. `True` Defaultnya `False` sehingga data yang tumpah ke S3 dienkripsi menggunakan AES-GCM — baik menggunakan kunci yang dihasilkan secara acak atau KMS untuk menghasilkan kunci. Menonaktifkan enkripsi tumpahan dapat meningkatkan kinerja, terutama jika lokasi tumpahan Anda menggunakan enkripsi sisi server.
- disable_glue — (Opsional) Jika ada dan disetel ke true, konektor tidak mencoba untuk mengambil metadata tambahan dari. AWS Glue
- glue_catalog - (Opsional) Gunakan opsi ini untuk menentukan katalog lintas akun. AWS Glue Secara default, konektor mencoba untuk mendapatkan metadata dari akunnya sendiri AWS Glue .
- default_docdb - Jika ada, menentukan string koneksi DocumentDB untuk digunakan ketika tidak ada variabel lingkungan khusus katalog ada.
- disable_projection_and_casing - (Opsional) Menonaktifkan proyeksi dan casing. Gunakan jika Anda ingin menanyakan tabel Amazon DocumentDB yang menggunakan nama kolom peka huruf besar/kecil. `disable_projection_and_casing`Parameter menggunakan nilai-nilai berikut untuk menentukan perilaku pemetaan casing dan kolom:
  - false — Ini adalah pengaturan default. Proyeksi diaktifkan, dan konektor mengharapkan semua nama kolom berada dalam huruf kecil.
  - benar - Menonaktifkan proyeksi dan casing. Saat menggunakan `disable_projection_and_casing` parameter, ingatlah poin-poin berikut:
    - Penggunaan parameter dapat menghasilkan penggunaan bandwidth yang lebih tinggi. Selain itu, jika fungsi Lambda Anda tidak sama Wilayah AWS dengan sumber data Anda, Anda akan dikenakan biaya transfer AWS lintas wilayah standar yang lebih tinggi sebagai akibat dari

penggunaan bandwidth yang lebih tinggi. Untuk informasi selengkapnya tentang biaya transfer lintas wilayah, lihat Biaya Transfer AWS Data untuk Arsitektur Server dan Tanpa Server di Blog Jaringan Mitra. AWS

  - Karena jumlah byte yang lebih besar ditransfer dan karena jumlah byte yang lebih besar memerlukan waktu deserialisasi yang lebih tinggi, latensi keseluruhan dapat meningkat.
- enable_case_insensitive_match — (Opsional) Saat, melakukan pencarian yang tidak peka huruf besar/kecil terhadap skema dan nama tabel di `true` Amazon DocumentDB. Nilai default-nya `false`. Gunakan jika kueri Anda berisi skema huruf besar atau nama tabel.

Menentukan string koneksi

Anda dapat memberikan satu atau beberapa properti yang menentukan detail koneksi DocumentDB untuk instance DocumentDB yang Anda gunakan dengan konektor. Untuk melakukan ini, tetapkan variabel lingkungan Lambda yang sesuai dengan nama katalog yang ingin Anda gunakan di Athena. Misalnya, Anda ingin menggunakan kueri berikut untuk menanyakan dua instance DocumentDB yang berbeda dari Athena:

```
SELECT * FROM "docdb_instance_1".database.table
```

```
SELECT * FROM "docdb_instance_2".database.table
```

Sebelum Anda dapat menggunakan dua pernyataan SQL ini, Anda harus menambahkan dua variabel lingkungan ke fungsi `docdb_instance_1` Lambda Anda: dan. `docdb_instance_2` Nilai untuk masing-masing harus berupa string koneksi DocumentDB dalam format berikut:

```
mongodb://:@:/?ssl=true&ssl_ca_certs=rds-combined-ca-bundle.pem&replicaSet=rs0
```

Menggunakan rahasia

Anda dapat secara opsional menggunakan AWS Secrets Manager sebagian atau seluruh nilai untuk detail string koneksi Anda. Untuk menggunakan fitur Kueri Federasi Athena dengan Secrets Manager, VPC yang terhubung ke fungsi Lambda Anda harus memiliki akses internet atau titik akhir VPC untuk terhubung ke Secrets Manager.

Jika Anda menggunakan sintaks `${my_secret}` untuk memasukkan nama rahasia dari Secrets Manager di string koneksi Anda, konektor akan menggantikan `${my_secret}` dengan nilai teks biasa dari Secrets Manager persis. Rahasia harus disimpan sebagai rahasia

teks biasa dengan nilai`<username>:<password>`. Rahasia yang disimpan sebagai tidak `{username:<username>,password:<password>}` akan diteruskan ke string koneksi dengan benar.

Rahasia juga dapat digunakan untuk seluruh string koneksi sepenuhnya, dan nama pengguna dan kata sandi dapat didefinisikan dalam rahasia.

Misalnya, Anda menyetel variabel lingkungan Lambda `docdb_instance_1` ke nilai berikut:

```
mongodb://${docdb_instance_1_creds}@myhostname.com:123/?ssl=true&ssl_ca_certs=rds-
combined-ca-bundle.pem&replicaSet=rs0
```

Athena Query Federation SDK secara otomatis mencoba untuk mengambil rahasia bernama dari Secrets `docdb_instance_1_creds` Manager dan menyuntikkan nilai tersebut sebagai pengganti. `${docdb_instance_1_creds}` Setiap bagian dari string koneksi yang dilampirkan oleh kombinasi `${ }` karakter ditafsirkan sebagai rahasia dari Secrets Manager. Jika Anda menentukan nama rahasia yang tidak dapat ditemukan konektor di Secrets Manager, konektor tidak menggantikan teks.

Menyiapkan database dan tabel di AWS Glue

Karena kemampuan inferensi skema bawaan konektor memindai sejumlah dokumen terbatas dan hanya mendukung sebagian tipe data, Anda mungkin ingin menggunakan AWS Glue metadata sebagai gantinya.

Untuk mengaktifkan AWS Glue tabel untuk digunakan dengan Amazon DocumentDB, Anda harus memiliki AWS Glue database dan tabel untuk database dan koleksi DocumentDB yang ingin Anda berikan metadata tambahan.

Untuk menggunakan AWS Glue tabel untuk metadata tambahan

1. Saat Anda mengedit tabel dan database di AWS Glue konsol, tambahkan properti tabel berikut.
   - docdb-metadata-flag- Properti ini menunjukkan ke konektor DocumentDB bahwa konektor dapat menggunakan tabel untuk metadata tambahan. Anda dapat memberikan nilai apa pun `docdb-metadata-flag` selama `docdb-metadata-flag` properti hadir dalam daftar properti tabel.
2. (Opsional) Tambahkan properti tabel SourceTable. Properti ini mendefinisikan nama tabel sumber di Amazon DocumentDB. Gunakan properti ini jika aturan penamaan AWS Glue tabel mencegah Anda membuat AWS Glue tabel dengan nama yang sama dengan tabel Amazon

DocumentDB Anda. Misalnya, huruf kapital tidak diizinkan dalam nama AWS Glue tabel, tetapi mereka diizinkan dalam nama tabel Amazon DocumentDB.

3. (Opsional) Tambahkan properti tabel ColumnMapping. Properti ini mendefinisikan pemetaan nama kolom. Gunakan properti ini jika aturan penamaan AWS Glue kolom mencegah Anda membuat AWS Glue tabel yang memiliki nama kolom yang sama dengan yang ada di tabel Amazon DocumentDB Anda. Ini dapat berguna karena huruf kapital diizinkan dalam nama kolom Amazon DocumentDB tetapi tidak diizinkan AWS Glue dalam nama kolom.

   Nilai `columnMapping` properti diharapkan menjadi satu set pemetaan dalam format. `col1=Col1,col2=Col2`

   > **Note**
   >
   > Pemetaan kolom hanya berlaku untuk nama kolom tingkat atas dan bukan untuk bidang bersarang.

   Setelah Anda menambahkan properti AWS Glue `columnMapping` tabel, Anda dapat menghapus variabel lingkungan `disable_projection_and_casing` Lambda.

4. Pastikan Anda menggunakan tipe data yang sesuai AWS Glue seperti yang tercantum dalam dokumen ini.

Dukungan tipe data

Bagian ini mencantumkan tipe data yang digunakan konektor DocumentDB untuk inferensi skema, dan tipe data saat metadata digunakan. AWS Glue

Jenis data inferensi skema

Fitur inferensi skema konektor DocumentDB mencoba menyimpulkan nilai sebagai milik salah satu tipe data berikut. Tabel menunjukkan tipe data yang sesuai untuk Amazon DocumentDB, Java, dan Apache Arrow.

Panah Apache	Java atau DocDB
VARCHAR	String
INT	Bilangan Bulat

Panah Apache	Java atau DocDB
BIGINT	Long
BIT	Boolean
FLOAT4	Desimal
FLOAT8	Ganda
STEMPEL WAKTU	Tanggal
VARCHAR	ObjectId
DAFTAR	Daftar
STRUCT	Dokumen

AWS Glue tipe data

Jika Anda menggunakan AWS Glue metadata tambahan, Anda dapat mengonfigurasi tipe data berikut. Tabel menunjukkan tipe data yang sesuai untuk AWS Glue dan Apache Arrow.

AWS Glue	Panah Apache
int	INT
bigint	BIGINT
double	FLOAT8
float	FLOAT4
boolean	BIT
biner	VARBINARY
string	VARCHAR
Daftar	DAFTAR

AWS Glue	Panah Apache
Struct	STRUCT

Izin yang Diperlukan

Untuk detail lengkap tentang kebijakan IAM yang diperlukan konektor ini, tinjau `Policies` bagian file athena-docdb.yaml. Daftar berikut merangkum izin yang diperlukan.

- Akses tulis Amazon S3 - Konektor memerlukan akses tulis ke lokasi di Amazon S3 untuk menumpahkan hasil dari kueri besar.
- Athena GetQueryExecution — Konektor menggunakan izin ini untuk gagal cepat ketika kueri Athena hulu telah dihentikan.
- AWS Glue Data Catalog- Konektor DocumentDB membutuhkan akses baca saja ke untuk mendapatkan informasi AWS Glue Data Catalog skema.
- CloudWatch Log — Konektor memerlukan akses ke CloudWatch Log untuk menyimpan log.
- AWS Secrets Manager akses baca - Jika Anda memilih untuk menyimpan detail titik akhir DocumentDB di Secrets Manager, Anda harus memberikan akses konektor ke rahasia tersebut.
- Akses VPC — Konektor memerlukan kemampuan untuk memasang dan melepaskan antarmuka ke VPC Anda sehingga dapat terhubung dengannya dan berkomunikasi dengan instance DocumentDB Anda.

Kinerja

Konektor Athena Amazon DocumentDB saat ini tidak mendukung pemindaian paralel tetapi mencoba untuk menekan predikat sebagai bagian dari kueri DocumentDB-nya, dan predikat terhadap indeks pada koleksi DocumentDB Anda menghasilkan data yang dipindai secara signifikan lebih sedikit.

Fungsi Lambda melakukan pushdown proyeksi untuk mengurangi data yang dipindai oleh kueri. Namun, memilih subset kolom terkadang menghasilkan runtime eksekusi kueri yang lebih lama. `LIMIT`klausa mengurangi jumlah data yang dipindai, tetapi jika Anda tidak memberikan predikat, Anda harus mengharapkan `SELECT` kueri dengan `LIMIT` klausa untuk memindai setidaknya 16 MB data.

Kueri passthrough

Konektor Athena Amazon DocumentDB mendukung kueri passthrough dan berbasis NoSQL. Untuk informasi tentang menanyakan Amazon DocumentDB, lihat Menanyakan di Panduan Pengembang Amazon DocumentDB.

Untuk menggunakan kueri passthrough dengan Amazon DocumentDB, gunakan sintaks berikut:

```
SELECT * FROM TABLE(
        system.query(
            database => 'database_name',
            collection => 'collection_name',
            filter => '{query_syntax}'
        ))
```

Contoh berikut menanyakan `example` database dalam `TPCDS` koleksi, memfilter semua buku dengan judul Bill of Rights.

```
SELECT * FROM TABLE(
        system.query(
            database => 'example',
            collection => 'tpcds',
            filter => '{title: "Bill of Rights"}'
        ))
```

Sumber daya tambahan

- Untuk artikel tentang penggunaan Kueri Federasi Amazon Athena untuk menghubungkan database MongoDB ke Amazon untuk membangun dasbor dan visualisasi, lihat Memvisualisasikan data MongoDB dari Amazon QuickSight menggunakan Kueri Federasi Amazon Athena di Blog Big Data. QuickSight AWS
- Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Amazon Athena DynamoDB

Konektor Amazon Athena DynamoDB memungkinkan Amazon Athena untuk berkomunikasi dengan DynamoDB sehingga Anda dapat mengkueri tabel Anda dengan SQL. Operasi tulis seperti INSERT INTO tidak didukung.

Jika Anda mengaktifkan Lake Formation di akun Anda, peran IAM untuk konektor Lambda federasi Athena yang Anda gunakan di harus memiliki akses baca di AWS Serverless Application Repository Lake Formation ke. AWS Glue Data Catalog

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengkonfigurasi konektor DynamoDB.

- spill_bucket - Menentukan bucket Amazon S3 untuk data yang melebihi batas fungsi Lambda.
- spill_prefix — (Opsional) Default ke subfolder dalam nama yang ditentukan. `spill_bucket athena-federation-spill` Kami menyarankan Anda mengonfigurasi siklus hidup penyimpanan Amazon S3 di lokasi ini untuk menghapus tumpahan yang lebih lama dari jumlah hari atau jam yang telah ditentukan sebelumnya.
- spill_put_request_headers — (Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject {"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.
- kms_key_id — (Opsional) Secara default, data apa pun yang tumpah ke Amazon S3 dienkripsi menggunakan mode enkripsi yang diautentikasi AES-GCM dan kunci yang dihasilkan secara acak. Agar fungsi Lambda Anda menggunakan kunci enkripsi yang lebih kuat yang dihasilkan oleh KMS sepertia`7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, Anda dapat menentukan ID kunci KMS.
- disable_spill_encryption — (Opsional) Ketika diatur ke, menonaktifkan enkripsi tumpahan. `True` Defaultnya `False` sehingga data yang tumpah ke S3 dienkripsi menggunakan AES-GCM — baik menggunakan kunci yang dihasilkan secara acak atau KMS untuk menghasilkan kunci. Menonaktifkan enkripsi tumpahan dapat meningkatkan kinerja, terutama jika lokasi tumpahan Anda menggunakan enkripsi sisi server.
- disable_glue — (Opsional) Jika ada dan disetel ke true, konektor tidak mencoba untuk mengambil metadata tambahan dari. AWS Glue

- glue_catalog - (Opsional) Gunakan opsi ini untuk menentukan katalog lintas akun. AWS Glue Secara default, konektor mencoba untuk mendapatkan metadata dari akunnya sendiri AWS Glue .
- disable_projection_and_casing - (Opsional) Menonaktifkan proyeksi dan casing. Gunakan jika Anda ingin menanyakan tabel DynamoDB yang memiliki casing di nama kolom mereka dan Anda tidak ingin menentukan properti di `columnMapping` meja Anda. AWS Glue

  `disable_projection_and_casing`Parameter menggunakan nilai-nilai berikut untuk menentukan perilaku pemetaan casing dan kolom:

  - auto - Menonaktifkan proyeksi dan casing ketika tipe yang sebelumnya tidak didukung terdeteksi dan pemetaan nama kolom tidak diatur di atas meja. Ini adalah pengaturan default.
  - selalu — Menonaktifkan proyeksi dan casing tanpa syarat. Ini berguna ketika Anda memiliki casing di nama kolom DynamoDB Anda tetapi tidak ingin menentukan pemetaan nama kolom apa pun.

  Saat menggunakan `disable_projection_and_casing` parameter, ingatlah poin-poin berikut:

  - Penggunaan parameter dapat menghasilkan penggunaan bandwidth yang lebih tinggi. Selain itu, jika fungsi Lambda Anda tidak sama Wilayah AWS dengan sumber data Anda, Anda akan dikenakan biaya transfer AWS lintas wilayah standar yang lebih tinggi sebagai akibat dari penggunaan bandwidth yang lebih tinggi. Untuk informasi selengkapnya tentang biaya transfer lintas wilayah, lihat Biaya Transfer AWS Data untuk Arsitektur Server dan Tanpa Server di Blog Jaringan Mitra. AWS
  - Karena jumlah byte yang lebih besar ditransfer dan karena jumlah byte yang lebih besar memerlukan waktu deserialisasi yang lebih tinggi, latensi keseluruhan dapat meningkat.

Menyiapkan database dan tabel di AWS Glue

Karena kemampuan inferensi skema bawaan konektor terbatas, Anda mungkin ingin menggunakannya AWS Glue untuk metadata. Untuk melakukan ini, Anda harus memiliki database dan tabel di AWS Glue. Untuk mengaktifkannya untuk digunakan dengan DynamoDB, Anda harus mengedit properti mereka.

Untuk mengedit properti database di AWS Glue konsol

1. Masuk ke AWS Management Console dan buka AWS Glue konsol di https://console.aws.amazon.com/glue/.
2. Di panel navigasi, perluas Katalog Data, lalu pilih Database.

Pada halaman Database, Anda dapat mengedit database yang ada, atau memilih Tambah database untuk membuatnya.

3. Dalam daftar database, pilih tautan untuk database yang ingin Anda edit.
4. Pilih Edit.
5. Pada halaman Perbarui database, di bawah pengaturan Database, untuk Lokasi, tambahkan string**`dynamo-db-flag`**. Kata kunci ini menunjukkan bahwa database berisi tabel yang digunakan konektor DynamoDB Athena untuk metadata tambahan dan diperlukan untuk database selain. AWS Glue `default` `dynamo-db-flag`Properti ini berguna untuk menyaring database dalam akun dengan banyak database.
6. Pilih Perbarui Database.

Untuk mengedit properti tabel di AWS Glue konsol

1. Masuk ke AWS Management Console dan buka AWS Glue konsol di https://console.aws.amazon.com/glue/.
2. Di panel navigasi, perluas Katalog Data, lalu pilih Tabel.
3. Pada halaman Tabel, dalam daftar tabel, pilih nama tertaut tabel yang ingin Anda edit.
4. Pilih Tindakan, Edit tabel.
5. Pada halaman Edit tabel, di bagian properti Tabel, tambahkan properti tabel berikut sesuai kebutuhan. Jika Anda menggunakan crawler AWS Glue DynamoDB, properti ini diatur secara otomatis.

   - dynamodb - String yang menunjukkan ke konektor DynamoDB Athena bahwa tabel dapat digunakan untuk metadata tambahan. Masukkan `dynamodb` string dalam properti tabel di bawah bidang yang disebut klasifikasi (sama persis).

     > **Note**
     >
     > Halaman Setel properti tabel yang merupakan bagian dari proses pembuatan tabel di AWS Glue konsol memiliki bagian Format data dengan bidang Klasifikasi. Anda tidak dapat masuk atau memilih `dynamodb` di sini. Sebagai gantinya, setelah Anda membuat tabel, ikuti langkah-langkah untuk mengedit tabel dan untuk memasukkan `classification` dan `dynamodb` sebagai pasangan kunci-nilai di bagian Properti tabel.

- SourceTable - Properti tabel opsional yang mendefinisikan nama tabel sumber di DynamoDB. Gunakan ini jika aturan penamaan AWS Glue tabel mencegah Anda membuat AWS Glue tabel dengan nama yang sama dengan tabel DynamoDB Anda. Misalnya, huruf kapital tidak diizinkan dalam nama AWS Glue tabel, tetapi mereka diizinkan dalam nama tabel DynamoDB.
- ColumnMapping - Properti tabel opsional yang mendefinisikan pemetaan nama kolom. Gunakan ini jika aturan penamaan AWS Glue kolom mencegah Anda membuat AWS Glue tabel dengan nama kolom yang sama dengan tabel DynamoDB Anda. Misalnya, huruf kapital tidak diizinkan dalam nama AWS Glue kolom tetapi diizinkan dalam nama kolom DynamoDB. Nilai properti diharapkan dalam format Col1 = Col1, Col2 = Col2. Perhatikan bahwa pemetaan kolom hanya berlaku untuk nama kolom tingkat atas dan bukan untuk bidang bersarang.
- defaultTimeZone- Properti tabel opsional yang diterapkan ke `date` atau `datetime` nilai yang tidak memiliki zona waktu eksplisit. Menyetel nilai ini adalah praktik yang baik untuk menghindari perbedaan antara zona waktu default sumber data dan zona waktu sesi Athena.
- datetimeFormatMapping- Properti tabel opsional yang menentukan `date` atau `datetime` format yang akan digunakan saat mengurai data dari kolom AWS Glue `date` atau tipe `timestamp` data. Jika properti ini tidak ditentukan, konektor mencoba menyimpulkan format ISO-8601. Jika konektor tidak dapat menyimpulkan `date` atau `datetime` memformat atau mengurai string mentah, maka nilainya dihilangkan dari hasilnya.

  `datetimeFormatMapping`Nilainya harus dalam formatcol1=someformat1,col2=someformat2. Berikut ini adalah beberapa contoh format:

  ```
  yyyyMMdd'T'HHmmss
  ddMMyyyy'T'HH:mm:ss
  ```

  Jika kolom Anda memiliki `date` atau `datetime` nilai tanpa zona waktu dan Anda ingin menggunakan kolom dalam `WHERE` klausa, atur `datetimeFormatMapping` properti untuk kolom tersebut.

6. Jika Anda menentukan kolom secara manual, pastikan Anda menggunakan tipe data yang sesuai. Jika Anda menggunakan crawler, validasi kolom dan jenis yang ditemukan crawler.
7. Pilih Simpan.

Izin yang Diperlukan

Untuk detail lengkap tentang kebijakan IAM yang diperlukan konektor ini, tinjau `Policies` bagian file athena-dynamodb.yaml. Daftar berikut merangkum izin yang diperlukan.

- Akses tulis Amazon S3 - Konektor memerlukan akses tulis ke lokasi di Amazon S3 untuk menumpahkan hasil dari kueri besar.
- Athena GetQueryExecution — Konektor menggunakan izin ini untuk gagal cepat ketika kueri Athena hulu telah dihentikan.
- AWS Glue Data Catalog- Konektor DynamoDB memerlukan akses baca saja ke untuk mendapatkan informasi AWS Glue Data Catalog skema.
- CloudWatch Log — Konektor memerlukan akses ke CloudWatch Log untuk menyimpan log.
- Akses baca DynamoDB — Konektor menggunakan `DescribeTable` operasi`ListSchemas`,,, `ListTablesQuery`, `Scan` dan API.

Kinerja

Konektor DynamoDB Athena mendukung pemindaian paralel dan mencoba menekan predikat sebagai bagian dari kueri DynamoDB-nya. Predikat kunci hash dengan nilai `X` yang berbeda menghasilkan panggilan `X` kueri ke DynamoDB. Semua skenario predikat lainnya menghasilkan `Y` jumlah panggilan pemindaian, di mana `Y` ditentukan secara heuristik berdasarkan ukuran tabel Anda dan throughput yang disediakan. Namun, memilih subset kolom terkadang menghasilkan runtime eksekusi kueri yang lebih lama.

`LIMIT`klausa dan predikat sederhana ditekan ke bawah dan dapat mengurangi jumlah data yang dipindai dan akan menyebabkan penurunan waktu eksekusi kueri.

Klausul LIMIT

`LIMIT N`Pernyataan mengurangi data yang dipindai oleh kueri. Dengan `LIMIT N` pushdown, konektor hanya mengembalikan `N` baris ke Athena.

Predikat

Predikat adalah ekspresi dalam `WHERE` klausa kueri SQL yang mengevaluasi nilai Boolean dan menyaring baris berdasarkan beberapa kondisi. Untuk meningkatkan fungsionalitas, dan untuk mengurangi jumlah data yang dipindai, konektor DynamoDB Athena dapat menggabungkan ekspresi ini dan mendorongnya langsung ke DynamoDB.

Operator konektor DynamoDB Athena berikut mendukung pushdown predikat:

- Boolean: DAN
- KESETARAAN: SAMA, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_NULL

Contoh pushdown gabungan

Untuk kemampuan kueri yang ditingkatkan, gabungkan jenis pushdown, seperti pada contoh berikut:

```
SELECT *
FROM my_table
WHERE col_a > 10 and col_b < 10
LIMIT 10
```

Untuk artikel tentang penggunaan pushdown predikat untuk meningkatkan kinerja dalam kueri federasi, termasuk DynamoDB, lihat Meningkatkan kueri federasi dengan pushdown predikat di Amazon Athena di Blog Big Data.AWS

Kueri passthrough

Konektor DynamoDB mendukung kueri passthrough dan menggunakan sintaks PartiQL. Operasi GetItemDynamoDB API tidak didukung. Untuk informasi tentang menanyakan DynamoDB menggunakan PartiQL, lihat pernyataan pilih PartiQL untuk DynamoDB di Panduan Pengembang Amazon DynamoDB.

Untuk menggunakan kueri passthrough dengan DynamoDB, gunakan sintaks berikut:

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

Contoh kueri passthrough DynamoDB berikut menggunakan PartiQL untuk mengembalikan daftar perangkat Fire TV Stick yang memiliki properti selambat-lambatnya 12/24/22. `DateWatched`

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

```
                    FROM WatchList
                    WHERE Devices.FireStick.DateWatched[0] > '12/24/22''
        ))
```

Pemecahan Masalah

Beberapa filter pada kolom kunci sortir

Pesan kesalahan: hanya KeyConditionExpressions boleh berisi satu kondisi per kunci

Penyebab: Masalah ini dapat terjadi di mesin Athena versi 3 dalam kueri yang memiliki filter batas bawah dan atas pada kolom kunci sortir DynamoDB. Karena DynamoDB tidak mendukung lebih dari satu kondisi filter pada kunci pengurutan, kesalahan muncul ketika konektor mencoba menekan kueri yang kedua kondisi diterapkan.

Solusi: Perbarui konektor ke versi 2023.11.1 atau yang lebih baru. Untuk petunjuk tentang memperbarui konektor, lihatMemperbarui konektor sumber data.

Biaya

Biaya untuk penggunaan konektor tergantung pada AWS sumber daya yang mendasari yang digunakan. Karena kueri yang menggunakan pemindaian dapat menggunakan sejumlah besar unit kapasitas baca (RCU), pertimbangkan informasi untuk harga Amazon DynamoDB dengan cermat.

Sumber daya tambahan

- Untuk pengenalan penggunaan konektor DynamoDB Amazon Athena, lihat Akses, kueri, dan bergabung dengan tabel Amazon DynamoDB menggunakan Athena dalam panduan Pola Panduan Preskriptif.AWS
- Untuk artikel tentang cara menggunakan konektor DynamoDB Athena untuk menanyakan data di DynamoDB dengan SQL dan memvisualisasikan wawasan di Amazon, QuickSight lihat posting AWS Blog Big Data Visualisasikan wawasan Amazon DynamoDB di Amazon menggunakan konektor DynamoDB Amazon Athena dan. QuickSight AWS Glue
- Untuk artikel tentang penggunaan konektor Amazon Athena DynamoDB dengan Amazon DynamoDB, Athena, dan Amazon untuk membuat dasbor tata kelola sederhana, lihat tabel QuickSight Amazon DynamoDB lintas akun Big Data Blog AWS post Query menggunakan Amazon Athena Federated Query.
- Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Google Amazon Athena BigQuery

Konektor Amazon Athena untuk Google BigQuerymemungkinkan Amazon Athena menjalankan kueri SQL pada data Google Anda. BigQuery

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Batasan

- Fungsi Lambda memiliki nilai batas waktu maksimum 15 menit. Setiap perpecahan mengeksekusi kueri BigQuery dan harus selesai dengan waktu yang cukup untuk menyimpan hasil agar Athena dapat dibaca. Jika fungsi Lambda habis waktu, kueri gagal.
- Google BigQuery adalah case sensitive. Konektor mencoba untuk memperbaiki kasus nama dataset dan nama tabel tetapi tidak melakukan koreksi kasus apa pun untuk ID proyek. Ini diperlukan karena Athena menurunkan semua metadata. Koreksi ini membuat banyak panggilan tambahan ke Google BigQuery.
- Tipe data biner tidak didukung.
- Karena BigQuery konkurensi Google dan batas kuota, konektor mungkin mengalami masalah batas kuota Google. Untuk menghindari masalah ini, dorong sebanyak mungkin kendala ke Google BigQuery . Untuk informasi tentang BigQuery kuota, lihat Kuota dan batasan dalam dokumentasi Google BigQuery .

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengonfigurasi konektor Google BigQuery .

- spill_bucket - Menentukan bucket Amazon S3 untuk data yang melebihi batas fungsi Lambda.
- spill_prefix — (Opsional) Default ke subfolder dalam nama yang ditentukan. `spill_bucket` `athena-federation-spill` Kami menyarankan Anda mengonfigurasi siklus hidup penyimpanan Amazon S3 di lokasi ini untuk menghapus tumpahan yang lebih lama dari jumlah hari atau jam yang telah ditentukan sebelumnya.

- spill_put_request_headers — (Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject {"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.
- kms_key_id — (Opsional) Secara default, data apa pun yang tumpah ke Amazon S3 dienkripsi menggunakan mode enkripsi yang diautentikasi AES-GCM dan kunci yang dihasilkan secara acak. Agar fungsi Lambda Anda menggunakan kunci enkripsi yang lebih kuat yang dihasilkan oleh KMS sepertia`7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, Anda dapat menentukan ID kunci KMS.
- disable_spill_encryption — (Opsional) Ketika diatur ke, menonaktifkan enkripsi tumpahan. `True` Defaultnya `False` sehingga data yang tumpah ke S3 dienkripsi menggunakan AES-GCM — baik menggunakan kunci yang dihasilkan secara acak atau KMS untuk menghasilkan kunci. Menonaktifkan enkripsi tumpahan dapat meningkatkan kinerja, terutama jika lokasi tumpahan Anda menggunakan enkripsi sisi server.
- gcp_project_id — ID proyek (bukan nama proyek) yang berisi kumpulan data yang harus dibaca konektor (misalnya,). `semiotic-primer-1234567`
- secret_manager_gcp_creds_name — Nama rahasia di dalamnya yang berisi kredensil Anda dalam AWS Secrets Manager format JSON (misalnya,). BigQuery `GoogleCloudPlatformCredentials`
- big_query_endpoint — (Opsional) URL dari endpoint pribadi. BigQuery Gunakan parameter ini saat Anda ingin mengakses BigQuery melalui titik akhir pribadi.

Perpecahan dan tampilan

Karena BigQuery konektor menggunakan BigQuery Storage Read API untuk menanyakan tabel, dan BigQuery Storage API tidak mendukung tampilan, konektor menggunakan BigQuery klien dengan satu split untuk tampilan.

Kinerja

Untuk menanyakan tabel, BigQuery konektor menggunakan BigQuery Storage Read API, yang menggunakan protokol berbasis RPC yang menyediakan akses cepat ke penyimpanan BigQuery terkelola. Untuk informasi selengkapnya tentang BigQuery Storage Read API, lihat Menggunakan BigQuery Storage Read API untuk membaca data tabel di dokumentasi Google Cloud.

Memilih subset kolom secara signifikan mempercepat runtime kueri dan mengurangi data yang dipindai. Konektor tunduk pada kegagalan kueri saat konkurensi meningkat, dan umumnya merupakan konektor yang lambat.

BigQuery Konektor Google Athena melakukan pushdown predikat untuk mengurangi data yang dipindai oleh kueri. `LIMIT`klausa, `ORDER BY` klausa, predikat sederhana, dan ekspresi kompleks didorong ke konektor untuk mengurangi jumlah data yang dipindai dan mengurangi waktu eksekusi kueri.

Klausul LIMIT

`LIMIT N`Pernyataan mengurangi data yang dipindai oleh kueri. Dengan `LIMIT N` pushdown, konektor hanya mengembalikan `N` baris ke Athena.

Kueri N teratas

`N`Kueri teratas menentukan urutan set hasil dan batas jumlah baris yang dikembalikan. Anda dapat menggunakan jenis kueri ini untuk menentukan nilai `N` maks teratas atau nilai `N` min teratas untuk kumpulan data Anda. Dengan `N` pushdown atas, konektor hanya mengembalikan baris yang `N` dipesan ke Athena.

Predikat

Predikat adalah ekspresi dalam `WHERE` klausa query SQL yang mengevaluasi nilai Boolean dan menyaring baris berdasarkan beberapa kondisi. BigQuery Konektor Google Athena dapat menggabungkan ekspresi ini dan mendorongnya langsung ke Google BigQuery untuk fungsionalitas yang ditingkatkan dan untuk mengurangi jumlah data yang dipindai.

Operator BigQuery konektor Google Athena berikut mendukung pushdown predikat:

- Boolean: DAN, ATAU, TIDAK
- KESETARAAN: SAMA, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Aritmatika: TAMBAHKAN, KURANGI, KALIKAN, BAGI, MODULUS, MENIADAKAN
- Lainnya: LIKE_PATTERN, IN

Contoh pushdown gabungan

Untuk kemampuan kueri yang ditingkatkan, gabungkan jenis pushdown, seperti pada contoh berikut:

```
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;
```

Kueri passthrough

BigQuery Konektor Google mendukung kueri passthrough. Kueri passthrough menggunakan fungsi tabel untuk mendorong kueri lengkap Anda ke sumber data untuk dieksekusi.

Untuk menggunakan kueri passthrough dengan Google BigQuery, Anda dapat menggunakan sintaks berikut:

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

Contoh kueri berikut mendorong kueri ke sumber data di Google BigQuery. Kueri memilih semua kolom dalam `customer` tabel, membatasi hasilnya menjadi 10.

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

Informasi lisensi

Proyek BigQuery konektor Google Amazon Athena dilisensikan di bawah Lisensi Apache-2.0.

Dengan menggunakan konektor ini, Anda mengakui penyertaan komponen pihak ketiga, daftar yang dapat ditemukan dalam file pom.xml untuk konektor ini, dan menyetujui persyaratan dalam masing-masing lisensi pihak ketiga yang disediakan dalam file LICENSE.txt di .com. GitHub

Sumber daya tambahan

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Penyimpanan Awan Google Amazon Athena

Konektor Amazon Athena Google Cloud Storage memungkinkan Amazon Athena menjalankan kueri pada file Parket dan CSV yang disimpan dalam bucket Google Cloud Storage (GCS). Setelah

mengelompokkan satu atau beberapa file Parket atau CSV dalam folder yang tidak dipartisi atau dipartisi dalam bucket GCS, Anda dapat mengaturnya dalam tabel database. AWS Glue

Jika Anda mengaktifkan Lake Formation di akun Anda, peran IAM untuk konektor Lambda federasi Athena yang Anda gunakan harus memiliki akses baca di AWS Serverless Application Repository Lake Formation ke. AWS Glue Data Catalog

Untuk artikel yang menunjukkan cara menggunakan Athena untuk menjalankan kueri pada file Parket atau CSV di bucket GCS, lihat posting Blog AWS Big Data Menggunakan Amazon Athena untuk menanyakan data yang disimpan di Google Cloud Platform.

Prasyarat

- Siapkan AWS Glue database dan tabel yang sesuai dengan bucket dan folder Anda di Google Cloud Storage. Untuk langkah-langkahnya, lihat Menyiapkan database dan tabel di AWS Glue nanti di dokumen ini.
- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Batasan

- Menulis operasi DDL tidak didukung.
- Batas Lambda yang relevan. Untuk informasi selengkapnya, lihat Kuota Lambda dalam Panduan Pengembang AWS Lambda .
- Saat ini, konektor hanya mendukung `VARCHAR` jenis untuk kolom partisi (`string`atau `varchar` dalam skema AWS Glue tabel). Jenis bidang partisi lainnya memunculkan kesalahan saat Anda menanyakannya di Athena.

Ketentuan

Istilah-istilah berikut berhubungan dengan konektor GCS.

- Handler — Penangan Lambda yang mengakses bucket GCS Anda. Handler bisa untuk metadata atau untuk catatan data.
- Metadata handler — Penangan Lambda yang mengambil metadata dari bucket GCS Anda.

- Record handler — Penangan Lambda yang mengambil catatan data dari bucket GCS Anda.
- Handler komposit — Penangan Lambda yang mengambil data metadata dan data dari bucket GCS Anda.

Tipe file yang didukung

Konektor GCS mendukung jenis file Parket dan CSV.

> **Note**
>
> Pastikan Anda tidak menempatkan file CSV dan Parket di bucket atau path GCS yang sama. Melakukannya dapat mengakibatkan kesalahan runtime ketika file Parket dicoba dibaca sebagai CSV atau sebaliknya.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengonfigurasi konektor GCS.

- spill_bucket - Menentukan bucket Amazon S3 untuk data yang melebihi batas fungsi Lambda.
- spill_prefix — (Opsional) Default ke subfolder dalam nama yang ditentukan. `spill_bucket` `athena-federation-spill` Kami menyarankan Anda mengonfigurasi siklus hidup penyimpanan Amazon S3 di lokasi ini untuk menghapus tumpahan yang lebih lama dari jumlah hari atau jam yang telah ditentukan sebelumnya.
- spill_put_request_headers — (Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject` `{"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.
- kms_key_id — (Opsional) Secara default, data apa pun yang tumpah ke Amazon S3 dienkripsi menggunakan mode enkripsi yang diautentikasi AES-GCM dan kunci yang dihasilkan secara acak. Agar fungsi Lambda Anda menggunakan kunci enkripsi yang lebih kuat yang dihasilkan oleh KMS seperti`a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, Anda dapat menentukan ID kunci KMS.
- disable_spill_encryption — (Opsional) Ketika diatur ke, menonaktifkan enkripsi tumpahan. `True` Defaultnya `False` sehingga data yang tumpah ke S3 dienkripsi menggunakan AES-GCM — baik menggunakan kunci yang dihasilkan secara acak atau KMS untuk menghasilkan kunci. Menonaktifkan enkripsi tumpahan dapat meningkatkan kinerja, terutama jika lokasi tumpahan Anda menggunakan enkripsi sisi server.

- secret_manager_gcp_creds_name — Nama rahasia yang berisi kredensi GCS Anda dalam AWS Secrets Manager format JSON (misalnya,). `GoogleCloudPlatformCredentials`

Menyiapkan database dan tabel di AWS Glue

Karena kemampuan inferensi skema bawaan dari konektor GCS terbatas, kami sarankan Anda menggunakannya AWS Glue untuk metadata Anda. Prosedur berikut menunjukkan cara membuat database dan tabel di mana Anda AWS Glue dapat mengakses dari Athena.

Membuat basis data di AWS Glue

Anda dapat menggunakan AWS Glue konsol untuk membuat database untuk digunakan dengan konektor GCS.

Untuk membuat database di AWS Glue

1. Masuk ke AWS Management Console dan buka AWS Glue konsol di https://console.aws.amazon.com/glue/.
2. Dari panel navigasi, pilih Databases.
3. Pilih Add database (Tambahkan basis data).
4. Untuk Nama, masukkan nama untuk database yang ingin Anda gunakan dengan konektor GCS.
5. Untuk Lokasi, tentukan Lokasi `s3://google-cloud-storage-flag.` ini memberi tahu konektor GCS bahwa AWS Glue database berisi tabel untuk data GCS yang akan ditanyakan di Athena. Konektor mengenali database di Athena yang memiliki bendera ini dan mengabaikan database yang tidak.
6. Pilih Buat basis data.

Membuat tabel di AWS Glue

Sekarang Anda dapat membuat tabel untuk database. Saat Anda membuat AWS Glue tabel untuk digunakan dengan konektor GCS, Anda harus menentukan metadata tambahan.

Untuk membuat tabel di AWS Glue konsol

1. Di AWS Glue konsol, dari panel navigasi, pilih Tabel.
2. Pada halaman Tabel, pilih Tambahkan tabel.
3. Pada halaman Set table properties, masukkan informasi berikut.

- Nama — Nama unik untuk tabel.
- Database — Pilih AWS Glue database yang Anda buat untuk konektor GCS.
- Sertakan jalur — Di bagian Penyimpanan data, untuk jalur Sertakan, masukkan lokasi URI untuk GCS yang diawali oleh `gs://` (misalnya,`gs://gcs_table/data/`). Jika Anda memiliki satu atau lebih folder partisi, jangan sertakan mereka di jalur.

  > **Note**
  >
  > Saat Anda memasukkan jalur non `s3://` tabel, AWS Glue konsol menunjukkan kesalahan. Anda dapat mengabaikan kesalahan ini. Tabel akan berhasil dibuat.

- Format data — Untuk Klasifikasi, pilih CSV atau Parket.

4. Pilih Selanjutnya.
5. Pada halaman Pilih atau tentukan skema, mendefinisikan skema tabel sangat disarankan, tetapi tidak wajib. Jika Anda tidak mendefinisikan skema, konektor GCS mencoba menyimpulkan skema untuk Anda.

   Lakukan salah satu hal berikut ini:

   - Jika Anda ingin konektor GCS mencoba menyimpulkan skema untuk Anda, pilih Berikutnya, lalu pilih Buat.
   - Untuk menentukan skema sendiri, ikuti langkah-langkah di bagian selanjutnya.

Mendefinisikan skema tabel di AWS Glue

Mendefinisikan skema tabel AWS Glue membutuhkan lebih banyak langkah tetapi memberi Anda kontrol yang lebih besar atas proses pembuatan tabel.

Untuk menentukan skema untuk tabel Anda di AWS Glue

1. Pada halaman Pilih atau tentukan skema, pilih Tambah.
2. Gunakan kotak dialog Tambahkan entri skema untuk memberikan nama kolom dan tipe data.
3. Untuk menunjuk kolom sebagai kolom partisi, pilih opsi Set as partition key.
4. Pilih Simpan untuk menyimpan kolom.
5. Pilih Tambah untuk menambahkan kolom lain.
6. Setelah selesai menambahkan kolom, pilih Berikutnya.

7. Pada halaman Tinjau dan buat, tinjau tabel, lalu pilih Buat.
8. Jika skema Anda berisi informasi partisi, ikuti langkah-langkah di bagian berikutnya untuk menambahkan pola partisi ke properti tabel di AWS Glue.

Menambahkan pola partisi ke properti tabel di AWS Glue

Jika bucket GCS Anda memiliki partisi, Anda harus menambahkan pola partisi ke properti tabel di. AWS Glue

Untuk menambahkan informasi partisi ke properti tabel AWS Glue

1. Pada halaman detail untuk tabel yang Anda buat AWS Glue, pilih Tindakan, Edit tabel.
2. Pada halaman Edit tabel, gulir ke bawah ke bagian Properti tabel.
3. Pilih Tambah untuk menambahkan kunci partisi.
4. Untuk Kunci, masukkan **`partition.pattern`**. Kunci ini mendefinisikan pola path folder.
5. Untuk Nilai, masukkan pola jalur folder seperti**`StateName=${statename}/ZipCode=${zipcode}/`**, di mana **`statename`** dan **`zipcode`** tertutup oleh **`${}`** adalah nama kolom partisi. Konektor GCS mendukung skema partisi Hive dan non-Hive.
6. Setelah selesai, pilih Simpan.
7. Untuk melihat properti tabel yang baru saja Anda buat, pilih tab Advanced properties.

Pada titik ini, Anda dapat menavigasi ke konsol Athena. Database dan tabel yang Anda buat AWS Glue tersedia untuk kueri di Athena.

Dukungan tipe data

Tabel berikut menunjukkan tipe data yang didukung untuk CSV dan untuk Parket.

CSV

Sifat data	Tipe Data yang Disimpulkan
Data terlihat seperti angka	BIGINT
Data terlihat seperti string	VARCHAR

Sifat data	Tipe Data yang Disimpulkan
Data terlihat seperti floating point (float, double, atau desimal)	DOUBLE
Data terlihat seperti Tanggal	Stempel Waktu
Data yang berisi nilai benar/salah	BOOL

Parquet

PARKET	Athena (Panah)
BINARY	VARCHAR
BOOLEAN	BOOL
DOUBLE	DOUBLE
ENUM	VARCHAR
FIXED_LEN _BYTE_ARRAY	DECIMAL
FLOAT	FLOAT (32-bit)
INT32	1. INT32 2. DATEDAY (ketika tipe logis kolom Parket adalah DATE)
INT64	1. INT64 2. TIMESTAMP (ketika tipe logis kolom Parket adalah TIMESTAMP)
INT96	Stempel Waktu
PETA	PETA
STRUCT	STRUCT
DAFTAR	DAFTAR

Izin yang Diperlukan

Untuk detail lengkap tentang kebijakan IAM yang diperlukan konektor ini, tinjau `Policies` bagian file athena-gcs.yaml. Daftar berikut merangkum izin yang diperlukan.

- Akses tulis Amazon S3 - Konektor memerlukan akses tulis ke lokasi di Amazon S3 untuk menumpahkan hasil dari kueri besar.
- Athena GetQueryExecution — Konektor menggunakan izin ini untuk gagal cepat ketika kueri Athena hulu telah dihentikan.
- AWS Glue Data Catalog- Konektor GCS memerlukan akses baca saja ke AWS Glue Data Catalog untuk mendapatkan informasi skema.
- CloudWatch Log — Konektor memerlukan akses ke CloudWatch Log untuk menyimpan log.

Kinerja

Ketika skema tabel berisi bidang partisi dan properti `partition.pattern` tabel dikonfigurasi dengan benar, Anda dapat menyertakan bidang partisi dalam `WHERE` klausa kueri Anda. Untuk kueri semacam itu, konektor GCS menggunakan kolom partisi untuk memperbaiki jalur folder GCS dan menghindari pemindaian file yang tidak dibutuhkan di folder GCS.

Untuk kumpulan data Parket, memilih subset kolom menghasilkan lebih sedikit data yang dipindai. Ini biasanya menghasilkan runtime eksekusi kueri yang lebih pendek ketika proyeksi kolom diterapkan.

Untuk kumpulan data CSV, proyeksi kolom tidak didukung dan tidak mengurangi jumlah data yang dipindai.

`LIMIT`klausa mengurangi jumlah data yang dipindai, tetapi jika Anda tidak memberikan predikat, Anda harus mengharapkan `SELECT` kueri dengan `LIMIT` klausa untuk memindai setidaknya 16 MB data. Konektor GCS memindai lebih banyak data untuk kumpulan data yang lebih besar daripada kumpulan data yang lebih kecil, terlepas dari klausa yang diterapkan. `LIMIT` Misalnya, kueri `SELECT * LIMIT 10000` memindai lebih banyak data untuk kumpulan data dasar yang lebih besar daripada yang lebih kecil.

Informasi lisensi

Dengan menggunakan konektor ini, Anda mengakui penyertaan komponen pihak ketiga, daftar yang dapat ditemukan dalam file pom.xml untuk konektor ini, dan menyetujui persyaratan dalam masing-masing lisensi pihak ketiga yang disediakan dalam file LICENSE.txt di .com. GitHub

Sumber daya tambahan

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Amazon Athena HBase

Konektor Amazon Athena HBase memungkinkan Amazon Athena berkomunikasi dengan instans Apache HBase Anda sehingga Anda dapat menanyakan data HBase Anda dengan SQL.

Tidak seperti penyimpanan data relasional tradisional, koleksi HBase tidak memiliki skema yang ditetapkan. HBase tidak memiliki toko metadata. Setiap entri dalam koleksi HBase dapat memiliki bidang dan tipe data yang berbeda.

Konektor HBase mendukung dua mekanisme untuk menghasilkan informasi skema tabel: inferensi skema dasar dan metadata. AWS Glue Data Catalog

Inferensi skema adalah default. Opsi ini memindai sejumlah kecil dokumen dalam koleksi Anda, membentuk gabungan semua bidang, dan memaksa bidang yang memiliki tipe data yang tidak tumpang tindih. Opsi ini berfungsi dengan baik untuk koleksi yang sebagian besar memiliki entri seragam.

Untuk koleksi dengan variasi tipe data yang lebih besar, konektor mendukung pengambilan metadata dari file. AWS Glue Data Catalog Jika konektor melihat AWS Glue database dan tabel yang cocok dengan namespace HBase dan nama koleksi Anda, ia mendapatkan informasi skema dari tabel yang sesuai. AWS Glue Saat Anda membuat AWS Glue tabel, kami sarankan Anda menjadikannya superset dari semua bidang yang mungkin ingin Anda akses dari koleksi HBase Anda.

Jika Anda mengaktifkan Lake Formation di akun Anda, peran IAM untuk konektor Lambda federasi Athena yang Anda gunakan harus memiliki akses baca di AWS Serverless Application Repository Lake Formation ke. AWS Glue Data Catalog

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengkonfigurasi konektor HBase.

- spill_bucket - Menentukan bucket Amazon S3 untuk data yang melebihi batas fungsi Lambda.
- spill_prefix — (Opsional) Default ke subfolder dalam nama yang ditentukan. `spill_bucket` `athena-federation-spill` Kami menyarankan Anda mengonfigurasi siklus hidup penyimpanan Amazon S3 di lokasi ini untuk menghapus tumpahan yang lebih lama dari jumlah hari atau jam yang telah ditentukan sebelumnya.
- spill_put_request_headers — (Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject` `{"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.
- kms_key_id — (Opsional) Secara default, data apa pun yang tumpah ke Amazon S3 dienkripsi menggunakan mode enkripsi yang diautentikasi AES-GCM dan kunci yang dihasilkan secara acak. Agar fungsi Lambda Anda menggunakan kunci enkripsi yang lebih kuat yang dihasilkan oleh KMS sepertia`7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, Anda dapat menentukan ID kunci KMS.
- disable_spill_encryption — (Opsional) Ketika diatur ke, menonaktifkan enkripsi tumpahan. `True` Defaultnya `False` sehingga data yang tumpah ke S3 dienkripsi menggunakan AES-GCM — baik menggunakan kunci yang dihasilkan secara acak atau KMS untuk menghasilkan kunci. Menonaktifkan enkripsi tumpahan dapat meningkatkan kinerja, terutama jika lokasi tumpahan Anda menggunakan enkripsi sisi server.
- disable_glue — (Opsional) Jika ada dan disetel ke true, konektor tidak mencoba untuk mengambil metadata tambahan dari. AWS Glue
- glue_catalog - (Opsional) Gunakan opsi ini untuk menentukan katalog lintas akun. AWS Glue Secara default, konektor mencoba untuk mendapatkan metadata dari akunnya sendiri AWS Glue .
- default_hbase - Jika ada, menentukan string koneksi HBase untuk digunakan ketika tidak ada variabel lingkungan khusus katalog.
- enable_case_insensitive_match — (Opsional) Saat, melakukan pencarian yang tidak peka huruf besar/kecil terhadap nama tabel di `true` HBase. Nilai default-nya `false`. Gunakan jika kueri Anda berisi nama tabel huruf besar.

Menentukan string koneksi

Anda dapat memberikan satu atau beberapa properti yang menentukan detail koneksi HBase untuk instance HBase yang Anda gunakan dengan konektor. Untuk melakukan ini, tetapkan variabel lingkungan Lambda yang sesuai dengan nama katalog yang ingin Anda gunakan di Athena. Misalnya, Anda ingin menggunakan kueri berikut untuk menanyakan dua instance HBase yang berbeda dari Athena:

```
SELECT * FROM "hbase_instance_1".database.table
```

```
SELECT * FROM "hbase_instance_2".database.table
```

Sebelum Anda dapat menggunakan dua pernyataan SQL ini, Anda harus menambahkan dua variabel lingkungan ke fungsi hbase_instance_1 Lambda Anda: dan. hbase_instance_2 Nilai untuk masing-masing harus berupa string koneksi HBase dalam format berikut:

```
master_hostname:hbase_port:zookeeper_port
```

Menggunakan rahasia

Anda dapat secara opsional menggunakan AWS Secrets Manager sebagian atau seluruh nilai untuk detail string koneksi Anda. Untuk menggunakan fitur Kueri Federasi Athena dengan Secrets Manager, VPC yang terhubung ke fungsi Lambda Anda harus memiliki akses internet atau titik akhir VPC untuk terhubung ke Secrets Manager.

Jika Anda menggunakan sintaks ${my_secret} untuk memasukkan nama rahasia dari Secrets Manager di string koneksi Anda, konektor menggantikan nama rahasia dengan nama pengguna dan nilai kata sandi Anda dari Secrets Manager.

Misalnya, Anda menyetel variabel lingkungan Lambda hbase_instance_1 ke nilai berikut:

```
${hbase_host_1}:${hbase_master_port_1}:${hbase_zookeeper_port_1}
```

Athena Query Federation SDK secara otomatis mencoba untuk mengambil rahasia bernama dari Secrets hbase_instance_1_creds Manager dan menyuntikkan nilai tersebut sebagai pengganti. ${hbase_instance_1_creds} Setiap bagian dari string koneksi yang dilampirkan oleh kombinasi ${ } karakter ditafsirkan sebagai rahasia dari Secrets Manager. Jika Anda menentukan nama rahasia yang tidak dapat ditemukan konektor di Secrets Manager, konektor tidak menggantikan teks.

Menyiapkan database dan tabel di AWS Glue

Inferensi skema bawaan konektor hanya mendukung nilai yang diserialkan dalam HBase sebagai string (misalnya,). String.valueOf(int) Karena kemampuan inferensi skema bawaan konektor terbatas, Anda mungkin ingin menggunakannya AWS Glue untuk metadata sebagai gantinya. Untuk mengaktifkan AWS Glue tabel untuk digunakan dengan HBase, Anda harus memiliki AWS Glue database dan tabel dengan nama yang cocok dengan namespace HBase dan tabel yang ingin

Anda berikan metadata tambahan. Penggunaan konvensi penamaan keluarga kolom HBase adalah opsional tetapi tidak diperlukan.

Untuk menggunakan AWS Glue tabel untuk metadata tambahan

1. Saat Anda mengedit tabel dan database di AWS Glue konsol, tambahkan properti tabel berikut:

   - hbase-metadata-flag- Properti ini menunjukkan ke konektor HBase bahwa konektor dapat menggunakan tabel untuk metadata tambahan. Anda dapat memberikan nilai apa pun `hbase-metadata-flag` selama `hbase-metadata-flag` properti hadir dalam daftar properti tabel.
   - hbase-native-storage-flag— Gunakan tanda ini untuk mengaktifkan dua mode serialisasi nilai yang didukung oleh konektor. Secara default, ketika bidang ini tidak ada, konektor mengasumsikan semua nilai disimpan dalam HBase sebagai string. Dengan demikian ia akan mencoba untuk mengurai tipe data seperti`INT`,`BIGINT`, dan `DOUBLE` dari HBase sebagai string. Jika bidang ini diatur dengan nilai apa pun pada tabel AWS Glue, konektor beralih ke mode penyimpanan “asli” dan mencoba membaca`INT`,`BIGINT`,`BIT`, dan `DOUBLE` sebagai byte dengan menggunakan fungsi berikut:

     ```
     ByteBuffer.wrap(value).getInt()
     ByteBuffer.wrap(value).getLong()
     ByteBuffer.wrap(value).get()
     ByteBuffer.wrap(value).getDouble()
     ```

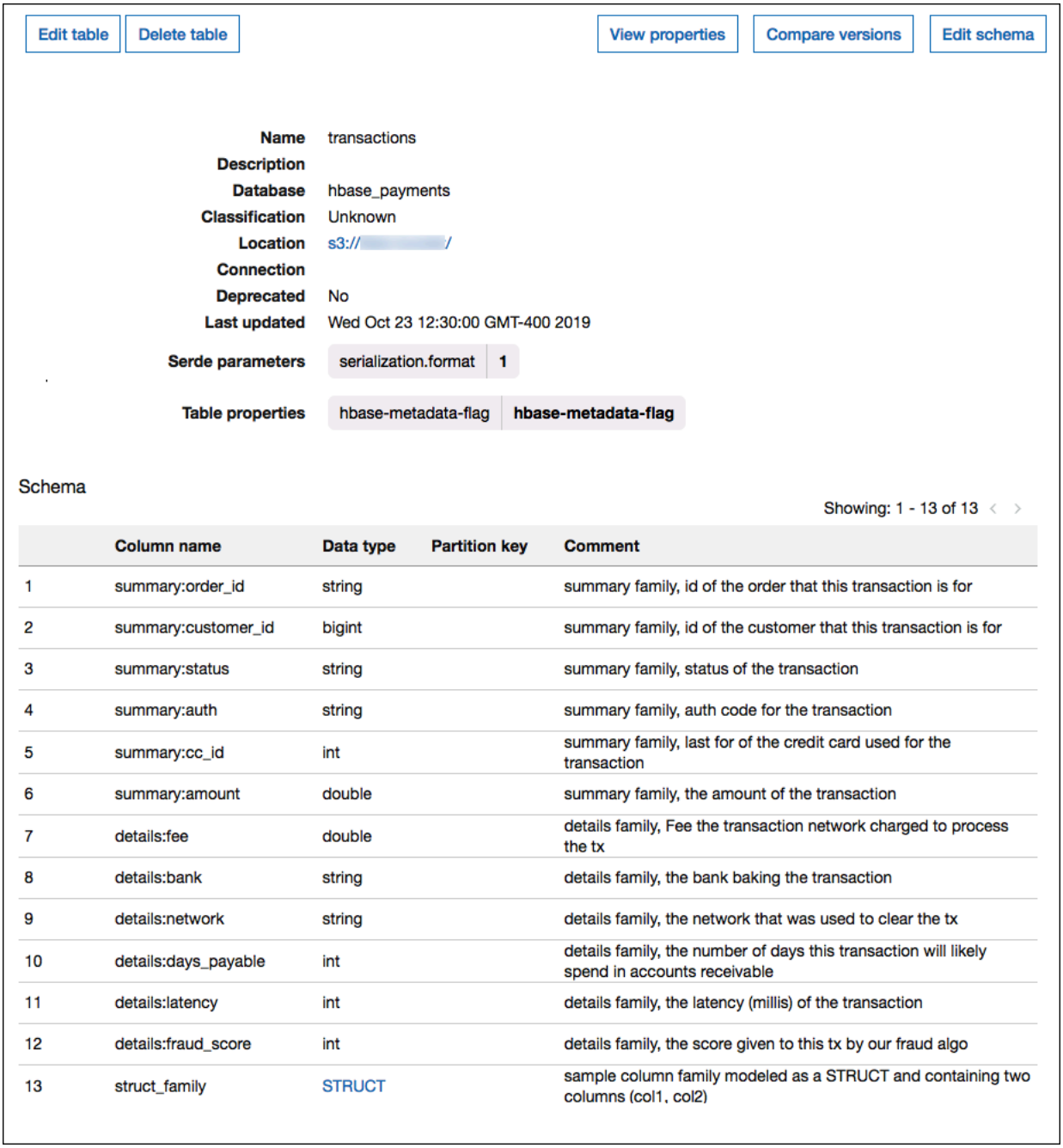
2. Pastikan Anda menggunakan tipe data yang sesuai AWS Glue seperti yang tercantum dalam dokumen ini.

Pemodelan keluarga kolom

Konektor Athena HBase mendukung dua cara untuk memodelkan keluarga kolom HBase: penamaan yang sepenuhnya memenuhi syarat (diratakan) seperti, atau menggunakan objek. `family:column` `STRUCT`

Dalam `STRUCT` model, nama `STRUCT` bidang harus sesuai dengan keluarga kolom, dan anak-anak `STRUCT` harus cocok dengan nama kolom keluarga. Namun, karena predikat push down dan pembacaan kolom belum sepenuhnya didukung untuk tipe kompleks seperti`STRUCT`, penggunaan saat ini tidak `STRUCT` disarankan.

Gambar berikut menunjukkan tabel yang dikonfigurasi di AWS Glue yang menggunakan kombinasi dari dua pendekatan.



	Column name	Data type	Partition key	Comment
1	summary:order_id	string		summary family, id of the order that this transaction is for
2	summary:customer_id	bigint		summary family, id of the customer that this transaction is for
3	summary:status	string		summary family, status of the transaction
4	summary:auth	string		summary family, auth code for the transaction
5	summary:cc_id	int		summary family, last for of the credit card used for the transaction
6	summary:amount	double		summary family, the amount of the transaction
7	details:fee	double		details family, Fee the transaction network charged to process the tx
8	details:bank	string		details family, the bank baking the transaction
9	details:network	string		details family, the network that was used to clear the tx
10	details:days_payable	int		details family, the number of days this transaction will likely spend in accounts receivable
11	details:latency	int		details family, the latency (millis) of the transaction
12	details:fraud_score	int		details family, the score given to this tx by our fraud algo
13	struct_family	STRUCT		sample column family modeled as a STRUCT and containing two columns (col1, col2)

Dukungan tipe data

Konektor mengambil semua nilai HBase sebagai tipe byte dasar. Kemudian, berdasarkan bagaimana Anda mendefinisikan tabel Anda di Katalog AWS Glue Data, itu memetakan nilai ke salah satu tipe data Apache Arrow dalam tabel berikut.

AWS Glue tipe data	Tipe data Apache Arrow
int	INT
bigint	BIGINT
double	FLOAT8
float	FLOAT4
boolean	BIT
biner	VARBINARY
string	VARCHAR

> **Note**
>
> Jika Anda tidak menggunakan AWS Glue untuk melengkapi metadata Anda, inferensi skema konektor hanya menggunakan tipe `BIGINT` data,, dan. `FLOAT8` `VARCHAR`

Izin yang Diperlukan

Untuk detail lengkap tentang kebijakan IAM yang diperlukan konektor ini, tinjau `Policies` bagian file athena-hbase.yaml. Daftar berikut merangkum izin yang diperlukan.

- Akses tulis Amazon S3 - Konektor memerlukan akses tulis ke lokasi di Amazon S3 untuk menumpahkan hasil dari kueri besar.
- Athena GetQueryExecution — Konektor menggunakan izin ini untuk gagal cepat ketika kueri Athena hulu telah dihentikan.
- AWS Glue Data Catalog- Konektor HBase memerlukan akses baca saja ke AWS Glue Data Catalog untuk mendapatkan informasi skema.
- CloudWatch Log — Konektor memerlukan akses ke CloudWatch Log untuk menyimpan log.
- AWS Secrets Manager akses baca - Jika Anda memilih untuk menyimpan detail titik akhir HBase di Secrets Manager, Anda harus memberikan akses konektor ke rahasia tersebut.

- Akses VPC — Konektor memerlukan kemampuan untuk memasang dan melepaskan antarmuka ke VPC Anda sehingga dapat terhubung dengannya dan berkomunikasi dengan instans HBase Anda.

Kinerja

Konektor Athena HBase mencoba memparalelkan kueri terhadap instance HBase Anda dengan membaca setiap server wilayah secara paralel. Konektor Athena HBase melakukan pushdown predikat untuk mengurangi data yang dipindai oleh kueri.

Fungsi Lambda juga melakukan pushdown proyeksi untuk mengurangi data yang dipindai oleh kueri. Namun, memilih subset kolom terkadang menghasilkan runtime eksekusi kueri yang lebih lama. `LIMIT`klausa mengurangi jumlah data yang dipindai, tetapi jika Anda tidak memberikan predikat, Anda harus mengharapkan `SELECT` kueri dengan `LIMIT` klausa untuk memindai setidaknya 16 MB data.

HBase rentan terhadap kegagalan kueri dan waktu eksekusi kueri variabel. Anda mungkin harus mencoba lagi pertanyaan Anda beberapa kali agar berhasil. Konektor HBase tahan terhadap pelambatan karena konkurensi.

Kueri passthrough

Konektor HBase mendukung kueri passthrough dan berbasis NoSQL. Untuk informasi tentang kueri Apache HBase menggunakan pemfilteran, lihat Menyaring bahasa dalam dokumentasi Apache.

Untuk menggunakan kueri passthrough dengan HBase, gunakan sintaks berikut:

```
SELECT * FROM TABLE(
        system.query(
            database => 'database_name',
            collection => 'collection_name',
            filter => '{query_syntax}'
        ))
```

Berikut contoh filter query passthrough HBase untuk karyawan berusia 24 atau 30 dalam `employee` koleksi database. `default`

```
SELECT * FROM TABLE(
        system.query(
            DATABASE => 'default',
```

```
           COLLECTION => 'employee',
           FILTER => 'SingleColumnValueFilter(''personaldata'', ''age'', =,
''binary:30'')' ||
                     ' OR SingleColumnValueFilter(''personaldata'', ''age'', =,
''binary:24'')'
       ))
```

Informasi lisensi

Proyek konektor Amazon Athena HBase dilisensikan di bawah Lisensi Apache-2.0.

Sumber daya tambahan

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Amazon Athena Hortonworks

Konektor Amazon Athena untuk Hortonworks memungkinkan Amazon Athena menjalankan kueri SQL pada platform data Cloudera Hortonworks. Konektor mengubah kueri Athena SQL Anda menjadi sintaks HiveQL yang setara.

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Batasan

- Menulis operasi DDL tidak didukung.
- Dalam pengaturan multiplexer, bucket tumpahan dan awalan dibagikan di semua instance database.
- Batas Lambda yang relevan. Untuk informasi selengkapnya, lihat Kuota Lambda dalam Panduan Pengembang AWS Lambda .

Ketentuan

Istilah-istilah berikut berhubungan dengan konektor Hortonworks Hive.

- Instans database — Instance apa pun dari database yang digunakan di tempat, di Amazon EC2, atau di Amazon RDS.
- Handler - Handler Lambda yang mengakses instance database Anda. Handler bisa untuk metadata atau untuk catatan data.
- Metadata handler — Penangan Lambda yang mengambil metadata dari instance database Anda.
- Record handler - Handler Lambda yang mengambil catatan data dari instance database Anda.
- Composite handler — Handler Lambda yang mengambil data metadata dan data dari instance database Anda.
- Properti atau parameter - Properti database yang digunakan oleh penangan untuk mengekstrak informasi database. Anda mengonfigurasi properti ini sebagai variabel lingkungan Lambda.
- Connection String — Sebuah string teks yang digunakan untuk membuat koneksi ke instance database.
- Katalog —AWS Glue Non-katalog yang terdaftar di Athena yang merupakan awalan yang diperlukan untuk properti. `connection_string`
- Multiplexing handler - Handler Lambda yang dapat menerima dan menggunakan beberapa koneksi database.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengonfigurasi konektor Hortonworks Hive.

String koneksi

Gunakan string koneksi JDBC dalam format berikut untuk terhubung ke instance database.

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

Menggunakan handler multiplexing

Anda dapat menggunakan multiplexer untuk terhubung ke beberapa instance database dengan satu fungsi Lambda. Permintaan dirutekan dengan nama katalog. Gunakan kelas berikut di Lambda.

Handler	Kelas
Handler komposit	`HiveMuxCompositeHandler`
Penangan metadata	`HiveMuxMetadataHandler`

Handler	Kelas
Rekam handler	`HiveMuxRecordHandler`

Parameter handler multiplexing

Parameter	Deskripsi
`$`*`catalog`*`_connection_string`	Wajib. Sebuah string koneksi instance database. Awalan variabel lingkungan dengan nama katalog yang digunakan di Athena. Misalnya, jika katalog terdaftar di Athena adalah`myhivecatalog` , maka nama variabel lingkungan adalah. `myhivecatalog_connection_string`
`default`	Wajib. String koneksi default. String ini digunakan ketika katalog `lambda:${` *`}AWS_LAMBDA_FUNCTION_NAME`* .

Contoh properti berikut adalah untuk fungsi Hive MUX Lambda yang mendukung dua instance database: `hive1` (default), dan. `hive2`

Properti	Nilai
`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`

Memberikan kredensi

Untuk memberikan nama pengguna dan kata sandi untuk database Anda dalam string koneksi JDBC Anda, Anda dapat menggunakan properti string koneksi atau. AWS Secrets Manager

- Connection String - Nama pengguna dan kata sandi dapat ditentukan sebagai properti dalam string koneksi JDBC.

  > **Important**
  >
  > Sebagai praktik keamanan terbaik, jangan gunakan kredensyal hardcode dalam variabel lingkungan atau string koneksi Anda. Untuk informasi tentang memindahkan rahasia hardcode Anda AWS Secrets Manager, lihat Memindahkan rahasia hardcode ke AWS Secrets Manager dalam Panduan Pengguna.AWS Secrets Manager

- AWS Secrets Manager— Untuk menggunakan fitur Query Federasi Athena dengan, VPC AWS Secrets Manager yang terhubung ke fungsi Lambda Anda harus memiliki akses internet atau titik akhir VPC untuk terhubung ke Secrets Manager.

  Anda dapat memasukkan nama rahasia AWS Secrets Manager dalam string koneksi JDBC Anda. Konektor menggantikan nama rahasia dengan `password` nilai `username` dan dari Secrets Manager.

  Untuk instans database Amazon RDS, dukungan ini terintegrasi dengan erat. Jika Anda menggunakan Amazon RDS, kami sangat menyarankan penggunaan AWS Secrets Manager dan rotasi kredensyal. Jika database Anda tidak menggunakan Amazon RDS, simpan kredensialnya sebagai JSON dalam format berikut:

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

Contoh string koneksi dengan nama rahasia

String berikut memiliki nama rahasia`${Test/RDS/hive1host}`.

```
hive://jdbc:hive2://hive1host:10000/default?...&${Test/RDS/hive1host}&...
```

Konektor menggunakan nama rahasia untuk mengambil rahasia dan memberikan nama pengguna dan kata sandi, seperti pada contoh berikut.

```
hive://jdbc:hive2://hive1host:10000/default?...&UID=sample2&PWD=sample2&...
```

Saat ini, konektor Hortonworks Hive mengenali properti dan JDBC. `UID` `PWD`

Menggunakan handler koneksi tunggal

Anda dapat menggunakan metadata koneksi tunggal berikut dan penangan rekaman untuk terhubung ke satu instance Hortonworks Hive.

Jenis handler	Kelas
Handler komposit	`HiveCompositeHandler`
Penangan metadata	`HiveMetadataHandler`
Rekam handler	`HiveRecordHandler`

Parameter handler koneksi tunggal

Parameter	Deskripsi
`default`	Wajib. String koneksi default.

Penangan koneksi tunggal mendukung satu instance database dan harus menyediakan parameter string `default` koneksi. Semua string koneksi lainnya diabaikan.

Properti contoh berikut adalah untuk satu contoh Hortonworks Hive didukung oleh fungsi Lambda.

Properti	Nilai
`default`	`hive://jdbc:hive2://hive1host:10000/default?secret=${Test/RDS/hive1host}`

Parameter tumpahan

Lambda SDK dapat menumpahkan data ke Amazon S3. Semua instance database yang diakses oleh fungsi Lambda yang sama tumpah ke lokasi yang sama.

Parameter	Deskripsi
`spill_bucket`	Wajib. Nama ember tumpahan.

Parameter	Deskripsi
`spill_prefix`	Wajib. Tumpahkan key prefix bucket.
`spill_put_request_headers`	(Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject {"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.

Dukungan tipe data

Tabel berikut menunjukkan tipe data yang sesuai untuk JDBC, Hortonworks Hive, dan Arrow.

JDBC	Sarang Hortonworks	Panah
Boolean	Boolean	Bit
Bilangan Bulat	TINYINT	Mungil
Pendek	SMALLINT	berkulit kecil
Bilangan Bulat	INT	Int
Long	BIGINT	Bigint
float	mengapung4	Mengapung4
Ganda	mengapung8	Mengapung8
Tanggal	tanggal	DateDay
Stempel Waktu	timestamp	DateMilli
String	VARCHAR	Varchar
Byte	byte	Varbiner
BigDecimal	Decimal	Decimal

JDBC	Sarang Hortonworks	Panah
ARRAY	N/A (lihat catatan)	Daftar

> **Note**
>
> Saat ini, Hortonworks Hive tidak mendukung jenis agregat`ARRAY`,,, `MAP` atau. `STRUCT` `UNIONTYPE` Kolom tipe agregat diperlakukan sebagai `VARCHAR` kolom dalam SQL.

Partisi dan split

Partisi digunakan untuk menentukan cara menghasilkan split untuk konektor. Athena membangun kolom sintetis tipe `varchar` yang mewakili skema partisi untuk tabel untuk membantu konektor menghasilkan split. Konektor tidak mengubah definisi tabel yang sebenarnya.

Kinerja

Hortonworks Hive mendukung partisi statis. Konektor Athena Hortonworks Hive dapat mengambil data dari partisi ini secara paralel. Jika Anda ingin menanyakan kumpulan data yang sangat besar dengan distribusi partisi seragam, partisi statis sangat disarankan. Memilih subset kolom secara signifikan mempercepat runtime kueri dan mengurangi data yang dipindai. Konektor Hortonworks Hive tahan terhadap pelambatan karena konkurensi.

Konektor Athena Hortonworks Hive melakukan pushdown predikat untuk mengurangi data yang dipindai oleh kueri. `LIMIT`klausa, predikat sederhana, dan ekspresi kompleks didorong ke konektor untuk mengurangi jumlah data yang dipindai dan mengurangi waktu eksekusi kueri.

Klausul LIMIT

`LIMIT N`Pernyataan mengurangi data yang dipindai oleh kueri. Dengan `LIMIT N` pushdown, konektor hanya mengembalikan `N` baris ke Athena.

Predikat

Predikat adalah ekspresi dalam `WHERE` klausa query SQL yang mengevaluasi nilai Boolean dan menyaring baris berdasarkan beberapa kondisi. Konektor Athena Hortonworks Hive dapat menggabungkan ekspresi ini dan mendorongnya langsung ke Hortonworks Hive untuk meningkatkan fungsionalitas dan untuk mengurangi jumlah data yang dipindai.

Operator konektor Athena Hortonworks Hive berikut mendukung pushdown predikat:

- Boolean: DAN, ATAU, TIDAK
- KESETARAAN: SAMA, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_NULL
- Aritmatika: TAMBAHKAN, KURANGI, KALIKAN, BAGI, MODULUS, MENIADAKAN
- Lainnya: LIKE_PATTERN, IN

Contoh pushdown gabungan

Untuk kemampuan kueri yang ditingkatkan, gabungkan jenis pushdown, seperti pada contoh berikut:

```
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;
```

Kueri passthrough

Konektor Hortonworks Hive mendukung kueri passthrough. Kueri passthrough menggunakan fungsi tabel untuk mendorong kueri lengkap Anda ke sumber data untuk dieksekusi.

Untuk menggunakan kueri passthrough dengan Hortonworks Hive, Anda dapat menggunakan sintaks berikut:

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

Contoh kueri berikut mendorong kueri ke sumber data di Hortonworks Hive. Kueri memilih semua kolom dalam `customer` tabel, membatasi hasilnya menjadi 10.

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

Informasi lisensi

Dengan menggunakan konektor ini, Anda mengakui penyertaan komponen pihak ketiga, daftar yang dapat ditemukan dalam file pom.xml untuk konektor ini, dan menyetujui persyaratan dalam masing-masing lisensi pihak ketiga yang disediakan dalam file LICENSE.txt di .com. GitHub

Sumber daya tambahan

Untuk informasi versi driver JDBC terbaru, lihat file pom.xml untuk konektor Hortonworks Hive di.com. GitHub

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Amazon Athena Apache Kafka

Konektor Amazon Athena untuk Apache Kafka memungkinkan Amazon Athena menjalankan kueri SQL pada topik Apache Kafka Anda. Gunakan konektor ini untuk melihat topik Apache Kafka sebagai tabel dan pesan sebagai baris di Athena.

Prasyarat

Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Batasan

- Menulis operasi DDL tidak didukung.
- Batas Lambda yang relevan. Untuk informasi selengkapnya, lihat Kuota Lambda dalam Panduan Pengembang AWS Lambda .
- Tipe data tanggal dan stempel waktu dalam kondisi filter harus dilemparkan ke tipe data yang sesuai.
- Tipe data tanggal dan stempel waktu tidak didukung untuk jenis file CSV dan diperlakukan sebagai nilai varchar.
- Pemetaan ke bidang JSON bersarang tidak didukung. Konektor hanya memetakan bidang tingkat atas.
- Konektor tidak mendukung tipe yang kompleks. Tipe kompleks ditafsirkan sebagai string.
- Untuk mengekstrak atau bekerja dengan nilai JSON yang kompleks, gunakan fungsi terkait JSON yang tersedia di Athena. Untuk informasi selengkapnya, lihat Mengekstrak data JSON dari string.

- Konektor tidak mendukung akses ke metadata pesan Kafka.

Ketentuan

- Metadata handler — Penangan Lambda yang mengambil metadata dari instance database Anda.
- Record handler - Handler Lambda yang mengambil catatan data dari instance database Anda.
- Composite handler — Handler Lambda yang mengambil data metadata dan data dari instance database Anda.
- Endpoint Kafka — String teks yang membuat koneksi ke instance Kafka.

Kompatibilitas cluster

Konektor Kafka dapat digunakan dengan jenis cluster berikut.

- Standalone Kafka — Koneksi langsung ke Kafka (diautentikasi atau tidak diautentikasi).
- Confluent — Koneksi langsung ke Confluent Kafka. Untuk informasi tentang penggunaan Athena dengan data Confluent Kafka, lihat Memvisualisasikan data Confluent di Amazon menggunakan QuickSight Amazon Athena di Blog Business Intelligence.AWS

Menghubungkan ke Confluent

Menghubungkan ke Confluent membutuhkan langkah-langkah berikut:

1. Buat kunci API dari Confluent.
2. Simpan nama pengguna dan kata sandi untuk kunci Confluent API ke dalam. AWS Secrets Manager
3. Berikan nama rahasia untuk variabel `secrets_manager_secret` lingkungan di konektor Kafka.
4. Ikuti langkah-langkah di Menyiapkan konektor Kafka bagian dokumen ini.

Metode otentikasi yang didukung

Konektor mendukung metode otentikasi berikut.

- SSL
- SELEMPANG

- SASL/PLAIN
- SELEMPANG/TEKS BIASA
- NO_AUTH
- Platform Kafka dan Confluent yang dikelola sendiri - SSL, SASL/SCRAM, SASL/PLAINTEXT, NO_AUTH
- Kafka dan Confluent Cloud yang dikelola sendiri - SASL/PLAIN

Untuk informasi selengkapnya, lihat Mengkonfigurasi otentikasi untuk konektor Athena Kafka.

Format data masukan yang didukung

Konektor mendukung format data input berikut.

- JSON
- CSV

Parameter

Gunakan variabel lingkungan Lambda yang disebutkan di bagian ini untuk mengonfigurasi konektor Athena Kafka.

- auth_type - Menentukan jenis otentikasi cluster. Konektor mendukung jenis otentikasi berikut:
  - NO_AUTH — Connect langsung ke Kafka (misalnya, ke cluster Kafka yang digunakan melalui instance EC2 yang tidak menggunakan otentikasi).
  - SASL_SSL_PLAIN — Metode ini menggunakan protokol `SASL_SSL` keamanan dan mekanisme SASL. `PLAIN` Untuk informasi selengkapnya, lihat konfigurasi SASL dalam dokumentasi Apache Kafka.
  - SASL_PLAINTEXT_PLAIN — Metode ini menggunakan protokol keamanan dan mekanisme SASL. `SASL_PLAINTEXT` `PLAIN` Untuk informasi selengkapnya, lihat konfigurasi SASL dalam dokumentasi Apache Kafka.
  - SASL_SSL_SCRAM_SHA512 - Anda dapat menggunakan jenis otentikasi ini untuk mengontrol akses ke cluster Apache Kafka Anda. Metode ini menyimpan nama pengguna dan kata sandi di AWS Secrets Manager. Rahasianya harus dikaitkan dengan cluster Kafka. Untuk informasi selengkapnya, lihat Otentikasi menggunakan SASL/SCRAM di dokumentasi Apache Kafka.
  - SASL_PLAINTEXT_SCRAM_SHA512 — Metode ini menggunakan protokol keamanan dan mekanisme. `SASL_PLAINTEXT` `SCRAM_SHA512` `SASL` Metode ini menggunakan nama

pengguna dan kata sandi yang disimpan di AWS Secrets Manager. Untuk informasi lebih lanjut, lihat bagian konfigurasi SASL dari dokumentasi Apache Kafka.

- SSL — Otentikasi SSL menggunakan penyimpanan kunci dan file toko kepercayaan untuk terhubung dengan cluster Apache Kafka. Anda harus membuat file trust store dan key store, mengunggahnya ke bucket Amazon S3, dan memberikan referensi ke Amazon S3 saat Anda menggunakan konektor. Toko kunci, toko kepercayaan, dan kunci SSL disimpan di AWS Secrets Manager. Klien Anda harus memberikan kunci AWS rahasia saat konektor dikerahkan. Untuk informasi selengkapnya, lihat Enkripsi dan Otentikasi menggunakan SSL di dokumentasi Apache Kafka.

Untuk informasi selengkapnya, lihat Mengkonfigurasi otentikasi untuk konektor Athena Kafka.

- certificates_s3_reference — Lokasi Amazon S3 yang berisi sertifikat (penyimpanan kunci dan file penyimpanan kepercayaan).
- disable_spill_encryption — (Opsional) Ketika diatur ke, menonaktifkan enkripsi tumpahan. `True` Defaultnya `False` sehingga data yang tumpah ke S3 dienkripsi menggunakan AES-GCM — baik menggunakan kunci yang dihasilkan secara acak atau KMS untuk menghasilkan kunci. Menonaktifkan enkripsi tumpahan dapat meningkatkan kinerja, terutama jika lokasi tumpahan Anda menggunakan enkripsi sisi server.
- kafka_endpoint — Detail titik akhir untuk diberikan kepada Kafka.
- secrets_manager_secret — Nama rahasia tempat AWS kredensialnya disimpan.
- Parameter tumpahan — Fungsi Lambda menyimpan sementara (“tumpahan”) data yang tidak sesuai dengan memori ke Amazon S3. Semua instance database yang diakses oleh fungsi Lambda yang sama tumpah ke lokasi yang sama. Gunakan parameter dalam tabel berikut untuk menentukan lokasi tumpahan.

Parameter	Deskripsi
`spill_bucket`	Wajib. Nama bucket Amazon S3 tempat fungsi Lambda dapat menumpahkan data.
`spill_prefix`	Wajib. Awalan dalam ember tumpahan tempat fungsi Lambda dapat menumpahkan data.
`spill_put_request_headers`	(Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject {"x-amz-`

Parameter	Deskripsi
	`server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.

- ID Subnet — Satu atau lebih ID subnet yang sesuai dengan subnet yang dapat digunakan fungsi Lambda untuk mengakses sumber data Anda.
  - Cluster Kafka Publik atau cluster Confluent Cloud standar — Kaitkan konektor dengan subnet pribadi yang memiliki NAT Gateway.
  - Cluster Confluent Cloud dengan konektivitas pribadi — Kaitkan konektor dengan subnet pribadi yang memiliki rute ke cluster Confluent Cloud.
    - Untuk AWS Transit Gateway, subnet harus berada dalam VPC yang dilampirkan ke gateway transit yang sama yang digunakan Confluent Cloud.
    - Untuk VPC Peering, subnet harus dalam VPC yang diintip ke Confluent Cloud VPC.
    - Untuk AWS PrivateLink, subnet harus dalam VPC yang memiliki rute ke titik akhir VPC yang terhubung ke Confluent Cloud.

> **Note**
>
> Jika Anda menyebarkan konektor ke VPC untuk mengakses sumber daya pribadi dan juga ingin terhubung ke layanan yang dapat diakses publik seperti Confluent, Anda harus mengaitkan konektor dengan subnet pribadi yang memiliki NAT Gateway. Untuk informasi selengkapnya, lihat gateway NAT di Panduan Pengguna Amazon VPC.

Dukungan tipe data

Tabel berikut menunjukkan tipe data yang sesuai yang didukung untuk Kafka dan Apache Arrow.

Kafka	Panah
CHAR	VARCHAR
VARCHAR	VARCHAR
TIMESTAMP	MILIDETIK

Kafka	Panah
TANGGAL	DAY
BOOLEAN	BOOL
SMALLINT	SMALLINT
INTEGER	INT
BIGINT	BIGINT
DECIMAL	FLOAT8
DOUBLE	FLOAT8

Partisi dan split

Topik Kafka dibagi menjadi beberapa partisi. Setiap partisi dipesan. Setiap pesan dalam partisi memiliki ID tambahan yang disebut offset. Setiap partisi Kafka dibagi lagi menjadi beberapa split untuk pemrosesan paralel. Data tersedia untuk periode retensi yang dikonfigurasi dalam cluster Kafka.

Praktik terbaik

Sebagai praktik terbaik, gunakan predikat pushdown saat Anda menanyakan Athena, seperti pada contoh berikut.

```
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'
```

Menyiapkan konektor Kafka

Sebelum Anda dapat menggunakan konektor, Anda harus mengatur cluster Apache Kafka Anda, menggunakan AWS Glue Schema Registry untuk menentukan skema Anda, dan mengkonfigurasi otentikasi untuk konektor.

Saat bekerja dengan AWS Glue Schema Registry, perhatikan poin-poin berikut:

- Pastikan bahwa teks di bidang Deskripsi dari AWS Glue Schema Registry menyertakan string`{AthenaFederationKafka}`. String penanda ini diperlukan untuk AWS Glue Registries yang Anda gunakan dengan konektor Amazon Athena Kafka.
- Untuk kinerja terbaik, gunakan hanya huruf kecil untuk nama database dan nama tabel Anda. Menggunakan casing campuran menyebabkan konektor melakukan pencarian case insensitive yang lebih intensif secara komputasi.

Untuk mengatur lingkungan Apache Kafka dan AWS Glue Registri Skema

1. Siapkan lingkungan Apache Kafka Anda.
2. Unggah file deskripsi topik Kafka (yaitu skema) dalam format JSON ke Schema Registry. AWS Glue Untuk informasi selengkapnya, lihat Mengintegrasikan dengan Registri AWS Glue Skema di Panduan AWS Glue Pengembang. Misalnya skema, lihat bagian berikut.

Contoh skema untuk Registri AWS Glue Skema

Gunakan format contoh di bagian ini saat Anda mengunggah skema Anda ke Registri AWS Glue Skema.

Contoh skema tipe JSON

Dalam contoh berikut, skema yang akan dibuat dalam AWS Glue Schema Registry menentukan `json` sebagai nilai untuk `dataFormat` dan menggunakan untuk. `datatypejson` `topicName`

> **Note**
>
> Nilai untuk `topicName` harus menggunakan casing yang sama dengan nama topik di Kafka.

```
{
  "topicName": "datatypejson",
```

```
  "message": {
    "dataFormat": "json",
    "fields": [
      {
        "name": "intcol",
        "mapping": "intcol",
        "type": "INTEGER"
      },
      {
        "name": "varcharcol",
        "mapping": "varcharcol",
        "type": "VARCHAR"
      },
      {
        "name": "booleancol",
        "mapping": "booleancol",
        "type": "BOOLEAN"
      },
      {
        "name": "bigintcol",
        "mapping": "bigintcol",
        "type": "BIGINT"
      },
      {
        "name": "doublecol",
        "mapping": "doublecol",
        "type": "DOUBLE"
      },
      {
        "name": "smallintcol",
        "mapping": "smallintcol",
        "type": "SMALLINT"
      },
      {
        "name": "tinyintcol",
        "mapping": "tinyintcol",
        "type": "TINYINT"
      },
      {
        "name": "datecol",
        "mapping": "datecol",
        "type": "DATE",
        "formatHint": "yyyy-MM-dd"
      },
```

```
      {
        "name": "timestampcol",
        "mapping": "timestampcol",
        "type": "TIMESTAMP",
        "formatHint": "yyyy-MM-dd HH:mm:ss.SSS"
      }
    ]
  }
}
```

Contoh skema tipe CSV

Dalam contoh berikut, skema yang akan dibuat dalam AWS Glue Schema Registry menentukan `csv` sebagai nilai untuk `dataFormat` dan menggunakan untuk. `datatypecsvbulk` `topicName` Nilai untuk `topicName` harus menggunakan casing yang sama dengan nama topik di Kafka.

```
{
  "topicName": "datatypecsvbulk",
  "message": {
    "dataFormat": "csv",
    "fields": [
      {
        "name": "intcol",
        "type": "INTEGER",
        "mapping": "0"
      },
      {
        "name": "varcharcol",
        "type": "VARCHAR",
        "mapping": "1"
      },
      {
        "name": "booleancol",
        "type": "BOOLEAN",
        "mapping": "2"
      },
      {
        "name": "bigintcol",
        "type": "BIGINT",
        "mapping": "3"
      },
      {
        "name": "doublecol",
```

```
        "type": "DOUBLE",
        "mapping": "4"
      },
      {
        "name": "smallintcol",
        "type": "SMALLINT",
        "mapping": "5"
      },
      {
        "name": "tinyintcol",
        "type": "TINYINT",
        "mapping": "6"
      },
      {
        "name": "floatcol",
        "type": "DOUBLE",
        "mapping": "7"
      }
    ]
  }
}
```

Mengkonfigurasi otentikasi untuk konektor Athena Kafka

Anda dapat menggunakan berbagai metode untuk mengautentikasi ke cluster Apache Kafka Anda, termasuk SSL, SASL/SCRAM, SASL/PLAIN, dan SASL/PLAINTEXT.

Tabel berikut menunjukkan jenis otentikasi untuk konektor dan protokol keamanan dan mekanisme SASL untuk masing-masing. Untuk informasi lebih lanjut, lihat bagian Keamanan dari dokumentasi Apache Kafka.

auth_type	security.protocol	sasl.mekanisme	Kompatibilitas tipe cluster
`SASL_SSL_PLAIN`	`SASL_SSL`	`PLAIN`	• Kafka yang dikelola sendiri • Platform Konfluen • Awan Konfluen
`SASL_PLAINTEXT_PLAIN`	`SASL_PLAINTEXT`	`PLAIN`	• Kafka yang dikelola sendiri

auth_type	security.protocol	sasl.mekanisme	Kompatibilitas tipe cluster
			• Platform Konfluen
`SASL_SSL_SCRAM_SHA512`	`SASL_SSL`	`SCRAM-SHA-512`	• Kafka yang dikelola sendiri • Platform Konfluen
`SASL_PLAINTEXT_SCRAM_SHA512`	`SASL_PLAINTEXT`	`SCRAM-SHA-512`	• Kafka yang dikelola sendiri • Platform Konfluen
`SSL`	`SSL`	N/A	• Kafka yang dikelola sendiri • Platform Konfluen

SSL

Jika klaster diautentikasi SSL, Anda harus membuat file trust store dan key store dan mengunggahnya ke bucket Amazon S3. Anda harus memberikan referensi Amazon S3 ini saat Anda menggunakan konektor. Toko kunci, toko kepercayaan, dan kunci SSL disimpan di file. AWS Secrets Manager Anda memberikan kunci AWS rahasia saat Anda menggunakan konektor.

Untuk informasi tentang cara membuat rahasia di Secrets Manager, lihat Membuat AWS Secrets Manager rahasia.

Untuk menggunakan jenis otentikasi ini, atur variabel lingkungan seperti yang ditunjukkan pada tabel berikut.

Parameter	Nilai
`auth_type`	`SSL`
`certificates_s3_reference`	Lokasi Amazon S3 yang berisi sertifikat.

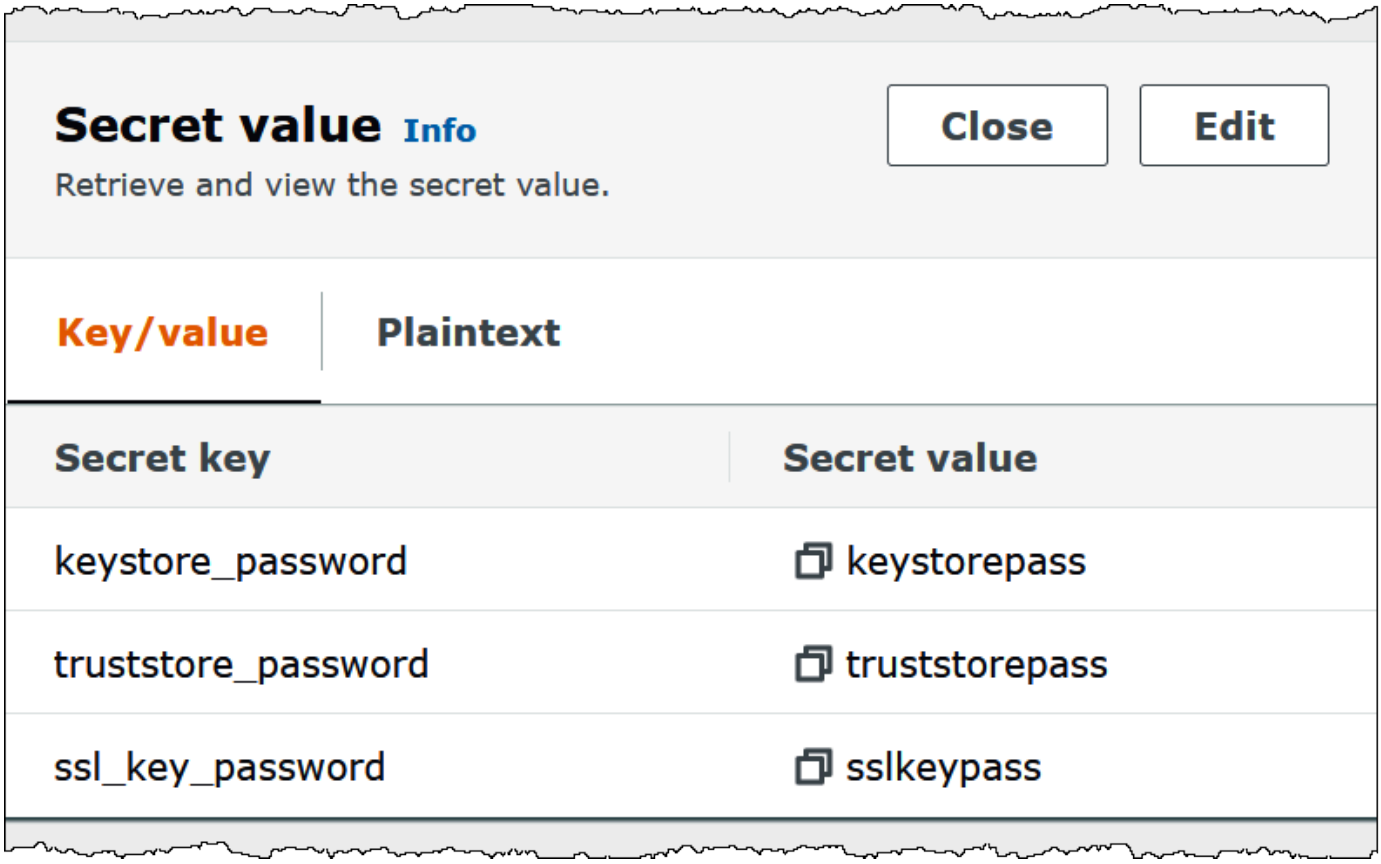
Parameter	Nilai
`secrets_manager_secret`	Nama kunci AWS rahasiamu.

Setelah Anda membuat rahasia di Secrets Manager, Anda dapat melihatnya di konsol Secrets Manager.

Untuk melihat rahasia Anda di Secrets Manager

1. Buka konsol Secrets Manager di https://console.aws.amazon.com/secretsmanager/.
2. Di panel navigasi, pilih Rahasia.
3. Pada halaman Rahasia, pilih tautan ke rahasia Anda.
4. Pada halaman detail untuk rahasia Anda, pilih Ambil nilai rahasia.

   Gambar berikut menunjukkan contoh rahasia dengan tiga pasangan kunci/nilai:`keystore_password`,`truststore_password`, dan. `ssl_key_password`



Untuk informasi selengkapnya tentang penggunaan SSL dengan Kafka, lihat Enkripsi dan Otentikasi menggunakan SSL di dokumentasi Apache Kafka.

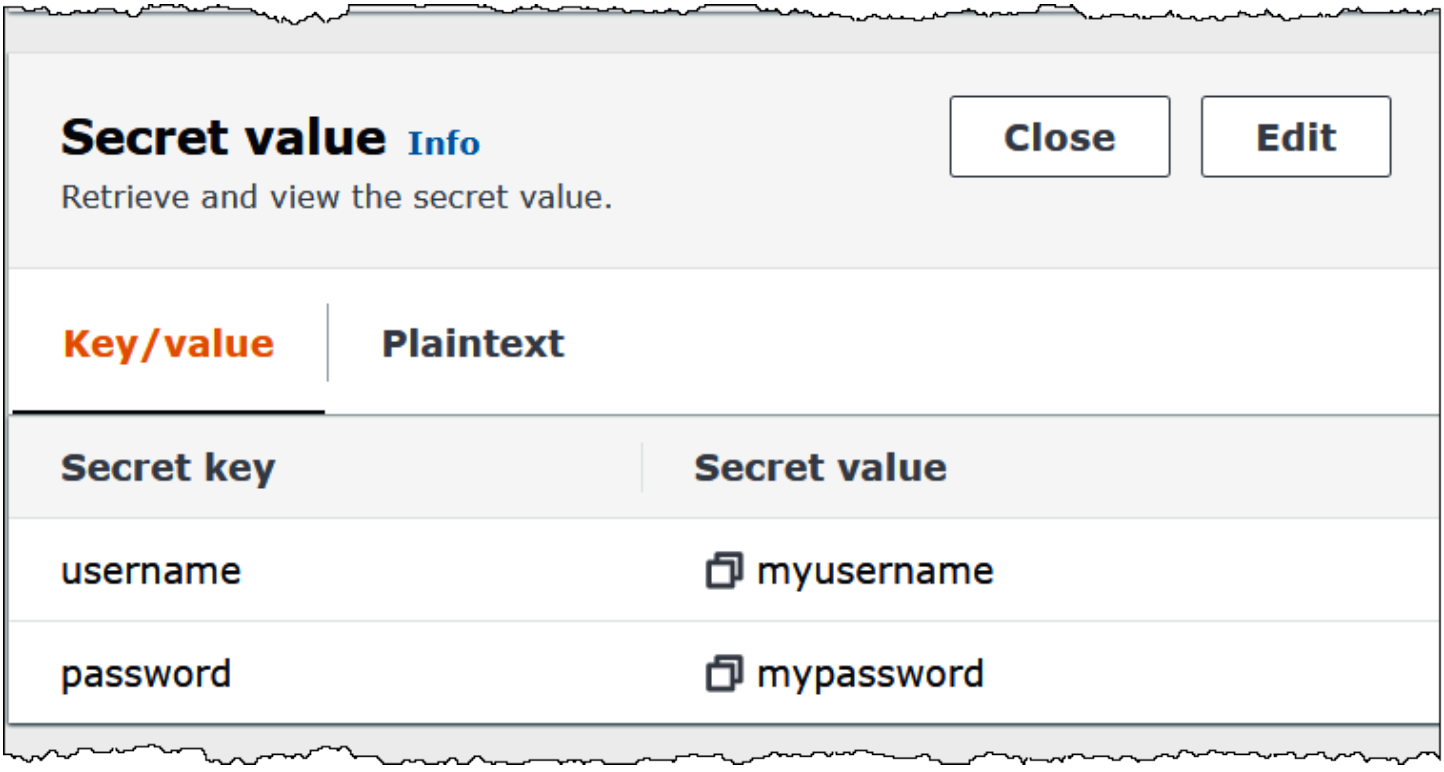
SASL/SCRAM

Jika klaster Anda menggunakan otentikasi SCRAM, berikan kunci Secrets Manager yang terkait dengan cluster saat Anda menerapkan konektor. AWS Kredensi pengguna (kunci rahasia dan kunci akses) digunakan untuk mengautentikasi dengan cluster.

Mengatur variabel lingkungan seperti yang ditunjukkan pada tabel berikut.

Parameter	Nilai
`auth_type`	`SASL_SSL_SCRAM_SHA512`
`secrets_manager_secret`	Nama kunci AWS rahasiamu.

Gambar berikut menunjukkan contoh rahasia di konsol Secrets Manager dengan dua pasangan kunci/nilai: satu untuk`username`, dan satu untuk. `password`



Untuk informasi selengkapnya tentang penggunaan SASL/SCRAM dengan Kafka, lihat Otentikasi menggunakan SASL/SCRAM di dokumentasi Apache Kafka.

Informasi lisensi

Dengan menggunakan konektor ini, Anda mengakui penyertaan komponen pihak ketiga, daftar yang dapat ditemukan dalam file pom.xml untuk konektor ini, dan menyetujui persyaratan dalam masing-masing lisensi pihak ketiga yang disediakan dalam file LICENSE.txt di .com. GitHub

Sumber daya tambahan

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor MSK Amazon Athena

Konektor Amazon Athena untuk Amazon MSK memungkinkan Amazon Athena menjalankan kueri SQL pada topik Apache Kafka Anda. Gunakan konektor ini untuk melihat topik Apache Kafka sebagai tabel dan pesan sebagai baris di Athena. Untuk informasi tambahan, lihat Menganalisis data streaming real-time di Amazon MSK dengan Amazon Athena di Big Data AWS Blog.

Prasyarat

Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Batasan

- Menulis operasi DDL tidak didukung.
- Batas Lambda yang relevan. Untuk informasi selengkapnya, lihat Kuota Lambda dalam Panduan Pengembang AWS Lambda .
- Tipe data tanggal dan stempel waktu dalam kondisi filter harus dilemparkan ke tipe data yang sesuai.
- Tipe data tanggal dan stempel waktu tidak didukung untuk jenis file CSV dan diperlakukan sebagai nilai varchar.
- Pemetaan ke bidang JSON bersarang tidak didukung. Konektor hanya memetakan bidang tingkat atas.
- Konektor tidak mendukung tipe yang kompleks. Tipe kompleks ditafsirkan sebagai string.
- Untuk mengekstrak atau bekerja dengan nilai JSON yang kompleks, gunakan fungsi terkait JSON yang tersedia di Athena. Untuk informasi selengkapnya, lihat Mengekstrak data JSON dari string.
- Konektor tidak mendukung akses ke metadata pesan Kafka.

Ketentuan

- Metadata handler — Penangan Lambda yang mengambil metadata dari instance database Anda.
- Record handler - Handler Lambda yang mengambil catatan data dari instance database Anda.
- Composite handler — Handler Lambda yang mengambil data metadata dan data dari instance database Anda.
- Endpoint Kafka — String teks yang membuat koneksi ke instance Kafka.

Kompatibilitas cluster

Konektor MSK dapat digunakan dengan jenis cluster berikut.

- MSK Provisioned cluster — Anda secara manual menentukan, memantau, dan menskalakan kapasitas cluster.
- MSK Serverless cluster — Menyediakan kapasitas sesuai permintaan yang diskalakan secara otomatis sebagai timbangan I/O aplikasi.
- Standalone Kafka — Koneksi langsung ke Kafka (diautentikasi atau tidak diautentikasi).

Metode otentikasi yang didukung

Konektor mendukung metode otentikasi berikut.

- SALL/IAM
- SSL
- SELEMPANG
- SASL/PLAIN
- SELEMPANG/TEKS BIASA
- NO_AUTH

  Untuk informasi selengkapnya, lihat Mengkonfigurasi otentikasi untuk konektor MSK Athena.

Format data input yang didukung

Konektor mendukung format data input berikut.

- JSON

- CSV

Parameter

Gunakan variabel lingkungan Lambda yang disebutkan di bagian ini untuk mengonfigurasi konektor MSK Athena.

- auth_type - Menentukan jenis otentikasi cluster. Konektor mendukung jenis otentikasi berikut:
  - NO_AUTH — Terhubung langsung ke Kafka tanpa otentikasi (misalnya, ke cluster Kafka yang digunakan melalui instans EC2 yang tidak menggunakan otentikasi).
  - SASL_SSL_PLAIN — Metode ini menggunakan protokol `SASL_SSL` keamanan dan mekanisme SASL. `PLAIN`
  - SASL_PLAINTEXT_PLAIN — Metode ini menggunakan protokol keamanan dan mekanisme SASL. `SASL_PLAINTEXT` `PLAIN`

    > **Note**
    >
    > Jenis `SASL_SSL_PLAIN` dan `SASL_PLAINTEXT_PLAIN` otentikasi didukung oleh Apache Kafka tetapi tidak oleh Amazon MSK.

  - SASL_SSL_AWS_MSK_IAM - Kontrol akses IAM untuk Amazon MSK memungkinkan Anda menangani otentikasi dan otorisasi untuk klaster MSK Anda. AWS Kredensyal pengguna Anda (kunci rahasia dan kunci akses) digunakan untuk terhubung dengan cluster. Untuk informasi selengkapnya, lihat Kontrol akses IAM di Panduan Pengembang Amazon Managed Streaming for Apache Kafka Kafka.
  - SASL_SSL_SCRAM_SHA512 - Anda dapat menggunakan jenis otentikasi ini untuk mengontrol akses ke kluster MSK Amazon Anda. Metode ini menyimpan nama pengguna dan kata sandi AWS Secrets Manager. Rahasianya harus dikaitkan dengan cluster MSK Amazon. Untuk informasi selengkapnya, lihat Menyiapkan autentikasi SASL/SCRAM untuk klaster MSK Amazon di Panduan Pengembang Amazon Managed Streaming for Apache Kafka.
  - SSL — Otentikasi SSL menggunakan penyimpanan kunci dan file penyimpanan kepercayaan untuk terhubung dengan kluster MSK Amazon. Anda harus membuat file trust store dan key store, mengunggahnya ke bucket Amazon S3, dan memberikan referensi ke Amazon S3 saat Anda menggunakan konektor. Toko kunci, toko kepercayaan, dan kunci SSL disimpan di AWS Secrets Manager. Klien Anda harus memberikan kunci AWS rahasia saat konektor dikerahkan.

Untuk informasi selengkapnya, lihat Autentikasi TLS Mutual di Amazon Managed Streaming for Apache Kafka Developer Guide.

Untuk informasi selengkapnya, lihat Mengkonfigurasi otentikasi untuk konektor MSK Athena.

- certificates_s3_reference — Lokasi Amazon S3 yang berisi sertifikat (penyimpanan kunci dan file penyimpanan kepercayaan).
- disable_spill_encryption — (Opsional) Ketika diatur ke, menonaktifkan enkripsi tumpahan. `True` Defaultnya `False` sehingga data yang tumpah ke S3 dienkripsi menggunakan AES-GCM — baik menggunakan kunci yang dihasilkan secara acak atau KMS untuk menghasilkan kunci. Menonaktifkan enkripsi tumpahan dapat meningkatkan kinerja, terutama jika lokasi tumpahan Anda menggunakan enkripsi sisi server.
- kafka_endpoint — Detail titik akhir untuk diberikan kepada Kafka. Misalnya, untuk klaster MSK Amazon, Anda menyediakan URL bootstrap untuk klaster.
- secrets_manager_secret — Nama rahasia tempat AWS kredensialnya disimpan. Parameter ini tidak diperlukan untuk otentikasi IAM.
- Parameter tumpahan — Fungsi Lambda menyimpan sementara (“tumpahan”) data yang tidak sesuai dengan memori ke Amazon S3. Semua instance database yang diakses oleh fungsi Lambda yang sama tumpah ke lokasi yang sama. Gunakan parameter dalam tabel berikut untuk menentukan lokasi tumpahan.

Parameter	Deskripsi
`spill_bucket`	Wajib. Nama bucket Amazon S3 tempat fungsi Lambda dapat menumpahkan data.
`spill_prefix`	Wajib. Awalan dalam ember tumpahan tempat fungsi Lambda dapat menumpahkan data.
`spill_put_request_headers`	(Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject {"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.

Dukungan tipe data

Tabel berikut menunjukkan tipe data yang sesuai yang didukung untuk Kafka dan Apache Arrow.

Kafka	Panah
CHAR	VARCHAR
VARCHAR	VARCHAR
TIMESTAMP	MILIDETIK
TANGGAL	DAY
BOOLEAN	BOOL
SMALLINT	SMALLINT
INTEGER	INT
BIGINT	BIGINT
DECIMAL	FLOAT8
DOUBLE	FLOAT8

Partisi dan split

Topik Kafka dibagi menjadi beberapa partisi. Setiap partisi dipesan. Setiap pesan dalam partisi memiliki ID tambahan yang disebut offset. Setiap partisi Kafka dibagi lagi menjadi beberapa split untuk pemrosesan paralel. Data tersedia untuk periode retensi yang dikonfigurasi dalam cluster Kafka.

Praktik terbaik

Sebagai praktik terbaik, gunakan predikat pushdown saat Anda menanyakan Athena, seperti pada contoh berikut.

```
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'
```

Menyiapkan konektor MSK

Sebelum Anda dapat menggunakan konektor, Anda harus mengatur cluster MSK Amazon Anda, menggunakan AWS Glue Schema Registry untuk menentukan skema Anda, dan mengkonfigurasi otentikasi untuk konektor.

> **Note**
>
> Jika Anda menyebarkan konektor ke VPC untuk mengakses sumber daya pribadi dan juga ingin terhubung ke layanan yang dapat diakses publik seperti Confluent, Anda harus mengaitkan konektor dengan subnet pribadi yang memiliki NAT Gateway. Untuk informasi selengkapnya, lihat gateway NAT di Panduan Pengguna Amazon VPC.

Saat bekerja dengan AWS Glue Schema Registry, perhatikan poin-poin berikut:

- Pastikan bahwa teks di bidang Deskripsi dari AWS Glue Schema Registry menyertakan string`{AthenaFederationMSK}`. String penanda ini diperlukan untuk AWS Glue Registries yang Anda gunakan dengan konektor MSK Amazon Athena.
- Untuk kinerja terbaik, gunakan hanya huruf kecil untuk nama database dan nama tabel Anda. Menggunakan casing campuran menyebabkan konektor melakukan pencarian case insensitive yang lebih intensif secara komputasi.

Untuk mengatur lingkungan MSK Amazon dan Registri AWS Glue Skema

1. Siapkan lingkungan MSK Amazon Anda. Untuk informasi dan langkah-langkahnya, lihat Menyiapkan MSK Amazon dan Memulai menggunakan MSK Amazon di Panduan Pengembang Amazon Managed Streaming for Apache Kafka.
2. Unggah file deskripsi topik Kafka (yaitu skema) dalam format JSON ke Schema Registry. AWS Glue Untuk informasi selengkapnya, lihat Mengintegrasikan dengan Registri AWS Glue Skema di Panduan AWS Glue Pengembang. Misalnya skema, lihat bagian berikut.

Contoh skema untuk Registri AWS Glue Skema

Gunakan format contoh di bagian ini saat Anda mengunggah skema Anda ke Registri AWS Glue Skema.

Contoh skema tipe JSON

Dalam contoh berikut, skema yang akan dibuat dalam AWS Glue Schema Registry menentukan `json` sebagai nilai untuk `dataFormat` dan menggunakan untuk. `datatypejson` `topicName`

> **Note**
>
> Nilai untuk `topicName` harus menggunakan casing yang sama dengan nama topik di Kafka.

```
{
  "topicName": "datatypejson",
  "message": {
    "dataFormat": "json",
    "fields": [
      {
        "name": "intcol",
        "mapping": "intcol",
        "type": "INTEGER"
      },
      {
        "name": "varcharcol",
        "mapping": "varcharcol",
        "type": "VARCHAR"
      },
      {
        "name": "booleancol",
        "mapping": "booleancol",
        "type": "BOOLEAN"
      },
      {
        "name": "bigintcol",
        "mapping": "bigintcol",
        "type": "BIGINT"
      },
      {
        "name": "doublecol",
        "mapping": "doublecol",
```

```
        "type": "DOUBLE"
      },
      {
        "name": "smallintcol",
        "mapping": "smallintcol",
        "type": "SMALLINT"
      },
      {
        "name": "tinyintcol",
        "mapping": "tinyintcol",
        "type": "TINYINT"
      },
      {
        "name": "datecol",
        "mapping": "datecol",
        "type": "DATE",
        "formatHint": "yyyy-MM-dd"
      },
      {
        "name": "timestampcol",
        "mapping": "timestampcol",
        "type": "TIMESTAMP",
        "formatHint": "yyyy-MM-dd HH:mm:ss.SSS"
      }
    ]
  }
}
```

Contoh skema tipe CSV

Dalam contoh berikut, skema yang akan dibuat dalam AWS Glue Schema Registry menentukan `csv` sebagai nilai untuk `dataFormat` dan menggunakan untuk. `datatypecsvbulk` `topicName` Nilai untuk `topicName` harus menggunakan casing yang sama dengan nama topik di Kafka.

```
{
  "topicName": "datatypecsvbulk",
  "message": {
    "dataFormat": "csv",
    "fields": [
      {
        "name": "intcol",
        "type": "INTEGER",
        "mapping": "0"
```

```
      },
      {
        "name": "varcharcol",
        "type": "VARCHAR",
        "mapping": "1"
      },
      {
        "name": "booleancol",
        "type": "BOOLEAN",
        "mapping": "2"
      },
      {
        "name": "bigintcol",
        "type": "BIGINT",
        "mapping": "3"
      },
      {
        "name": "doublecol",
        "type": "DOUBLE",
        "mapping": "4"
      },
      {
        "name": "smallintcol",
        "type": "SMALLINT",
        "mapping": "5"
      },
      {
        "name": "tinyintcol",
        "type": "TINYINT",
        "mapping": "6"
      },
      {
        "name": "floatcol",
        "type": "DOUBLE",
        "mapping": "7"
      }
    ]
  }
}
```

Mengkonfigurasi otentikasi untuk konektor MSK Athena

Anda dapat menggunakan berbagai metode untuk mengautentikasi ke cluster MSK Amazon Anda, termasuk IAM, SSL, SCRAM, dan Kafka mandiri.

Tabel berikut menunjukkan jenis otentikasi untuk konektor dan protokol keamanan dan mekanisme SASL untuk masing-masing. Untuk informasi selengkapnya, lihat Otentikasi dan otorisasi untuk Apache Kafka API di Panduan Pengembang Amazon Managed Streaming for Apache Kafka.

auth_type	security.protocol	sasl.mekanisme
`SASL_SSL_PLAIN`	`SASL_SSL`	`PLAIN`
`SASL_PLAINTEXT_PLAIN`	`SASL_PLAINTEXT`	`PLAIN`
`SASL_SSL_AWS_MSK_IAM`	`SASL_SSL`	`AWS_MSK_IAM`
`SASL_SSL_SCRAM_SHA512`	`SASL_SSL`	`SCRAM-SHA-512`
`SSL`	`SSL`	N/A

> **Note**
>
> Jenis `SASL_SSL_PLAIN` dan `SASL_PLAINTEXT_PLAIN` otentikasi didukung oleh Apache Kafka tetapi tidak oleh Amazon MSK.

SELEMPANG/IAM

Jika klaster menggunakan autentikasi IAM, Anda harus mengonfigurasi kebijakan IAM untuk pengguna saat menyiapkan klaster. Untuk informasi selengkapnya, lihat Kontrol akses IAM di Panduan Pengembang Amazon Managed Streaming for Apache Kafka Kafka.

Untuk menggunakan jenis otentikasi ini, setel variabel lingkungan `auth_type` Lambda untuk konektor. `SASL_SSL_AWS_MSK_IAM`

SSL

Jika klaster diautentikasi SSL, Anda harus membuat file trust store dan key store dan mengunggahnya ke bucket Amazon S3. Anda harus memberikan referensi Amazon S3 ini saat Anda

menggunakan konektor. Toko kunci, toko kepercayaan, dan kunci SSL disimpan di file. AWS Secrets Manager Anda memberikan kunci AWS rahasia saat Anda menggunakan konektor.

Untuk informasi tentang cara membuat rahasia di Secrets Manager, lihat Membuat AWS Secrets Manager rahasia.

Untuk menggunakan jenis otentikasi ini, atur variabel lingkungan seperti yang ditunjukkan pada tabel berikut.

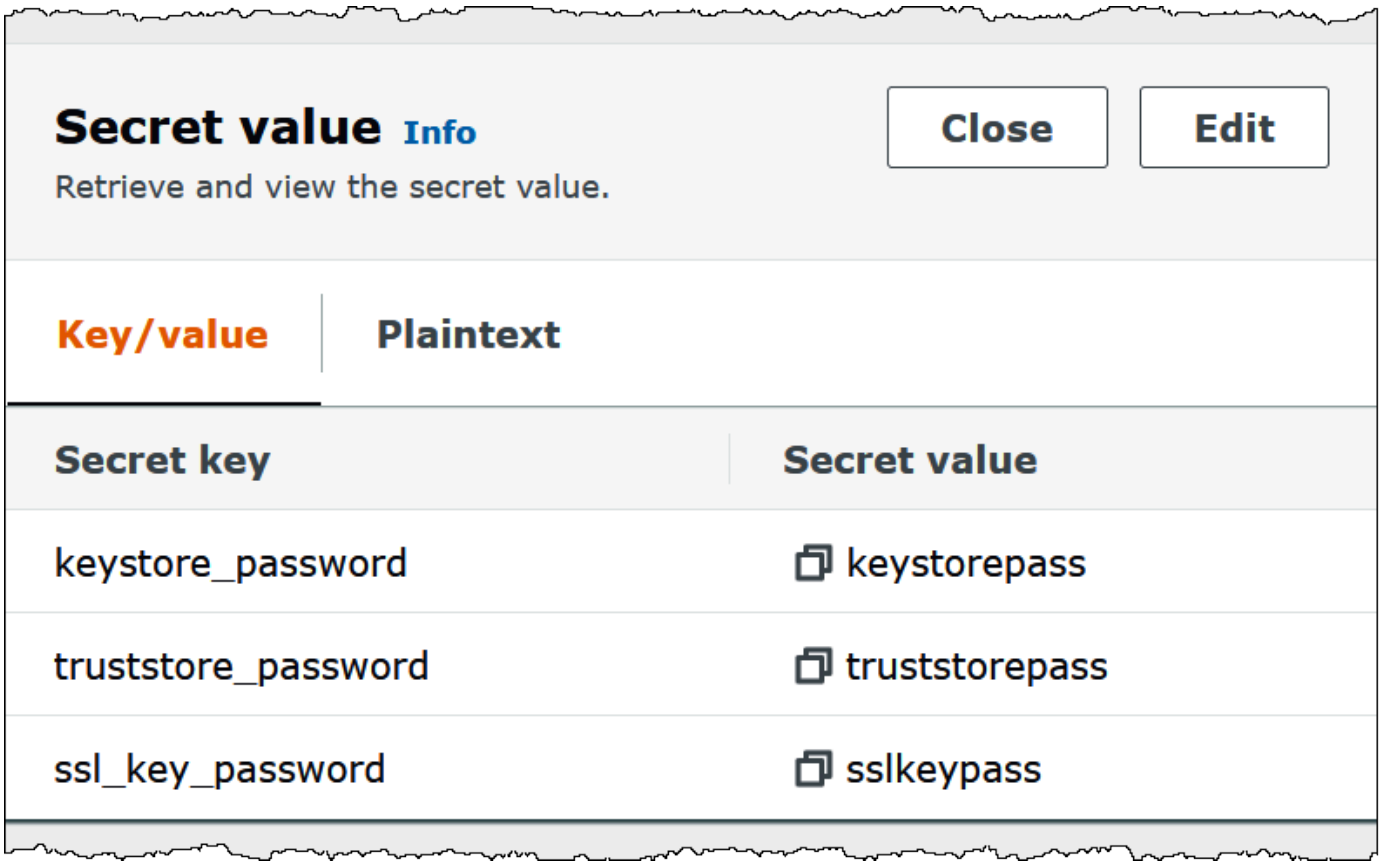
Parameter	Nilai
`auth_type`	`SSL`
`certificates_s3_reference`	Lokasi Amazon S3 yang berisi sertifikat.
`secrets_manager_secret`	Nama kunci AWS rahasiamu.

Setelah Anda membuat rahasia di Secrets Manager, Anda dapat melihatnya di konsol Secrets Manager.

Untuk melihat rahasia Anda di Secrets Manager

1. Buka konsol Secrets Manager di https://console.aws.amazon.com/secretsmanager/.
2. Di panel navigasi, pilih Rahasia.
3. Pada halaman Rahasia, pilih tautan ke rahasia Anda.
4. Pada halaman detail untuk rahasia Anda, pilih Ambil nilai rahasia.

   Gambar berikut menunjukkan contoh rahasia dengan tiga pasangan kunci/nilai:`keystore_password`,`truststore_password`, dan. `ssl_key_password`



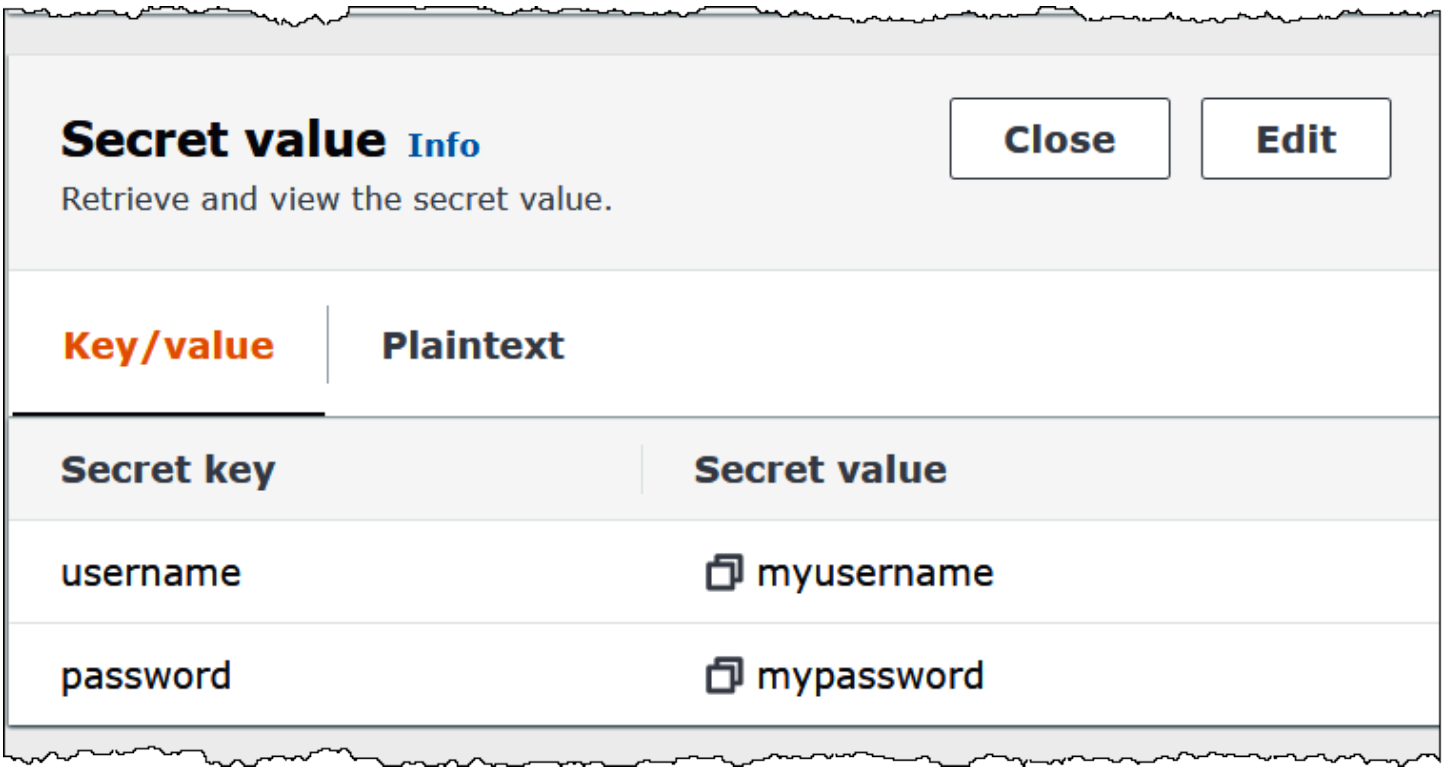
SASL/SCRAM

Jika klaster Anda menggunakan otentikasi SCRAM, berikan kunci Secrets Manager yang terkait dengan cluster saat Anda menerapkan konektor. AWS Kredensyal pengguna (kunci rahasia dan kunci akses) digunakan untuk mengautentikasi dengan cluster.

Mengatur variabel lingkungan seperti yang ditunjukkan pada tabel berikut.

Parameter	Nilai
`auth_type`	`SASL_SSL_SCRAM_SHA512`
`secrets_manager_secret`	Nama kunci AWS rahasiamu.

Gambar berikut menunjukkan contoh rahasia di konsol Secrets Manager dengan dua pasangan kunci/nilai: satu untuk`username`, dan satu untuk. `password`



Informasi lisensi

Dengan menggunakan konektor ini, Anda mengakui penyertaan komponen pihak ketiga, daftar yang dapat ditemukan dalam file pom.xml untuk konektor ini, dan menyetujui persyaratan dalam masing-masing lisensi pihak ketiga yang disediakan dalam file LICENSE.txt di .com. GitHub

Sumber daya tambahan

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor MySQL Amazon Athena

Konektor MySQL Amazon Athena Lambda memungkinkan Amazon Athena mengakses database MySQL.

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.
- Siapkan VPC dan grup keamanan sebelum Anda menggunakan konektor ini. Untuk informasi selengkapnya, lihat Membuat VPC untuk konektor sumber data.

Batasan

- Menulis operasi DDL tidak didukung.
- Dalam pengaturan multiplexer, bucket tumpahan dan awalan dibagikan di semua instance database.
- Batas Lambda yang relevan. Untuk informasi selengkapnya, lihat Kuota Lambda dalam Panduan Pengembang AWS Lambda .
- Karena Athena mengonversi kueri ke huruf kecil, nama tabel MySQL harus dalam huruf kecil. Misalnya, kueri Athena terhadap tabel bernama akan `myTable` gagal.

Ketentuan

Istilah berikut berhubungan dengan konektor MySQL.

- Instans database — Instance apa pun dari database yang digunakan di tempat, di Amazon EC2, atau di Amazon RDS.
- Handler - Handler Lambda yang mengakses instance database Anda. Handler bisa untuk metadata atau untuk catatan data.
- Metadata handler — Penangan Lambda yang mengambil metadata dari instance database Anda.
- Record handler - Handler Lambda yang mengambil catatan data dari instance database Anda.
- Composite handler — Handler Lambda yang mengambil data metadata dan data dari instance database Anda.
- Properti atau parameter - Properti database yang digunakan oleh penangan untuk mengekstrak informasi database. Anda mengonfigurasi properti ini sebagai variabel lingkungan Lambda.
- Connection String — Sebuah string teks yang digunakan untuk membuat koneksi ke instance database.
- Katalog —AWS Glue Non-katalog yang terdaftar di Athena yang merupakan awalan yang diperlukan untuk properti. `connection_string`
- Multiplexing handler - Handler Lambda yang dapat menerima dan menggunakan beberapa koneksi database.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengkonfigurasi konektor MySQL.

String koneksi

Gunakan string koneksi JDBC dalam format berikut untuk terhubung ke instance database.

```
mysql://${jdbc_connection_string}
```

> **Note**
>
> Jika Anda menerima kesalahan java.sql.SqlException: Nilai tanggal nol dilarang saat melakukan kueri SELECT pada tabel MySQL, tambahkan parameter berikut ke string koneksi Anda:
>
> ```
> zeroDateTimeBehavior=convertToNull
> ```
>
> Untuk informasi selengkapnya, lihat Kesalahan 'Nilai tanggal nol dilarang' saat mencoba memilih dari tabel MySQL di.com. GitHub

Menggunakan handler multiplexing

Anda dapat menggunakan multiplexer untuk terhubung ke beberapa instance database dengan satu fungsi Lambda. Permintaan dirutekan dengan nama katalog. Gunakan kelas berikut di Lambda.

Handler	Kelas
Pawang komposit	`MySqlMuxCompositeHandler`
Penangan metadata	`MySqlMuxMetadataHandler`
Rekam handler	`MySqlMuxRecordHandler`

Parameter handler multiplexing

Parameter	Deskripsi
`$catalog_connection_string`	Wajib. Sebuah string koneksi instance database. Awalan variabel lingkungan dengan nama katalog yang digunakan di Athena. Misalnya, jika katalog terdaftar di Athena adalah`mymysqlca`

Parameter	Deskripsi
	`talog` , maka nama variabel lingkungan adalah. `mymysqlca talog_connection_string`
`default`	Wajib. String koneksi default. String ini digunakan ketika katalog `lambda:${ }AWS_LAMBDA_FUNCTION_NAME` .

Contoh properti berikut adalah untuk fungsi MySql MUX Lambda yang mendukung dua instance database`mysql1`: (default), dan. `mysql2`

Properti	Nilai
`default`	`mysql://jdbc:mysql://mysql2 .host:3333/default? user=samp` le2&password=sample2
`mysql_catalog1_connection_string`	`mysql://jdbc:mysql://mysql1 .host:3306/default?${Test/RDS/ MySql1}`
`mysql_catalog2_connection_string`	`mysql://jdbc:mysql://mysql2 .host:3333/default? user=samp` le2&password=sample2

Memberikan kredensi

Untuk memberikan nama pengguna dan kata sandi untuk database Anda dalam string koneksi JDBC Anda, Anda dapat menggunakan properti string koneksi atau. AWS Secrets Manager

- Connection String - Nama pengguna dan kata sandi dapat ditentukan sebagai properti dalam string koneksi JDBC.

> **Important**
>
> Sebagai praktik keamanan terbaik, jangan gunakan kredensyal hardcode dalam variabel lingkungan atau string koneksi Anda. Untuk informasi tentang memindahkan rahasia

hardcode Anda AWS Secrets Manager, lihat Memindahkan rahasia hardcode ke AWS Secrets Manager dalam Panduan Pengguna.AWS Secrets Manager

- AWS Secrets Manager— Untuk menggunakan fitur Query Federasi Athena dengan, VPC AWS Secrets Manager yang terhubung ke fungsi Lambda Anda harus memiliki akses internet atau titik akhir VPC untuk terhubung ke Secrets Manager.

  Anda dapat memasukkan nama rahasia ke AWS Secrets Manager dalam string koneksi JDBC Anda. Konektor menggantikan nama rahasia dengan `password` nilai `username` dan dari Secrets Manager.

  Untuk instans database Amazon RDS, dukungan ini terintegrasi erat. Jika Anda menggunakan Amazon RDS, kami sangat menyarankan penggunaan AWS Secrets Manager dan rotasi kredensyal. Jika database Anda tidak menggunakan Amazon RDS, simpan kredensialnya sebagai JSON dalam format berikut:

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

Contoh string koneksi dengan nama rahasia

String berikut memiliki nama rahasia`${Test/RDS/MySql1}`.

```
mysql://jdbc:mysql://mysql1.host:3306/default?...&${Test/RDS/MySql1}&...
```

Konektor menggunakan nama rahasia untuk mengambil rahasia dan memberikan nama pengguna dan kata sandi, seperti pada contoh berikut.

```
mysql://jdbc:mysql://mysql1host:3306/default?...&user=sample2&password=sample2&...
```

Saat ini, konektor MySQL mengenali `user` properti dan JDBC. `password`

Menggunakan handler koneksi tunggal

Anda dapat menggunakan metadata koneksi tunggal berikut dan penangan rekaman untuk terhubung ke satu instance MySQL.

Jenis handler	Kelas
Pawang komposit	`MySqlCompositeHandler`
Penangan metadata	`MySqlMetadataHandler`
Rekam handler	`MySqlRecordHandler`

Parameter handler koneksi tunggal

Parameter	Deskripsi
`default`	Wajib. String koneksi default.

Penangan koneksi tunggal mendukung satu instance database dan harus menyediakan parameter string `default` koneksi. Semua string koneksi lainnya diabaikan.

Properti contoh berikut adalah untuk contoh MySQL tunggal didukung oleh fungsi Lambda.

Properti	Nilai
`default`	`mysql://mysql1.host:3306/default?secret=Test/RDS/` MySql1

Parameter tumpahan

Lambda SDK dapat menumpahkan data ke Amazon S3. Semua instance database yang diakses oleh fungsi Lambda yang sama tumpah ke lokasi yang sama.

Parameter	Deskripsi
`spill_bucket`	Wajib. Nama ember tumpahan.
`spill_prefix`	Wajib. Tumpahkan key prefix bucket.
`spill_put_request_headers`	(Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk

Parameter	Deskripsi
	menumpahkan (misalnya,). `putObject {"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.

Dukungan tipe data

Tabel berikut menunjukkan tipe data yang sesuai untuk JDBC dan Arrow.

JDBC	Panah
Boolean	Bit
Bilangan Bulat	Kecil
Pendek	Kecil
Bilangan Bulat	Int
Long	Bigint
float	Mengapung4
Ganda	Mengapung8
Tanggal	DateDay
Stempel Waktu	DateMilli
String	Varchar
Byte	Varbiner
BigDecimal	Decimal
ARRAY	Daftar

Partisi dan split

Partisi digunakan untuk menentukan cara menghasilkan split untuk konektor. Athena membangun kolom sintetis tipe `varchar` yang mewakili skema partisi untuk tabel untuk membantu konektor menghasilkan split. Konektor tidak mengubah definisi tabel yang sebenarnya.

Kinerja

MySQL mendukung partisi asli. Konektor MySQL Athena dapat mengambil data dari partisi ini secara paralel. Jika Anda ingin menanyakan kumpulan data yang sangat besar dengan distribusi partisi yang seragam, partisi asli sangat disarankan.

Konektor MySQL Athena melakukan pushdown predikat untuk mengurangi data yang dipindai oleh kueri. `LIMIT`klausa, predikat sederhana, dan ekspresi kompleks didorong ke konektor untuk mengurangi jumlah data yang dipindai dan mengurangi waktu eksekusi kueri.

Klausul LIMIT

`LIMIT N`Pernyataan mengurangi data yang dipindai oleh kueri. Dengan `LIMIT N` pushdown, konektor hanya mengembalikan `N` baris ke Athena.

Predikat

Predikat adalah ekspresi dalam `WHERE` klausa kueri SQL yang mengevaluasi nilai Boolean dan menyaring baris berdasarkan beberapa kondisi. Konektor MySQL Athena dapat menggabungkan ekspresi ini dan mendorongnya langsung ke MySQL untuk meningkatkan fungsionalitas dan untuk mengurangi jumlah data yang dipindai.

Operator konektor MySQL Athena berikut mendukung pushdown predikat:

- Boolean: DAN, ATAU, TIDAK
- KESETARAAN: SAMA, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Aritmatika: TAMBAHKAN, KURANGI, KALIKAN, BAGI, MODULUS, MENIADAKAN
- Lainnya: LIKE_PATTERN, IN

Contoh pushdown gabungan

Untuk kemampuan kueri yang ditingkatkan, gabungkan jenis pushdown, seperti pada contoh berikut:

```
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;
```

Untuk artikel tentang penggunaan pushdown predikat untuk meningkatkan kinerja dalam kueri federasi, termasuk MySQL, lihat Meningkatkan kueri federasi dengan pushdown predikat di Amazon Athena di Blog Big Data.AWS

Kueri passthrough

Konektor MySQL mendukung kueri passthrough. Kueri passthrough menggunakan fungsi tabel untuk mendorong kueri lengkap Anda ke sumber data untuk dieksekusi.

Untuk menggunakan kueri passthrough dengan MySQL, Anda dapat menggunakan sintaks berikut:

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

Contoh query berikut mendorong ke bawah query ke sumber data di MySQL. Kueri memilih semua kolom dalam `customer` tabel, membatasi hasilnya menjadi 10.

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

Informasi lisensi

Dengan menggunakan konektor ini, Anda mengakui penyertaan komponen pihak ketiga, daftar yang dapat ditemukan dalam file pom.xml untuk konektor ini, dan menyetujui persyaratan dalam masing-masing lisensi pihak ketiga yang disediakan dalam file LICENSE.txt di .com. GitHub

Sumber daya tambahan

Untuk informasi versi driver JDBC terbaru, lihat file pom.xml untuk konektor MySQL di.com. GitHub

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Amazon Athena Neptunus

Amazon Neptune adalah layanan basis data grafik yang cepat, andal, terkelola penuh yang memudahkan membangun dan menjalankan aplikasi yang bekerja dengan set data yang sangat terhubung. Neptunus yang dibuat khusus, mesin basis data grafik berperforma tinggi menyimpan miliaran hubungan secara optimal dan grafik kueri dengan latensi hanya milidetik. Untuk informasi selengkapnya, lihat Panduan Pengguna Neptune.

Konektor Amazon Athena Neptune memungkinkan Athena untuk berkomunikasi dengan contoh basis data grafik Neptune Anda, membuat data grafik Neptune Anda dapat diakses oleh kueri SQL.

Jika Anda mengaktifkan Lake Formation di akun Anda, peran IAM untuk konektor Lambda federasi Athena yang Anda gunakan di harus memiliki akses baca di AWS Serverless Application Repository Lake Formation ke. AWS Glue Data Catalog

Prasyarat

Menggunakan konektor Neptunus membutuhkan tiga langkah berikut.

- Menyiapkan cluster Neptunus
- Menyiapkan AWS Glue Data Catalog
- Menyebarkan konektor ke Anda Akun AWS. Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data. Untuk detail tambahan khusus tentang penerapan konektor Neptunus, lihat Menerapkan Konektor Neptunus Amazon Athena di.com. GitHub

Batasan

Saat ini, Konektor Neptunus memiliki batasan berikut.

- Memproyeksikan kolom, termasuk kunci utama (ID), tidak didukung.

Menyiapkan cluster Neptunus

Jika Anda tidak memiliki cluster Amazon Neptunus dan kumpulan data grafik properti di dalamnya yang ingin Anda gunakan, Anda harus mengaturnya.

Pastikan Anda memiliki gateway internet dan gateway NAT di VPC yang menampung cluster Neptunus Anda. Subnet pribadi yang digunakan oleh fungsi Lambda konektor Neptunus harus

memiliki rute ke internet melalui NAT Gateway ini. Fungsi Lambda konektor Neptunus menggunakan NAT Gateway untuk berkomunikasi dengannya. AWS Glue

Untuk petunjuk cara menyiapkan kluster Neptunus baru dan memuatnya dengan kumpulan data sampel, lihat Contoh Pengaturan Kluster Neptunus di.com. GitHub

Menyiapkan AWS Glue Data Catalog

Tidak seperti penyimpanan data relasional tradisional, node dan tepi DB grafik Neptunus tidak menggunakan skema yang ditetapkan. Setiap entri dapat memiliki bidang dan tipe data yang berbeda. Namun, karena konektor Neptunus mengambil metadata dari, Anda harus membuat database AWS Glue Data Catalog yang memiliki tabel AWS Glue dengan skema yang diperlukan. Setelah Anda membuat AWS Glue database dan tabel, konektor dapat mengisi daftar tabel yang tersedia untuk kueri dari Athena.

Mengaktifkan pencocokan kolom yang tidak peka huruf besar/kecil

Untuk menyelesaikan nama kolom dari tabel Neptunus Anda dengan casing yang benar bahkan ketika nama kolom semuanya lebih rendah, Anda dapat mengonfigurasi konektor Neptunus untuk AWS Glue pencocokan yang tidak peka huruf besar/kecil.

Untuk mengaktifkan fitur ini, atur variabel lingkungan fungsi konektor Neptunus Lambda ke. `enable_caseinsensitivematch` `true`

Menentukan parameter tabel AWS Glue glabel untuk nama tabel cased

Karena hanya AWS Glue mendukung nama tabel huruf kecil, penting untuk menentukan parameter `glabel` AWS Glue tabel saat Anda membuat tabel untuk Neptunus dan nama AWS Glue tabel Neptunus Anda menyertakan casing.

Dalam definisi AWS Glue tabel Anda, sertakan `glabel` parameter dan atur nilainya ke nama tabel Anda dengan casing aslinya. Ini memastikan bahwa casing yang benar dipertahankan saat AWS Glue berinteraksi dengan tabel Neptunus Anda. Contoh berikut menetapkan nilai `glabel` untuk nama tabel`Airport`.

```
glabel = Airport
```

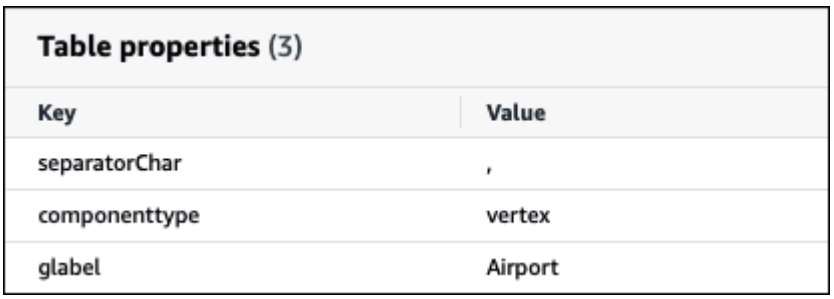


Table properties (3)

Key	Value
separatorChar	,
componenttype	vertex
glabel	Airport

Untuk informasi selengkapnya tentang pengaturan AWS Glue Data Catalog untuk bekerja dengan Neptunus, lihat AWS Glue Mengatur Katalog di.com. GitHub

Kinerja

Konektor Athena Neptunus melakukan pushdown predikat untuk mengurangi data yang dipindai oleh kueri. Namun, predikat yang menggunakan kunci utama mengakibatkan kegagalan kueri. `LIMIT`klausa mengurangi jumlah data yang dipindai, tetapi jika Anda tidak memberikan predikat, Anda harus mengharapkan `SELECT` kueri dengan `LIMIT` klausa untuk memindai setidaknya 16 MB data. Konektor Neptunus tahan terhadap pelambatan karena konkurensi.

Kueri passthrough

Konektor Neptunus mendukung kueri passthrough. Anda dapat menggunakan fitur ini untuk menjalankan kueri Gremlin pada grafik properti dan menjalankan kueri SPARQL pada data RDF.

Untuk membuat kueri passthrough dengan Neptunus, gunakan sintaks berikut:

```
SELECT * FROM TABLE(
        system.query(
            DATABASE => 'database_name',
            COLLECTION => 'collection_name',
            QUERY => 'query_string'
        ))
```

Berikut contoh filter query passthrough Neptunus untuk bandara dengan kode. `ATL` Kutipan tunggal berlipat ganda adalah untuk melarikan diri.

```
SELECT * FROM TABLE(
        system.query(
            DATABASE => 'graph-database',
            COLLECTION => 'airport',
            QUERY => 'g.V().has(''airport'', ''code'', ''ATL'').valueMap()'
```

```
    ))
```

Sumber daya tambahan

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Amazon Athena OpenSearch

OpenSearch Layanan

OpenSearch Konektor Amazon Athena memungkinkan Amazon Athena berkomunikasi dengan instans OpenSearch Anda sehingga Anda dapat menggunakan SQL untuk menanyakan data Anda. OpenSearch

> **Note**
>
> Karena masalah yang diketahui, OpenSearch konektor tidak dapat digunakan dengan VPC.

Jika Anda mengaktifkan Lake Formation di akun Anda, peran IAM untuk konektor Lambda federasi Athena yang Anda gunakan di harus memiliki akses baca di AWS Serverless Application Repository Lake Formation ke. AWS Glue Data Catalog

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Ketentuan

Istilah-istilah berikut berhubungan dengan OpenSearch konektor.

- Domain — Nama yang dihubungkan oleh konektor ini dengan titik akhir instans Anda OpenSearch . Domain juga digunakan sebagai nama database. Untuk OpenSearch instance yang ditentukan dalam OpenSearch Layanan Amazon, domain dapat ditemukan secara otomatis. Untuk contoh lain, Anda harus menyediakan pemetaan antara nama domain dan titik akhir.
- Indeks — Tabel database yang didefinisikan dalam OpenSearch contoh Anda.

- Pemetaan — Jika indeks adalah tabel database, maka pemetaan adalah skemanya (yaitu, definisi bidang dan atributnya).

  Konektor ini mendukung pengambilan metadata dari OpenSearch instance dan dari file. AWS Glue Data Catalog Jika konektor menemukan AWS Glue database dan tabel yang cocok dengan nama OpenSearch domain dan indeks Anda, konektor mencoba menggunakannya untuk definisi skema. Kami menyarankan Anda membuat AWS Glue tabel Anda sehingga merupakan superset dari semua bidang dalam OpenSearch indeks Anda.
- Document — Sebuah catatan dalam tabel database.
- Aliran data — Data berbasis waktu yang terdiri dari beberapa indeks dukungan. Untuk informasi selengkapnya, lihat Aliran data dalam OpenSearch dokumentasi dan Memulai aliran data di Panduan Pengembang OpenSearch Layanan Amazon.

> **Note**
>
> Karena indeks aliran data dibuat dan dikelola secara internal oleh pencarian terbuka, konektor memilih pemetaan skema dari indeks pertama yang tersedia. Untuk alasan ini, kami sangat menyarankan untuk menyiapkan AWS Glue tabel sebagai sumber metadata tambahan. Untuk informasi selengkapnya, lihat Menyiapkan database dan tabel di AWS Glue.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengonfigurasi konektor. OpenSearch

- spill_bucket - Menentukan bucket Amazon S3 untuk data yang melebihi batas fungsi Lambda.
- spill_prefix — (Opsional) Default ke subfolder dalam nama yang ditentukan. `spill_bucket` `athena-federation-spill` Kami menyarankan Anda mengonfigurasi siklus hidup penyimpanan Amazon S3 di lokasi ini untuk menghapus tumpahan yang lebih lama dari jumlah hari atau jam yang telah ditentukan sebelumnya.
- spill_put_request_headers — (Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject` `{"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.
- kms_key_id — (Opsional) Secara default, data apa pun yang tumpah ke Amazon S3 dienkripsi menggunakan mode enkripsi yang diautentikasi AES-GCM dan kunci yang dihasilkan secara acak.

Agar fungsi Lambda Anda menggunakan kunci enkripsi yang lebih kuat yang dihasilkan oleh KMS sepertia7e63k4b-8loc-40db-a2a1-4d0en2cd8331, Anda dapat menentukan ID kunci KMS.

- disable_spill_encryption — (Opsional) Ketika diatur ke, menonaktifkan enkripsi tumpahan. True Defaultnya False sehingga data yang tumpah ke S3 dienkripsi menggunakan AES-GCM — baik menggunakan kunci yang dihasilkan secara acak atau KMS untuk menghasilkan kunci. Menonaktifkan enkripsi tumpahan dapat meningkatkan kinerja, terutama jika lokasi tumpahan Anda menggunakan enkripsi sisi server.
- disable_glue — (Opsional) Jika ada dan disetel ke true, konektor tidak mencoba untuk mengambil metadata tambahan dari. AWS Glue
- query_timeout_cluster — Periode batas waktu, dalam hitungan detik, untuk kueri kesehatan klaster yang digunakan dalam pembuatan pemindaian paralel.
- query_timeout_search — Periode batas waktu, dalam hitungan detik, untuk kueri penelusuran yang digunakan dalam pengambilan dokumen dari indeks.
- auto_discover_endpoint — Boolean. Nilai default-nya true. Saat Anda menggunakan OpenSearch Layanan Amazon dan menyetel parameter ini ke true, konektor dapat secara otomatis menemukan domain dan titik akhir Anda dengan memanggil operasi API deskripsi atau daftar yang sesuai di Layanan. OpenSearch Untuk jenis OpenSearch instance lainnya (misalnya, di-host sendiri), Anda harus menentukan titik akhir domain terkait dalam variabel. domain_mapping Jikaauto_discover_endpoint=true, konektor menggunakan AWS kredensil untuk mengautentikasi ke Layanan. OpenSearch Jika tidak, konektor mengambil nama pengguna dan kredenal kata sandi dari variabel AWS Secrets Manager . domain_mapping
- domain_mapping - Digunakan hanya ketika auto_discover_endpoint disetel ke false dan mendefinisikan pemetaan antara nama domain dan titik akhir yang terkait. domain_mappingVariabel dapat mengakomodasi beberapa OpenSearch titik akhir dalam format berikut:

```
domain1=endpoint1,domain2=endpoint2,domain3=endpoint3,...
```

Untuk tujuan otentikasi ke OpenSearch titik akhir, konektor mendukung string substitusi yang disuntikkan menggunakan format ${SecretName}: dengan nama pengguna dan kata sandi yang diambil dari. AWS Secrets Manager Titik dua (:) di akhir ekspresi berfungsi sebagai pemisah dari sisa titik akhir.

> **Important**
>
> Sebagai praktik keamanan terbaik, jangan gunakan kredensi hardcode dalam variabel lingkungan atau string koneksi Anda. Untuk informasi tentang memindahkan rahasia hardcode Anda AWS Secrets Manager, lihat Memindahkan rahasia hardcode ke AWS Secrets Manager dalam Panduan Pengguna.AWS Secrets Manager

Contoh berikut menggunakan `opensearch-creds` rahasia.

```
movies=https://${opensearch-creds}:search-movies-ne...qu.us-east-1.es.amazonaws.com
```

Saat runtime, `${opensearch-creds}` dirender sebagai nama pengguna dan kata sandi, seperti pada contoh berikut.

```
movies=https://myusername@mypassword:search-movies-ne...qu.us-east-1.es.amazonaws.com
```

Dalam `domain_mapping` parameter, setiap pasangan domain-endpoint dapat menggunakan rahasia yang berbeda. *Rahasia itu sendiri harus ditentukan dalam format user_name @ password.* Meskipun kata sandi mungkin berisi @ tanda yang disematkan, yang pertama @ berfungsi sebagai pemisah dari *user_name*.

Penting juga untuk dicatat bahwa koma (,) dan tanda sama dengan (=) digunakan oleh konektor ini sebagai pemisah untuk pasangan domain-endpoint. Untuk alasan ini, Anda tidak boleh menggunakannya di mana pun di dalam rahasia yang disimpan.

Menyiapkan database dan tabel di AWS Glue

Konektor memperoleh informasi metadata dengan menggunakan atau. AWS Glue OpenSearch Anda dapat mengatur AWS Glue tabel sebagai sumber definisi metadata tambahan. Untuk mengaktifkan fitur ini, tentukan AWS Glue database dan tabel yang cocok dengan domain dan indeks sumber yang Anda tambahkan. Konektor juga dapat memanfaatkan definisi metadata yang disimpan dalam OpenSearch instance dengan mengambil pemetaan untuk indeks yang ditentukan.

Mendefinisikan metadata untuk array di OpenSearch

OpenSearch tidak memiliki tipe data array khusus. Bidang apa pun dapat berisi nol atau lebih nilai asalkan memiliki tipe data yang sama. Jika ingin digunakan OpenSearch sebagai sumber definisi metadata, Anda harus menentukan `_meta` properti untuk semua indeks yang digunakan dengan Athena untuk bidang yang akan dianggap sebagai daftar atau larik. Jika Anda gagal menyelesaikan langkah ini, kueri hanya mengembalikan elemen pertama di bidang daftar. Saat Anda menentukan `_meta` properti, nama bidang harus sepenuhnya memenuhi syarat untuk struktur JSON bersarang (misalnya`address.street`, di mana `street` adalah bidang bersarang di dalam struktur). `address`

Contoh berikut mendefinisikan `actor` dan `genre` daftar dalam `movies` tabel.

```
PUT movies/_mapping
{
  "_meta": {
    "actor": "list",
    "genre": "list"
  }
}
```

Jenis data

OpenSearch Konektor dapat mengekstrak definisi metadata dari salah satu AWS Glue atau instance. OpenSearch Konektor menggunakan pemetaan dalam tabel berikut untuk mengonversi definisi ke tipe data Apache Arrow, termasuk titik-titik yang dicatat di bagian berikut.

OpenSearch	Panah Apache	AWS Glue
teks, kata kunci, biner	VARCHAR	string
panjang	BIGINT	bigint
scaled_float	BIGINT	SCALED_FLOAT (...)
integer	INT	int
pendek	SMALLINT	smallint
byte	TINYINT	tinyint
double	FLOAT8	double

OpenSearch	Panah Apache	AWS Glue
mengapung, half_float	FLOAT4	float
boolean	BIT	boolean
tanggal, date_nanos	DATEMILLI	timestamp
Struktur JSON	STRUCT	STRUCT
_meta (untuk informasi, lihat bagianMendefinisikan metadata untuk array di OpenSearch.)	DAFTAR	ARRAY

Catatan tentang tipe data

- Saat ini, konektor hanya mendukung OpenSearch dan AWS Glue tipe data yang tercantum dalam tabel sebelumnya.
- A `scaled_float` adalah angka floating-point yang diskalakan oleh faktor penskalaan ganda tetap dan direpresentasikan sebagai dalam Apache Arrow. `BIGINT` Misalnya, 0,756 dengan faktor penskalaan 100 dibulatkan menjadi 76.
- *Untuk menentukan scaled_float in AWS Glue, Anda harus memilih jenis array kolom dan mendeklarasikan bidang menggunakan format SCALED_FLOAT (scaling_factor).*

  Contoh-contoh berikut ini valid:

```
SCALED_FLOAT(10.51)
SCALED_FLOAT(100)
SCALED_FLOAT(100.0)
```

  Contoh berikut tidak valid:

```
SCALED_FLOAT(10.)
SCALED_FLOAT(.5)
```

- Saat mengonversi dari `date_nanos` ke`DATEMILLI`, nanodetik dibulatkan ke milidetik terdekat. Nilai yang valid untuk `date` dan `date_nanos` termasuk, tetapi tidak terbatas pada, format berikut:

```
"2020-05-18T10:15:30.123456789"
"2020-05-15T06:50:01.123Z"
"2020-05-15T06:49:30.123-05:00"
1589525370001 (epoch milliseconds)
```

- An OpenSearch `binary` adalah representasi string dari nilai biner yang dikodekan menggunakan `Base64` dan dikonversi ke a. `VARCHAR`

Menjalankan kueri SQL

Berikut ini adalah contoh query DDL yang dapat Anda gunakan dengan konektor ini. Dalam contoh, *`function_name`* sesuai dengan nama fungsi Lambda Anda, *`domain`* adalah nama domain yang ingin Anda kueri, dan *`indeks adalah nama indeks`* Anda.

```
SHOW DATABASES in `lambda:function_name`
```

```
SHOW TABLES in `lambda:function_name`.domain
```

```
DESCRIBE `lambda:function_name`.domain.index
```

Kinerja

OpenSearch Konektor Athena mendukung pemindaian paralel berbasis shard. Konektor menggunakan informasi kesehatan cluster yang diambil dari OpenSearch instance untuk menghasilkan beberapa permintaan untuk kueri penelusuran dokumen. Permintaan dibagi untuk setiap pecahan dan dijalankan secara bersamaan.

Konektor juga menekan predikat sebagai bagian dari permintaan pencarian dokumennya. Contoh query berikut dan predikat menunjukkan bagaimana konektor menggunakan predikat push down.

Kueri

```
SELECT * FROM "lambda:elasticsearch".movies.movies
WHERE year >= 1955 AND year <= 1962 OR year = 1996
```

Predikat

```
(_exists_:year) AND year:([1955 TO 1962] OR 1996)
```

Kueri passthrough

OpenSearch Konektor mendukung kueri passthrough dan menggunakan bahasa Query DSL. Untuk informasi selengkapnya tentang kueri dengan Query DSL, lihat Kueri DSL di dokumentasi Elasticsearch atau Query DSL dalam dokumentasi. OpenSearch

Untuk menggunakan kueri passthrough dengan OpenSearch konektor, gunakan sintaks berikut:

```
SELECT * FROM TABLE(
        system.query(
            schema => 'schema_name',
            index => 'index_name',
            query => "{query_string}"
        ))
```

OpenSearch Contoh filter kueri passthrough berikut untuk karyawan dengan status pekerjaan aktif dalam `employee` indeks `default` skema.

```
SELECT * FROM TABLE(
        system.query(
            schema => 'default',
            index => 'employee',
            query => "{ ''bool'':{''filter'':{''term'':{''status'': ''active''}}}}"
        ))
```

Sumber daya tambahan

- Untuk artikel tentang penggunaan OpenSearch konektor Amazon Athena untuk menanyakan data di OpenSearch Layanan Amazon dan Amazon S3 dalam satu kueri, lihat Kueri data di Layanan Amazon menggunakan SQL dari OpenSearch Amazon Athena di Blog Big Data.AWS
- Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Amazon Athena Oracle

Konektor Amazon Athena untuk Oracle memungkinkan Amazon Athena menjalankan kueri SQL pada data yang disimpan di Oracle yang berjalan di lokasi atau di Amazon EC2 atau Amazon RDS. Anda juga dapat menggunakan konektor untuk menanyakan data pada Oracle exadata.

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Batasan

- Menulis operasi DDL tidak didukung.
- Dalam pengaturan multiplexer, bucket tumpahan dan awalan dibagikan di semua instance database.
- Batas Lambda yang relevan. Untuk informasi selengkapnya, lihat Kuota Lambda dalam Panduan Pengembang AWS Lambda .

Ketentuan

Istilah-istilah berikut berhubungan dengan konektor Oracle.

- Instans database — Instance apa pun dari database yang digunakan di tempat, di Amazon EC2, atau di Amazon RDS.
- Handler - Handler Lambda yang mengakses instance database Anda. Handler bisa untuk metadata atau untuk catatan data.
- Metadata handler — Penangan Lambda yang mengambil metadata dari instance database Anda.
- Record handler - Handler Lambda yang mengambil catatan data dari instance database Anda.
- Composite handler — Handler Lambda yang mengambil data metadata dan data dari instance database Anda.
- Properti atau parameter - Properti database yang digunakan oleh penangan untuk mengekstrak informasi database. Anda mengonfigurasi properti ini sebagai variabel lingkungan Lambda.
- Connection String — Sebuah string teks yang digunakan untuk membuat koneksi ke instance database.
- Katalog —AWS Glue Non-katalog yang terdaftar di Athena yang merupakan awalan yang diperlukan untuk properti. `connection_string`
- Multiplexing handler - Handler Lambda yang dapat menerima dan menggunakan beberapa koneksi database.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengkonfigurasi konektor Oracle.

String koneksi

Gunakan string koneksi JDBC dalam format berikut untuk terhubung ke instance database.

```
oracle://${jdbc_connection_string}
```

> **Note**
>
> Jika kata sandi Anda berisi karakter khusus (misalnya,`some.password`), lampirkan kata sandi Anda dalam tanda kutip ganda saat Anda meneruskannya ke string koneksi (misalnya,`"some.password"`). Kegagalan untuk melakukannya dapat mengakibatkan kesalahan yang ditentukan URL Oracle Tidak Valid.

Menggunakan handler multiplexing

Anda dapat menggunakan multiplexer untuk terhubung ke beberapa instance database dengan satu fungsi Lambda. Permintaan dirutekan dengan nama katalog. Gunakan kelas berikut di Lambda.

Handler	Kelas
Handler komposit	`OracleMuxCompositeHandler`
Penangan metadata	`OracleMuxMetadataHandler`
Rekam handler	`OracleMuxRecordHandler`

Parameter handler multiplexing

Parameter	Deskripsi
$*catalog*_connection_string	Wajib. Sebuah string koneksi instance database. Awalan variabel lingkungan dengan nama katalog yang digunakan di Athena.

Parameter	Deskripsi
	Misalnya, jika katalog terdaftar di Athena adalah`myoraclecatalog` , maka nama variabel lingkungan adalah. `myoraclecatalog_connection_string`
`default`	Wajib. String koneksi default. String ini digunakan ketika katalog `lambda:${ }AWS_LAMBDA_FUNCTION_NAME` .

Contoh properti berikut adalah untuk fungsi Oracle MUX Lambda yang mendukung dua instance database: `oracle1` (default), dan. `oracle2`

Properti	Nilai
`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`

Memberikan kredensi

Untuk memberikan nama pengguna dan kata sandi untuk database Anda dalam string koneksi JDBC Anda, Anda dapat menggunakan properti string koneksi atau. AWS Secrets Manager

- Connection String - Nama pengguna dan kata sandi dapat ditentukan sebagai properti dalam string koneksi JDBC.

> **Important**
>
> Sebagai praktik keamanan terbaik, jangan gunakan kredensyal hardcode dalam variabel lingkungan atau string koneksi Anda. Untuk informasi tentang memindahkan rahasia

hardcode Anda AWS Secrets Manager, lihat Memindahkan rahasia hardcode ke AWS Secrets Manager dalam Panduan Pengguna.AWS Secrets Manager

- AWS Secrets Manager— Untuk menggunakan fitur Query Federasi Athena dengan, VPC AWS Secrets Manager yang terhubung ke fungsi Lambda Anda harus memiliki akses internet atau titik akhir VPC untuk terhubung ke Secrets Manager.

  Anda dapat memasukkan nama rahasia ke AWS Secrets Manager dalam string koneksi JDBC Anda. Konektor menggantikan nama rahasia dengan `password` nilai `username` dan dari Secrets Manager.

  Untuk instans database Amazon RDS, dukungan ini terintegrasi erat. Jika Anda menggunakan Amazon RDS, kami sangat menyarankan penggunaan AWS Secrets Manager dan rotasi kredensyal. Jika database Anda tidak menggunakan Amazon RDS, simpan kredensialnya sebagai JSON dalam format berikut:

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

> **Note**
>
> Jika kata sandi Anda berisi karakter khusus (misalnya,`some.password`), lampirkan kata sandi Anda dalam tanda kutip ganda saat Anda menyimpannya di Secrets Manager (misalnya,`"some.password"`). Kegagalan untuk melakukannya dapat mengakibatkan kesalahan yang ditentukan URL Oracle Tidak Valid.

Contoh string koneksi dengan nama rahasia

String berikut memiliki nama rahasia`${Test/RDS/Oracle}`.

```
oracle://jdbc:oracle:thin:${Test/RDS/Oracle}@//hostname:port/servicename
```

Konektor menggunakan nama rahasia untuk mengambil rahasia dan memberikan nama pengguna dan kata sandi, seperti pada contoh berikut.

```
oracle://jdbc:oracle:thin:username/password@//hostname:port/servicename
```

Saat ini, konektor Oracle mengenali properti `UID` dan `PWD` JDBC.

Menggunakan handler koneksi tunggal

Anda dapat menggunakan metadata koneksi tunggal berikut dan penangan rekaman untuk terhubung ke satu instance Oracle.

Jenis handler	Kelas
Handler komposit	`OracleCompositeHandler`
Penangan metadata	`OracleMetadataHandler`
Rekam handler	`OracleRecordHandler`

Parameter handler koneksi tunggal

Parameter	Deskripsi
`default`	Wajib. String koneksi default.
`IsFIPSEnabled`	Tidak wajib. Setel ke `true` saat mode FIPS diaktifkan. Nilai default-nya `false`.

Penangan koneksi tunggal mendukung satu instance database dan harus menyediakan parameter string `default` koneksi. Semua string koneksi lainnya diabaikan.

Konektor mendukung koneksi berbasis SSL ke instans Amazon RDS. Support terbatas pada protokol Transport Layer Security (TLS) dan otentikasi server oleh klien. Otentikasi timbal balik itu tidak didukung di Amazon RDS. Baris kedua dalam tabel di bawah ini menunjukkan sintaks untuk menggunakan SSL.

Properti contoh berikut adalah untuk satu contoh Oracle didukung oleh fungsi Lambda.

Properti	Nilai
`default`	`oracle://jdbc:oracle:thin:${Test/RDS/Oracle}@//hostname:port/servicename`

Properti	Nilai
	`oracle://jdbc:oracle:thin:${Test/RDS/Oracle}@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS) (HOST=<HOST_NAME>)(PORT=))(CONNECT_DATA=(SID=))(SECURITY=(SSL_SERVER_CERT_DN=)))`

Parameter tumpahan

Lambda SDK dapat menumpahkan data ke Amazon S3. Semua instance database yang diakses oleh fungsi Lambda yang sama tumpah ke lokasi yang sama.

Parameter	Deskripsi
`spill_bucket`	Wajib. Nama ember tumpahan.
`spill_prefix`	Wajib. Tumpahkan key prefix bucket.
`spill_put_request_headers`	(Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject {"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.

Dukungan tipe data

Tabel berikut menunjukkan tipe data yang sesuai untuk JDBC, Oracle, dan Arrow.

JDBC	Oracle	Panah
Boolean	boolean	Bit
Bilangan Bulat	N/A	Mungil
Pendek	smallint	berkulit kecil
Bilangan Bulat	integer	Int

JDBC	Oracle	Panah
Long	bigint	Bigint
float	mengapung4	Mengapung4
Ganda	mengapung8	Mengapung8
Tanggal	tanggal	DateDay
Stempel Waktu	timestamp	DateMilli
String	text	Varchar
Byte	byte	Varbiner
BigDecimal	numeric(p,s)	Decimal
ARRAY	N/A (lihat catatan)	Daftar

Partisi dan split

Partisi digunakan untuk menentukan cara menghasilkan split untuk konektor. Athena membangun kolom sintetis tipe `varchar` yang mewakili skema partisi untuk tabel untuk membantu konektor menghasilkan split. Konektor tidak mengubah definisi tabel yang sebenarnya.

Kinerja

Oracle mendukung partisi asli. Konektor Athena Oracle dapat mengambil data dari partisi ini secara paralel. Jika Anda ingin menanyakan kumpulan data yang sangat besar dengan distribusi partisi yang seragam, partisi asli sangat disarankan. Memilih subset kolom secara signifikan mempercepat runtime kueri dan mengurangi data yang dipindai. Konektor Oracle tahan terhadap pelambatan karena konkurensi. Namun, runtime kueri cenderung panjang.

Konektor Athena Oracle melakukan pushdown predikat untuk mengurangi data yang dipindai oleh kueri. Predikat sederhana dan ekspresi kompleks didorong ke konektor untuk mengurangi jumlah data yang dipindai dan mengurangi waktu eksekusi kueri.

Predikat

Predikat adalah ekspresi dalam `WHERE` klausa kueri SQL yang mengevaluasi nilai Boolean dan menyaring baris berdasarkan beberapa kondisi. Konektor Athena Oracle dapat menggabungkan ekspresi ini dan mendorongnya langsung ke Oracle untuk meningkatkan fungsionalitas dan untuk mengurangi jumlah data yang dipindai.

Operator konektor Athena Oracle berikut mendukung pushdown predikat:

- Boolean: DAN, ATAU, TIDAK
- KESETARAAN: SAMA, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_NULL
- Aritmatika: TAMBAHKAN, KURANGI, KALIKAN, BAGI, MENIADAKAN
- Lainnya: LIKE_PATTERN, IN

Contoh pushdown gabungan

Untuk kemampuan kueri yang ditingkatkan, gabungkan jenis pushdown, seperti pada contoh berikut:

```
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%');
```

Kueri passthrough

Konektor Oracle mendukung kueri passthrough. Kueri passthrough menggunakan fungsi tabel untuk mendorong kueri lengkap Anda ke sumber data untuk dieksekusi.

Untuk menggunakan kueri passthrough dengan Oracle, Anda dapat menggunakan sintaks berikut:

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

Contoh query berikut mendorong ke bawah query ke sumber data di Oracle. Query memilih semua kolom dalam `customer` tabel.

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

Informasi lisensi

Dengan menggunakan konektor ini, Anda mengakui penyertaan komponen pihak ketiga, daftar yang dapat ditemukan dalam file pom.xml untuk konektor ini, dan menyetujui persyaratan dalam masing-masing lisensi pihak ketiga yang disediakan dalam file LICENSE.txt di .com. GitHub

Sumber daya tambahan

Untuk informasi versi driver JDBC terbaru, lihat file pom.xml untuk konektor Oracle di.com. GitHub

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Amazon Athena PostgreSQL

Konektor Amazon Athena PostgreSQL memungkinkan Athena mengakses database PostgreSQL Anda.

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Batasan

- Menulis operasi DDL tidak didukung.
- Dalam pengaturan multiplexer, bucket tumpahan dan awalan dibagikan di semua instance database.
- Batas Lambda yang relevan. Untuk informasi selengkapnya, lihat Kuota Lambda dalam Panduan Pengembang AWS Lambda .

Ketentuan

Istilah-istilah berikut berhubungan dengan konektor PostgreSQL.

- Instans database — Instance apa pun dari database yang digunakan di tempat, di Amazon EC2, atau di Amazon RDS.
- Handler - Handler Lambda yang mengakses instance database Anda. Handler bisa untuk metadata atau untuk catatan data.
- Metadata handler — Penangan Lambda yang mengambil metadata dari instance database Anda.
- Record handler - Handler Lambda yang mengambil catatan data dari instance database Anda.
- Composite handler — Handler Lambda yang mengambil data metadata dan data dari instance database Anda.
- Properti atau parameter - Properti database yang digunakan oleh penangan untuk mengekstrak informasi database. Anda mengonfigurasi properti ini sebagai variabel lingkungan Lambda.
- Connection String — Sebuah string teks yang digunakan untuk membuat koneksi ke instance database.
- Katalog —AWS Glue Non-katalog yang terdaftar di Athena yang merupakan awalan yang diperlukan untuk properti. `connection_string`
- Multiplexing handler - Handler Lambda yang dapat menerima dan menggunakan beberapa koneksi database.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengkonfigurasi konektor PostgreSQL.

String koneksi

Gunakan string koneksi JDBC dalam format berikut untuk terhubung ke instance database.

```
postgres://${jdbc_connection_string}
```

Menggunakan handler multiplexing

Anda dapat menggunakan multiplexer untuk terhubung ke beberapa instance database dengan satu fungsi Lambda. Permintaan dirutekan dengan nama katalog. Gunakan kelas berikut di Lambda.

Handler	Kelas
Pawang komposit	`PostGreSqlMuxCompositeHandler`

Handler	Kelas
Penangan metadata	`PostGreSqlMuxMetadataHandler`
Rekam handler	`PostGreSqlMuxRecordHandler`

Parameter handler multiplexing

Parameter	Deskripsi
`$`*`catalog`*`_connection_string`	Wajib. Sebuah string koneksi instance database. Awalan variabel lingkungan dengan nama katalog yang digunakan di Athena. Misalnya, jika katalog terdaftar di Athena adalah`mypostgrescatalog` , maka nama variabel lingkungan adalah. `mypostgrescatalog_connection_string`
`default`	Wajib. String koneksi default. String ini digunakan ketika katalog `lambda:${`*`AWS_LAMBDA_FUNCTION_NAME`*`}` .

Contoh properti berikut adalah untuk fungsi PostGreSql MUX Lambda yang mendukung dua instance database`postgres1`: (default), dan. `postgres2`

Properti	Nilai
`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`

Memberikan kredensi

Untuk memberikan nama pengguna dan kata sandi untuk database Anda dalam string koneksi JDBC Anda, Anda dapat menggunakan properti string koneksi atau. AWS Secrets Manager

- Connection String - Nama pengguna dan kata sandi dapat ditentukan sebagai properti dalam string koneksi JDBC.

  > **Important**
  >
  > Sebagai praktik keamanan terbaik, jangan gunakan kredensyal hardcode dalam variabel lingkungan atau string koneksi Anda. Untuk informasi tentang memindahkan rahasia hardcode Anda AWS Secrets Manager, lihat Memindahkan rahasia hardcode ke AWS Secrets Manager dalam Panduan Pengguna.AWS Secrets Manager

- AWS Secrets Manager— Untuk menggunakan fitur Query Federasi Athena dengan, VPC AWS Secrets Manager yang terhubung ke fungsi Lambda Anda harus memiliki akses internet atau titik akhir VPC untuk terhubung ke Secrets Manager.

  Anda dapat memasukkan nama rahasia ke AWS Secrets Manager dalam string koneksi JDBC Anda. Konektor menggantikan nama rahasia dengan `password` nilai `username` dan dari Secrets Manager.

  Untuk instans database Amazon RDS, dukungan ini terintegrasi erat. Jika Anda menggunakan Amazon RDS, kami sangat menyarankan penggunaan AWS Secrets Manager dan rotasi kredensyal. Jika database Anda tidak menggunakan Amazon RDS, simpan kredensialnya sebagai JSON dalam format berikut:

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

Contoh string koneksi dengan nama rahasia

String berikut memiliki nama rahasia`${Test/RDS/PostGres1}`.

```
postgres://jdbc:postgresql://postgres1.host:5432/default?...&${Test/RDS/PostGres1}&...
```

Konektor menggunakan nama rahasia untuk mengambil rahasia dan memberikan nama pengguna dan kata sandi, seperti pada contoh berikut.

```
postgres://jdbc:postgresql://postgres1.host:5432/
default?...&user=sample2&password=sample2&...
```

Saat ini, konektor PostgreSQL mengenali properti dan JDBC. `user` `password`

Mengaktifkan SSL

Untuk mendukung SSL dalam koneksi PostgreSQL Anda, tambahkan yang berikut ini ke string koneksi Anda:

```
&sslmode=verify-ca&sslfactory=org.postgresql.ssl.DefaultJavaSSLFactory
```

Contoh

Contoh string koneksi berikut tidak menggunakan SSL.

```
postgres://jdbc:postgresql://example-asdf-aurora-postgres-endpoint:5432/asdf?
user=someuser&password=somepassword
```

Untuk mengaktifkan SSL, ubah string sebagai berikut.

```
postgres://jdbc:postgresql://example-asdf-aurora-postgres-
endpoint:5432/asdf?user=someuser&password=somepassword&sslmode=verify-
ca&sslfactory=org.postgresql.ssl.DefaultJavaSSLFactory
```

Menggunakan handler koneksi tunggal

Anda dapat menggunakan metadata koneksi tunggal berikut dan penangan rekaman untuk terhubung ke satu instance PostgreSQL.

Jenis handler	Kelas
Pawang komposit	`PostGreSqlCompositeHandler`
Penangan metadata	`PostGreSqlMetadataHandler`
Rekam handler	`PostGreSqlRecordHandler`

Parameter handler koneksi tunggal

Parameter	Deskripsi
`default`	Wajib. String koneksi default.

Penangan koneksi tunggal mendukung satu instance database dan harus menyediakan parameter string `default` koneksi. Semua string koneksi lainnya diabaikan.

Properti contoh berikut adalah untuk instance PostgreSQL tunggal yang didukung oleh fungsi Lambda.

Properti	Nilai
`default`	`postgres://jdbc:postgresql://postgres1.host:5432/default?secret=${Test/RDS/PostgreSQL1}`

Parameter tumpahan

Lambda SDK dapat menumpahkan data ke Amazon S3. Semua instance database yang diakses oleh fungsi Lambda yang sama tumpah ke lokasi yang sama.

Parameter	Deskripsi
`spill_bucket`	Wajib. Nama ember tumpahan.
`spill_prefix`	Wajib. Tumpahkan key prefix bucket.
`spill_put_request_headers`	(Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject {"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.

Dukungan tipe data

Tabel berikut menunjukkan tipe data yang sesuai untuk JDBC, PostGre SQL, dan Arrow.

JDBC	PostGreSQL	Panah
Boolean	Boolean	Bit
Bilangan Bulat	N/A	Kecil
Pendek	smallint	Kecil
Bilangan Bulat	integer	Int
Long	bigint	Bigint
float	mengapung4	Mengapung4
Ganda	mengapung8	Mengapung8
Tanggal	tanggal	DateDay
Stempel Waktu	timestamp	DateMilli
String	text	Varchar
Byte	byte	Varbiner
BigDecimal	numeric(p,s)	Decimal
ARRAY	N/A (lihat catatan)	Daftar

Note

ARRAYTipe ini didukung untuk konektor PostgreSQL dengan batasan berikut: Array multidimensi (atau array bersarang) tidak didukung. `<data_type>[][]` Kolom dengan ARRAY tipe data yang tidak didukung dikonversi ke array elemen string (). `array<varchar>`

Partisi dan split

Partisi digunakan untuk menentukan cara menghasilkan split untuk konektor. Athena membangun kolom sintetis tipe `varchar` yang mewakili skema partisi untuk tabel untuk membantu konektor menghasilkan split. Konektor tidak mengubah definisi tabel yang sebenarnya.

Kinerja

PostgreSQL mendukung partisi asli. Konektor PostgreSQL Athena dapat mengambil data dari partisi ini secara paralel. Jika Anda ingin menanyakan kumpulan data yang sangat besar dengan distribusi partisi yang seragam, partisi asli sangat disarankan.

Konektor PostgreSQL Athena melakukan pushdown predikat untuk mengurangi data yang dipindai oleh kueri. `LIMIT`klausa, predikat sederhana, dan ekspresi kompleks didorong ke konektor untuk mengurangi jumlah data yang dipindai dan mengurangi waktu eksekusi kueri. Namun, memilih subset kolom terkadang menghasilkan runtime eksekusi kueri yang lebih lama.

Klausul LIMIT

`LIMIT N`Pernyataan mengurangi data yang dipindai oleh kueri. Dengan `LIMIT N` pushdown, konektor hanya mengembalikan `N` baris ke Athena.

Predikat

Predikat adalah ekspresi dalam `WHERE` klausa kueri SQL yang mengevaluasi nilai Boolean dan menyaring baris berdasarkan beberapa kondisi. Konektor PostgreSQL Athena dapat menggabungkan ekspresi ini dan mendorongnya langsung ke PostgreSQL untuk meningkatkan fungsionalitas dan untuk mengurangi jumlah data yang dipindai.

Operator konektor PostgreSQL Athena berikut mendukung pushdown predikat:

- Boolean: DAN, ATAU, TIDAK
- KESETARAAN: SAMA, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Aritmatika: TAMBAHKAN, KURANGI, KALIKAN, BAGI, MODULUS, MENIADAKAN
- Lainnya: LIKE_PATTERN, IN

Contoh pushdown gabungan

Untuk kemampuan kueri yang ditingkatkan, gabungkan jenis pushdown, seperti pada contoh berikut:

```
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;
```

Kueri passthrough

Konektor PostgreSQL mendukung kueri passthrough. Kueri passthrough menggunakan fungsi tabel untuk mendorong kueri lengkap Anda ke sumber data untuk dieksekusi.

Untuk menggunakan kueri passthrough dengan PostgreSQL, Anda dapat menggunakan sintaks berikut:

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

Contoh query berikut mendorong ke bawah query ke sumber data di PostgreSQL. Kueri memilih semua kolom dalam `customer` tabel, membatasi hasilnya menjadi 10.

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

Sumber daya tambahan

Untuk informasi versi driver JDBC terbaru, lihat file pom.xml untuk konektor PostgreSQL di.com. GitHub

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Amazon Athena Redis

Konektor Amazon Athena Redis memungkinkan Amazon Athena untuk berkomunikasi dengan contoh Redis Anda sehingga Anda dapat mengkueri data Redis Anda dengan SQL. Anda dapat menggunakan AWS Glue Data Catalog untuk memetakan pasangan nilai kunci Redis Anda ke dalam tabel virtual.

Tidak seperti penyimpanan data relasional tradisional, Redis tidak memiliki konsep tabel atau kolom. Sebaliknya, Redis menawarkan pola akses kunci-nilai di mana kunci pada dasarnya adalah a `string` dan nilainya adalah`string`,, `z-set` atau. `hmap`

Anda dapat menggunakan AWS Glue Data Catalog untuk membuat skema dan mengkonfigurasi tabel virtual. Properti tabel khusus memberi tahu konektor Athena Redis cara memetakan kunci dan nilai Redis Anda ke dalam tabel. Untuk informasi selengkapnya, lihat Menyiapkan database dan tabel di AWS Glue nanti dalam dokumen ini.

Jika Anda mengaktifkan Lake Formation di akun Anda, peran IAM untuk konektor Lambda federasi Athena yang Anda gunakan di harus memiliki akses baca di AWS Serverless Application Repository Lake Formation ke. AWS Glue Data Catalog

Konektor Amazon Athena Redis mendukung Amazon MemoryDB untuk Redis dan Amazon untuk Redis. ElastiCache

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.
- Siapkan VPC dan grup keamanan sebelum Anda menggunakan konektor ini. Untuk informasi selengkapnya, lihat Membuat VPC untuk konektor sumber data.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengkonfigurasi konektor Redis.

- spill_bucket - Menentukan bucket Amazon S3 untuk data yang melebihi batas fungsi Lambda.
- spill_prefix — (Opsional) Default ke subfolder dalam nama yang ditentukan. `spill_bucket` `athena-federation-spill` Kami menyarankan Anda mengonfigurasi siklus hidup penyimpanan Amazon S3 di lokasi ini untuk menghapus tumpahan yang lebih lama dari jumlah hari atau jam yang telah ditentukan sebelumnya.
- spill_put_request_headers — (Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject` `{"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.

- kms_key_id — (Opsional) Secara default, data apa pun yang tumpah ke Amazon S3 dienkripsi menggunakan mode enkripsi yang diautentikasi AES-GCM dan kunci yang dihasilkan secara acak. Agar fungsi Lambda Anda menggunakan kunci enkripsi yang lebih kuat yang dihasilkan oleh KMS sepertia`7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, Anda dapat menentukan ID kunci KMS.
- disable_spill_encryption — (Opsional) Ketika diatur ke, menonaktifkan enkripsi tumpahan. `True` Defaultnya `False` sehingga data yang tumpah ke S3 dienkripsi menggunakan AES-GCM — baik menggunakan kunci yang dihasilkan secara acak atau KMS untuk menghasilkan kunci. Menonaktifkan enkripsi tumpahan dapat meningkatkan kinerja, terutama jika lokasi tumpahan Anda menggunakan enkripsi sisi server.
- glue_catalog - (Opsional) Gunakan opsi ini untuk menentukan katalog lintas akun. AWS Glue Secara default, konektor mencoba untuk mendapatkan metadata dari akunnya sendiri AWS Glue .

Menyiapkan database dan tabel di AWS Glue

Untuk mengaktifkan AWS Glue tabel untuk digunakan dengan Redis, Anda dapat mengatur properti tabel berikut pada tabel:`redis-endpoint`,`redis-value-type`, dan salah satu `redis-keys-zset` atau`redis-key-prefix`.

Selain itu, setiap AWS Glue database yang berisi tabel Redis harus memiliki `redis-db-flag` properti URI database. Untuk menyetel properti `redis-db-flag` URI, gunakan AWS Glue konsol untuk mengedit database.

Daftar berikut menjelaskan properti tabel.

- redis-endpoint — (Wajib) `:` *`Kata sandi : port hostname`* dari server Redis yang berisi data untuk tabel ini (misalnya,`athena-federation-demo.cache.amazonaws.com:6379`) Atau, Anda dapat menyimpan titik akhir, atau bagian dari titik akhir, AWS Secrets Manager dengan menggunakan $ {*`Secret_Name`*} sebagai nilai properti tabel.

> **Note**
>
> Untuk menggunakan fitur Kueri Federasi Athena, VPC yang terhubung ke fungsi Lambda Anda harus memiliki akses internet atau titik akhir VPC untuk terhubung ke Secrets Manager. AWS Secrets Manager

- redis-keys-zset— (Diperlukan jika tidak `redis-key-prefix` digunakan) Daftar kunci yang dipisahkan koma yang nilainya adalah zset (misalnya,). `active-orders,pending-orders` Masing-masing nilai dalam zset diperlakukan sebagai kunci yang merupakan bagian dari tabel. Baik `redis-keys-zset` properti atau `redis-key-prefix` properti harus ditetapkan.
- redis-key-prefix— (Diperlukan jika `redis-keys-zset` tidak digunakan) Daftar awalan kunci yang dipisahkan koma untuk memindai nilai dalam tabel (misalnya,). `accounts-*,acct-` Baik `redis-key-prefix` properti atau `redis-keys-zset` properti harus ditetapkan.
- redis-value-type— (Wajib) Mendefinisikan bagaimana nilai untuk kunci didefinisikan oleh salah satu `redis-key-prefix` atau `redis-keys-zset` dipetakan ke tabel Anda. Sebuah peta literal ke satu kolom. Sebuah zset juga memetakan ke satu kolom, tetapi setiap tombol dapat menyimpan banyak baris. Sebuah hash memungkinkan setiap kunci menjadi baris dengan beberapa kolom (misalnya, hash, literal, atau zset.)
- redis-ssl-flag— (Opsional) Saat`True`, membuat koneksi Redis yang menggunakan SSL/TLS. Nilai default-nya `False`.
- redis-cluster-flag— (Opsional) Kapan`True`, aktifkan dukungan untuk instance Redis yang dikelompokkan. Nilai default-nya `False`.
- redis-db-number— (Opsional) Hanya berlaku untuk instance mandiri dan tidak berkerumun.) Tetapkan nomor ini (misalnya 1, 2, atau 3) untuk dibaca dari database Redis non-default. Defaultnya adalah database logis Redis 0. Nomor ini tidak merujuk ke database di Athena atau AWS Glue, tetapi ke database logis Redis. Untuk informasi selengkapnya, lihat INDEKS SELECT dalam dokumentasi Redis.

Jenis data

Konektor Redis mendukung tipe data berikut. Aliran Redis tidak didukung.

- Tali
- Hash
- Set Diurutkan (ZSet)

Semua nilai Redis diambil sebagai tipe `string` data. Kemudian mereka dikonversi ke salah satu tipe data Apache Arrow berikut berdasarkan bagaimana tabel Anda didefinisikan dalam AWS Glue Data Catalog.

AWS Glue tipe data	Tipe data Apache Arrow
int	INT
string	VARCHAR
bigint	BIGINT
double	FLOAT8
float	FLOAT4
smallint	SMALLINT
tinyint	TINYINT
boolean	BIT
biner	VARBINARY

Izin yang Diperlukan

Untuk detail lengkap tentang kebijakan IAM yang diperlukan konektor ini, tinjau `Policies` bagian file athena-redis.yaml. Daftar berikut merangkum izin yang diperlukan.

- Akses tulis Amazon S3 - Konektor memerlukan akses tulis ke lokasi di Amazon S3 untuk menumpahkan hasil dari kueri besar.
- Athena GetQueryExecution — Konektor menggunakan izin ini untuk gagal cepat ketika kueri Athena hulu telah dihentikan.
- AWS Glue Data Catalog— Konektor Redis memerlukan akses baca saja ke AWS Glue Data Catalog untuk mendapatkan informasi skema.
- CloudWatch Log — Konektor memerlukan akses ke CloudWatch Log untuk menyimpan log.
- AWS Secrets Manager akses baca - Jika Anda memilih untuk menyimpan detail titik akhir Redis di Secrets Manager, Anda harus memberikan akses konektor ke rahasia tersebut.
- Akses VPC — Konektor memerlukan kemampuan untuk memasang dan melepaskan antarmuka ke VPC Anda sehingga dapat terhubung dengannya dan berkomunikasi dengan instans Redis Anda.

Kinerja

Konektor Athena Redis mencoba untuk memparalelkan kueri terhadap instance Redis Anda sesuai dengan jenis tabel yang telah Anda tetapkan (misalnya, kunci zset atau kunci awalan).

Konektor Athena Redis melakukan pushdown predikat untuk mengurangi data yang dipindai oleh kueri. Namun, kueri yang berisi predikat terhadap kunci utama gagal dengan batas waktu. `LIMIT`klausa mengurangi jumlah data yang dipindai, tetapi jika Anda tidak memberikan predikat, Anda harus mengharapkan `SELECT` kueri dengan `LIMIT` klausa untuk memindai setidaknya 16 MB data. Konektor Redis tahan terhadap pelambatan karena konkurensi.

Informasi lisensi

Proyek konektor Amazon Athena Redis dilisensikan di bawah Lisensi Apache-2.0.

Sumber daya tambahan

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Pergeseran Merah Amazon Athena

Konektor Amazon Athena Redshift memungkinkan Amazon Athena mengakses database Amazon Redshift dan Amazon Redshift Serverless Anda, termasuk tampilan Redshift Tanpa Server. Anda dapat terhubung ke salah satu layanan menggunakan pengaturan konfigurasi string koneksi JDBC yang dijelaskan di halaman ini.

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Batasan

- Menulis operasi DDL tidak didukung.
- Dalam pengaturan multiplexer, bucket tumpahan dan awalan dibagikan di semua instance database.
- Batas Lambda yang relevan. Untuk informasi selengkapnya, lihat Kuota Lambda dalam Panduan Pengembang AWS Lambda .

- Karena Redshift tidak mendukung partisi eksternal, semua data yang ditentukan oleh kueri diambil setiap saat.

Ketentuan

Istilah-istilah berikut berhubungan dengan konektor Redshift.

- Instans database — Instance apa pun dari database yang digunakan di tempat, di Amazon EC2, atau di Amazon RDS.
- Handler - Handler Lambda yang mengakses instance database Anda. Handler bisa untuk metadata atau untuk catatan data.
- Metadata handler — Penangan Lambda yang mengambil metadata dari instance database Anda.
- Record handler - Handler Lambda yang mengambil catatan data dari instance database Anda.
- Composite handler — Handler Lambda yang mengambil data metadata dan data dari instance database Anda.
- Properti atau parameter - Properti database yang digunakan oleh penangan untuk mengekstrak informasi database. Anda mengonfigurasi properti ini sebagai variabel lingkungan Lambda.
- Connection String — Sebuah string teks yang digunakan untuk membuat koneksi ke instance database.
- Katalog —AWS Glue Non-katalog yang terdaftar di Athena yang merupakan awalan yang diperlukan untuk properti. `connection_string`
- Multiplexing handler - Handler Lambda yang dapat menerima dan menggunakan beberapa koneksi database.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengonfigurasi konektor Redshift.

String koneksi

Gunakan string koneksi JDBC dalam format berikut untuk terhubung ke instance database.

```
redshift://${jdbc_connection_string}
```

Menggunakan handler multiplexing

Anda dapat menggunakan multiplexer untuk terhubung ke beberapa instance database dengan satu fungsi Lambda. Permintaan dirutekan dengan nama katalog. Gunakan kelas berikut di Lambda.

Handler	Kelas
Handler komposit	`RedshiftMuxCompositeHandler`
Penangan metadata	`RedshiftMuxMetadataHandler`
Rekam handler	`RedshiftMuxRecordHandler`

Parameter handler multiplexing

Parameter	Deskripsi
$*catalog*_connection_string	Wajib. Sebuah string koneksi instance database. Awalan variabel lingkungan dengan nama katalog yang digunakan di Athena. Misalnya, jika katalog terdaftar di Athena adalah`myredshiftcatalog` , maka nama variabel lingkungan adalah. `myredshiftcatalog_connection_string`
`default`	Wajib. String koneksi default. String ini digunakan ketika katalog `lambda:${` *`}AWS_LAMBDA_FUNCTION_NAME`* .

Contoh properti berikut adalah untuk fungsi Redshift MUX Lambda yang mendukung dua instance database`redshift1`: (default), dan. `redshift2`

Properti	Nilai
`default`	`redshift://jdbc:redshift://redshift1.host:5439/dev?user=sample2&password=sample2`

Properti	Nilai
`redshift_catalog1_connection_string`	`redshift://jdbc:redshift://redshift1.host:3306/default?${Test/RDS/Redshift1}`
`redshift_catalog2_connection_string`	`redshift://jdbc:redshift://redshift2.host:3333/default?user=sample2&password=sample2`

Memberikan kredensi

Untuk memberikan nama pengguna dan kata sandi untuk database Anda dalam string koneksi JDBC Anda, Anda dapat menggunakan properti string koneksi atau. AWS Secrets Manager

- Connection String - Nama pengguna dan kata sandi dapat ditentukan sebagai properti dalam string koneksi JDBC.

  > **Important**
  >
  > Sebagai praktik keamanan terbaik, jangan gunakan kredensyal hardcode dalam variabel lingkungan atau string koneksi Anda. Untuk informasi tentang memindahkan rahasia hardcode Anda AWS Secrets Manager, lihat Memindahkan rahasia hardcode ke AWS Secrets Manager dalam Panduan Pengguna.AWS Secrets Manager

- AWS Secrets Manager— Untuk menggunakan fitur Query Federasi Athena dengan, VPC AWS Secrets Manager yang terhubung ke fungsi Lambda Anda harus memiliki akses internet atau titik akhir VPC untuk terhubung ke Secrets Manager.

  Anda dapat memasukkan nama rahasia ke AWS Secrets Manager dalam string koneksi JDBC Anda. Konektor menggantikan nama rahasia dengan `password` nilai `username` dan dari Secrets Manager.

  Untuk instans database Amazon RDS, dukungan ini terintegrasi dengan erat. Jika Anda menggunakan Amazon RDS, kami sangat menyarankan penggunaan AWS Secrets Manager dan rotasi kredenal. Jika database Anda tidak menggunakan Amazon RDS, simpan kredensialnya sebagai JSON dalam format berikut:

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

Contoh string koneksi dengan nama rahasia

String berikut memiliki nama rahasia $ {test/rds/Redshift1}.

```
redshift://jdbc:redshift://redshift1.host:3306/default?...&${Test/RDS/Redshift1}&...
```

Konektor menggunakan nama rahasia untuk mengambil rahasia dan memberikan nama pengguna dan kata sandi, seperti pada contoh berikut.

```
redshift://jdbc:redshift://redshift1.host:3306/
default?...&user=sample2&password=sample2&...
```

Saat ini, konektor Redshift mengenali properti `user` dan `password` JDBC.

Parameter tumpahan

Lambda SDK dapat menumpahkan data ke Amazon S3. Semua instance database yang diakses oleh fungsi Lambda yang sama tumpah ke lokasi yang sama.

Parameter	Deskripsi
`spill_bucket`	Wajib. Nama ember tumpahan.
`spill_prefix`	Wajib. Tumpahkan key prefix bucket.
`spill_put_request_headers`	(Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject {"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.

Dukungan tipe data

Tabel berikut menunjukkan tipe data yang sesuai untuk JDBC dan Apache Arrow.

JDBC	Panah
Boolean	Bit
Bilangan Bulat	Mungil
Pendek	berkulit kecil
Bilangan Bulat	Int
Long	Bigint
float	Mengapung4
Ganda	Mengapung8
Tanggal	DateDay
Stempel Waktu	DateMilli
String	Varchar
Byte	Varbiner
BigDecimal	Decimal
ARRAY	Daftar

Partisi dan split

Redshift tidak mendukung partisi eksternal. Untuk informasi tentang masalah terkait kinerja, lihatKinerja.

Kinerja

Konektor Athena Redshift melakukan pushdown predikat untuk mengurangi data yang dipindai oleh kueri. `LIMIT`klausa, `ORDER BY` klausa, predikat sederhana, dan ekspresi kompleks didorong ke konektor untuk mengurangi jumlah data yang dipindai dan mengurangi waktu eksekusi kueri. Namun, memilih subset kolom terkadang menghasilkan runtime eksekusi kueri yang lebih lama. Amazon Redshift sangat rentan terhadap perlambatan eksekusi kueri saat Anda menjalankan beberapa kueri secara bersamaan.

Klausul LIMIT

`LIMIT N`Pernyataan mengurangi data yang dipindai oleh kueri. Dengan `LIMIT N` pushdown, konektor hanya mengembalikan `N` baris ke Athena.

Kueri N teratas

`N`Kueri teratas menentukan urutan set hasil dan batas jumlah baris yang dikembalikan. Anda dapat menggunakan jenis kueri ini untuk menentukan nilai `N` maks teratas atau nilai `N` min teratas untuk kumpulan data Anda. Dengan `N` pushdown atas, konektor hanya mengembalikan baris yang `N` dipesan ke Athena.

Predikat

Predikat adalah ekspresi dalam `WHERE` klausa kueri SQL yang mengevaluasi nilai Boolean dan menyaring baris berdasarkan beberapa kondisi. Konektor Athena Redshift dapat menggabungkan ekspresi ini dan mendorongnya langsung ke Redshift untuk meningkatkan fungsionalitas dan untuk mengurangi jumlah data yang dipindai.

Operator konektor Athena Redshift berikut mendukung pushdown predikat:

- Boolean: DAN, ATAU, TIDAK
- KESETARAAN: SAMA, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Aritmatika: TAMBAHKAN, KURANGI, KALIKAN, BAGI, MODULUS, MENIADAKAN
- Lainnya: LIKE_PATTERN, IN

Contoh pushdown gabungan

Untuk kemampuan kueri yang ditingkatkan, gabungkan jenis pushdown, seperti pada contoh berikut:

```
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;
```

Untuk artikel tentang penggunaan pushdown predikat untuk meningkatkan kinerja dalam kueri federasi, termasuk Amazon Redshift, lihat Meningkatkan kueri federasi dengan pushdown predikat di Amazon Athena di Blog Big Data.AWS

Kueri passthrough

Konektor Redshift mendukung kueri passthrough. Kueri passthrough menggunakan fungsi tabel untuk mendorong kueri lengkap Anda ke sumber data untuk dieksekusi.

Untuk menggunakan kueri passthrough dengan Redshift, Anda dapat menggunakan sintaks berikut:

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

Contoh kueri berikut mendorong kueri ke sumber data di Redshift. Kueri memilih semua kolom dalam `customer` tabel, membatasi hasilnya menjadi 10.

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

Sumber daya tambahan

Untuk informasi versi driver JDBC terbaru, lihat file pom.xml untuk konektor Redshift di.com. GitHub

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Amazon Athena SAP HANA

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Batasan

- Menulis operasi DDL tidak didukung.

- Dalam pengaturan multiplexer, bucket tumpahan dan awalan dibagikan di semua instance database.
- Batas Lambda yang relevan. Untuk informasi selengkapnya, lihat Kuota Lambda dalam Panduan Pengembang AWS Lambda .
- Dalam SAP HANA, nama objek dikonversi ke huruf besar ketika mereka disimpan dalam database SAP HANA. Namun, karena nama dalam tanda kutip peka huruf besar/kecil, dimungkinkan untuk dua tabel memiliki nama yang sama dalam huruf kecil dan huruf besar (misalnya, `EMPLOYEE` dan`employee`).

  Di Athena Federated Query, nama tabel skema disediakan untuk fungsi Lambda dalam huruf kecil. Untuk mengatasi masalah ini, Anda dapat memberikan petunjuk `@schemaCase` kueri untuk mengambil data dari tabel yang memiliki nama peka huruf besar/kecil. Berikut ini adalah dua contoh query dengan petunjuk query.

```
SELECT *
FROM "lambda:saphanaconnector".SYSTEM."MY_TABLE@schemaCase=upper&tableCase=upper"
```

```
SELECT *
FROM "lambda:saphanaconnector".SYSTEM."MY_TABLE@schemaCase=upper&tableCase=lower"
```

Ketentuan

Istilah-istilah berikut berhubungan dengan konektor SAP HANA.

- Instans database — Instance apa pun dari database yang digunakan di tempat, di Amazon EC2, atau di Amazon RDS.
- Handler - Handler Lambda yang mengakses instance database Anda. Handler bisa untuk metadata atau untuk catatan data.
- Metadata handler — Penangan Lambda yang mengambil metadata dari instance database Anda.
- Record handler - Handler Lambda yang mengambil catatan data dari instance database Anda.
- Composite handler — Handler Lambda yang mengambil data metadata dan data dari instance database Anda.
- Properti atau parameter - Properti database yang digunakan oleh penangan untuk mengekstrak informasi database. Anda mengonfigurasi properti ini sebagai variabel lingkungan Lambda.

- Connection String — Sebuah string teks yang digunakan untuk membuat koneksi ke instance database.
- Katalog —AWS Glue Non-katalog yang terdaftar di Athena yang merupakan awalan yang diperlukan untuk properti. `connection_string`
- Multiplexing handler - Handler Lambda yang dapat menerima dan menggunakan beberapa koneksi database.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengkonfigurasi konektor SAP HANA.

String koneksi

Gunakan string koneksi JDBC dalam format berikut untuk terhubung ke instance database.

```
saphana://${jdbc_connection_string}
```

Menggunakan handler multiplexing

Anda dapat menggunakan multiplexer untuk terhubung ke beberapa instance database dengan satu fungsi Lambda. Permintaan dirutekan dengan nama katalog. Gunakan kelas berikut di Lambda.

Handler	Kelas
Handler komposit	`SaphanaMuxCompositeHandler`
Penangan metadata	`SaphanaMuxMetadataHandler`
Rekam handler	`SaphanaMuxRecordHandler`

Parameter handler multiplexing

Parameter	Deskripsi
$*catalog*_connection_string	Wajib. Sebuah string koneksi instance database. Awalan variabel lingkungan dengan nama katalog yang digunakan di Athena. Misalnya, jika katalog terdaftar di Athena adalah`mysaphana`

Parameter	Deskripsi
	`catalog` , maka nama variabel lingkungan adalah. `mysaphana catalog_connection_string`
`default`	Wajib. String koneksi default. String ini digunakan ketika katalog `lambda:${ }AWS_LAMBDA_FUNCTION_NAME` .

Contoh properti berikut adalah untuk fungsi Saphana MUX Lambda yang mendukung dua instance database: `saphana1` (default), dan. `saphana2`

Properti	Nilai
`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`

Memberikan kredensi

Untuk memberikan nama pengguna dan kata sandi untuk database Anda dalam string koneksi JDBC Anda, Anda dapat menggunakan properti string koneksi atau. AWS Secrets Manager

- Connection String - Nama pengguna dan kata sandi dapat ditentukan sebagai properti dalam string koneksi JDBC.

> **Important**
>
> Sebagai praktik keamanan terbaik, jangan gunakan kredensyal hardcode dalam variabel lingkungan atau string koneksi Anda. Untuk informasi tentang memindahkan rahasia

hardcode Anda AWS Secrets Manager, lihat Memindahkan rahasia hardcode ke AWS Secrets Manager dalam Panduan Pengguna.AWS Secrets Manager

- AWS Secrets Manager— Untuk menggunakan fitur Query Federasi Athena dengan, VPC AWS Secrets Manager yang terhubung ke fungsi Lambda Anda harus memiliki akses internet atau titik akhir VPC untuk terhubung ke Secrets Manager.

  Anda dapat memasukkan nama rahasia ke AWS Secrets Manager dalam string koneksi JDBC Anda. Konektor menggantikan nama rahasia dengan `password` nilai `username` dan dari Secrets Manager.

  Untuk instans database Amazon RDS, dukungan ini terintegrasi erat. Jika Anda menggunakan Amazon RDS, kami sangat menyarankan penggunaan AWS Secrets Manager dan rotasi kredensyal. Jika database Anda tidak menggunakan Amazon RDS, simpan kredensialnya sebagai JSON dalam format berikut:

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

Contoh string koneksi dengan nama rahasia

String berikut memiliki nama rahasia`${Test/RDS/Saphana1}`.

```
saphana://jdbc:sap://saphana1.host:port/?${Test/RDS/Saphana1}&...
```

Konektor menggunakan nama rahasia untuk mengambil rahasia dan memberikan nama pengguna dan kata sandi, seperti pada contoh berikut.

```
saphana://jdbc:sap://saphana1.host:port/?user=sample2&password=sample2&...
```

Saat ini, konektor SAP HANA mengenali properti `user` dan `password` JDBC.

Menggunakan handler koneksi tunggal

Anda dapat menggunakan metadata koneksi tunggal berikut dan penangan rekaman untuk terhubung ke satu instance SAP HANA.

Jenis handler	Kelas
Handler komposit	`SaphanaCompositeHandler`
Penangan metadata	`SaphanaMetadataHandler`
Rekam handler	`SaphanaRecordHandler`

Parameter handler koneksi tunggal

Parameter	Deskripsi
`default`	Wajib. String koneksi default.

Penangan koneksi tunggal mendukung satu instance database dan harus menyediakan parameter string `default` koneksi. Semua string koneksi lainnya diabaikan.

Properti contoh berikut adalah untuk satu instance SAP HANA yang didukung oleh fungsi Lambda.

Properti	Nilai
`default`	`saphana://jdbc:sap://saphana1.host:port/?secret=Test/RDS/Saphana1`

Parameter tumpahan

Lambda SDK dapat menumpahkan data ke Amazon S3. Semua instance database yang diakses oleh fungsi Lambda yang sama tumpah ke lokasi yang sama.

Parameter	Deskripsi
`spill_bucket`	Wajib. Nama ember tumpahan.
`spill_prefix`	Wajib. Tumpahkan key prefix bucket.
`spill_put_request_headers`	(Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk

Parameter	Deskripsi
	menumpahkan (misalnya,). `putObject {"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.

Dukungan tipe data

Tabel berikut menunjukkan tipe data yang sesuai untuk JDBC dan Apache Arrow.

JDBC	Panah
Boolean	Bit
Bilangan Bulat	Mungil
Pendek	Orang kecil
Bilangan Bulat	Int
Long	Bigint
float	Mengapung4
Ganda	Mengapung8
Tanggal	DateDay
Stempel Waktu	DateMilli
String	Varchar
Byte	Varbiner
BigDecimal	Decimal
ARRAY	Daftar

Konversi jenis data

Selain konversi JDBC ke Arrow, konektor melakukan konversi tertentu lainnya untuk membuat sumber SAP HANA dan tipe data Athena kompatibel. Konversi ini membantu memastikan bahwa kueri berhasil dieksekusi. Tabel berikut menunjukkan konversi ini.

Tipe data sumber (SAP HANA)	Tipe data yang dikonversi (Athena)
DECIMAL	BIGINT
INTEGER	INT
DATE	TANGGAL
TIMESTAMP	DATEMILLI

Semua tipe data lain yang tidak didukung dikonversi ke`VARCHAR`.

Partisi dan split

Partisi diwakili oleh kolom partisi tunggal tipe`Integer`. Kolom berisi nama partisi dari partisi yang didefinisikan pada tabel SAP HANA. Untuk tabel yang tidak memiliki nama partisi, * dikembalikan, yang setara dengan satu partisi. Partisi setara dengan split.

Nama	Tipe	Deskripsi
PART_ID	Bilangan Bulat	Dinamakan partisi di SAP HANA.

Kinerja

SAP HANA mendukung partisi asli. Konektor Athena SAP HANA dapat mengambil data dari partisi ini secara paralel. Jika Anda ingin menanyakan kumpulan data yang sangat besar dengan distribusi partisi yang seragam, partisi asli sangat disarankan. Memilih subset kolom secara signifikan mempercepat runtime kueri dan mengurangi data yang dipindai. Konektor menunjukkan pelambatan yang signifikan, dan terkadang kegagalan kueri, karena konkurensi.

Konektor Athena SAP HANA melakukan pushdown predikat untuk mengurangi data yang dipindai oleh kueri. `LIMIT`klausa, predikat sederhana, dan ekspresi kompleks didorong ke konektor untuk mengurangi jumlah data yang dipindai dan mengurangi waktu eksekusi kueri.

Klausul LIMIT

`LIMIT N`Pernyataan mengurangi data yang dipindai oleh kueri. Dengan `LIMIT N` pushdown, konektor hanya mengembalikan `N` baris ke Athena.

Predikat

Predikat adalah ekspresi dalam `WHERE` klausa kueri SQL yang mengevaluasi nilai Boolean dan menyaring baris berdasarkan beberapa kondisi. Konektor Athena SAP HANA dapat menggabungkan ekspresi ini dan mendorongnya langsung ke SAP HANA untuk meningkatkan fungsionalitas dan untuk mengurangi jumlah data yang dipindai.

Operator konektor Athena SAP HANA berikut mendukung pushdown predikat:

- Boolean: DAN, ATAU, TIDAK
- KESETARAAN: SAMA, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Aritmatika: TAMBAHKAN, KURANGI, KALIKAN, BAGI, MODULUS, MENIADAKAN
- Lainnya: LIKE_PATTERN, IN

Contoh pushdown gabungan

Untuk kemampuan kueri yang ditingkatkan, gabungkan jenis pushdown, seperti pada contoh berikut:

```
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;
```

Kueri passthrough

Konektor SAP HANA mendukung kueri passthrough. Kueri passthrough menggunakan fungsi tabel untuk mendorong kueri lengkap Anda ke sumber data untuk dieksekusi.

Untuk menggunakan kueri passthrough dengan SAP HANA, Anda dapat menggunakan sintaks berikut:

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

Contoh query berikut mendorong ke bawah query ke sumber data di SAP HANA. Kueri memilih semua kolom dalam `customer` tabel, membatasi hasilnya menjadi 10.

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

Informasi lisensi

Dengan menggunakan konektor ini, Anda mengakui penyertaan komponen pihak ketiga, daftar yang dapat ditemukan dalam file pom.xml untuk konektor ini, dan menyetujui persyaratan dalam masing-masing lisensi pihak ketiga yang disediakan dalam file LICENSE.txt di .com. GitHub

Sumber daya tambahan

Untuk informasi versi driver JDBC terbaru, lihat file pom.xml untuk konektor SAP HANA di.com. GitHub

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Kepingan Salju Amazon Athena

Konektor Amazon Athena untuk Snowflake memungkinkan Amazon Athena menjalankan kueri SQL pada data yang disimpan dalam database SQL Snowflake atau instans RDS menggunakan JDBC.

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Batasan

- Menulis operasi DDL tidak didukung.
- Dalam pengaturan multiplexer, bucket tumpahan dan awalan dibagikan di semua instance database.
- Batas Lambda yang relevan. Untuk informasi selengkapnya, lihat Kuota Lambda dalam Panduan Pengembang AWS Lambda .
- Saat ini, tampilan Snowflake dengan single split didukung.
- Di Snowflake, karena nama objek peka huruf besar/kecil, dua tabel dapat memiliki nama yang sama dalam huruf kecil dan huruf besar (misalnya, `EMPLOYEE` dan`employee`). Di Athena Federated Query, nama tabel skema disediakan untuk fungsi Lambda dalam huruf kecil. Untuk mengatasi masalah ini, Anda dapat memberikan petunjuk `@schemaCase` kueri untuk mengambil data dari tabel yang memiliki nama peka huruf besar/kecil. Berikut ini adalah dua contoh query dengan petunjuk query.

```
SELECT *
FROM "lambda:snowflakeconnector".SYSTEM."MY_TABLE@schemaCase=upper&tableCase=upper"
```

```
SELECT *
FROM "lambda:snowflakeconnector".SYSTEM."MY_TABLE@schemaCase=upper&tableCase=lower"
```

Ketentuan

Istilah-istilah berikut berhubungan dengan konektor Snowflake.

- Instans database — Instance apa pun dari database yang digunakan di tempat, di Amazon EC2, atau di Amazon RDS.
- Handler - Handler Lambda yang mengakses instance database Anda. Handler bisa untuk metadata atau untuk catatan data.
- Metadata handler — Penangan Lambda yang mengambil metadata dari instance database Anda.
- Record handler - Handler Lambda yang mengambil catatan data dari instance database Anda.
- Composite handler — Handler Lambda yang mengambil data metadata dan data dari instance database Anda.

- Properti atau parameter - Properti database yang digunakan oleh penangan untuk mengekstrak informasi database. Anda mengonfigurasi properti ini sebagai variabel lingkungan Lambda.
- Connection String — Sebuah string teks yang digunakan untuk membuat koneksi ke instance database.
- Katalog —AWS Glue Non-katalog yang terdaftar di Athena yang merupakan awalan yang diperlukan untuk properti. `connection_string`
- Multiplexing handler - Handler Lambda yang dapat menerima dan menggunakan beberapa koneksi database.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengkonfigurasi konektor Snowflake.

String koneksi

Gunakan string koneksi JDBC dalam format berikut untuk terhubung ke instance database.

```
snowflake://${jdbc_connection_string}
```

Menggunakan handler multiplexing

Anda dapat menggunakan multiplexer untuk terhubung ke beberapa instance database dengan satu fungsi Lambda. Permintaan dirutekan dengan nama katalog. Gunakan kelas berikut di Lambda.

Handler	Kelas
Handler komposit	`SnowflakeMuxCompositeHandler`
Penangan metadata	`SnowflakeMuxMetadataHandler`
Rekam handler	`SnowflakeMuxRecordHandler`

Parameter handler multiplexing

Parameter	Deskripsi
$*catalog*_connection_string	Wajib. Sebuah string koneksi instance database. Awalan variabel lingkungan dengan nama katalog yang digunakan di Athena.

Parameter	Deskripsi
	Misalnya, jika katalog terdaftar di Athena adalah`mysnowflakecatalog` , maka nama variabel lingkungan adalah. `mysnowflakecatalog_connection_string`
`default`	Wajib. String koneksi default. String ini digunakan ketika katalog `lambda:${ }AWS_LAMBDA_FUNCTION_NAME` .

Contoh properti berikut adalah untuk fungsi Snowflake MUX Lambda yang mendukung dua instance database`snowflake1`: (default), dan. `snowflake2`

Properti	Nilai
`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`

Memberikan kredensi

Untuk memberikan nama pengguna dan kata sandi untuk database Anda dalam string koneksi JDBC Anda, Anda dapat menggunakan properti string koneksi atau. AWS Secrets Manager

- Connection String - Nama pengguna dan kata sandi dapat ditentukan sebagai properti dalam string koneksi JDBC.

> **Important**
>
> Sebagai praktik keamanan terbaik, jangan gunakan kredensyal hardcode dalam variabel lingkungan atau string koneksi Anda. Untuk informasi tentang memindahkan rahasia hardcode Anda AWS Secrets Manager, lihat Memindahkan rahasia hardcode ke AWS Secrets Manager dalam Panduan Pengguna.AWS Secrets Manager

- AWS Secrets Manager— Untuk menggunakan fitur Query Federasi Athena dengan, VPC AWS Secrets Manager yang terhubung ke fungsi Lambda Anda harus memiliki akses internet atau titik akhir VPC untuk terhubung ke Secrets Manager.

  Anda dapat memasukkan nama rahasia ke AWS Secrets Manager dalam string koneksi JDBC Anda. Konektor menggantikan nama rahasia dengan `password` nilai `username` dan dari Secrets Manager.

  Untuk instans database Amazon RDS, dukungan ini terintegrasi erat. Jika Anda menggunakan Amazon RDS, kami sangat menyarankan penggunaan AWS Secrets Manager dan rotasi kredensyal. Jika database Anda tidak menggunakan Amazon RDS, simpan kredensialnya sebagai JSON dalam format berikut:

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

Contoh string koneksi dengan nama rahasia

String berikut memiliki nama rahasia`${Test/RDS/Snowflake1}`.

```
snowflake://jdbc:snowflake://snowflake1.host:port/?
warehouse=warehousename&db=db1&schema=schema1${Test/RDS/Snowflake1}&...
```

Konektor menggunakan nama rahasia untuk mengambil rahasia dan memberikan nama pengguna dan kata sandi, seperti pada contoh berikut.

```
snowflake://jdbc:snowflake://snowflake1.host:port/
warehouse=warehousename&db=db1&schema=schema1&user=sample2&password=sample2&...
```

Saat ini, Snowflake mengenali properti `user` dan `password` JDBC. Ini juga menerima nama pengguna dan kata sandi dalam kata *`/sandi format nama pengguna`* tanpa kunci `user` ataupun`password`.

Menggunakan handler koneksi tunggal

Anda dapat menggunakan metadata koneksi tunggal berikut dan penangan rekaman untuk terhubung ke satu instance Snowflake.

Jenis handler	Kelas
Handler komposit	`SnowflakeCompositeHandler`
Penangan metadata	`SnowflakeMetadataHandler`
Rekam handler	`SnowflakeRecordHandler`

Parameter handler koneksi tunggal

Parameter	Deskripsi
`default`	Wajib. String koneksi default.

Penangan koneksi tunggal mendukung satu instance database dan harus menyediakan parameter string `default` koneksi. Semua string koneksi lainnya diabaikan.

Properti contoh berikut adalah untuk instance Snowflake tunggal yang didukung oleh fungsi Lambda.

Properti	Nilai
`default`	`snowflake://jdbc:snowflake://snowflake1.host:port/?secret=Test/RDS/Snowflake1`

Parameter tumpahan

Lambda SDK dapat menumpahkan data ke Amazon S3. Semua instance database yang diakses oleh fungsi Lambda yang sama tumpah ke lokasi yang sama.

Parameter	Deskripsi
`spill_bucket`	Wajib. Nama ember tumpahan.
`spill_prefix`	Wajib. Tumpahkan key prefix bucket.
`spill_put_request_headers`	(Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject {"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.

Dukungan tipe data

Tabel berikut menunjukkan tipe data yang sesuai untuk JDBC dan Apache Arrow.

JDBC	Panah
Boolean	Bit
Bilangan Bulat	Mungil
Pendek	berkulit kecil
Bilangan Bulat	Int
Long	Bigint
float	Mengapung4
Ganda	Mengapung8
Tanggal	DateDay
Stempel Waktu	DateMilli

JDBC	Panah
String	Varchar
Byte	Varbiner
BigDecimal	Decimal
ARRAY	Daftar

Konversi jenis data

Selain konversi JDBC ke Arrow, konektor melakukan konversi tertentu lainnya untuk membuat sumber Snowflake dan tipe data Athena kompatibel. Konversi ini membantu memastikan bahwa kueri berhasil dieksekusi. Tabel berikut menunjukkan konversi ini.

Tipe data sumber (Snowflake)	Tipe data yang dikonversi (Athena)
TIMESTAMP	TIMESTAMPMILLI
DATE	TIMESTAMPMILLI
INTEGER	INT
DECIMAL	BIGINT
TIMESTAMP_NTZ	TIMESTAMPMILLI

Semua tipe data lain yang tidak didukung dikonversi ke`VARCHAR`.

Partisi dan split

Partisi digunakan untuk menentukan cara menghasilkan split untuk konektor. Athena membangun kolom sintetis tipe `varchar` yang mewakili skema partisi untuk tabel untuk membantu konektor menghasilkan split. Konektor tidak mengubah definisi tabel yang sebenarnya.

Untuk membuat kolom sintetis ini dan partisi, Athena memerlukan kunci primer untuk didefinisikan. Namun, karena Snowflake tidak memberlakukan kendala kunci utama, Anda harus menegakkan keunikan sendiri. Kegagalan untuk melakukannya menyebabkan Athena default ke satu split.

Kinerja

Untuk kinerja optimal, gunakan filter dalam kueri bila memungkinkan. Selain itu, kami sangat merekomendasikan partisi asli untuk mengambil kumpulan data besar yang memiliki distribusi partisi seragam. Memilih subset kolom secara signifikan mempercepat runtime kueri dan mengurangi data yang dipindai. Konektor Snowflake tahan terhadap pelambatan karena konkurensi.

Konektor Athena Snowflake melakukan pushdown predikat untuk mengurangi data yang dipindai oleh kueri. `LIMIT`klausa, predikat sederhana, dan ekspresi kompleks didorong ke konektor untuk mengurangi jumlah data yang dipindai dan mengurangi waktu eksekusi kueri.

Klausul LIMIT

`LIMIT N`Pernyataan mengurangi data yang dipindai oleh kueri. Dengan `LIMIT N` pushdown, konektor hanya mengembalikan `N` baris ke Athena.

Predikat

Predikat adalah ekspresi dalam `WHERE` klausa kueri SQL yang mengevaluasi nilai Boolean dan menyaring baris berdasarkan beberapa kondisi. Konektor Athena Snowflake dapat menggabungkan ekspresi ini dan mendorongnya langsung ke Snowflake untuk meningkatkan fungsionalitas dan untuk mengurangi jumlah data yang dipindai.

Operator konektor Athena Snowflake berikut mendukung pushdown predikat:

- Boolean: DAN, ATAU, TIDAK
- KESETARAAN: SAMA, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Aritmatika: TAMBAHKAN, KURANGI, KALIKAN, BAGI, MODULUS, MENIADAKAN
- Lainnya: LIKE_PATTERN, IN

Contoh pushdown gabungan

Untuk kemampuan kueri yang ditingkatkan, gabungkan jenis pushdown, seperti pada contoh berikut:

```
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;
```

Kueri passthrough

Konektor Snowflake mendukung kueri passthrough. Kueri passthrough menggunakan fungsi tabel untuk mendorong kueri lengkap Anda ke sumber data untuk dieksekusi.

Untuk menggunakan kueri passthrough dengan Snowflake, Anda dapat menggunakan sintaks berikut:

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

Contoh query berikut mendorong ke bawah query ke sumber data di Snowflake. Kueri memilih semua kolom dalam `customer` tabel, membatasi hasilnya menjadi 10.

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

Informasi lisensi

Dengan menggunakan konektor ini, Anda mengakui penyertaan komponen pihak ketiga, daftar yang dapat ditemukan dalam file pom.xml untuk konektor ini, dan menyetujui persyaratan dalam masing-masing lisensi pihak ketiga yang disediakan dalam file LICENSE.txt di .com. GitHub

Sumber daya tambahan

Untuk informasi versi driver JDBC terbaru, lihat file pom.xml untuk konektor Snowflake di.com. GitHub

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Amazon Athena Microsoft SQL Server

Konektor Amazon Athena untuk Microsoft SQL Server memungkinkan Amazon Athena menjalankan kueri SQL pada data Anda yang disimpan di Microsoft SQL Server menggunakan JDBC.

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Batasan

- Menulis operasi DDL tidak didukung.
- Dalam pengaturan multiplexer, bucket tumpahan dan awalan dibagikan di semua instance database.
- Batas Lambda yang relevan. Untuk informasi selengkapnya, lihat Kuota Lambda dalam Panduan Pengembang AWS Lambda .
- Dalam kondisi filter, Anda harus mentransmisikan tipe `Date` dan `Timestamp` data ke tipe data yang sesuai.
- Untuk mencari nilai negatif dari jenis `Real` dan`Float`, gunakan >= operator <= atau.
- Tipe `rowversion` data `binary` `varbinaryimage`,, dan tidak didukung.

Ketentuan

Istilah berikut berhubungan dengan konektor SQL Server.

- Instans database — Instance apa pun dari database yang digunakan di tempat, di Amazon EC2, atau di Amazon RDS.
- Handler - Handler Lambda yang mengakses instance database Anda. Handler bisa untuk metadata atau untuk catatan data.
- Metadata handler — Penangan Lambda yang mengambil metadata dari instance database Anda.
- Record handler - Handler Lambda yang mengambil catatan data dari instance database Anda.
- Composite handler — Handler Lambda yang mengambil data metadata dan data dari instance database Anda.
- Properti atau parameter - Properti database yang digunakan oleh penangan untuk mengekstrak informasi database. Anda mengonfigurasi properti ini sebagai variabel lingkungan Lambda.
- Connection String — Sebuah string teks yang digunakan untuk membuat koneksi ke instance database.

- Katalog —AWS Glue Non-katalog yang terdaftar di Athena yang merupakan awalan yang diperlukan untuk properti. `connection_string`
- Multiplexing handler - Handler Lambda yang dapat menerima dan menggunakan beberapa koneksi database.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengkonfigurasi konektor SQL Server.

String koneksi

Gunakan string koneksi JDBC dalam format berikut untuk terhubung ke instance database.

```
sqlserver://${jdbc_connection_string}
```

Menggunakan handler multiplexing

Anda dapat menggunakan multiplexer untuk terhubung ke beberapa instance database dengan satu fungsi Lambda. Permintaan dirutekan dengan nama katalog. Gunakan kelas berikut di Lambda.

Handler	Kelas
Pawang komposit	`SqlServerMuxCompositeHandler`
Penangan metadata	`SqlServerMuxMetadataHandler`
Rekam handler	`SqlServerMuxRecordHandler`

Parameter handler multiplexing

Parameter	Deskripsi
$*catalog*_connection_string	Wajib. Sebuah string koneksi instance database. Awalan variabel lingkungan dengan nama katalog yang digunakan di Athena. Misalnya, jika katalog terdaftar di Athena adalah`mysqlservercatalog` , maka nama variabel lingkungan adalah. `mysqlservercatalog_connection_string`

Parameter	Deskripsi
`default`	Wajib. String koneksi default. String ini digunakan ketika katalog `lambda:${ }AWS_LAMBDA_FUNCTION_NAME .`

Contoh properti berikut adalah untuk fungsi SqlServer MUX Lambda yang mendukung dua instance databases`sqlserver1`: (default), dan. `sqlserver2`

Properti	Nilai
`default`	`sqlserver://jdbc:sqlserver://sqlserver1. hostname:port;databaseName= <database_name> ;${secret1_name }`
`sqlserver_catalog1_connection_string`	`sqlserver://jdbc:sqlserver://sqlserver1. hostname:port;databaseName= <database_name> ;${secret1_name }`
`sqlserver_catalog2_connection_string`	`sqlserver://jdbc:sqlserver://sqlserver2. hostname:port;databaseName= <database_name> ;${secret2_name }`

Memberikan kredensi

Untuk memberikan nama pengguna dan kata sandi untuk database Anda dalam string koneksi JDBC Anda, Anda dapat menggunakan properti string koneksi atau. AWS Secrets Manager

- Connection String - Nama pengguna dan kata sandi dapat ditentukan sebagai properti dalam string koneksi JDBC.

> **Important**
>
> Sebagai praktik keamanan terbaik, jangan gunakan kredensyal hardcode dalam variabel lingkungan atau string koneksi Anda. Untuk informasi tentang memindahkan rahasia hardcode Anda AWS Secrets Manager, lihat Memindahkan rahasia hardcode ke AWS Secrets Manager dalam Panduan Pengguna.AWS Secrets Manager

- AWS Secrets Manager— Untuk menggunakan fitur Query Federasi Athena dengan, VPC AWS Secrets Manager yang terhubung ke fungsi Lambda Anda harus memiliki akses internet atau titik akhir VPC untuk terhubung ke Secrets Manager.

  Anda dapat memasukkan nama rahasia ke AWS Secrets Manager dalam string koneksi JDBC Anda. Konektor menggantikan nama rahasia dengan `password` nilai `username` dan dari Secrets Manager.

  Untuk instans database Amazon RDS, dukungan ini terintegrasi erat. Jika Anda menggunakan Amazon RDS, kami sangat menyarankan penggunaan AWS Secrets Manager dan rotasi kredensyal. Jika database Anda tidak menggunakan Amazon RDS, simpan kredensialnya sebagai JSON dalam format berikut:

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

Contoh string koneksi dengan nama rahasia

String berikut memiliki nama rahasia`${secret_name}`.

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

Konektor menggunakan nama rahasia untuk mengambil rahasia dan memberikan nama pengguna dan kata sandi, seperti pada contoh berikut.

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

Menggunakan handler koneksi tunggal

Anda dapat menggunakan metadata koneksi tunggal berikut dan penangan rekaman untuk terhubung ke satu instance SQL Server.

Jenis handler	Kelas
Pawang komposit	`SqlServerCompositeHandler`
Penangan metadata	`SqlServerMetadataHandler`

Jenis handler	Kelas
Rekam handler	`SqlServerRecordHandler`

Parameter handler koneksi tunggal

Parameter	Deskripsi
`default`	Wajib. String koneksi default.

Penangan koneksi tunggal mendukung satu instance database dan harus menyediakan parameter string `default` koneksi. Semua string koneksi lainnya diabaikan.

Properti contoh berikut adalah untuk contoh SQL Server tunggal didukung oleh fungsi Lambda.

Properti	Nilai
`default`	`sqlserver://jdbc:sqlserver://` *`hostname`*`:`*`port`*`;database Name=` *`<database_name>`* `;${`*`secret_name`*` }`

Parameter tumpahan

Lambda SDK dapat menumpahkan data ke Amazon S3. Semua instance database yang diakses oleh fungsi Lambda yang sama tumpah ke lokasi yang sama.

Parameter	Deskripsi
`spill_bucket`	Wajib. Nama ember tumpahan.
`spill_prefix`	Wajib. Tumpahkan key prefix bucket.
`spill_put_request_ headers`	(Opsional) Peta header permintaan dan nilai yang disandika n JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject {"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan

Parameter	Deskripsi
	header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.

Dukungan tipe data

Tabel berikut menunjukkan tipe data yang sesuai untuk SQL Server dan Apache Arrow.

SQL Server	Panah
bit	TINYINT
tinyint	SMALLINT
smallint	SMALLINT
int	INT
bigint	BIGINT
desimal	DECIMAL
numerik	FLOAT8
uang kecil	FLOAT8
money	DECIMAL
mengapung [24]	FLOAT4
mengapung [53]	FLOAT8
real	FLOAT4
datetime	Tanggal (MILLISECOND)
tanggal2	Tanggal (MILLISECOND)
smalldatetime	Tanggal (MILLISECOND)

SQL Server	Panah
date	Tanggal (HARI)
Waktu	VARCHAR
datetimeoffset	Tanggal (MILLISECOND)
arang [n]	VARCHAR
varchar [n/maks]	VARCHAR
nchar [n]	VARCHAR
nvarchar [n/max]	VARCHAR
text	VARCHAR
nteks	VARCHAR

Partisi dan split

Partisi diwakili oleh kolom partisi tunggal tipe`varchar`. Dalam kasus konektor SQL Server, fungsi partisi menentukan bagaimana partisi diterapkan di atas meja. Fungsi partisi dan informasi nama kolom diambil dari tabel metadata SQL Server. Kueri kustom kemudian mendapatkan partisi. Pemisahan dibuat berdasarkan jumlah partisi berbeda yang diterima.

Kinerja

Memilih subset kolom secara signifikan mempercepat runtime kueri dan mengurangi data yang dipindai. Konektor SQL Server tahan terhadap pelambatan karena konkurensi.

Konektor Athena SQL Server melakukan pushdown predikat untuk mengurangi data yang dipindai oleh kueri. Predikat sederhana dan ekspresi kompleks didorong ke konektor untuk mengurangi jumlah data yang dipindai dan mengurangi waktu eksekusi kueri.

Predikat

Predikat adalah ekspresi dalam `WHERE` klausa query SQL yang mengevaluasi nilai Boolean dan menyaring baris berdasarkan beberapa kondisi. Konektor Athena SQL Server dapat menggabungkan

ekspresi ini dan mendorongnya langsung ke SQL Server untuk fungsionalitas yang ditingkatkan dan untuk mengurangi jumlah data yang dipindai.

Operator konektor Athena SQL Server berikut mendukung pushdown predikat:

- Boolean: DAN, ATAU, TIDAK
- KESETARAAN: SAMA, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Aritmatika: TAMBAHKAN, KURANGI, KALIKAN, BAGI, MODULUS, MENIADAKAN
- Lainnya: LIKE_PATTERN, IN

Contoh pushdown gabungan

Untuk kemampuan kueri yang ditingkatkan, gabungkan jenis pushdown, seperti pada contoh berikut:

```
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%');
```

Kueri passthrough

Konektor SQL Server mendukung kueri passthrough. Kueri passthrough menggunakan fungsi tabel untuk mendorong kueri lengkap Anda ke sumber data untuk dieksekusi.

Untuk menggunakan kueri passthrough dengan SQL Server, Anda dapat menggunakan sintaks berikut:

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

Contoh query berikut mendorong ke bawah query ke sumber data di SQL Server. Kueri memilih semua kolom dalam `customer` tabel, membatasi hasilnya menjadi 10.

```
SELECT * FROM TABLE(
        system.query(
```

```
            query => 'SELECT * FROM customer LIMIT 10'
        ))
```

Informasi lisensi

Dengan menggunakan konektor ini, Anda mengakui penyertaan komponen pihak ketiga, daftar yang dapat ditemukan dalam file pom.xml untuk konektor ini, dan menyetujui persyaratan dalam masing-masing lisensi pihak ketiga yang disediakan dalam file LICENSE.txt di .com. GitHub

Sumber daya tambahan

Untuk informasi versi driver JDBC terbaru, lihat file pom.xml untuk konektor SQL Server di.com. GitHub

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Amazon Athena Teradata

Konektor Amazon Athena untuk Teradata memungkinkan Athena menjalankan kueri SQL pada data yang disimpan dalam database Teradata Anda.

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Batasan

- Menulis operasi DDL tidak didukung.
- Dalam pengaturan multiplexer, bucket tumpahan dan awalan dibagikan di semua instance database.
- Batas Lambda yang relevan. Untuk informasi selengkapnya, lihat Kuota Lambda dalam Panduan Pengembang AWS Lambda .

Ketentuan

Istilah-istilah berikut berhubungan dengan konektor Teradata.

- Instans database — Instance apa pun dari database yang digunakan di tempat, di Amazon EC2, atau di Amazon RDS.
- Handler - Handler Lambda yang mengakses instance database Anda. Handler bisa untuk metadata atau untuk catatan data.
- Metadata handler — Penangan Lambda yang mengambil metadata dari instance database Anda.
- Record handler - Handler Lambda yang mengambil catatan data dari instance database Anda.
- Composite handler — Handler Lambda yang mengambil data metadata dan data dari instance database Anda.
- Properti atau parameter - Properti database yang digunakan oleh penangan untuk mengekstrak informasi database. Anda mengonfigurasi properti ini sebagai variabel lingkungan Lambda.
- Connection String — Sebuah string teks yang digunakan untuk membuat koneksi ke instance database.
- Katalog —AWS Glue Non-katalog yang terdaftar di Athena yang merupakan awalan yang diperlukan untuk properti. `connection_string`
- Multiplexing handler - Handler Lambda yang dapat menerima dan menggunakan beberapa koneksi database.

Prasyarat lapisan Lambda

Untuk menggunakan konektor Teradata dengan Athena, Anda harus membuat layer Lambda yang menyertakan driver Teradata JDBC. Lapisan Lambda adalah arsip `.zip` file yang berisi kode tambahan untuk fungsi Lambda. Saat Anda menerapkan konektor Teradata ke akun Anda, Anda menentukan ARN layer. Ini menempelkan lapisan Lambda dengan driver Teradata JDBC ke konektor Teradata sehingga Anda dapat menggunakannya dengan Athena.

Untuk informasi selengkapnya tentang layer Lambda, lihat Membuat dan berbagi layer Lambda di Panduan Pengembang.AWS Lambda

Untuk membuat layer Lambda untuk konektor teradata

1. Jelajahi halaman unduhan driver Teradata JDBC di https://downloads.teradata.com/download/connectivity/jdbc-driver.
2. Unduh driver Teradata JDBC. Situs web mengharuskan Anda untuk membuat akun dan menerima perjanjian lisensi untuk mengunduh file.
3. Ekstrak `terajdbc4.jar` file dari file arsip yang Anda unduh.

4. Buat struktur folder berikut dan tempatkan `.jar` file di dalamnya.

   ```
   java\lib\terajdbc4.jar
   ```

5. Buat `.zip` file dari seluruh struktur folder yang berisi `terajdbc4.jar` file.
6. Masuk ke AWS Management Console dan buka AWS Lambda konsol di https://console.aws.amazon.com/lambda/.
7. Di panel navigasi, pilih Layers, lalu pilih Create layer.
8. Untuk Nama, masukkan nama untuk layer (misalnya,`TeradataJava11LambdaLayer`).
9. Pastikan opsi Unggah file.zip dipilih.
10. Pilih Unggah, lalu unggah folder zip yang berisi driver Teradata JDBC.
11. Pilih Buat.
12. Pada halaman detail untuk layer, salin layer ARN dengan memilih ikon clipboard di bagian atas halaman.
13. Simpan ARN untuk referensi.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengonfigurasi konektor Teradata.

String koneksi

Gunakan string koneksi JDBC dalam format berikut untuk terhubung ke instance database.

```
teradata://${jdbc_connection_string}
```

Menggunakan handler multiplexing

Anda dapat menggunakan multiplexer untuk terhubung ke beberapa instance database dengan satu fungsi Lambda. Permintaan dirutekan dengan nama katalog. Gunakan kelas berikut di Lambda.

Handler	Kelas
Handler komposit	`TeradataMuxCompositeHandler`
Penangan metadata	`TeradataMuxMetadataHandler`

Handler	Kelas
Rekam handler	`TeradataMuxRecordHandler`

Parameter handler multiplexing

Parameter	Deskripsi
`$`*`catalog`*`_connection_string`	Wajib. Sebuah string koneksi instance database. Awalan variabel lingkungan dengan nama katalog yang digunakan di Athena. Misalnya, jika katalog terdaftar di Athena adalah`myteradatacatalog` , maka nama variabel lingkungan adalah. `myteradatacatalog_connection_string`
`default`	Wajib. String koneksi default. String ini digunakan ketika katalog `lambda:${`*`}AWS_LAMBDA_FUNCTION_NAME`* .

Contoh properti berikut adalah untuk fungsi Teradata MUX Lambda yang mendukung dua instance database`teradata1`: (default), dan. `teradata2`

Properti	Nilai
`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`

Memberikan kredensi

Untuk memberikan nama pengguna dan kata sandi untuk database Anda dalam string koneksi JDBC Anda, Anda dapat menggunakan properti string koneksi atau. AWS Secrets Manager

- Connection String - Nama pengguna dan kata sandi dapat ditentukan sebagai properti dalam string koneksi JDBC.

  > **Important**
  >
  > Sebagai praktik keamanan terbaik, jangan gunakan kredensyal hardcode dalam variabel lingkungan atau string koneksi Anda. Untuk informasi tentang memindahkan rahasia hardcode Anda AWS Secrets Manager, lihat Memindahkan rahasia hardcode ke AWS Secrets Manager dalam Panduan Pengguna.AWS Secrets Manager

- AWS Secrets Manager— Untuk menggunakan fitur Query Federasi Athena dengan, VPC AWS Secrets Manager yang terhubung ke fungsi Lambda Anda harus memiliki akses internet atau titik akhir VPC untuk terhubung ke Secrets Manager.

  Anda dapat memasukkan nama rahasia ke AWS Secrets Manager dalam string koneksi JDBC Anda. Konektor menggantikan nama rahasia dengan `password` nilai `username` dan dari Secrets Manager.

  Untuk instans database Amazon RDS, dukungan ini terintegrasi erat. Jika Anda menggunakan Amazon RDS, kami sangat menyarankan penggunaan AWS Secrets Manager dan rotasi kredensyal. Jika database Anda tidak menggunakan Amazon RDS, simpan kredensialnya sebagai JSON dalam format berikut:

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

Contoh string koneksi dengan nama rahasia

String berikut memiliki nama rahasia`${Test/RDS/Teradata1}`.

```
teradata://jdbc:teradata1.host/TMODE=ANSI,CHARSET=UTF8,DATABASE=TEST,${Test/RDS/
Teradata1}&...
```

Konektor menggunakan nama rahasia untuk mengambil rahasia dan memberikan nama pengguna dan kata sandi, seperti pada contoh berikut.

```
teradata://jdbc:teradata://teradata1.host/
TMODE=ANSI,CHARSET=UTF8,DATABASE=TEST,...&user=sample2&password=sample2&...
```

Saat ini, Teradata mengenali properti `user` dan `password` JDBC. Ini juga menerima nama pengguna dan kata sandi dalam kata *`/sandi format nama pengguna`* tanpa kunci `user` atau`password`.

Menggunakan handler koneksi tunggal

Anda dapat menggunakan metadata koneksi tunggal berikut dan penangan rekaman untuk terhubung ke satu instance Teradata.

Jenis handler	Kelas
Handler komposit	`TeradataCompositeHandler`
Penangan metadata	`TeradataMetadataHandler`
Rekam handler	`TeradataRecordHandler`

Parameter handler koneksi tunggal

Parameter	Deskripsi
`default`	Wajib. String koneksi default.

Penangan koneksi tunggal mendukung satu instance database dan harus menyediakan parameter string `default` koneksi. Semua string koneksi lainnya diabaikan.

Properti contoh berikut adalah untuk instance Teradata tunggal yang didukung oleh fungsi Lambda.

Properti	Nilai
`default`	`teradata://jdbc:teradata://teradata1.host/TMODE=ANSI,CHARSET=UTF8,DATABASE=TEST,secret=Test/RDS/Teradata1`

Parameter tumpahan

Lambda SDK dapat menumpahkan data ke Amazon S3. Semua instance database yang diakses oleh fungsi Lambda yang sama tumpah ke lokasi yang sama.

Parameter	Deskripsi
`spill_bucket`	Wajib. Nama ember tumpahan.
`spill_prefix`	Wajib. Tumpahkan key prefix bucket.
`spill_put_request_headers`	(Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject {"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.

Dukungan tipe data

Tabel berikut menunjukkan tipe data yang sesuai untuk JDBC dan Apache Arrow.

JDBC	Panah
Boolean	Bit
Bilangan Bulat	Mungil
Pendek	berkulit kecil
Bilangan Bulat	Int
Long	Bigint
float	Mengapung4
Ganda	Mengapung8
Tanggal	DateDay

JDBC	Panah
Stempel Waktu	DateMilli
String	Varchar
Byte	Varbiner
BigDecimal	Decimal
ARRAY	Daftar

Partisi dan split

Partisi diwakili oleh kolom partisi tunggal tipe`Integer`. Kolom berisi nama partisi dari partisi yang didefinisikan pada tabel Teradata. Untuk tabel yang tidak memiliki nama partisi, * dikembalikan, yang setara dengan satu partisi. Partisi setara dengan split.

Nama	Tipe	Deskripsi
partisi	Bilangan Bulat	Dinamakan partisi di Teradata.

Kinerja

Teradata mendukung partisi asli. Konektor Athena Teradata dapat mengambil data dari partisi ini secara paralel. Jika Anda ingin menanyakan kumpulan data yang sangat besar dengan distribusi partisi yang seragam, partisi asli sangat disarankan. Memilih subset kolom secara signifikan memperlambat runtime kueri. Konektor menunjukkan beberapa pelambatan karena konkurensi.

Konektor Athena Teradata melakukan pushdown predikat untuk mengurangi data yang dipindai oleh kueri. Predikat sederhana dan ekspresi kompleks didorong ke konektor untuk mengurangi jumlah data yang dipindai dan mengurangi waktu eksekusi kueri.

Predikat

Predikat adalah ekspresi dalam `WHERE` klausa kueri SQL yang mengevaluasi nilai Boolean dan menyaring baris berdasarkan beberapa kondisi. Konektor Athena Teradata dapat menggabungkan ekspresi ini dan mendorongnya langsung ke Teradata untuk meningkatkan fungsionalitas dan untuk mengurangi jumlah data yang dipindai.

Operator konektor Athena Teradata berikut mendukung pushdown predikat:

- Boolean: DAN, ATAU, TIDAK
- KESETARAAN: SAMA, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, NULL_IF, IS_NULL
- Aritmatika: TAMBAHKAN, KURANGI, KALIKAN, BAGI, MODULUS, MENIADAKAN
- Lainnya: LIKE_PATTERN, IN

Contoh pushdown gabungan

Untuk kemampuan kueri yang ditingkatkan, gabungkan jenis pushdown, seperti pada contoh berikut:

```
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%');
```

Kueri passthrough

Konektor Teradata mendukung kueri passthrough. Kueri passthrough menggunakan fungsi tabel untuk mendorong kueri lengkap Anda ke sumber data untuk dieksekusi.

Untuk menggunakan kueri passthrough dengan Teradata, Anda dapat menggunakan sintaks berikut:

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

Contoh query berikut mendorong ke bawah query ke sumber data di Teradata. Kueri memilih semua kolom dalam `customer` tabel, membatasi hasilnya menjadi 10.

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

Informasi lisensi

Dengan menggunakan konektor ini, Anda mengakui penyertaan komponen pihak ketiga, daftar yang dapat ditemukan dalam file pom.xml untuk konektor ini, dan menyetujui persyaratan dalam masing-masing lisensi pihak ketiga yang disediakan dalam file LICENSE.txt di .com. GitHub

Sumber daya tambahan

Untuk informasi versi driver JDBC terbaru, lihat file pom.xml untuk konektor Teradata di.com. GitHub

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Timestream Amazon Athena

Konektor Timestream Amazon Athena memungkinkan Amazon Athena berkomunikasi dengan Amazon Timestream, membuat data deret waktu Anda dapat diakses melalui Amazon Athena. Anda dapat secara opsional menggunakan AWS Glue Data Catalog sebagai sumber metadata tambahan.

Amazon Timestream adalah cepat, scalable, sepenuhnya dikelola, tujuan dibangun basis data seri waktu yang membuatnya mudah untuk menyimpan dan menganalisis triliunan titik data time series per hari. Timestream menghemat waktu dan biaya Anda dalam mengelola siklus hidup data seri waktu dengan menyimpan data terbaru dalam memori dan memindahkan data historis ke level penyimpanan yang dioptimalkan biaya berdasarkan kebijakan yang ditetapkan pengguna.

Jika Anda mengaktifkan Lake Formation di akun Anda, peran IAM untuk konektor Lambda federasi Athena yang Anda gunakan harus memiliki akses baca di AWS Serverless Application Repository Lake Formation ke. AWS Glue Data Catalog

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengkonfigurasi konektor Timestream.

- spill_bucket - Menentukan bucket Amazon S3 untuk data yang melebihi batas fungsi Lambda.

- spill_prefix — (Opsional) Default ke subfolder dalam nama yang ditentukan. `spill_bucket athena-federation-spill` Kami menyarankan Anda mengonfigurasi siklus hidup penyimpanan Amazon S3 di lokasi ini untuk menghapus tumpahan yang lebih lama dari jumlah hari atau jam yang telah ditentukan sebelumnya.
- spill_put_request_headers — (Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject {"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.
- kms_key_id — (Opsional) Secara default, data apa pun yang tumpah ke Amazon S3 dienkripsi menggunakan mode enkripsi yang diautentikasi AES-GCM dan kunci yang dihasilkan secara acak. Agar fungsi Lambda Anda menggunakan kunci enkripsi yang lebih kuat yang dihasilkan oleh KMS sepertia`7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, Anda dapat menentukan ID kunci KMS.
- disable_spill_encryption — (Opsional) Ketika diatur ke, menonaktifkan enkripsi tumpahan. `True` Defaultnya `False` sehingga data yang tumpah ke S3 dienkripsi menggunakan AES-GCM — baik menggunakan kunci yang dihasilkan secara acak atau KMS untuk menghasilkan kunci. Menonaktifkan enkripsi tumpahan dapat meningkatkan kinerja, terutama jika lokasi tumpahan Anda menggunakan enkripsi sisi server.
- glue_catalog - (Opsional) Gunakan opsi ini untuk menentukan katalog lintas akun. AWS Glue Secara default, konektor mencoba untuk mendapatkan metadata dari akunnya sendiri AWS Glue .

Menyiapkan database dan tabel di AWS Glue

Anda dapat secara opsional menggunakan AWS Glue Data Catalog sebagai sumber metadata tambahan. Untuk mengaktifkan AWS Glue tabel untuk digunakan dengan Timestream, Anda harus memiliki AWS Glue database dan tabel dengan nama yang cocok dengan database Timestream dan tabel yang ingin Anda berikan metadata tambahan.

> **Note**
>
> Untuk kinerja terbaik, gunakan hanya huruf kecil untuk nama database dan nama tabel Anda. Menggunakan casing campuran menyebabkan konektor melakukan pencarian case insensitive yang lebih intensif secara komputasi.

Untuk mengkonfigurasi AWS Glue tabel untuk digunakan dengan Timestream, Anda harus mengatur properti tabelnya. AWS Glue

Untuk menggunakan AWS Glue tabel untuk metadata tambahan

1. Edit tabel di AWS Glue konsol untuk menambahkan properti tabel berikut:

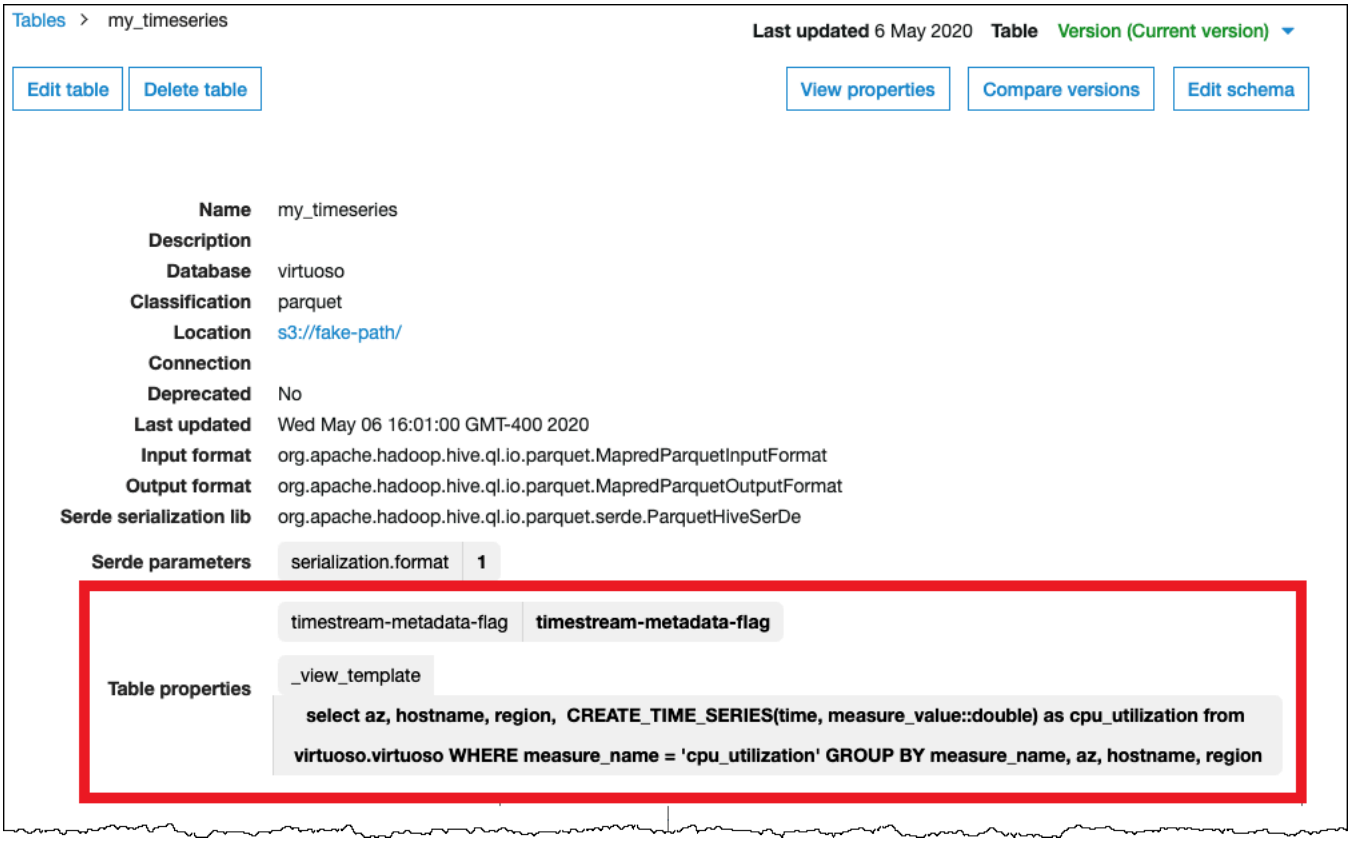
   - timestream-metadata-flag- Properti ini menunjukkan ke konektor Timestream bahwa konektor dapat menggunakan tabel untuk metadata tambahan. Anda dapat memberikan nilai apa pun `timestream-metadata-flag` selama `timestream-metadata-flag` properti hadir dalam daftar properti tabel.
   - _view_template - Bila Anda menggunakan AWS Glue metadata tambahan, Anda dapat menggunakan properti tabel ini dan menentukan SQL Timestream sebagai tampilan. Konektor Athena Timestream menggunakan SQL dari tampilan bersama dengan SQL Anda dari Athena untuk menjalankan kueri Anda. Ini berguna jika Anda ingin menggunakan fitur Timestream SQL yang tidak tersedia di Athena.
2. Pastikan Anda menggunakan tipe data yang sesuai AWS Glue seperti yang tercantum dalam dokumen ini.

Jenis data

Saat ini, konektor Timestream hanya mendukung sebagian dari tipe data yang tersedia di Timestream, khususnya: nilai `varchar` skalar,, dan. `double` `timestamp`

Untuk menanyakan tipe `timeseries` data, Anda harus mengonfigurasi tampilan dalam properti AWS Glue tabel yang menggunakan `CREATE_TIME_SERIES` fungsi Timestream. Anda juga perlu menyediakan skema untuk tampilan yang menggunakan sintaks `ARRAY<STRUCT<time:timestamp,measure_value::double:double>>` sebagai tipe untuk kolom deret waktu Anda. Pastikan untuk mengganti `double` dengan jenis skalar yang sesuai untuk tabel Anda.

Gambar berikut menunjukkan contoh properti AWS Glue tabel dikonfigurasi untuk mengatur tampilan selama deret waktu.



Izin yang Diperlukan

Untuk detail lengkap tentang kebijakan IAM yang diperlukan konektor ini, tinjau `Policies` bagian file athena-timestream.yaml. Daftar berikut merangkum izin yang diperlukan.

- Akses tulis Amazon S3 - Konektor memerlukan akses tulis ke lokasi di Amazon S3 untuk menumpahkan hasil dari kueri besar.
- Athena GetQueryExecution — Konektor menggunakan izin ini untuk gagal cepat ketika kueri Athena hulu telah dihentikan.
- AWS Glue Data Catalog- Konektor Timestream membutuhkan akses baca saja ke AWS Glue Data Catalog untuk mendapatkan informasi skema.
- CloudWatch Log — Konektor memerlukan akses ke CloudWatch Log untuk menyimpan log.
- Akses Timestream — Untuk menjalankan kueri Timestream.

Kinerja

Kami menyarankan Anda menggunakan `LIMIT` klausa untuk membatasi data yang dikembalikan (bukan data yang dipindai) hingga kurang dari 256 MB untuk memastikan bahwa kueri interaktif berkinerja baik.

Konektor Athena Timestream melakukan pushdown predikat untuk mengurangi data yang dipindai oleh kueri. `LIMIT`klausa mengurangi jumlah data yang dipindai, tetapi jika Anda tidak memberikan predikat, Anda harus mengharapkan `SELECT` kueri dengan `LIMIT` klausa untuk memindai setidaknya 16 MB data. Memilih subset kolom secara signifikan mempercepat runtime kueri dan mengurangi data yang dipindai. Konektor Timestream tahan terhadap throttling karena konkurensi.

Kueri passthrough

Konektor Timestream mendukung kueri passthrough. Kueri passthrough menggunakan fungsi tabel untuk mendorong kueri lengkap Anda ke sumber data untuk dieksekusi.

Untuk menggunakan kueri passthrough dengan Timestream, Anda dapat menggunakan sintaks berikut:

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

Contoh query berikut mendorong ke bawah query ke sumber data di Timestream. Kueri memilih semua kolom dalam `customer` tabel, membatasi hasilnya menjadi 10.

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

Informasi lisensi

Proyek konektor Timestream Amazon Athena dilisensikan di bawah Lisensi Apache-2.0.

Sumber daya tambahan

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor DS (TPC-DS) patokan Amazon Athena TPC

Konektor Amazon Athena TPC-DS memungkinkan Amazon Athena untuk berkomunikasi dengan sumber data TPC Benchmark DS yang dihasilkan secara acak untuk digunakan dalam pembandingan dan pengujian fungsional Federasi Athena. Konektor Athena TPC-DS menghasilkan basis data yang sesuai dengan TPC-DS pada salah satu dari empat faktor skala. Kami tidak merekomendasikan penggunaan konektor ini sebagai alternatif untuk tes kinerja data lake berbasis Amazon S3.

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.

Parameter

Gunakan variabel lingkungan Lambda di bagian ini untuk mengkonfigurasi konektor TPC-DS.

- spill_bucket - Menentukan bucket Amazon S3 untuk data yang melebihi batas fungsi Lambda.
- spill_prefix — (Opsional) Default ke subfolder dalam nama yang ditentukan. `spill_bucket` `athena-federation-spill` Kami menyarankan Anda mengonfigurasi siklus hidup penyimpanan Amazon S3 di lokasi ini untuk menghapus tumpahan yang lebih lama dari jumlah hari atau jam yang telah ditentukan sebelumnya.
- spill_put_request_headers — (Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject` `{"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.
- kms_key_id — (Opsional) Secara default, data apa pun yang tumpah ke Amazon S3 dienkripsi menggunakan mode enkripsi yang diautentikasi AES-GCM dan kunci yang dihasilkan secara acak. Agar fungsi Lambda Anda menggunakan kunci enkripsi yang lebih kuat yang dihasilkan oleh KMS sepertia`7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, Anda dapat menentukan ID kunci KMS.
- disable_spill_encryption — (Opsional) Ketika diatur ke, menonaktifkan enkripsi tumpahan. `True` Defaultnya `False` sehingga data yang tumpah ke S3 dienkripsi menggunakan AES-GCM — baik menggunakan kunci yang dihasilkan secara acak atau KMS untuk menghasilkan kunci.

Menonaktifkan enkripsi tumpahan dapat meningkatkan kinerja, terutama jika lokasi tumpahan Anda menggunakan enkripsi sisi server.

Uji database dan tabel

Konektor Athena TPC-DS menghasilkan database yang sesuai dengan TPC-DS di salah satu dari empat faktor skala,,,, atau. `tpcds1` `tpcds10` `tpcds100` `tpcds250` `tpcds1000`

Ringkasan tabel

Untuk daftar lengkap tabel dan kolom data pengujian, jalankan `SHOW TABLES` atau `DESCRIBE TABLE` kueri. Ringkasan tabel berikut disediakan untuk kenyamanan.

1. call_center
2. catalog_page
3. catalog_returns
4. catalog_sales
5. pelanggan
6. customer_address
7. customer_demografi
8. date_dim
9. dbgen_version
10. demografi rumah tangga
11. pendapatan_band
12. inventaris
13. item
14. promosi
15. akal budi
16. ship_mode
17. Toko
18. store_returns
19. store_sales

20.waktu_redup

21.gudang

22.web_halaman

23.web_returns

24.web_penjualan

25.situs web

Untuk kueri TPC-DS yang kompatibel dengan skema dan data yang dihasilkan ini, lihat direktori athena-tpcds/src/main/resources/queries/ di. GitHub

Kueri contoh

Contoh `SELECT` kueri berikut menanyakan `tpcds` katalog untuk demografi pelanggan di kabupaten tertentu.

```
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
```

Izin yang Diperlukan

Untuk detail lengkap tentang kebijakan IAM yang diperlukan konektor ini, tinjau `Policies` bagian file athena-tpcds.yaml. Daftar berikut merangkum izin yang diperlukan.

- Akses tulis Amazon S3 - Konektor memerlukan akses tulis ke lokasi di Amazon S3 untuk menumpahkan hasil dari kueri besar.
- Athena GetQueryExecution — Konektor menggunakan izin ini untuk gagal cepat ketika kueri Athena hulu telah dihentikan.

Kinerja

Konektor Athena TPC-DS mencoba memparalelkan kueri berdasarkan faktor skala yang Anda pilih. Predikat pushdown dilakukan dalam fungsi Lambda.

Informasi lisensi

Proyek konektor Amazon Athena TPC-DS dilisensikan di bawah Lisensi Apache-2.0.

Sumber daya tambahan

Untuk informasi tambahan tentang konektor ini, kunjungi situs terkait GitHub di.com.

Konektor Amazon Athena Vertica

Vertica adalah platform basis data kolumnar yang dapat digunakan di cloud atau di tempat yang mendukung gudang data skala exabyte. Anda dapat menggunakan konektor Amazon Athena Vertica dalam kueri gabungan untuk mengkueri sumber data Vertica dari Athena. Misalnya, Anda dapat menjalankan kueri analitis atas data warehouse di Vertica dan danau data di Amazon S3.

Prasyarat

- Menyebarkan konektor ke Anda Akun AWS menggunakan konsol Athena atau. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menyebarkan konektor sumber data atau MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data.
- Siapkan VPC dan grup keamanan sebelum Anda menggunakan konektor ini. Untuk informasi selengkapnya, lihat Membuat VPC untuk konektor sumber data.

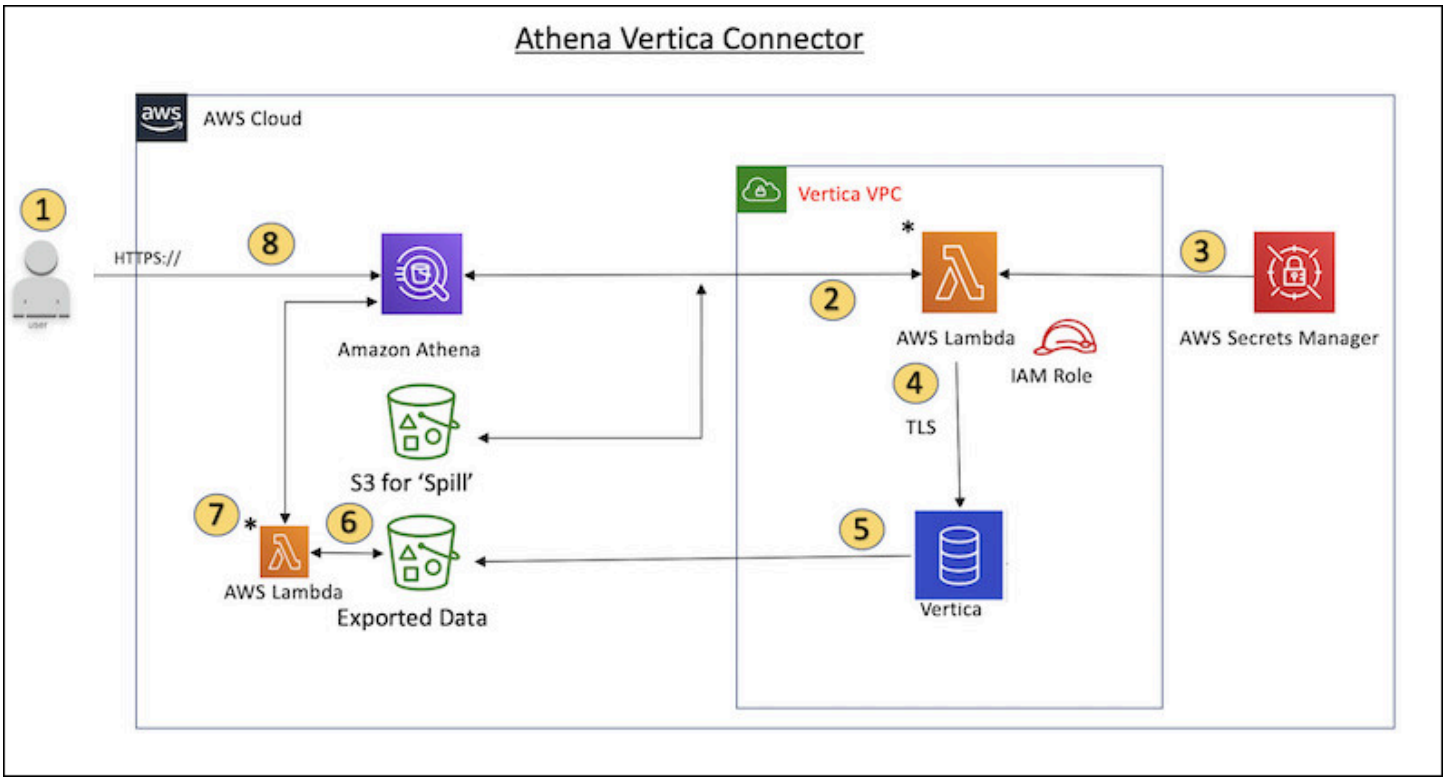
Batasan

- Karena konektor Athena Vertica menggunakan Amazon S3 Select untuk membaca file Parket dari Amazon S3, kinerja konektor bisa lambat. Saat Anda menanyakan tabel besar, kami sarankan Anda menggunakan CREATE TABLE AS (SELECT...) query dan predikat SQL.
- Saat ini, karena masalah yang diketahui di Kueri Federasi Athena, konektor menyebabkan Vertica mengekspor semua kolom tabel kueri ke Amazon S3, tetapi hanya kolom yang ditanyakan yang terlihat di hasil di konsol Athena.
- Menulis operasi DDL tidak didukung.

- Batas Lambda yang relevan. Untuk informasi selengkapnya, lihat Kuota Lambda dalam Panduan Pengembang AWS Lambda .

Alur kerja

Diagram berikut menunjukkan alur kerja kueri yang menggunakan konektor Vertica.



1. Kueri SQL dikeluarkan terhadap satu atau lebih tabel di Vertica.
2. Konektor mem-parsing query SQL untuk mengirim bagian yang relevan ke Vertica melalui koneksi JDBC.
3. String koneksi menggunakan nama pengguna dan kata sandi yang disimpan AWS Secrets Manager untuk mendapatkan akses ke Vertica.
4. Konektor membungkus query SQL dengan EXPORT perintah Vertica, seperti pada contoh berikut.

```
EXPORT TO PARQUET (directory = 's3://DOC-EXAMPLE-BUCKET/folder_name,
   Compression='Snappy', fileSizeMB=64) OVER() as
SELECT
PATH_ID,
...
SOURCE_ITEMIZED,
SOURCE_OVERRIDE
```

```
FROM DELETED_OBJECT_SCHEMA.FORM_USAGE_DATA
WHERE PATH_ID <= 5;
```

5. Vertica memproses kueri SQL dan mengirimkan hasil yang disetel ke bucket Amazon S3. Untuk throughput yang lebih baik, Vertica menggunakan `EXPORT` opsi untuk memparalelkan operasi penulisan beberapa file Parket.
6. Athena memindai bucket Amazon S3 untuk menentukan jumlah file yang akan dibaca untuk set hasil.
7. Athena membuat beberapa panggilan ke fungsi Lambda dan menggunakan Amazon S3 Select untuk membaca file Parket dari set hasil. Beberapa panggilan memungkinkan Athena untuk memparalelkan pembacaan file Amazon S3 dan mencapai throughput hingga 100GB per detik.
8. Athena memproses data yang dikembalikan dari Vertica dengan data yang dipindai dari danau data dan mengembalikan hasilnya.

Ketentuan

Istilah-istilah berikut berhubungan dengan konektor Vertica.

- Instans database — Setiap instance dari database Vertica yang digunakan di Amazon EC2.
- Handler - Handler Lambda yang mengakses instance database Anda. Handler bisa untuk metadata atau untuk catatan data.
- Metadata handler — Penangan Lambda yang mengambil metadata dari instance database Anda.
- Record handler - Handler Lambda yang mengambil catatan data dari instance database Anda.
- Composite handler — Handler Lambda yang mengambil data metadata dan data dari instance database Anda.
- Properti atau parameter - Properti database yang digunakan oleh penangan untuk mengekstrak informasi database. Anda mengonfigurasi properti ini sebagai variabel lingkungan Lambda.
- Connection String — Sebuah string teks yang digunakan untuk membuat koneksi ke instance database.
- Katalog —AWS Glue Non-katalog yang terdaftar di Athena yang merupakan awalan yang diperlukan untuk properti. `connection_string`

Parameter

Konektor Amazon Athena Vertica memperlihatkan opsi konfigurasi melalui variabel lingkungan Lambda. Anda dapat menggunakan variabel lingkungan Lambda berikut untuk mengkonfigurasi konektor.

- AthenaCatalogName— Nama fungsi Lambda
- ExportBucket- Bucket Amazon S3 tempat hasil kueri Vertica diekspor.
- SpillBucket— Nama bucket Amazon S3 tempat fungsi ini dapat menumpahkan data.
- SpillPrefix— Awalan untuk `SpillBucket` lokasi di mana fungsi ini dapat menumpahkan data.
- SecurityGroupIds— Satu atau lebih ID yang sesuai dengan grup keamanan yang harus diterapkan ke fungsi Lambda (misalnya,, `sg1` `sg2`, atau `sg3`).
- SubnetIdsSatu atau lebih ID subnet yang sesuai dengan subnet yang dapat digunakan fungsi Lambda untuk mengakses sumber data Anda (misalnya,, `subnet1` atau). `subnet2`
- SecretNameOrPrefix— Nama atau awalan dari satu set nama di Secrets Manager yang fungsi ini memiliki akses ke (misalnya,`vertica-*`)
- VerticaConnectionString— Detail koneksi Vertica untuk digunakan secara default jika tidak ada koneksi khusus katalog yang ditentukan. String secara opsional dapat menggunakan AWS Secrets Manager sintaks (misalnya,`${secret_name}`).
- ID VPC — ID VPC yang akan dilampirkan ke fungsi Lambda.

String koneksi

Gunakan string koneksi JDBC dalam format berikut untuk terhubung ke instance database.

```
vertica://jdbc:vertica://host_name:port/database?user=vertica-
username&password=vertica-password
```

Menggunakan handler koneksi tunggal

Anda dapat menggunakan metadata koneksi tunggal berikut dan penangan rekaman untuk terhubung ke satu instance Vertica.

Jenis handler	Kelas
Handler komposit	`VerticaCompositeHandler`

Jenis handler	Kelas
Penangan metadata	`VerticaMetadataHandler`
Rekam handler	`VerticaRecordHandler`

Parameter handler koneksi tunggal

Parameter	Deskripsi
`default`	Wajib. String koneksi default.

Penangan koneksi tunggal mendukung satu instance database dan harus menyediakan parameter string `default` koneksi. Semua string koneksi lainnya diabaikan.

Memberikan kredensi

Untuk memberikan nama pengguna dan kata sandi untuk database Anda dalam string koneksi JDBC Anda, Anda dapat menggunakan properti string koneksi atau. AWS Secrets Manager

- Connection String - Nama pengguna dan kata sandi dapat ditentukan sebagai properti dalam string koneksi JDBC.

  > **Important**
  >
  > Sebagai praktik keamanan terbaik, jangan gunakan kredensi hardcode dalam variabel lingkungan atau string koneksi Anda. Untuk informasi tentang memindahkan rahasia hardcode Anda AWS Secrets Manager, lihat Memindahkan rahasia hardcode ke AWS Secrets Manager dalam Panduan Pengguna.AWS Secrets Manager

- AWS Secrets Manager— Untuk menggunakan fitur Query Federasi Athena dengan, VPC AWS Secrets Manager yang terhubung ke fungsi Lambda Anda harus memiliki akses internet atau titik akhir VPC untuk terhubung ke Secrets Manager.

  Anda dapat memasukkan nama rahasia ke AWS Secrets Manager dalam string koneksi JDBC Anda. Konektor menggantikan nama rahasia dengan `password` nilai `username` dan dari Secrets Manager.

Untuk instans database Amazon RDS, dukungan ini terintegrasi dengan erat. Jika Anda menggunakan Amazon RDS, kami sangat menyarankan penggunaan AWS Secrets Manager dan rotasi kredenal. Jika database Anda tidak menggunakan Amazon RDS, simpan kredensialnya sebagai JSON dalam format berikut:

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

Contoh string koneksi dengan nama rahasia

String berikut memiliki nama rahasia ${vertica-username} dan${vertica-password}.

```
vertica://jdbc:vertica://host_name:port/database?user=${vertica-username}&password=
${vertica-password}
```

Konektor menggunakan nama rahasia untuk mengambil rahasia dan memberikan nama pengguna dan kata sandi, seperti pada contoh berikut.

```
vertica://jdbc:vertica://host_name:port/database?user=sample-user&password=sample-
password
```

Saat ini, konektor Vertica mengenali properti `vertica-username` dan `vertica-password` JDBC.

Parameter tumpahan

Lambda SDK dapat menumpahkan data ke Amazon S3. Semua instance database yang diakses oleh fungsi Lambda yang sama tumpah ke lokasi yang sama.

Parameter	Deskripsi
`spill_bucket`	Wajib. Nama ember tumpahan.
`spill_prefix`	Wajib. Tumpahkan key prefix bucket.
`spill_put_request_headers`	(Opsional) Peta header permintaan dan nilai yang disandikan JSON untuk permintaan Amazon S3 yang digunakan untuk menumpahkan (misalnya,). `putObject` `{"x-amz-server-side-encryption" : "AES256"}` Untuk kemungkinan

Parameter	Deskripsi
	header lainnya, lihat PutObjectdi Referensi API Amazon Simple Storage Service.

Dukungan tipe data

Tabel berikut menunjukkan tipe data yang didukung untuk konektor Vertica.

Boolean
BigInt
Pendek
Bilangan Bulat
Long
Desimal
Ganda
Tanggal
Varchar
Byte
BigDecimal
TimeStamp sebagai Varchar

Kinerja

Fungsi Lambda melakukan pushdown proyeksi untuk mengurangi data yang dipindai oleh kueri. `LIMIT`klausa mengurangi jumlah data yang dipindai, tetapi jika Anda tidak memberikan predikat, Anda harus mengharapkan `SELECT` kueri dengan `LIMIT` klausa untuk memindai setidaknya 16 MB data. Konektor Vertica tahan terhadap pelambatan karena konkurensi.

Kueri passthrough

Konektor Vertica mendukung kueri passthrough. Kueri passthrough menggunakan fungsi tabel untuk mendorong kueri lengkap Anda ke sumber data untuk dieksekusi.

Untuk menggunakan kueri passthrough dengan Vertica, Anda dapat menggunakan sintaks berikut:

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

Contoh query berikut mendorong ke bawah query ke sumber data di Vertica. Kueri memilih semua kolom dalam `customer` tabel, membatasi hasilnya menjadi 10.

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

Informasi lisensi

Dengan menggunakan konektor ini, Anda mengakui penyertaan komponen pihak ketiga, daftar yang dapat ditemukan dalam file pom.xml untuk konektor ini, dan menyetujui persyaratan dalam masing-masing lisensi pihak ketiga yang disediakan dalam file LICENSE.txt di .com. GitHub

Sumber daya tambahan

Untuk informasi versi driver JDBC terbaru, lihat file pom.xml untuk konektor Vertica di.com. GitHub

Untuk informasi tambahan tentang konektor ini, lihat situs terkait GitHub di.com dan Menanyakan sumber data Vertica di Amazon Athena menggunakan Athena Federated Query SDK di Blog Big Data.AWS

Menyebarkan konektor sumber data

Bersiap untuk membuat kueri gabungan adalah proses dua bagian: men-deploy konektor sumber data fungsi Lambda, dan menghubungkan fungsi Lambda ke sumber data. Dalam proses ini, Anda memberi nama fungsi Lambda yang nantinya dapat Anda pilih di konsol Athena dan memberi konektor nama yang dapat Anda referensikan dalam kueri SQL Anda.

> **Note**
>
> Untuk menggunakan fitur Kueri Federasi Athena AWS Secrets Manager, Anda harus mengonfigurasi titik akhir pribadi Amazon VPC untuk Secrets Manager. Untuk informasi selengkapnya, lihat Membuat titik akhir pribadi VPC Secrets Manager di Panduan Pengguna.AWS Secrets Manager

Topik

Menggunakan konsol Athena

Untuk memilih, nama, dan menggunakan penyambung sumber data, anda menggunakan konsol Athena dan Lambda dalam proses bersepadu.

Cara menggunakan konektor sumber data

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.

3. Di panel navigasi, pilih Sumber data.
4. Pada halaman Sumber data, pilih Buat sumber data.
5. Untuk Pilih sumber data, pilih sumber data yang ingin Athena kueri, dengan mempertimbangkan pedoman berikut:

   - Pilih opsi kueri gabungan yang sesuai dengan sumber data Anda. Athena memiliki konektor sumber data bawaan yang dapat Anda konfigurasikan untuk sumber termasuk MySQL, Amazon DocumentDB, dan PostgreSQL.
   - Pilih S3 - AWS Glue Data Catalog jika Anda ingin menanyakan data di Amazon S3 dan Anda tidak menggunakan metastore Apache Hive atau salah satu opsi sumber data kueri gabungan lainnya di halaman ini. Athena menggunakan AWS Glue Data Catalog untuk menyimpan metadata dan informasi skema untuk sumber data di Amazon S3. Ini adalah opsi default (non-federasi). Untuk informasi selengkapnya, lihat Menggunakan AWS Glue untuk terhubung ke sumber data di Amazon S3.
   - Pilih S3 - Apache Hive metastore untuk kueri set data di Amazon S3 yang menggunakan metastore Apache Hive. Untuk informasi selengkapnya tentang metrik ini, lihat Menghubungkan Athena ke metastore Apache Hive.
   - Pilih konektor Kustom atau bersama jika Anda ingin membuat konektor sumber data Anda sendiri untuk digunakan dengan Athena. Untuk informasi tentang menulis konektor sumber data, lihatMengembangkan konektor sumber data menggunakan Athena Query Federation SDK.

   Tutorial ini memilih Amazon CloudWatch Logs sebagai sumber data federasi.
6. Pilih Berikutnya.
7. Pada halaman Masukkan detail sumber data, untuk nama sumber data, masukkan nama yang ingin Anda gunakan dalam pernyataan SQL saat Anda menanyakan sumber data dari Athena (misalnya`CloudWatchLogs`,). Nama bisa sampai 127 karakter dan harus unik di akun Anda. Itu tidak dapat diubah setelah Anda membuatnya. Karakter yang valid adalah a-z, A-Z, 0-9, _ (garis bawah), @ (pada tanda) dan - (tanda hubung). Nama`awsdatacatalog`,, `hivejmx`, dan dicadangkan `system` oleh Athena dan tidak dapat digunakan untuk nama sumber data.
8. Untuk fungsi Lambda, pilih fungsi Buat Lambda. Halaman fungsi untuk konektor yang Anda pilih terbuka di AWS Lambda konsol. Halaman ini mencakup informasi rinci tentang konektor.
9. Di bawahPengaturan aplikasi, baca deskripsi untuk setiap pengaturan aplikasi dengan hati-hati, dan kemudian masukkan nilai yang sesuai dengan kebutuhan Anda.

Pengaturan aplikasi yang Anda lihat bervariasi tergantung pada konektor untuk sumber data Anda. Pengaturan minimum yang diperlukan meliputi:

- AthenaCatalogName— Nama, dalam huruf kecil, untuk fungsi Lambda yang menunjukkan sumber data yang ditargetkan, seperti. `cloudwatchlogs`
- SpillBucket- Bucket Amazon S3 di akun Anda untuk menyimpan data yang melebihi batas ukuran respons fungsi Lambda.

> **Note**
>
> Data yang tumpah tidak digunakan kembali dalam eksekusi berikutnya dan dapat dihapus dengan aman setelah 12 jam. Athena tidak menghapus data ini untuk Anda. Untuk mengelola objek ini, pertimbangkan untuk menambahkan kebijakan siklus hidup objek yang menghapus data lama dari bucket tumpahan Amazon S3 Anda. Untuk informasi selengkapnya, lihat Mengelola siklus hidup penyimpanan di Panduan Pengguna Amazon S3.

10. Pilih Saya mengakui bahwa aplikasi ini membuat peran IAM khusus dan kebijakan sumber daya. Untuk informasi selengkapnya, pilih tautan Info.
11. Pilih Deploy. Saat penerapan selesai, fungsi Lambda muncul di bagian Sumber Daya di konsol Lambda.

Menghubungkan ke sumber data

Setelah Anda menyebarkan konektor sumber data ke akun Anda, Anda dapat menghubungkan Athena ke sana.

Untuk menghubungkan Athena ke sumber data menggunakan konektor yang telah Anda gunakan ke akun Anda

1. Kembali ke halaman Masukkan detail sumber data di konsol Athena.
2. Di bagian Detail koneksi, pilih ikon penyegaran di sebelah Pilih atau masukkan kotak pencarian fungsi Lambda.
3. Pilih nama fungsi yang baru saja Anda buat di konsol Lambda. ARN dari fungsi Lambda ditampilkan.

4. (Opsional) Untuk Tag, tambahkan pasangan kunci-nilai untuk dikaitkan dengan sumber data ini. Untuk informasi selengkapnya tentang tag, lihat Menandai sumber daya Athena.
5. Pilih Berikutnya.
6. Pada halaman Tinjau dan buat, tinjau detail sumber data, lalu pilih Buat sumber data.
7. Bagian Detail sumber data pada halaman untuk sumber data Anda menunjukkan informasi tentang konektor baru Anda. Anda sekarang dapat menggunakan konektor dalam kueri Athena Anda.

   Untuk informasi tentang penggunaan konektor data dalam kueri, lihatMenjalankan kueri federasi.

MenggunakanAWS Serverless Application Repositoryuntuk menyebarkan konektor sumber data

Untuk menyebarkan konektor sumber data, Anda dapat menggunakanAWS Serverless Application Repositoryalih-alih memulai dengan konsol Athena. GunakanAWS Serverless Application Repositoryuntuk menemukan konektor yang ingin Anda gunakan, berikan parameter yang dibutuhkan konektor, lalu gunakan konektor ke akun Anda. Kemudian, setelah Anda menggunakan konektor, Anda menggunakan konsol Athena untuk membuat sumber data tersedia untuk Athena.

Menerapkan konektor ke Akun Anda

Untuk menggunakanAWS Serverless Application Repositoryuntuk menggunakan konektor sumber data ke akun

1. Masuklah keAWS Management Consoledan bukaRepositori Aplikasi Tanpa Server.
2. Di panel navigasi, pilihAplikasi yang tersedia.
3. Pilih opsiMenampilkan aplikasi yang membuat IAM role khusus atau kebijakan sumber daya.
4. Di kotak pencarian, ketik nama konektor. Untuk daftar konektor data Athena prebuilt, lihatKonektor sumber data yang tersedia.
5. Pilih nama konektornya. Memilih konektor membuka fungsi LambdaDetail aplikasihalaman diAWS Lambdakonsol.
6. Di sisi kanan halaman detail, untukPengaturan aplikasi, isi informasi yang diperlukan. Pengaturan minimum yang diperlukan meliputi yang berikut ini. Untuk informasi tentang opsi yang dapat dikonfigurasi yang tersisa untuk konektor data yang dibangun oleh Athena, lihat yang sesuaiKonektor yang tersediatopik padaGitHub.

   - AthenaCatalogName- Nama untuk fungsi Lambda dalam huruf kecil yang menunjukkan sumber data yang ditargetkannya, seperti`cloudwatchlogs`.

- SpillBucket— Tentukan bucket Amazon S3 di akun Anda untuk menerima data dari payload respons besar yang melebihi batas ukuran respons fungsi Lambda.

7. PilihSaya mengakui bahwa aplikasi ini membuat peran IAM khusus dan kebijakan sumber daya. Untuk informasi selengkapnya, pilih tautan Info.
8. Di kanan bawahPengaturan aplikasibagian, pilihMenyebarkan. Ketika penyebaran selesai, fungsi Lambda muncul diSumber Dayabagian di konsol Lambda.

Membuat konektor tersedia di Athena

Sekarang Anda siap menggunakan konsol Athena untuk membuat konektor sumber data tersedia untuk Athena.

Untuk membuat konektor sumber data tersedia untuk Athena

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.

3. Di panel navigasi, pilih Sumber data.
4. PadaSumber datahalaman, pilihBuat sumber data.
5. UntukPilih sumber data, pilih sumber data yang Anda buat konektor diAWS Serverless Application Repository. Tutorial ini menggunakanAmazonCloudWatchLogsebagai sumber data federasi.
6. Pilih Selanjutnya.

7. PadaMasukkan rincian sumber datahalaman, untukNama sumber data, masukkan nama yang ingin Anda gunakan dalam pernyataan SQL Anda saat Anda meminta sumber data dari Athena (misalnya,`CloudWatchLogs`). Nama bisa sampai 127 karakter dan harus unik di akun Anda. Itu tidak dapat diubah setelah Anda membuatnya. Karakter yang valid adalah a-z, A-Z, 0-9, _ (garis bawah), @ (pada tanda) dan - (tanda hubung). Nama-namaawsdatacatalog,hive,jmx, dansystemdicadangkan oleh Athena dan tidak dapat digunakan untuk nama sumber data.
8. Di dalamDetail koneksibagian, gunakanPilih atau masukkan fungsi Lambdakotak untuk memilih nama fungsi yang baru saja Anda buat. ARN fungsi Lambda ditampilkan.
9. (Opsional) UntukTag, tambahkan pasangan kunci-nilai untuk mengasosiasikan dengan sumber data ini. Untuk informasi selengkapnya tentang tanda, lihat Menandai sumber daya Athena.
10. Pilih Selanjutnya.
11. PadaTinjau dan buathalaman, meninjau rincian sumber data, dan kemudian memilihBuat sumber data.
12. YangRincian sumber databagian halaman untuk sumber data Anda menunjukkan informasi tentang konektor baru Anda. Anda sekarang dapat menggunakan konektor dalam kueri Athena Anda.

    Untuk informasi tentang menggunakan konektor data dalam kueri, lihatMenjalankan kueri federasi.

Membuat VPC untuk konektor sumber data

Beberapa konektor sumber data Athena memerlukan VPC dan grup keamanan. Topik ini menunjukkan cara membuat VPC dengan subnet dan grup keamanan untuk VPC. Sebagai bagian dari proses ini, Anda mengambil ID untuk VPC, subnet, dan grup keamanan yang Anda buat. ID ini diperlukan saat Anda mengonfigurasi konektor untuk digunakan dengan Athena.

Untuk membuat VPC untuk konektor sumber data Athena

1. Masuk ke AWS Management Console dan buka konsol VPC Amazon di https://console.aws.amazon.com/vpc/.
2. Pilih Buat VPC.
3. Pada halaman Buat VPC, di bawah Pengaturan VPC, agar Sumber Daya dapat dibuat, pilih VPC dan lainnya.
4. Di bawah Generasi otomatis tag nama, untuk Generasi otomatis, masukkan nilai yang akan digunakan untuk menghasilkan tag nama untuk semua sumber daya di VPC Anda.

5. Pilih Buat VPC.
6. Ketika proses selesai, pilih Lihat VPC.
7. Di bagian Detail, untuk ID VPC, salin ID VPC Anda untuk referensi nanti.

Sekarang Anda siap untuk mengambil subnet ID untuk VPC yang baru saja Anda buat.

Untuk mengambil ID subnet VPC Anda

1. Di panel navigasi konsol VPC, pilih Subnet.
2. Pilih nama subnet yang kolom VPC-nya memiliki ID VPC yang Anda catat.
3. Di bagian Detail, untuk Subnet ID, salin subnet ID Anda untuk referensi nanti.

Selanjutnya, Anda membuat grup keamanan untuk VPC Anda.

Untuk membuat grup keamanan untuk VPC Anda

1. Di panel navigasi konsol VPC, pilih Keamanan, Grup Keamanan.
2. Pilih Buat grup keamanan.
3. Pada halaman Buat grup keamanan, masukkan informasi berikut:
   - Untuk nama grup Keamanan, masukkan nama untuk grup keamanan Anda.
   - Untuk Deskripsi, masukkan deskripsi untuk grup keamanan. Deskripsi diperlukan.
   - Untuk VPC, pilih ID VPC dari VPC yang Anda buat untuk konektor sumber data Anda.
   - Untuk aturan Inbound dan aturan Outbound, tambahkan aturan masuk dan keluar yang Anda butuhkan.
4. Pilih Buat grup keamanan.
5. Pada halaman Detail untuk grup keamanan, salin ID grup Keamanan untuk referensi selanjutnya.

Mengaktifkan kueri federasi lintas akun

Kueri gabungan memungkinkan Anda untuk menanyakan sumber data selain Amazon S3 menggunakan konektor sumber data yang digunakan. AWS Lambda Fitur kueri federasi lintas akun memungkinkan fungsi Lambda dan sumber data yang akan ditanyakan ditempatkan di akun yang berbeda.

Sebagai administrator data, Anda dapat mengaktifkan kueri federasi lintas akun dengan membagikan konektor data Anda dengan akun analis data atau, sebagai analis data, dengan menggunakan Lambda ARN bersama dari administrator data untuk ditambahkan ke akun Anda. Ketika perubahan konfigurasi dibuat ke konektor di akun asal, konfigurasi yang diperbarui secara otomatis diterapkan ke instance bersama konektor di akun pengguna lain.

Pertimbangan dan batasan

- Fitur kueri federasi lintas akun tersedia untuk konektor data metastore non-HIVE yang menggunakan sumber data berbasis Lambda.
- Fitur ini tidak tersedia untuk tipe sumber AWS Glue Data Catalog data. Untuk informasi tentang akses lintas akun ke AWS Glue Data Catalog s, lihatAkses lintas akun ke katalog AWS Glue data.
- Jika respons dari fungsi Lambda konektor Anda melebihi batas ukuran respons Lambda sebesar 6MB, Athena secara otomatis mengenkripsi, mengelompokkan, dan menumpahkan respons ke bucket Amazon S3 yang Anda konfigurasikan. Entitas yang menjalankan kueri Athena harus memiliki akses ke lokasi tumpahan agar Athena dapat membaca data yang tumpah. Sebaiknya Anda menetapkan kebijakan siklus hidup Amazon S3 untuk menghapus objek dari lokasi tumpahan karena data tidak diperlukan setelah kueri selesai.
- Menggunakan kueri federasi di seluruh tidak Wilayah AWS didukung.

Izin yang diperlukan

- Agar administrator data Akun A berbagi fungsi Lambda dengan analis data Akun B, Akun B memerlukan fungsi pemanggilan Lambda dan akses tumpahan bucket. Oleh karena itu, Akun A harus menambahkan kebijakan berbasis sumber daya ke fungsi Lambda dan akses utama ke bucket tumpahannya di Amazon S3.
  1. Kebijakan berikut memberikan izin fungsi pemanggilan Lambda ke Akun B pada fungsi Lambda di Akun A.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CrossAccountInvocationStatement",
      "Effect": "Allow",
      "Principal": {
        "AWS": ["arn:aws:iam::account-B-id:user/username"]
      },
```

```
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:aws-region:account-A-id:function:lambda-function-
name"
    }
  ]
}
```

2. Kebijakan berikut memungkinkan akses bucket tumpahan ke prinsipal di Akun 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. Jika fungsi Lambda mengenkripsi bucket tumpahan dengan AWS KMS kunci, bukan enkripsi default yang ditawarkan oleh federasi SDK, kebijakan AWS KMS kunci di Akun A harus memberikan akses ke pengguna di Akun B, seperti pada contoh berikut.

```
{
    "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.
```

```
}
```

- Agar Akun A dapat berbagi konektornya dengan Akun B, Akun B harus membuat peran `AthenaCrossAccountCreate-`*`account-A-id`* yang disebut Akun A dengan memanggil tindakan AssumeRoleAPI Layanan Token AWS Keamanan.

  Kebijakan berikut, yang memungkinkan `CreateDataCatalog` tindakan, harus dibuat di Akun B dan ditambahkan ke `AthenaCrossAccountCreate-`*`account-A-id`* peran yang dibuat Akun B untuk Akun A.

```
{
 "Effect": "Allow",
 "Action": "athena:CreateDataCatalog",
 "Resource": "arn:aws:athena:*:account-B-id:datacatalog/*"
}
```

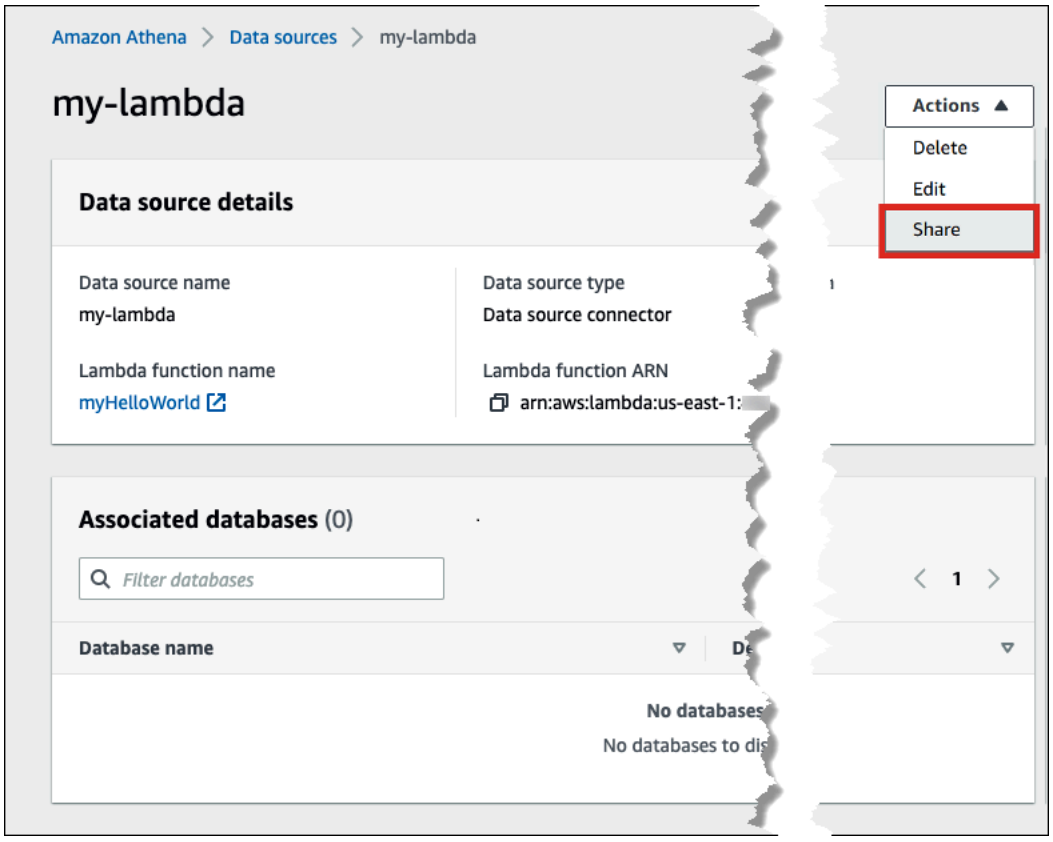
Berbagi sumber data di Akun A dengan Akun B

Setelah izin diberlakukan, Anda dapat menggunakan halaman Sumber data di konsol Athena untuk berbagi konektor data di akun Anda (Akun A) dengan akun lain (Akun B). Akun A mempertahankan kontrol penuh dan kepemilikan konektor. Ketika Akun A membuat perubahan konfigurasi pada konektor, konfigurasi yang diperbarui berlaku untuk konektor bersama di Akun B.

Untuk berbagi sumber data Lambda di Akun A dengan Akun B

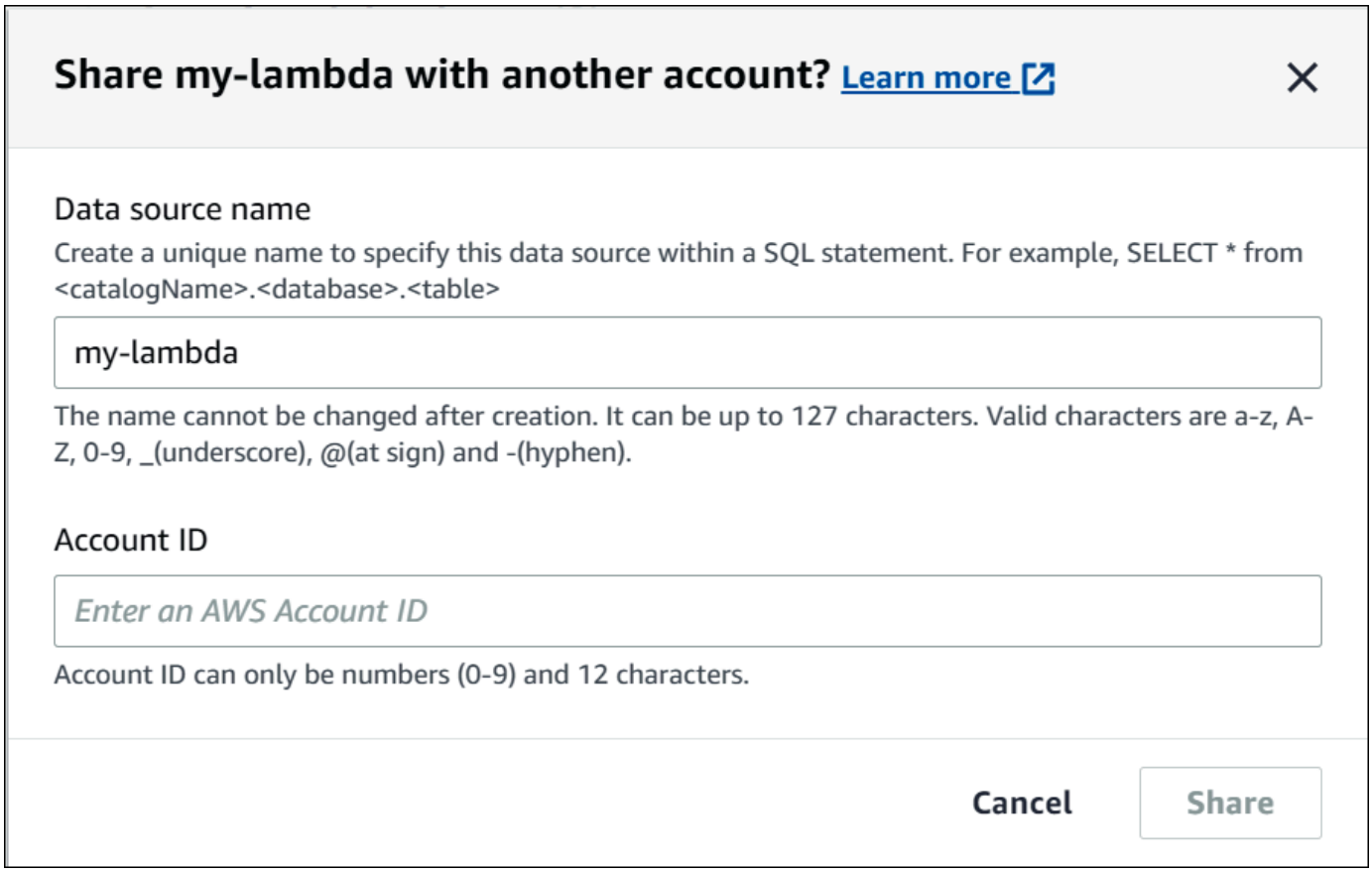
1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.

3. Pilih Sumber data.
4. Pada halaman Sumber data, pilih tautan konektor yang ingin Anda bagikan.
5. Pada halaman detail untuk sumber data Lambda, pilih opsi Bagikan di sudut kanan atas.



6. Di kotak dialog Bagikan ***nama Lambda*** dengan akun lain, masukkan informasi yang diperlukan.

- Untuk nama sumber data, masukkan nama sumber data yang disalin seperti yang Anda inginkan muncul di akun lain.
- Untuk ID Akun, masukkan ID akun yang ingin Anda bagikan sumber data Anda (dalam hal ini, Akun B).



7. Pilih Bagikan. Konektor data bersama yang Anda tentukan dibuat di Akun B. perubahan konfigurasi ke konektor di Akun A berlaku untuk konektor di Akun B.

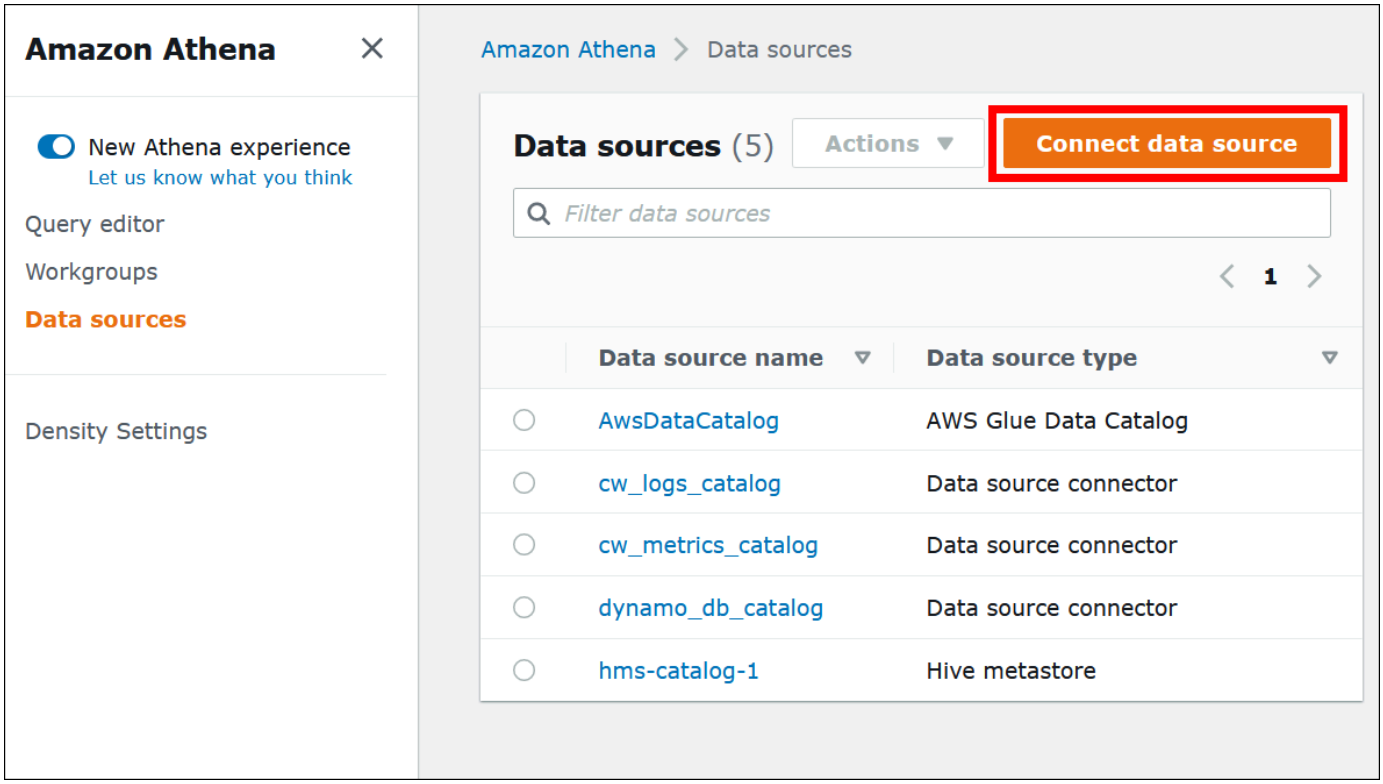
Menambahkan sumber data bersama dari Akun A ke Akun B

Sebagai analis data, Anda mungkin diberikan ARN konektor untuk ditambahkan ke akun Anda dari administrator data. Anda dapat menggunakan halaman Sumber data konsol Athena untuk menambahkan Lambda ARN yang disediakan oleh administrator ke akun Anda.

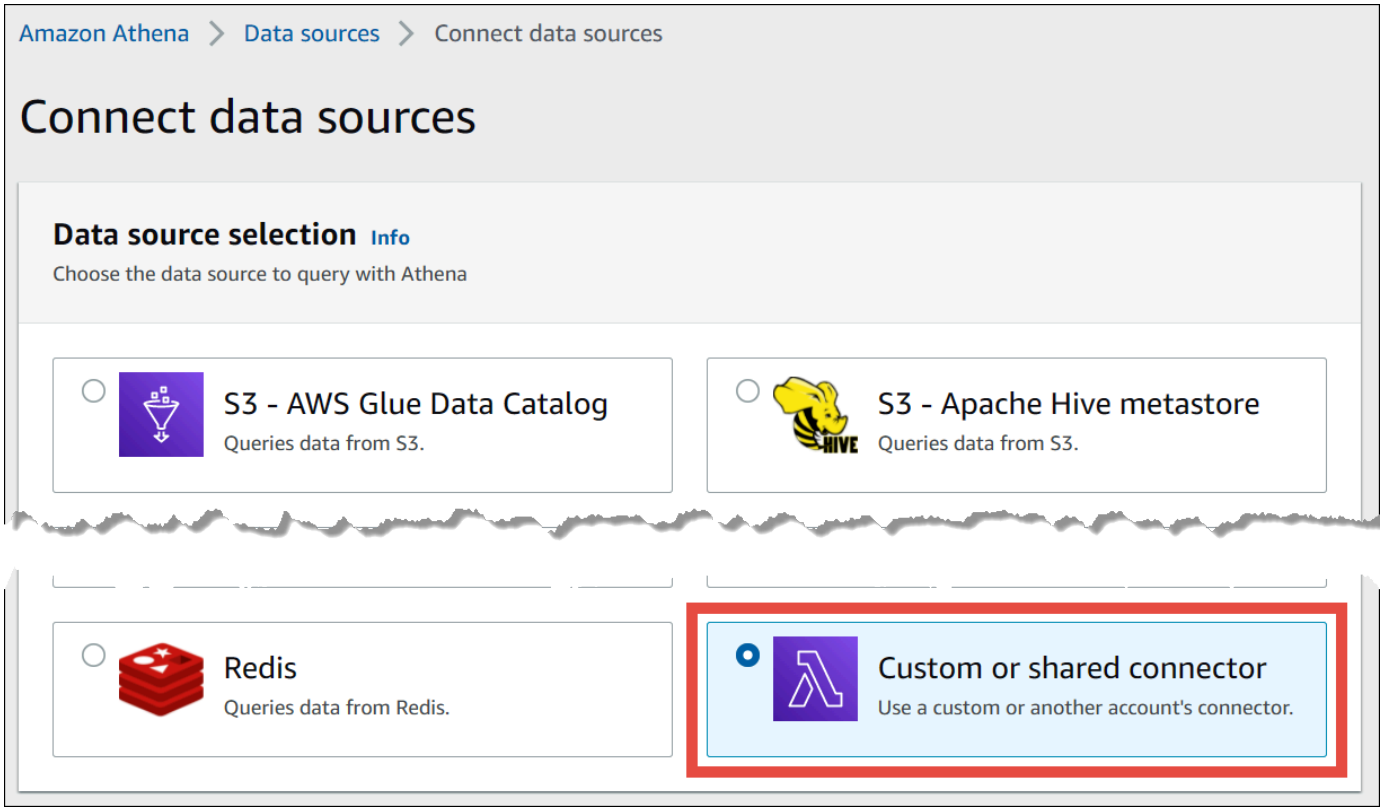
Untuk menambahkan Lambda ARN dari konektor data bersama ke akun Anda

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.

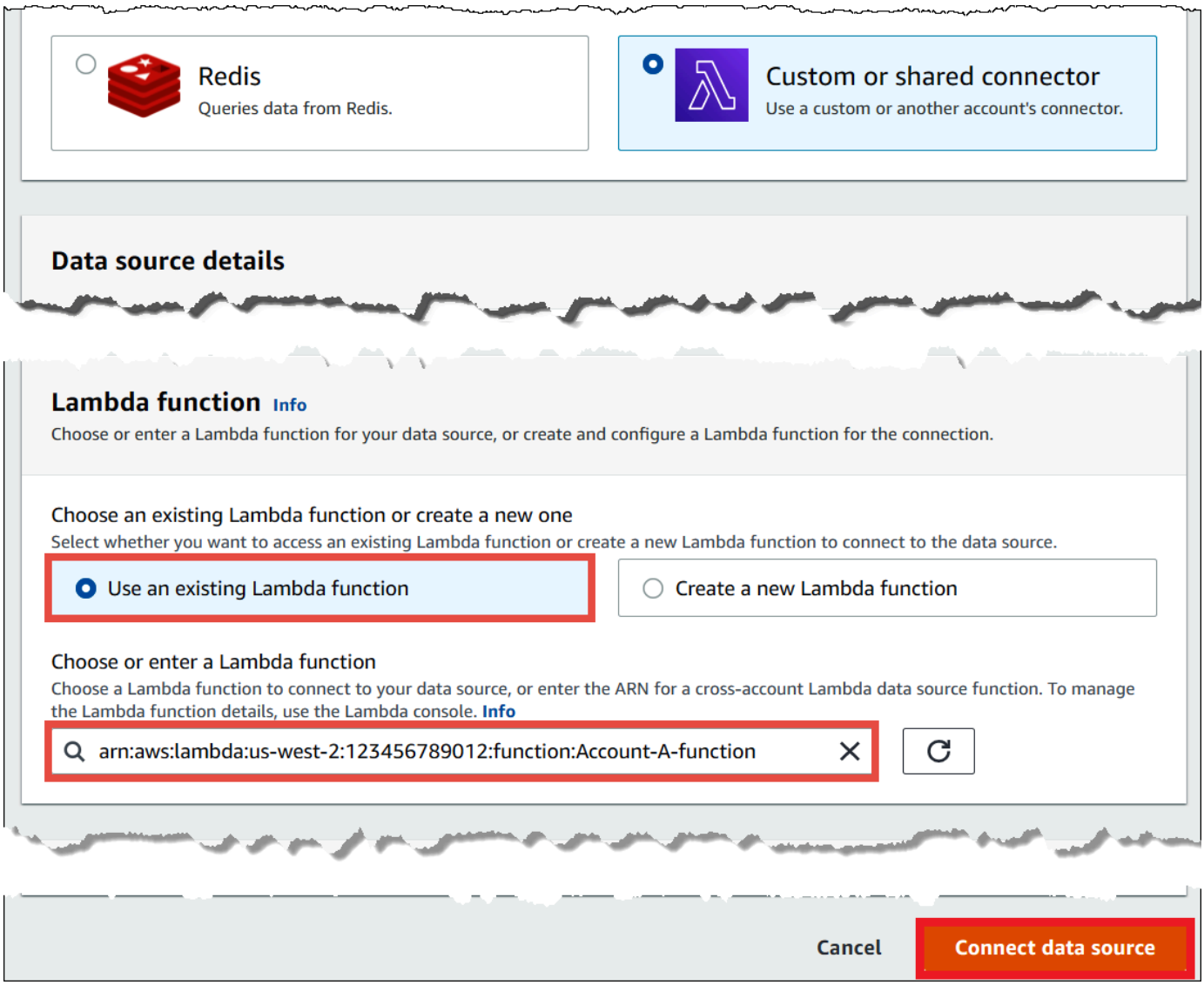
2. Jika Anda menggunakan pengalaman konsol baru dan panel navigasi tidak terlihat, pilih menu ekspansi di sebelah kiri.
3. Pilih Sumber data.
4. Pada halaman Sumber data, pilih Connect data source.



5. Pilih Konektor khusus atau bersama.



6. Di bagian fungsi Lambda, pastikan bahwa opsi Gunakan fungsi Lambda yang ada dipilih.



7. Untuk Pilih atau masukkan fungsi Lambda, masukkan Lambda ARN dari Akun A.
8. PilihConnect sumber.

Pemecahan Masalah

Jika Anda menerima pesan kesalahan bahwa Akun A tidak memiliki izin untuk mengambil peran di Akun B, pastikan bahwa nama peran yang dibuat di Akun B dieja dengan benar dan memiliki kebijakan yang sesuai.

Memperbarui konektor sumber data

Athena merekomendasikan agar Anda secara teratur memperbarui konektor sumber data yang Anda gunakan ke versi terbaru untuk memanfaatkan fitur dan penyempurnaan baru. Untuk memulai, Anda harus menemukan nomor versi terbaru.

Menemukan versi Athena Query Federation terbaru

Nomor versi terbaru dari konektor sumber data Athena sesuai dengan versi Athena Query Federation terbaru. Dalam kasus-kasus tertentu,GitHubrilis bisa sedikit lebih baru dari apa yang tersedia diAWS Serverless Application Repository(SAR).

Untuk menemukan nomor versi Athena Query Federation terbaru

1. KunjungiGitHubURLhttps://github.com/awslabs/aws-athena-query-federation/rilis/terbaru.
2. Perhatikan nomor rilis di judul halaman utama dalam format berikut:

   Rilis v *tahun*.*minggu_of_tahun*.*iterasi_of_week* Federasi kueri Athena

   Misalnya, nomor rilis untukRilis v2023.8.3 dari Federasi Kueri Athenaadalah 2023.8.3.

Menemukan dan mencatat nama sumber daya

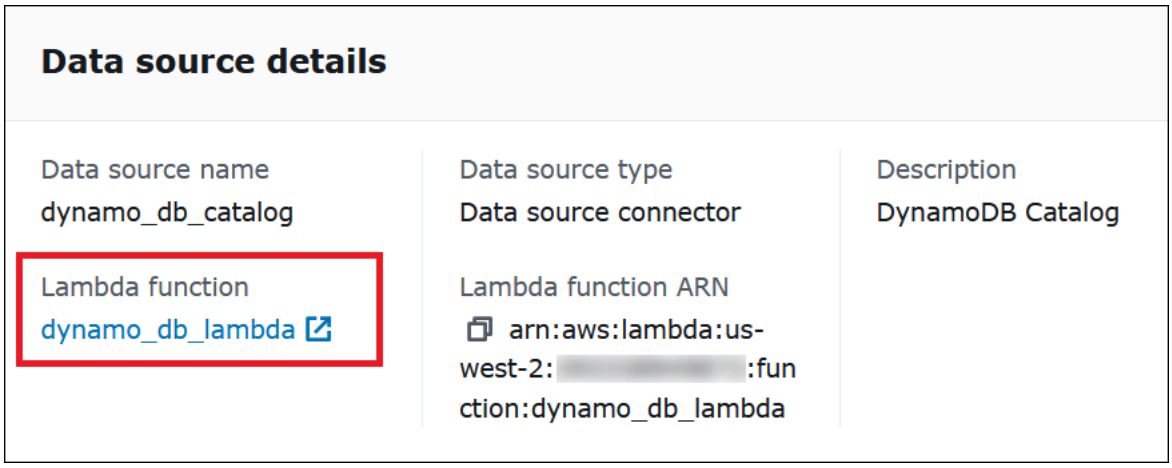
Dalam persiapan untuk upgrade, Anda harus menemukan dan mencatat informasi berikut:

1. Nama fungsi Lambda untuk konektor.
2. Variabel lingkungan fungsi Lambda.
3. Nama aplikasi Lambda, yang mengelola fungsi Lambda untuk konektor.

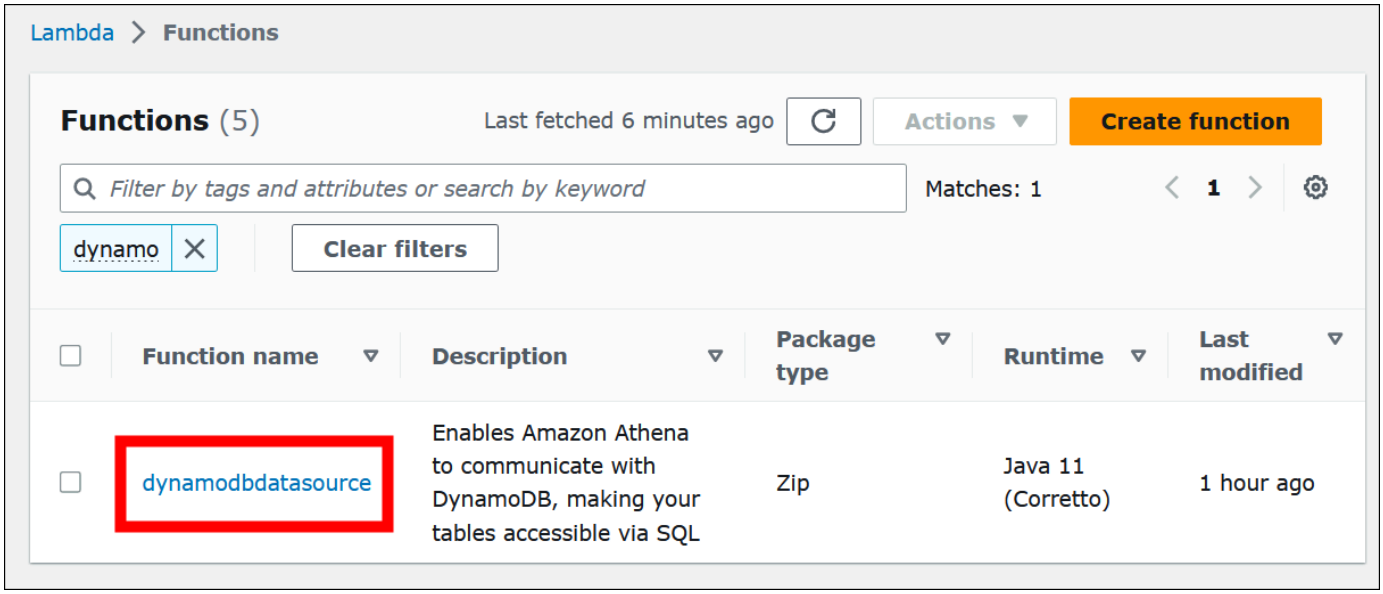
Untuk menemukan nama sumber daya dari konsol Athena

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.

3. Di panel navigasi, pilih Sumber data.
4. DalamNama sumber datakolom, pilih tautan ke sumber data untuk konektor Anda.
5. DalamRincian sumber databagian, di bawahFungsi lambda, pilih tautan ke fungsi Lambda Anda.



6. PadaFungsihalaman, diNama fungsikolom, perhatikan nama fungsi untuk konektor Anda.



7. Pilih link nama fungsi.
8. Di bawahIkhtisar fungsibagian, pilihKonfigurasitab.
9. Di panel di sebelah kiri, pilihVariabel lingkungan.
10. DalamVariabel lingkunganbagian, membuat catatan dari kunci dan nilai-nilai yang sesuai mereka.
11. Gulir ke bagian atas halaman.
12. Dalam pesanFungsi ini milik aplikasi. Klik di sini untuk mengelolanya, pilihKlik di sinitautan.
13. Padatanpa server***_aplikasi_namamu***halaman, membuat catatan nama aplikasi Anda tanparepo tanpa server. Misalnya, jika nama aplikasitanpa serverDynamoDbTestApp, maka nama aplikasi AndaDynamoDbTestApp.
14. Tetap di halaman konsol Lambda untuk aplikasi Anda, dan kemudian lanjutkan dengan langkah-langkah diMenemukan versi konektor yang Anda gunakan.

Menemukan versi konektor yang Anda gunakan

Ikuti langkah-langkah ini untuk menemukan versi konektor yang Anda gunakan.

Untuk menemukan versi konektor yang Anda gunakan

1. Pada halaman konsol Lambda untuk aplikasi Lambda Anda, pilihDeploymenttab.
2. PadaDeploymenttab, memperluasTemplat SAM.
3. Cari CodeUri.

4. DalamKuncibidang di bawahCodeUri, temukan string berikut:

```
applications-connector_name-
versions-year.week_of_year.iteration_of_week/hash_number
```

   Contoh berikut menunjukkan string untukCloudWatchkonektor:

```
applications-AthenaCloudwatchConnector-versions-2021.42.1/15151159...
```

5. Catat nilai untuk*tahun.minggu_of_tahun.iterasi_of_week*(misalnya,2021.42.1). Ini adalah versi untuk konektor Anda.

Menerapkan versi baru konektor Anda

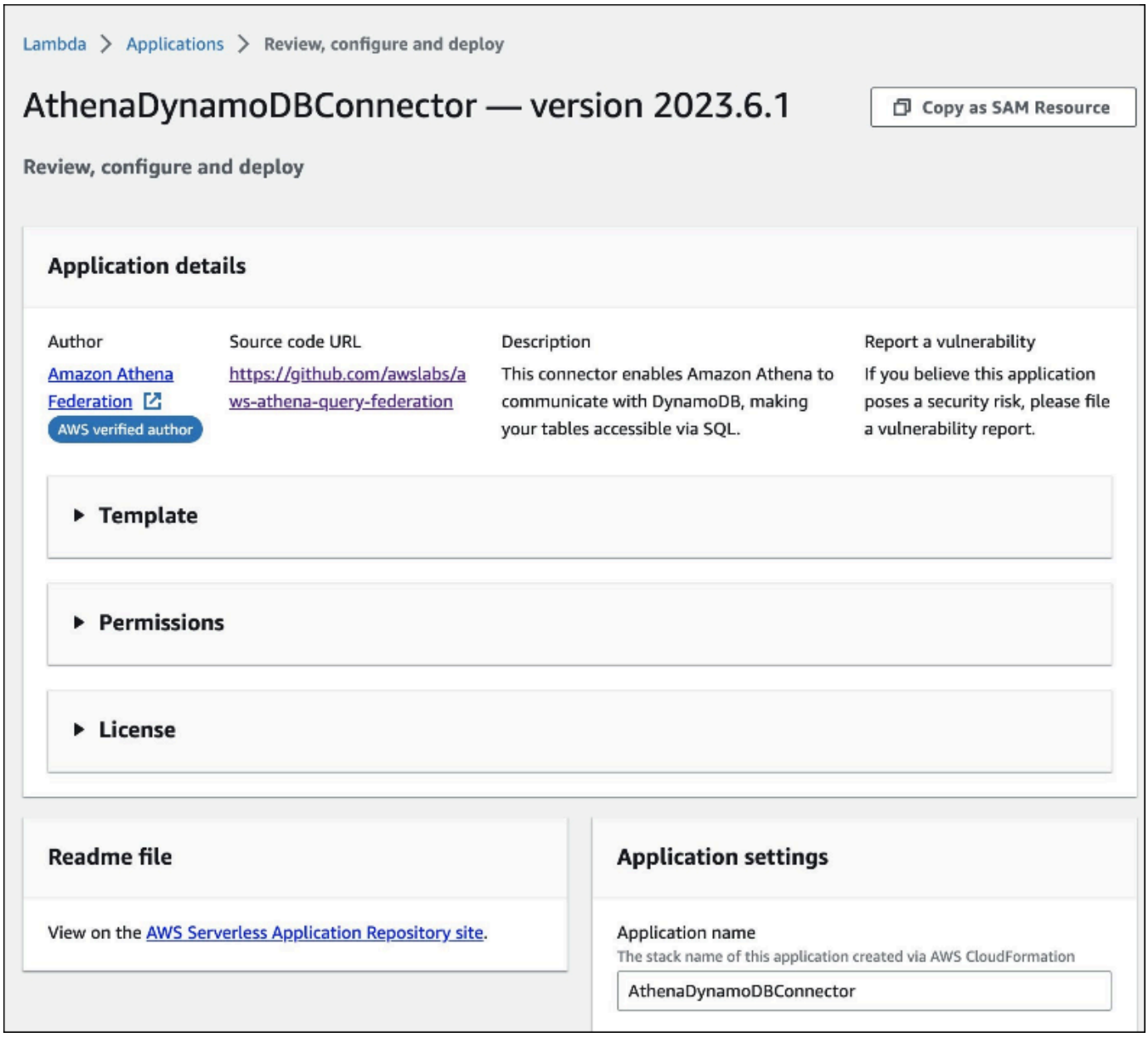
Ikuti langkah-langkah ini untuk menerapkan versi baru konektor Anda.

Untuk menerapkan versi baru konektor Anda

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.

3. Di panel navigasi, pilih Sumber data.
4. PadaSumber datahalaman, pilihBuat sumber data.
5. Pilih sumber data yang ingin Anda tingkatkan, lalu pilihBerikutnya.

6. DalamDetail koneksibagian, pilihBuat fungsi Lambda. Ini membuka konsol Lambda di mana Anda akan dapat menerapkan aplikasi yang diperbarui.

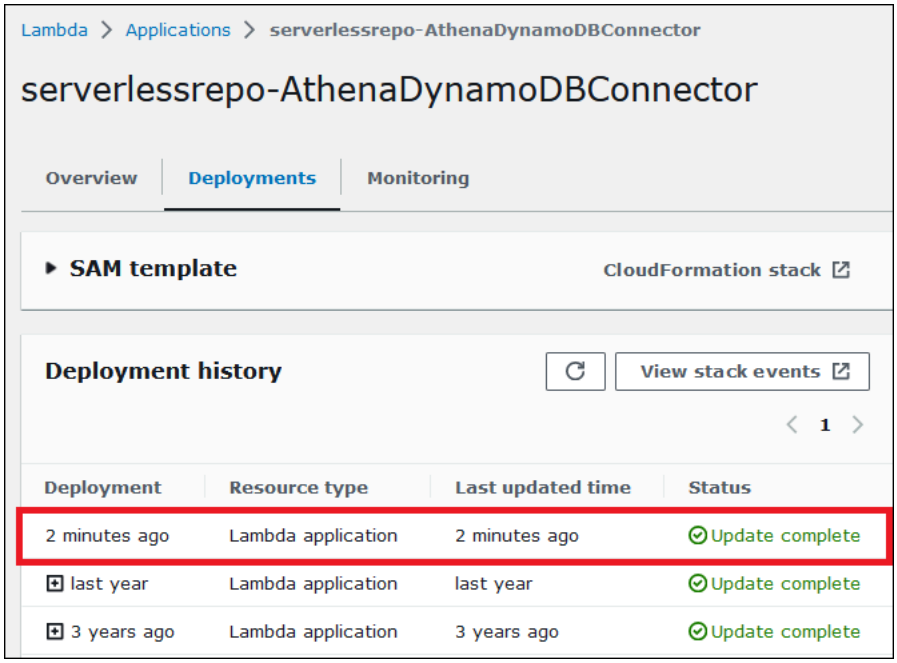


7. Karena Anda tidak benar-benar membuat sumber data baru, Anda dapat menutup tab konsol Athena.
8. Pada halaman konsol Lambda untuk konektor, lakukan langkah-langkah berikut:
   a. Pastikan bahwa Anda telah menghapustanpa serverawalan dari nama aplikasi Anda, dan kemudian salin nama aplikasi keNama aplikasibidang.
   b. Salin nama fungsi Lambda Anda keAthenaCatalogNamebidang. Beberapa konektor memanggil bidang iniLambdaFunctionName.

c. Salin variabel lingkungan yang Anda rekam ke bidang yang sesuai.

9. Pilih opsiSaya mengakui bahwa aplikasi ini membuat peran IAM khusus dan kebijakan sumber daya, dan kemudian pilihMenyebarkan.
10. Untuk memverifikasi bahwa aplikasi Anda telah diperbarui, pilihDeploymenttab.

    YangRiwayat penyebaranbagian menunjukkan bahwa pembaruan Anda selesai.



11. Untuk mengonfirmasi nomor versi baru, Anda dapat memperluasTemplat SAMseperti sebelumnya, temukanCodeUri, dan periksa nomor versi konektor diKuncibidang.

Anda sekarang dapat menggunakan konektor Anda diperbarui untuk membuat Athena query federasi.

Menjalankan kueri federasi

Setelah Anda mengonfigurasi satu atau lebih konektor data dan menyebarkannya ke akun Anda, Anda dapat menggunakannya dalam kueri Athena Anda.

Menanyakan satu sumber data

Contoh di bagian ini mengasumsikan bahwa Anda telah mengonfigurasi dan menerapkan Konektor Amazon Athena CloudWatch ke akun Anda. Gunakan pendekatan yang sama untuk mengkueri saat Anda menggunakan konektor lain.

Untuk membuat kueri Athena yang menggunakan konektor CloudWatch

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Di editor kueri Athena, buat kueri SQL yang menggunakan sintaks berikut dalam klausa. FROM

```
MyCloudwatchCatalog.database_name.table_name
```

Contoh

Contoh berikut menggunakan CloudWatch konektor Athena untuk terhubung ke `all_log_streams` tampilan di grup Log /var/ecommerce-engine/order-processor CloudWatch Log. Parameter`all_log_streams`tampilan adalah tampilan dari semua aliran log dalam grup log. Contoh kueri membatasi jumlah baris kembali ke 100.

```
SELECT *
FROM "MyCloudwatchCatalog"."/var/ecommerce-engine/order-processor".all_log_streams
LIMIT 100;
```

Contoh berikut mem-parsing informasi dari tampilan yang sama seperti contoh sebelumnya. Contoh ekstrak urutan ID dan level log dan menyaring pesan yang memiliki level`INFO`.

```
SELECT
    log_stream as ec2_instance,
    Regexp_extract(message '.*orderId=(\d+) .*', 1) AS orderId,
    message AS order_processor_log,
    Regexp_extract(message, '(.*):.*', 1) AS log_level
FROM MyCloudwatchCatalog."/var/ecommerce-engine/order-processor".all_log_streams
WHERE Regexp_extract(message, '(.*):.*', 1) != 'INFO'
```

Meminta beberapa sumber data

Sebagai contoh yang lebih kompleks, bayangkan sebuah perusahaan e-commerce yang menggunakan sumber data berikut untuk menyimpan data yang terkait dengan pembelian pelanggan:

- Amazon RDS for MySQL untuk menyimpan data katalog produk
- Amazon DocumentDB untuk menyimpan data akun pelanggan seperti email dan alamat pengiriman
- Amazon DynamoDB untuk menyimpan pengiriman pesanan dan data pelacakan

Bayangkan seorang analis data untuk aplikasi e-commerce ini mengetahui bahwa waktu pengiriman di beberapa daerah telah dipengaruhi oleh kondisi cuaca setempat. Analis ingin mengetahui berapa banyak pesanan yang tertunda, di mana pelanggan yang terkena dampak berada, dan produk mana yang paling terpengaruh. Alih-alih menyelidiki sumber informasi secara terpisah, analis menggunakan Athena untuk menggabungkan data bersama dalam satu kueri federasi.

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
```

Menanyakan pandangan federasi

Saat menanyakan sumber federasi, Anda dapat menggunakan tampilan untuk mengaburkan sumber data yang mendasarinya atau menyembunyikan gabungan kompleks dari analis lain yang menanyakan data.

Pertimbangan dan batasan

- Tampilan federasi membutuhkan mesin Athena versi 3.
- Tampilan federasi disimpan di AWS Glue, bukan dengan sumber data yang mendasarinya.
- Tampilan yang dibuat dengan katalog federasi harus menggunakan sintaks nama yang sepenuhnya memenuhi syarat, seperti pada contoh berikut:

  ```
  "ddbcatalog"."default"."customers"
  ```

- Pengguna yang menjalankan kueri pada sumber federasi harus memiliki izin untuk menanyakan sumber federasi.

- `athena:GetDataCatalog`Izin diperlukan untuk pandangan federasi. Untuk informasi selengkapnya, lihat Contoh kebijakan izin IAM untuk mengizinkan Kueri Federasi Athena.

Contoh

Contoh berikut membuat tampilan yang disebut `customers` pada data yang disimpan dalam sumber data federasi.

Example

```
CREATE VIEW customers AS
SELECT *
FROM my_federated_source.default.table
```

Contoh kueri berikut menunjukkan kueri yang mereferensikan `customers` tampilan, bukan sumber data federasi yang mendasarinya.

Example

```
SELECT id, SUM(order_amount)
FROM customers
GROUP by 1
ORDER by 2 DESC
LIMIT 50
```

Contoh berikut membuat tampilan yang disebut `order_summary` yang menggabungkan data dari sumber data federasi dan dari sumber data Amazon S3. Dari sumber federasi, yang telah dibuat di Athena, tampilan menggunakan tabel `person` dan`profile`. Dari Amazon S3, tampilan menggunakan tabel `purchase` dan`payment`. Untuk merujuk ke Amazon S3, pernyataan tersebut menggunakan kata kunci. `awsdatacatalog` *Perhatikan bahwa sumber data federasi menggunakan sintaks nama yang sepenuhnya memenuhi syarat federated_source_name. federated_source_database*. *federated_source_table*.

Example

```
CREATE VIEW default.order_summary AS
SELECT *
FROM federated_source_name.federated_source_database."person" p
    JOIN federated_source_name.federated_source_database."profile" pr ON pr.id = p.id
```

```
    JOIN awsdatacatalog.default.purchase i ON p.id = i.id
    JOIN awsdatacatalog.default.payment pay ON pay.id = p.id
```

Sumber daya tambahan

- Untuk contoh tampilan federasi yang dipisahkan dari sumber asalnya dan tersedia untuk analisis sesuai permintaan dalam model multi-pengguna, lihat Memperluas mesh data Anda dengan Amazon Athena dan tampilan gabungan di Blog Big Data.AWS
- Untuk informasi lebih lanjut tentang bekerja dengan pemandangan di Athena, lihat. Bekerja dengan pandangan

Menjalankan kueri passthrough federasi

Di Athena, Anda dapat menjalankan kueri pada sumber data federasi menggunakan bahasa kueri dari sumber data itu sendiri dan mendorong kueri lengkap ke sumber data untuk dieksekusi. Kueri ini disebut kueri passthrough. Untuk menjalankan kueri passthrough, Anda menggunakan fungsi tabel dalam kueri Athena Anda. Anda menyertakan kueri passthrough untuk dijalankan pada sumber data di salah satu argumen ke fungsi tabel. Melewati kueri mengembalikan tabel yang dapat Anda analisis menggunakan Athena SQL.

Konektor yang didukung

Konektor sumber data Athena berikut mendukung kueri passthrough.

- Penyimpanan Danau Data Azure
- Sinaps Azure
- Sarang Cloudera
- Cloudera Impala
- CloudWatch
- Db2
- Db2 iSeries
- DocumentDB
- DynamoDB
- HBase
- Google BigQuery

- Hortonworks
- MySQL
- Neptune
- OpenSearch
- Oracle
- PostgreSQL
- Redshift
- SAP HANA
- Kepingan salju
- SQL Server
- Teradata
- Timestream
- Vertica

Pertimbangan dan batasan

Saat menggunakan kueri passthrough di Athena, pertimbangkan hal-hal berikut:

- Passthrough kueri hanya didukung untuk pernyataan `SELECT` Athena atau operasi baca.
- Kueri passthrough harus dijalankan dalam konteks katalog kueri luar (yaitu, kueri yang memanggil fungsi tabel).
- Kinerja kueri dapat bervariasi tergantung pada konfigurasi sumber data.
- Kueri passthrough tidak didukung untuk tampilan.

Sintaks

Sintaks passthrough kueri Athena umum adalah sebagai berikut.

```
SELECT * FROM TABLE(system.function_name(arg1 => 'arg1Value'[, arg2 =>
 'arg2Value', ...]))
```

Untuk sebagian besar sumber data, argumen pertama dan satu-satunya `query` diikuti oleh operator panah => dan string kueri.

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

Untuk mempermudah, Anda dapat menghilangkan argumen bernama opsional `query` dan operator `=>` panah.

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

Jika sumber data membutuhkan lebih dari string kueri, gunakan argumen bernama dalam urutan yang diharapkan oleh sumber data. Misalnya, ekspresi *`arg1`* `=>` '*`arg1Value`*' berisi argumen pertama dan nilainya. Nama *`arg1`* khusus untuk sumber data dan dapat berbeda dari konektor ke konektor.

```
SELECT * FROM TABLE(
        system.query(
            arg1 => 'arg1Value',
            arg2 => 'arg2Value',
            arg3 => 'arg3Value'
        ));
```

Untuk informasi tentang sintaks yang tepat untuk digunakan dengan konektor tertentu, lihat halaman konektor individual.

Penggunaan tanda kutip

Nilai argumen, termasuk string kueri yang Anda lewati, harus diapit dalam tanda kutip tunggal, seperti pada contoh berikut.

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

Ketika string kueri dikelilingi oleh tanda kutip ganda, kueri gagal. Kueri berikut gagal dengan pesan kesalahan COLUMN_NOT_FOUND: baris 1:43: Kolom 'pilih * dari batas testdb.persons 10' tidak dapat diselesaikan.

```
SELECT * FROM TABLE(system.query(query => "SELECT * FROM testdb.persons LIMIT 10"))
```

Untuk menghindari satu kutipan, tambahkan satu kutipan ke aslinya (misalnya, `terry's_group keterry''s_group`).

Contoh

Contoh query berikut mendorong ke bawah query ke sumber data. Kueri memilih semua kolom dalam `customer` tabel, membatasi hasilnya menjadi 10.

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

Pernyataan berikut menjalankan query yang sama, tetapi menghilangkan argumen bernama opsional `query` dan operator panah=>.

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

Athena dan kualifikasi nama tabel federasi

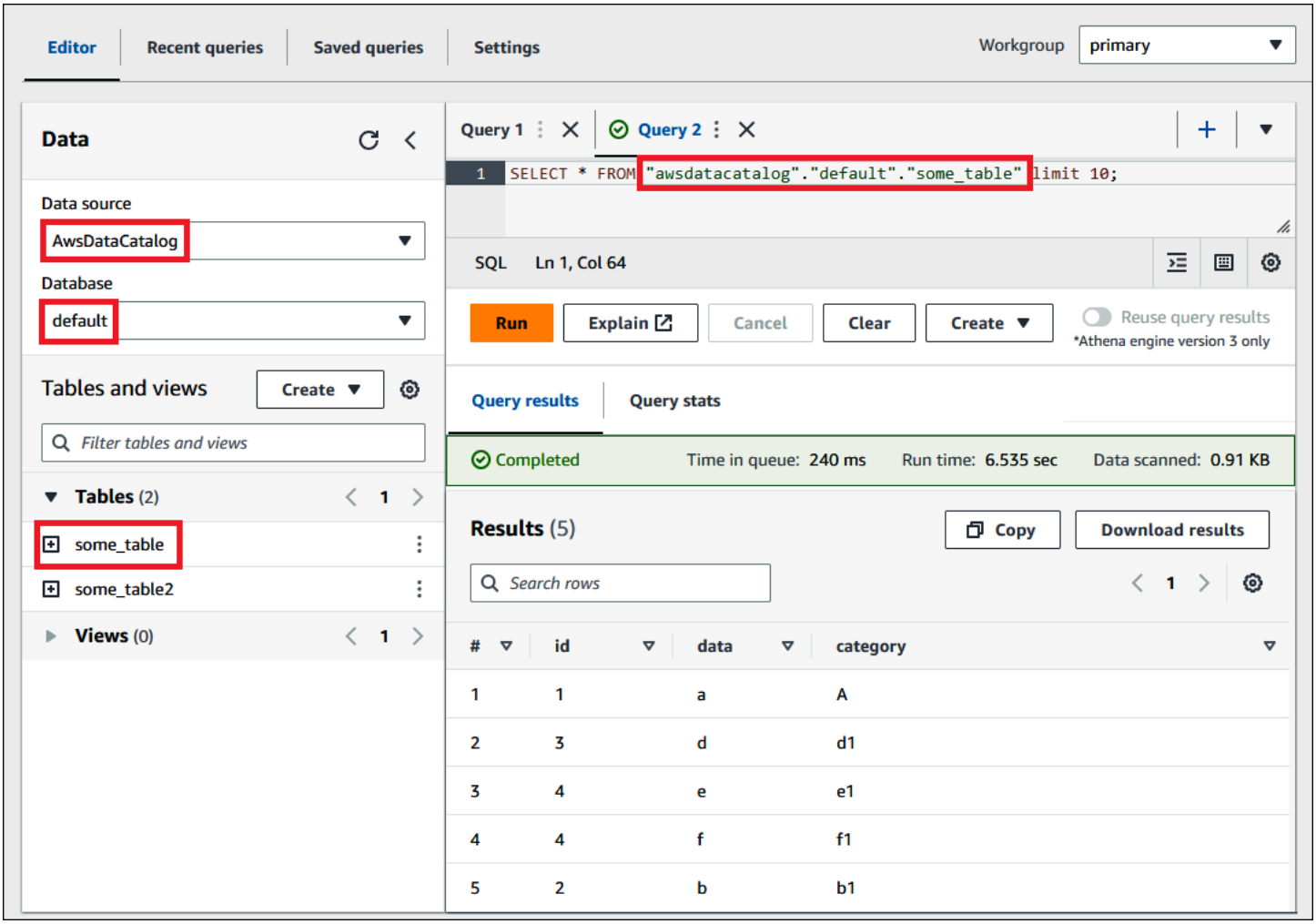
Athena menggunakan istilah-istilah berikut untuk merujuk pada hierarki objek data:

- Sumber data- sekelompok database
- Basis data- sekelompok tabel
- Tabel- Data yang diatur sebagai sekelompok baris atau kolom

Terkadang benda-benda ini juga disebut dengan nama alternatif tetapi setara seperti berikut ini:

- Sumber data kadang-kadang disebut sebagaikatalog.
- Database kadang-kadang disebut sebagaiskema.

Contoh berikut query di konsol Athena menggunakan`awsdatacatalog`sumber data,`default`database, dan`some_table`meja.



Ketentuan dalam sumber data federasi

Saat Anda mengkueri sumber data federasi, perhatikan bahwa sumber data yang mendasarinya mungkin tidak menggunakan terminologi yang sama dengan Athena. Ingatlah perbedaan ini saat Anda menulis kueri federasi Anda. Bagian berikut menjelaskan bagaimana istilah objek data di Athena sesuai dengan yang ada di sumber data federasi.

Amazon Redshift

Amazon Redshiftbasis dataadalah sekelompok Redshiftskemayang berisi sekelompok Redshifttabel.

Athena	Pergeseran merah
Sumber data Redshift	Sebuah fungsi konektor Redshift Lambda dikonfigurasi untuk menunjuk ke Redshift`database`.

Athena	Pergeseran merah
`data_source.database.table`	`database.schema.table`

Kueri contoh

```
SELECT * FROM
Athena_Redshift_connector_data_source.Redshift_schema_name.Redshift_table_name
```

Untuk informasi selengkapnya tentang konektor ini, lihatKonektor Pergeseran Merah Amazon Athena.

Sarang Cloudera

Sebuah Sarang Clouderapeladenataugugusadalah sekelompok Cloudera Hivebasis datayang berisi sekelompok Cloudera Hivetabel.

Athena	Hive
Sumber data Cloudera Hive	Konektor Cloudera Hive Fungsi Lambda dikonfigurasi untuk menunjuk ke Cloudera Hive`server`.
`data_source.database.table`	`server.database.table`

Kueri contoh

```
SELECT * FROM
Athena_Cloudera_Hive_connector_data_source.Cloudera_Hive_database_name.Cloudera_Hive_table_name
```

Untuk informasi selengkapnya tentang konektor ini, lihatKonektor Amazon Athena Cloudera Hive.

Cloudera

Impalapeladenataugugusadalah sekelompok Impalabasis datayang berisi sekelompok Impalatabel.

Athena	Impala
Sumber data Impala	Impala konektor fungsi Lambda dikonfigurasi untuk menunjuk ke Impala`server`.
`data_source.database.table`	`server.database.table`

Kueri contoh

```
SELECT * FROM
Athena_Impala_connector_data_source.Impala_database_name.Impala_table_name
```

Untuk informasi selengkapnya tentang konektor ini, lihatKonektor Amazon Athena Cloudera Impala.

MySQL

Sebuah MySQLpeladenadalah kelompok MySQLbasis datayang berisi sekelompok MySQLtabel.

Athena	MySQL
Sumber data MySQL	Konektor MySQL fungsi Lambda dikonfigurasi untuk menunjuk ke MySQL`server`.
`data_source.database.table`	`server.database.table`

Kueri contoh

```
SELECT * FROM
Athena_MySQL_connector_data source.MySQL_database_name.MySQL_table_name
```

Untuk informasi selengkapnya tentang konektor ini, lihatKonektor MySQL Amazon Athena.

Oracle

Oraclepeladen(ataubasis data) adalah sekelompok Oracleskemayang berisi sekelompok Oracletabel.

Athena	Oracle
Sumber data Oracle	Oracle konektor fungsi Lambda dikonfigurasi untuk menunjuk ke Oracle`server`.
`data_source.database.table`	`server.schema.table`

Kueri contoh

```
SELECT * FROM
Athena_Oracle_connector_data_source.Oracle_schema_name.Oracle_table_name
```

Untuk informasi selengkapnya tentang konektor ini, lihatKonektor Amazon Athena Oracle.

Postgres

Sebuah Postgrespeladen(ataugugus) adalah sekelompok Postgresbasis data. Sebuah Postgresbasis dataadalah sekelompok Postgresskemayang berisi sekelompok Postgrestabel.

Athena	Postgres
Sumber data Postgres	Postgres konektor fungsi Lambda dikonfigurasi untuk menunjuk ke Postgres sebuah`server`dan`database`.
`data_source.database.table`	`server.database.schema.table`

Kueri contoh

```
SELECT * FROM
Athena_Postgres_connector_data_source.Postgres_schema_name.Postgres_table_name
```

Untuk informasi selengkapnya tentang konektor ini, lihatKonektor Amazon Athena PostgreSQL.

Mengembangkan konektor sumber data menggunakan Athena Query Federation SDK

Untuk menulis sendiriKonektor sumber, Anda dapat menggunakanAthena Kueri Gabungan SDK. Athena Kueri Federation SDK mendefinisikan satu set antarmuka dan kawat protokol yang dapat

Anda gunakan untuk mengaktifkan Athena untuk mendelegasikan bagian dari rencana eksekusi kueri untuk kode yang Anda tulis dan men-deploy. SDK termasuk suite penyambung dan penyambung contoh.

Anda juga dapat menyesuaikan Amazon Athenakonektor prebuiltuntuk penggunaan Anda sendiri. Anda dapat memodifikasi salinan kode sumber dari GitHub dan kemudian menggunakan alat publikasi Konektor untuk membuat AWS Serverless Application Repository paket Anda sendiri. Setelah Anda men-deploy konektor Anda dengan cara ini, Anda dapat menggunakannya dalam pertanyaan Athena Anda.

Untuk informasi tentang cara mengunduh SDK dan petunjuk terperinci untuk menulis konektor Anda sendiri, lihat Contoh konektor Athena aktif. GitHub

Konektor sumber data Athena untuk Apache Spark

Beberapa konektor sumber data Athena tersedia sebagai konektor Spark DSV2. Nama konektor Spark DSV2 memiliki `-dsv2` akhiran (misalnya,). `athena-dynamodb-dsv2`

Berikut ini adalah konektor DSV2 yang tersedia saat ini, nama `.format()` kelas Spark mereka, dan tautan ke dokumentasi Kueri Federasi Amazon Athena yang sesuai:

Konektor DSV2	Spark .format () nama kelas	Dokumentasi
athena-cloudwatch-dsv2	`com.amazonaws.athena.connectors.dsv2.cloudwatch.CloudwatchTableProvider`	CloudWatch
athena-cloudwatch-metrics-dsv2	`com.amazonaws.athena.connectors.dsv2.cloudwatch.metrics.CloudwatchMetricsTableProvider`	CloudWatch metrik
athena-aws-cmdb-dsv2	`com.amazonaws.athena.connectors.dsv2`	CMDB

Konektor DSV2	Spark .format () nama kelas	Dokumentasi
	`.aws.cmdb.AwsCmdbTableProvider`	
athena-dynamodb-dsv2	`com.amazonaws.athena.connectors.dsv2.dynamodb.DDBTableProvider`	DynamoDB

Untuk mengunduh **`.jar`** file untuk konektor DSV2, kunjungi halaman GitHub DSV2 Federasi Kueri Amazon Athena dan lihat bagian Rilis, Rilis, Aset. <version>

Menentukan toples ke Spark

Untuk menggunakan konektor Athena DSV2 dengan Spark, Anda mengirimkan `.jar` file untuk konektor ke lingkungan Spark yang Anda gunakan. Bagian berikut menjelaskan kasus-kasus tertentu.

Athena untuk Spark

Untuk informasi tentang menambahkan `.jar` file kustom dan konfigurasi kustom ke Amazon Athena untuk Apache Spark, lihat. Menambahkan file JAR dan konfigurasi Spark khusus

General Spark

Untuk meneruskan `.jar` file konektor ke Spark, gunakan `spark-submit` perintah dan tentukan `.jar` file dalam `--jars` opsi, seperti pada contoh berikut:

```
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

Untuk menjalankan `spark-submit` perintah dengan `--jars` parameter di Amazon EMR, Anda harus menambahkan langkah ke cluster Amazon EMR Spark Anda. Untuk detail tentang cara menggunakan `spark-submit` di Amazon EMR, lihat Menambahkan langkah Spark di Panduan Rilis Amazon EMR.

AWS Glue Percikan ETL

Untuk AWS Glue ETL, Anda dapat meneruskan URL GitHub .com `.jar` file ke `--extra-jars` argumen `aws glue start-job-run` perintah. AWS Glue Dokumentasi menjelaskan `--extra-jars` parameter sebagai mengambil jalur Amazon S3, tetapi parameter juga dapat mengambil URL HTTPS. Untuk informasi selengkapnya, lihat Referensi parameter Job di Panduan AWS Glue Pengembang.

Meminta konektor pada Spark

Untuk mengirimkan yang setara dengan kueri federasi Athena Anda yang ada di Apache Spark, gunakan fungsinya. `spark.sql()` Misalnya, Anda memiliki kueri Athena berikut yang ingin Anda gunakan pada Apache Spark.

```
SELECT somecola, somecolb, somecolc
FROM ddb_datasource.some_schema_or_glue_database.some_ddb_or_glue_table
WHERE somecola > 1
```

Untuk melakukan kueri yang sama di Spark menggunakan konektor Amazon Athena DynamoDB DSV2, gunakan kode berikut:

```
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
''')
```

Menentukan parameter

Versi DSV2 dari konektor sumber data Athena menggunakan parameter yang sama dengan konektor sumber data Athena yang sesuai. Untuk informasi parameter, lihat dokumentasi untuk konektor sumber data Athena yang sesuai.

Dalam PySpark kode Anda, gunakan sintaks berikut untuk mengonfigurasi parameter Anda.

```
spark.read.option("athena.connectors.conf.parameter", "value")
```

Misalnya, kode berikut menetapkan parameter konektor DynamoDB Amazon Athena ke. `disable_projection_and_casing` `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())
```

Kebijakan IAM untuk mengakses katalog data

Untuk mengontrol akses ke katalog data, gunakan izin IAM level sumber daya atau kebijakan IAM berbasis identitas.

Prosedur berikut khusus untuk Athena.

Untuk informasi khusus IAM, lihat tautan yang tercantum di akhir bagian ini. Untuk informasi tentang kebijakan katalog data contoh JSON, lihat Contoh kebijakan Katalog Data.

Untuk menggunakan editor visual di konsol IAM untuk membuat kebijakan katalog data

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi yang ada di sebelah kiri, pilih Kebijakan, lalu pilih Buat kebijakan.
3. Pada tab Editor visual, pilih Pilih layanan. Pilih Athena untuk ditambahkan ke kebijakan.
4. Pilih Pilih tindakan, kemudian pilih tindakan untuk ditambahkan ke kebijakan. Editor visual menunjukkan tindakan yang tersedia di Athena. Untuk informasi selengkapnya, lihat Kunci tindakan, sumber daya, dan kondisi untuk Amazon Athena di Referensi Otorisasi Layanan.
5. Pilih tambahkan tindakanuntuk mengetik tindakan tertentu atau gunakan wildcard (*) untuk menentukan beberapa tindakan.

   Secara default, kebijakan yang Anda buat mengizinkan tindakan yang Anda pilih. Jika Anda memilih satu atau lebih tindakan yang mendukung izin level sumber daya ke sumber daya `datacatalog` di Athena, editor akan mencantumkan sumber daya `datacatalog` tersebut.

6. Pilih Sumber Daya untuk menentukan katalog data khusus bagi kebijakan Anda. Misalnya kebijakan katalog data JSON, lihat Contoh kebijakan Katalog Data.
7. Tentukan sumber daya `datacatalog` sebagai berikut:

```
arn:aws:athena:<region>:<user-account>:datacatalog/<datacatalog-name>
```

8. Pada halaman Tinjau kebijakan, ketik Nama dan Deskripsi (opsional) untuk kebijakan yang Anda buat. Tinjau ringkasan kebijakan untuk memastikan bahwa Anda memberikan izin yang dimaksud.
9. Pilih Buat kebijakan untuk menyimpan kebijakan baru Anda.
10. Lampirkan kebijakan berbasis identitas ini ke pengguna, grup, atau peran dan tentukan sumber daya `datacatalog` yang dapat mereka akses.

Untuk informasi selengkapnya, lihat topik berikut di Referensi Otorisasi Layanan dan Panduan Pengguna IAM:

- Tindakan, sumber daya, dan kunci kondisi untuk Amazon Athena
- Membuat kebijakan dengan editor visual
- Menambahkan dan menghapus kebijakan IAM
- Mengontrol akses ke sumber daya

Misalnya kebijakan katalog data JSON, lihat Contoh kebijakan Katalog Data.

Untuk informasi tentang AWS Glue izin dan izin AWS Glue crawler, lihat Menyiapkan izin IAM untuk AWS Glue dan Prasyarat Crawler di Panduan Pengembang.AWS Glue

Untuk daftar lengkap tindakan Amazon Athena, lihat nama tindakan API di Referensi API Amazon Athena.

Contoh kebijakan Katalog Data

Bagian ini mencakup contoh kebijakan yang dapat Anda gunakan untuk mengaktifkan berbagai tindakan pada katalog data.

Katalog data adalah sumber daya IAM yang dikelola oleh Athena. Oleh karena itu, jika kebijakan katalog data Anda menggunakan tindakan yang mengambil `datacatalog` sebagai input, Anda harus menentukan ARN katalog data sebagai berikut:

```
"Resource": [arn:aws:athena:<region>:<user-account>:datacatalog/<datacatalog-name>]
```

<datacatalog-name> adalah nama katalog data Anda. Misalnya, untuk katalog data bernama test_datacatalog, tentukan sebagai sumber daya sebagai berikut:

```
"Resource": ["arn:aws:athena:us-east-1:123456789012:datacatalog/test_datacatalog"]
```

Untuk daftar lengkap tindakan Amazon Athena, lihat nama tindakan API di Referensi API Amazon Athena. Untuk informasi selengkapnya tentang kebijakan IAM, lihat Membuat kebijakan dengan editor visual di Panduan Pengguna IAM. Untuk informasi selengkapnya tentang cara membuat kebijakan IAM untuk grup kerja, lihat Kebijakan IAM untuk mengakses katalog data.

- 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 Contoh kebijakan untuk akses penuh ke semua katalog data

Kebijakan berikut memungkinkan akses penuh ke semua sumber daya katalog data yang mungkin ada di akun. Kami menyarankan Anda menggunakan kebijakan ini untuk pengguna di akun Anda yang harus mengatur dan mengelola katalog data untuk semua pengguna lain.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "athena:*"
         ],
         "Resource":[
            "*"
         ]
      }
```

```
   ]
}
```

Example Contoh kebijakan untuk akses penuh ke Katalog Data tertentu

Kebijakan berikut memungkinkan akses penuh ke sumber daya katalog data spesifik tunggal, bernama `datacatalogA`. Anda dapat menggunakan kebijakan ini untuk pengguna dengan kontrol penuh atas katalog data tertentu.

```
{
   "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 Contoh kebijakan untuk menanyakan Katalog Data tertentu

Dalam kebijakan berikut, pengguna diizinkan untuk menjalankan kueri pada `datacatalogA` yang ditentukan. Pengguna tidak diizinkan untuk melakukan tugas manajemen untuk katalog data itu sendiri, seperti memperbarui atau menghapusnya.

```
{
   "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 Contoh kebijakan untuk operasi manajemen pada Katalog Data tertentu

Dalam kebijakan berikut, pengguna diperbolehkan untuk membuat, menghapus, mendapatkan detail, dan memperbarui katalog data `datacatalogA`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:CreateDataCatalog",
                "athena:GetDataCatalog",
                "athena:DeleteDataCatalog",
                "athena:UpdateDataCatalog"
            ],
            "Resource": [
                "arn:aws:athena:us-east-1:123456789012:datacatalog/datacatalogA"
            ]
        }
    ]
}
```

Example Contoh kebijakan untuk mencantumkan katalog data

Kebijakan berikut memungkinkan semua pengguna untuk mencantumkan semua katalog data:

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

```
                "athena:ListDataCatalogs"
            ],
            "Resource": "*"
        }
    ]
}
```

Example Contoh kebijakan untuk operasi metadata pada katalog data

Kebijakan berikut memungkinkan operasi metadata pada katalog data:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:GetDatabase",
                "athena:GetTableMetadata",
                "athena:ListDatabases",
                "athena:ListTableMetadata"
            ],
            "Resource": "*"
        }
    ]
}
```

Mengelola sumber data

Anda dapat menggunakanSumber Datahalaman konsol Athena untuk mengelola sumber data yang Anda buat.

Untuk melihat sumber data

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.

3. Di panel navigasi, pilih Sumber data.
4. Dari daftar sumber data, pilih nama sumber data yang ingin Anda lihat.

> **Note**
>
> Item dalamNama sumber datakolom sesuai dengan output dariListDataCatalogsAksi API danlist-data-catalogsPerintah CLI.

Untuk mengedit sumber data

1. PadaSumber datahalaman, lakukan salah satu dari berikut ini:
   - Pilih tombol di sebelah nama katalog, lalu pilihTindakan,Mengedit.
   - Pilih nama sumber data. Kemudian pada halaman detail, pilihTindakan,Mengedit.
2. PadaMengedithalaman, Anda dapat memilih fungsi Lambda yang berbeda untuk sumber data, mengubah deskripsi, atau menambahkan tag kustom. Untuk informasi selengkapnya tentang tanda, lihat Menandai sumber daya Athena.
3. Pilih Save (Simpan).
4. Untuk mengeditAwsDataCatalogsumber data, pilihAwsDataCataloglink untuk membuka halaman detailnya. Kemudian, pada halaman detail, pilih tautan keAWS Gluekonsol tempat Anda dapat mengedit katalog Anda.

Untuk berbagi sumber data

Untuk informasi tentang berbagi sumber data, kunjungi tautan berikut.

- Untuk sumber data non-Hive berbasis Lambda, lihatMengaktifkan kueri federasi lintas akun.
- UntukAWS Glue Data Catalogs, lihatAkses lintas akun ke katalog AWS Glue data.

Untuk menghapus sumber data

1. PadaSumber datahalaman, lakukan salah satu dari berikut ini:
   - Pilih tombol di sebelah nama katalog, lalu pilihTindakan,Hapus.
   - Pilih nama sumber data, dan kemudian, pada halaman detail, pilihTindakan,Hapus.

   > **Note**
   >
   > YangAwsDataCatalogadalah sumber data default di akun Anda dan tidak dapat dihapus.

   Anda diperingatkan bahwa ketika Anda menghapus sumber data, katalog data, tabel, dan tampilan yang sesuai akan dihapus dari editor kueri. Kueri tersimpan yang menggunakan sumber data tidak akan lagi berjalan di Athena.
2. Untuk mengonfirmasi penghapusan, ketik nama sumber data, lalu pilihHapus.

Menggunakan Amazon DataZone di Athena

Anda dapat menggunakan Amazon DataZone untuk berbagi, mencari, dan menemukan data dalam skala besar melintasi batas-batas organisasi. DataZone menyederhanakan pengalaman Anda di seluruh layanan AWS analitik seperti AthenaAWS Glue,, dan. AWS Lake Formation Misalnya, jika Anda memiliki petabyte data di sumber data yang berbeda, Anda dapat menggunakan Amazon DataZone untuk membangun pengelompokan orang, data, dan alat berbasis kasus penggunaan bisnis. Untuk informasi selengkapnya, lihat Apa yang dimaksud dengan Amazon DataZone? .

Di Athena, Anda dapat menggunakan editor kueri untuk mengakses dan menanyakan DataZone lingkungan. DataZone Lingkungan menentukan kombinasi DataZone proyek dan domain. Bila Anda menggunakan DataZone lingkungan dari konsol Athena, Anda mengambil peran IAM dari DataZone

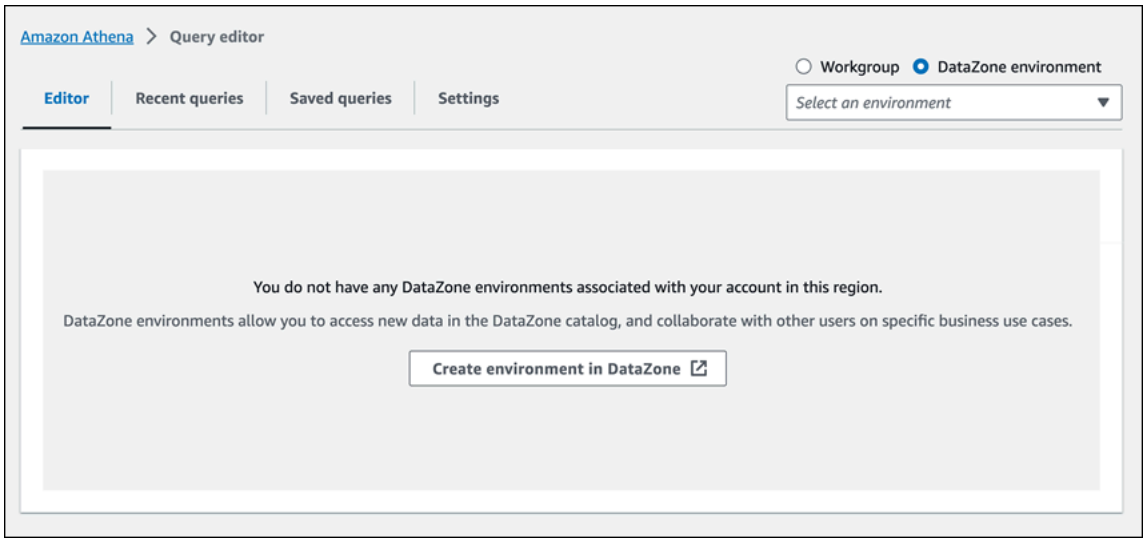
lingkungan, dan Anda hanya melihat database dan tabel milik lingkungan tersebut. Izin ditentukan oleh peran yang Anda tentukan. DataZone

Di Athena, Anda dapat menggunakan pemilih DataZone lingkungan pada halaman editor kueri untuk memilih lingkungan. DataZone

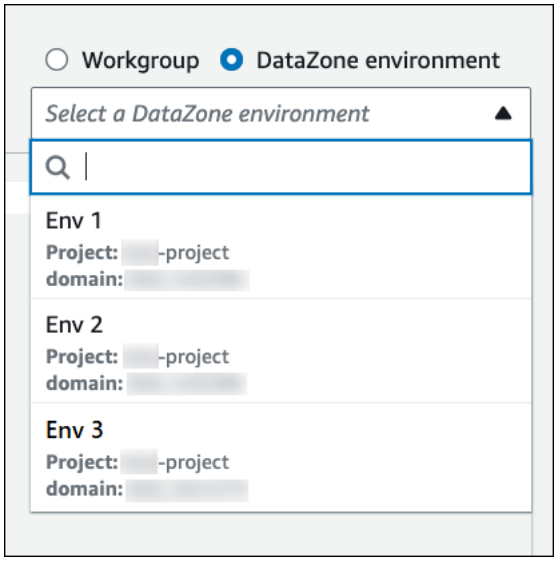
Untuk membuka DataZone lingkungan di Athena

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Di kanan atas konsol Athena, di sebelah Workgroup, pilih lingkungan. DataZone

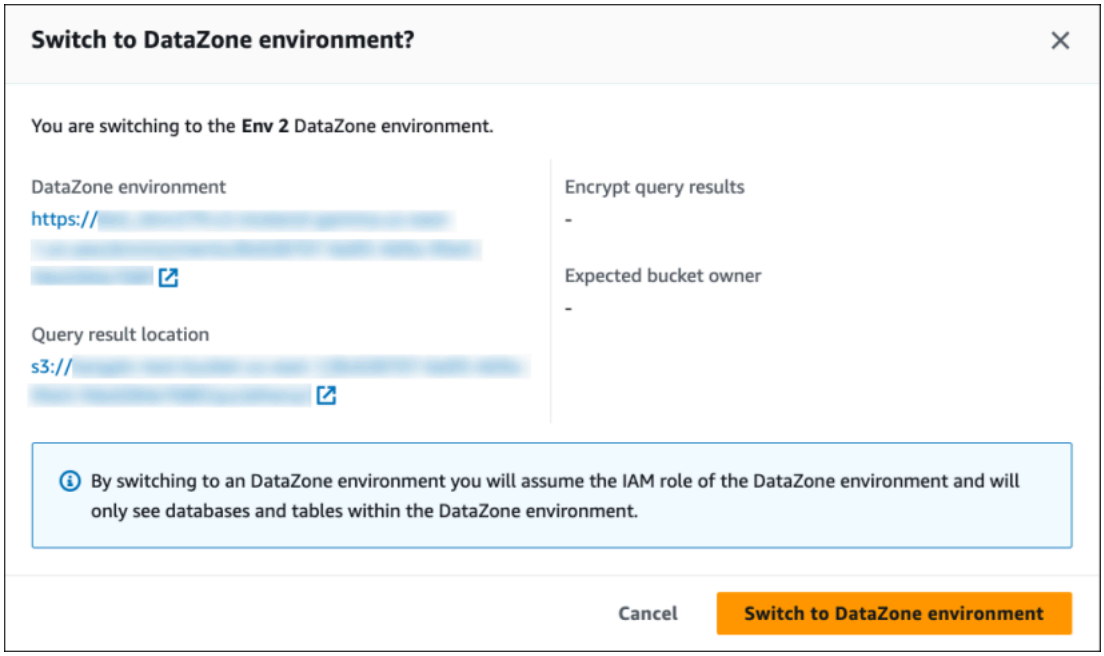
> **Note**
>
> Opsi DataZone lingkungan hadir hanya ketika Anda memiliki satu atau lebih domain yang tersedia di DataZone.



3. Gunakan pemilih DataZone lingkungan untuk memilih DataZone lingkungan.



4. Di kotak dialog Beralih ke DataZone lingkungan, verifikasi bahwa lingkungan adalah yang Anda inginkan, lalu pilih Beralih ke DataZone lingkungan.



Untuk informasi selengkapnya tentang memulai DataZone dan Athena, lihat tutorial Memulai di DataZone Panduan Pengguna Amazon.

Menghubungkan ke Amazon Athena dengan driver ODBC dan JDBC

Untuk mengeksplorasi dan memvisualisasikan data Anda dengan alat intelijen bisnis, unduh, instal, dan konfigurasikan driver ODBC (Open Basis data Connectivity) atau JDBC (Java Basis data Connectivity).

Topik

Lihat juga topik Pusat AWS Pengetahuan dan Blog AWS Big Data berikut:

- Bagaimana saya bisa menggunakan kredensi peran IAM saya atau beralih ke peran IAM lain saat menghubungkan ke Athena menggunakan driver JDBC?
- Menyiapkan kepercayaan antara ADFS dan AWS dan menggunakan kredenal Direktori Aktif untuk terhubung ke Amazon Athena dengan driver ODBC

Menghubungkan ke Amazon Athena dengan JDBC

Amazon Athena menawarkan dua driver JDBC, versi 2.x dan 3.x. Driver Athena JDBC 3.x adalah driver generasi baru yang menawarkan kinerja dan kompatibilitas yang lebih baik. Driver JDBC 3.x mendukung pembacaan hasil kueri langsung dari Amazon S3, yang meningkatkan kinerja aplikasi yang mengkonsumsi hasil kueri besar. Driver baru ini juga memiliki lebih sedikit dependensi pihak ketiga, yang membuat integrasi dengan alat BI dan aplikasi khusus lebih mudah. Dalam kebanyakan kasus, Anda dapat menggunakan driver baru tanpa atau sedikit perubahan pada konfigurasi yang ada.

- Untuk mengunduh driver JDBC 3.x, lihat. Athena JDBC 3.x driver
- Untuk mengunduh driver JDBC 2.x, lihat. Athena JDBC 2.x driver

Topik

Athena JDBC 3.x driver

Anda dapat menggunakan driver Athena JDBC untuk terhubung ke Amazon Athena dari banyak alat klien SQL pihak ketiga dan dari aplikasi khusus.

Persyaratan Sistem

- Lingkungan runtime Java 8 (atau lebih tinggi)
- Setidaknya 20 MB ruang disk yang tersedia

Pertimbangan dan batasan

Berikut ini adalah beberapa pertimbangan dan batasan untuk driver Athena JDBC 3.x.

- Logging — Driver 3.x menggunakan SLF4J, yang merupakan lapisan abstraksi yang memungkinkan penggunaan salah satu dari beberapa sistem logging saat runtime.
- Enkripsi — Saat menggunakan pengambil Amazon S3 dengan opsi `CSE_KMS` enkripsi, klien Amazon S3 tidak dapat mendekripsi hasil yang disimpan dalam bucket Amazon S3. Jika Anda memerlukan `CSE_KMS` enkripsi, Anda dapat terus menggunakan streaming fetcher. Support untuk `CSE_KMS` enkripsi dengan fetcher Amazon S3 direncanakan.

Unduhan driver JDBC 3.x

Bagian ini berisi informasi unduhan dan lisensi untuk driver JDBC 3.x.

> **Important**
>
> Saat Anda menggunakan driver JDBC 3.x, pastikan untuk mencatat persyaratan berikut:
>
> - Buka port 444 - Simpan port 444, yang digunakan Athena untuk mengalirkan hasil kueri, terbuka untuk lalu lintas keluar. Saat Anda menggunakan PrivateLink titik akhir untuk terhubung ke Athena, pastikan grup keamanan yang terpasang pada titik akhir terbuka untuk PrivateLink lalu lintas masuk di port 444.
> - athena: GetQueryResultsStream policy — Tambahkan tindakan `athena:GetQueryResultsStream` kebijakan ke kepala sekolah IAM yang menggunakan driver JDBC. Tindakan dasar ini tidak terdedah secara langsung dengan API. Ini hanya digunakan dengan driver ODBC dan JDBC sebagai bagian

dari dukungan hasil streaming. Untuk contoh kebijakan, lihat AWS kebijakan terkelola: AWSQuicksightAthenaAccess.

Untuk mengunduh driver Amazon Athena 3.x JDBC, kunjungi tautan berikut.

Toples uber driver JDBC

Unduhan berikut mengemas driver dan semua dependensinya dalam file yang sama `.jar`. Download ini biasanya digunakan untuk klien SQL pihak ketiga.

3.2.0 toples uber toples uber

Toples ramping driver JDBC

Download berikut adalah `.zip` file yang berisi lean `.jar` untuk driver dan `.jar` file terpisah untuk dependensi driver. Download ini biasanya digunakan untuk aplikasi kustom yang mungkin memiliki dependensi yang bertentangan dengan dependensi yang digunakan driver. Unduhan ini berguna jika Anda ingin memilih dependensi driver mana yang akan disertakan dengan lean jar, dan mana yang harus dikecualikan jika aplikasi khusus Anda sudah berisi satu atau lebih dari mereka.

3.2.0 toples tanpa lemak toples tanpa lemak

Lisensi

Tautan berikut berisi perjanjian lisensi untuk driver JDBC 3.x.

Lisensi

Topik

- Memulai dengan driver JDBC 3.x
- Parameter koneksi Amazon Athena JDBC 3.x
- Konfigurasi JDBC 3.x lainnya
- Catatan rilis Amazon Athena JDBC 3.x
- Versi sebelumnya dari driver Athena JDBC 3.x

Memulai dengan driver JDBC 3.x

Gunakan informasi di bagian ini untuk memulai dengan driver Amazon Athena JDBC 3.x.

Topik

Petunjuk Instalasi

Anda dapat menggunakan driver JDBC 3.x dalam aplikasi khusus atau dari klien SQL pihak ketiga.

Dalam aplikasi khusus

Unduh `.zip` file yang berisi jar driver dan dependensinya. Setiap dependensi memiliki `.jar` file sendiri. Tambahkan jar driver sebagai dependensi dalam aplikasi kustom Anda. Secara selektif tambahkan dependensi jar driver berdasarkan apakah Anda telah menambahkan dependensi tersebut ke aplikasi Anda dari sumber lain.

Di klien SQL pihak ketiga

Unduh file jar uber driver dan tambahkan ke klien SQL pihak ketiga mengikuti instruksi untuk klien itu.

Menjalankan pengemudi

Untuk menjalankan driver, Anda dapat menggunakan aplikasi khusus atau klien SQL pihak ketiga.

Dalam aplikasi khusus

Gunakan antarmuka JDBC untuk berinteraksi dengan driver JDBC dari suatu program. Kode berikut menunjukkan contoh aplikasi Java kustom.

```
public static void main(String args[]) throws SQLException {
    Properties connectionParameters = new Properties();
    connectionParameters.setProperty("Workgroup", "primary");
    connectionParameters.setProperty("Region", "us-east-2");
    connectionParameters.setProperty("Catalog", "AwsDataCatalog");
    connectionParameters.setProperty("Database","sampledatabase");
    connectionParameters.setProperty("OutputLocation","s3://DOC-EXAMPLE-BUCKET");
    connectionParameters.setProperty("CredentialsProvider","DefaultChain");
    String url = "jdbc:athena://";
```

```
    AthenaDriver driver = new AthenaDriver();
    Connection connection = driver.connect(url, connectionParameters);
    Statement statement = connection.createStatement();
    String query = "SELECT * from sample_table LIMIT 10";
    ResultSet resultSet = statement.executeQuery(query);
    printResults(resultSet); // A custom-defined method for iterating over a
                             // result set and printing its contents
}
```

Di klien SQL pihak ketiga

Ikuti dokumentasi untuk klien SQL yang Anda gunakan. Biasanya, Anda menggunakan antarmuka pengguna grafis klien SQL untuk memasukkan dan mengirimkan kueri, dan hasil kueri ditampilkan dalam antarmuka yang sama.

Mengkonfigurasi driver

Anda dapat menggunakan parameter koneksi untuk mengonfigurasi driver Amazon Athena JDBC. Untuk parameter koneksi yang didukung, lihatParameter koneksi Amazon Athena JDBC 3.x.

Dalam aplikasi khusus

Untuk mengatur parameter koneksi untuk driver JDBC dalam aplikasi khusus, lakukan salah satu hal berikut:

- Tambahkan nama parameter dan nilainya ke `Properties` objek. Saat Anda meneleponConnection#connect, berikan objek itu bersama dengan URL. Sebagai contoh, lihat contoh aplikasi Java diMenjalankan pengemudi.
- Dalam string koneksi (URL), gunakan format berikut untuk menambahkan nama parameter dan nilainya langsung setelah awalan protokol.

  ```
  <parameterName>=<parameterValue>;
  ```

  Gunakan titik koma di akhir setiap pasangan nama/parameter nilai parameter, dan tidak meninggalkan spasi putih setelah titik koma, seperti pada contoh berikut.

  ```
  String url = "jdbc:athena://WorkGroup=primary;Region=us-east-1;...;";AthenaDriver
   driver = new AthenaDriver();Connection connection = driver.connect(url, null);
  ```

> **Note**
>
> Jika parameter ditentukan baik dalam string koneksi dan `Properties` objek, nilai dalam string koneksi diutamakan. Menentukan parameter yang sama di kedua tempat tidak disarankan.

- Tambahkan nilai parameter sebagai argumen untuk metode`AthenaDataSource`, seperti dalam contoh berikut.

```
AthenaDataSource dataSource = new AthenaDataSource();
    dataSource.setWorkGroup("primary");
    dataSource.setRegion("us-east-2");
    ...
    Connection connection = dataSource.getConnection();
    ...
```

Di klien SQL pihak ketiga

Ikuti instruksi dari klien SQL yang Anda gunakan. Biasanya, klien menyediakan antarmuka pengguna grafis untuk memasukkan nama parameter dan nilainya.

Memutakhirkan dari driver Athena JDBC v2

Sebagian besar parameter koneksi JDBC versi 3 kompatibel dengan driver JDBC versi 2 (Simba). Ini berarti bahwa string koneksi versi 2 dapat digunakan kembali dengan versi 3 driver. Namun, beberapa parameter koneksi telah berubah. Perubahan ini dijelaskan di sini. Saat Anda meningkatkan ke driver JDBC versi 3, perbarui konfigurasi yang ada jika perlu.

Kelas pengemudi

Beberapa alat BI meminta Anda untuk memberikan kelas driver dari file driver JDBC. `.jar` Sebagian besar alat menemukan kelas ini secara otomatis. Nama kelas yang sepenuhnya memenuhi syarat dalam driver versi 3 adalah`com.amazon.athena.jdbc.AthenaDriver`. Di driver versi 2, kelasnya`com.simba.athena.jdbc.Driver`.

String koneksi

Driver versi 3 digunakan `jdbc:athena://` untuk protokol di awal URL string koneksi JDBC. Driver versi 3 juga mendukung protokol versi 2`jdbc:awsathena://`, tetapi penggunaan protokol versi 2

tidak digunakan lagi. Untuk menghindari perilaku yang tidak terdefinisi, versi 3 tidak menerima string koneksi yang dimulai dengan `jdbc:awsathena://` jika versi 2 (atau driver lain yang menerima string koneksi yang dimulai dengan`jdbc:awsathena://`) telah terdaftar di kelas. DriverManager

Penyedia kredensyal

Driver versi 2 menggunakan nama yang sepenuhnya memenuhi syarat untuk mengidentifikasi penyedia kredensyal yang berbeda (misalnya,. `com.simba.athena.amazonaws.auth.DefaultAWSCredentialsProviderChain` Driver versi 3 menggunakan nama yang lebih pendek (misalnya,`DefaultChain`). Nama-nama baru dijelaskan di bagian yang sesuai untuk setiap penyedia kredensyal.

Penyedia kredensi khusus yang ditulis untuk driver versi 2 perlu dimodifikasi untuk driver versi 3 untuk mengimplementasikan AwsCredentialsProviderantarmuka dari yang baru AWS SDK for Java alih-alih AWSCredentialsProviderantarmuka dari sebelumnya. AWS SDK for Java

Tidak `PropertiesFileCredentialsProvider` didukung di driver JDBC 3.x. Penyedia digunakan dalam driver JDBC 2.x tetapi milik versi SDK for AWS Java sebelumnya yang mendekati akhir dukungan. Untuk mencapai fungsionalitas yang sama di driver JDBC 3.x, gunakan penyedia sebagai AWS kredensial profil konfigurasi gantinya.

Tingkat log

Tabel berikut menunjukkan perbedaan `LogLevel` parameter dalam driver JDBC versi 2 dan versi 3.

Versi driver JDBC	Nama parameter	Jenis parameter	Nilai default	Kemungkinan nilai	Contoh string koneksi
v2	`LogLevel`	Opsional	0	0-6	`LogLevel=6;`
v3	`LogLevel`	Opsional	JEJAK	MATI, KESALAHAN, PERINGATAN, INFO, DEBUG, LACAK	`LogLevel=INFO;`

Pengambilan ID kueri

Dalam driver versi 2, Anda membuka `Statement` instance kecom.interfaces.core.IStatementQueryInfoProvider, antarmuka yang memiliki dua metode: #getPReparedQueryId dan#getQueryId. Anda dapat menggunakan metode ini untuk mendapatkan ID eksekusi kueri dari kueri yang telah berjalan.

Dalam driver versi 3, Anda membuka, `StatementPreparedStatement`, dan `ResultSet` instance ke antarmuka. `com.amazon.athena.jdbc.AthenaResultSet` Antarmuka memiliki satu metode:`#getQueryExecutionId`.

Parameter koneksi Amazon Athena JDBC 3.x

Parameter koneksi yang didukung dibagi di sini menjadi tiga bagian:Parameter koneksi dasar,Parameter koneksi lanjutan, danParameter koneksi otentikasi. Parameter koneksi lanjutan dan Parameter koneksi otentikasi memiliki subbagian yang mengelompokkan parameter terkait bersama-sama.

Topik

- Parameter koneksi dasar
- Parameter koneksi lanjutan
- Parameter koneksi otentikasi

Parameter koneksi dasar

Bagian berikut menjelaskan parameter koneksi dasar untuk driver JDBC 3.x.

Wilayah

Di Wilayah AWS mana kueri akan dijalankan. Untuk daftar wilayah, lihat titik akhir dan kuota Amazon Athena.

Nama parameter	Alias	Jenis parameter	Nilai default
Wilayah	AwsRegion (usang)	Wajib (tetapi jika tidak disediakan, akan dicari menggunakan) DefaultAwsRegionProviderChain	none

Katalog

Katalog yang berisi database dan tabel yang akan diakses dengan driver. Untuk informasi tentang katalog, lihat. DataCatalog

Nama parameter	Alias	Jenis parameter	Nilai default
Katalog	none	Opsional	AwsDataCatalog

Basis data

Database tempat kueri akan berjalan. Tabel yang tidak secara eksplisit memenuhi syarat dengan nama database diselesaikan ke database ini. Untuk informasi tentang database, lihat Database.

Nama parameter	Alias	Jenis parameter	Nilai default
Basis data	Skema	Opsional	default

Grup Kerja

Workgroup di mana query akan berjalan. Untuk informasi tentang kelompok kerja, lihat WorkGroup.

Nama parameter	Alias	Jenis parameter	Nilai default
WorkGroup	none	Opsional	utama

Lokasi keluaran

Lokasi di Amazon S3 tempat hasil kueri akan disimpan. Untuk informasi tentang lokasi keluaran, lihat ResultConfiguration.

Nama parameter	Alias	Jenis parameter	Nilai default
OutputLocation	S3 OutputLocation (usang)	Wajib (kecuali workgroup menentukan lokasi output)	none

Parameter koneksi lanjutan

Bagian berikut menjelaskan parameter koneksi lanjutan untuk driver JDBC 3.x.

Topik

Parameter enkripsi hasil

Perhatikan bidang berikut:

- AWS KMS Kunci harus `EncryptionOption` ditentukan kapan `SSE_KMS` atau`CSE_KMS`.
- AWS KMS Kunci tidak dapat `EncryptionOption` ditentukan kapan tidak ditentukan atau kapan `EncryptionOption``SSE_S3`.

Opsi enkripsi

Jenis enkripsi yang akan digunakan untuk hasil kueri karena disimpan di Amazon S3. Untuk informasi tentang enkripsi hasil kueri, lihat EncryptionConfigurationdi Referensi API Amazon Athena.

Nama parameter	Alias	Jenis parameter	Nilai default	Kemungkinan nilai
EncryptionOption	S3 OutputEnc Option (usang)	Opsional	none	SSE_S3, SSE_KMS, CSE_KMS

Kunci KMS

Kunci KMS ARN atau ID, `SSE_KMS` jika `CSE_KMS` atau dipilih sebagai opsi enkripsi. Untuk informasi selengkapnya, lihat EncryptionConfigurationdi Referensi API Amazon Athena.

Nama parameter	Alias	Jenis parameter	Nilai default
KmsKey	S3 OutputEnc KMSKey (usang)	Opsional	none

Parameter pengambilan hasil

Pengambil hasil

Pengambil yang akan digunakan untuk mengunduh hasil kueri.

Pengambil hasil default,`S3`, mengunduh hasil kueri langsung dari Amazon S3 tanpa menggunakan API Athena. Ini adalah opsi tercepat dalam banyak kasus. Opsi ini tidak tersedia jika hasil kueri Anda dienkripsi dengan `CSE_KMS` atau jika kebijakan yang memungkinkan pengguna mengakses hasil kueri hanya mengizinkan panggilan dari Athena menggunakan. `s3:CalledVia`

Nama parameter	Alias	Jenis parameter	Nilai default	Kemungkinan nilai
ResultFetcher	none	Opsional	S3	S3,, GetQueryResults GetQueryResultsStream

> **Note**
>
> Di driver JDBC 2.x, `UseResultsetStreaming = 1` pengaturan mengonfigurasi driver untuk menggunakan API streaming set hasil. Pada driver JDBC 3.x, pengaturan yang setara adalah. `ResultFetcher=GetQueryResultsStream`

Ambil ukuran

Nilai parameter ini digunakan sebagai minimum untuk buffer internal dan sebagai ukuran halaman target saat mengambil hasil. Nilai 0 (nol) berarti bahwa driver harus menggunakan defaultnya seperti yang dijelaskan di bawah ini. Nilai maksimumnya adalah 1.000.000.

Nama parameter	Alias	Jenis parameter	Nilai default
FetchSize	RowsToFetchPerBlock (usang)	Opsional	0

- `GetQueryResults`Pengambil akan selalu menggunakan ukuran halaman 1.000, yang merupakan nilai maksimum yang didukung oleh panggilan API. Ketika ukuran fetch lebih tinggi dari 1.000, beberapa panggilan API berturut-turut dilakukan untuk mengisi buffer di atas minimum.
- `GetQueryResultsStream`Pengambil akan menggunakan ukuran pengambilan yang dikonfigurasi sebagai ukuran halaman, atau 10.000 secara default.
- `S3`Pengambil akan menggunakan ukuran pengambilan yang dikonfigurasi sebagai ukuran halaman, atau 10.000 secara default.

Parameter penggunaan kembali hasil kueri

Aktifkan penggunaan kembali hasil

Menentukan apakah hasil sebelumnya untuk query yang sama dapat digunakan kembali ketika query dijalankan. Untuk informasi tentang penggunaan kembali hasil kueri, lihat ResultReuseByAgeConfiguration.

Nama parameter	Alias	Jenis parameter	Nilai default
EnableResultReuseByAge	none	Opsional	SALAH

Hasil penggunaan kembali usia maks

Usia maksimum, dalam hitungan menit, dari hasil kueri sebelumnya yang harus dipertimbangkan Athena untuk digunakan kembali. Untuk informasi tentang usia maksimal penggunaan kembali hasil, lihat ResultReuseByAgeConfiguration.

Nama parameter	Alias	Jenis parameter	Nilai default
MaxResultReuseAgeInMinutes	none	Opsional	60

Parameter polling eksekusi kueri

Interval polling eksekusi kueri minimum

Waktu minimum, dalam milidetik, untuk menunggu sebelum polling Athena untuk status eksekusi kueri.

Nama parameter	Alias	Jenis parameter	Nilai default
MinQueryExecutionPollingIntervalMillis	MinQueryExecutionPollingInterval (usang)	Opsional	100

Interval polling eksekusi kueri maksimum

Waktu maksimum, dalam milidetik, untuk menunggu sebelum polling Athena untuk status eksekusi kueri.

Nama parameter	Alias	Jenis parameter	Nilai default
MaxQueryExecutionPollingIntervalMillis	MaxQueryExecutionPollingInterval (usang)	Opsional	5000

Pengganda interval polling eksekusi kueri

Faktor untuk meningkatkan masa pemungutan suara. Secara default, polling akan dimulai dengan nilai untuk `MinQueryExecutionPollingIntervalMillis` dan dua kali lipat dengan setiap jajak pendapat hingga mencapai nilai untuk`MaxQueryExecutionPollingIntervalMillis`.

Nama parameter	Alias	Jenis parameter	Nilai default
QueryExecutionPollingIntervalMultiplier	none	Opsional	2

Parameter penggantian titik akhir

Pengesampingan titik akhir Athena

Titik akhir yang akan digunakan driver untuk melakukan panggilan API ke Athena.

Perhatikan bidang berikut:

- Jika `http://` protokol `https://` atau tidak ditentukan dalam URL yang disediakan, driver memasukkan awalan. `https://`
- Jika parameter ini tidak ditentukan, driver menggunakan endpoint default.

Nama parameter	Alias	Jenis parameter	Nilai default
AthenaEndpoint	EndpointOverride (usang)	Opsional	none

Pengesampingan titik akhir layanan streaming Athena

Titik akhir yang akan digunakan driver untuk mengunduh hasil kueri saat menggunakan layanan streaming Athena. Layanan streaming Athena tersedia di port 444.

Perhatikan bidang berikut:

- Jika `http://` protokol `https://` atau tidak ditentukan dalam URL yang disediakan, driver memasukkan awalan. `https://`
- Jika port tidak ditentukan dalam URL yang disediakan, driver memasukkan port layanan streaming 444.
- Jika `AthenaStreamingEndpoint` parameter tidak ditentukan, driver menggunakan `AthenaEndpoint` override. Jika baik penggantian `AthenaStreamingEndpoint` maupun `AthenaEndpoint` penggantian tidak ditentukan, driver menggunakan titik akhir streaming default.

Nama parameter	Alias	Jenis parameter	Nilai default
AthenaStreamingEndpoint	StreamingEndpointOverride (usang)	Opsional	none

LakeFormation penggantian titik akhir

Endpoint yang akan digunakan driver untuk layanan Lake Formation saat menggunakan AWS Lake Formation AssumeDecoratedRoleWithSAMP API untuk mengambil kredensi sementara. Jika parameter ini tidak ditentukan, driver menggunakan endpoint Lake Formation default.

Perhatikan bidang berikut:

- Jika `http://` protokol `https://` atau tidak ditentukan dalam URL yang disediakan, driver memasukkan awalan. `https://`

Nama parameter	Alias	Jenis parameter	Nilai default
LakeFormationEndpoint	LfEndpointOverride (usang)	Opsional	none

Pengesampingan titik akhir S3

Titik akhir yang akan digunakan driver untuk mengunduh hasil kueri saat menggunakan pengambil Amazon S3. Jika parameter ini tidak ditentukan, driver menggunakan endpoint Amazon S3 default.

Perhatikan bidang berikut:

- Jika `http://` protokol `https://` atau tidak ditentukan dalam URL yang disediakan, driver memasukkan awalan. `https://`

Nama parameter	Alias	Jenis parameter	Nilai default
Titik Akhir S3	Tidak ada	Opsional	none

Pengesampingan titik akhir STS

Titik akhir yang akan digunakan driver untuk AWS STS layanan saat menggunakan AWS STS AssumeRoleWithSAMP API untuk mengambil kredensi sementara. Jika parameter ini tidak ditentukan, driver menggunakan AWS STS endpoint default.

Perhatikan bidang berikut:

- Jika `http://` protokol `https://` atau tidak ditentukan dalam URL yang disediakan, driver memasukkan awalan. `https://`

Nama parameter	Alias	Jenis parameter	Nilai default
StsEndpoint	StsEndpointOverrid e(usang)	Opsional	none

Parameter konfigurasi proxy

Tuan rumah proxy

URL dari host proxy. Gunakan parameter ini jika Anda memerlukan permintaan Athena untuk melalui proxy.

> **Note**
>
> Pastikan untuk menyertakan protokol `https://` atau `http://` di awal URL untuk`ProxyHost`.

Nama parameter	Alias	Jenis parameter	Nilai default
ProxyHost	none	Opsional	none

Port proxy

Port yang akan digunakan pada host proxy. Gunakan parameter ini jika Anda memerlukan permintaan Athena untuk melalui proxy.

Nama parameter	Alias	Jenis parameter	Nilai default
ProxyPort	none	Opsional	none

Nama pengguna proxy

Nama pengguna untuk mengautentikasi pada server proxy. Gunakan parameter ini jika Anda memerlukan permintaan Athena untuk melalui proxy.

Nama parameter	Alias	Jenis parameter	Nilai default
ProxyUsername	ProxYuID (usang)	Opsional	none

Kata sandi proxy

Kata sandi untuk mengautentikasi pada server proxy. Gunakan parameter ini jika Anda memerlukan permintaan Athena untuk melalui proxy.

Nama parameter	Alias	Jenis parameter	Nilai default
ProxyPassword	ProxyPwd (usang)	Opsional	none

Host bebas proxy

Satu set nama host yang terhubung ke driver tanpa menggunakan proxy saat proxy diaktifkan (yaitu, ketika parameter `ProxyHost` dan `ProxyPort` koneksi diatur). Host harus dipisahkan oleh karakter pipe (`|`) (misalnya,`host1.com|host2.com`).

Nama parameter	Alias	Jenis parameter	Nilai default
ProxyExemptHosts	NonProxyHosts	Opsional	none

Proxy diaktifkan untuk penyedia identitas

Menentukan apakah proxy harus digunakan ketika driver terhubung ke penyedia identitas.

Nama parameter	Alias	Jenis parameter	Nilai default
ProxyEnabledForIdP	UseProxyForIdP	Opsional	SALAH

Parameter pencatatan

Bagian ini menjelaskan parameter yang terkait dengan logging.

Tingkat log

Menentukan tingkat untuk logging driver. Tidak ada yang dicatat kecuali `LogPath` parameternya juga disetel.

> **Note**
>
> Kami merekomendasikan pengaturan hanya `LogPath` parameter kecuali Anda memiliki persyaratan khusus. Pengaturan hanya `LogPath` parameter memungkinkan logging dan menggunakan tingkat `TRACE` log default. Level `TRACE` log menyediakan logging paling rinci.

Nama parameter	Alias	Jenis parameter	Nilai default	Kemungkinan nilai
LogLevel	none	Opsional	JEJAK	MATI, KESALAHAN, PERINGATAN, INFO, DEBUG, JEJAK

Jalur log

Jalur ke direktori di komputer yang menjalankan driver tempat log driver akan disimpan. File log dengan nama unik akan dibuat dalam direktori yang ditentukan. Jika disetel, aktifkan pencatatan driver.

Nama parameter	Alias	Jenis parameter	Nilai default
LogPath	none	Opsional	none

Nama aplikasi

Nama aplikasi yang menggunakan driver. Jika nilai untuk parameter ini ditentukan, nilai tersebut disertakan dalam string agen pengguna dari panggilan API yang dilakukan driver ke Athena.

> **Note**
>
> Anda juga dapat mengatur nama aplikasi dengan memanggil `setApplicationName` `DataSource` objek.

Nama parameter	Alias	Jenis parameter	Nilai default
ApplicationName	none	Opsional	none

Uji Koneksi

Jika disetel keTRUE, driver melakukan tes koneksi setiap kali koneksi JDBC dibuat, bahkan jika kueri tidak dijalankan pada koneksi.

Nama parameter	Alias	Jenis parameter	Nilai default
ConnectionTest	none	Opsional	BETUL

> **Note**
>
> Tes koneksi mengirimkan `SELECT 1` kueri ke Athena untuk memverifikasi bahwa koneksi telah dikonfigurasi dengan benar. Ini berarti bahwa dua file akan disimpan di Amazon S3 (set hasil dan metadata), dan biaya tambahan dapat berlaku sesuai dengan kebijakan harga Amazon Athena.

Jumlah percobaan

Jumlah maksimum kali pengemudi harus mengirim ulang permintaan yang dapat diambil ke Athena.

Nama parameter	Alias	Jenis parameter	Nilai default
NumRetries	MaxErrorRetry (usang)	Opsional	none

Parameter koneksi otentikasi

Driver Athena JDBC 3.x mendukung beberapa metode otentikasi. Parameter koneksi yang diperlukan bergantung pada metode otentikasi yang Anda gunakan.

Topik

- Kredensial IAM
- Kredensial default
- AWS kredensial profil konfigurasi
- Kredensial profil instans
- Kredensial kustom
- Kredensi JWT
- Kredensi Azure AD
- Kredensi Okta
- Kredensi ping
- Kredensi AD FS
- Kredensi Browser Azure AD
- Kredensi SAM Browser

Kredensial IAM

Anda dapat menggunakan kredensil IAM Anda dengan driver JDBC untuk terhubung ke Amazon Athena dengan mengatur parameter koneksi berikut.

Pengguna

ID kunci AWS akses Anda. Untuk informasi tentang kunci akses, lihat kredensil AWS keamanan di Panduan Pengguna IAM.

Nama parameter	Alias	Jenis parameter	Nilai default
Pengguna	AccessKeyId	Wajib	none

Kata sandi

ID kunci AWS rahasiamu. Untuk informasi tentang kunci akses, lihat kredensil AWS keamanan di Panduan Pengguna IAM.

Nama parameter	Alias	Jenis parameter	Nilai default
Kata sandi	SecretAccessKey	Opsional	none

Token sesi

Jika Anda menggunakan AWS kredensil sementara, Anda harus menentukan token sesi. Untuk informasi tentang kredensil sementara, lihat Kredensil keamanan sementara di IAM di Panduan Pengguna IAM.

Nama parameter	Alias	Jenis parameter	Nilai default
SessionToken	none	Opsional	none

Kredensial default

Anda dapat menggunakan kredenal default yang Anda konfigurasikan pada sistem klien Anda untuk terhubung ke Amazon Athena dengan mengatur parameter koneksi berikut. Untuk informasi tentang menggunakan kredensil default, lihat Menggunakan Rantai Penyedia Kredenal Default di Panduan Pengembang AWS SDK for Java .

Penyedia kredensil

Penyedia kredensi yang akan digunakan untuk mengautentikasi permintaan ke. AWS Tetapkan nilai parameter ini ke`DefaultChain`.

Nama parameter	Alias	Jenis parameter	Nilai default	Nilai untuk digunakan
CredentialsProvider	AWSCredentialsProviderClass(usang)	Wajib	none	`DefaultChain`

AWS kredensial profil konfigurasi

Anda dapat menggunakan kredensil yang disimpan dalam profil AWS konfigurasi dengan mengatur parameter koneksi berikut. AWS profil konfigurasi biasanya disimpan dalam file di `~/.aws` direktori). Untuk informasi tentang profil AWS konfigurasi, lihat Menggunakan profil di Panduan AWS SDK for Java Pengembang.

Penyedia kredensil

Penyedia kredensi yang akan digunakan untuk mengautentikasi permintaan ke. AWS Tetapkan nilai parameter ini ke`ProfileCredentials`.

Nama parameter	Alias	Jenis parameter	Nilai default	Nilai untuk digunakan
CredentialsProvider	AWSCredentialsProviderClass(usang)	Wajib	none	`ProfileCredentials`

Nama profil

Nama profil AWS konfigurasi yang kredensialnya harus digunakan untuk mengautentikasi permintaan ke Athena.

Nama parameter	Alias	Jenis parameter	Nilai default
ProfileName	none	Wajib	none

> **Note**
>
> Nama profil juga dapat ditentukan sebagai nilai `CredentialsProviderArguments` parameter, meskipun penggunaan ini tidak digunakan lagi.

Kredensial profil instans

Jenis otentikasi ini digunakan pada instans Amazon EC2. Profil instans adalah profil yang dilampirkan ke instans Amazon EC2. Menggunakan penyedia kredensial profil instans mendelegasikan pengelolaan AWS kredensional ke Layanan Metadata Instans Amazon EC2. Ini menghilangkan kebutuhan pengembang untuk menyimpan kredensil secara permanen di instans Amazon EC2 atau khawatir tentang memutar atau mengelola kredensil sementara.

Penyedia kredensil

Penyedia kredensi yang akan digunakan untuk mengautentikasi permintaan ke. AWS Tetapkan nilai parameter ini ke`InstanceProfile`.

Nama parameter	Alias	Jenis parameter	Nilai default	Nilai untuk digunakan
CredentialsProvider	AWSCredentialsProviderClass(usang)	Wajib	none	`InstanceProfile`

Kredensial kustom

Anda dapat menggunakan jenis otentikasi ini untuk memberikan kredensil Anda sendiri dengan menggunakan kelas Java yang mengimplementasikan antarmuka. AwsCredentialsProvider

Penyedia kredensil

Penyedia kredensi yang akan digunakan untuk mengautentikasi permintaan ke. AWS Tetapkan nilai parameter ini ke nama kelas yang sepenuhnya memenuhi syarat dari kelas kustom yang mengimplementasikan AwsCredentialsProviderantarmuka. Saat runtime, kelas itu harus berada di jalur kelas Java dari aplikasi yang menggunakan driver JDBC.

Nama parameter	Alias	Jenis parameter	Nilai default	Nilai untuk digunakan
CredentialsProvider	AWSCredentialsProviderClass (usang)	Wajib	none	Nama kelas yang sepenuhnya memenuhi syarat dari implementasi kustom `AwsCredentialsProvider`

Argumen penyedia kredensial

Daftar argumen string yang dipisahkan koma untuk konstruktor penyedia kredensial kustom.

Nama parameter	Alias	Jenis parameter	Nilai default
CredentialsProviderArguments	AwsCredentialsProviderArguments (usang)	Opsional	none

Kredensi JWT

Dengan jenis otentikasi ini, Anda dapat menggunakan token web JSON (JWT) yang diperoleh dari penyedia identitas eksternal sebagai parameter koneksi untuk mengautentikasi dengan Athena. Penyedia kredensi eksternal harus sudah difederasi dengan. AWS

Penyedia kredensil

Penyedia kredensi yang akan digunakan untuk mengautentikasi permintaan ke. AWS Tetapkan nilai parameter ini ke`JWT`.

Nama parameter	Alias	Jenis parameter	Nilai default	Nilai untuk digunakan
CredentialsProvider	AWSCredentialsProv	Wajib	none	`JWT`

Nama parameter	Alias	Jenis parameter	Nilai default	Nilai untuk digunakan
	iderClass (usang)			

Token identitas web JWT

Token JWT diperoleh dari penyedia identitas federasi eksternal. Token ini akan digunakan untuk mengautentikasi dengan Athena.

Nama parameter	Alias	Jenis parameter	Nilai default
JwtWebIdentityToken	web_identity_token (usang)	Wajib	none

Peran JWT ARN

Amazon Resource Name (ARN) dari peran yang diambil. Untuk informasi tentang mengasumsikan peran, lihat AssumeRoledi Referensi AWS Security Token Service API.

Nama parameter	Alias	Jenis parameter	Nilai default
JwtRoleArn	role_arn (usang)	Wajib	none

Nama sesi peran JWT

Nama sesi saat Anda menggunakan kredensi JWT untuk otentikasi. Nama dapat berupa nama apa saja yang Anda pilih.

Nama parameter	Alias	Jenis parameter	Nilai default
JwtRoleSessionName	role_session_name (usang)	Wajib	none

Kredensi Azure AD

Mekanisme otentikasi berbasis SALL yang memungkinkan otentikasi ke Athena menggunakan penyedia identitas Azure AD. Metode ini mengasumsikan bahwa federasi telah dibentuk antara Athena dan Azure AD.

> **Note**
>
> Beberapa nama parameter di bagian ini memiliki alias. Alias adalah setara fungsional dari nama parameter dan telah disediakan untuk kompatibilitas mundur dengan driver JDBC 2.x. Karena nama parameter telah diperbaiki untuk mengikuti konvensi penamaan yang lebih jelas dan lebih konsisten, sebaiknya Anda menggunakannya alih-alih alias, yang telah usang.

Penyedia kredensil

Penyedia kredensi yang akan digunakan untuk mengautentikasi permintaan ke. AWS Tetapkan nilai parameter ini ke`AzureAD`.

Nama parameter	Alias	Jenis parameter	Nilai default	Nilai untuk digunakan
CredentialsProvider	AWSCredentialsProviderClass (usang)	Wajib	none	`AzureAD`

Pengguna

Alamat email pengguna Azure AD yang akan digunakan untuk otentikasi dengan Azure AD.

Nama parameter	Alias	Jenis parameter	Nilai default
Pengguna	UID (usang)	Wajib	none

Kata sandi

Kata sandi untuk pengguna Azure AD.

Nama parameter	Alias	Jenis parameter	Nilai default
Kata sandi	PWD (usang)	Wajib	none

ID penyewa Azure AD

ID penyewa aplikasi Azure AD Anda.

Nama parameter	Alias	Jenis parameter	Nilai default
AzureAdTenantId	tenant_id (usang)	Wajib	none

ID klien Azure AD

ID klien aplikasi Azure AD Anda.

Nama parameter	Alias	Jenis parameter	Nilai default
AzureAdClientId	client_id (usang)	Wajib	none

Rahasia klien Azure AD

Rahasia klien aplikasi Azure AD Anda.

Nama parameter	Alias	Jenis parameter	Nilai default
AzureAdClientSecret	client_secret (usang)	Wajib	none

Peran yang disukai

Amazon Resource Name (ARN) dari peran yang diambil. Untuk informasi tentang peran ARN, lihat AssumeRoledi Referensi AWS Security Token Service API.

Nama parameter	Alias	Jenis parameter	Nilai default
PreferredRole	preferred_role (tidak digunakan lagi)	Opsional	none

Durasi sesi peran

Durasi, dalam hitungan detik, dari sesi peran. Untuk informasi selengkapnya, lihat AssumeRoledi Referensi AWS Security Token Service API.

Nama parameter	Alias	Jenis parameter	Nilai default
RoleSessionDuration	Durasi (usang)	Opsional	3600

Lake Formation diaktifkan

Menentukan apakah akan menggunakan tindakan API Lake Formation AssumeDecoratedRoleWithSAMP untuk mengambil kredenal IAM sementara, bukan tindakan SAMP API. AssumeRoleWith AWS STS

Nama parameter	Alias	Jenis parameter	Nilai default
LakeFormationEnabled	none	Opsional	SALAH

Kredensi Okta

Mekanisme otentikasi berbasis SALL yang memungkinkan otentikasi ke Athena menggunakan penyedia identitas Okta. Metode ini mengasumsikan bahwa federasi telah dibentuk antara Athena dan Okta.

Penyedia kredensil

Penyedia kredensi yang akan digunakan untuk mengautentikasi permintaan ke. AWS Tetapkan nilai parameter ini ke`Okta`.

Nama parameter	Alias	Jenis parameter	Nilai default	Nilai untuk digunakan
CredentialsProvider	AWSCredentialsProviderClass (usang)	Wajib	none	`Okta`

Pengguna

Alamat email pengguna Okta untuk digunakan untuk otentikasi dengan Okta.

Nama parameter	Alias	Jenis parameter	Nilai default
Pengguna	UID (usang)	Wajib	none

Kata sandi

Kata sandi untuk pengguna Okta.

Nama parameter	Alias	Jenis parameter	Nilai default
Kata sandi	PWD (usang)	Wajib	none

Nama host Okta

URL untuk organisasi Okta Anda. Anda dapat mengekstrak `idp_host` parameter dari URL Embed Link di aplikasi Okta Anda. Untuk langkah, lihat Ambil informasi konfigurasi ODBC dari Okta. Segmen pertama setelah`https://`, hingga dan termasuk`okta.com`, adalah host idP Anda (misalnya, `trial-1234567.okta.com` untuk URL yang dimulai dengan`https://trial-1234567.okta.com`).

Nama parameter	Alias	Jenis parameter	Nilai default
OktaHostName	IDP_host (usang)	Wajib	none

ID aplikasi Okta

Pengenal dua bagian untuk aplikasi Anda. Anda dapat mengekstrak ID aplikasi dari URL Embed Link di aplikasi Okta Anda. Untuk langkah, lihat Ambil informasi konfigurasi ODBC dari Okta. ID aplikasi adalah dua segmen terakhir dari URL, termasuk garis miring ke depan di tengah. Segmen adalah dua string 20 karakter dengan campuran angka dan huruf besar dan kecil (misalnya,). `Abc1de2fghi3J45kL678/abc1defghij2klmNo3p4`

Nama parameter	Alias	Jenis parameter	Nilai default
OktaAppId	App_id (usang)	Wajib	none

Nama aplikasi Okta

Nama aplikasi Okta Anda.

Nama parameter	Alias	Jenis parameter	Nilai default
OktaAppName	App_Name (usang)	Wajib	none

Jenis MFA Okta

Jika Anda telah mengatur Okta untuk memerlukan otentikasi multi-faktor (MFA), Anda perlu menentukan jenis MFA Okta dan parameter tambahan tergantung pada faktor kedua yang ingin Anda gunakan.

Jenis Okta MFA adalah jenis faktor otentikasi kedua (setelah kata sandi) yang digunakan untuk mengautentikasi dengan Okta. Faktor kedua yang didukung termasuk pemberitahuan push yang dikirimkan melalui aplikasi Okta Verify dan kata sandi satu kali sementara (TOTP) yang dihasilkan oleh Okta Verify, Google Authenticator, atau dikirim melalui SMS. Kebijakan keamanan organisasi individu menentukan apakah MFA diperlukan atau tidak untuk login pengguna.

Nama parameter	Alias	Jenis parameter	Nilai default	Kemungkinan nilai
OktaMfaType	okta_mfa_type (usang)	Diperlukan, jika Okta diatur untuk	none	`oktaverif ywithpush`

Nama parameter	Alias	Jenis parameter	Nilai default	Kemungkinan nilai
		membutuhkan MFA		, `oktaverifywithtotp` , `googleauthenticator` , `smsauthentication`

Nomor telepon Okta

Nomor telepon tempat Okta akan mengirim kata sandi satu kali sementara menggunakan SMS ketika jenis `smsauthentication` MFA dipilih. Nomor telepon harus berupa nomor telepon AS atau Kanada.

Nama parameter	Alias	Jenis parameter	Nilai default
OktaPhoneNumber	okta_phone_number (usang)	Diperlukan, OktaMfaType jika `smsauthentication`	none

Okta MFA waktu tunggu

Durasi, dalam hitungan detik, untuk menunggu pengguna mengakui pemberitahuan push dari Okta sebelum pengemudi melempar pengecualian batas waktu.

Nama parameter	Alias	Jenis parameter	Nilai default
OktaMfaWaitTime	okta_mfa_wait_time (usang)	Opsional	60

Peran yang disukai

Amazon Resource Name (ARN) dari peran yang diambil. Untuk informasi tentang peran ARN, lihat AssumeRoledi Referensi AWS Security Token Service API.

Nama parameter	Alias	Jenis parameter	Nilai default
PreferredRole	preferred_role (tidak digunakan lagi)	Opsional	none

Durasi sesi peran

Durasi, dalam hitungan detik, dari sesi peran. Untuk informasi selengkapnya, lihat AssumeRoledi Referensi AWS Security Token Service API.

Nama parameter	Alias	Jenis parameter	Nilai default
RoleSessionDuration	Durasi (usang)	Opsional	3600

Lake Formation diaktifkan

Menentukan apakah akan menggunakan tindakan API Lake Formation AssumeDecoratedRoleWithSAMP untuk mengambil kredenal IAM sementara, bukan tindakan SAMP API. AssumeRoleWith AWS STS

Nama parameter	Alias	Jenis parameter	Nilai default
LakeFormationEnabled	none	Opsional	SALAH

Kredensi ping

Mekanisme otentikasi berbasis SAML yang memungkinkan otentikasi ke Athena menggunakan penyedia identitas Federasi Ping. Metode ini mengasumsikan bahwa federasi telah dibentuk antara Athena dan Federasi Ping.

Penyedia kredensil

Penyedia kredensi yang akan digunakan untuk mengautentikasi permintaan ke. AWS Tetapkan nilai parameter ini ke`Ping`.

Nama parameter	Alias	Jenis parameter	Nilai default	Nilai untuk digunakan
CredentialsProvider	AWSCredentialsProviderClass (usang)	Wajib	none	`Ping`

Pengguna

Alamat email pengguna Ping Federate untuk digunakan untuk otentikasi dengan Ping Federate.

Nama parameter	Alias	Jenis parameter	Nilai default
Pengguna	UID (usang)	Wajib	none

Kata sandi

Kata sandi untuk pengguna Federasi Ping.

Nama parameter	Alias	Jenis parameter	Nilai default
Kata sandi	PWD (usang)	Wajib	none

PingHostName

Alamat untuk server Ping Anda. Untuk menemukan alamat Anda, kunjungi URL berikut dan lihat bidang SSO Application Endpoint.

```
https://your-pf-host-#:9999/pingfederate/your-pf-app#/spConnections
```

Nama parameter	Alias	Jenis parameter	Nilai default
PingHostName	IDP_host (usang)	Wajib	none

PingPortNumber

Nomor port yang digunakan untuk terhubung ke host iDP Anda.

Nama parameter	Alias	Jenis parameter	Nilai default
PingPortNumber	IDP_port (usang)	Wajib	none

PingPartnerSpId

Alamat penyedia layanan. Untuk menemukan alamat penyedia layanan, kunjungi URL berikut dan lihat bidang Titik Akhir Aplikasi SSO.

```
https://your-pf-host-#:9999/pingfederate/your-pf-app#/spConnections
```

Nama parameter	Alias	Jenis parameter	Nilai default
PingPartnerSpId	Partner_SPID (usang)	Wajib	none

Peran yang disukai

Amazon Resource Name (ARN) dari peran yang diambil. Untuk informasi tentang peran ARN, lihat AssumeRoledi Referensi AWS Security Token Service API.

Nama parameter	Alias	Jenis parameter	Nilai default
PreferredRole	preferred_role (tidak digunakan lagi)	Opsional	none

Durasi sesi peran

Durasi, dalam hitungan detik, dari sesi peran. Untuk informasi selengkapnya, lihat AssumeRoledi Referensi AWS Security Token Service API.

Nama parameter	Alias	Jenis parameter	Nilai default
RoleSessionDuration	Durasi (usang)	Opsional	3600

Lake Formation diaktifkan

Menentukan apakah akan menggunakan tindakan API Lake Formation AssumeDecoratedRoleWithSAMP untuk mengambil kredenal IAM sementara, bukan tindakan SAMP API. AssumeRoleWith AWS STS

Nama parameter	Alias	Jenis parameter	Nilai default
LakeFormationEnabled	none	Opsional	SALAH

Kredensi AD FS

Mekanisme otentikasi berbasis SAMP yang memungkinkan otentikasi ke Athena menggunakan Microsoft Active Directory Federation Services (AD FS). Metode ini mengasumsikan bahwa pengguna telah mengatur federasi antara Athena dan AD FS.

Penyedia kredensil

Penyedia kredensi yang akan digunakan untuk mengautentikasi permintaan ke. AWS Tetapkan nilai parameter ini keADFS.

Nama parameter	Alias	Jenis parameter	Nilai default	Nilai untuk digunakan
CredentialsProvider	AWSCredentialsProviderClass (usang)	Wajib	none	ADFS

Pengguna

Alamat email pengguna AD FS yang akan digunakan untuk otentikasi dengan AD FS.

Nama parameter	Alias	Jenis parameter	Nilai default
Pengguna	UID (usang)	Diperlukan untuk otentikasi berbasis formulir. Opsional untuk Otentikasi Terpadu Windows.	none

Kata sandi

Kata sandi untuk pengguna AD FS.

Nama parameter	Alias	Jenis parameter	Nilai default
Kata sandi	PWD (usang)	Diperlukan untuk otentikasi berbasis formulir. Opsional untuk Otentikasi Terpadu Windows.	none

Nama host ADFS

Alamat untuk server AD FS Anda.

Nama parameter	Alias	Jenis parameter	Nilai default
AdfsHostName	IDP_host (usang)	Wajib	none

Nomor port ADFS

Nomor port yang digunakan untuk terhubung ke server AD FS Anda.

Nama parameter	Alias	Jenis parameter	Nilai default
AdfsPortNumber	IDP_port (usang)	Wajib	none

Partai mengandalkan ADFS

Partai mengandalkan tepercaya. Gunakan parameter ini untuk mengganti URL titik akhir pihak yang mengandalkan AD FS.

Nama parameter	Alias	Jenis parameter	Nilai default
AdfsRelyingParty	LoginToRP (usang)	Opsional	`urn:amazon:webservices`

ADFS WIA diaktifkan

Boolean. Gunakan parameter ini untuk mengaktifkan Windows Integrated Authentication (WIA) dengan AD FS.

Nama parameter	Alias	Jenis parameter	Nilai default
AdfsWiaEnabled	none	Opsional	`FALSE`

Peran yang disukai

Amazon Resource Name (ARN) dari peran yang diambil. Untuk informasi tentang peran ARN, lihat `AssumeRole`di Referensi AWS Security Token Service API.

Nama parameter	Alias	Jenis parameter	Nilai default
PreferredRole	preferred_role (tidak digunakan lagi)	Opsional	none

Durasi sesi peran

Durasi, dalam hitungan detik, dari sesi peran. Untuk informasi selengkapnya, lihat `AssumeRole` di dalam Referensi API AWS Security Token Service .

Nama parameter	Alias	Jenis parameter	Nilai default
RoleSessionDuration	Durasi (usang)	Opsional	`3600`

Lake Formation diaktifkan

Menentukan apakah akan menggunakan aksi `AssumeDecoratedRoleWithSAML`Lake Formation API untuk mengambil kredenal IAM sementara, bukan tindakan API. `AssumeRoleWithSAML` AWS STS

Nama parameter	Alias	Jenis parameter	Nilai default
LakeFormationEnabled	`none`	Opsional	SALAH

Kredensi Browser Azure AD

Browser Azure AD adalah mekanisme otentikasi berbasis SAMP yang bekerja dengan penyedia identitas Azure AD dan mendukung otentikasi multi-faktor. Berbeda dengan mekanisme otentikasi Azure AD standar, mekanisme ini tidak memerlukan nama pengguna, kata sandi, atau rahasia klien dalam parameter koneksi. Seperti mekanisme otentikasi Azure AD standar, Browser Azure AD juga mengasumsikan pengguna telah mengatur federasi antara Athena dan Azure AD.

Penyedia kredensil

Penyedia kredensi yang akan digunakan untuk mengautentikasi permintaan ke. AWS Tetapkan nilai parameter ini ke`BrowserAzureAD`.

Nama parameter	Alias	Jenis parameter	Nilai default	Nilai untuk digunakan
CredentialsProvider	AWSCredentialsProv	Wajib	none	`BrowserAzureAD`

Nama parameter	Alias	Jenis parameter	Nilai default	Nilai untuk digunakan
	iderClass (usang)			

ID penyewa Azure AD

ID penyewa aplikasi Azure AD Anda

Nama parameter	Alias	Jenis parameter	Nilai default
AzureAdTenantId	tenant_id (usang)	Wajib	none

ID klien Azure AD

ID klien aplikasi Azure AD Anda

Nama parameter	Alias	Jenis parameter	Nilai default
AzureAdClientId	client_id (usang)	Wajib	none

Batas waktu respons penyedia identitas

Durasi, dalam hitungan detik, sebelum pengemudi berhenti menunggu respons SAFL dari Azure AD.

Nama parameter	Alias	Jenis parameter	Nilai default
IdpResponseTimeout	idp_response_timeout (usang)	Opsional	120

Peran yang disukai

Amazon Resource Name (ARN) dari peran yang diambil. Untuk informasi tentang peran ARN, lihat AssumeRoledi Referensi AWS Security Token Service API.

Nama parameter	Alias	Jenis parameter	Nilai default
PreferredRole	preferred_role (tidak digunakan lagi)	Opsional	none

Durasi sesi peran

Durasi, dalam hitungan detik, dari sesi peran. Untuk informasi selengkapnya, lihat AssumeRoledi Referensi AWS Security Token Service API.

Nama parameter	Alias	Jenis parameter	Nilai default
RoleSessionDuration	Durasi (usang)	Opsional	3600

Lake Formation diaktifkan

Menentukan apakah akan menggunakan tindakan API Lake Formation AssumeDecoratedRoleWithSAMP untuk mengambil kredenal IAM sementara, bukan tindakan SAMP API. AssumeRoleWith AWS STS

Nama parameter	Alias	Jenis parameter	Nilai default
LakeFormationEnabled	none	Opsional	SALAH

Kredensi SAM Browser

Browser SAMP adalah plugin otentikasi generik yang dapat bekerja dengan penyedia identitas berbasis SAMP dan mendukung otentikasi multi-faktor.

Penyedia kredensil

Penyedia kredensi yang akan digunakan untuk mengautentikasi permintaan ke. AWS Tetapkan nilai parameter ini ke`BrowserSaml`.

Nama parameter	Alias	Jenis parameter	Nilai default	Nilai untuk digunakan
CredentialsProvider	AWSCredentialsProviderClass (usang)	Wajib	none	`BrowserSaml`

URL login masuk tunggal

URL masuk tunggal untuk aplikasi Anda pada penyedia identitas berbasis SALL.

Nama parameter	Alias	Jenis parameter	Nilai default
SsoLoginUrl	login_url (usang)	Wajib	none

Dengarkan port

Nomor port yang digunakan untuk mendengarkan respons SAFL. Nilai ini harus sesuai dengan URL yang Anda gunakan untuk mengonfigurasi penyedia identitas berbasis SAFL (misalnya,`http://localhost:7890/athena`).

Nama parameter	Alias	Jenis parameter	Nilai default
ListenPort	listen_port (usang)	Opsional	`7890`

Batas waktu respons penyedia identitas

Durasi, dalam hitungan detik, sebelum pengemudi berhenti menunggu respons SAFL dari Azure AD.

Nama parameter	Alias	Jenis parameter	Nilai default
IdpResponseTimeout	idp_response_timeout (usang)	Opsional	120

Peran yang disukai

Amazon Resource Name (ARN) dari peran yang diambil. Untuk informasi tentang peran ARN, lihat AssumeRoledi Referensi AWS Security Token Service API.

Nama parameter	Alias	Jenis parameter	Nilai default
PreferredRole	preferred_role (tidak digunakan lagi)	Opsional	none

Durasi sesi peran

Durasi, dalam hitungan detik, dari sesi peran. Untuk informasi selengkapnya, lihat AssumeRoledi Referensi AWS Security Token Service API.

Nama parameter	Alias	Jenis parameter	Nilai default
RoleSessionDuration	Durasi (usang)	Opsional	3600

Lake Formation diaktifkan

Menentukan apakah akan menggunakan tindakan API Lake Formation AssumeDecoratedRoleWithSAMP untuk mengambil kredenal IAM sementara, bukan tindakan SAMP API. AssumeRoleWith AWS STS

Nama parameter	Alias	Jenis parameter	Nilai default
LakeFormationEnabled	none	Opsional	SALAH

Konfigurasi JDBC 3.x lainnya

Bagian berikut menjelaskan beberapa pengaturan konfigurasi tambahan untuk driver JDBC 3.x.

Batas waktu jaringan

Jumlah waktu, dalam milidetik, pengemudi akan menunggu respons ketika melakukan panggilan API ke Athena. Setelah waktu ini, pengemudi melempar pengecualian batas waktu.

Batas waktu jaringan tidak dapat ditetapkan sebagai parameter koneksi. Untuk mengaturnya, panggil `setNetworkTimeout` metode pada objek JDBC`Connection`. Nilai ini dapat diubah selama siklus hidup koneksi JDBC. Nilai default dari parameter ini adalah`infinity`.

Contoh berikut menetapkan batas waktu jaringan menjadi 5000 milidetik.

```
...
AthenaDriver driver = new AthenaDriver();
Connection connection = driver.connect(url, connectionParameters);
connection.setNetworkTimeout(null, 5000);
...
```

Batas waktu kueri

Jumlah waktu, dalam hitungan detik, pengemudi akan menunggu permintaan selesai di Athena setelah kueri diajukan. Setelah waktu ini, pengemudi mencoba membatalkan kueri yang dikirimkan dan melempar pengecualian batas waktu.

Batas waktu kueri tidak dapat ditetapkan sebagai parameter koneksi. Untuk mengaturnya, panggil `setQueryTimeout` metode pada objek JDBC`Statement`. Nilai ini dapat diubah selama siklus hidup pernyataan JDBC. Nilai default parameter ini adalah `0` (nol). Nilai `0` berarti bahwa kueri dapat dijalankan hingga selesai (tergantung padaService Quotas).

Contoh berikut menetapkan batas waktu query untuk 5 detik.

```
...
AthenaDriver driver = new AthenaDriver();
Connection connection = driver.connect(url, connectionParameters);
Statement statement = connection.createStatement();
statement.setQueryTimeout(5);
...
```

Catatan rilis Amazon Athena JDBC 3.x

Catatan rilis ini memberikan rincian perbaikan dan perbaikan di driver Amazon Athena JDBC 3.x.

3.2.0

Dirilis 2024-04-26

Perbaikan

- Kinerja operasi katalog — Kinerja telah ditingkatkan untuk operasi katalog yang tidak menggunakan karakter wildcard.
- Perubahan interval polling minimum — Default interval polling minimum telah dimodifikasi untuk mengurangi jumlah panggilan API yang dilakukan pengemudi ke Athena. Penyelesaian kueri masih terdeteksi sesegera mungkin.
- Penemuan alat BI — Pengemudi telah dibuat lebih mudah ditemukan untuk alat intelijen bisnis.
- Pemetaan tipe data — Pemetaan tipe data ke Athena`binary`,`array`, dan tipe data `struct` DDL telah ditingkatkan.
- AWS Versi SDK — Versi AWS SDK yang digunakan dalam driver telah diperbarui ke 2.25.34.

Perbaikan

- Daftar tabel katalog federasi - Memperbaiki masalah yang menyebabkan katalog federasi mengembalikan daftar tabel kosong.
- getSchemas — Memperbaiki masalah yang menyebabkan metode JDBC DatabaseMetaData#getSchemas mengambil database hanya dari katalog default, bukan dari semua katalog.
- getColumns — Memperbaiki masalah yang menyebabkan katalog null dikembalikan saat metode JDBC DatabaseMetaData#getColumns dipanggil dengan nama katalog null.

3.1.0

Dirilis 2024-02-15

Perbaikan

- Support ditambahkan untuk Microsoft Active Directory Federation Services (AD FS) Windows Integrated Authentication dan otentikasi berbasis formulir.
- Untuk kompatibilitas mundur dengan versi 2.x, sub-protokol `awsathena` JDBC sekarang diterima tetapi menghasilkan peringatan penghentian. Gunakan sub-protokol `athena` JDBC sebagai gantinya.
- `AwsDataCatalog`sekarang default untuk parameter katalog, dan `default` merupakan default untuk parameter database. Perubahan ini memastikan bahwa nilai yang benar untuk katalog dan database saat ini dikembalikan, bukan null.

- Sesuai dengan spesifikasi JDBC, `IS_AUTOINCREMENT` dan `IS_GENERATEDCOLUMN` sekarang kembalikan string kosong, bukan. `NO`
- Tipe `int` data Athena sekarang memetakan ke tipe JDBC yang sama dengan Athena, bukan ke. `integer` `other`
- Ketika metadata kolom dari Athena tidak berisi opsional `precision` dan `scale` bidang, driver sekarang mengembalikan nol untuk nilai yang sesuai dalam kolom. `ResultSet`
- Versi AWS SDK telah diperbarui ke 2.21.39.

Perbaikan

- Memperbaiki masalah `GetQueryResultsStream` yang menyebabkan pengecualian terjadi ketika hasil teks biasa dari Athena memiliki jumlah kolom yang tidak konsisten dengan jumlah kolom di metadata hasil Athena.

3.0.0

Dirilis 2023-11-16

Driver Athena JDBC 3.x adalah driver generasi baru yang menawarkan kinerja dan kompatibilitas yang lebih baik. Driver JDBC 3.x mendukung pembacaan hasil kueri langsung dari Amazon S3, yang meningkatkan kinerja aplikasi yang mengkonsumsi hasil kueri besar. Driver baru ini juga memiliki lebih sedikit dependensi pihak ketiga, yang membuat integrasi dengan alat BI dan aplikasi khusus lebih mudah.

Versi sebelumnya dari driver Athena JDBC 3.x

Kami sangat menyarankan agar Anda menggunakan versi terbaru dari driver JDBC 3.x. Versi terbaru dari driver berisi perbaikan dan perbaikan terbaru. Gunakan versi yang lebih lama hanya jika aplikasi Anda mengalami ketidakcocokan dengan versi terbaru.

Toples uber driver JDBC

Unduhan berikut mengemas driver dan semua dependensinya dalam file yang sama `.jar`. Unduhan ini biasanya digunakan untuk klien SQL pihak ketiga.

- 3.1.0 toples uber toples uber
- https://downloads.athena.us-east-1.amazonaws.com/drivers/JDBC/3.0.0/athena-jdbc-3.0.0-with-dependencies.jar

Toples ramping driver JDBC

Download berikut adalah `.zip` file yang berisi lean `.jar` untuk driver dan `.jar` file terpisah untuk dependensi driver. Download ini biasanya digunakan untuk aplikasi kustom yang mungkin memiliki dependensi yang bertentangan dengan dependensi yang digunakan driver. Unduhan ini berguna jika Anda ingin memilih dependensi driver mana yang akan disertakan dengan lean jar, dan mana yang harus dikecualikan jika aplikasi khusus Anda sudah berisi satu atau lebih dari mereka.

- 3.1.0 toples tanpa lemak toples tanpa lemak
- https://downloads.athena.us-east-1.amazonaws.com/drivers/JDBC/3.0.0/athena-jdbc-3.0.0-lean-jar-and-separate-dependencies-jars.zip

Athena JDBC 2.x driver

Anda dapat menggunakan koneksi JDBC untuk menghubungkan Athena ke alat intelijen bisnis dan aplikasi lain, seperti meja kerja SQL. Untuk melakukan ini, gunakan tautan Amazon S3 di halaman ini untuk mengunduh, menginstal, dan mengonfigurasi driver Athena JDBC 2.x. Untuk informasi tentang membangun URL koneksi JDBC, lihat panduan instalasi dan konfigurasi driver JDBC yang dapat diunduh Panduan instalasi dan konfigurasi driver . Untuk informasi izin, lihat Akses melalui koneksi JDBC dan ODBC. Untuk mengirimkan umpan balik mengenai driver JDBC, kirim email ke athena-feedback@amazon.com. Dimulai dengan versi 2.0.24, dua versi driver tersedia: satu yang menyertakan AWS SDK, dan satu yang tidak.

Important

Saat Anda menggunakan driver JDBC, pastikan untuk mencatat persyaratan berikut:

- Buka port 444 - Simpan port 444, yang digunakan Athena untuk mengalirkan hasil kueri, terbuka untuk lalu lintas keluar. Saat Anda menggunakan PrivateLink titik akhir untuk terhubung ke Athena, pastikan grup keamanan yang terpasang pada titik akhir terbuka untuk PrivateLink lalu lintas masuk di port 444. Jika port 444 diblokir, Anda mungkin menerima pesan kesalahan[Simba] [AthenaJDBC] (100123) Telah terjadi kesalahan. Pengecualian selama inisialisasi kolom.
- athena: GetQueryResultsStream policy — Tambahkan tindakan `athena:GetQueryResultsStream` kebijakan ke kepala sekolah IAM yang menggunakan driver JDBC. Tindakan dasar ini tidak terdedah secara langsung dengan API. Ini hanya digunakan dengan driver ODBC dan JDBC sebagai bagian

dari dukungan hasil streaming. Untuk contoh kebijakan, lihat AWS kebijakan terkelola: AWSQuicksightAthenaAccess.

- Menggunakan driver JDBC untuk beberapa katalog data — Untuk menggunakan driver JDBC untuk beberapa katalog data dengan Athena (misalnya, saat menggunakan metastore Hive eksternal atau kueri federasi), sertakan dalam string koneksi JDBC Anda. `MetadataRetrievalMethod=ProxyAPI`
- 4.1 driver - Mulai tahun 2023, dukungan driver untuk JDBC versi 4.1 dihentikan. Tidak ada pembaruan lebih lanjut yang akan dirilis. Jika Anda menggunakan driver JDBC 4.1, migrasi ke driver 4.2 sangat disarankan.

Driver JDBC 2.x dengan SDK AWS

Driver JDBC versi 2.1.5 sesuai dengan standar data JDBC API 4.2 dan memerlukan JDK 8.0 atau yang lebih baru. Untuk informasi tentang memeriksa versi Java Runtime Environment (JRE) yang Anda gunakan, lihat dokumentasi Java.

Gunakan tautan berikut untuk mengunduh file driver `.jar` JDBC 4.2.

- https://downloads.athena.us-east-1.amazonaws.com/drivers/JDBC/SimbaAthenaJDBC-2.1.5.1000/AthenaJDBC42-2.1.5.1000.jar

Unduhan `.zip` file berikut berisi `.jar` file untuk JDBC 4.2 dan mencakup AWS SDK dan dokumentasi, catatan rilis, lisensi, dan perjanjian yang menyertainya.

- SimbaAthena

Driver JDBC 2.x tanpa SDK AWS

Driver JDBC versi 2.1.5 sesuai dengan standar data JDBC API 4.2 dan memerlukan JDK 8.0 atau yang lebih baru. Untuk informasi tentang memeriksa versi Java Runtime Environment (JRE) yang Anda gunakan, lihat dokumentasi Java.

Gunakan tautan berikut untuk mengunduh `.jar` file driver JDBC 4.2 tanpa SDK AWS .

- https://downloads.athena.us-east-1.amazonaws.com/drivers/JDBC/SimbaAthenaJDBC-2.1.5.1001/AthenaJDBC42-2.1.5.1001.jar

Unduhan `.zip` file berikut berisi `.jar` file untuk JDBC 4.2 dan dokumentasi, catatan rilis, lisensi, dan perjanjian yang menyertainya. Itu tidak termasuk AWS SDK.

- SimbaAthena

Catatan rilis driver JDBC 2.x, perjanjian lisensi, dan pemberitahuan

Setelah mengunduh versi yang Anda butuhkan, baca catatan rilis, dan tinjau Perjanjian Lisensi dan Pemberitahuan.

- Catatan rilis Catatan
- Perjanjian lisensi Perjanjian
- Pemberitahuan
- Lisensi pihak ketiga Lisensi

Dokumentasi driver JDBC 2.x

Download dokumentasi berikut untuk driver:

- Panduan instalasi dan konfigurasi driver JDBC Panduan instalasi dan konfigurasi . Gunakan panduan ini untuk menginstal dan mengonfigurasi driver.
- Panduan migrasi driver JDBC Panduan migrasi driver . Gunakan panduan ini untuk bermigrasi dari versi sebelumnya ke versi saat ini.

Menghubungkan ke Amazon Athena dengan ODBC

Amazon Athena menawarkan dua driver ODBC, versi 1.x dan 2.x. Driver Athena ODBC 2.x adalah alternatif baru yang mendukung sistem Linux, macOS ARM, macOS Intel, dan Windows 64-bit. Driver Athena 2.x mendukung semua plugin otentikasi yang didukung driver 1.x ODBC, dan hampir semua parameter koneksi kompatibel ke belakang.

- Untuk mengunduh driver ODBC 2.x, lihat. Amazon Athena ODBC 2.x
- Untuk mengunduh driver ODBC 1.x, lihat. Athena ODBC 1.x driver

Topik

- Amazon Athena ODBC 2.x

- Athena ODBC 1.x driver
- Menggunakan konektor Amazon Athena Power BI

Amazon Athena ODBC 2.x

Anda dapat menggunakan koneksi ODBC untuk terhubung ke Amazon Athena dari banyak alat dan aplikasi klien SQL pihak ketiga. Anda mengatur koneksi ODBC di komputer klien Anda.

Pertimbangan dan batasan

- Untuk informasi tentang migrasi dari driver Athena ODBC 1.x ke driver Athena 2.x ODBC, lihat. Migrasi ke driver ODBC 2.x
- Saat menggunakan fetcher S3 dengan opsi `CSE_KMS` enkripsi, klien Amazon S3 tidak dapat mendekripsi hasil yang disimpan di bucket Amazon S3. Sebagai solusinya, gunakan opsi API streaming Athena untuk mengambil set hasil.

Unduhan driver ODBC 2.x

Untuk mengunduh driver Amazon Athena 2.x ODBC, kunjungi tautan di halaman ini.

Important

Saat Anda menggunakan driver ODBC 2.x, pastikan untuk mencatat persyaratan berikut:

- Buka port 444 - Simpan port 444, yang digunakan Athena untuk mengalirkan hasil kueri, terbuka untuk lalu lintas keluar. Saat Anda menggunakan PrivateLink titik akhir untuk terhubung ke Athena, pastikan grup keamanan yang terpasang pada titik akhir terbuka untuk PrivateLink lalu lintas masuk di port 444.
- athena: GetQueryResultsStream policy — Tambahkan tindakan `athena:GetQueryResultsStream` kebijakan ke kepala sekolah IAM yang menggunakan driver ODBC. Tindakan dasar ini tidak terdedah secara langsung dengan API. Ini hanya digunakan dengan driver ODBC dan JDBC sebagai bagian dari dukungan hasil streaming. Untuk contoh kebijakan, lihat AWS kebijakan terkelola: AWSQuicksightAthenaAccess.

Linux

Versi Driver	Tautan unduhan
ODBC 2.0.3.0 untuk Linux 64-bit	Linux 64 bit ODBC driver 2.0.3.0 ODBC driver 2.0.3.0

macOS (ARM)

Versi Driver	Tautan unduhan
ODBC 2.0.3.0 untuk macOS 64-bit (ARM)	macOS 64 bit ODBC driver 2.0.3.0 (ARM) macOS bit ODBC driver 2.0.3.0 (ARM)

macOS (Intel)

Versi Driver	Tautan unduhan
ODBC 2.0.3.0 untuk macOS 64-bit (Intel)	macOS 64 bit ODBC driver 2.0.3.0 (Intel) macOS bit ODBC driver 2.0.3.0 (Intel)

Windows

Versi Driver	Tautan unduhan
ODBC 2.0.3.0 untuk Windows 64-bit	Windows 64 bit ODBC driver 2.0.3.0 ODBC driver 2.0.3.0

Topik

- Memulai dengan driver ODBC 2.x
- Parameter koneksi Athena ODBC 2.x
- Migrasi ke driver ODBC 2.x
- Memecahkan masalah driver ODBC 2.x

- Catatan rilis Amazon Athena ODBC 2.x

Memulai dengan driver ODBC 2.x

Gunakan informasi di bagian ini untuk memulai dengan driver Amazon Athena ODBC 2.x. Driver didukung pada sistem operasi Windows, Linux, dan macOS.

Topik

- Windows
- Linux
- macOS

Windows

Jika Anda ingin menggunakan komputer klien Windows untuk mengakses Amazon Athena, driver Amazon Athena ODBC diperlukan.

Persyaratan sistem Windows

Instal driver Amazon Athena ODBC di komputer klien yang akan mengakses database Amazon Athena secara langsung alih-alih menggunakan browser web.

Sistem Windows yang Anda gunakan harus memenuhi persyaratan berikut:

- Anda memiliki hak administrator
- Salah satu sistem operasi berikut:
  - Windows 11, 10, atau 8.1
  - Windows Server 2019, 2016, atau 2012
- Setidaknya 100 MB ruang disk yang tersedia
- Microsoft Visual C ++ Redistributable untuk Visual Studio untuk Windows 64-bit diinstal.

Menginstal driver Amazon Athena ODBC

Untuk mengunduh dan menginstal driver Amazon Athena ODBC untuk Windows

1. Unduh file `AmazonAthenaODBC-2.x.x.x.msi` instalasi.
2. Luncurkan file instalasi, lalu pilih Berikutnya.

3. Untuk menerima ketentuan perjanjian lisensi, pilih kotak centang, lalu pilih Berikutnya.
4. Untuk mengubah lokasi instalasi, pilih Browse, browse ke folder yang diinginkan, dan kemudian pilih OK.
5. Untuk menerima lokasi instalasi, pilih Berikutnya.
6. Pilih Instal.
7. Ketika instalasi selesai, pilih Selesai.

Cara untuk mengatur opsi konfigurasi driver

Untuk mengontrol perilaku driver Amazon Athena ODBC di Windows, Anda dapat mengatur opsi konfigurasi driver dengan cara berikut:

- Dalam program Administrator Sumber Data ODBC saat Anda mengonfigurasi nama sumber data (DSN).
- Dengan menambahkan atau mengubah kunci registri Windows di lokasi berikut:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI\YOUR_DSN_NAME
```

- Dengan mengatur opsi driver dalam string koneksi saat Anda terhubung secara terprogram.

Mengkonfigurasi nama sumber data pada Windows

Setelah mengunduh dan menginstal driver ODBC, Anda harus menambahkan entri nama sumber data (DSN) ke komputer klien atau instans Amazon EC2. Alat klien SQL menggunakan sumber data ini untuk terhubung ke dan menanyakan Amazon Athena.

Untuk membuat entri sistem DSN

1. Dari menu Start Windows, klik kanan Sumber Data ODBC (64 bit), lalu pilih Lainnya, Jalankan sebagai administrator.
2. Di Administrator Sumber Data ODBC, pilih tab Drivers.
3. Di kolom Nama, verifikasi bahwa Amazon Athena ODBC (x64) ada.
4. Lakukan salah satu hal berikut ini:
    - Untuk mengkonfigurasi driver untuk semua pengguna di komputer, pilih tab Sistem DSN. Karena aplikasi yang menggunakan akun berbeda untuk memuat data mungkin tidak dapat

mendeteksi DSN pengguna dari akun lain, kami merekomendasikan opsi konfigurasi DSN sistem.

> **Note**
>
> Menggunakan opsi Sistem DSN membutuhkan hak administratif.

- Untuk mengonfigurasi driver hanya untuk akun pengguna Anda, pilih tab DSN Pengguna.

5. Pilih Tambahkan. Kotak dialog Create New Data Source terbuka.
6. Pilih Amazon Athena ODBC (x64), lalu pilih Selesai.
7. Di kotak dialog Konfigurasi ODBC Amazon Athena, masukkan informasi berikut. Untuk informasi detail tentang opsi ini, lihat Parameter koneksi ODBC 2.x utama.

   - Untuk Nama Sumber Data, masukkan nama yang ingin Anda gunakan untuk mengidentifikasi sumber data.
   - Untuk Deskripsi, masukkan deskripsi untuk membantu Anda mengidentifikasi sumber data.
   - Untuk Wilayah, masukkan nama tempat Wilayah AWS Anda akan menggunakan Athena (misalnya, **`us-west-1`**).
   - Untuk Katalog, masukkan nama katalog Amazon Athena. Defaultnya adalah AwsDataCatalog, yang digunakan oleh AWS Glue.
   - Untuk Database, masukkan nama database Amazon Athena. Defaultnya adalah default.
   - Untuk Workgroup, masukkan nama workgroup Amazon Athena. Defaultnya adalah primer.
   - Untuk Lokasi Output S3, masukkan lokasi di Amazon S3 tempat hasil kueri akan disimpan (misalnya**`s3://DOC-EXAMPLE-BUCKET/`**,).
   - (Opsional) Untuk Opsi Enkripsi, pilih opsi enkripsi. Nilai default-nya `NOT_SET`.
   - (Opsional) Untuk Kunci KMS, pilih kunci KMS enkripsi jika diperlukan.

8. Untuk menentukan opsi konfigurasi untuk autentikasi IAM, pilih Opsi Otentikasi.
9. Masukkan informasi berikut:

   - Untuk Jenis Otentikasi, pilih Kredensial IAM. Ini adalah opsi default. Untuk informasi selengkapnya tentang jenis otentikasi yang tersedia, lihatPilihan otentikasi.
   - Untuk Nama Pengguna, masukkan nama pengguna.
   - Untuk Kata Sandi, masukkan kata sandi.

- Untuk Token Sesi, masukkan token sesi jika Anda ingin menggunakan AWS kredensil sementara. Untuk informasi tentang kredensil sementara, lihat Menggunakan kredensil sementara dengan AWS sumber daya di Panduan Pengguna IAM.

10. Pilih OK.
11. Di bagian bawah kotak dialog Konfigurasi ODBC Amazon Athena, pilih Uji. Jika komputer klien berhasil terhubung ke Amazon Athena, kotak uji Koneksi melaporkan Koneksi berhasil. Jika tidak, kotak melaporkan Koneksi gagal dengan informasi kesalahan yang sesuai.
12. Pilih OK untuk menutup tes koneksi. Sumber data yang Anda buat sekarang muncul dalam daftar nama sumber data.

Menggunakan koneksi tanpa DSN di Windows

Anda dapat menggunakan koneksi tanpa DSN untuk terhubung ke database tanpa Nama Sumber Data (DSN). Contoh berikut menunjukkan string koneksi untuk driver ODBC Amazon Athena ODBC (x64) yang terhubung ke Amazon Athena.

```
DRIVER={Amazon Athena ODBC (x64)};Catalog=AwsDataCatalog;AwsRegion=us-
west-1;Schema=test_schema;S3OutputLocation=
s3://DOC-EXAMPLE-BUCKET/;AuthenticationType=IAM Credentials;UID=YOUR_UID;PWD=YOUR_PWD;
```

Linux

Jika Anda ingin menggunakan komputer klien Linux untuk mengakses Amazon Athena, driver Amazon Athena ODBC diperlukan.

Persyaratan sistem Linux

Setiap komputer klien Linux tempat Anda menginstal driver harus memenuhi persyaratan berikut.

- Anda memiliki akses root.
- Gunakan salah satu distribusi Linux berikut:
  - Red Hat Enterprise Linux (RHEL) 7 atau 8
  - CentOS 7 atau 8.
- Memiliki 100 MB ruang disk yang tersedia.
- Gunakan UnixODBC versi 2.3.1 atau yang lebih baru.
- Gunakan versi 2.26 atau yang lebih baru dari Perpustakaan GNU C (glibc).

Menginstal konektor data ODBC di Linux

Gunakan prosedur berikut untuk menginstal driver Amazon Athena ODBC pada sistem operasi Linux.

Untuk menginstal driver Amazon Athena ODBC di Linux

1. Masukkan salah satu perintah berikut:

   ```
   sudo rpm -Uvh AmazonAthenaODBC-2.X.Y.Z.rpm
   ```

   atau

   ```
   sudo yum --nogpgcheck localinstall AmazonAthenaODBC-2.X.Y.Z.rpm
   ```

2. Setelah instalasi selesai, masukkan salah satu perintah berikut untuk memverifikasi bahwa driver diinstal:

   - ```
     yum list | grep amazon-athena-odbc-driver
     ```

     Output:

     ```
     amazon-athena-odbc-driver.x86_64 2.0.2.1-1.amzn2int installed
     ```

   - ```
     rpm -qa | grep amazon
     ```

     Output:

     ```
     amazon-athena-odbc-driver-2.0.2.1-1.amzn2int.x86_64
     ```

Mengkonfigurasi nama sumber data di Linux

Setelah driver diinstal, Anda dapat menemukan contoh `.odbc.ini` dan `.odbcinst.ini` file di lokasi berikut:

- `/opt/athena/odbc/ini/`.

Gunakan `.ini` file di lokasi ini sebagai contoh untuk mengonfigurasi driver Amazon Athena ODBC dan nama sumber data (DSN).

> **Note**
>
> Secara default, manajer driver ODBC menggunakan file konfigurasi tersembunyi `.odbc.ini` dan `.odbcinst.ini`, yang terletak di direktori home.

Untuk menentukan jalur ke `.odbcinst.ini` file `.odbc.ini` dan menggunakan UnixODBC, lakukan langkah-langkah berikut.

Untuk menentukan lokasi **`.ini`** file ODBC menggunakan UnixODBC

1. Atur `ODBCINI` ke path lengkap dan nama file `odbc.ini` file, seperti pada contoh berikut.

   ```
   export ODBCINI=/opt/athena/odbc/ini/odbc.ini
   ```

2. Setel `ODBCSYSINI` ke path lengkap direktori yang berisi `odbcinst.ini` file, seperti pada contoh berikut.

   ```
   export ODBCSYSINI=/opt/athena/odbc/ini
   ```

3. Masukkan perintah berikut untuk memverifikasi bahwa Anda menggunakan pengelola driver UnixODBC dan file yang benar: `odbc*.ini`

   ```
   username % odbcinst -j
   ```

   Output sampel

   ```
   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. Jika Anda ingin menggunakan nama sumber data (DSN) untuk terhubung ke penyimpanan data Anda, konfigurasikan `odbc.ini` file untuk menentukan nama sumber data (DSN). Tetapkan properti dalam `odbc.ini` file untuk membuat DSN yang menentukan informasi koneksi untuk penyimpanan data Anda, seperti pada contoh berikut.

```
[ODBC Data Sources]
athena_odbc_test=Amazon Athena ODBC (x64)

[ATHENA_WIDE_SETTINGS]  # Special DSN-name to signal driver about logging
 configuration.
LogLevel=0              # To enable ODBC driver logs, set this to 1.
UseAwsLogger=0          # To enable AWS-SDK logs, set this to 1.
LogPath=/opt/athena/odbc/logs/ # Path to store the log files. Permissions to the
 location are required.

[athena_odbc_test]
Driver=/opt/athena/odbc/lib/libathena-odbc.so
AwsRegion=us-west-1
Workgroup=primary
Catalog=AwsDataCatalog
Schema=default
AuthenticationType=IAM Credentials
UID=
PWD=
S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/
```

5. Konfigurasikan `odbcinst.ini` file, seperti pada contoh berikut.

```
[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. Setelah Anda menginstal dan mengonfigurasi driver Amazon Athena ODBC, gunakan alat `isql` baris perintah UnixODBC untuk memverifikasi koneksi, seperti pada contoh berikut.

```
username % isql -v "athena_odbc_test"
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
```

```
SQL>
```

macOS

Jika Anda ingin menggunakan komputer klien macOS untuk mengakses Amazon Athena, driver ODBC Amazon Athena diperlukan.

Persyaratan sistem macOS

Setiap komputer macOS tempat Anda menginstal driver harus memenuhi persyaratan berikut.

- Gunakan macOS versi 14 atau yang lebih baru.
- Memiliki 100 MB ruang disk yang tersedia.
- Gunakan versi 3.52.16 atau yang lebih baru dari iodBC.

Menginstal konektor data ODBC di macOS

Gunakan prosedur berikut untuk mengunduh dan menginstal driver Amazon Athena ODBC untuk sistem operasi macOS.

Untuk mengunduh dan menginstal driver Amazon Athena ODBC untuk macOS

1. Unduh file `.pkg` paket.
2. Klik dua kali `.pkg` file tersebut.
3. Ikuti langkah-langkah di wizard untuk menginstal driver.
4. Pada halaman Perjanjian Lisensi, tekan Lanjutkan, lalu pilih Setuju.
5. Pilih Instal.
6. Ketika instalasi selesai, pilih Selesai.
7. Masukkan perintah berikut untuk memverifikasi bahwa driver diinstal:

   ```
   > pkgutil --pkgs | grep athenaodbc
   ```

   Tergantung pada sistem Anda, output dapat terlihat seperti salah satu dari berikut ini.

   ```
   com.amazon.athenaodbc-x86_64.Config
   com.amazon.athenaodbc-x86_64.Driver
   ```

atau

```
com.amazon.athenaodbc-arm64.Config
com.amazon.athenaodbc-arm64.Driver
```

Mengonfigurasi nama sumber data di macOS

Setelah driver diinstal, Anda dapat menemukan contoh `.odbc.ini` dan `.odbcinst.ini` file di lokasi berikut:

- Komputer prosesor Intel: `/opt/athena/odbc/x86_64/ini/`
- Komputer prosesor ARM: `/opt/athena/odbc/arm64/ini/`

Gunakan `.ini` file di lokasi ini sebagai contoh untuk mengonfigurasi driver Amazon Athena ODBC dan nama sumber data (DSN).

> **Note**
>
> Secara default, manajer driver ODBC menggunakan file konfigurasi tersembunyi `.odbc.ini` dan`.odbcinst.ini`, yang terletak di direktori home.

Untuk menentukan jalur ke `.odbcinst.ini` file `.odbc.ini` dan menggunakan manajer driver IodBC, lakukan langkah-langkah berikut.

Untuk menentukan lokasi **`.ini`** file ODBC menggunakan manajer driver iodBC

1. Atur `ODBCINI` ke path lengkap dan nama file `odbc.ini` file.

   - Untuk komputer macOS yang memiliki prosesor Intel, gunakan sintaks berikut.

     ```
     export ODBCINI=/opt/athena/odbc/x86_64/ini/odbc.ini
     ```

   - Untuk komputer macOS yang memiliki prosesor ARM, gunakan sintaks berikut.

     ```
     export ODBCINI=/opt/athena/odbc/arm64/ini/odbc.ini
     ```

2. Atur `ODBCSYSINI` ke path lengkap dan nama file `odbcinst.ini` file.

- Untuk komputer macOS yang memiliki prosesor Intel, gunakan sintaks berikut.

```
export ODBCSYSINI=/opt/athena/odbc/x86_64/ini/odbcinst.ini
```

- Untuk komputer macOS yang memiliki prosesor ARM, gunakan sintaks berikut.

```
export ODBCSYSINI=/opt/athena/odbc/arm64/ini/odbcinst.ini
```

3. Jika Anda ingin menggunakan nama sumber data (DSN) untuk terhubung ke penyimpanan data Anda, konfigurasikan `odbc.ini` file untuk menentukan nama sumber data (DSN). Tetapkan properti dalam `odbc.ini` file untuk membuat DSN yang menentukan informasi koneksi untuk penyimpanan data Anda, seperti pada contoh berikut.

```
[ODBC Data Sources]
athena_odbc_test=Amazon Athena ODBC (x64)

[ATHENA_WIDE_SETTINGS] # Special DSN-name to signal driver about logging
 configuration.
LogLevel=0             # set to 1 to enable ODBC driver logs
UseAwsLogger=0         # set to 1 to enable AWS-SDK logs
LogPath=/opt/athena/odbc/logs/ # Path to store the log files. Permissions to the
 location are required.

[athena_odbc_test]
Description=Amazon Athena ODBC (x64)
# For ARM:
Driver=/opt/athena/odbc/arm64/lib/libathena-odbc-arm64.dylib
# For Intel:
# Driver=/opt/athena/odbc/x86_64/lib/libathena-odbc-x86_64.dylib
AwsRegion=us-west-1
Workgroup=primary
Catalog=AwsDataCatalog
Schema=default
AuthenticationType=IAM Credentials
UID=
PWD=
S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/
```

4. Konfigurasikan `odbcinst.ini` file, seperti pada contoh berikut.

```
[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. Setelah Anda menginstal dan mengkonfigurasi driver Amazon Athena ODBC, gunakan alat `iodbctest` baris perintah untuk memverifikasi koneksi, seperti pada contoh berikut.

```
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>
```

Parameter koneksi Athena ODBC 2.x

YangKonfigurasi ODBC Amazon AthenaPilihan kotak dialog termasukPilihan Otentikasi,Opsi Lanjutan,Opsi Logging,Penggantian EndpointdanOpsi Proxy. Untuk informasi terperinci tentang masing-masing, kunjungi tautan yang sesuai.

- Parameter koneksi ODBC 2.x utama
- Pilihan otentikasi
- Opsi lanjutan
- Opsi pencatatan
- Pengganti titik akhir

- Opsi proxy

Parameter koneksi ODBC 2.x utama

Bagian berikut menjelaskan masing-masing parameter koneksi utama.

Nama sumber data

Menentukan nama sumber data Anda.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
DSN	Opsional untuk jenis koneksi tanpa DSN	none	`DSN=AmazonAthena0dbcUsWest1;`

Deskripsi

Berisi deskripsi sumber data Anda.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
Deskripsi	Opsional	none	`Description=Connection to Amazon Athena us-west-1;`

Katalog

Menentukan nama katalog data. Untuk informasi selengkapnya tentang katalog, lihat DataCatalogdi Referensi API Amazon Athena.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
Katalog	Opsional	`AwsDataCatalog`	`Catalog=AwsDataCatalog;`

Wilayah

Menentukan. Wilayah AWS Untuk selengkapnya Wilayah AWS, lihat Wilayah dan Availability Zone.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
AwsRegion	Wajib	`none`	`AwsRegion` = kami-barat-1;

Basis Data

Menentukan nama database. Untuk informasi selengkapnya tentang database, lihat Database di Referensi API Amazon Athena.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
Skema	Opsional	`default`	`Schema=default;`

Grup Kerja

Menentukan nama workgroup. Untuk informasi selengkapnya tentang workgroup, lihat WorkGroupdi Referensi API Amazon Athena.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
Grup Kerja	Opsional	`primary`	`Workgroup =primary;`

Lokasi keluaran

Menentukan lokasi di Amazon S3 tempat hasil kueri disimpan. Untuk informasi selengkapnya tentang lokasi keluaran, lihat ResultConfigurationdi Referensi API Amazon Athena.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
S3 OutputLocation	Wajib	`none`	`S3OutputL ocation=s3://`

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
			`DOC-EXAMPLE-BUCKET/;`

Opsi enkripsi

Nama parameter dialog: Opsi enkripsi

Menentukan opsi enkripsi. Untuk informasi selengkapnya tentang opsi enkripsi, lihat EncryptionConfigurationdi Referensi API Amazon Athena.

Nama string koneksi	Jenis parameter	Nilai default	Nilai yang mungkin	Contoh string koneksi
S3 OutputEnc Option	Opsional	`none`	`NOT_SET`, `SSE_S3`, `SSE_KMS`, `CSE_KMS`	`S3OutputEncOption=SSE_S3;`

Kunci KMS

Menentukan kunci KMS untuk enkripsi. Untuk informasi selengkapnya tentang konfigurasi enkripsi untuk KMS Keys, lihat EncryptionConfigurationdi Referensi API Amazon Athena.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
S3 OutputEnc KMSKey	Opsional	`none`	`S3OutputEncKMSKey=your_key;`

Tes koneksi

Administrator Sumber Data ODBC menyediakan opsi Uji yang dapat Anda gunakan untuk menguji koneksi ODBC 2.x Anda ke Amazon Athena. Untuk langkah, lihat Mengkonfigurasi nama sumber data pada Windows. Saat Anda menguji koneksi, driver ODBC memanggil aksi GetWorkGroupAthena API. Panggilan menggunakan jenis otentikasi dan penyedia kredensi terkait

yang Anda tentukan untuk mengambil kredensialnya. Tidak ada biaya untuk tes koneksi saat Anda menggunakan driver ODBC 2.x. Pengujian tidak menghasilkan hasil kueri di bucket Amazon S3 Anda.

Pilihan otentikasi

Anda dapat terhubung ke Amazon Athena menggunakan jenis autentikasi berikut. Untuk semua jenis, nama string koneksi adalah`AuthenticationType`, jenis parameternya adalah`Required`, dan nilai defaultnya adalah`IAM Credentials`. Untuk informasi tentang parameter untuk setiap jenis otentikasi, kunjungi tautan yang sesuai. Untuk parameter otentikasi umum, lihatParameter otentikasi umum.

Jenis otentikasi	Contoh string koneksi
Kredensial IAM	`AuthenticationType=IAM Credentials;`
Profil IAM	`AuthenticationType=IAM Profile;`
IKLAN FS	`AuthenticationType=ADFS;`
Iklan Azure	`AuthenticationType=AzureAD;`
Browser Azure	`AuthenticationType=BrowserAzureAD;`
Peramban SALL	`AuthenticationType=BrowserSAML;`
Browser SSO OIDC	`AuthenticationType=BrowserSSOOIDC;`
Kredensial default	`AuthenticationType=Default Credentials;`
Kredensial eksternal	`AuthenticationType=External Credentials;`
Profil instans	`AuthenticationType=Instance Profile;`
JWT	`AuthenticationType=JWT;`
Okta	`AuthenticationType=Okta;`
Ping	`AuthenticationType=Ping;`

Kredensial IAM

Anda dapat menggunakan kredensi IAM Anda untuk terhubung ke Amazon Athena dengan driver ODBC menggunakan parameter string koneksi yang dijelaskan di bagian ini.

Jenis otentikasi

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
AuthenticationType	Wajib	`IAM Credentials`	`AuthenticationType=IAM Credentials;`

ID Pengguna

ID Kunci AWS Akses Anda. Untuk informasi selengkapnya tentang kunci akses, lihat kredensi AWS keamanan di Panduan Pengguna IAM.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
UID	Wajib	`none`	`UID=AKIAIOSFODNN7EXAMPLE;`

Kata sandi

Id kunci AWS rahasiamu. Untuk informasi selengkapnya tentang kunci akses, lihat kredensi AWS keamanan di Panduan Pengguna IAM.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
PWD	Wajib	`none`	`PWD=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKE;`

Token sesi

Jika Anda menggunakan AWS kredensi sementara, Anda harus menentukan token sesi. Untuk informasi tentang kredensial sementara, lihat Kredensial keamanan sementara di IAM di Panduan Pengguna IAM.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
SessionToken	Opsional	none	`SessionToken=AQoDYXdzEJr...<remainder of session token>;`

Profil IAM

Anda dapat mengonfigurasi profil bernama untuk terhubung ke Amazon Athena menggunakan driver ODBC. Untuk menggunakan kredensi yang tersedia di profil instans Amazon EC2 hosting Anda, tetapkan`credential_source`parameter ke`Ec2InstanceMetadata`.
Jika Anda ingin menggunakan penyedia kredensi kustom di profil bernama, tentukan nilai untuk`plugin_name`parameter dalam konfigurasi profil Anda.

Jenis otentikasi

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
AuthenticationType	Diperlukan	`IAM Credentials`	`AuthenticationType=IAM Profile;`

AWSprofil

Nama profil yang digunakan untuk koneksi ODBC Anda. Untuk informasi selengkapnya tentang profil, lihatMenggunakan profil bernamadi dalamAWS Command Line InterfacePanduan Pengguna.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
AWSProfil	Diperlukan	none	`AWSProfile=default;`

Peran pilihan

Amazon Resource Name (ARN) dari peran yang diambil. Parameter peran yang disukai digunakan saat penyedia kredensi kustom ditentukan oleh`plugin_name`parameter dalam konfigurasi profil Anda. Untuk informasi selengkapnya tentang peran ARN, lihatAssumeRoledi dalamAWS Security Token ServiceReferensi API.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
preferred_role	Opsional	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Durasi sesi

Durasi, dalam hitungan detik, dari sesi peran. Untuk informasi selengkapnya tentang durasi sesi, lihatAssumeRoledi dalamAWS Security Token ServiceReferensi API. Parameter durasi sesi digunakan saat penyedia kredensi kustom ditentukan oleh`plugin_name`parameter dalam konfigurasi profil Anda.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
durasi	Opsional	`900`	`duration=900;`

Nama plugin

Menentukan nama penyedia kredensi kustom yang digunakan dalam profil bernama. Parameter ini dapat mengambil nilai yang sama seperti yang ada diJenis Otentikasibidang Administrator Sumber Data ODBC, tetapi hanya digunakan oleh`AWSProfile`konfigurasi.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
plugin_name	Opsional	`none`	`plugin_name=AzureAD;`

IKLAN FS

AD FS adalah plugin otentikasi berbasis SALL yang bekerja dengan penyedia identitas Active Directory Federation Service (AD FS). Plugin ini mendukungOtentikasi Windows terintegrasidan otentikasi berbasis bentuk. Jika Anda menggunakan Integrated Windows Authentication, Anda dapat menghilangkan nama pengguna dan kata sandi. Untuk informasi tentang mengonfigurasi AD FS dan Athena, lihatMengkonfigurasi akses federasi ke Amazon Athena untuk pengguna Microsoft AD FS menggunakan klien ODBC.

Jenis otentikasi

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
AuthenticationType	Diperlukan	`IAM Credentials`	`AuthenticationType=ADFS;`

ID Pengguna

Nama pengguna Anda untuk terhubung ke server AD FS. Untuk Integrated Windows Authentication, Anda dapat menghilangkan nama pengguna. Jika pengaturan AD FS memerlukan nama pengguna, Anda harus memberikannya dalam parameter koneksi.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
UID	Opsional untuk otentikasi terintegrasi windows	`none`	`UID=domain\username;`

Kata sandi

Kata sandi Anda untuk menghubungkan ke server AD FS. Seperti bidang nama pengguna, Anda dapat menghilangkan nama pengguna jika Anda menggunakan Integrated Windows Authentication. Jika pengaturan AD FS Anda memerlukan kata sandi, Anda harus memberikannya dalam parameter koneksi.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
PWD	Opsional untuk otentikasi terintegrasi windows	none	`PWD=password_3EXAMPLE;`

Peran yang disukai

Amazon Resource Name (ARN) dari peran yang diambil. Jika pernyataan SALL Anda memiliki beberapa peran, Anda dapat menentukan parameter ini untuk memilih peran yang akan diasumsikan. Peran ini harus hadir dalam pernyataan SALL. Untuk informasi selengkapnya tentang peran ARN, lihatAssumeRoledi dalamAWS Security Token ServiceReferensi API.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
preferred_role	Opsional	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Durasi sesi

Durasi, dalam hitungan detik, dari sesi peran. Untuk informasi selengkapnya tentang durasi sesi, lihatAssumeRoledi dalamAWS Security Token ServiceReferensi API.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
durasi	Opsional	`900`	`duration=900;`

Tuan rumah IdP

Nama host layanan AD FS.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
idp_host	Membutuhkan	none	`idp_host=<server-name>.<company.com>;`

Pelabuhan IdP

Port yang digunakan untuk terhubung ke host AD FS.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
idp_port	Diperlukan	none	`idp_port=443;`

LoginToRP

Pihak yang dipercaya mengandalkan. Gunakan parameter ini untuk mengganti URL endpoint pihak yang mengandalkan AD FS.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
LoginToRP	Opsional	`urn:amazon:webservices`	`LoginToRP=trustedparty;`

Iklan Azure

Azure AD adalah plugin otentikasi berbasis SALL yang bekerja dengan penyedia identitas Azure AD. Plugin ini tidak mendukung otentikasi multifaktor (MFA). Jika Anda memerlukan dukungan MFA, pertimbangkan untuk menggunakan`BrowserAzureAD`Plugin sebagai gantinya.

Jenis Autentikasi

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
AuthenticationType	Diperlukan	`IAM Credentials`	`AuthenticationType =AzureAD;`

Peran yang disukai

Amazon Resource Name (ARN) dari peran yang diambil. Untuk informasi tentang peran ARN, lihatAssumeRoledi dalamAWS Security Token ServiceReferensi API.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
preferred_role	Opsional	`none`	`preferred_role=arn:aws:iam: :123456789012:id/user1;`

Durasi sesi

Durasi, dalam hitungan detik, dari sesi peran. Untuk informasi selengkapnya, lihat AssumeRole dalam Referensi API AWS Security Token Service.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
durasi	Opsional	`900`	`duration=900;`

ID Penyewa

Menentukan ID penyewa aplikasi Anda.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
idp_tenant	Diperlukan	none	`idp_tenant=123zz112z-z12d-1z1f-11zz-f111aa111234;`

ID Klien

Menentukan ID klien aplikasi Anda.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
client_id	Diperlukan	none	`client_id=9178ac27-a1bc-1a2b-1a2b-a123abcd1234;`

Rahasia klien

Menentukan rahasia klien Anda.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
client_secret	Diperlukan	none	`client_secret=zG12q~.xzG1xxxZ1wX1.~ZzXXX1XxkHZizeT1zzZ;`

Browser Azure

Browser Azure AD adalah plugin otentikasi berbasis SALL yang bekerja dengan penyedia identitas Azure AD dan mendukung otentikasi multi-faktor. Berbeda dengan plugin Azure AD standar, plugin ini tidak memerlukan nama pengguna, kata sandi, atau rahasia klien dalam parameter koneksi.

Jenis Autentikasi

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
AuthenticationType	Diperlukan	`IAM Credentials`	`AuthenticationType=BrowserAzureAD;`

Peran pilihan

Amazon Resource Name (ARN) dari peran yang diambil. Jika pernyataan SAFL Anda memiliki beberapa peran, Anda dapat menentukan parameter ini untuk memilih peran yang akan diasumsikan. Peran yang ditentukan harus ada dalam pernyataan SAFL. Untuk informasi selengkapnya tentang peran ARN, lihat AssumeRoledi Referensi AWS Security Token ServiceAPI.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
preferred_role	Opsional	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Durasi sesi

Durasi dalam detik dari sesi peran. Untuk informasi selengkapnya tentang durasi sesi, lihat AssumeRoledi Referensi AWS Security Token Service API.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
durasi	Opsional	`900`	`duration=900;`

ID Penyewa

Menentukan ID penyewa aplikasi Anda.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
idp_penyewa	Diperlukan	none	`idp_tenant=123zz112z-z12d-1z1f-11zz-f111aa111234;`

ID klien

Menentukan ID klien aplikasi Anda.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
client_id	Diperlukan	none	`client_id=9178ac27-a1bc-1a2b-1a2b-a123abcd1234;`

Waktu habis

Durasi, dalam hitungan detik, sebelum plugin berhenti menunggu respons SAFL dari Azure AD.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
batas waktu	Opsional	`120`	`timeout=90;`

Aktifkan cache file Azure

Mengaktifkan cache kredensil sementara. Parameter koneksi ini memungkinkan kredensil sementara untuk di-cache dan digunakan kembali di antara beberapa proses. Gunakan opsi ini untuk mengurangi jumlah jendela browser yang dibuka saat Anda menggunakan alat BI seperti Microsoft Power BI.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
browser_azure_cache	Opsional	1	`browser_azure_cache=0;`

Peramban SALL

Browser SALL adalah plugin otentikasi generik yang dapat bekerja dengan penyedia identitas berbasis SALL dan mendukung otentikasi multi-faktor. Untuk informasi konfigurasi rinci, lihatMengkonfigurasi sistem masuk tunggal menggunakan ODBC, SAMP 2.0, dan Penyedia Identitas Okta.

Jenis otentikasi

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
AuthenticationType	Diperlukan	`IAM Credentials`	`AuthenticationType=BrowserSAML;`

Peran yang disukai

Amazon Resource Name (ARN) dari peran yang diambil. Jika pernyataan SAMM Anda memiliki beberapa peran, Anda dapat menentukan parameter ini untuk memilih peran yang akan diasumsikan. Peran ini harus hadir dalam pernyataan SALL. Untuk informasi selengkapnya tentang peran ARN, lihatAssumeRoledi dalamAWS Security Token ServiceReferensi API.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
preferred_role	Opsional	`none`	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Durasi sesi

Durasi, dalam hitungan detik, dari sesi peran. Untuk informasi selengkapnya, lihat AssumeRole dalam Referensi API AWS Security Token Service.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
durasi	Opsional	`900`	`duration=900;`

URL Masuk

URL masuk tunggal yang ditampilkan untuk aplikasi Anda.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
login_url	Diperlukan	none	`login_url =https:// trial-123 4567.okta.com/ app/trial-123 4567_okta browsersaml_1/ zzz4izzzAzDFB zZz1234/sso/ saml;`

Dengarkan port

Nomor port yang digunakan untuk mendengarkan respon SALL. Nilai ini harus sesuai dengan URL IAM Identity Center yang Anda konfigurasikan dengan IDP (misalnya,`http://localhost:7890/athena`).

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
listen_port	Opsional	`7890`	`listen_po rt=7890;`

Waktu habis

Durasi, dalam hitungan detik, sebelum plugin berhenti menunggu respons SALL dari penyedia identitas.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
batas waktu	Opsional	`120`	`timeout=90;`

Browser SSO OIDC

Browser SSO OIDC adalah plugin otentikasi yang berfungsi dengan. AWS IAM Identity CenterUntuk informasi tentang mengaktifkan dan menggunakan Pusat Identitas IAM, lihat Langkah 1: Aktifkan Pusat Identitas IAM di Panduan Pengguna.AWS IAM Identity Center

Jenis otentikasi

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
AuthenticationType	Wajib	`IAM Credentials`	`AuthenticationType=BrowserSSOOIDC;`

URL Mulai Pusat Identitas IAM

URL untuk portal AWS akses. Tindakan StartDeviceAuthorizationAPI Pusat Identitas IAM menggunakan nilai ini untuk `startUrl` parameter.

Untuk menyalin URL portal AWS akses

1. Masuk ke AWS Management Console dan buka AWS IAM Identity Center konsol di https://console.aws.amazon.com/singlesignon/.
2. Pada panel navigasi, silakan pilih Pengaturan.
3. Pada halaman Pengaturan, di bawah Sumber identitas, pilih ikon clipboard untuk URL portal AWS akses.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
sso_oidc_start_url	Wajib	none	`sso_oidc_start_url=https://app_id.awsapps.com/start;`

Wilayah Pusat Identitas IAM

Wilayah AWS Tempat SSO Anda dikonfigurasi. Klien `SSOOIDCClient` dan `SSOClient` AWS SDK menggunakan nilai ini untuk `region` parameter.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
sso_oidc_wilayah	Wajib	`none`	`sso_oidc_region=us-east-1;`

Lingkup

Daftar cakupan yang ditentukan oleh klien. Setelah otorisasi, daftar ini membatasi izin saat token akses diberikan. Tindakan RegisterClientAPI Pusat Identitas IAM menggunakan nilai ini untuk `scopes` parameter.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
sso_oidc_cakupan	Opsional	`none`	`sso_oidc_scopes=scope1,scope2,scope3;`

account-id

Pengidentifikasi untuk Akun AWS yang ditugaskan ke pengguna. IAM Identity Center GetRoleCredentialsAPI menggunakan nilai ini untuk `accountId` parameter.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
sso_oidc_account_id	Wajib	`none`	`sso_oidc_account_id=123456789123;`

Nama peran

Nama ramah dari peran yang diberikan kepada pengguna. Nama yang Anda tentukan untuk set izin ini muncul di portal AWS akses sebagai peran yang tersedia. Tindakan GetRoleCredentialsAPI Pusat Identitas IAM menggunakan nilai ini untuk `roleName` parameter.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
sso_oidc_role_name	Wajib	none	`sso_oidc_role_name=AthenaReadAccess;`

Waktu habis

Jumlah detik SSO API polling harus memeriksa token akses.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
sso_oidc_timeout	Opsional	120	`sso_oidc_timeout=60;`

Aktifkan cache file

Mengaktifkan cache kredensial sementara. Parameter koneksi ini memungkinkan kredenal sementara untuk di-cache dan digunakan kembali di antara beberapa proses. Gunakan opsi ini untuk mengurangi jumlah jendela browser yang dibuka saat Anda menggunakan alat BI seperti Microsoft Power BI.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
sso_oidc_cache	Opsional	1	`sso_oidc_cache=0;`

Kredensial default

Anda dapat menggunakan kredensi default yang Anda konfigurasikan pada sistem klien Anda untuk terhubung ke Amazon Athena. Untuk informasi tentang penggunaan kredensi default,

lihatMenggunakan Rantai Penyedia Kredensi Defaultdi dalamAWS SDK for JavaPanduan Pengembang.

Jenis otentikasi

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
AuthenticationType	Diperlukan	`IAM Credentials`	`AuthenticationType=Default Credentials;`

Kredensial eksternal

Kredensil eksternal adalah plugin otentikasi generik yang dapat Anda gunakan untuk terhubung ke penyedia identitas berbasis SALL eksternal. Untuk menggunakan plugin, Anda meneruskan file yang dapat dieksekusi yang mengembalikan respons SALL.

Jenis otentikasi

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
AuthenticationType	Diperlukan	`IAM Credentials`	`AuthenticationType=External Credentials;`

Jalur yang dapat dieksekusi

Jalur ke executable yang memiliki logika penyedia kredensi berbasis SALL kustom Anda. Output dari executable harus respon SAKL diurai dari penyedia identitas.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
ExecutablePath	Diperlukan	none	ExecutablePath= C:\Users*nama pengguna**external_credential.exe*

Daftar argumen

Daftar argumen yang ingin Anda sampaikan ke executable.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
ArgumentList	Opsional	none	ArgumentList= *arg1 arg2 arg3*

Profil instans

Jenis otentikasi ini digunakan pada instans EC2 dan dikirimkan melalui layanan metadata Amazon EC2.

Jenis otentikasi

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
AuthenticationType	Diperlukan	IAM Credentials	AuthenticationType =Instance Profile;

JWT

Plugin JWT (JSON Web Token) menyediakan antarmuka yang menggunakan Token Web JSON untuk mengambil peran Amazon IAM. Konfigurasi tergantung pada penyedia identitas. Untuk

informasi tentang mengonfigurasi federasi untuk Google Cloud danAWS, lihatKonfigurasikan federasi identitas beban kerja denganAWSatau Azuredalam dokumentasi Google Cloud.

Jenis otentikasi

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
AuthenticationType	Diperlukan	`IAM Credentials`	`AuthenticationType=JWT;`

Peran pilihan

Amazon Resource Name (ARN) dari peran yang diambil. Untuk informasi selengkapnya tentang peran ARN, lihatAssumeRoledi dalamAWS Security Token ServiceReferensi API.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
preferred_role	Opsional	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Durasi sesi

Durasi, dalam hitungan detik, dari sesi peran. Untuk informasi selengkapnya tentang durasi sesi, lihatAssumeRoledi dalamAWS Security Token ServiceReferensi API.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
durasi	Opsional	`900`	`duration=900;`

Token web JSON

Token web JSON yang digunakan untuk mengambil kredensi sementara IAM menggunakanAssumeRoleWithWebIdentity AWS STSTindakan API. Untuk informasi tentang

membuat token web JSON untuk pengguna Google Cloud Platform (GCP), lihatMenggunakan token JWT OAuthdalam dokumentasi Google Cloud.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
`web_identity_token`	Diperlukan	`none`	`web_identity_token=eyJhbGc...<remainder of token>;`

Nama sesi peran

Sebuah nama untuk sesi. Teknik yang umum adalah menggunakan nama atau pengenal pengguna aplikasi Anda sebagai nama sesi peran. Ini dengan mudah mengaitkan kredensi keamanan sementara yang digunakan aplikasi Anda dengan pengguna terkait.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
role_session_name	Diperlukan	`none`	`role_session_name=familiarname;`

Okta

Okta adalah plugin otentikasi berbasis SAMP yang bekerja dengan penyedia identitas Okta. Untuk informasi tentang mengonfigurasi federasi untuk Okta dan Amazon Athena, lihat. Mengkonfigurasi SSO untuk ODBC menggunakan plugin Okta dan Penyedia Identitas Okta

Jenis Autentikasi

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
AuthenticationType	Wajib	`IAM Credentials`	`AuthenticationType=Okta;`

ID Pengguna

Nama pengguna Okta Anda.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
UID	Wajib	none	`UID=jane.doe@org.com;`

Kata sandi

Kata sandi pengguna Okta Anda.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
PWD	Wajib	none	`PWD=oktauserpasswordexample;`

Peran yang disukai

Amazon Resource Name (ARN) dari peran yang diambil. Untuk informasi selengkapnya tentang peran ARN, lihat AssumeRoledi Referensi AWS Security Token Service API.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
preferred_role	Opsional	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Durasi sesi

Durasi, dalam hitungan detik, dari sesi peran. Untuk informasi selengkapnya, lihat AssumeRoledi Referensi AWS Security Token Service API.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
durasi	Opsional	`900`	`duration=900;`

Host iDP

URL untuk organisasi Okta Anda. Anda dapat mengekstrak `idp_host` parameter dari URL Embed Link di aplikasi Okta Anda. Untuk langkah, lihat Ambil informasi konfigurasi ODBC dari Okta. Segmen pertama setelah`https://`, hingga dan termasuk `okta.com` adalah host idP Anda (misalnya,`http://trial-1234567.okta.com`).

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
idp_host	Wajib	`None`	`idp_host= dev-99999 999.okta.com;`

Port iDP

Nomor port yang digunakan untuk terhubung ke host IDP Anda.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
idp_port	Wajib	`None`	`idp_port=443;`

ID aplikasi Okta

Pengidentifikasi dua bagian untuk aplikasi Anda. Anda dapat mengekstrak `app_id` parameter dari URL Embed Link di aplikasi Okta Anda. Untuk langkah, lihat Ambil informasi konfigurasi ODBC dari Okta. ID aplikasi adalah dua segmen terakhir dari URL, termasuk garis miring ke depan di tengah. Segmen adalah dua string 20 karakter dengan campuran angka dan huruf besar dan kecil (misalnya,). `Abc1de2fghi3J45kL678/abc1defghij2klmNo3p4`

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
app_id	Wajib	None	app_id=0oa25kx8ze9A3example/alnexamplea0piaWa0g7;

Nama aplikasi Okta

Nama aplikasi Okta.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
app_name	Wajib	None	app_name=amazon_aws_redshift;

Okta waktu tunggu

Menentukan durasi dalam hitungan detik untuk menunggu kode otentikasi multifaktor (MFA).

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
okta_mfa_wait_time	Opsional	10	okta_mfa_wait_time=20;

Jenis MFA Okta

Jenis faktor MFA. Jenis yang didukung adalah Google Authenticator, SMS (Okta), Okta Verify with Push, dan Okta Verify dengan TOTP. Kebijakan keamanan organisasi individu menentukan apakah MFA diperlukan atau tidak untuk login pengguna.

Nama string koneksi	Jenis parameter	Nilai default	Nilai yang mungkin	Contoh string koneksi
okta_mfa_type	Optional	None	googleauthenticator, smsauthentication, oktaverifywithpush, oktaverifywithtotp	okta_mfa_type=oktaverifywithpush;

Nomor telepon Okta

Nomor telepon yang akan digunakan dengan AWS SMS otentikasi. Parameter ini hanya diperlukan untuk pendaftaran multifaktor. Jika nomor ponsel Anda sudah terdaftar, atau jika AWS SMS otentikasi tidak digunakan oleh kebijakan keamanan, Anda dapat mengabaikan bidang ini.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
okta_mfa_phone_number	Diperlukan untuk pendaftaran MFA, opsional sebaliknya	None	okta_mfa_phone_number=19991234567;

Aktifkan cache file Okta

Mengaktifkan cache kredensial sementara. Parameter koneksi ini memungkinkan kredenal sementara untuk di-cache dan digunakan kembali antara beberapa proses yang dibuka oleh aplikasi BI. Gunakan opsi ini untuk menghindari batas pelambatan API Okta.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
okta_cache	Opsional	0	okta_cache=1;

Ping

Ping adalah plugin berbasis SAMP yang bekerja dengan penyedia PingFederateidentitas.

Jenis otentikasi

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
AuthenticationType	Wajib	`IAM Credentials`	`AuthenticationType=Ping;`

ID Pengguna

Nama pengguna untuk PingFederate server.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
UID	Wajib	`none`	`UID=pingusername@domain.com;`

Kata sandi

Kata sandi untuk PingFederate server.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
PWD	Wajib	`none`	`PWD=pingpassword;`

Peran yang disukai

Amazon Resource Name (ARN) dari peran yang diambil. Jika pernyataan SAFL Anda memiliki beberapa peran, Anda dapat menentukan parameter ini untuk memilih peran yang akan diasumsikan. Peran ini harus ada dalam pernyataan SAMP. Untuk informasi selengkapnya tentang peran ARN, lihat AssumeRoledi Referensi AWS Security Token Service API.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
preferred_role	Opsional	none	`preferred_role=arn:aws:iam::123456789012:id/user1;`

Durasi sesi

Durasi, dalam hitungan detik, dari sesi peran. Untuk informasi selengkapnya tentang durasi sesi, lihat AssumeRoledi Referensi AWS Security Token Service API.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
durasi	Opsional	`900`	`duration=900;`

Host iDP

Alamat untuk server Ping Anda. Untuk menemukan alamat Anda, kunjungi URL berikut dan lihat bidang SSO Application Endpoint.

```
https://your-pf-host-#:9999/pingfederate/your-pf-app#/spConnections
```

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
idp_host	Wajib	none	`idp_host=ec2-1-83-65-12.compute-1.amazonaws.com;`

Port iDP

Nomor port yang digunakan untuk terhubung ke host iDP Anda.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
idp_port	Wajib	None	`idp_port=443;`

Mitra SPID

Alamat penyedia layanan. Untuk menemukan alamat penyedia layanan, kunjungi URL berikut dan lihat bidang Titik Akhir Aplikasi SSO.

```
https://your-pf-host-#:9999/pingfederate/your-pf-app#/spConnections
```

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
partner_spid	Wajib	None	`partner_spid=https://us-east-1.signin.aws.amazon.com/platform/saml/<...>;`

Param URI Ping

Melewati argumen URI untuk permintaan otentikasi ke Ping. Gunakan parameter ini untuk melewati batasan peran tunggal Lake Formation. Konfigurasikan Ping untuk mengenali parameter yang diteruskan, dan verifikasi bahwa peran yang diteruskan ada dalam daftar peran yang ditetapkan kepada pengguna. Kemudian, kirim peran tunggal dalam pernyataan SAMP.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
ping_uri_param	Opsional	None	`ping_uri_param=role=my_iam_role;`

Parameter otentikasi umum

Parameter di bagian ini umum untuk jenis otentikasi seperti yang disebutkan.

Gunakan Proxy untuk IdP

Memungkinkan komunikasi antara driver dan IdP melalui proxy. Opsi ini tersedia untuk plugin otentikasi berikut:

- IKLAN FS
- Iklan Azure
- Browser Azure AD
- Peramban SSO OIDC
- JWT
- Okta
- Ping

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
UseProxyForIdP	Opsional	`0`	`UseProxyForIdP=1;`

Gunakan Formasi Danau

MenggunakanAssumeDecoratedRoleWithSAMLTindakan API Lake Formation untuk mengambil kredensi IAM sementara, bukanAssumeRoleWithSAML AWS STSTindakan API. Opsi ini tersedia untuk plugin autentikasi Azure AD, Browser Azure AD, Browser SALL, Okta, Ping, dan AD FS.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
LakeformationEnabled	Opsional	`0`	`LakeformationEnabled=1;`

SSL tidak aman (iDP)

Menonaktifkan SSL saat berkomunikasi dengan IdP. Opsi ini tersedia untuk plugin autentikasi Azure AD, Browser Azure AD, Okta, Ping, dan AD FS.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
SSL_Tidak Aman	Opsional	`0`	`SSL_Insecure=1;`

Pengganti titik akhir

Pengalihan titik akhir Athena

`endpointOverride ClientConfiguration`Kelas menggunakan nilai ini mengganti titik akhir HTTP default untuk klien Amazon Athena. Untuk informasi selengkapnya, lihat Konfigurasi AWS klien di Panduan AWS SDK for C++ Pengembang.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
EndpointOverride	Opsional	none	`EndpointOverride=athena.us-west-2.amazonaws.com;`

Pengalihan titik akhir streaming Athena

`ClientConfiguration.endpointOverride`Metode ini menggunakan nilai ini untuk mengganti titik akhir HTTP default untuk klien streaming Amazon Athena. Untuk informasi selengkapnya, konfigurasi AWS Klien di Panduan AWS SDK for C++ Pengembang. Layanan Streaming Athena tersedia melalui port 444.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
Streaming EndpointOverride	Opsional	none	`StreamingEndpointOverride=athena.us-west-1.amazonaws.com:444;`

AWS STS penggantian titik akhir

`ClientConfiguration.endpointOverride`Metode ini menggunakan nilai ini untuk mengganti titik akhir HTTP default untuk klien. AWS STS Untuk informasi selengkapnya, lihat Konfigurasi AWS klien di Panduan AWS SDK for C++ Pengembang.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
StsEndpointOverride	Opsional	none	`StsEndpointOverride=sts.us-west-1.amazonaws.com;`

Pengalihan titik akhir Lake Formation

`ClientConfiguration.endpointOverride`Metode ini menggunakan nilai ini untuk mengganti titik akhir HTTP default untuk klien Lake Formation. Untuk informasi selengkapnya, lihat Konfigurasi AWS klien di Panduan AWS SDK for C++ Pengembang.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
LakeFormationEndpointOverride	Opsional	none	`LakeFormationEndpointOverride=lakeformation.us-west-1.amazonaws.com;`

Pengalihan titik akhir SSO

`ClientConfiguration.endpointOverride`Metode ini menggunakan nilai ini untuk mengganti titik akhir HTTP default untuk klien SSO. Untuk informasi selengkapnya, lihat Konfigurasi AWS klien di Panduan AWS SDK for C++ Pengembang.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
SSO EndpointOverride	Opsional	none	`SSOEndpointOverride=portal.sso.us-east-2.amazonaws.com;`

Pengalihan titik akhir SSO OIDC

`ClientConfiguration.endpointOverride`Metode ini menggunakan nilai ini untuk mengganti titik akhir HTTP default untuk klien SSO OIDC. Untuk informasi selengkapnya, lihat Konfigurasi AWS klien di Panduan AWS SDK for C++ Pengembang.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
SSOOIDC EndpointOverride	Opsional	none	`SSOOIDCEndpointOverride=oidc.us-east-2.amazonaws.com`

Opsi lanjutan

Ambil ukuran

Jumlah maksimum hasil (baris) untuk kembali dalam permintaan ini. Untuk informasi parameter, lihat GetQuery MaxResults. Untuk API streaming, nilai maksimumnya adalah 10000000.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
RowsToFetchPerBlock	Opsional	`1000`untuk non-streaming `20000`untuk streaming	`RowsToFetchPerBlock=20000;`

Aktifkan penggunaan kembali hasil

Menentukan apakah hasil query sebelumnya dapat digunakan kembali ketika query dijalankan. Untuk informasi parameter, lihat ResultReuseByAgeConfiguration.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
EnableResultReuse	Opsional	`0`	`EnableResultReuse=1;`

Hasil penggunaan kembali usia maksimum

Menentukan, dalam hitungan menit, usia maksimum hasil query sebelumnya yang Athena harus mempertimbangkan untuk digunakan kembali. Untuk informasi parameter, lihat ResultReuseByAgeConfiguration.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
ReusedResultMaxAgeInMinutes	Opsional	`60`	`ReusedResultMaxAgeInMinutes=90;`

Aktifkan API streaming

Memilih apakah akan menggunakan API streaming Athena untuk mengambil set hasil.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
UseResultsetStreaming	Opsional	`0`	`UseResultsetStreaming=1;`

Aktifkan S3 fetcher

Mengambil set hasil yang dihasilkan oleh Athena dari bucket Amazon S3 dengan berinteraksi langsung dengan Amazon S3.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
Aktifkan3Fetcher	Opsional	`1`	`EnableS3Fetcher=1;`

Gunakan beberapa utas S3

Mengambil data dari Amazon S3 menggunakan beberapa utas. Saat opsi ini diaktifkan, file hasil yang disimpan di bucket Amazon S3 diambil secara paralel menggunakan beberapa utas.

Aktifkan opsi ini hanya jika Anda memiliki bandwidth jaringan yang baik. Misalnya, dalam pengukuran kami pada instance EC2 c5.2xlarge, klien S3 single-threaded mencapai 1 Gbps, sementara klien S3 multi-threaded mencapai 4 Gbps throughput jaringan.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
UseMultipleBenang S3	Opsional	`0`	`UseMultipleS3Threa ds=1;`

Gunakan katalog dan skema tunggal

Secara default, driver ODBC meminta Athena untuk mendapatkan daftar katalog dan skema yang tersedia. Opsi ini memaksa pengemudi untuk menggunakan katalog dan skema yang ditentukan oleh kotak dialog konfigurasi Administrator Sumber Data ODBC atau parameter koneksi.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
UseSingleCatalogAn dSchema	Opsional	`0`	`UseSingleCatalogAn dSchema=1;`

Gunakan kueri untuk daftar tabel

Untuk jenis `LAMBDA` katalog, memungkinkan driver ODBC mengirimkan SHOW TABLES kueri untuk mendapatkan daftar tabel yang tersedia. Ini adalah pengaturan default. Jika parameter ini disetel ke 0, driver ODBC menggunakan ListTableMetadataAthena API untuk mendapatkan daftar tabel yang tersedia. Perhatikan bahwa, untuk jenis `LAMBDA` katalog, menggunakan `ListTableMetadata` lead ke regresi kinerja.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
UseQueryToListTabl es	Opsional	`1`	`UseQueryToListTabl es=1;`

Gunakan WCHAR untuk tipe string

Secara default, driver ODBC menggunakan `SQL_CHAR` dan `SQL_VARCHAR` untuk Athena `char` tipe data string`varchar`,,,`string`, `array` `map<>struct<>`, dan. `row` Mengatur parameter ini untuk `1` memaksa driver untuk menggunakan `SQL_WCHAR` dan `SQL_WVARCHAR` untuk tipe data string. Karakter lebar dan tipe karakter variabel lebar digunakan untuk memastikan bahwa karakter dari berbagai bahasa dapat disimpan dan diambil dengan benar.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
UseW CharForSt ringTypes	Opsional	`0`	`UseWCharForStringT ypes=1;`

Kueri katalog eksternal

Menentukan apakah driver perlu query katalog eksternal dari Athena. Untuk informasi selengkapnya, lihat Migrasi ke driver ODBC 2.x.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
QueryExternalCatal ogs	Opsional	`0`	`QueryExte rnalCatal ogs=1;`

Verifikasi SSL

Mengontrol apakah akan memverifikasi sertifikat SSL saat Anda menggunakan AWS SDK. Nilai ini diteruskan ke `ClientConfiguration.verifySSL` parameter. Untuk informasi selengkapnya, lihat Konfigurasi AWS klien di Panduan AWS SDK for C++ Pengembang.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
VerifySSL	Opsional	`1`	`VerifySSL=0;`

Ukuran blok hasil S3

Menentukan, dalam byte, ukuran blok yang akan diunduh untuk satu permintaan API Amazon GetObjectS3. Nilai default adalah 67108864 (64 MB). Nilai minimum dan maksimum yang diizinkan adalah 10485760 (10 MB) dan 2146435072 (sekitar 2 GB).

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
S3 ResultBlockSize	Opsional	`67108864`	`S3ResultBlockSize= 268435456;`

Panjang kolom string

Menentukan panjang kolom untuk kolom dengan tipe `string` data. Karena Athena menggunakan tipe data string Apache Hive, yang tidak memiliki presisi yang ditentukan, panjang default yang dilaporkan oleh Athena adalah 2147483647 (). `INT_MAX` Karena alat BI biasanya mengalokasikan memori untuk kolom, ini dapat menyebabkan konsumsi memori yang tinggi. Untuk menghindari hal ini, driver Athena ODBC membatasi presisi yang dilaporkan untuk kolom tipe `string` data dan mengekspos parameter `StringColumnLength` koneksi sehingga nilai default dapat diubah.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
StringColumnLength	Opsional	255	`StringColumnLength =65535;`

Panjang kolom tipe kompleks

Menentukan panjang kolom untuk kolom dengan tipe data yang kompleks sepertimap,`struct`, dan`array`. Seperti StringColumnLength, Athena melaporkan 0 presisi untuk kolom dengan tipe data yang kompleks. Driver Athena ODBC menetapkan presisi default untuk kolom dengan tipe data yang kompleks dan mengekspos parameter `ComplexTypeColumnLength` koneksi sehingga nilai default dapat diubah.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
ComplexTy peColumnLength	Opsional	65535	`ComplexTypeColumnL ength=123456;`

Sertifikat CA tepercaya

Menginstruksikan klien HTTP di mana menemukan toko kepercayaan sertifikat SSL Anda. Nilai ini diteruskan ke `ClientConfiguration.caFile` parameter. Untuk informasi selengkapnya, lihat Konfigurasi AWS klien di Panduan AWS SDK for C++ Pengembang.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
TrustedCerts	Opsional	`%INSTALL_PATH%/bin`	`TrustedCerts=C:\\Program Files\\Amazon Athena ODBC Driver\\bin\\cacert.pem;`

Periode jajak pendapat min

Menentukan nilai minimum dalam milidetik untuk menunggu sebelum polling Athena untuk status eksekusi query.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
MinQueryExecutionPollingInterval	Opsional	`100`	`MinQueryExecutionPollingInterval=200;`

Periode jajak pendapat maks

Menentukan nilai maksimum dalam milidetik untuk menunggu sebelum polling Athena untuk status eksekusi query.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
MaxQueryExecutionPollingInterval	Opsional	`60000`	`MaxQueryExecutionPollingInterval=1000;`

Pengganda jajak pendapat

Menentukan faktor untuk meningkatkan periode polling. Secara default, polling dimulai dengan nilai periode polling min dan berlipat ganda dengan setiap polling hingga mencapai nilai periode polling maksimal.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
QueryExecutionPollingIntervalMultiplier	Opsional	2	`QueryExecutionPollingIntervalMultiplier=2;`

Durasi jajak pendapat maks

Menentukan nilai maksimum dalam milidetik bahwa pengemudi dapat polling Athena untuk status eksekusi query.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
MaxPollDuration	Opsional	`1800000`	`MaxPollDuration=1800000;`

Batas waktu koneksi

Jumlah waktu (dalam milidetik) koneksi HTTP menunggu untuk membuat koneksi. Nilai ini ditetapkan untuk klien `ClientConfiguration.connectTimeoutMs` Athena. Jika tidak ditentukan, nilai default curl digunakan. Untuk informasi tentang parameter koneksi, lihat Konfigurasi Klien di Panduan AWS SDK for Java Pengembang.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
ConnectionTimeout	Opsional	`0`	`ConnectionTimeout=2000;`

Batas waktu permintaan

Menentukan batas waktu baca soket untuk klien HTTP. Nilai ini ditetapkan untuk `ClientConfiguration.requestTimeoutMs` parameter klien Athena. Untuk informasi parameter, lihat Konfigurasi Klien di Panduan AWS SDK for Java Pengembang.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
RequestTimeout	Opsional	`10000`	`RequestTimeout=30000;`

Opsi proxy

Tuan rumah proksi

Jika Anda mengharuskan pengguna untuk pergi melalui proxy, gunakan parameter ini untuk mengatur host proxy. Parameter ini sesuai dengan`ClientConfiguration.proxyHost`parameter dalamAWSSDK. Untuk informasi lebih lanjut, lihatAWSKonfigurasi klendi dalamAWS SDK for C++Panduan Pengembang.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
ProxyHost	Opsional	`none`	`ProxyHost=127.0.0.1;`

Pelabuhan proxy

Gunakan parameter ini untuk mengatur port proxy. Parameter ini sesuai dengan`ClientConfiguration.proxyPort`parameter dalamAWSSDK. Untuk informasi lebih lanjut, lihatAWSKonfigurasi klendi dalamAWS SDK for C++Panduan Pengembang.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
ProxyPort	Opsional	`none`	`ProxyPort=8888;`

Nama pengguna proxy

Gunakan parameter ini untuk mengatur nama pengguna proxy. Parameter ini sesuai dengan`ClientConfiguration.proxyUserName`parameter dalamAWSSDK. Untuk informasi lebih lanjut, lihatAWSKonfigurasi klendi dalamAWS SDK for C++Panduan Pengembang.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
ProxyUID	Opsional	none	`ProxyUID=username;`

Kata sandi proxy

Gunakan parameter ini untuk mengatur kata sandi proxy. Parameter ini sesuai dengan`ClientConfiguration.proxyPassword`parameter dalamAWSSDK. Untuk informasi lebih lanjut, lihatAWSKonfigurasi klendi dalamAWS SDK for C++Panduan Pengembang.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
ProxyPWD	Opsional	none	`ProxyPWD=password;`

Host non proxy

Gunakan parameter opsional ini untuk menentukan host yang dihubungkan oleh driver tanpa menggunakan proxy. Parameter ini sesuai dengan`ClientConfiguration.nonProxyHosts`parameter dalamAWSSDK. Untuk informasi lebih lanjut, lihatAWSKonfigurasi klendi dalamAWS SDK for C++Panduan Pengembang.

Yang`NonProxyHost`parameter koneksi dilewatkan ke`CURLOPT_NOPROXY`opsi curl. Untuk informasi tentang`CURLOPT_NOPROXY`format, lihatCURLOPT_NOPROXYdalam dokumentasi curl.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
NonProxyHost	Opsional	none	`NonProxyHost=.amazonaws.com,localhost,.example.net,.example.com;`

Gunakan proxy

Memungkinkan lalu lintas pengguna melalui proxy yang ditentukan.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
UseProxy	Opsional	none	`UseProxy=1;`

Opsi pencatatan

Hak administrator diperlukan untuk mengubah pengaturan yang dijelaskan di sini. Untuk membuat perubahan, Anda dapat menggunakan Administrator Sumber Data ODBCOpsi Loggingkotak dialog atau memodifikasi registri Windows secara langsung.

Tingkat log

Opsi ini memungkinkan log driver ODBC. Di Windows, Anda dapat menggunakan registri atau kotak dialog untuk mengaktifkan atau menonaktifkan logging. Opsi ini terletak di jalur registri berikut:

```
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Amazon Athena\ODBC\Driver
```

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
LogLevel	Opsional	`0`	`LogLevel=1;`

Jalur log

Menentukan path ke file di mana log driver ODBC disimpan. Anda dapat menggunakan registri atau kotak dialog untuk mengatur nilai ini. Opsi ini terletak di jalur registri berikut:

```
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Amazon Athena\ODBC\Driver
```

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
LogPath	Opsional	none	`LogPath=C:\Users\ username\projects \internal\trunk\;`

GunakanAWSLogger

Menentukan jikaAWSPencatatan SDK diaktifkan. Tentukan 1 untuk mengaktifkan, 0 untuk menonaktifkan.

Nama string koneksi	Jenis parameter	Nilai default	Contoh string koneksi
UseAwsLogger	Opsional	`0`	`UseAwsLogger=1;`

Migrasi ke driver ODBC 2.x

Karena sebagian besar parameter koneksi Athena ODBC 2.x yang kompatibel dengan driver ODBC 1.x, Anda dapat menggunakan kembali sebagian besar string koneksi yang ada dengan driver Athena ODBC 2.x. Namun, parameter koneksi berikut memerlukan modifikasi.

Tingkat log

Sementara driver ODBC saat ini menyediakan berbagai opsi logging yang tersedia, mulai dari`LOG_OFF (0)`kepada`LOG_TRACE (6)`, driver Amazon Athena ODBC hanya memiliki dua nilai: 0 (dinonaktifkan) dan 1 (diaktifkan).

Untuk informasi selengkapnya tentang pencatatan driver ODBC 2.x, lihatOpsi pencatatan.

	ODBC 1.x driver	ODBC 2.x driver
Nama string koneksi	`LogLevel`	`LogLevel`
Jenis parameter	Opsional	Opsional
Nilai default	`0`	`0`
Nilai yang mungkin	`0-6`	`0,1`
Contoh string koneksi	`LogLevel=6;`	`LogLevel=1;`

MetadataRetrievalMethod

Driver ODBC saat ini menyediakan beberapa opsi untuk mengambil metadata dari Athena. Pengemudi Amazon Athena ODBC menghentikan`MetadataRetrievalMethod`dan selalu menggunakan Amazon Athena API untuk mengekstrak metadata.

Athena memperkenalkan bendera`QueryExternalCatalogs`untuk query katalog eksternal. Untuk query katalog eksternal dengan driver ODBC saat ini, set`MetadataRetrievalMethod`kepada`ProxyAPI`. Untuk query katalog eksternal dengan driver Athena ODBC, set`QueryExternalCatalogs`kepada`1`.

	ODBC 1.x driver	ODBC 2.x driver
Nama string koneksi	`MetadataRetrievalMethod`	`QueryExternalCatalogs`
Jenis parameter	Opsional	Opsional
Nilai default	`Auto`	`0`
Nilai yang mungkin	`Auto`, `AWS Glue`, `ProxyAPI`, `Query`	`0`,`1`
Contoh string koneksi	`MetadataRetrievalMethod=ProxyAPI;`	`QueryExternalCatalogs=1;`

Tes koneksi

Saat Anda menguji koneksi driver ODBC 1.x, driver menjalankan`SELECT 1`kueri yang menghasilkan dua file di bucket Amazon S3 Anda: satu untuk set hasil, dan satu untuk metadata. Koneksi uji dibebankan sesuai dengan[Harga Amazon Athena](#)kebijakan.

Saat Anda menguji koneksi driver ODBC 2.x, driver memanggil[GetWorkGroup](#)Aksi API Athena. Panggilan menggunakan jenis otentikasi dan penyedia kredensi terkait yang Anda tentukan untuk mengambil kredensialnya. Tidak ada biaya untuk pengujian koneksi saat Anda menggunakan driver ODBC 2.x, dan pengujian tidak menghasilkan hasil kueri di bucket Amazon S3 Anda.

Memecahkan masalah driver ODBC 2.x

Jika Anda mengalami masalah dengan driver Amazon Athena ODBC, Anda dapat menghubungiAWS Support(dalamAWS Management Console, pilihDukungan,Pusat Dukungan).

Pastikan untuk menyertakan informasi berikut, dan berikan detail tambahan yang akan membantu tim dukungan memahami kasus penggunaan Anda.

- Deskripsi- (Diperlukan) Deskripsi yang mencakup informasi terperinci tentang kasus penggunaan Anda dan perbedaan antara perilaku yang diharapkan dan diamati. Sertakan informasi apa pun yang dapat membantu teknisi dukungan menavigasi masalah dengan mudah. Jika masalah terputus-putus, tentukan tanggal, stempel waktu, atau titik interval saat masalah terjadi.
- Informasi versi— (Diperlukan) Informasi tentang versi driver, sistem operasi, dan aplikasi yang Anda gunakan. Misalnya, “Driver ODBC versi 1.2.3, Windows 10 (x64), Power BI.”
- File log- (Diperlukan) Jumlah minimum file log driver ODBC yang diperlukan untuk memahami masalah ini. Untuk informasi tentang opsi logging untuk driver ODBC 2.x, lihatOpsi pencatatan.
- String koneksi- (Diperlukan) String koneksi ODBC Anda atau tangkapan layar kotak dialog yang menunjukkan parameter koneksi yang Anda gunakan. Untuk informasi tentang parameter koneksi, lihatParameter koneksi Athena ODBC 2.x.
- Langkah masalah- (Opsional) Jika memungkinkan, sertakan langkah-langkah atau program mandiri yang dapat membantu mereproduksi masalah.
- Informasi galat kueri- (Opsional) Jika Anda memiliki kesalahan yang melibatkan DML/DDL query, termasuk informasi berikut:
  - Sebuah versi lengkap atau disederhanakan dari DML-gagal atau DDL query.
  - ID akun danWilayah AWSdigunakan, dan ID eksekusi query.
- Kesalahan SALL- (Opsional) Jika Anda memiliki masalah terkait otentikasi dengan pernyataan SALL, sertakan informasi berikut:
  - Penyedia identitas dan plugin otentikasi yang digunakan.
  - Contoh dengan token SALL.

Catatan rilis Amazon Athena ODBC 2.x

Catatan rilis ini memberikan detail penyempurnaan, fitur, masalah yang diketahui, dan perubahan alur kerja di driver Amazon Athena ODBC 2.x.

2.0.3.0

Dirilis 2024-04-08

Driver Amazon Athena ODBC v2.0.3.0 berisi perbaikan dan perbaikan berikut.

Perbaikan

- Menambahkan dukungan MFA untuk plugin otentikasi Okta di platform Linux dan Mac.
- Baik `athena-odbc.dll` perpustakaan dan `AmazonAthenaODBC-2.x.x.x.msi` penginstal untuk Windows sekarang ditandatangani.
- Memperbarui `cacert.pem` file sertifikat CA yang diinstal dengan driver.
- Meningkatkan waktu yang dibutuhkan untuk daftar tabel di bawah katalog Lambda. Untuk jenis `LAMBDA` katalog, driver ODBC sekarang dapat mengirimkan SHOW TABLES kueri untuk mendapatkan daftar tabel yang tersedia. Untuk informasi selengkapnya, lihat Gunakan kueri untuk daftar tabel.
- Memperkenalkan parameter `UseWCharForStringTypes` koneksi untuk melaporkan tipe data string menggunakan `SQL_WCHAR` dan`SQL_WVARCHAR`. Untuk informasi selengkapnya, lihat Gunakan WCHAR untuk tipe string.

Perbaikan

- Memperbaiki peringatan korupsi registri yang terjadi saat OdbcDsn PowerShell alat Get- digunakan.
- Memperbarui logika parsing untuk menangani komentar di awal string kueri.
- Tipe data tanggal dan stempel waktu sekarang memungkinkan nol di bidang tahun.

Untuk mengunduh driver ODBC v2 baru, lihatUnduhan driver ODBC 2.x. Untuk informasi koneksi, lihatAmazon Athena ODBC 2.x.

2.0.2.2

Dirilis 2024-02-13

Driver Amazon Athena ODBC v2.0.2.2 berisi perbaikan dan perbaikan berikut.

Perbaikan

- Ditambahkan dua parameter koneksi, `StringColumnLength` dan`ComplexTypeColumnLength`, yang dapat Anda gunakan untuk mengubah panjang kolom default untuk string dan tipe data yang kompleks. Untuk informasi selengkapnya, lihat Panjang kolom string dan Panjang kolom tipe kompleks.
- Support telah ditambahkan untuk sistem operasi Linux dan macOS (Intel dan ARM). Untuk informasi selengkapnya, lihat Linux dan macOS.
- AWS-SDK-CPP telah diperbarui ke versi tag 1.11.245.
- Perpustakaan curl telah diperbarui ke versi 8.6.0.

Perbaikan

- Menyelesaikan masalah yang menyebabkan nilai yang salah dilaporkan dalam metadata kumpulan hasil untuk tipe data seperti string di kolom presisi.

Untuk mengunduh driver ODBC v2, lihatUnduhan driver ODBC 2.x. Untuk informasi koneksi, lihatAmazon Athena ODBC 2.x.

2.0.2.1

Dirilis 2023-12-07

Driver Amazon Athena ODBC v2.0.2.1 berisi perbaikan dan perbaikan berikut.

Perbaikan

- Peningkatan keamanan thread driver ODBC untuk semua antarmuka.
- Saat logging diaktifkan, nilai datetime sekarang direkam dengan presisi milidetik.
- Selama otentikasi dengan Browser SSO OIDC plugin, terminal sekarang terbuka untuk menampilkan kode perangkat kepada pengguna.

Perbaikan

- Menyelesaikan masalah rilis memori yang terjadi saat mengurai hasil dari API streaming.

- Permintaan untuk antarmukaSQLTablePrivileges(),SQLSpecialColumns(),SQLProcedureColumns(), dan SQLProcedures() sekarang mengembalikan set hasil kosong.

Untuk mengunduh driver ODBC v2, lihatUnduhan driver ODBC 2.x. Untuk informasi koneksi, lihatAmazon Athena ODBC 2.x.

2.0.2.0

Dirilis 2023-10-17

Driver Amazon Athena ODBC v2.0.2.0 berisi perbaikan dan perbaikan berikut.

Perbaikan

- Fitur cache file ditambahkan untuk Browser Azure AD, Browser SSO OIDC, dan plugin otentikasi berbasis browser Okta.

  BI Tools seperti Power BI dan plugin berbasis browser menggunakan beberapa jendela browser. Parameter koneksi cache file baru memungkinkan kredensil sementara untuk di-cache dan digunakan kembali antara beberapa proses yang dibuka oleh aplikasi BI.
- Aplikasi sekarang dapat meminta informasi tentang hasil yang ditetapkan setelah pernyataan disiapkan.
- Koneksi default dan batas waktu permintaan telah ditingkatkan untuk digunakan dengan jaringan klien yang lebih lambat. Untuk informasi selengkapnya, lihat Batas waktu koneksi dan Batas waktu permintaan.
- Penggantian titik akhir telah ditambahkan untuk SSO dan SSO OIDC. Untuk informasi selengkapnya, lihat Pengganti titik akhir.
- Menambahkan parameter koneksi untuk meneruskan argumen URI untuk permintaan otentikasi ke Ping. Anda dapat menggunakan parameter ini untuk melewati batasan peran tunggal Lake Formation. Untuk informasi selengkapnya, lihat Param URI Ping.

Perbaikan

- Memperbaiki masalah luapan integer yang terjadi saat menggunakan mekanisme pengikatan berbasis baris.
- Batas waktu dihapus dari daftar parameter koneksi yang diperlukan untuk plugin otentikasi Browser SSO OIDC.

- Menambahkan antarmuka yang hilang untuk `SQLStatistics()SQLPrimaryKeys()`,`SQLForeignKeys()`,, dan`SQLColumnPrivileges()`, dan menambahkan kemampuan untuk mengembalikan set hasil kosong berdasarkan permintaan.

Untuk mengunduh driver ODBC v2 baru, lihatUnduhan driver ODBC 2.x. Untuk informasi koneksi, lihatAmazon Athena ODBC 2.x.

2.0.1.1

Dirilis 2023-08-10

Driver Amazon Athena ODBC v2.0.1.1 berisi perbaikan dan perbaikan berikut.

Perbaikan

- Menambahkan pencatatan URI ke plugin otentikasi Okta.
- Menambahkan parameter peran yang disukai ke plugin penyedia kredensyal eksternal.
- Menambahkan penanganan untuk awalan profil di nama profil file AWS konfigurasi.

Perbaikan

- Memperbaiki masalah Wilayah AWS penggunaan yang terjadi saat bekerja dengan Lake Formation dan AWS STS klien.
- Memulihkan kunci partisi yang hilang ke daftar kolom tabel.
- Menambahkan jenis `BrowserSSOOIDC` otentikasi yang hilang ke AWS profil.

Untuk mengunduh driver ODBC v2 baru, lihatUnduhan driver ODBC 2.x.

2.0.1.0

Dirilis 2023-06-29

Amazon Athena merilis driver ODBC v2.0.1.0.

Athena telah merilis driver ODBC baru yang meningkatkan pengalaman menghubungkan, menanyakan, dan memvisualisasikan data dari pengembangan SQL yang kompatibel dan aplikasi intelijen bisnis. Versi terbaru dari driver Athena ODBC mendukung fitur driver yang ada dan mudah

untuk ditingkatkan. Versi baru mencakup dukungan untuk mengautentikasi pengguna melalui AWS IAM Identity Center. Ini juga menawarkan opsi untuk membaca hasil kueri dari Amazon S3, yang dapat membuat hasil kueri tersedia untuk Anda lebih cepat.

Untuk informasi selengkapnya, lihat Amazon Athena ODBC 2.x.

Athena ODBC 1.x driver

Gunakan tautan di halaman ini untuk mengunduh Perjanjian Lisensi Pengemudi Amazon Athena 1.x ODBC, driver ODBC, dan dokumentasi ODBC. Untuk informasi tentang string koneksi ODBC, lihat file PDF Panduan Instalasi dan Konfigurasi Driver ODBC, yang dapat diunduh dari halaman ini. Untuk informasi izin, lihat Akses melalui koneksi JDBC dan ODBC.

Important

Saat Anda menggunakan driver ODBC 1.x, pastikan untuk mencatat persyaratan berikut:

- Buka port 444 - Simpan port 444, yang digunakan Athena untuk mengalirkan hasil kueri, terbuka untuk lalu lintas keluar. Saat Anda menggunakan PrivateLink titik akhir untuk terhubung ke Athena, pastikan grup keamanan yang terpasang pada titik akhir terbuka untuk PrivateLink lalu lintas masuk di port 444.
- athena: GetQueryResultsStream policy — Tambahkan tindakan `athena:GetQueryResultsStream` kebijakan ke kepala sekolah IAM yang menggunakan driver ODBC. Tindakan dasar ini tidak terdedah secara langsung dengan API. Ini hanya digunakan dengan driver ODBC dan JDBC sebagai bagian dari dukungan hasil streaming. Untuk contoh kebijakan, lihat AWS kebijakan terkelola: AWSQuicksightAthenaAccess.

Windows

Versi Driver	Tautan unduhan
ODBC 1.2.3.1000 untuk Windows 32-bit	Driver ODBC Windows 32 bit 1.2.3.1000 Driver ODBC bit 1.2.3.1000
ODBC 1.2.3.1000 untuk Windows 64-bit	Driver ODBC Windows 64 bit 1.2.3.1000 Driver ODBC bit 1.2.3.1000

Linux

Versi Driver	Tautan unduhan
ODBC 1.2.3.1000 untuk Linux 32-bit	Linux 32 bit ODBC driver 1.2.3.1000 bit ODBC driver 1.2.3.1000
ODBC 1.2.3.1000 untuk Linux 64-bit	Linux 64 bit ODBC driver 1.2.3.1000 bit ODBC driver 1.2.3.1000

OSX

Versi Driver	Tautan unduhan
ODBC 1.2.3.1000 untuk OSX	

Dokumentasi

Daftar isi	Tautan dokumentasi
Perjanjian SIM Amazon Athena ODBC	Perjanjian lisensi Perjanjian
Dokumentasi untuk ODBC 1.2.3.1000	Panduan instalasi dan konfigurasi driver ODBC versi 1.2.3.1000 Panduan versi 1.2.3.1000
Catatan Rilis untuk ODBC 1.2.3.1000	Catatan rilis driver ODBC versi 1.2.3.1000 Catatan rilis driver versi 1.2.3.1000

Catatan pengemudi ODBC

Menghubungkan Tanpa Menggunakan Proxy

Jika Anda ingin menentukan host tertentu yang terhubung ke driver tanpa menggunakan proxy, Anda dapat menggunakan `NonProxyHost` properti opsional dalam string koneksi ODBC Anda.

`NonProxyHost`Properti menentukan daftar host yang dipisahkan koma yang dapat diakses konektor tanpa melalui server proxy ketika koneksi proxy diaktifkan, seperti pada contoh berikut:

```
.amazonaws.com,localhost,.example.net,.example.com
```

Parameter `NonProxyHost` koneksi diteruskan ke opsi `CURLOPT_NOPROXY` curl. Untuk informasi tentang `CURLOPT_NOPROXY` format, lihat CURLOPT_NOPROXY dalam dokumentasi curl.

Mengkonfigurasi akses federasi ke Amazon Athena untuk pengguna Microsoft AD FS menggunakan klien ODBC

Untuk mengatur akses federasi ke Amazon Athena untuk pengguna Microsoft Active Directory Federation Services (AD FS) menggunakan klien ODBC, pertama-tama Anda membangun kepercayaan antara AD FS danAWSakun. Dengan kepercayaan ini di tempat, pengguna AD Anda dapatfederasikeAWSmenggunakan kredensi AD mereka dan menganggap izin dariAWS Identity and Access Management(IAM) peran untuk mengaksesAWSsumber daya seperti Athena API.

Untuk membuat kepercayaan ini, Anda menambahkan AD FS sebagai penyedia SALL keAkun AWSdan membuat peran IAM yang dapat diasumsikan oleh pengguna federasi. Di sisi AD FS, Anda menambahkanAWSsebagai pihak yang mengandalkan dan menulis aturan klaim SALL untuk mengirim atribut pengguna yang tepatAWSuntuk otorisasi (khususnya, Athena dan Amazon S3).

Mengkonfigurasi akses AD FS ke Athena melibatkan langkah-langkah utama berikut:

1. Menyiapkan penyedia dan peran IAM SALL

2. Mengkonfigurasi AD FS

3. Membuat pengguna dan grup Active Directory

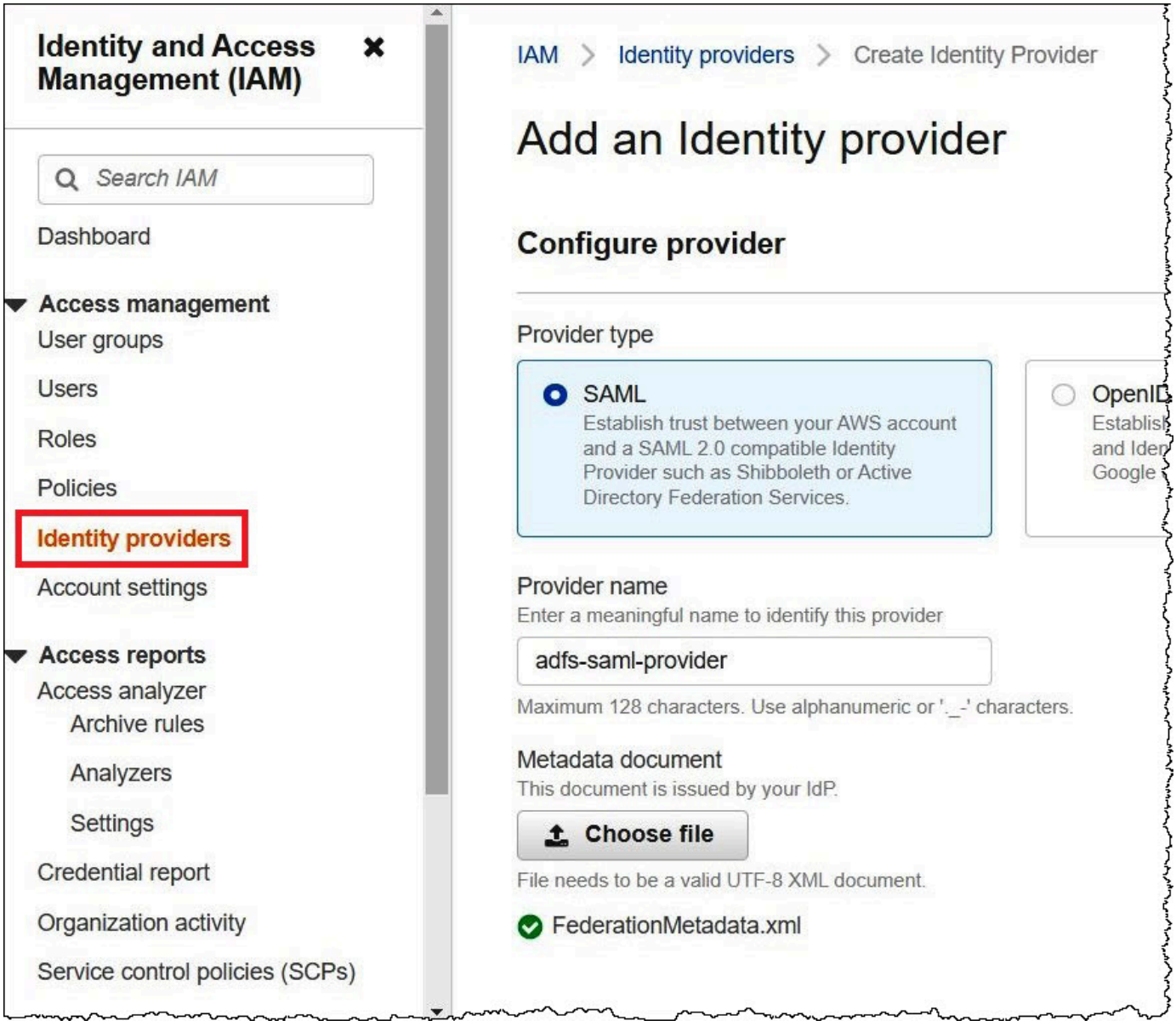
4. Mengkonfigurasi koneksi AD FS ODBC ke Athena

1. Menyiapkan penyedia dan peran IAM SALL

Di bagian ini, Anda menambahkan AD FS sebagai penyedia SALL keAWSakun dan buat peran IAM yang dapat diasumsikan oleh pengguna federasi Anda.

Untuk menyiapkan penyedia SALL

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Penyedia identitas.
3. PilihTambah penyedia.
4. UntukJenis penyedia, pilihSAML.



5. UntukNama penyedia, masuklah**`adfs-saml-provider`**.
6. Di browser, masukkan alamat berikut untuk mengunduh file XMLfederasi untuk server AD FS Anda. Untuk melakukan langkah ini, browser Anda harus memiliki akses ke server AD FS.

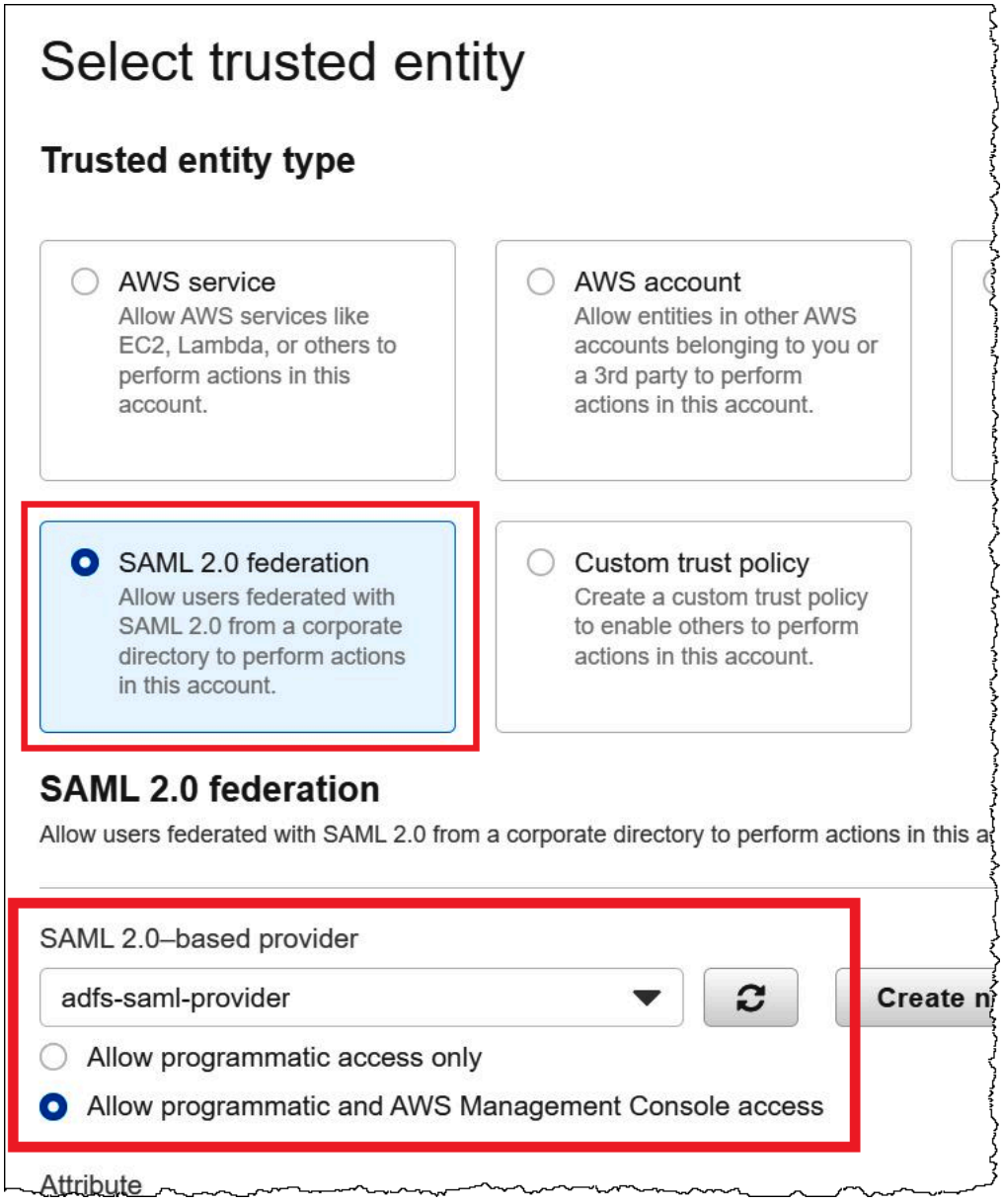
```
https://adfs-server-name/federationmetadata/2007-06/federationmetadata.xml
```

7. Di konsol IAM, untukDokumen metadata, pilihPilih file, dan kemudian unggah file metadata federasi keAWS.
8. Untuk menyelesaikan, pilihTambah penyedia.

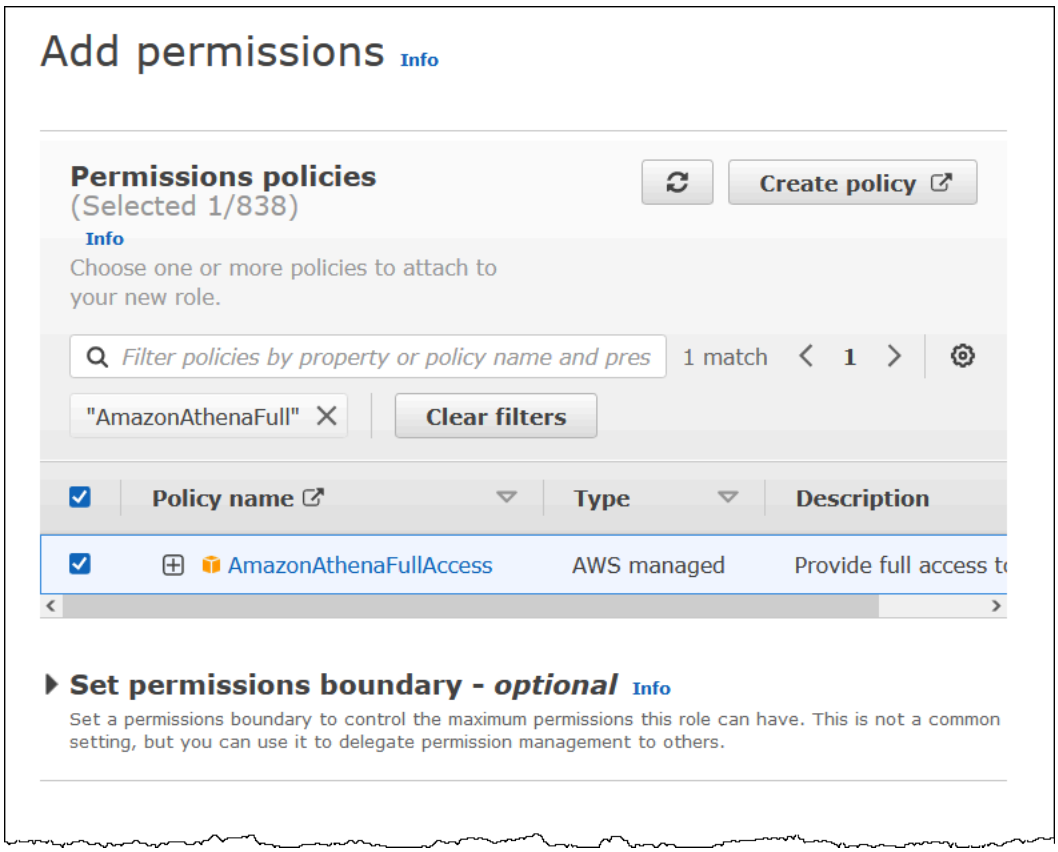
Selanjutnya, Anda membuat peran IAM yang dapat diasumsikan oleh pengguna federasi Anda.

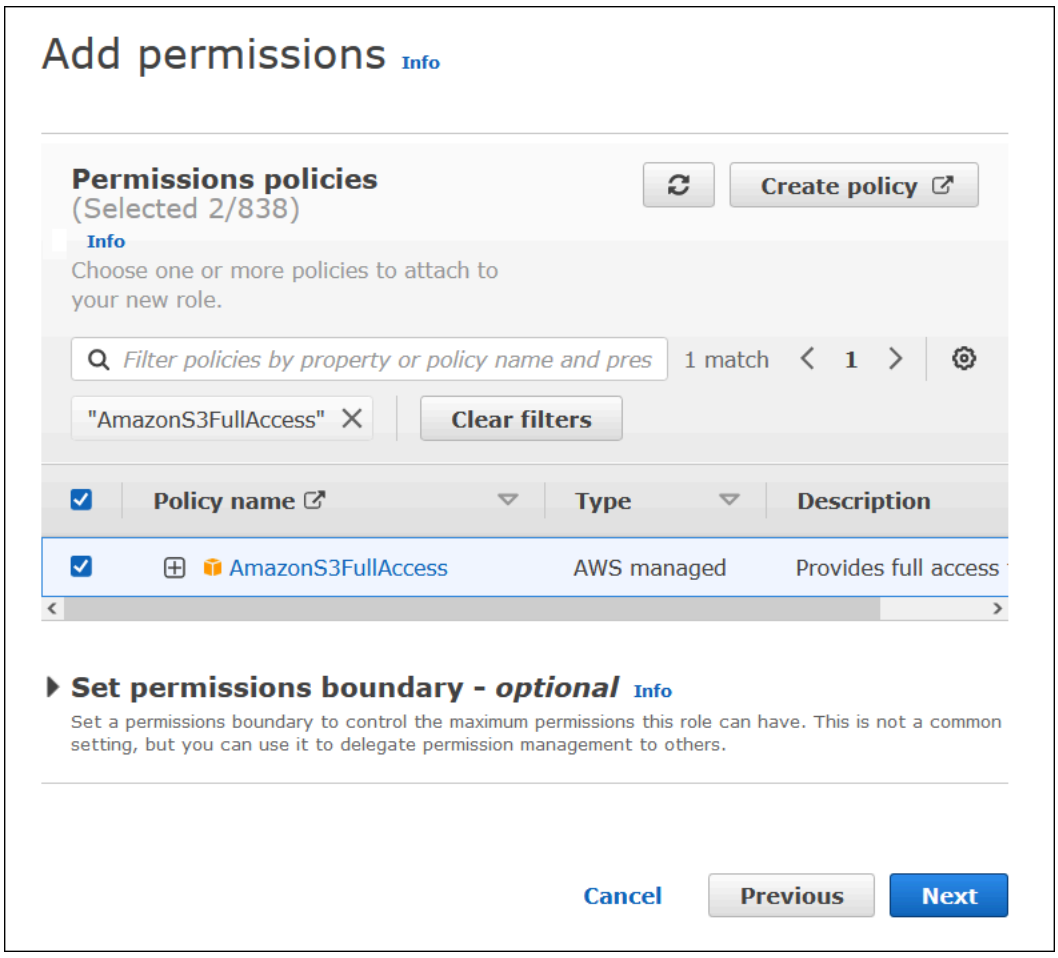
Untuk membuat peran IAM bagi pengguna federasi

1. Di panel navigasi konsol IAM, pilihPeran.
2. Pilih Create role (Buat peran).
3. UntukJenis entitas tepercaya, pilihSALL 2.0 federasi.
4. UntukPenyedia berbasis SALL 2.0, pilihadfs-saml-providerpenyedia yang Anda buat.
5. PilihIzinkan program danAWSAkses Konsol Manajemen, lalu pilihBerikutnya.



6. PadaTambahkan izinhalaman, filter kebijakan izin IAM yang Anda perlukan untuk peran ini, lalu pilih kotak centang yang sesuai. Tutorial ini melampirkan`AmazonAthenaFullAccess`dan`AmazonS3FullAccess`kebijakan.


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.

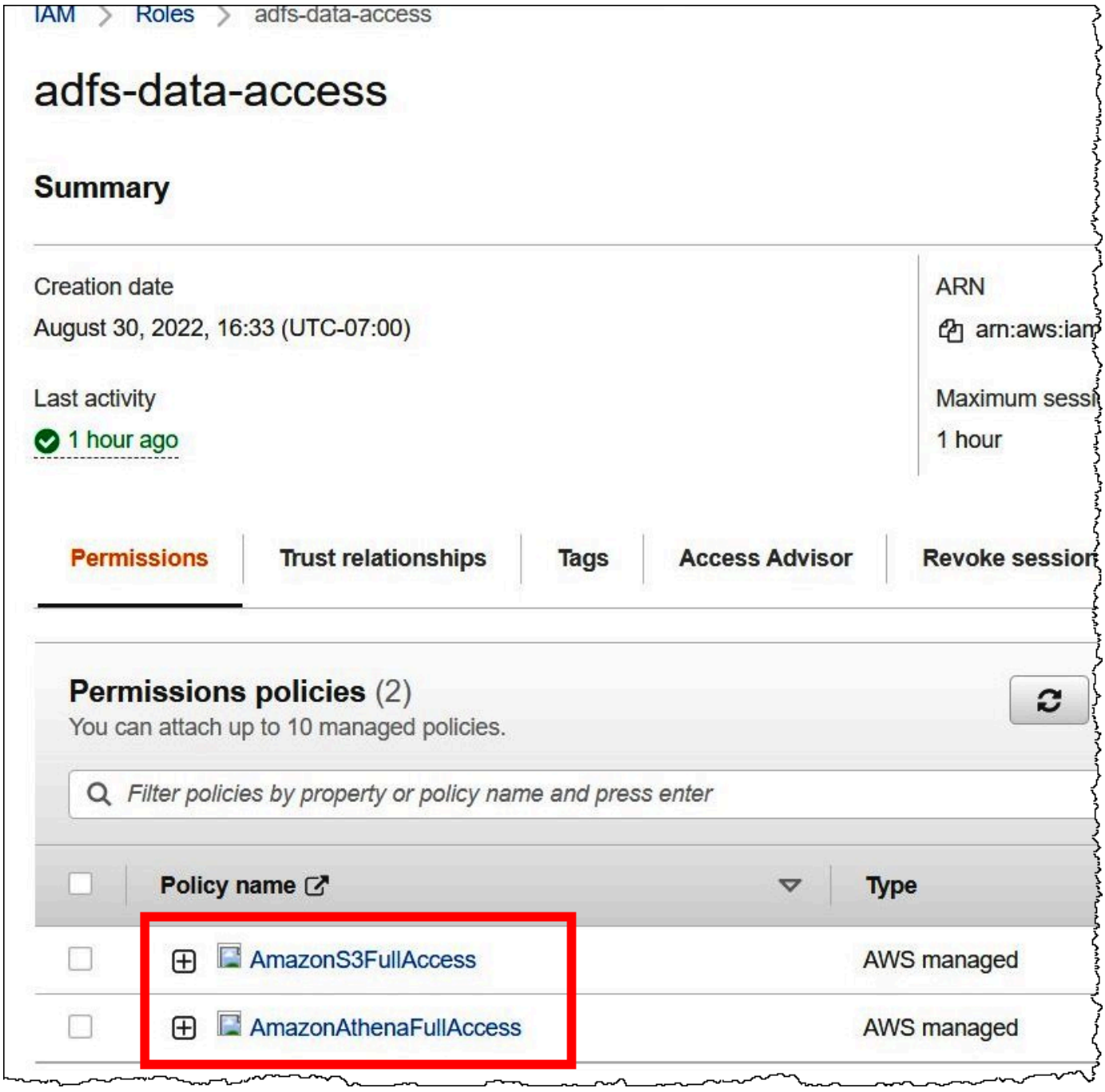


7. Pilih Selanjutnya.
8. PadaNama, ulasan, dan buathalaman, untukNama peran, masukkan nama untuk peran tersebut. Tutorial ini menggunakan namaadfs-data-access.

   DalamLangkah 1: Pilih entitas tepercaya, yangKepala Sekolahbidang harus secara otomatis diisi dengan`"Federated:" "arn:aws:iam::account_id:saml-provider/adfs-saml-provider"`. Yang`Condition`bidang harus berisi`"SAML:aud"`dan`"https://signin.aws.amazon.com/saml"`.

Langkah 2: Tambahkan izinmenunjukkan kebijakan yang telah Anda lampirkan pada peran tersebut.

9. Pilih Create role (Buat peran). Pesan spanduk mengonfirmasi pembuatan peran.
10. PadaPeranhalaman, pilih nama peran yang baru saja Anda buat. Halaman ringkasan untuk peran menunjukkan kebijakan yang telah dilampirkan.



2. Mengkonfigurasi AD FS

Sekarang Anda siap untuk menambahkanAWSsebagai pihak yang mengandalkan dan menulis aturan klaim SALL sehingga Anda dapat mengirim atribut pengguna yang tepatAWSuntuk otorisasi.

Federasi berbasis SALL memiliki dua pihak peserta: IDP (Active Directory) dan pihak yang mengandalkan (AWS), yang merupakan layanan atau aplikasi yang menggunakan otentikasi dari IdP.

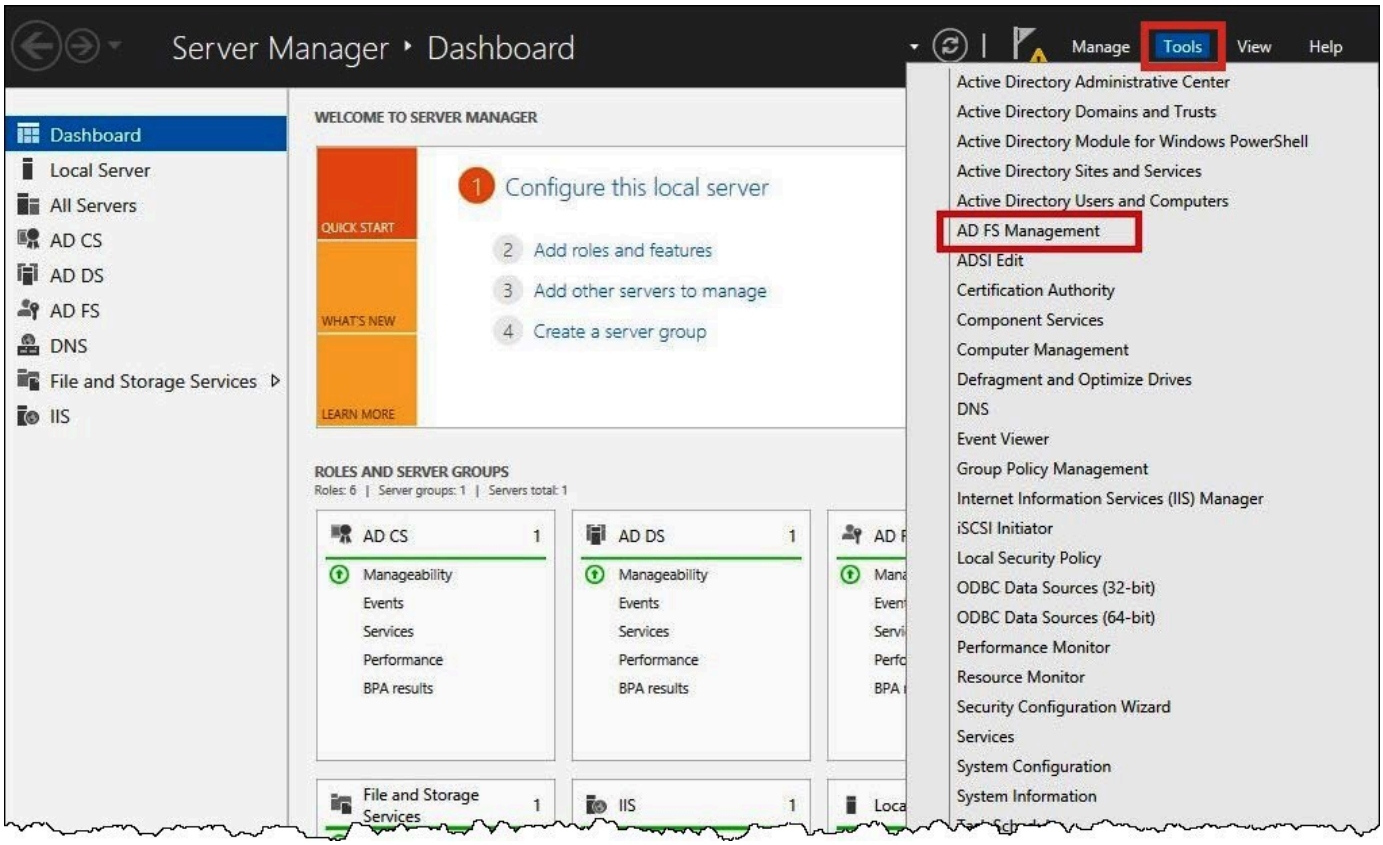
Untuk mengonfigurasi AD FS, pertama-tama Anda menambahkan kepercayaan pihak yang mengandalkan, lalu mengonfigurasi aturan klaim SALL untuk pihak yang mengandalkan. AD FS menggunakan aturan klaim untuk membentuk pernyataan SALL yang dikirim ke pihak yang mengandalkan. Pernyataan SALL menyatakan bahwa informasi tentang pengguna AD benar, dan bahwa itu telah diautentikasi pengguna.

Menambahkan kepercayaan partai yang mengandalkan

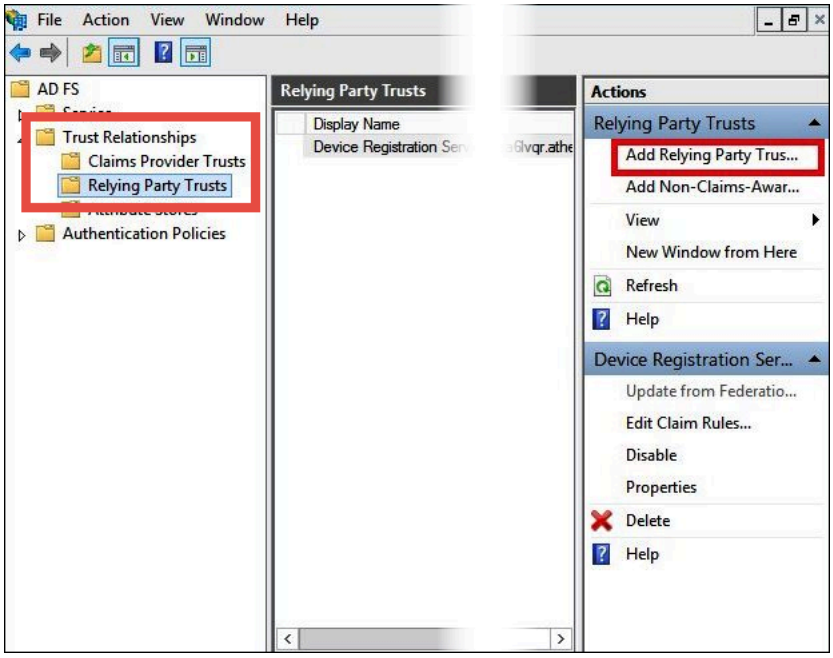
Untuk menambahkan kepercayaan pihak yang mengandalkan pada AD FS, Anda menggunakan manajer server AD FS.

Untuk menambahkan kepercayaan pihak yang mengandalkan pada AD FS

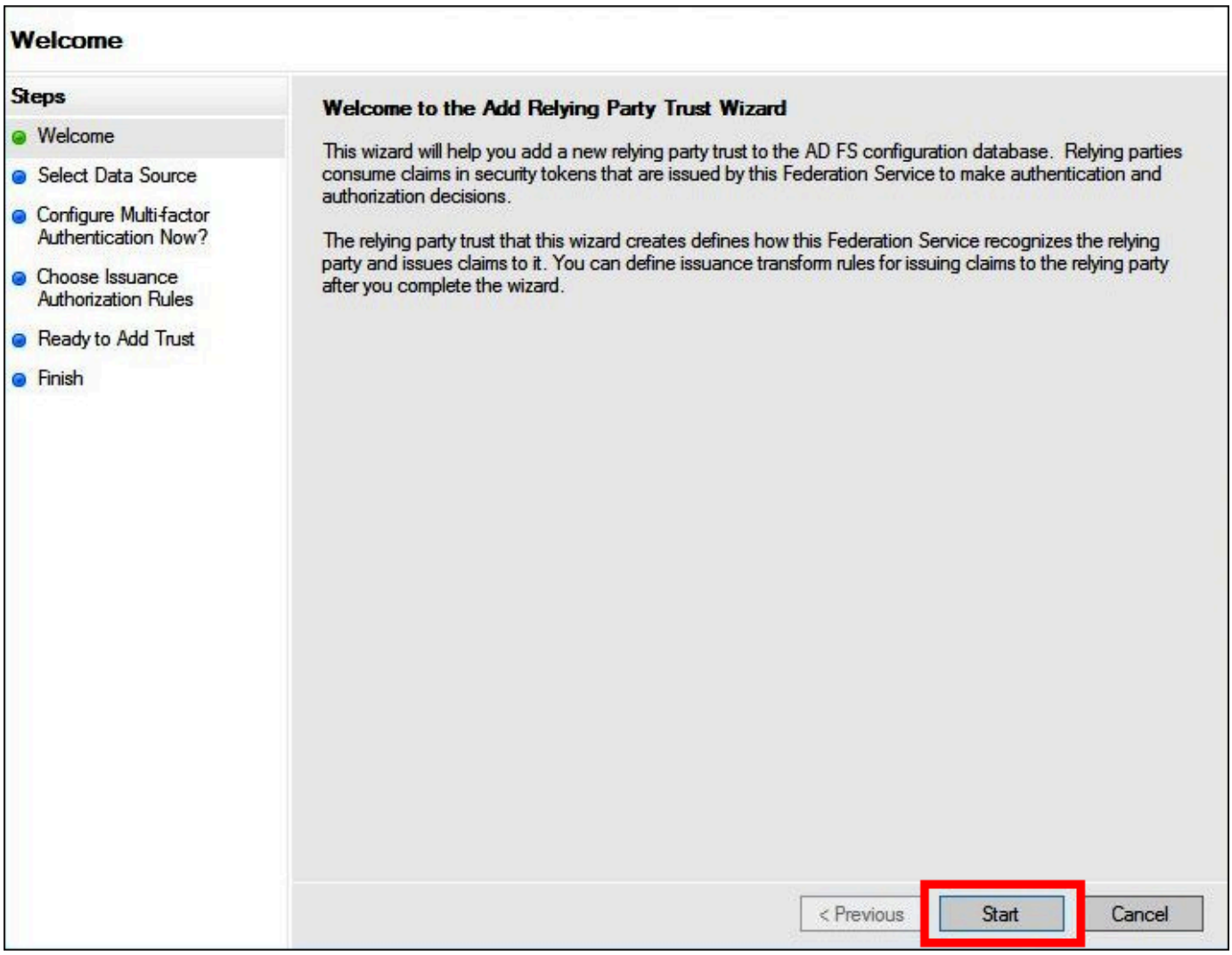
1. Masuk ke server AD FS.
2. PadaMulaimenu, bukaManajer Server.
3. PilihAlat, lalu pilihManajemen AD FS.



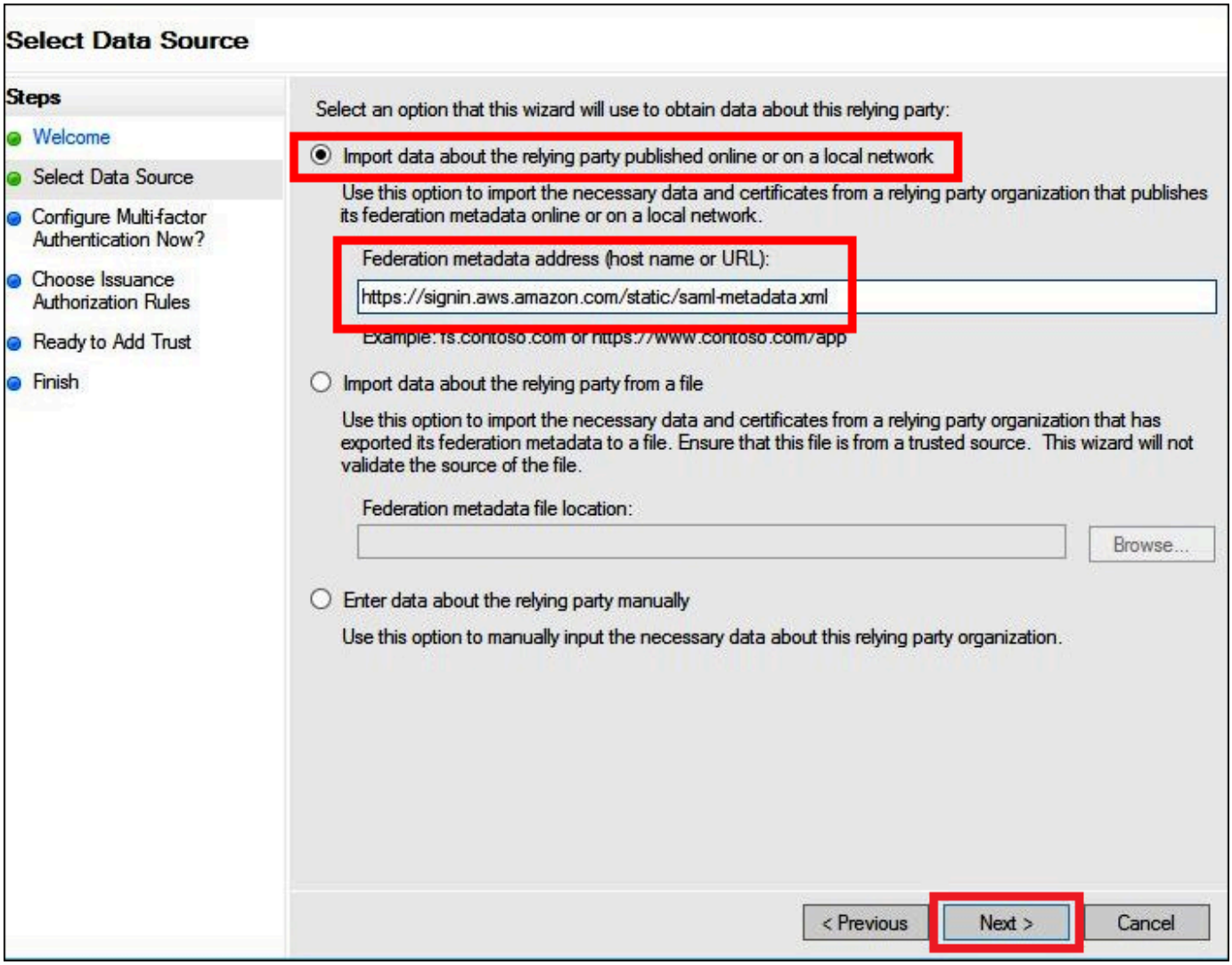
4. Di panel navigasi, di bawahHubungan Kepercayaan, pilihMengandalkan Perwalian Partai.
5. Di bawahTindakan, pilihTambahkan Mengandalkan Kepercayaan Partai.



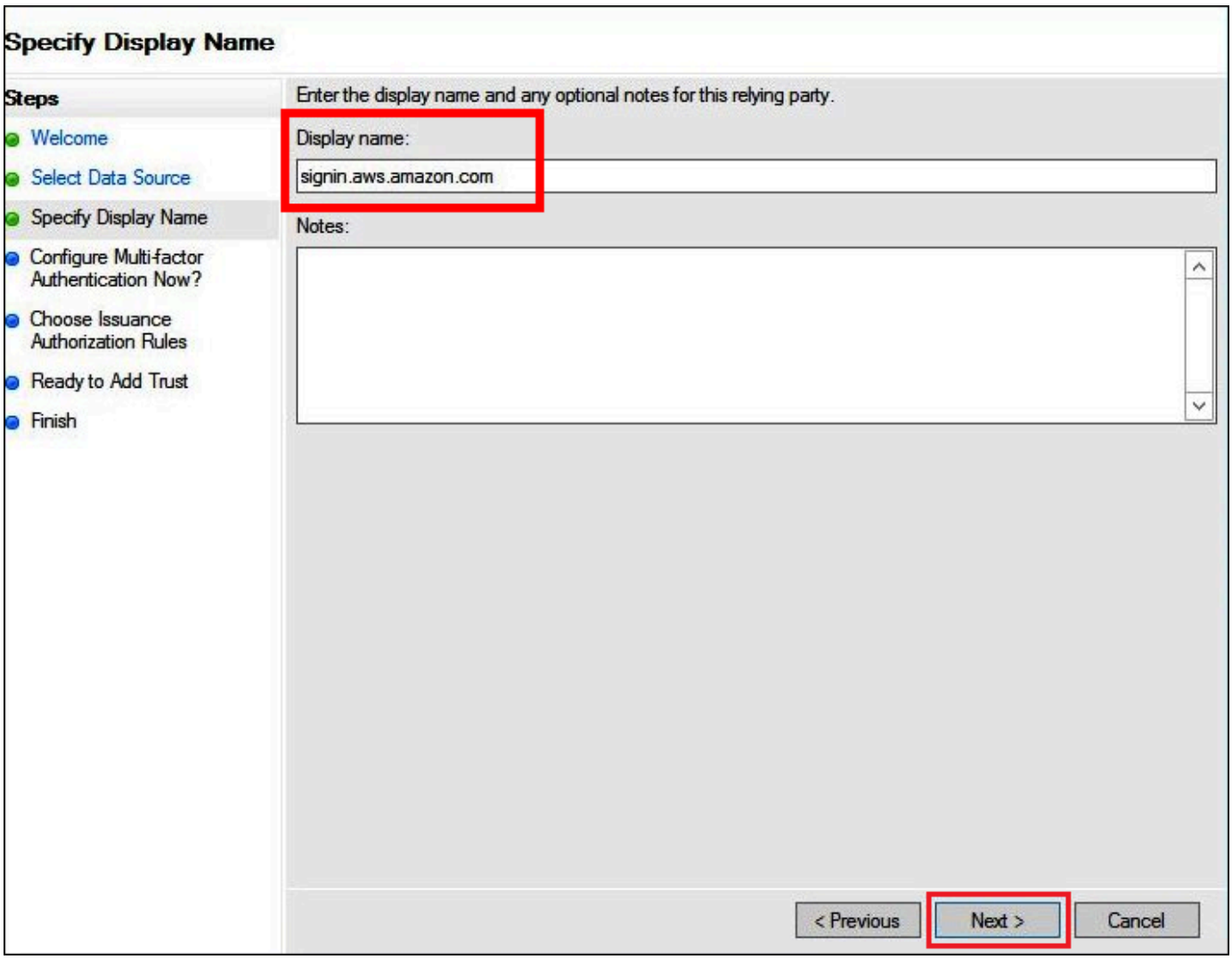
6. PadaTambahkan Wisaya Kepercayaan Partai yang Mengandalkanhalaman, pilihMulai.



7. PadaPilih Sumber Datalayar, pilih opsiMengimpor data tentang pihak yang mengandalkan yang dipublikasikan secara online atau di jaringan lokal.
8. UntukAlamat metadata Federasi (nama host atau URL), masukkan URL **`https://signin.aws.amazon.com/static/saml-metadata.xml`**
9. Pilih Selanjutnya.

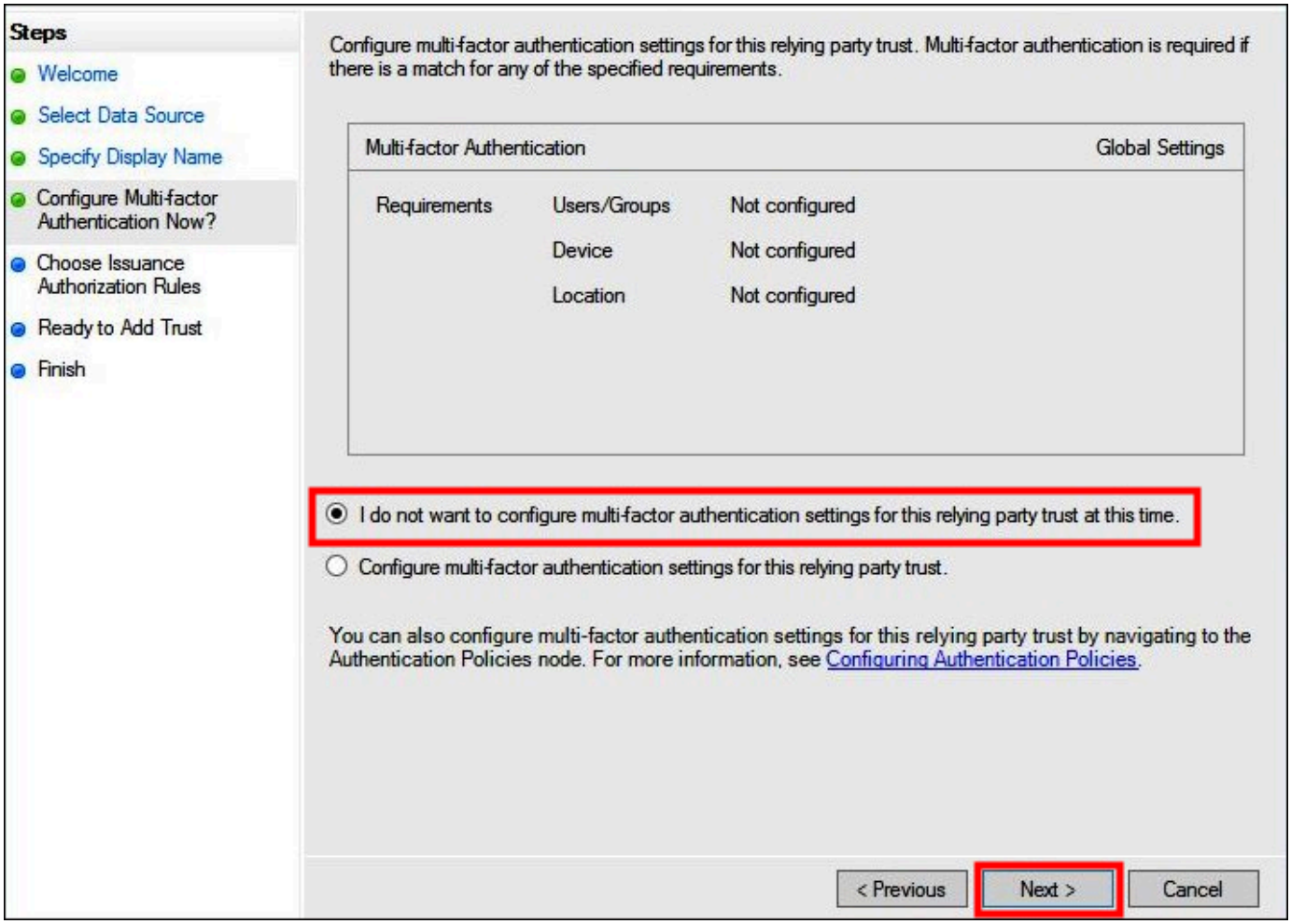


10. PadaTentukan Nama Tampilanhalaman, untukNama tampilan, masukkan nama tampilan untuk pihak yang Anda andalkan, lalu pilihBerikutnya.



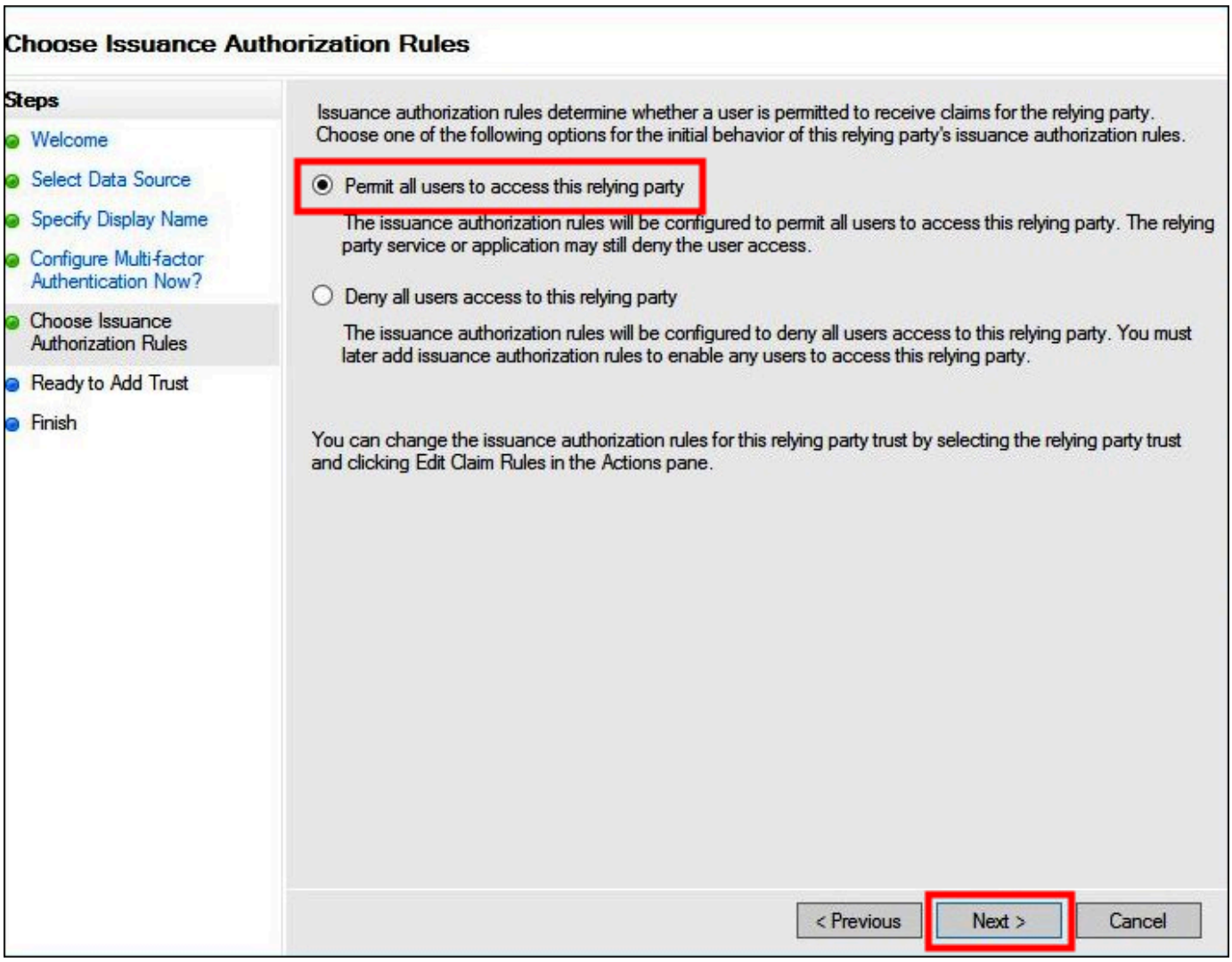
11. PadaKonfigurasikan Otentikasi Multi-faktor Sekaranghalaman, tutorial ini memilihSaya tidak ingin mengkonfigurasi otentikasi multi-faktor untuk kepercayaan pihak yang mengandalkan ini saat ini.

    Untuk meningkatkan keamanan, sebaiknya konfigurasikan autentikasi multi-faktor untuk membantu melindungiAWSsumber daya. Karena menggunakan kumpulan data sampel, tutorial ini tidak mengaktifkan otentikasi multi-faktor.

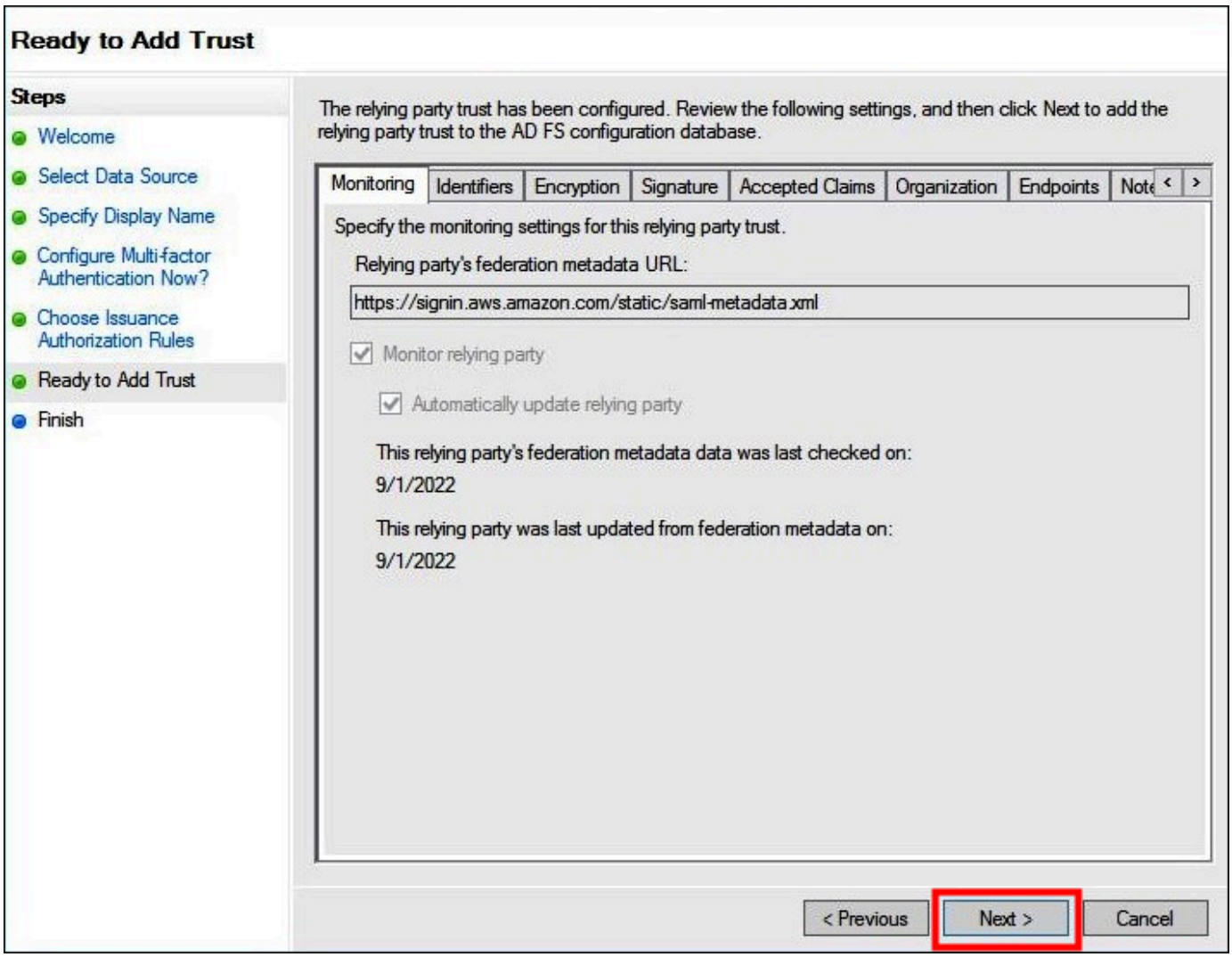


12. Pilih Selanjutnya.
13. PadaPilih Aturan Otorisasi Penerbitanhalaman, pilihIzinkan semua pengguna untuk mengakses pihak yang mengandalkan ini.

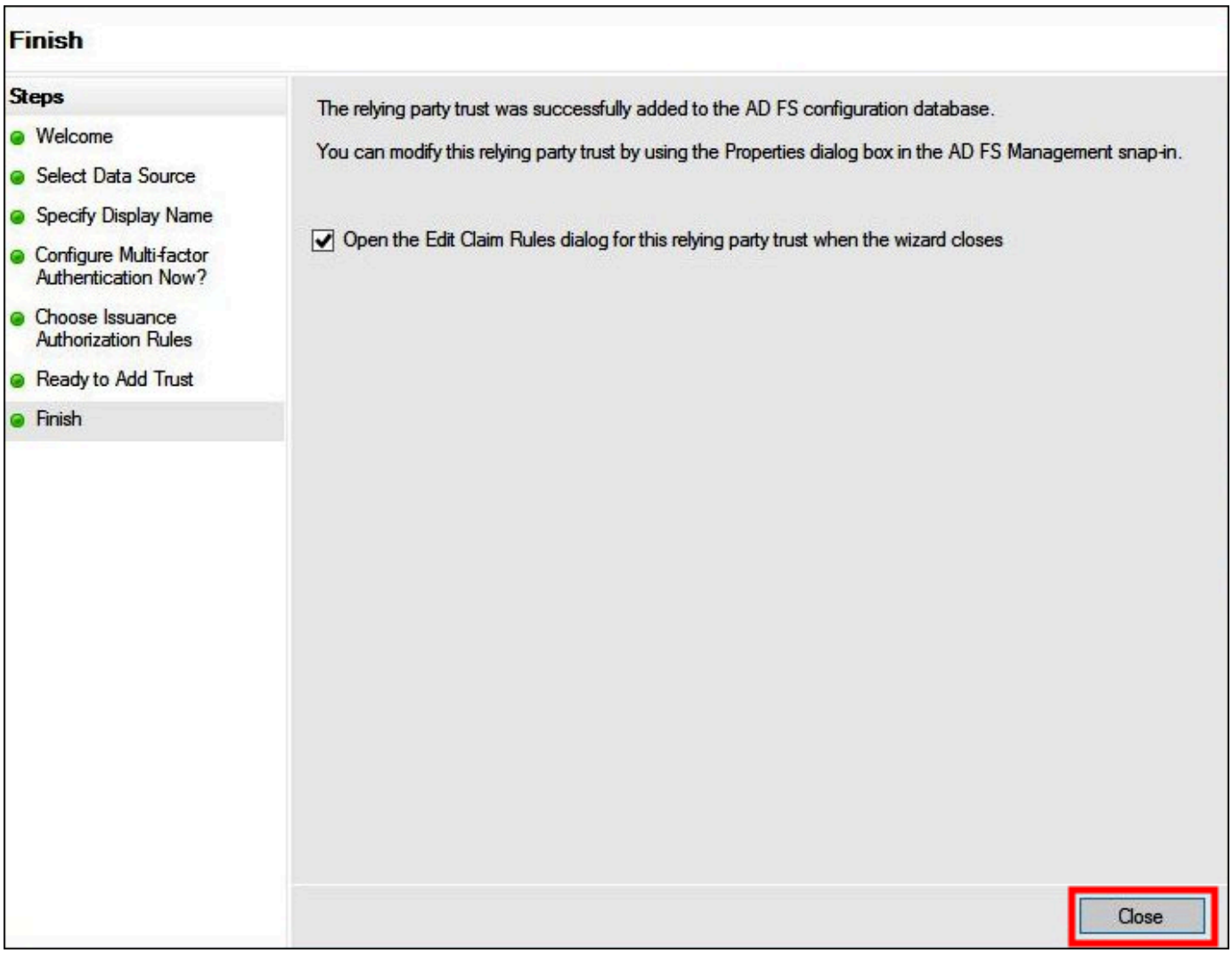
    Opsi ini memungkinkan semua pengguna di Active Directory untuk menggunakan AD FS denganAWSsebagai pihak yang mengandalkan. Anda harus mempertimbangkan persyaratan keamanan Anda dan menyesuaikan konfigurasi ini sesuai.



14. Pilih Selanjutnya.
15. PadaSiap Tambah Kepercayaanhalaman, pilihBerikutnyauntuk menambahkan kepercayaan pihak yang mengandalkan ke database konfigurasi AD FS.



16. PadaSelesaihalaman, pilihTutup.



Mengonfigurasi aturan klaim SALL untuk pihak yang mengandalkan

Dalam tugas ini, Anda membuat dua set aturan klaim.

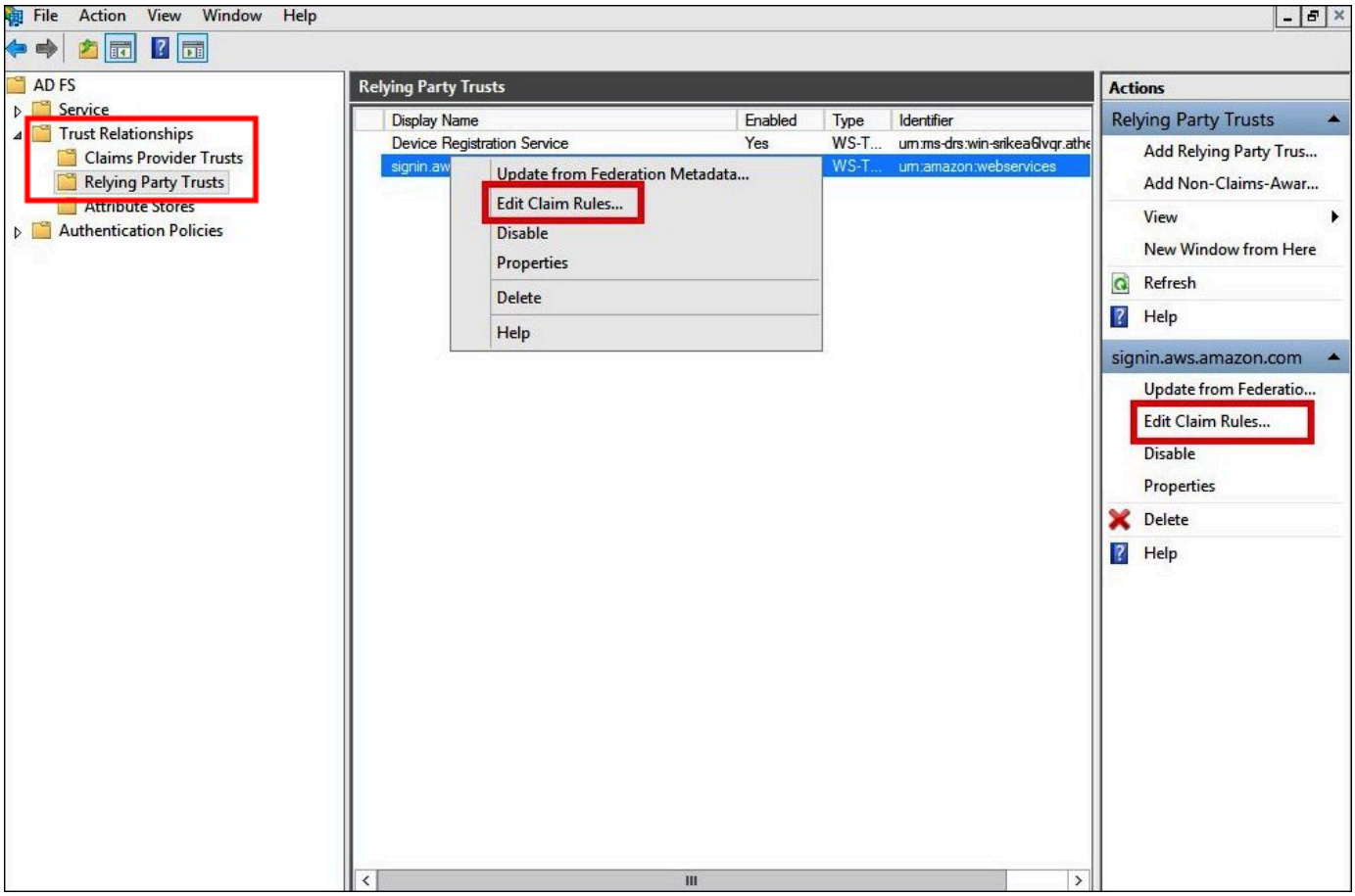
Set pertama, aturan 1—4, berisi aturan klaim AD FS yang diperlukan untuk mengambil peran IAM berdasarkan keanggotaan grup AD. Ini adalah aturan yang sama yang Anda buat jika Anda ingin membuat akses federasi keAWS Management Console.

Set kedua, aturan 5—6, adalah aturan klaim yang diperlukan untuk kontrol akses Athena.

Untuk membuat aturan klaim AD FS

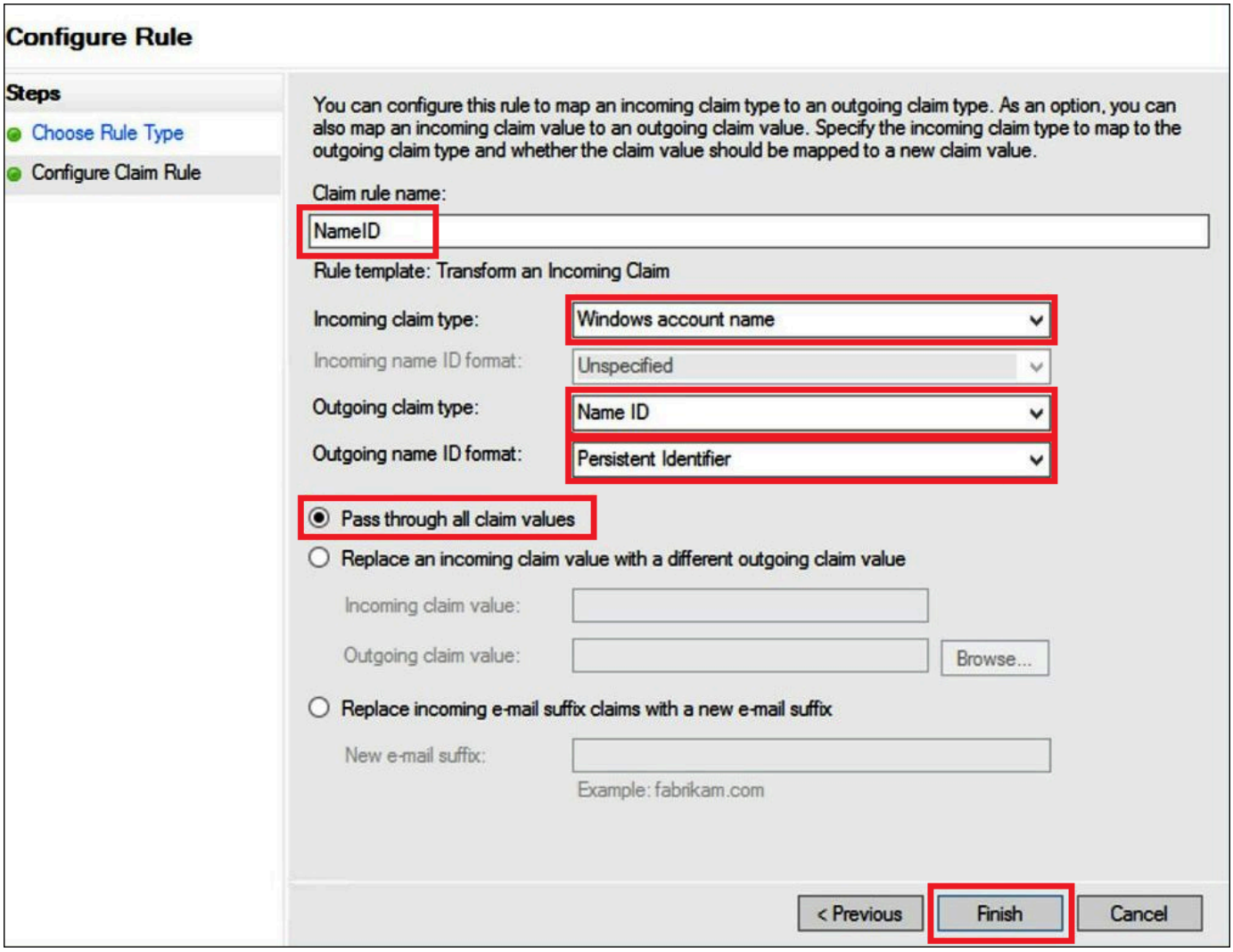
1. Di panel navigasi konsol AD FS Management, pilihHubungan Kepercayaan,Mengandalkan Perwalian Partai.

2. Temukan pihak yang mengandalkan yang Anda buat di bagian sebelumnya.
3. Klik kanan pihak yang mengandalkan dan pilihEdit Aturan Klaim, atau pilihEdit Aturan KlaimdariTindakanmenu.



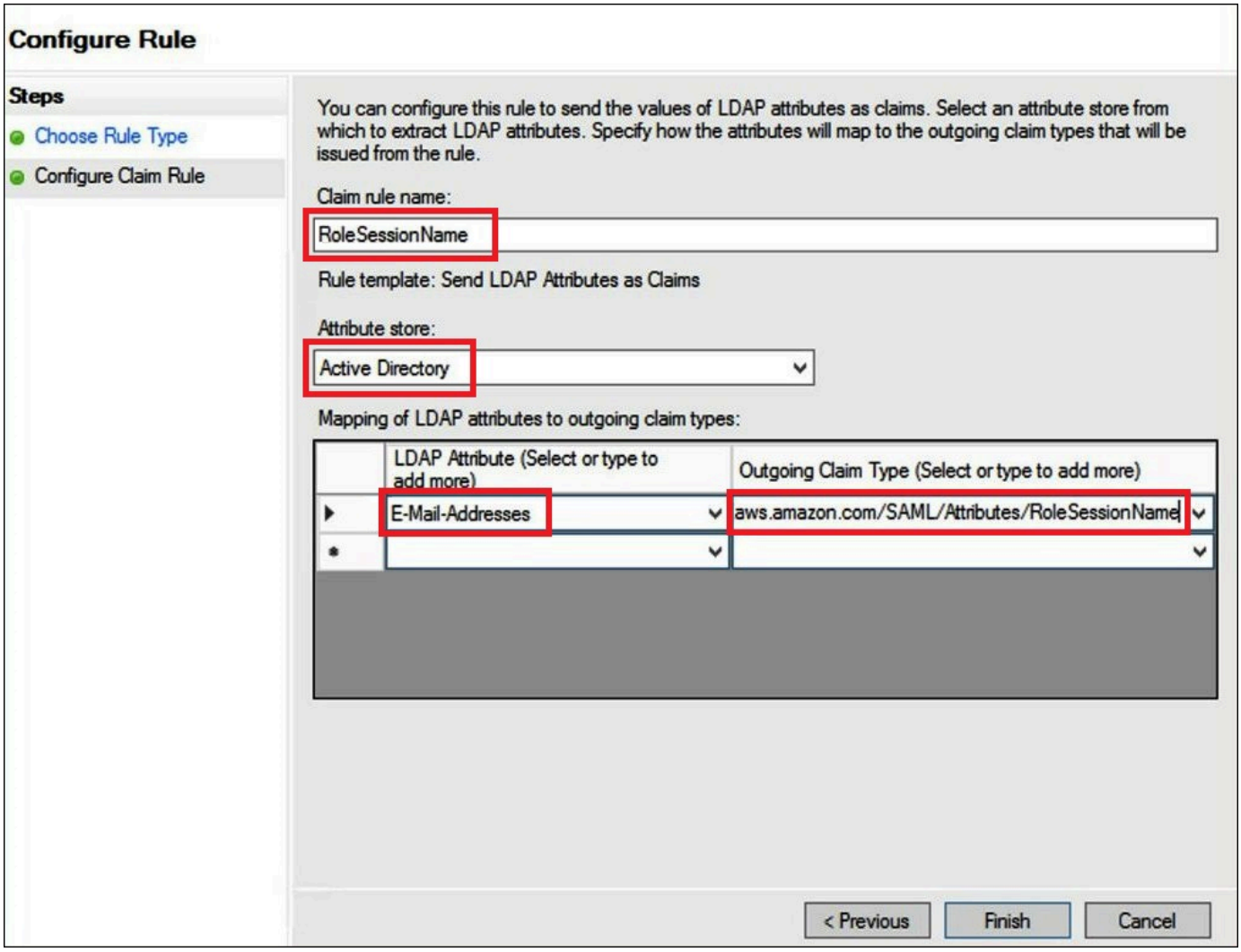
4. Pilih Tambahkan aturan.
5. PadaKonfigurasi Aturanhalaman Wizard Add Transform Claim Rule, masukkan informasi berikut untuk membuat aturan klaim 1, lalu pilihSelesai.

   - UntukNama Aturan Klaim, masuklah**`NameID`**.
   - UntukTemplat aturan, gunakanMengubah Klaim Masuk.
   - UntukJenis klaim masuk, pilihNama akun Windows.
   - UntukJenis klaim keluar, pilihID Nama.
   - UntukFormat ID nama keluar, pilihPengenal Persisten.
   - PilihMelewati semua nilai klaim.



6. PilihTambahkan Aturan, lalu masukkan informasi berikut untuk membuat aturan klaim 2, lalu pilihSelesai.

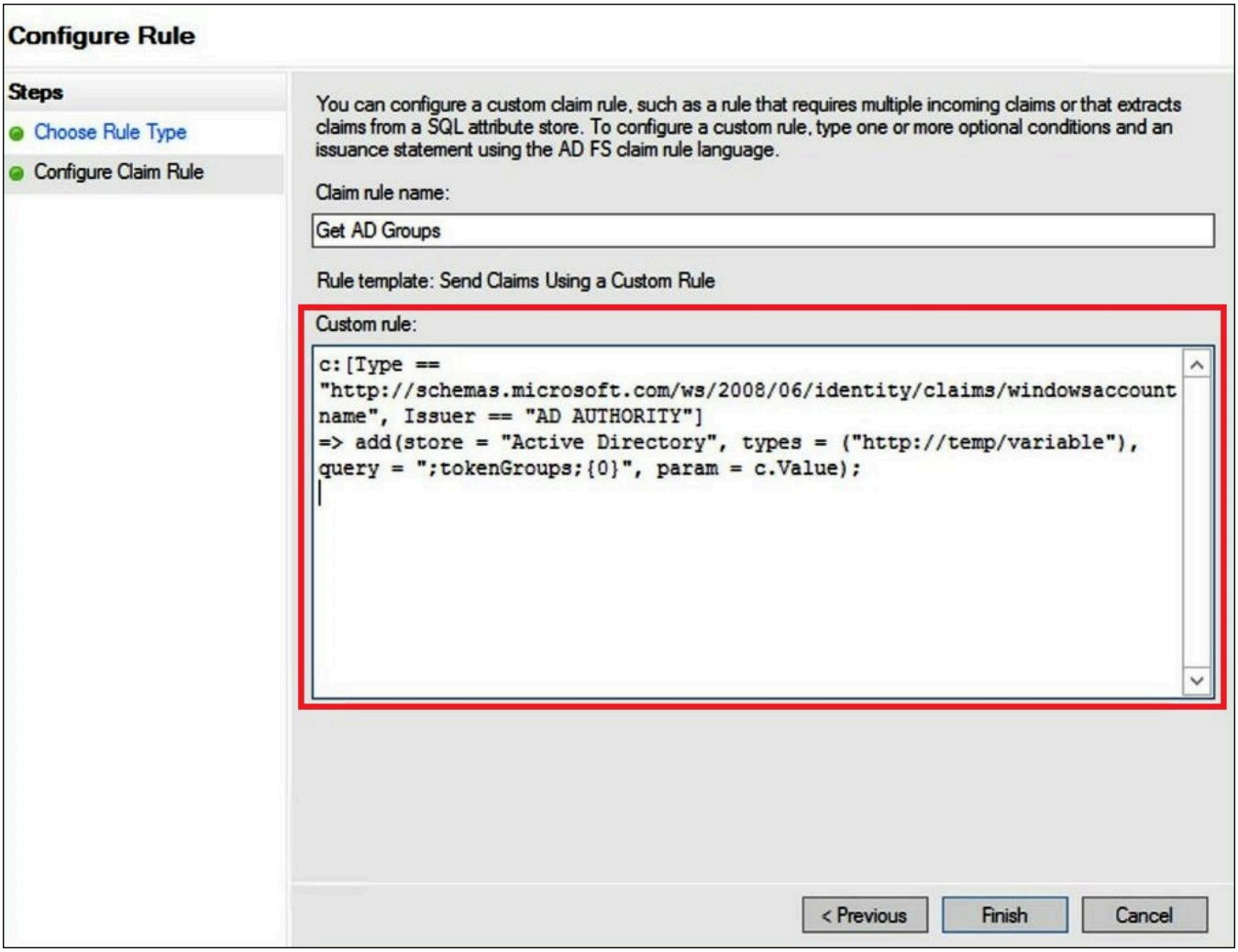
   - UntukNama aturan klaim, masuklah**`RoleSessionName`**.
   - UntukTemplat aturan, gunakanKirim Atribut LDAP sebagai Klaim.
   - UntukToko atribut, pilihDirektori Aktif.
   - UntukPemetaan atribut LDAP ke jenis klaim keluar, tambahkan atribut**`E-Mail-Addresses`**. UntukJenis Klaim Keluar, masuklah **`https://aws.amazon.com/SAML/Attributes/RoleSessionName`**.



7. PilihTambahkan Aturan, lalu masukkan informasi berikut untuk membuat aturan klaim 3, lalu pilihSelesai.

   - UntukNama aturan klaim, masuklah**`Get AD Groups`**.
   - UntukTemplat aturan, gunakanMengirim Klaim Menggunakan Aturan Khusus.
   - UntukAturan khusus, masukkan kode berikut:

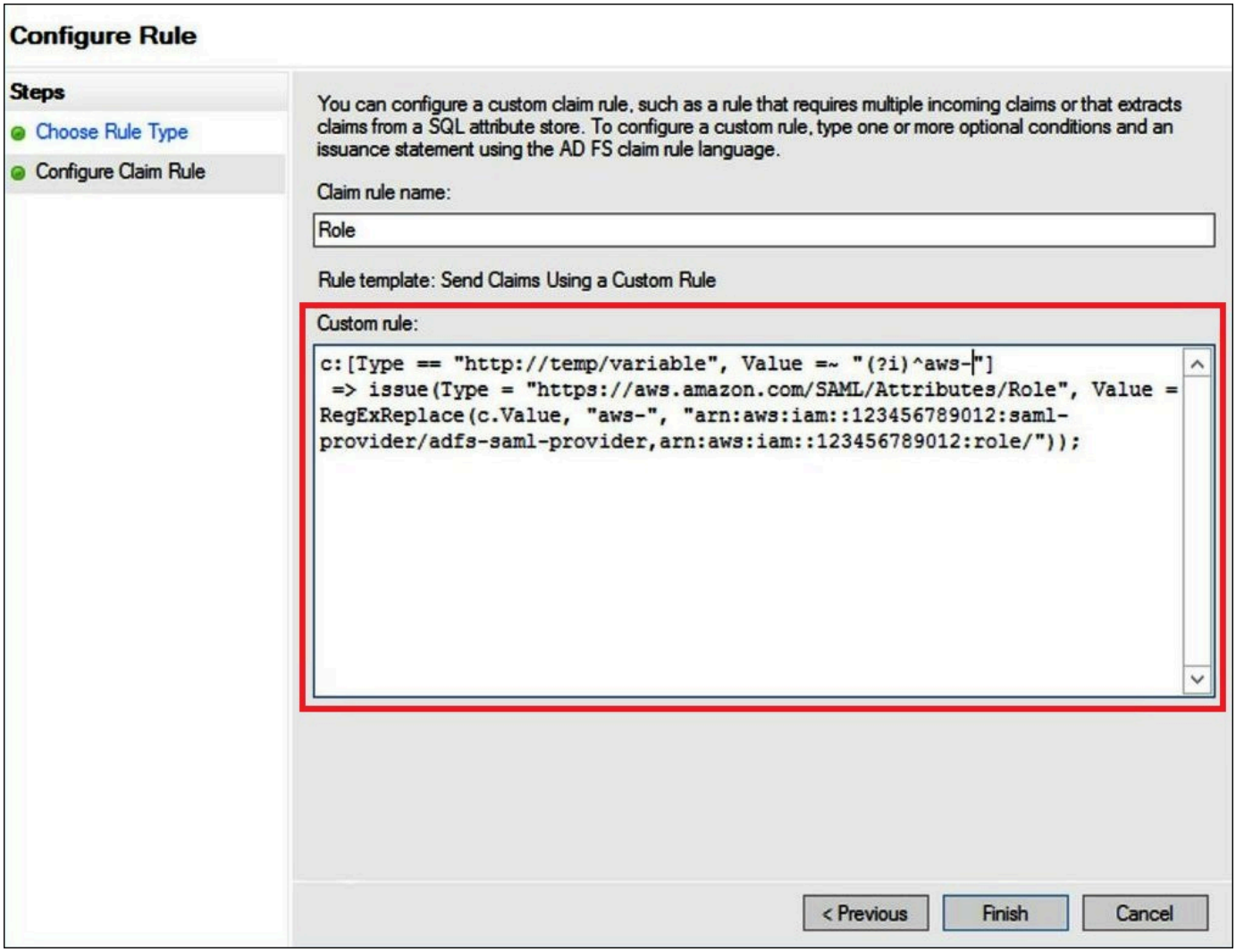
```
c:[Type == "http://schemas.microsoft.com/ws/2008/06/identity/claims/
windowsaccountname",
 Issuer == "AD AUTHORITY"]=> add(store = "Active Directory", types = ("http://
temp/variable"),
 query = ";tokenGroups;{0}", param = c.Value);
```



8. Pilih Tambahkan aturan. Masukkan informasi berikut untuk membuat aturan klaim 4, lalu pilihSelesai.

   - UntukNama aturan klaim, masuklah**`Role`**.
   - UntukTemplat aturan, gunakanMengirim Klaim Menggunakan Aturan Khusus.
   - UntukAturan khusus, masukkan kode berikut dengan nomor akun Anda dan nama penyedia SAKL yang Anda buat sebelumnya:

```
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/"));
```

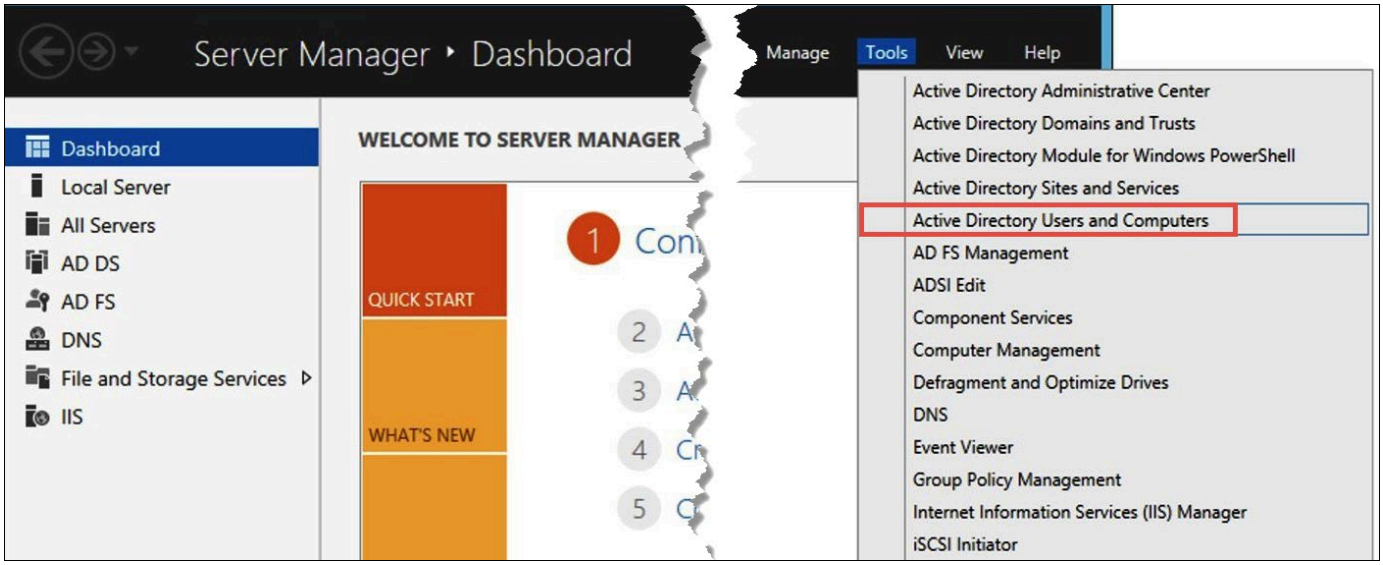


3. Membuat pengguna dan grup Active Directory

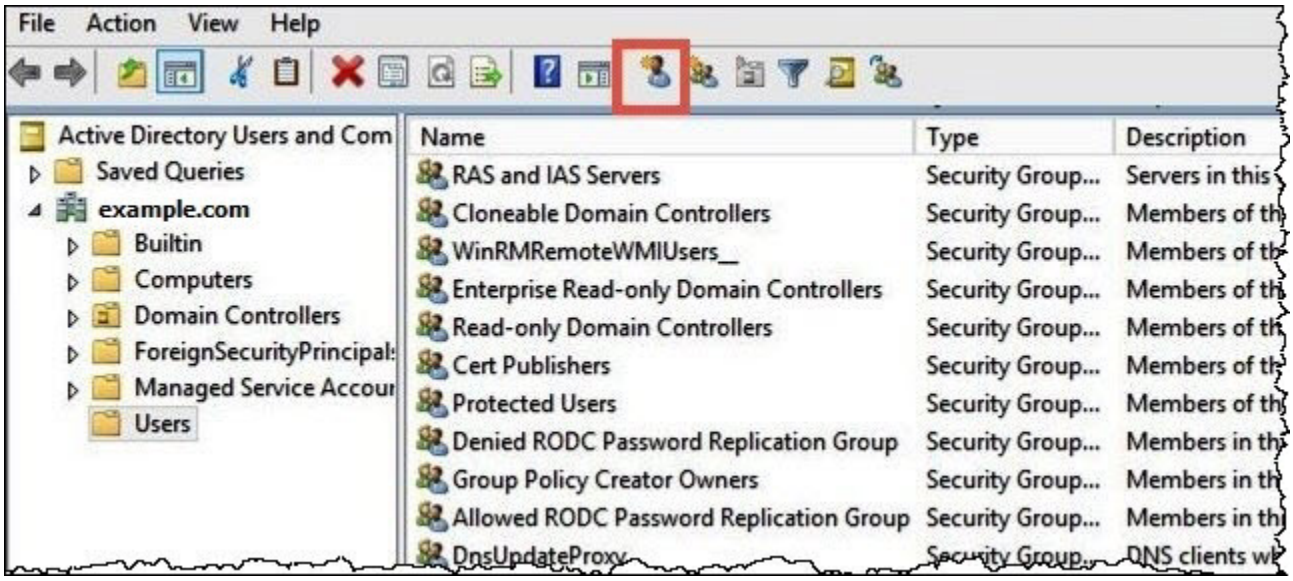
Sekarang Anda siap untuk membuat pengguna AD yang akan mengakses Athena, dan grup AD untuk menempatkan mereka di sehingga Anda dapat mengontrol tingkat akses oleh kelompok. Setelah membuat grup AD yang mengkategorikan pola akses data, Anda menambahkan pengguna ke grup tersebut.

Untuk membuat pengguna AD untuk akses ke Athena

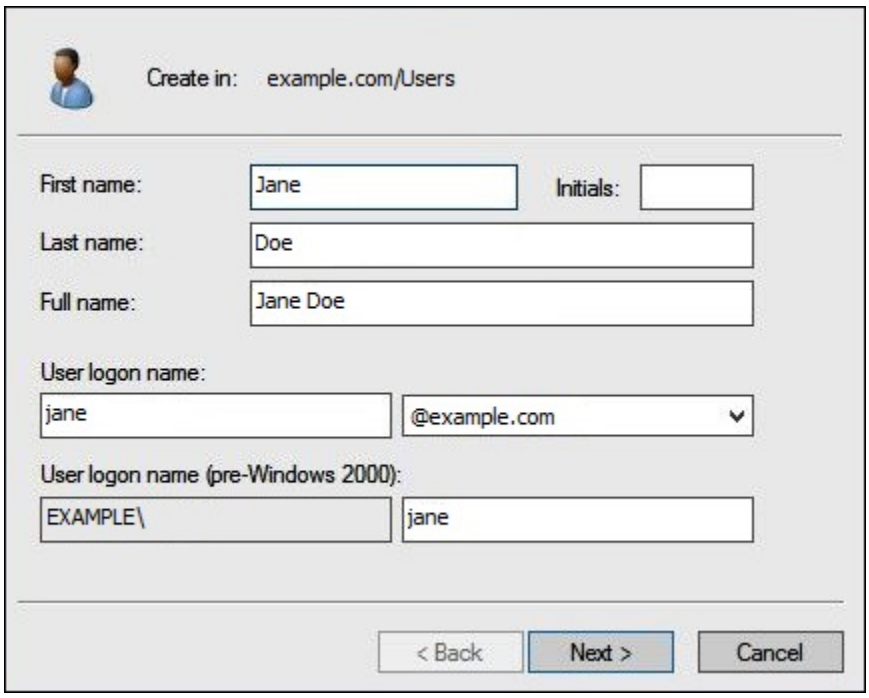
1. Di dasbor Server Manager, pilihAlat, lalu pilihPengguna Active Directory dan Komputer.



2. Di panel navigasi, pilih Users (Pengguna).
3. PadaPengguna Active Directory dan Komputertool bar, pilihBuat penggunapilihan.

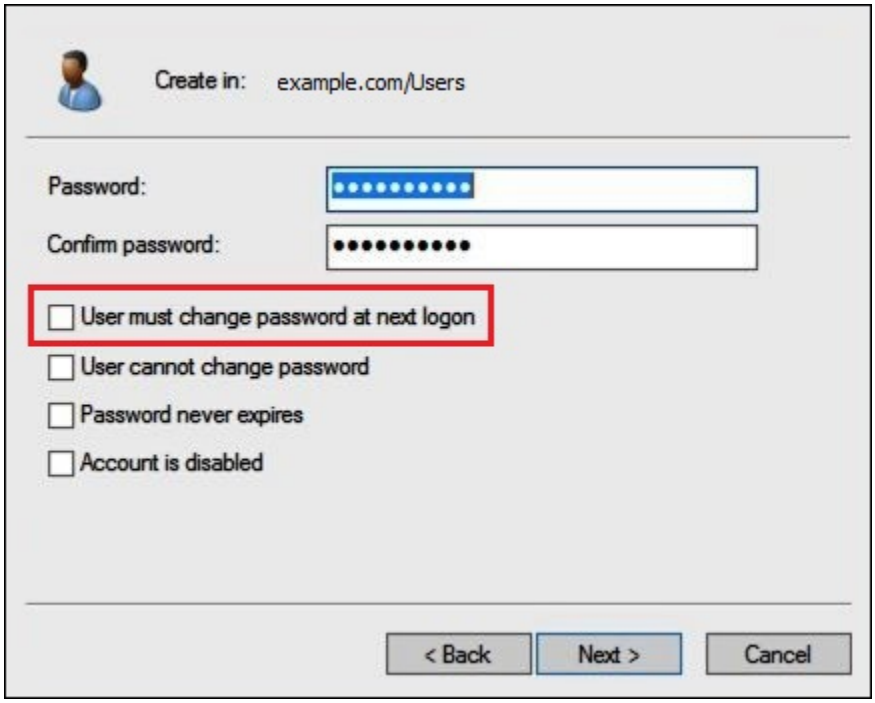


4. Di dalamNew Object - Penggunakotak dialog, untukNama pertama,Nama belakang, danNama lengkap, masukkan nama. Tutorial ini menggunakan**`Jane Doe`**.

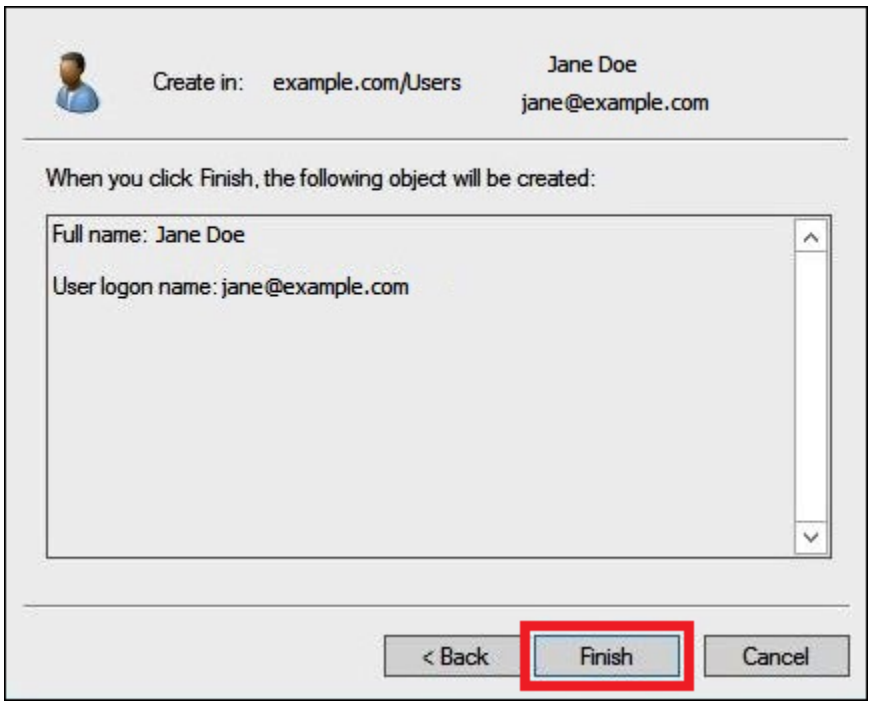


5. Pilih Selanjutnya.
6. UntukKata Sandi, masukkan kata sandi, lalu ketik ulang untuk mengonfirmasi.

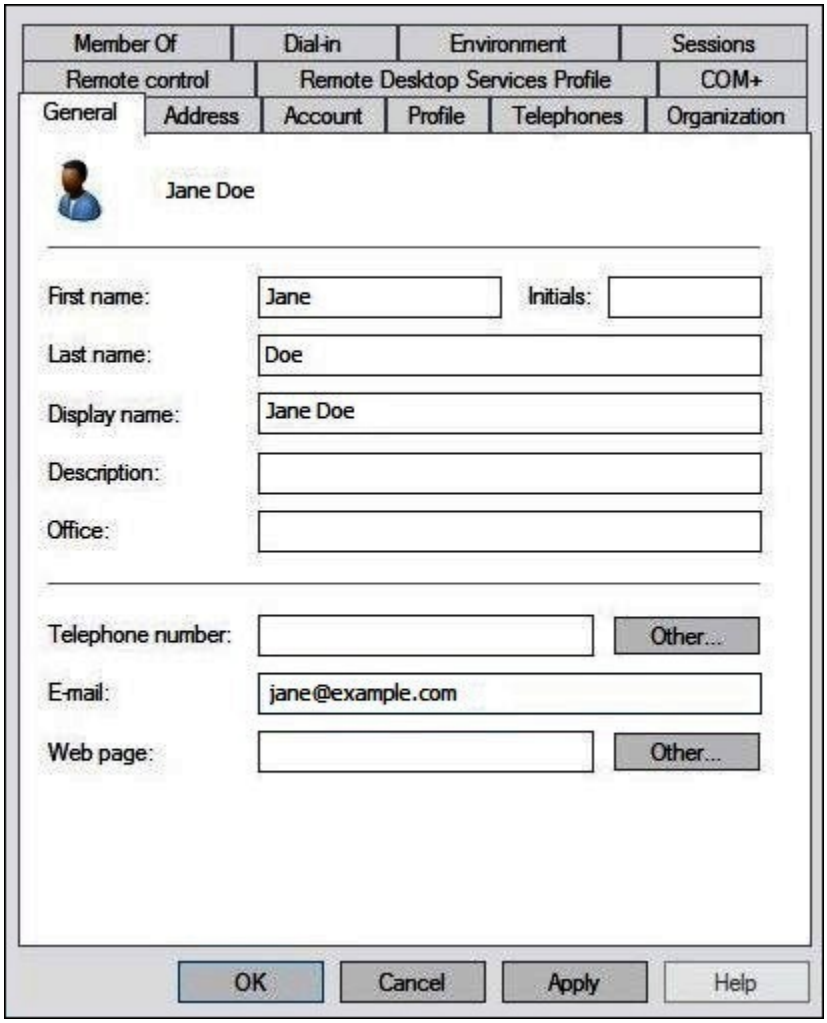
   Untuk mempermudah, tutorial ini membatalkan pilihanPengguna harus mengubah kata sandi pada saat sign on berikutnya. Dalam skenario dunia nyata, Anda harus meminta pengguna yang baru dibuat untuk mengubah kata sandi mereka.



7. Pilih Selanjutnya.
8. Pilih Selesai.



9. DalamPengguna Active Directory dan Komputer, pilih nama pengguna.
10. Di dalamPropertikotak dialog untuk pengguna, untukE-mail, masukkan alamat email. Tutorial ini menggunakan**`jane@example.com`**.



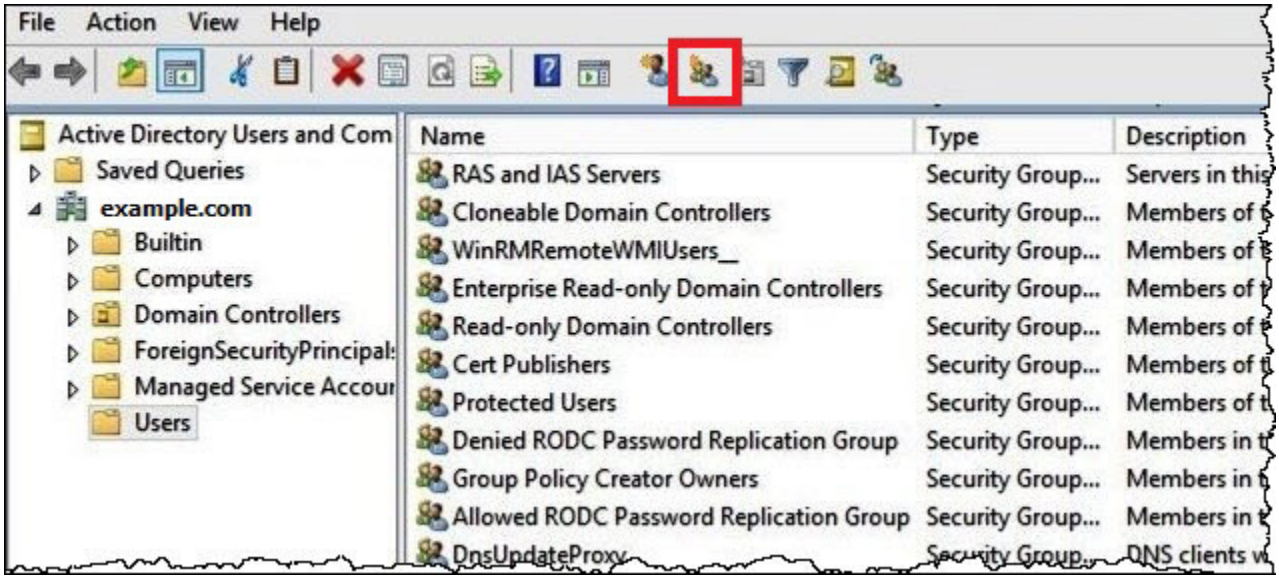
11. Pilih OKE.

Buat grup AD untuk mewakili pola akses data

Anda dapat membuat grup AD yang anggotanya menganggap`adfs-data-access`Peran IAM saat mereka masukAWS. Contoh berikut membuat grup AD yang disebutaws-adfs-data-access.

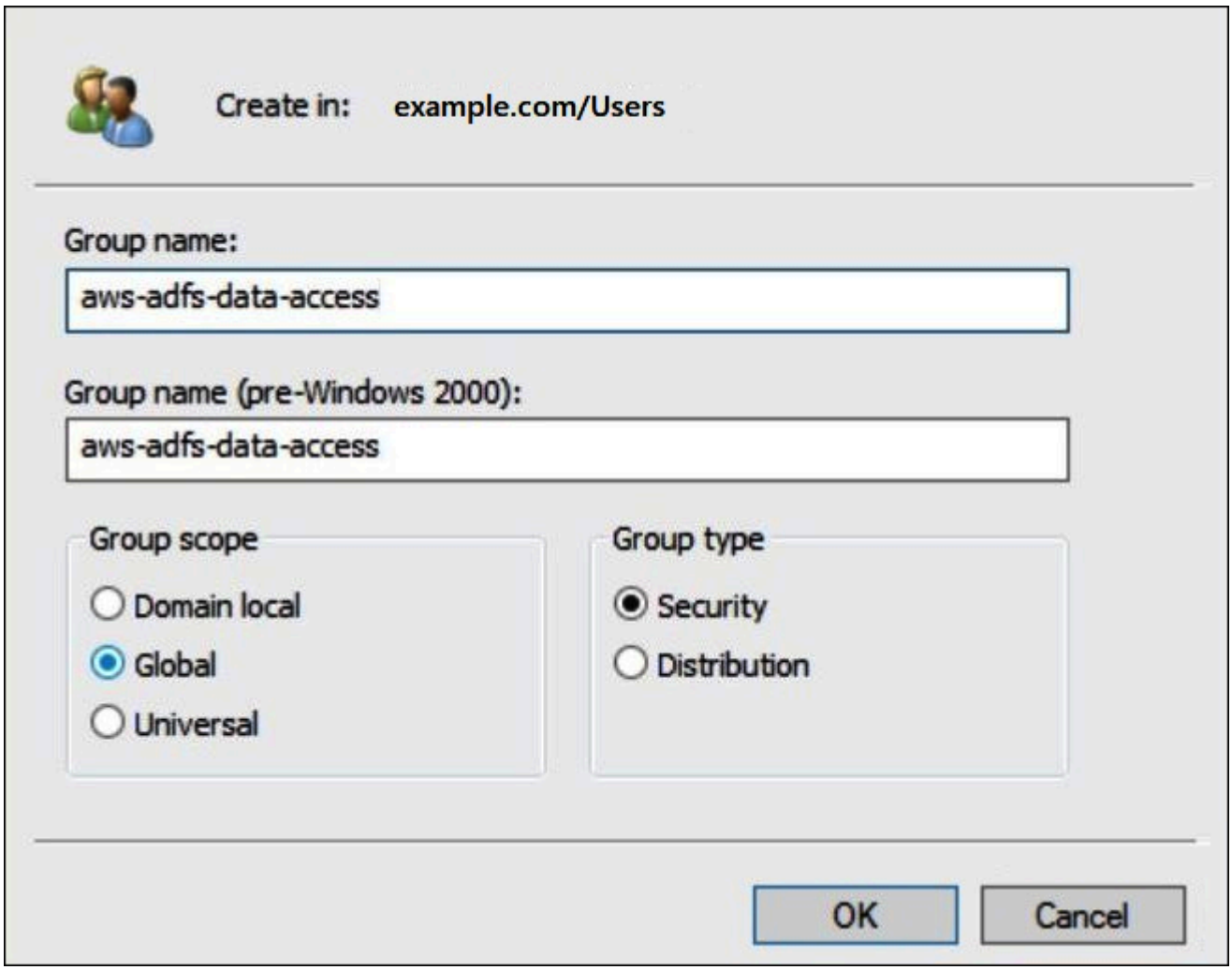
Membuat grup AD

1. Di Dasbor Server Manager, dariAlatmenu, pilihPengguna Active Directory dan Komputer.
2. Pada bilah alat, pilihBuat grup barupilihan.



3. DalamObyek Baru - Grupkotak dialog, masukkan informasi berikut:

   - UntukNama grup, masuklah**`aws-adfs-data-access`**.
   - UntukLingkup kelompok, pilihGlobal.
   - UntukTipe grup, pilihKeamanan.



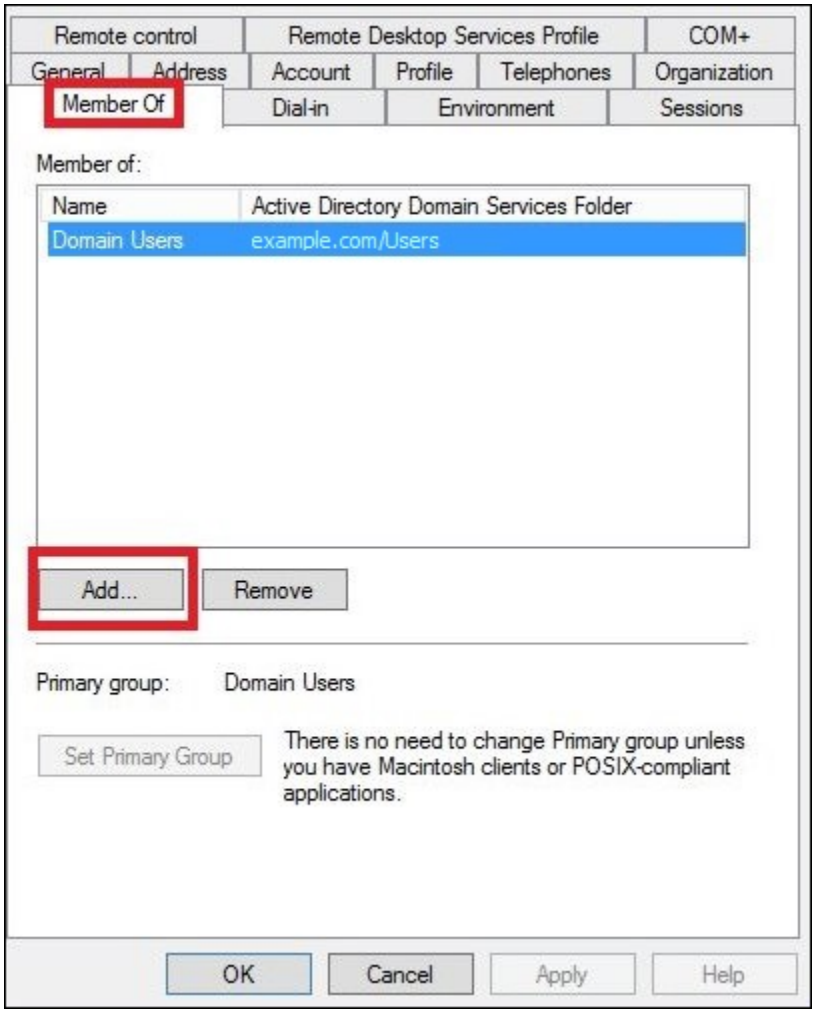
4. Pilih OKE.

Menambahkan pengguna AD ke grup yang sesuai

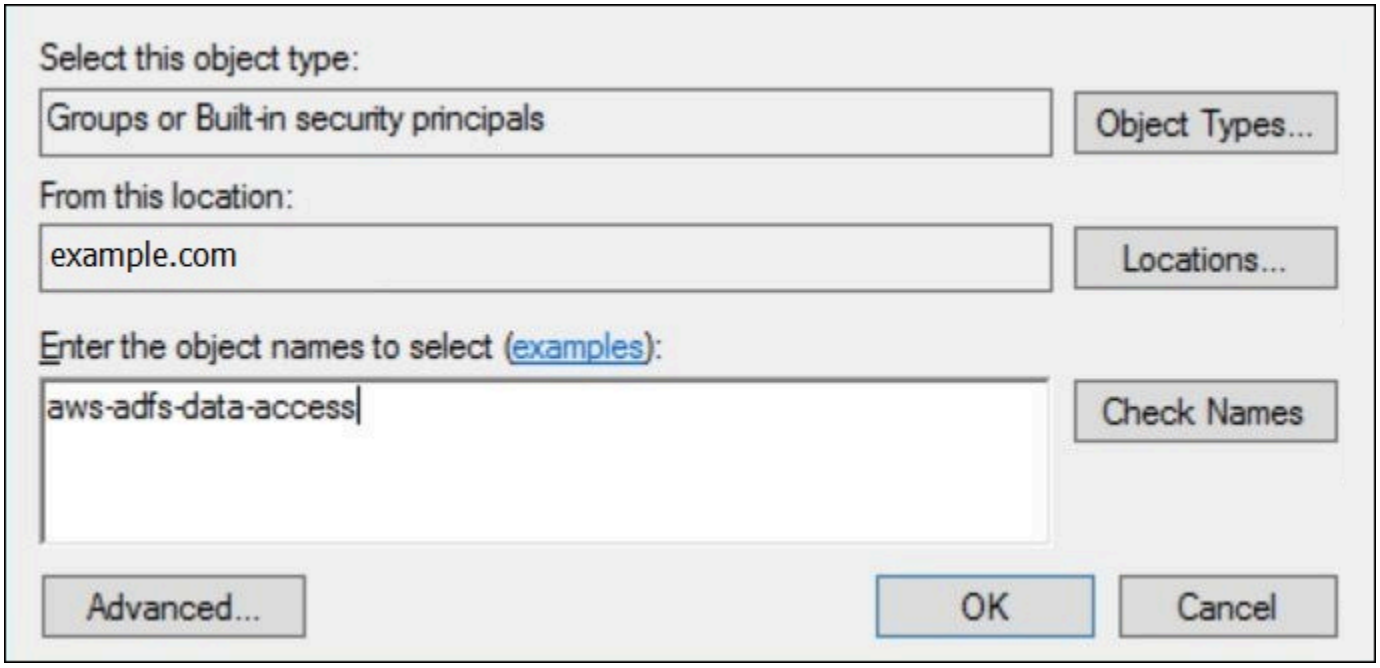
Sekarang Anda telah membuat pengguna AD dan grup AD, Anda dapat menambahkan pengguna ke grup.

Menambahkan pengguna AD ke grup AD

1. Di Dasbor Server Manager, diAlatmenu, pilihPengguna Active Directory dan Komputer.
2. UntukNama pertamadanNama belakang, pilih pengguna (misalnya,Jane Doe).
3. DalamPropertikotak dialog untuk pengguna, padaAnggota Daritab, pilihMenambahkan.

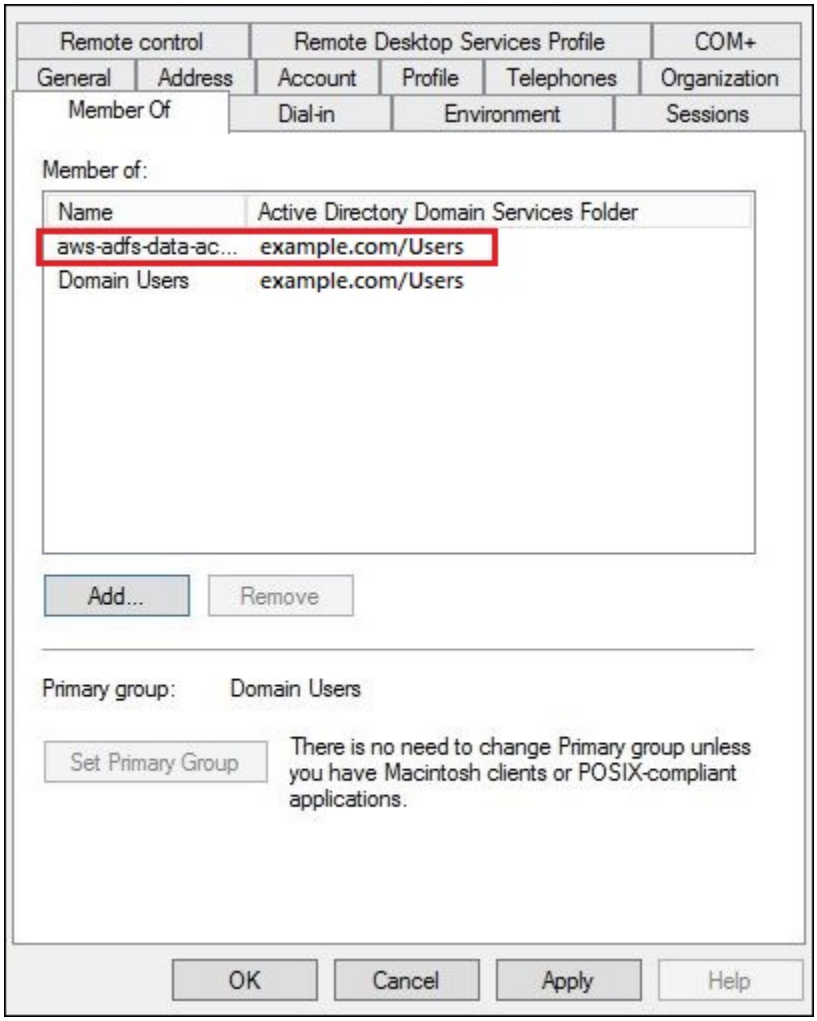


4. Tambahkan satu atau beberapa grup AD FS sesuai dengan kebutuhan Anda. Tutorial ini menambahkanaws-adfs-data-accesskelompok.
5. DalamPilih Grupkotak dialog, untukMasukkan nama objek yang akan dipilih, masukkan nama grup AD FS yang Anda buat (misalnya,**`aws-adfs-data-access`**), dan kemudian pilihPeriksa Nama.



6. Pilih OKE.

   DalamPropertikotak dialog untuk pengguna, nama grup AD muncul diAnggota daridaftar.



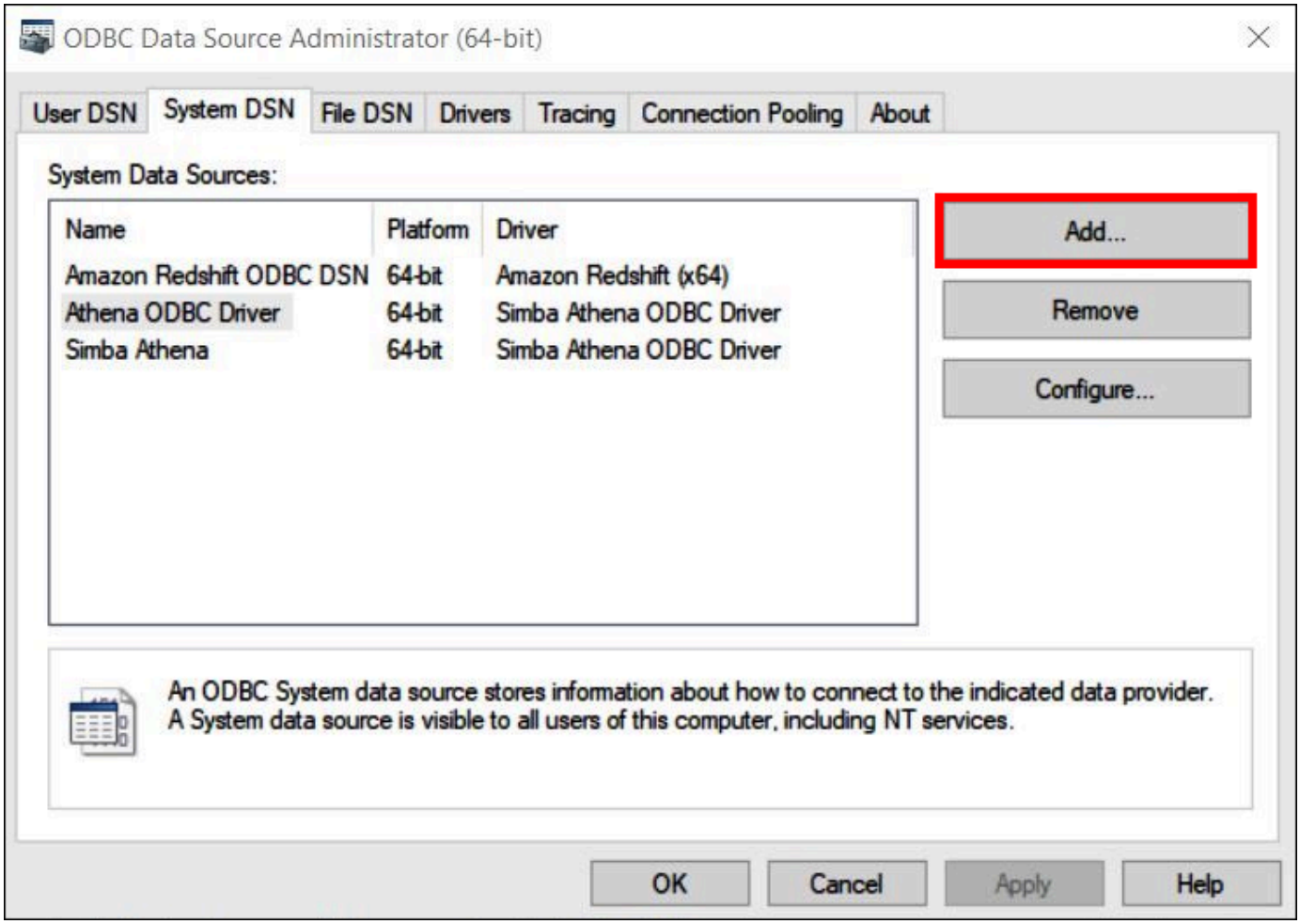
7. PilihTerapkan, lalu pilihOKE.

4. Mengkonfigurasi koneksi AD FS ODBC ke Athena

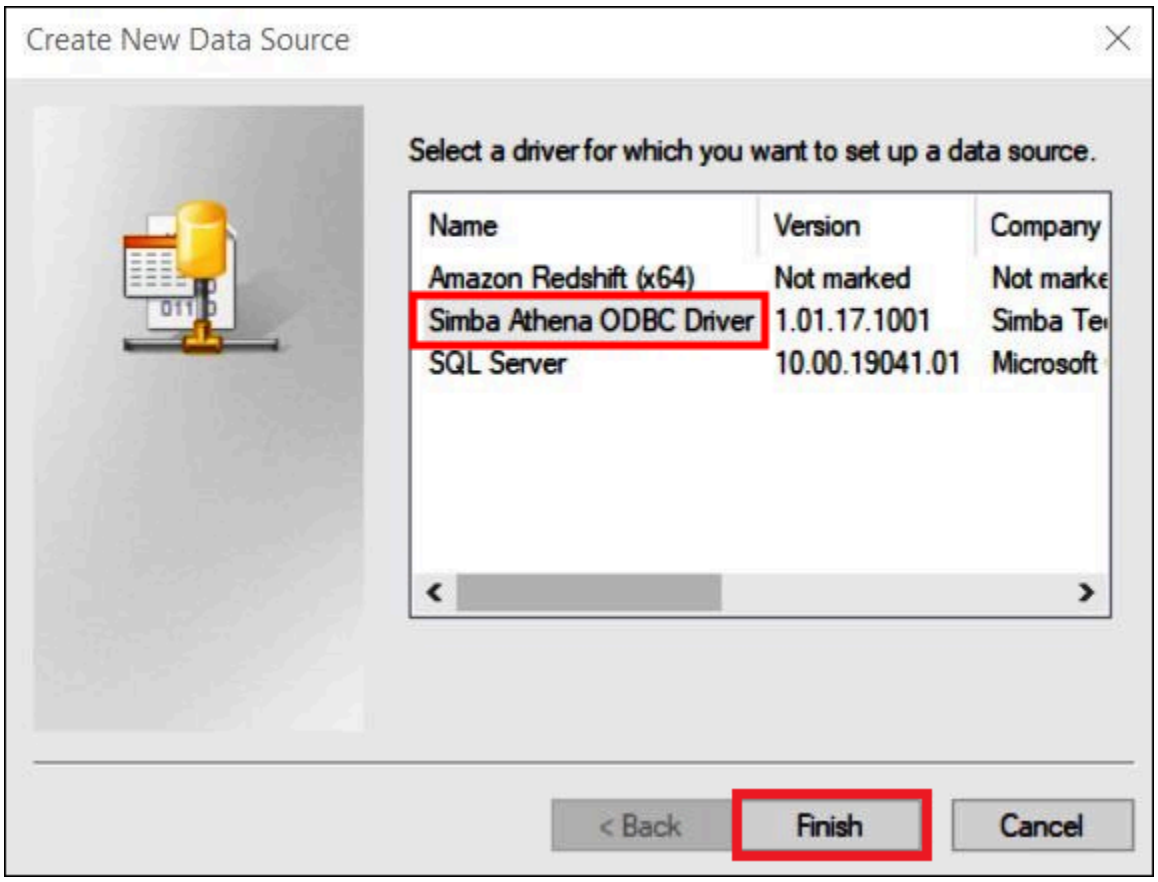
Setelah Anda membuat pengguna dan grup AD Anda, Anda siap untuk menggunakan program Sumber Data ODBC di Windows untuk mengkonfigurasi koneksi Athena ODBC Anda untuk AD FS.

Untuk mengkonfigurasi koneksi AD FS ODBC ke Athena

1. Instal driver ODBC untuk Athena. Untuk tautan unduhan, lihatMenghubungkan ke Amazon Athena dengan ODBC.
2. Di Windows, pilihMulai,Sumber Data ODBC.
3. DalamAdministrator Sumber Data ODBCprogram, pilihMenambahkan.

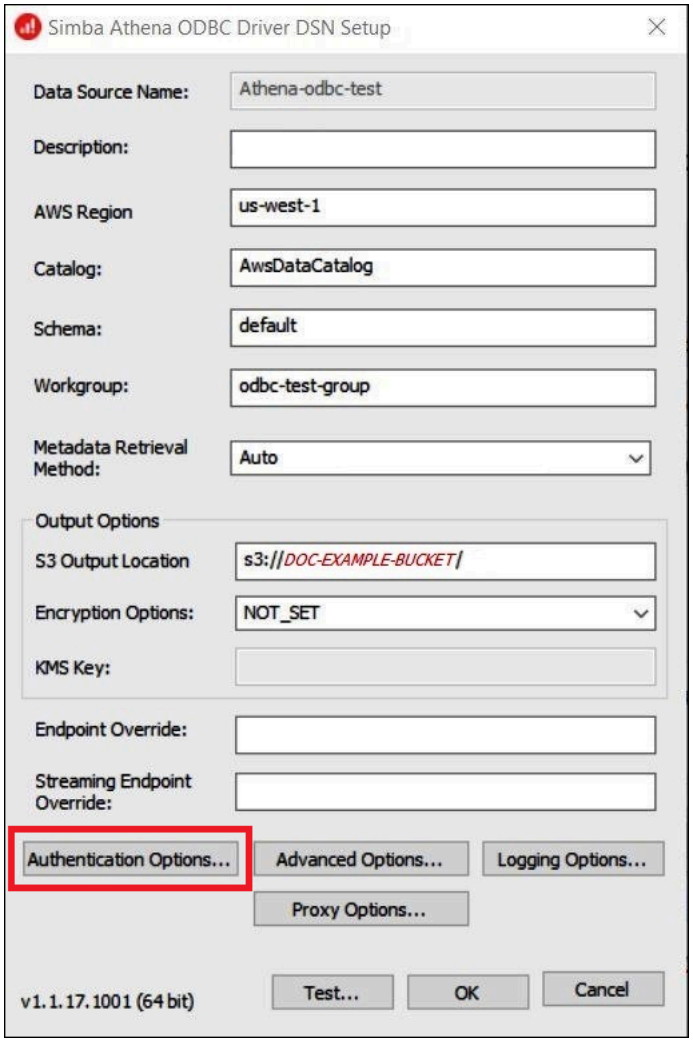


4. DalamBuat Sumber Data Barukotak dialog, pilihSimba Athena ODBC Driver, lalu pilihSelesai.



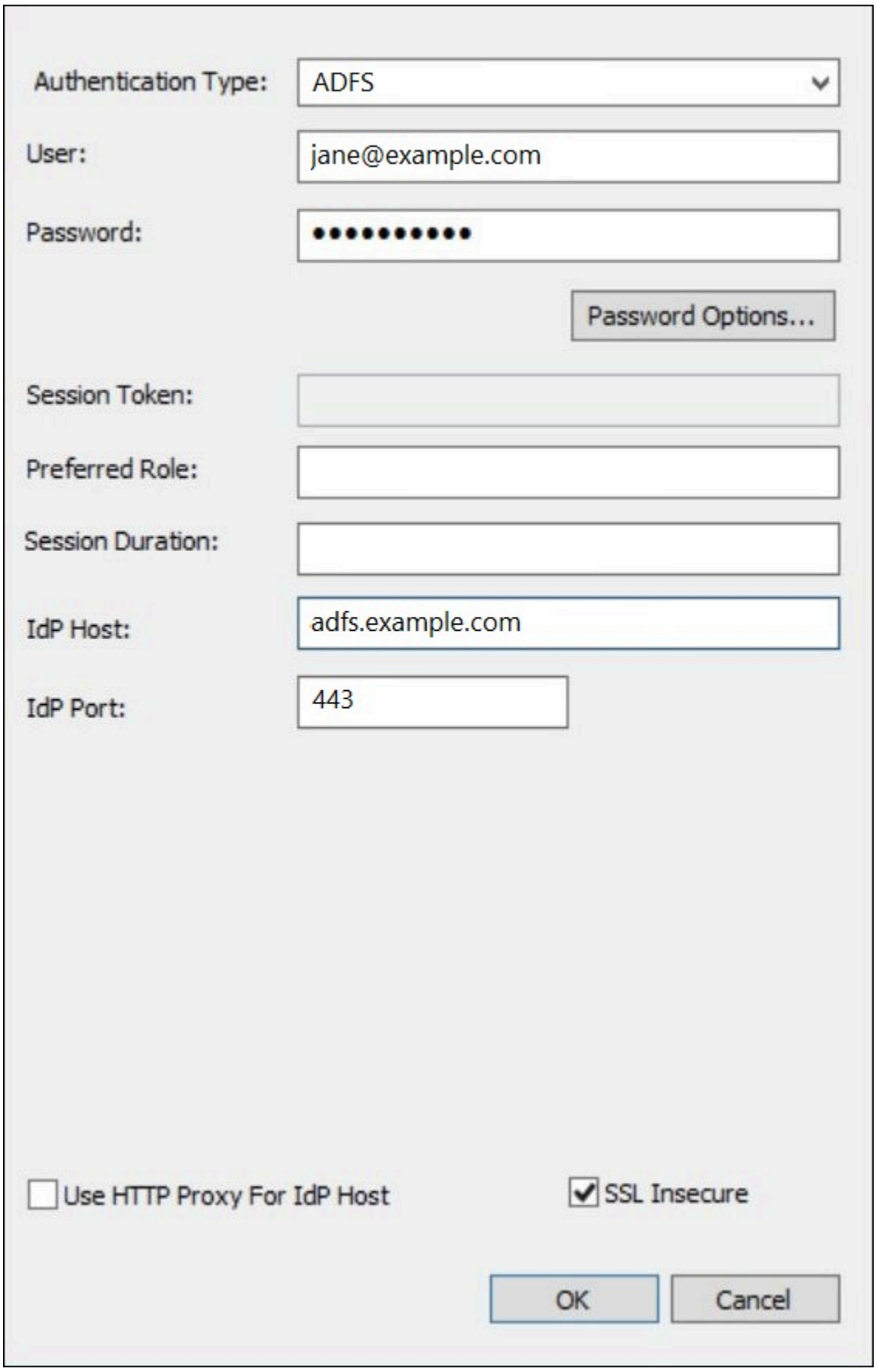
5. DalamSimba Athena ODBC Driver DSN Pengaturankotak dialog, masukkan nilai-nilai berikut:

   - UntukNama Sumber Data,masukkan nama untuk sumber data Anda (misalnya, **`Athena-odbc-test`**).
   - UntukDeskripsi, masukkan deskripsi untuk sumber data Anda.
   - UntukWilayah AWS, masukkanWilayah AWSyang Anda gunakan (misalnya, **`us-west-1`**).
   - UntukLokasi Output S3, masukkan jalur Amazon S3 tempat Anda ingin output Anda disimpan.



6. PilihPilihan Otentikasi.
7. DalamPilihan Otentikasikotak dialog, tentukan nilai berikut:

   - UntukJenis Otentikasi, pilihADF.
   - UntukPengguna,masukkan alamat email pengguna (misalnya,**`jane@example.com`**).
   - UntukKata Sandi, masukkan kata sandi ADFS pengguna.
   - UntukTuan Rumah IdP, masukkan nama server AD FS (misalnya,**`adfs.example.com`**).
   - UntukIdP Pelabuhan, gunakan nilai default443.
   - PilihSSL Tidak Amanpilihan.


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. PilihOKEmenutupPilihan Otentikasi.
9. PilihTesuntuk menguji koneksi, atauOKEuntuk menyelesaikan.

Mengkonfigurasi SSO untuk ODBC menggunakan plugin Okta dan Penyedia Identitas Okta

Halaman ini menjelaskan cara mengonfigurasi driver ODBC Amazon Athena dan plugin Okta untuk menambahkan kemampuan masuk tunggal (SSO) menggunakan penyedia identitas Okta.

Prasyarat

Menyelesaikan langkah-langkah dalam tutorial ini membutuhkan yang berikut:

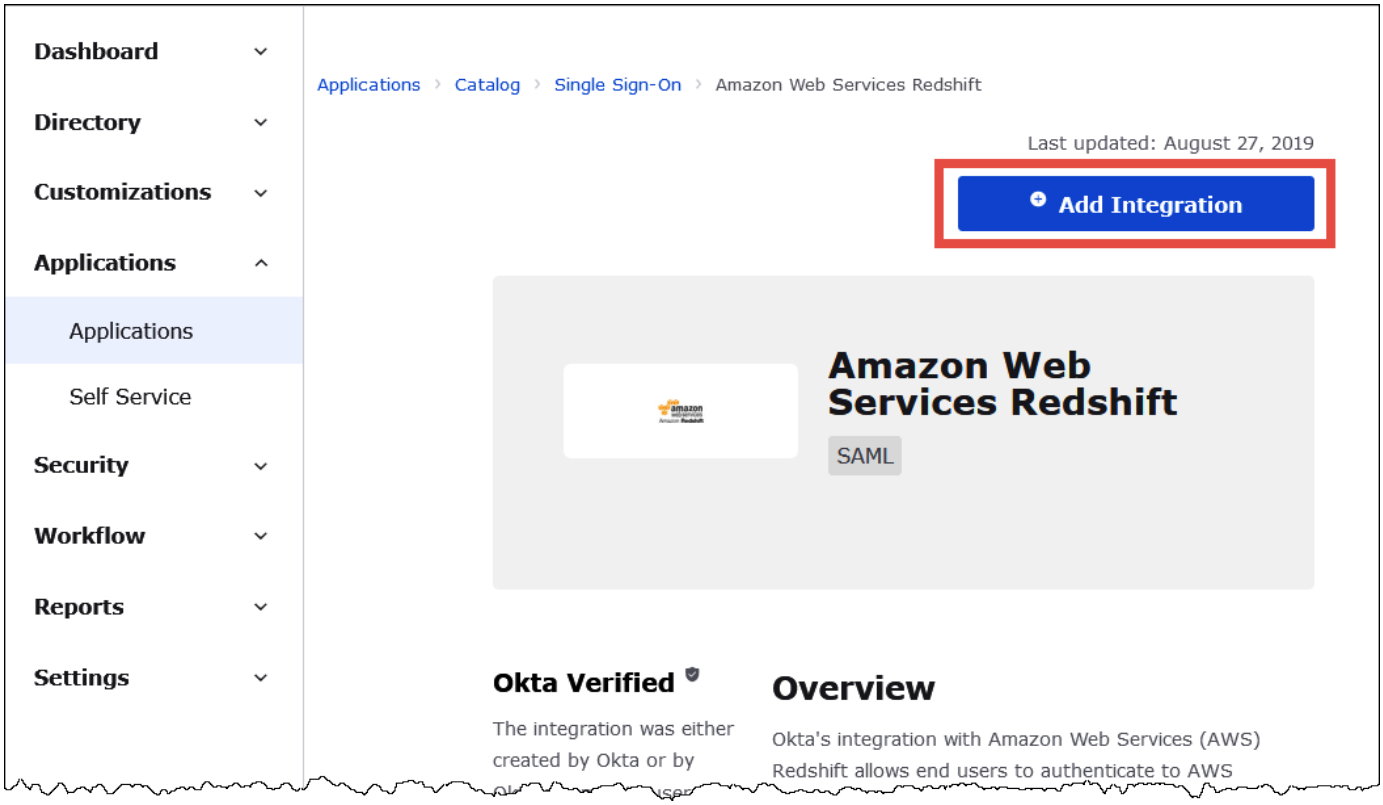
- Pengemudi Amazon Athena ODBC. Untuk tautan unduhan, lihatMenghubungkan ke Amazon Athena dengan ODBC.
- Peran IAM yang ingin Anda gunakan dengan SAMP. Untuk informasi selengkapnya, lihat Membuat peran untuk federasi SAMP 2.0 di Panduan Pengguna IAM.
- Akun Okta. Untuk informasi, kunjungi Okta.com.

Membuat integrasi aplikasi di Okta

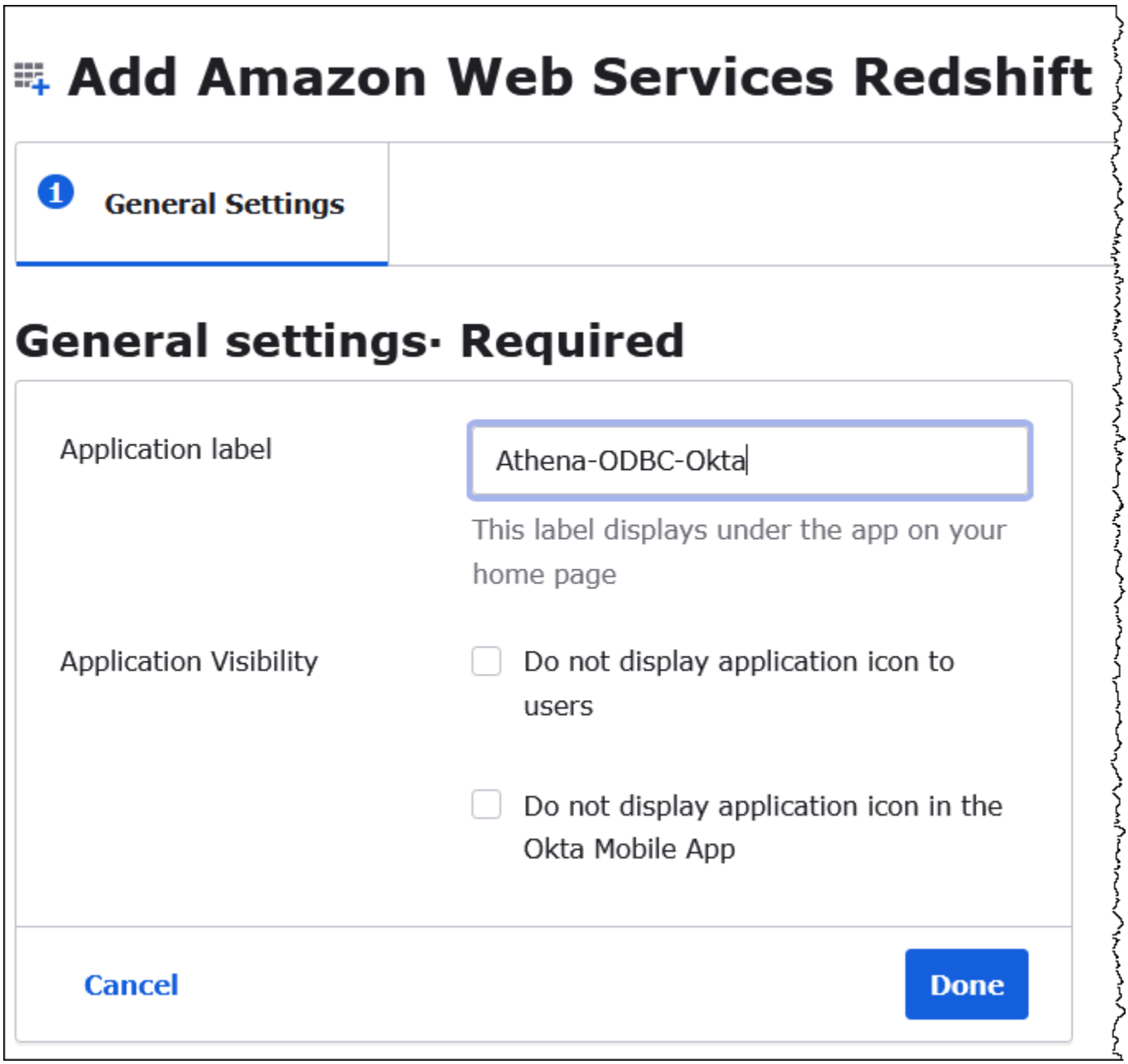
Pertama, gunakan dasbor Okta untuk membuat dan mengonfigurasi aplikasi SAMP 2.0 untuk masuk tunggal ke Athena. Anda dapat menggunakan aplikasi Redshift yang ada di Okta untuk mengonfigurasi akses ke Athena.

Untuk membuat integrasi aplikasi di Okta

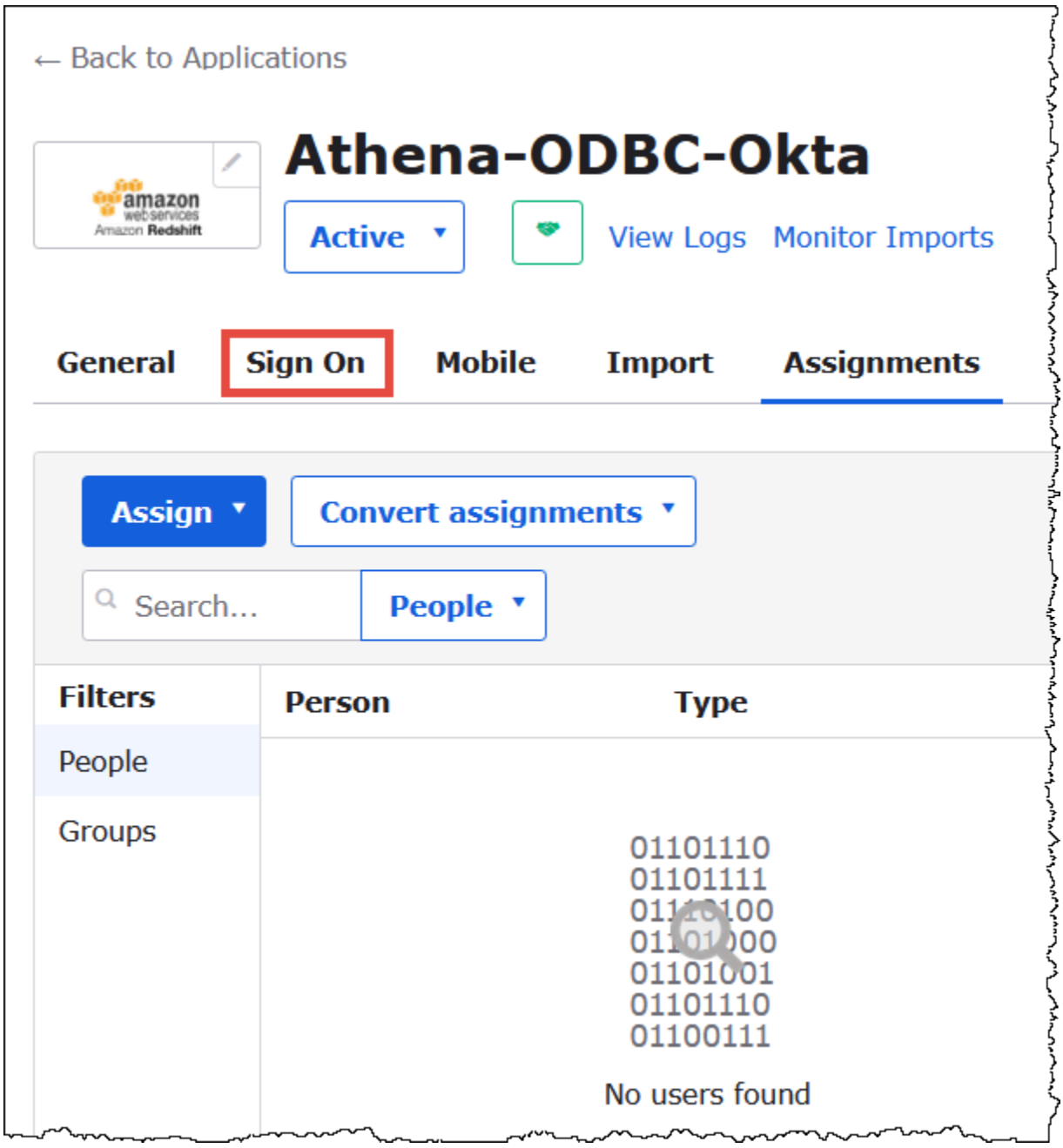
1. Masuk ke halaman admin untuk akun Anda di Okta.com.
2. Di panel navigasi, pilih Aplikasi, Aplikasi.
3. Pada halaman Aplikasi, pilih Jelajahi Katalog Aplikasi.
4. Pada halaman Browse App Integration Catalog, di bagian Use Case, pilih Semua Integrasi.
5. Di kotak pencarian, masukkan Amazon Web Services Redshift, lalu pilih Amazon Web Services Redshift SAMP.
6. Pilih Tambahkan Integrasi.



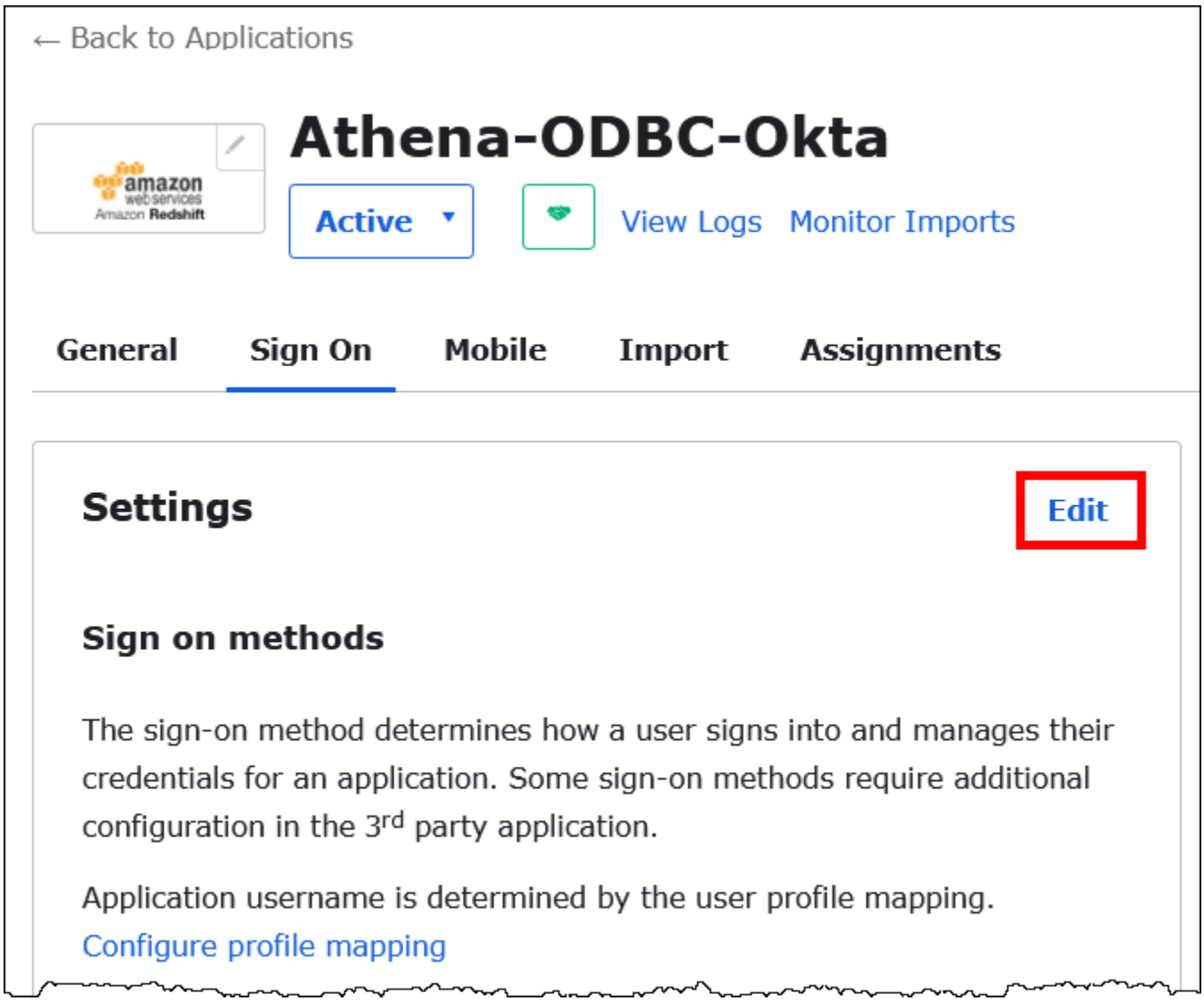
7. Di bagian General Settings Required, untuk label Aplikasi, masukkan nama untuk aplikasi. Tutorial ini menggunakan nama Athena-ODBC-OKTA.



8. Pilih Selesai.
9. Pada halaman untuk aplikasi Okta Anda (misalnya, Athena-ODBC-OKTA), pilih Masuk.



10. Di bagian Pengaturan, pilih Edit.



11. Di bagian Pengaturan Masuk Lanjutan, konfigurasikan nilai berikut.

    - Untuk IDP ARN dan ARN Peran, masukkan ARN IDP dan ARN Peran Anda sebagai nilai yang AWS dipisahkan koma. Untuk informasi tentang format peran IAM, lihat Mengonfigurasi pernyataan SAMP untuk respons autentikasi di Panduan Pengguna IAM.
    - Untuk Durasi Sesi, masukkan nilai antara 900 dan 43200 detik. Tutorial ini menggunakan default 3600 (1 jam).

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

Pengaturan DbUser Format, AutoCreate, dan DBGroups yang Diizinkan tidak digunakan oleh Athena. Anda tidak perlu mengkonfigurasinya.

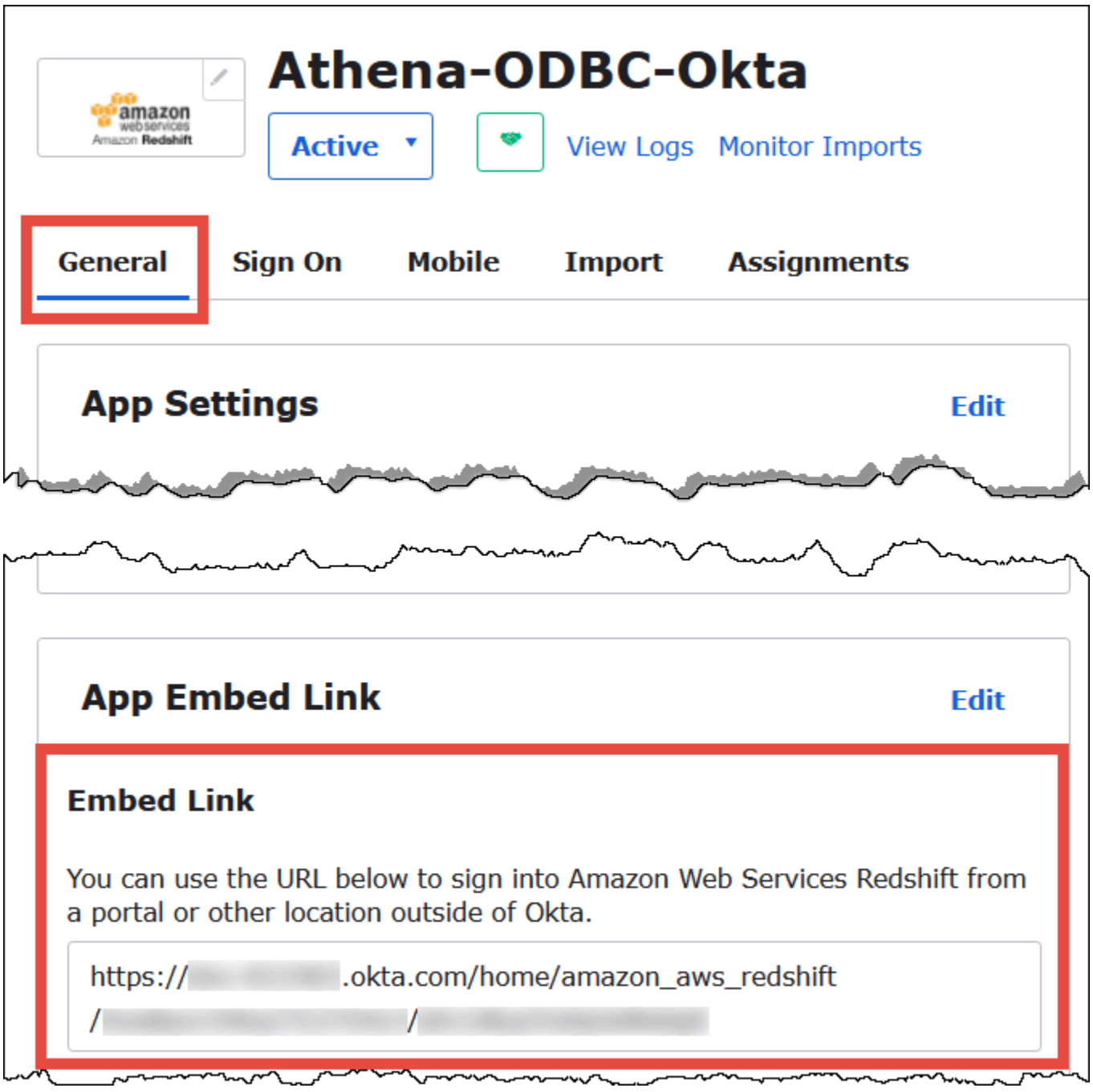
12. Pilih Simpan.

Ambil informasi konfigurasi ODBC dari Okta

Sekarang setelah Anda membuat aplikasi Okta, Anda siap untuk mengambil ID aplikasi dan URL host IDP. Anda akan memerlukan ini nanti ketika Anda mengkonfigurasi ODBC untuk koneksi ke Athena.

Untuk mengambil informasi konfigurasi ODBC dari Okta

1. Pilih tab Umum aplikasi Okta Anda, lalu gulir ke bawah ke bagian App Embed Link.



URL Embed Link Anda dalam format berikut:

```
https://trial-1234567.okta.com/home/amazon_aws_redshift/Abc1de2fghi3J45kL678/
abc1defghij2klmNo3p4
```

2. Dari URL Embed Link Anda, ekstrak dan simpan potongan-potongan berikut:

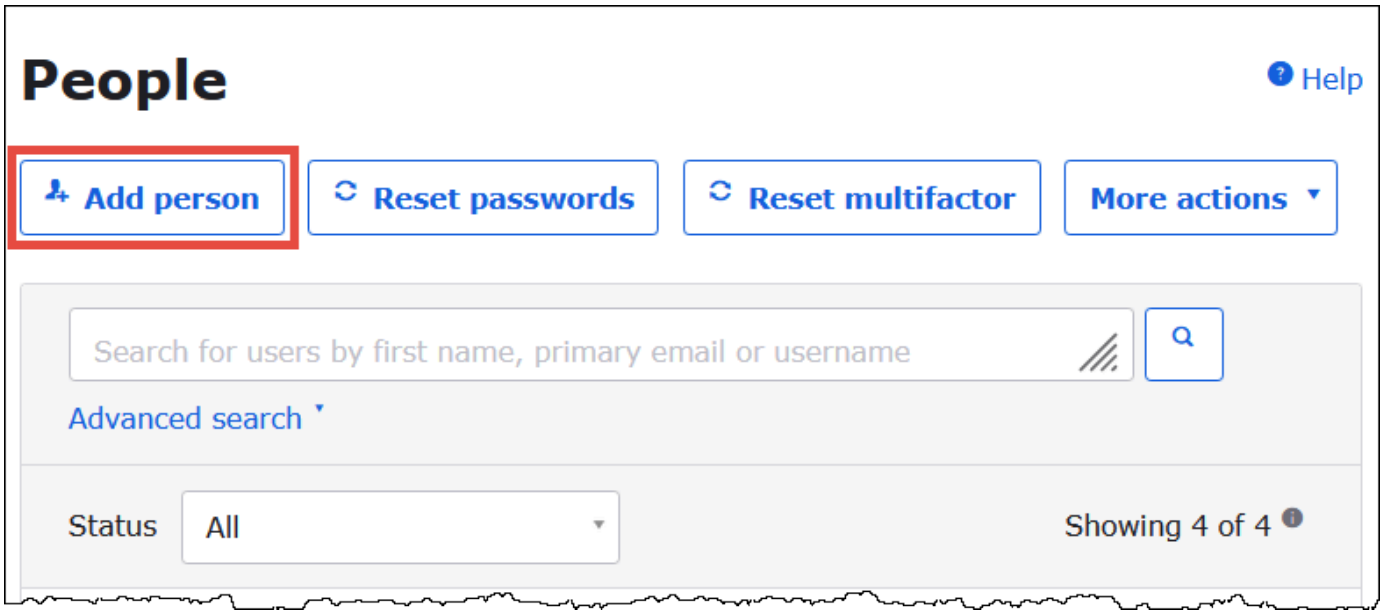
- Segmen pertama setelah`https://`, hingga dan termasuk `okta.com` (misalnya, trial-1234567.okta.com). Ini adalah host IDP Anda.
- Dua segmen terakhir dari URL, termasuk garis miring ke depan di tengah. Segmen adalah dua string 20 karakter dengan campuran angka dan huruf besar dan kecil (misalnya, ABC1de2FGHI3J45KL678/ABC1defgHiJ2KLmno3P4). Ini adalah ID aplikasi Anda.

Tambahkan pengguna ke aplikasi Okta

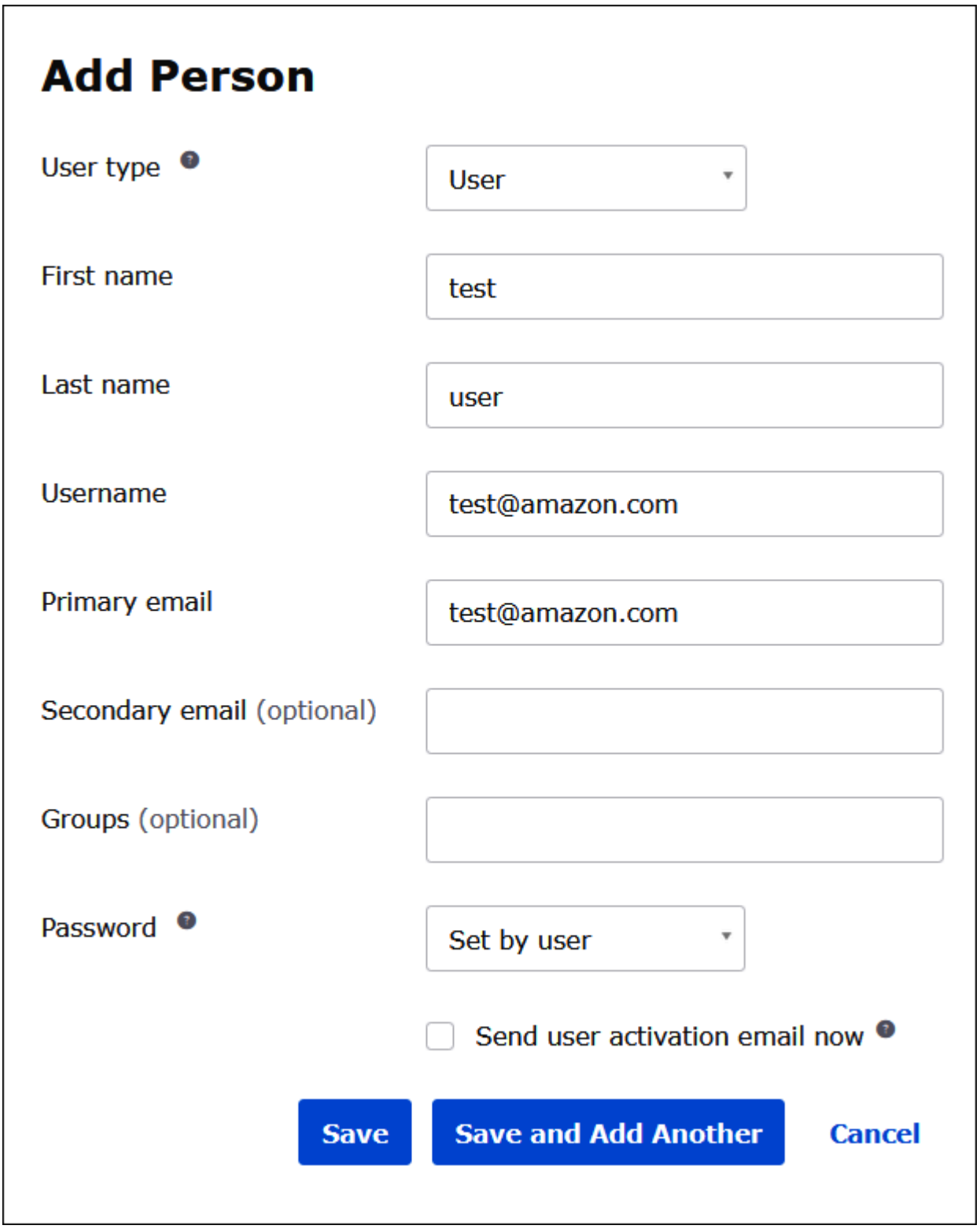
Sekarang Anda siap untuk menambahkan pengguna ke aplikasi Okta Anda.

Untuk menambahkan pengguna ke aplikasi Okta

1. Di panel navigasi kiri, pilihDirektori, lalu pilihOrang.
2. Pilih Tambah orang.



3. Dalam kotak dialog Tambah Orang, masukkan informasi berikut.
   - Masukkan nilai untukNama depandanNama terakhir. Tutorial ini menggunakan **`test user`**.
   - Masukkan nilai untuk Nama Pengguna dan email Utama. Tutorial ini digunakan **`test@amazon.com`** untuk keduanya. Persyaratan keamanan Anda untuk kata sandi mungkin berbeda.

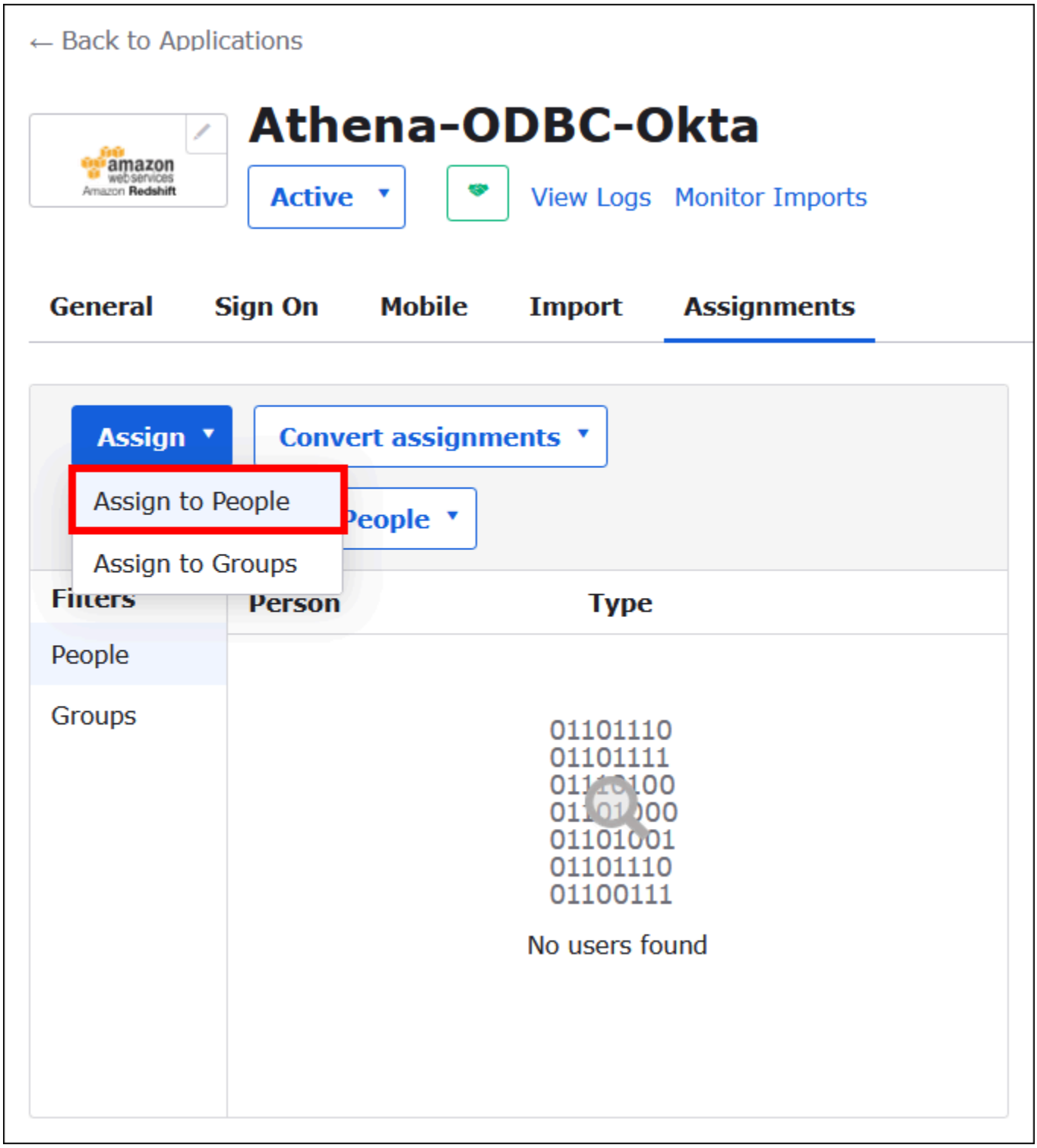


4. Pilih Simpan.

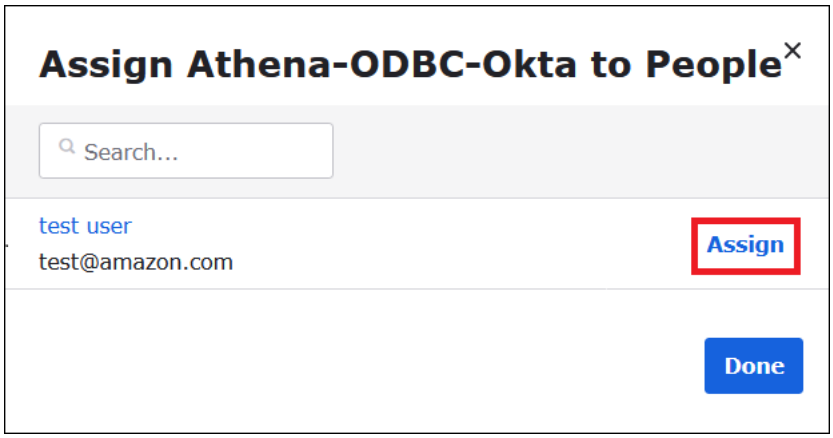
Sekarang Anda siap untuk menetapkan pengguna yang Anda buat untuk aplikasi Anda.

Untuk menetapkan pengguna ke aplikasi Anda:

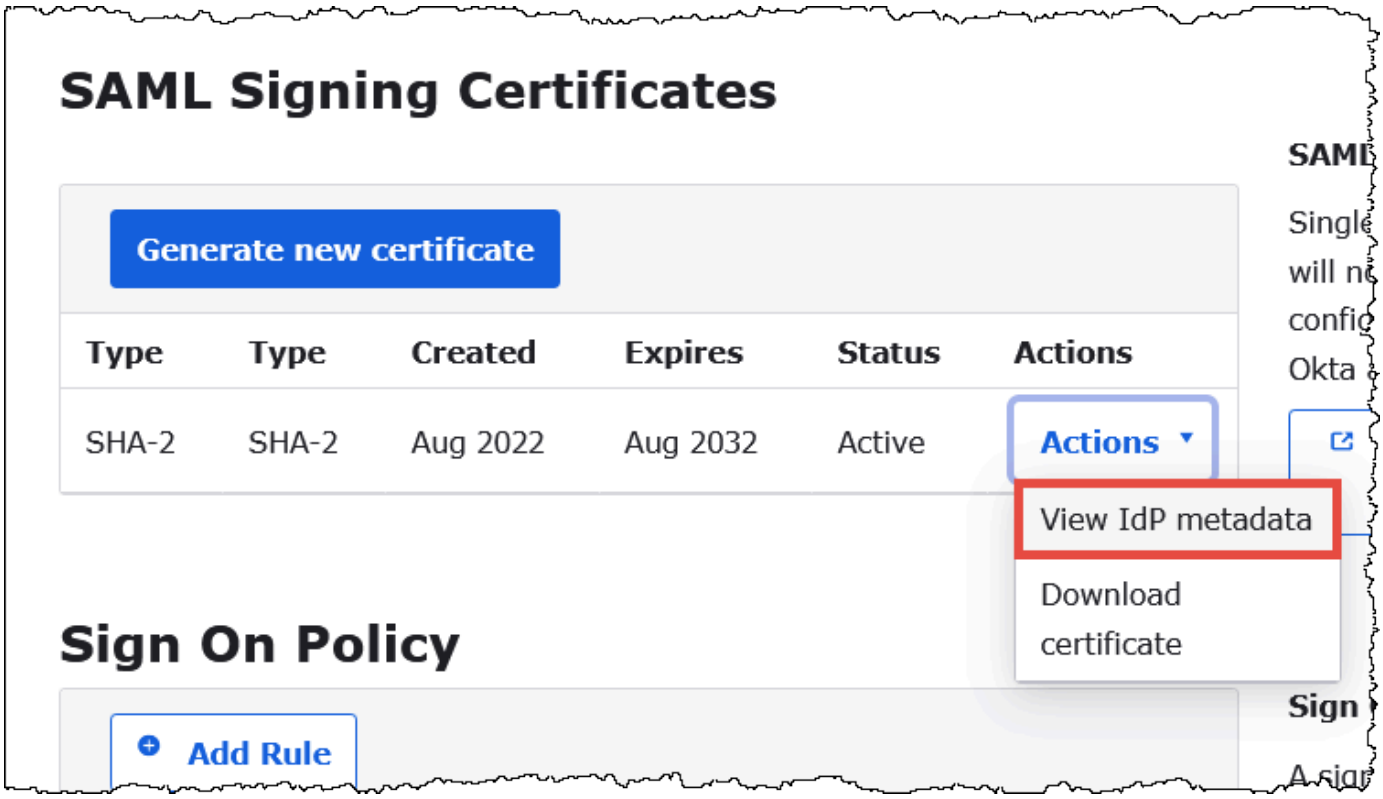
1. Di panel navigasi, pilih Aplikasi, Aplikasi, lalu pilih nama aplikasi Anda (misalnya, Athena-ODBC-OKTA).
2. Pilih Tetapkan, lalu pilih Tetapkan ke Orang.



3. Pilih opsi Tetapkan untuk pengguna Anda, lalu pilih Selesai.



4. Pada prompt, pilih Simpan dan Kembali. Kotak dialog menunjukkan status pengguna sebagai Ditugaskan.
5. Pilih Selesai.
6. Pilih tab Masuk.
7. Gulir ke bawah ke bagian Sertifikat Penandatanganan SAMP.
8. Pilih Tindakan.
9. Buka menu konteks (klik kanan) untuk Lihat metadata IDP, lalu pilih opsi browser untuk menyimpan file.
10. Simpan file dengan `.xml` ekstensi.

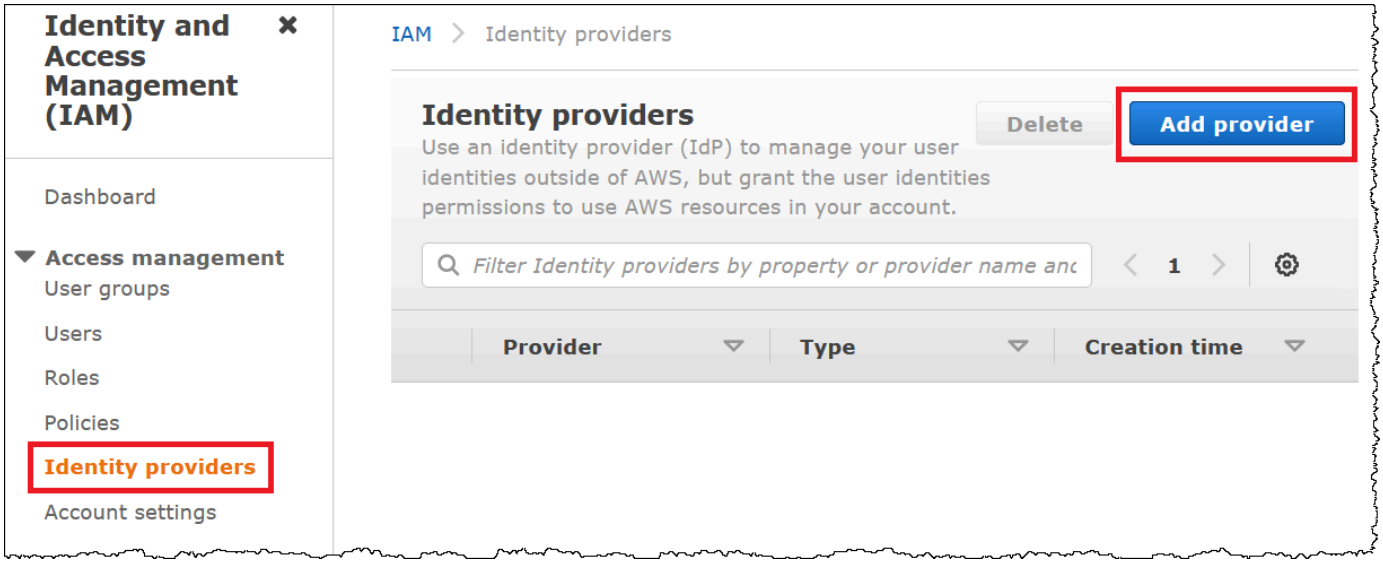


Buat Penyedia dan AWS Peran Identitas SAMP

Sekarang Anda siap untuk mengunggah file XMLmetadata ke konsol IAM di. AWS Anda akan menggunakan file ini untuk membuat penyedia dan peran identitas AWS SAMP. Gunakan akun administrator AWS Layanan untuk melakukan langkah-langkah ini.

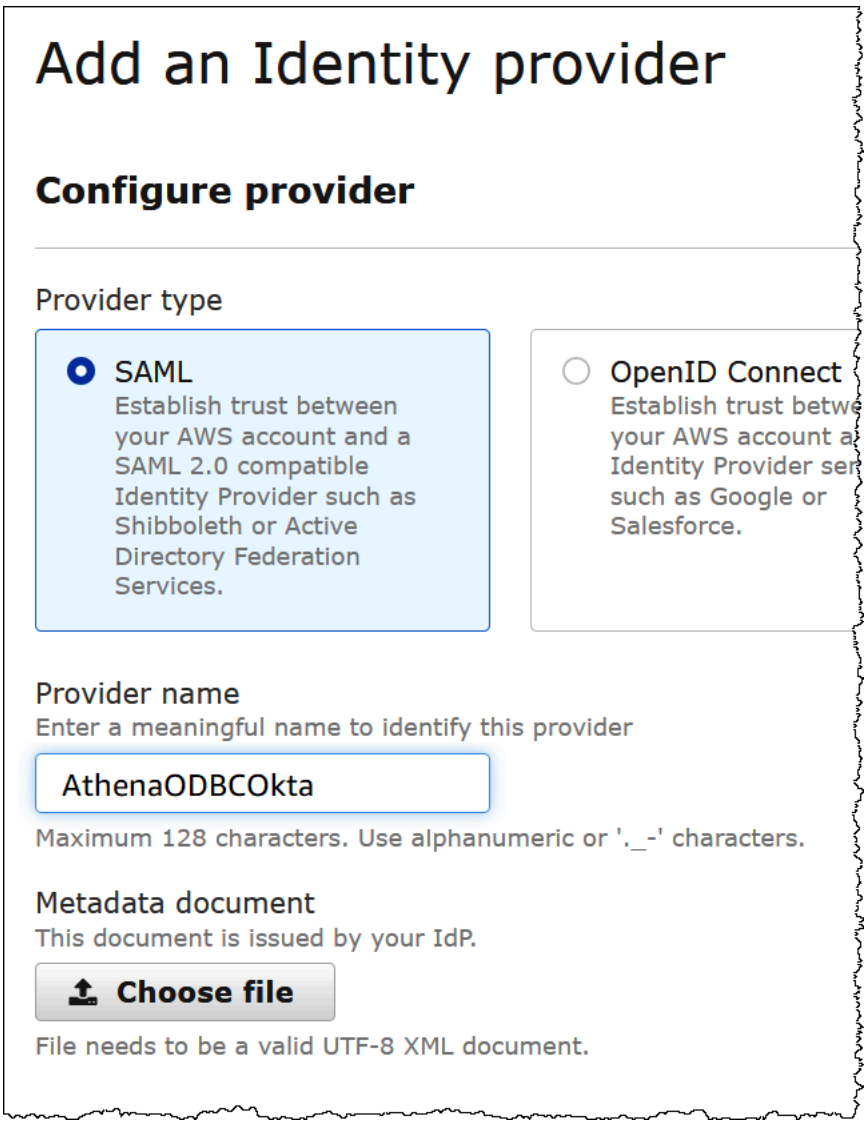
Untuk membuat penyedia identitas SAMP dan berperan dalam AWS

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/IAM/.
2. Di panel navigasi, pilih Penyedia identitas, lalu pilih Tambahkan penyedia.



3. Pada halaman Add an Identity provider, untuk Configure provider, masukkan informasi berikut.

   - Untuk tipe Provider, pilih SAMP.
   - Untuk nama Penyedia, masukkan nama untuk penyedia Anda (misalnya, **`AthenaODBCOkta`**).
   - Untuk dokumen Metadata, gunakan opsi Pilih file untuk mengunggah file XMLmetadata penyedia identitas (iDP) yang Anda unduh.



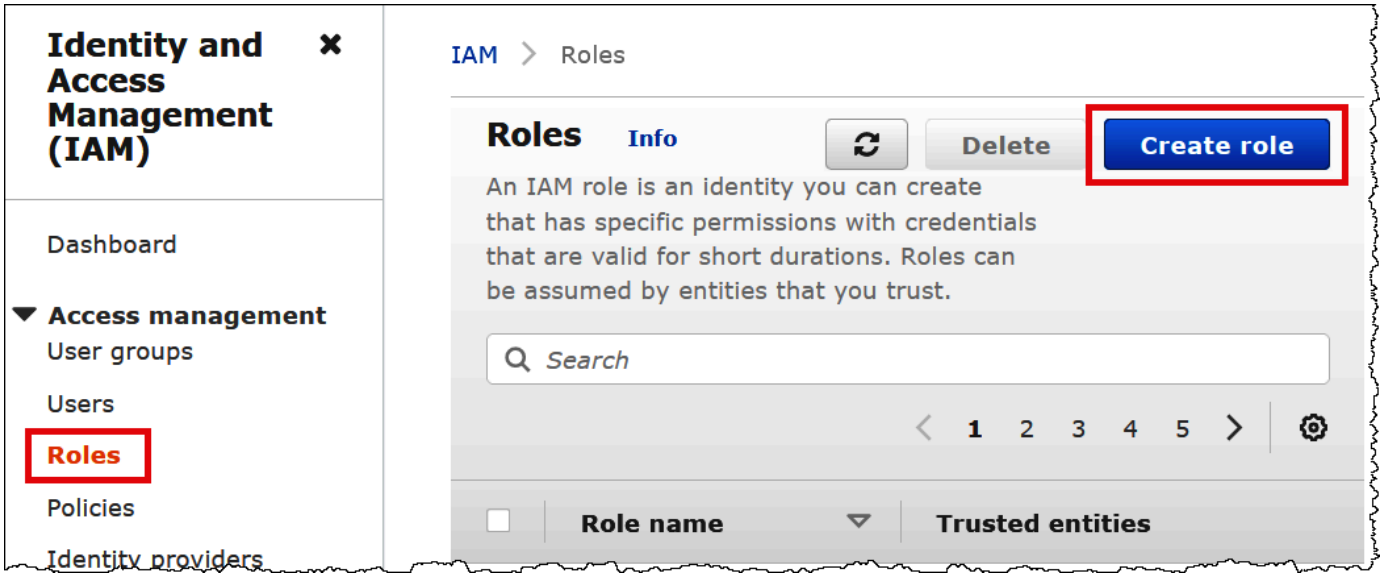
4. Pilih Tambah penyedia.

Membuat peran IAM untuk akses Athena dan Amazon S3

Sekarang Anda siap membuat peran IAM untuk akses Athena dan Amazon S3. Anda akan menetapkan peran ini untuk pengguna Anda. Dengan begitu, Anda dapat memberi pengguna akses masuk tunggal ke Athena.

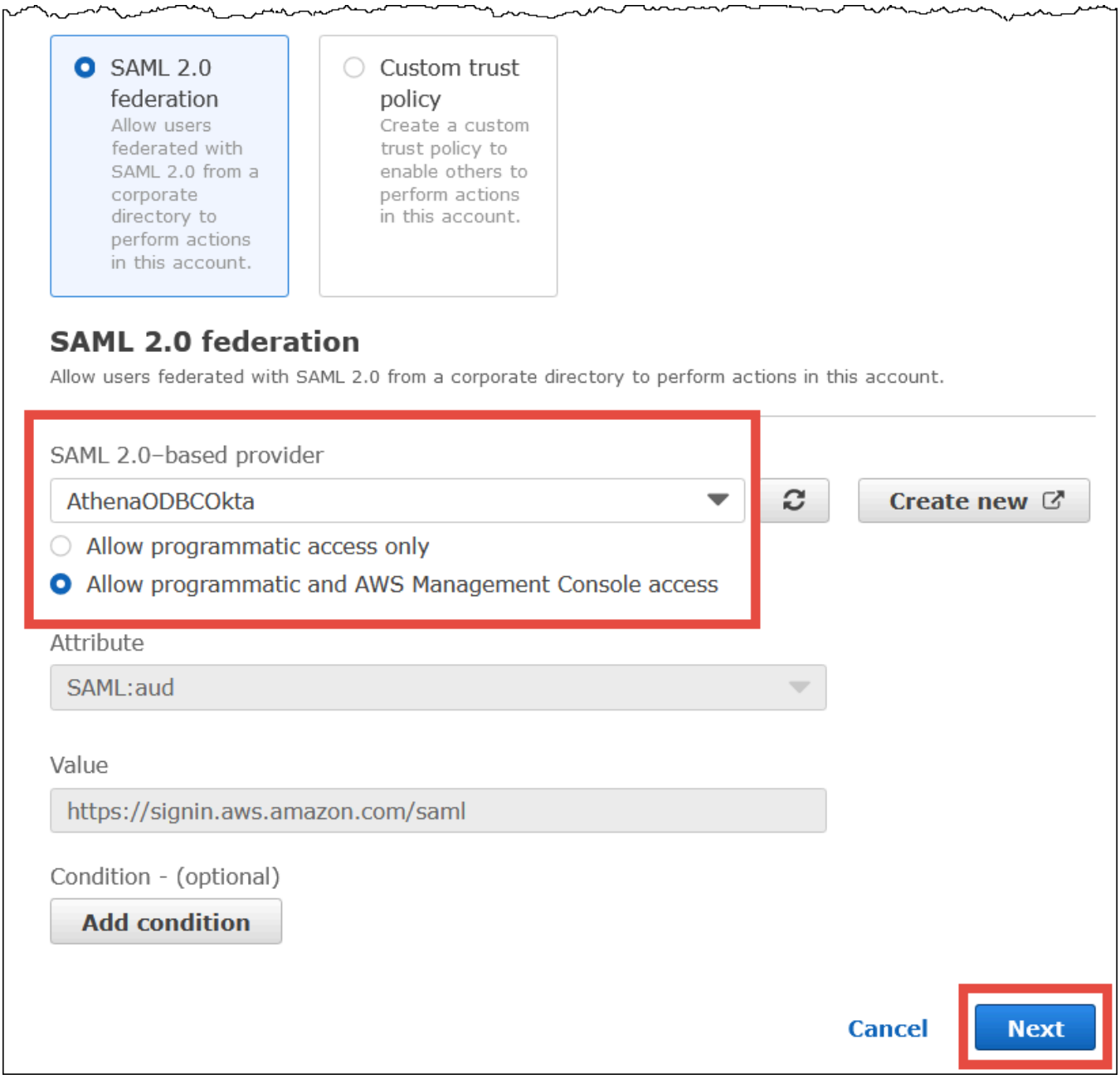
Untuk membuat peran IAM bagi pengguna Anda

1. Di panel navigasi konsol IAM, pilih Peran, dan lalu pilih Buat peran.

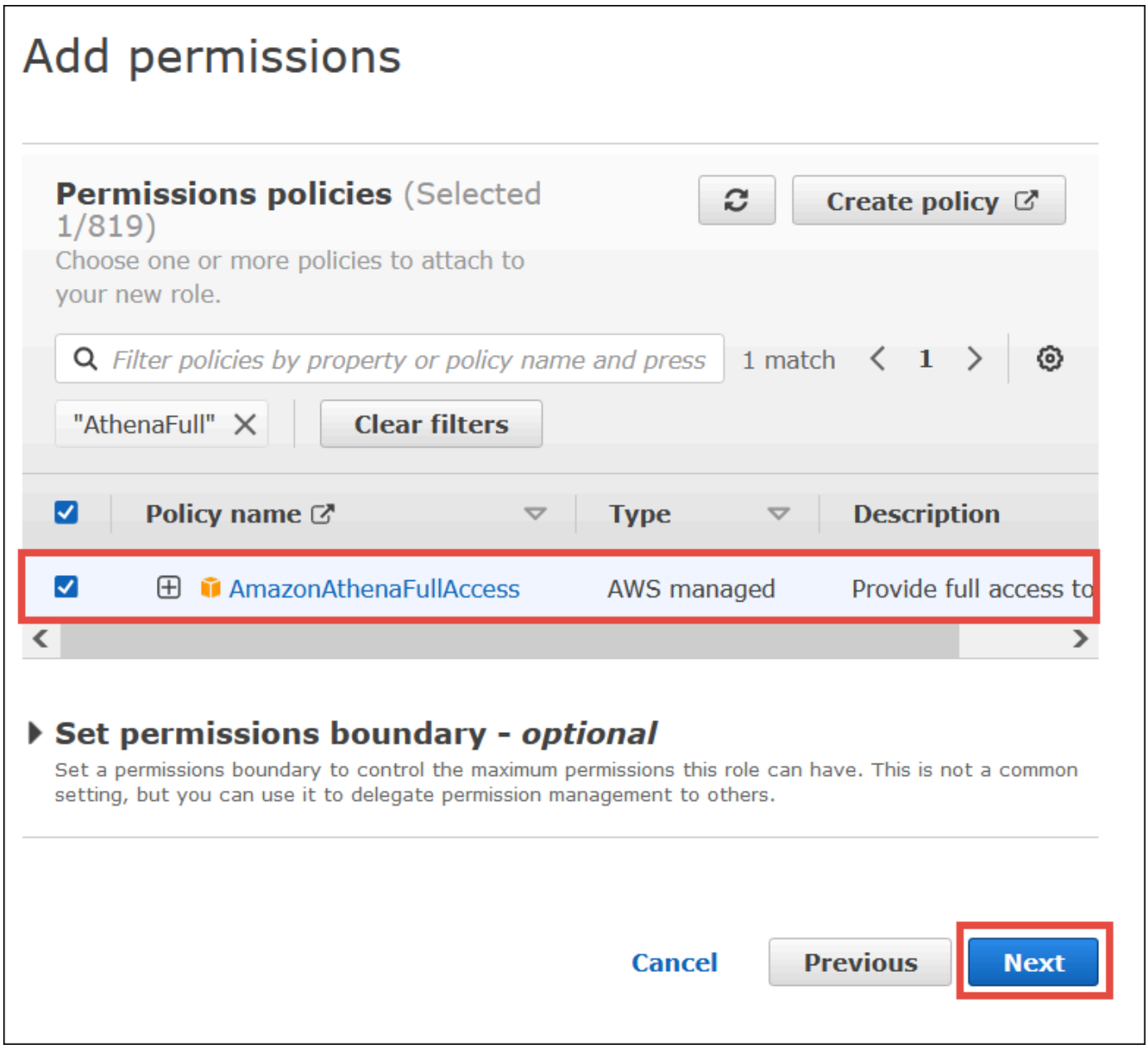


2. Pada halaman Buat peran, pilih opsi berikut:

   - Untuk Pilih jenis entitas terpercaya, pilih Federasi SAML 2.0.
   - Untuk penyedia berbasis SAMP 2.0, pilih penyedia identitas SAMP yang Anda buat (misalnya, AthenaODBCokta).
   - Pilih Izinkan programatik dan AWS Management Console akses.



3. Pilih Berikutnya.
4. Pada halaman Tambahkan Izin, untuk kebijakan Filter, masukkan**`AthenaFull`**, lalu tekan ENTER.
5. Pilih kebijakan `AmazonAthenaFullAccess` terkelola, lalu pilih Berikutnya.



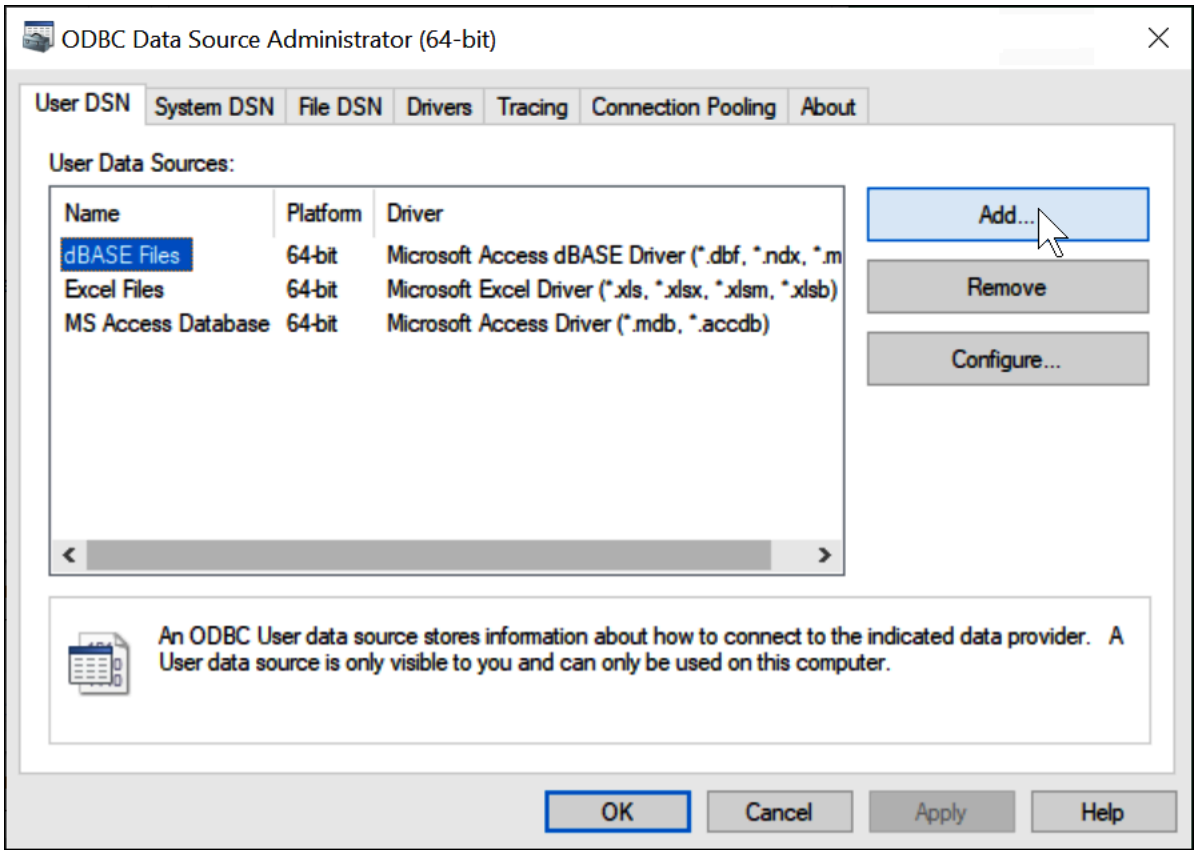
6. Pada halaman Nama, tinjau, dan buat, untuk nama Peran, masukkan nama untuk peran tersebut (misalnya,**`Athena-ODBC-OktaRole`**), lalu pilih Buat peran.

Mengkonfigurasi koneksi Okta ODBC ke Athena

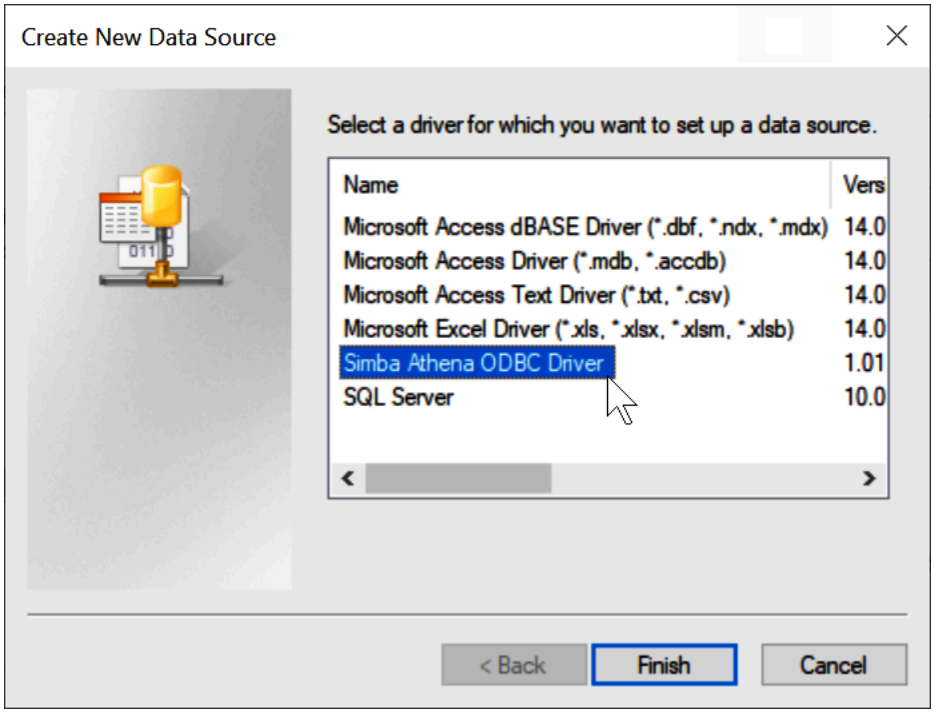
Sekarang Anda siap untuk mengkonfigurasi koneksi Okta ODBC ke Athena menggunakan program Sumber Data ODBC di Windows.

Untuk mengonfigurasi koneksi Okta ODBC Anda ke Athena

1. Di Windows, luncurkan program Sumber Data ODBC.
2. Dalam program Administrator Sumber Data ODBC, pilih Tambah.

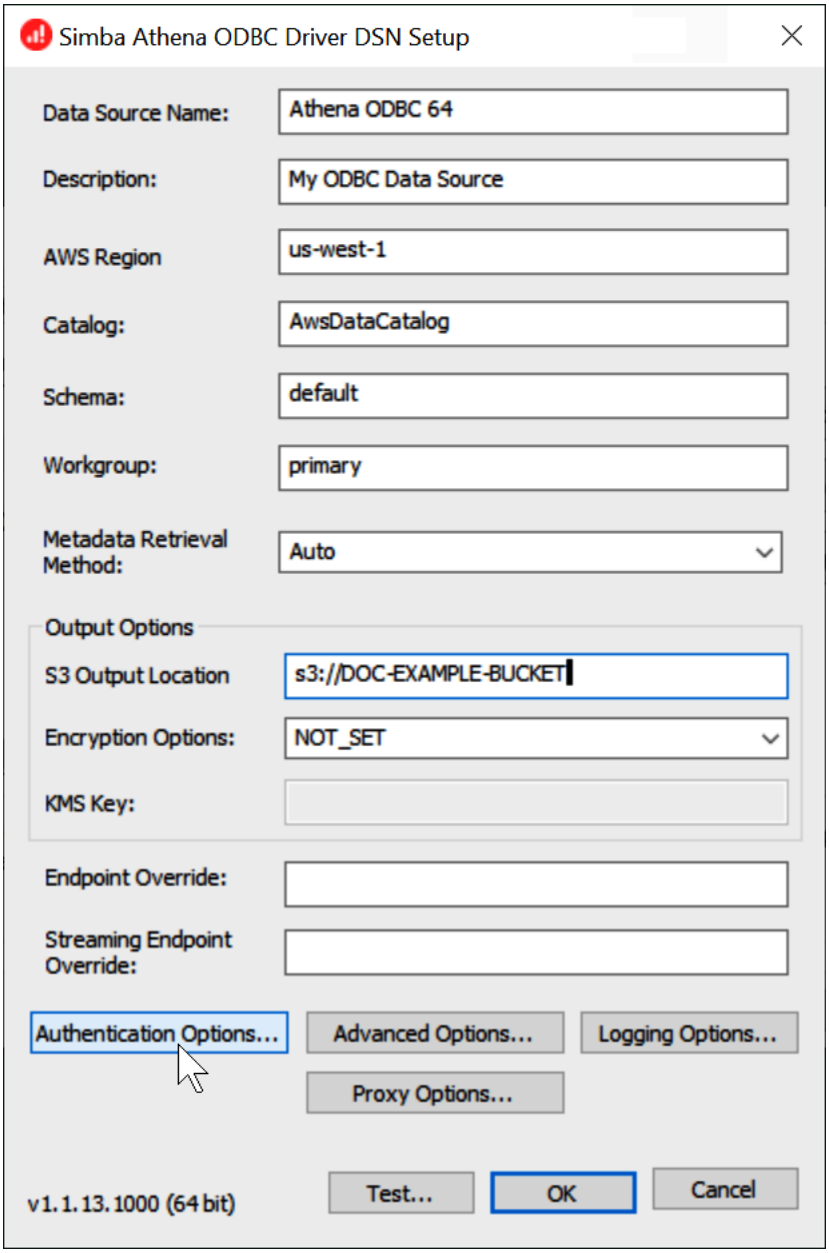


3. Pilih Simba Athena ODBC Driver, lalu pilih Finish.



4. Di Simba Athena ODBC Driver DSN Setup dialog, masukkan nilai yang dijelaskan.

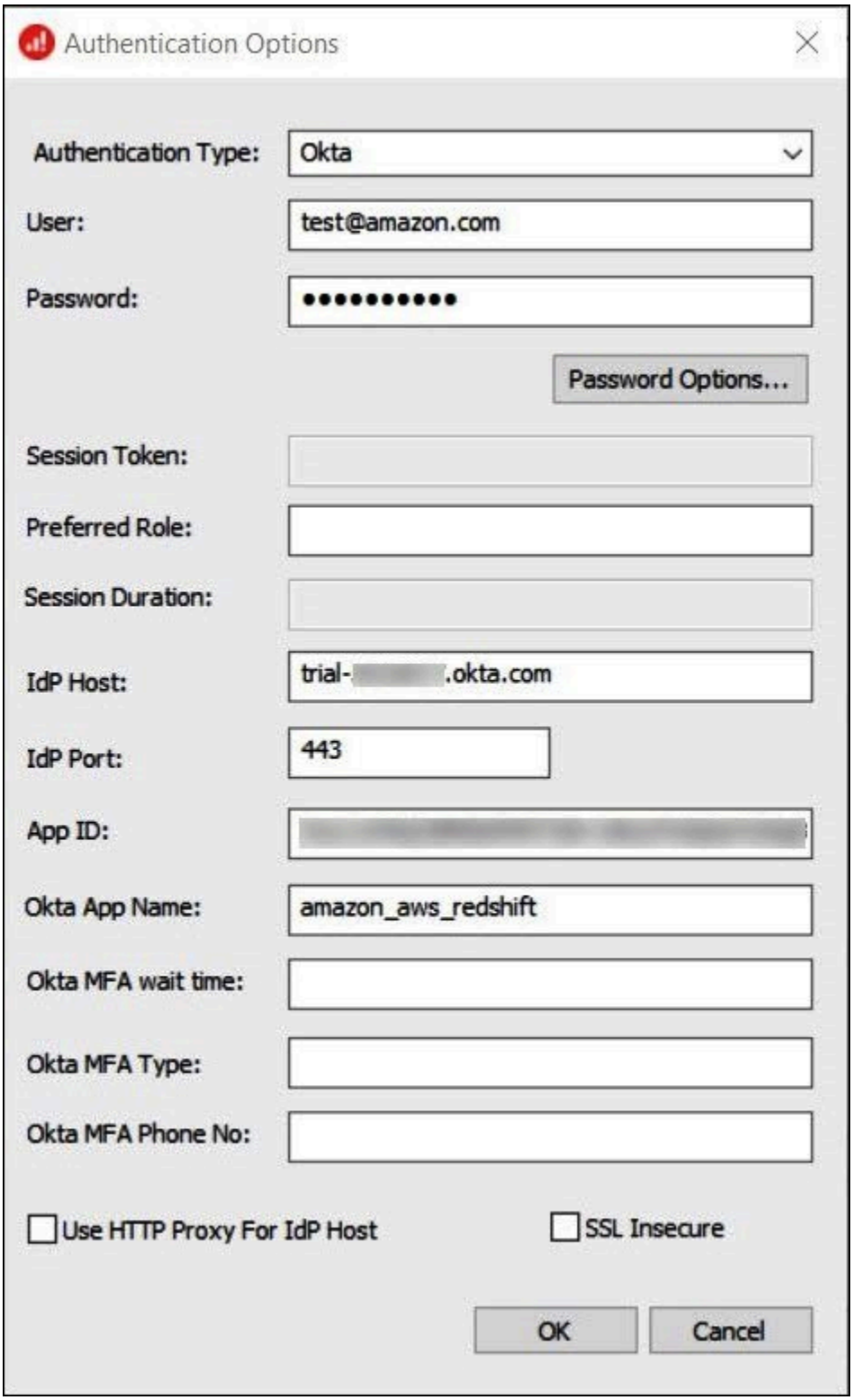
   - Untuk Nama Sumber Data, masukkan nama untuk sumber data Anda (misalnya,**`Athena ODBC 64`**).
   - Untuk Deskripsi, masukkan deskripsi untuk sumber data Anda.
   - Untuk Wilayah AWS, masukkan Wilayah AWS yang Anda gunakan (misalnya,**`us-west-1`**).
   - Untuk Lokasi Output S3, masukkan jalur Amazon S3 tempat Anda ingin output disimpan.



5. Pilih Opsi Otentikasi.
6. Dalam kotak dialog Opsi Otentikasi, pilih atau masukkan nilai berikut.

   - Untuk Jenis Otentikasi, pilih Okta.
   - Untuk Pengguna, masukkan nama pengguna Okta Anda.
   - Untuk Kata Sandi, masukkan kata sandi Okta Anda.
   - Untuk IDP Host, masukkan nilai yang Anda rekam sebelumnya (misalnya,**`trial-1234567.okta.com`**).
   - Untuk IDP Port, masukkan. **`443`**

- Untuk ID Aplikasi, masukkan nilai yang Anda rekam sebelumnya (dua segmen terakhir dari tautan embed Okta Anda).
- Untuk Nama Aplikasi Okta, masukkan**`amazon_aws_redshift`**.



7. Pilih OK.
8. Pilih Uji untuk menguji koneksi atau OK untuk menyelesaikan.

Mengkonfigurasi sistem masuk tunggal menggunakan ODBC, SAMP 2.0, dan Penyedia Identitas Okta

Untuk terhubung ke sumber data, Anda dapat menggunakan Amazon Athena dengan penyedia identitas (IdPs) seperti PingOne, Okta OneLogin, dan lainnya. Dimulai dengan driver Athena ODBC versi 1.1.13 dan driver Athena JDBC versi 2.0.25, plugin SALL browser disertakan yang dapat Anda konfigurasikan untuk bekerja dengan penyedia SAFL 2.0 apa pun. Topik ini menunjukkan cara mengonfigurasi driver ODBC Amazon Athena dan plugin SAFL berbasis browser untuk menambahkan kemampuan masuk tunggal (SSO) menggunakan penyedia identitas Okta.

Prasyarat

Menyelesaikan langkah-langkah dalam tutorial ini membutuhkan yang berikut:

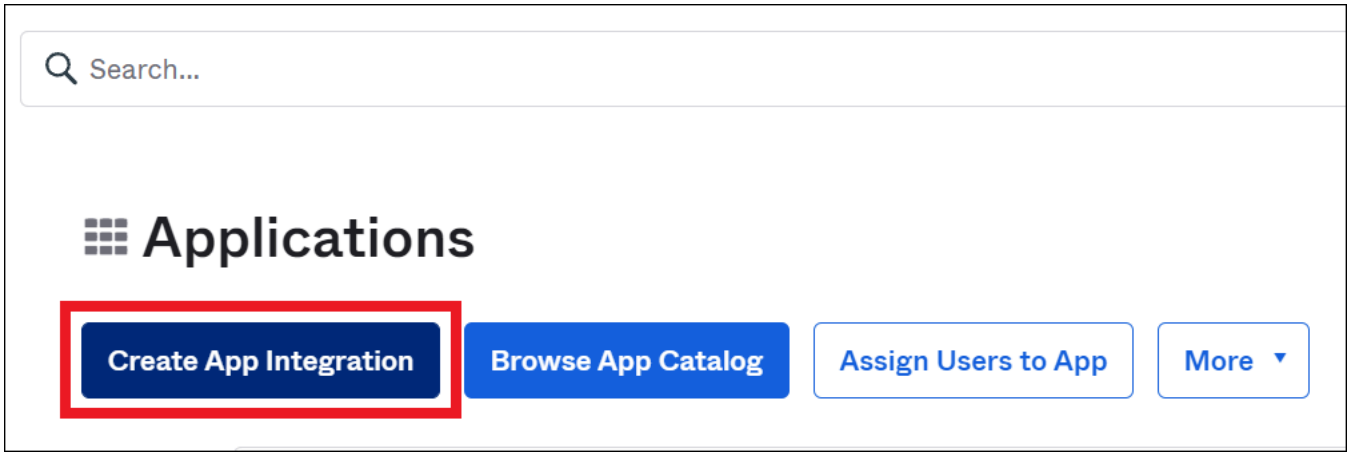
- Athena ODBC driver versi 1.1.13 atau yang lebih baru. Versi 1.1.13 dan yang lebih baru termasuk dukungan SAMP browser. Untuk tautan unduhan, lihat Menghubungkan ke Amazon Athena dengan ODBC.
- Peran IAM yang ingin Anda gunakan dengan SAMP. Untuk informasi selengkapnya, lihat Membuat peran untuk federasi SAMP 2.0 di Panduan Pengguna IAM.
- Akun Okta. Untuk informasi, kunjungi okta.com.

Membuat integrasi aplikasi di Okta

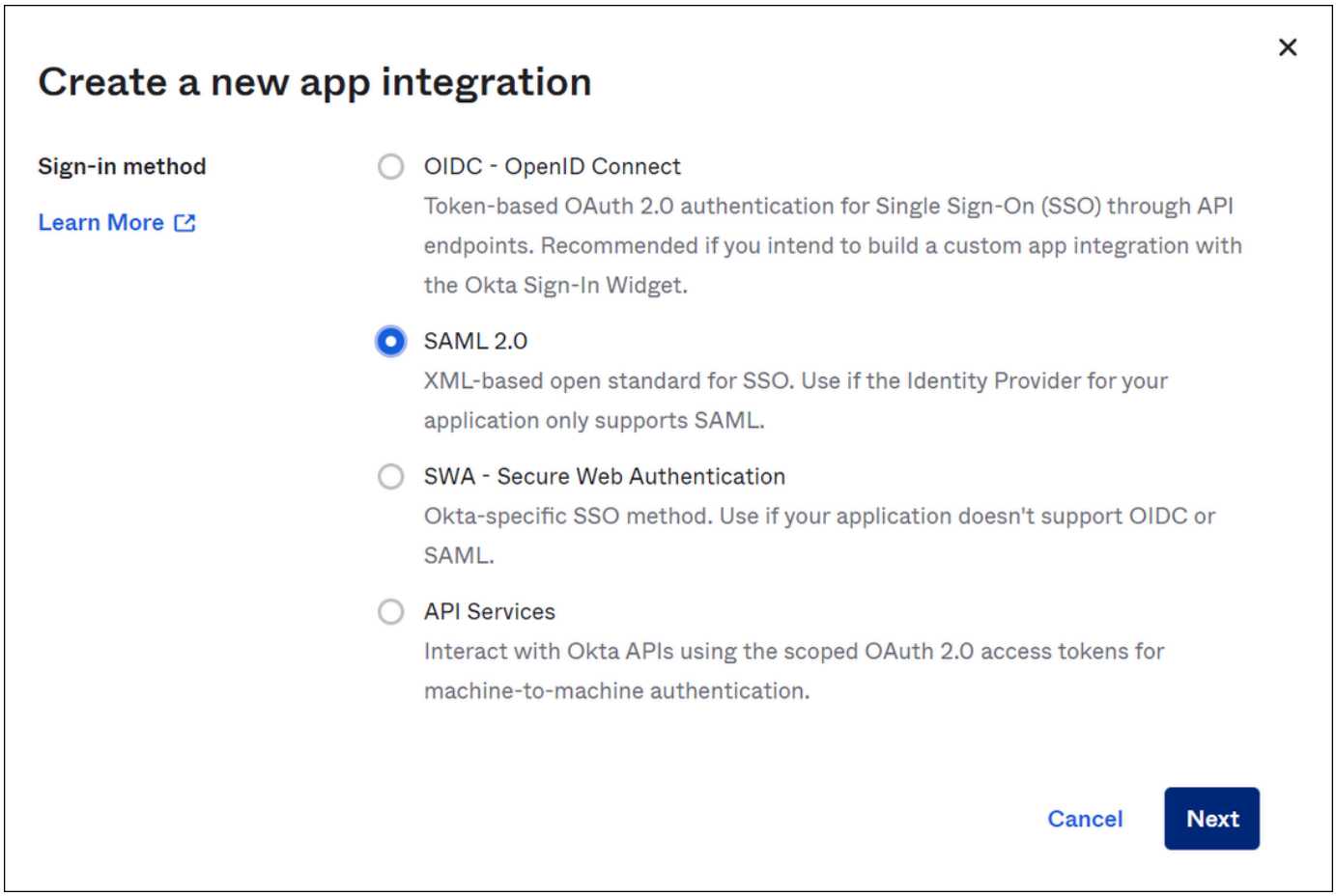
Pertama, gunakan dasbor Okta untuk membuat dan mengonfigurasi aplikasi SAMP 2.0 untuk masuk tunggal ke Athena.

Untuk menggunakan dasbor Okta untuk mengatur sistem masuk tunggal untuk Athena

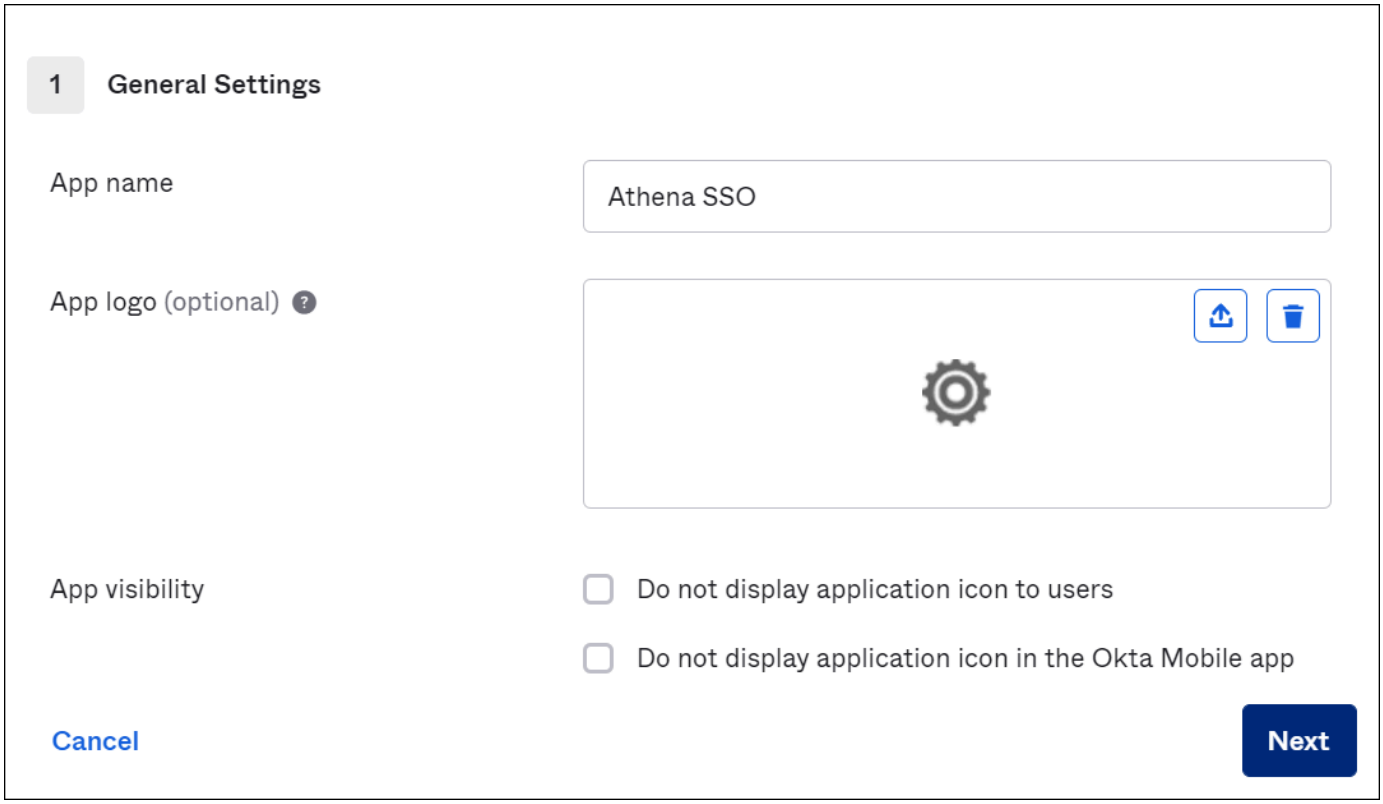
1. Masuk ke halaman admin Okta di `okta.com`.
2. Di panel navigasi, pilih Aplikasi, Aplikasi.
3. Pada halaman Aplikasi, pilih Buat Integrasi Aplikasi.



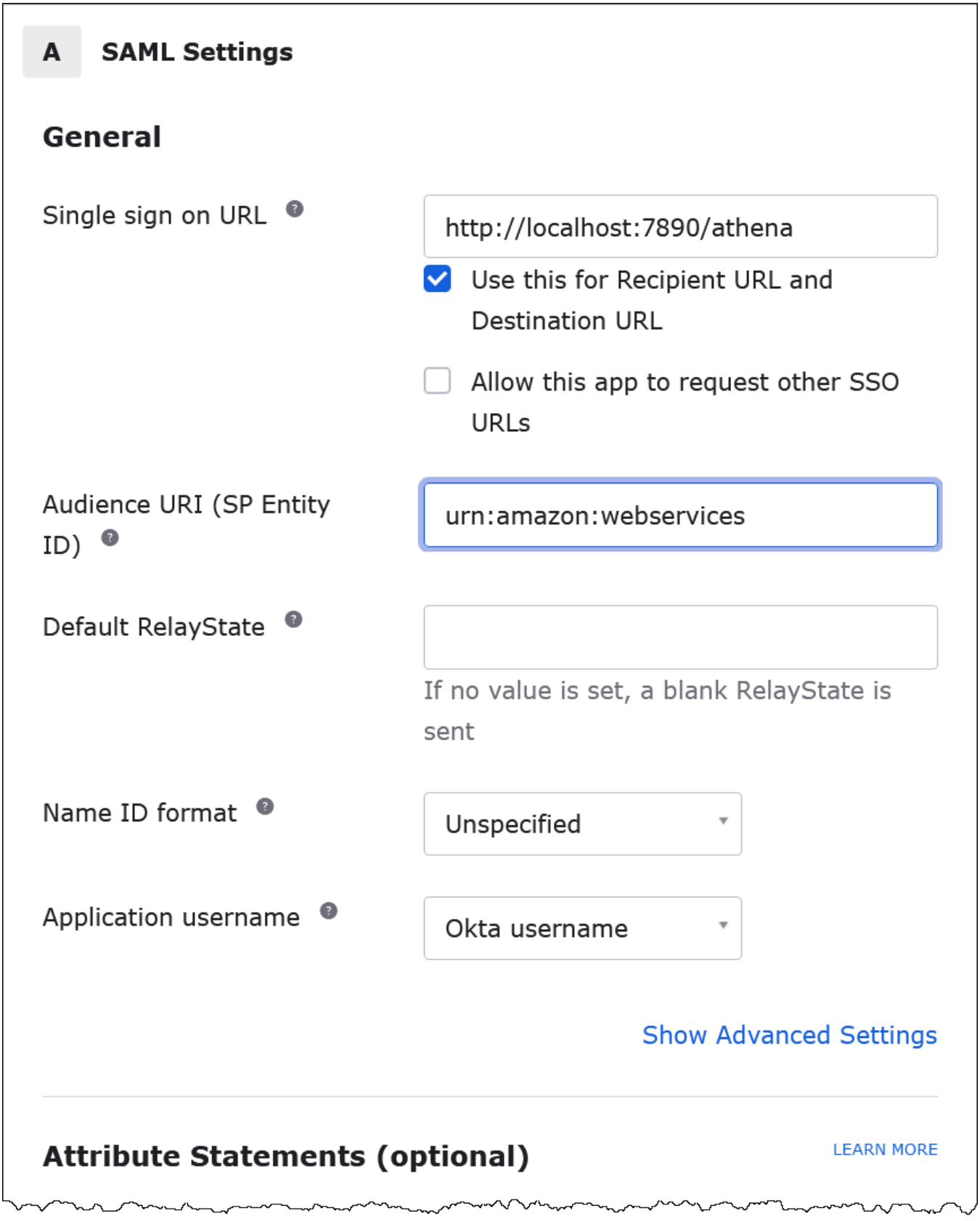
4. Di kotak dialog Buat integrasi aplikasi baru, untuk metode Masuk, pilih SAFL 2.0, lalu pilih Berikutnya.



5. Pada halaman Buat Integrasi SAM, di bagian Pengaturan Umum, masukkan nama untuk aplikasi. Tutorial ini menggunakan nama Athena SSO.



6. Pilih Berikutnya.
7. Pada halaman Konfigurasi SAMP, di bagian Pengaturan SAMP, masukkan nilai berikut:
   - Untuk Single sign on URL, masukkan **`http://localhost:7890/athena`**
   - Untuk URI Audiens, masukkan **`urn:amazon:webservices`**


A
SAML Settings
General
Single sign on URL
http://localhost:7890/athena
Use this for Recipient URL and Destination URL
Allow this app to request other SSO URLs
Audience URI (SP Entity ID)
urn:amazon:webservices
Default RelayState
If no value is set, a blank RelayState is sent
Name ID format
Unspecified
Application username
Okta username
Show Advanced Settings
Attribute Statements (optional)
LEARN MORE

8. Untuk Pernyataan Atribut (opsional), masukkan dua pasangan nama/nilai berikut. Ini adalah atribut pemetaan yang diperlukan.

   - Untuk Nama, masukkan URL berikut:

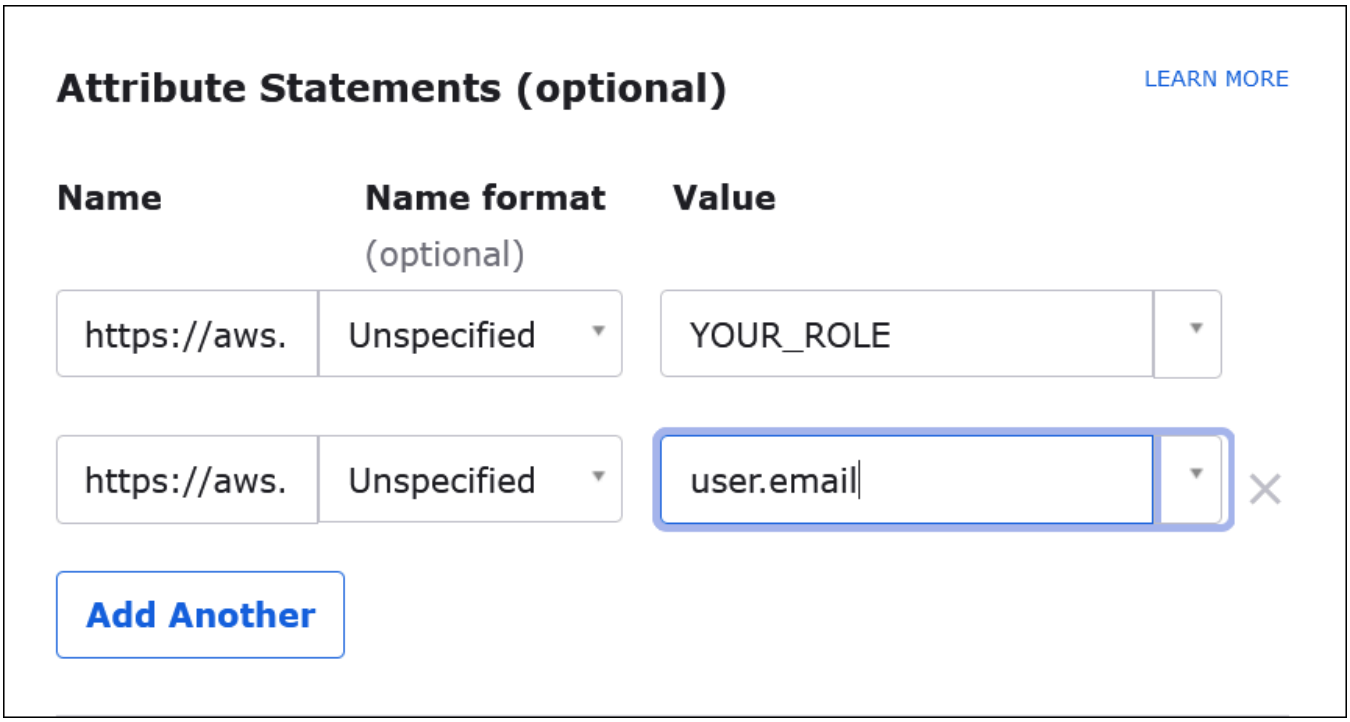
     `https://aws.amazon.com/SAML/Attributes/Role`

     Untuk Nilai, masukkan nama peran IAM Anda. Untuk informasi tentang format peran IAM, lihat Mengonfigurasi pernyataan SAFL untuk respons autentikasi di Panduan Pengguna IAM.

   - Untuk Nama, masukkan URL berikut:

     `https://aws.amazon.com/SAML/Attributes/RoleSessionName`

     Untuk Nilai, masukkan **`user.email`**.



9. Pilih Selanjutnya, dan kemudian pilih Selesai.

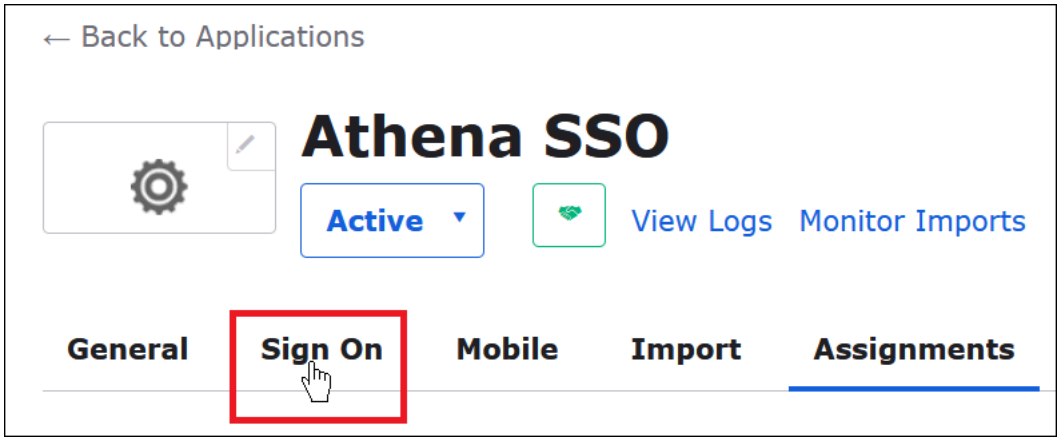
   Ketika Okta membuat aplikasi, itu juga membuat URL login Anda, yang akan Anda ambil berikutnya.

Mendapatkan URL login dari dasbor Okta

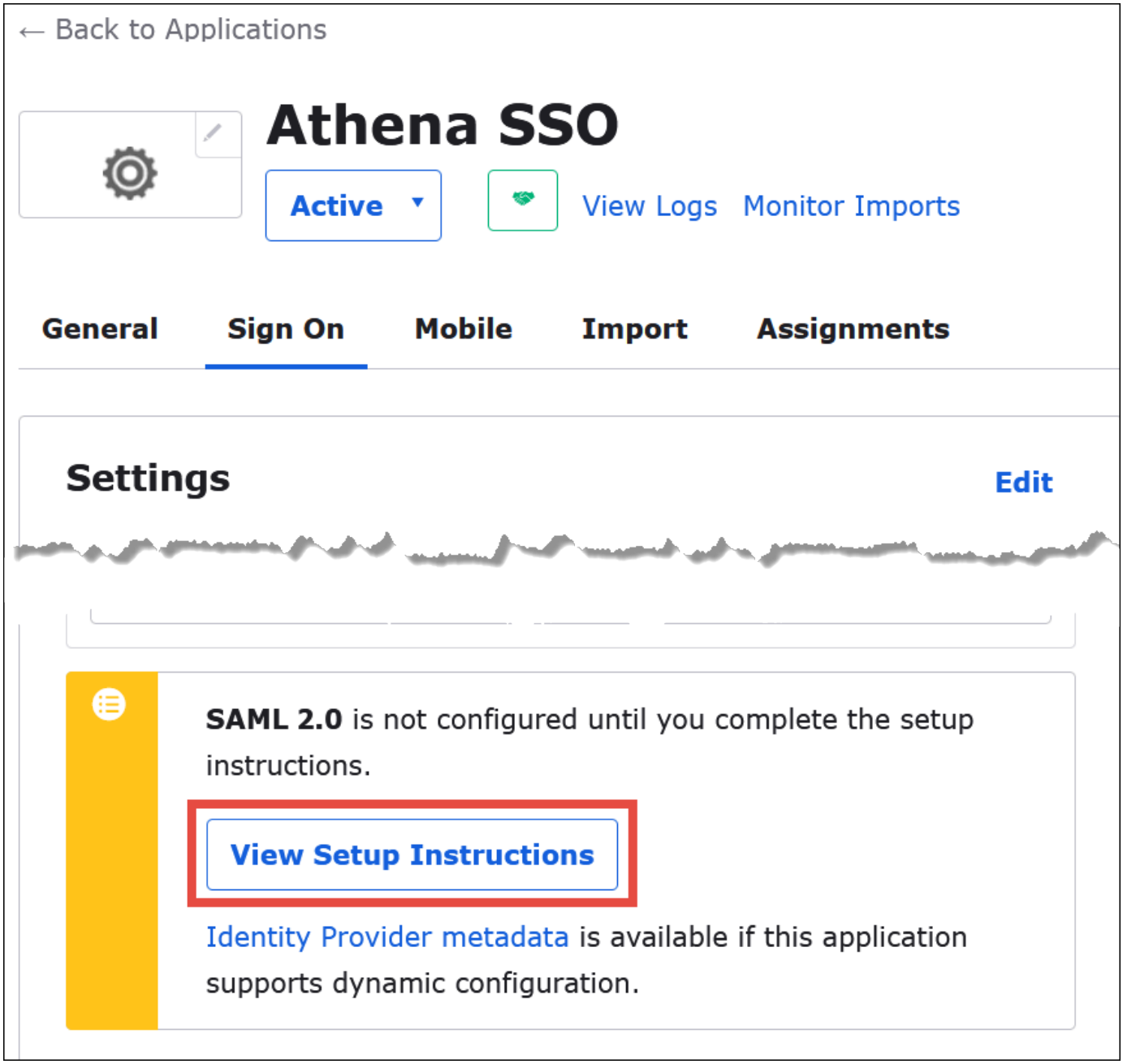
Sekarang aplikasi Anda telah dibuat, Anda dapat memperoleh URL login dan metadata lainnya dari dasbor Okta.

Untuk mendapatkan URL login dari dasbor Okta

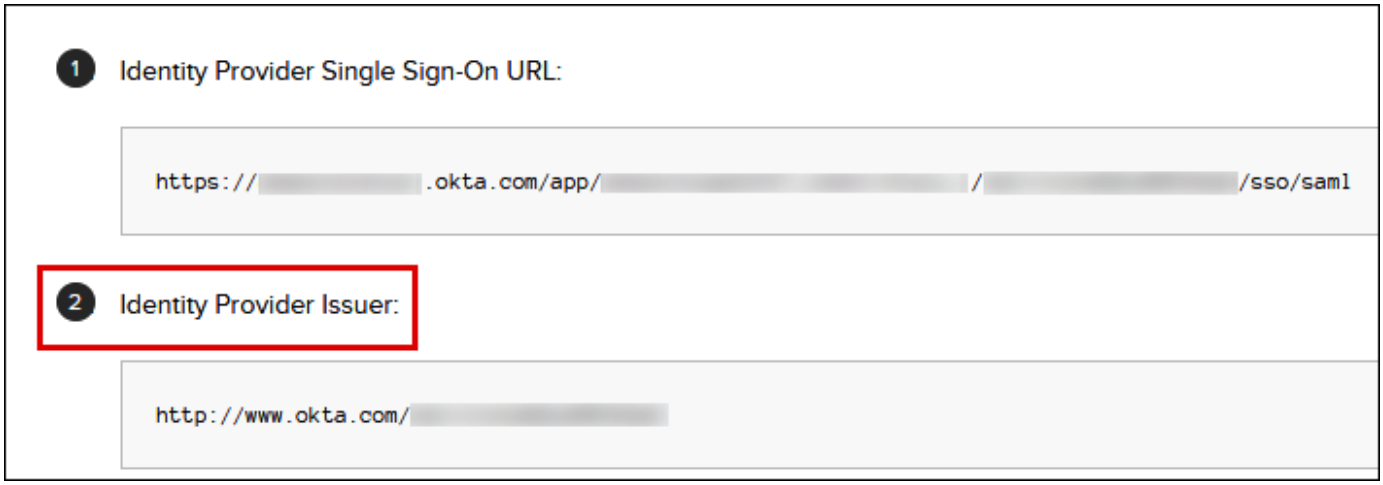
1. Di panel navigasi Okta, pilih Aplikasi, Aplikasi.
2. Pilih aplikasi yang ingin Anda temukan URL loginnya (misalnya, AthenAsso).
3. Pada halaman aplikasi Anda, pilih Sign On.



4. Pilih Lihat Petunjuk Pengaturan.



5. Pada halaman Cara Mengkonfigurasi SAFL 2.0 untuk Athena SSO, temukan URL untuk Identity Provider Issuer. Beberapa tempat di dasbor Okta menyebut URL ini sebagai ID penerbit SAMP.



6. Salin atau simpan nilai untuk URL Masuk Tunggal Penyedia Identitas.

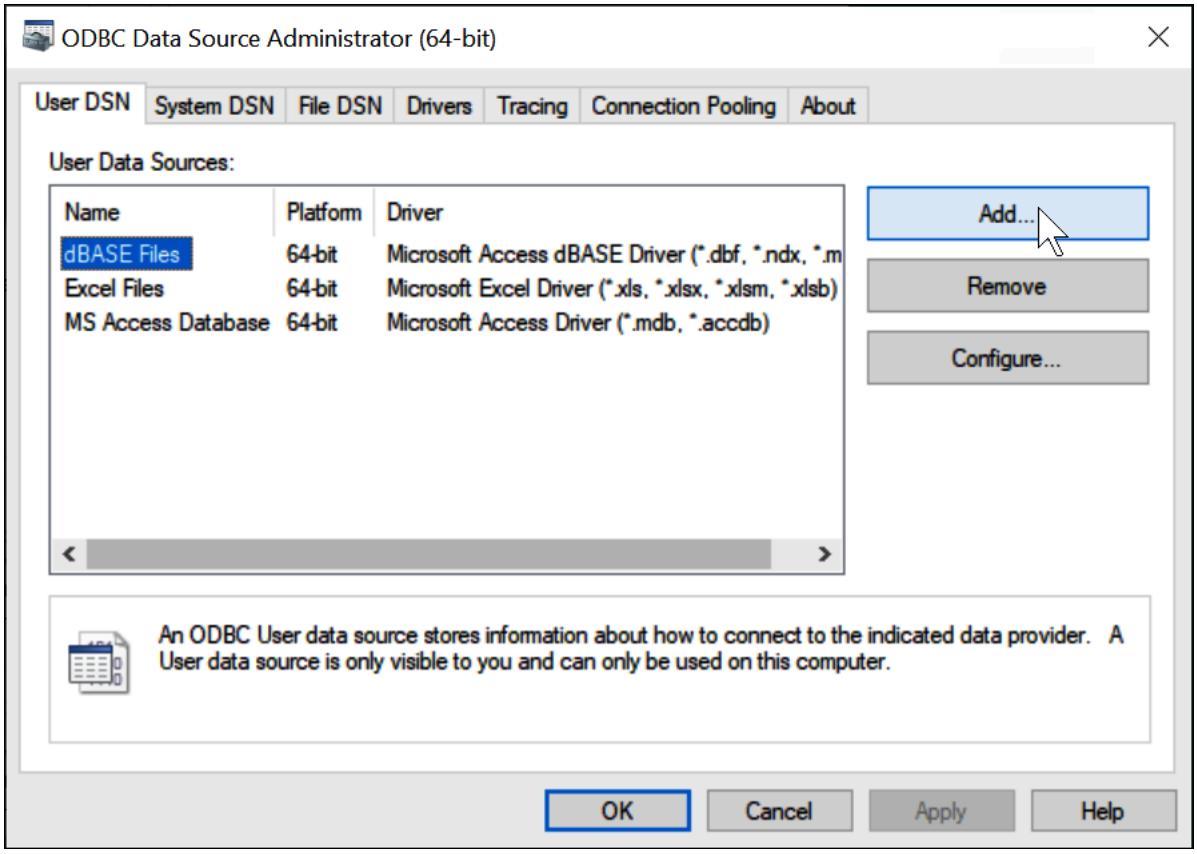
   Di bagian selanjutnya, saat Anda mengkonfigurasi koneksi ODBC, Anda akan memberikan nilai ini sebagai parameter koneksi URL Login untuk plugin SALL browser.

Mengkonfigurasi koneksi SAMP ODBC browser ke Athena

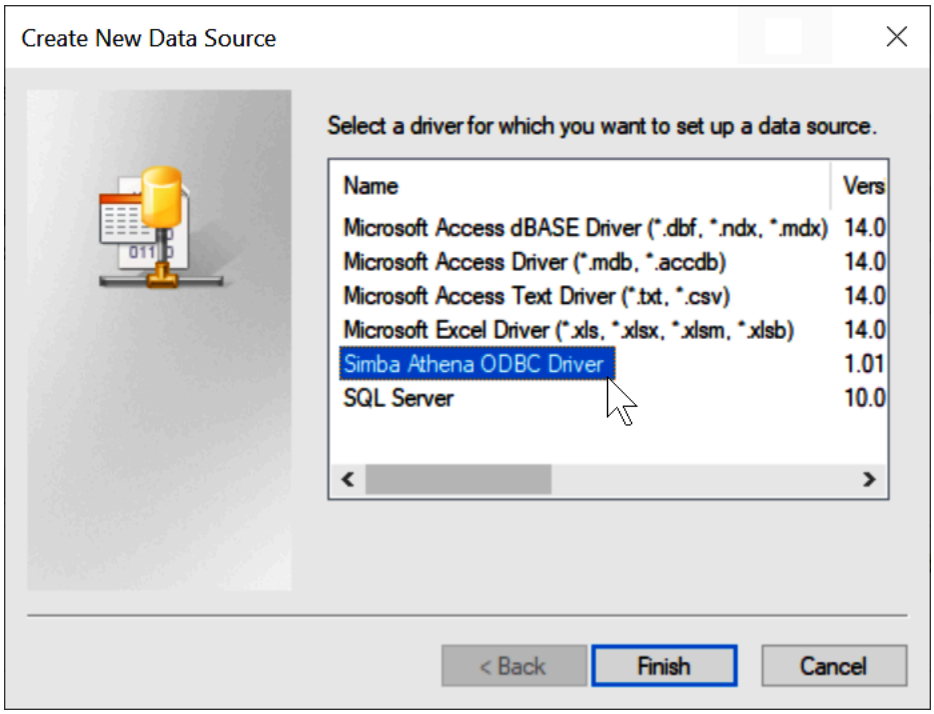
Sekarang Anda siap untuk mengkonfigurasi koneksi SALL browser ke Athena menggunakan program Sumber Data ODBC di Windows.

Untuk mengkonfigurasi koneksi SAMP ODBC browser ke Athena

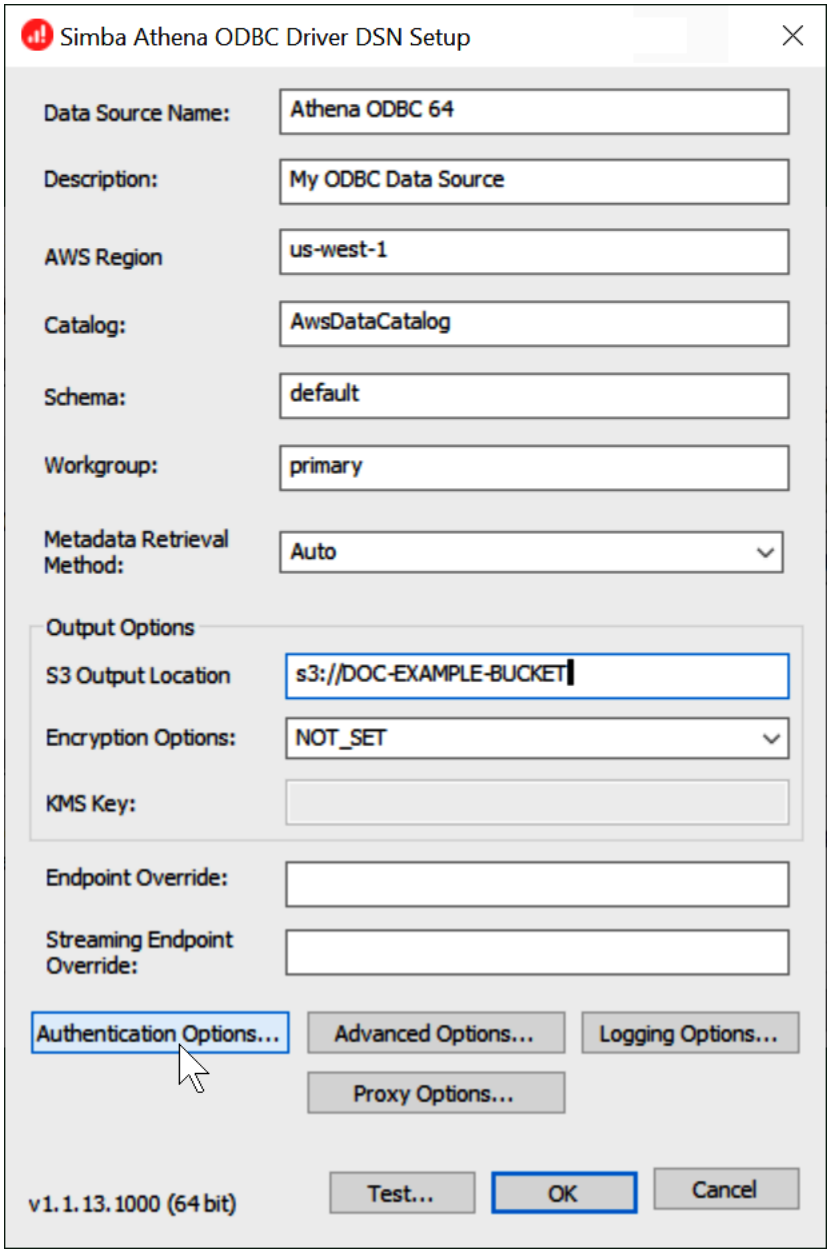
1. Di Windows, luncurkan program Sumber Data ODBC.
2. Dalam program Administrator Sumber Data ODBC, pilih Tambah.



3. Pilih Simba Athena ODBC Driver, lalu pilih Finish.

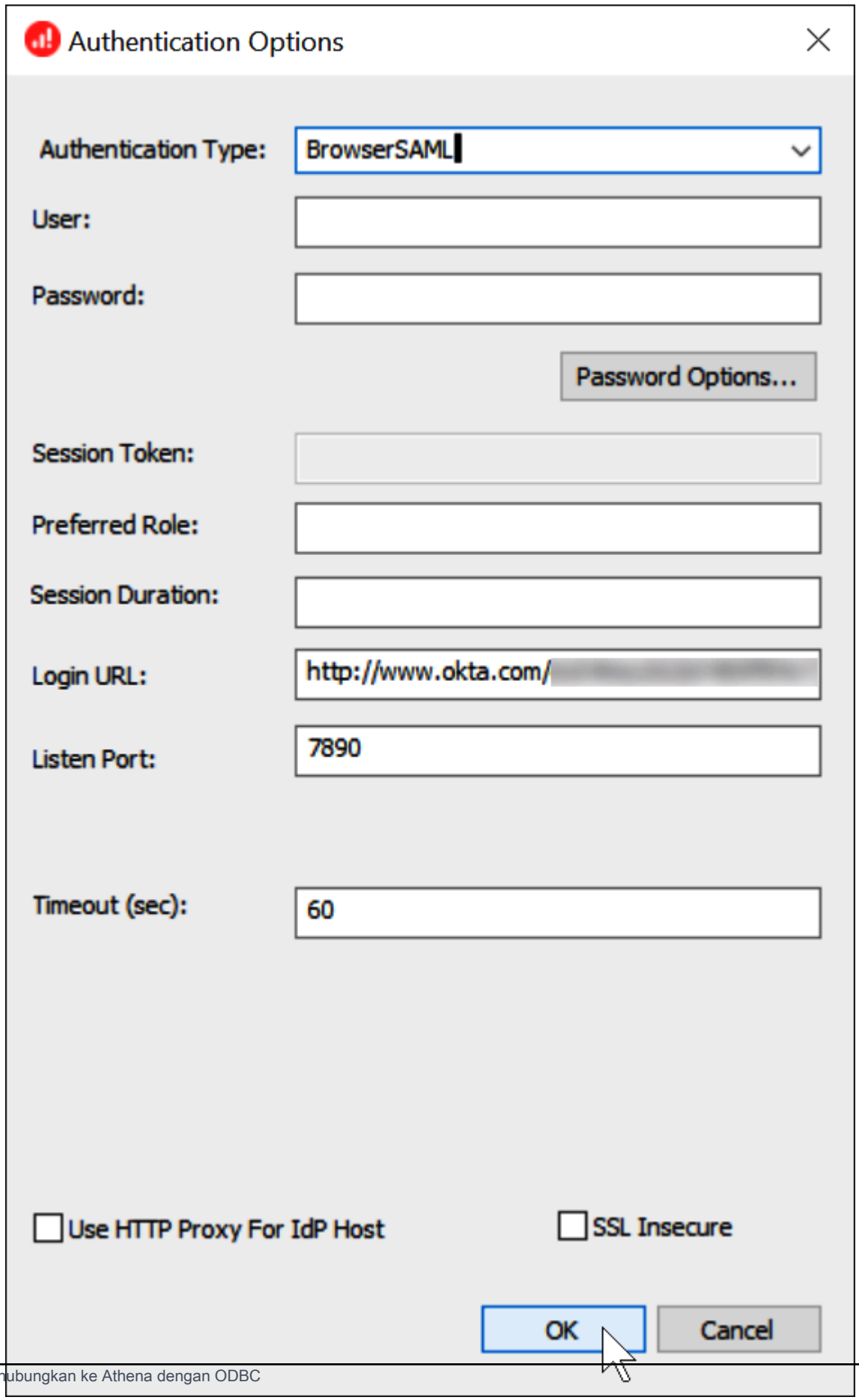


4. Di Simba Athena ODBC Driver DSN Setup dialog, masukkan nilai yang dijelaskan.



- Untuk Nama Sumber Data, masukkan nama untuk sumber data Anda (misalnya, Athena ODBC 64).
- Untuk Deskripsi, masukkan deskripsi untuk sumber data Anda.
- Untuk Wilayah AWS, masukkan Wilayah AWS yang Anda gunakan (misalnya,**`us-west-1`**).
- Untuk Lokasi Output S3, masukkan jalur Amazon S3 tempat Anda ingin output disimpan.

5. Pilih Opsi Otentikasi.
6. Dalam kotak dialog Opsi Otentikasi, pilih atau masukkan nilai berikut.


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

- Untuk Jenis Otentikasi, pilih BrowsersAML.
- Untuk URL Login, masukkan URL Single Sign-On Penyedia Identitas yang Anda peroleh dari dasbor Okta.
- Untuk Listen Port, masukkan 7890.
- Untuk Timeout (detik), masukkan nilai batas waktu koneksi dalam hitungan detik.

7. Pilih OK untuk menutup Opsi Otentikasi.
8. Pilih Uji untuk menguji koneksi, atau OK untuk menyelesaikan.

Menggunakan konektor Amazon Athena Power BI

Pada sistem operasi Windows, Anda dapat menggunakan konektor Microsoft Power BI untuk Amazon Athena untuk menganalisis data dari Amazon Athena di Microsoft Power BI Desktop. Untuk informasi tentang Power BI, lihat Microsoft power BI. Setelah mempublikasikan konten ke layanan Power BI, Anda dapat menggunakan gateway Power BI Juli 2021 atau yang lebih baru untuk menjaga konten tetap mutakhir melalui penyegaran sesuai permintaan atau terjadwal.

Prasyarat

Sebelum memulai, pastikan lingkungan Anda memenuhi persyaratan berikut. Pemandu Amazon Athena ODBC diperlukan.

- Akun AWS
- Izin untuk menggunakan Athena
- Pengemudi Amazon Athena ODBC
- Desktop Power BI

Kemampuan yang didukung

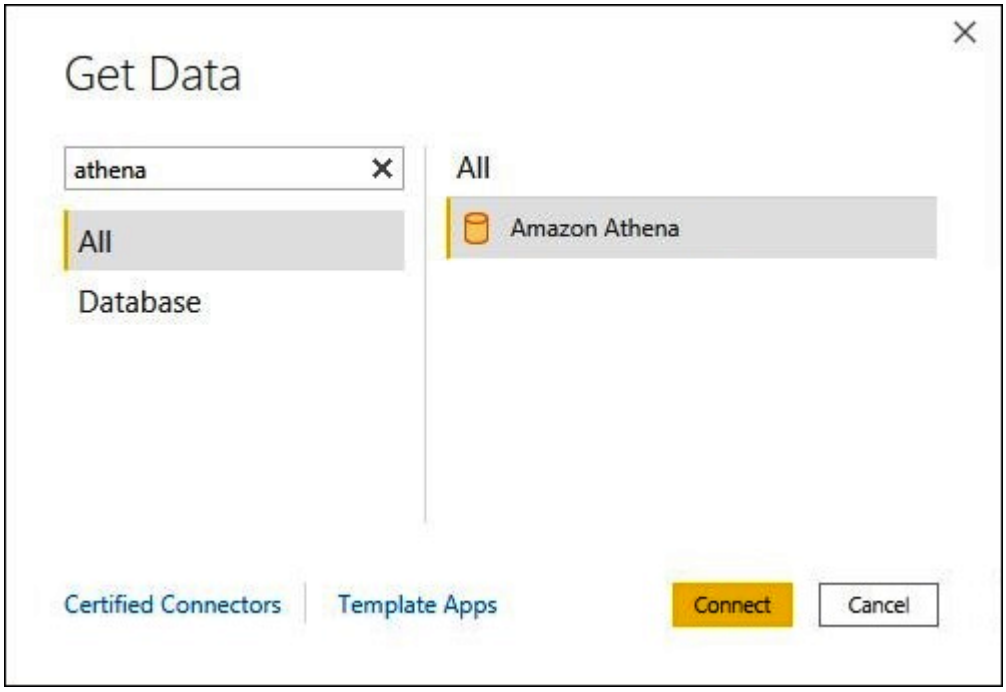
- Impor– Dipilih tabel dan kolom diimpor ke Power BI Desktop untuk mengkueri.
- DirectQuery— Tidak ada data yang diimpor atau disalin ke Power BI Desktop. Power BI Desktop kueri sumber data yang mendasari langsung.
- Gateway Power BI — Gateway data lokal di tempat Anda Akun AWS yang berfungsi seperti jembatan antara Layanan Microsoft Power BI dan Athena. Gateway diperlukan untuk melihat data Anda pada Microsoft Power BI Service.

Connect ke Amazon Athena

Untuk menghubungkan Power BI desktop ke data Amazon Athena, lakukan langkah berikut.

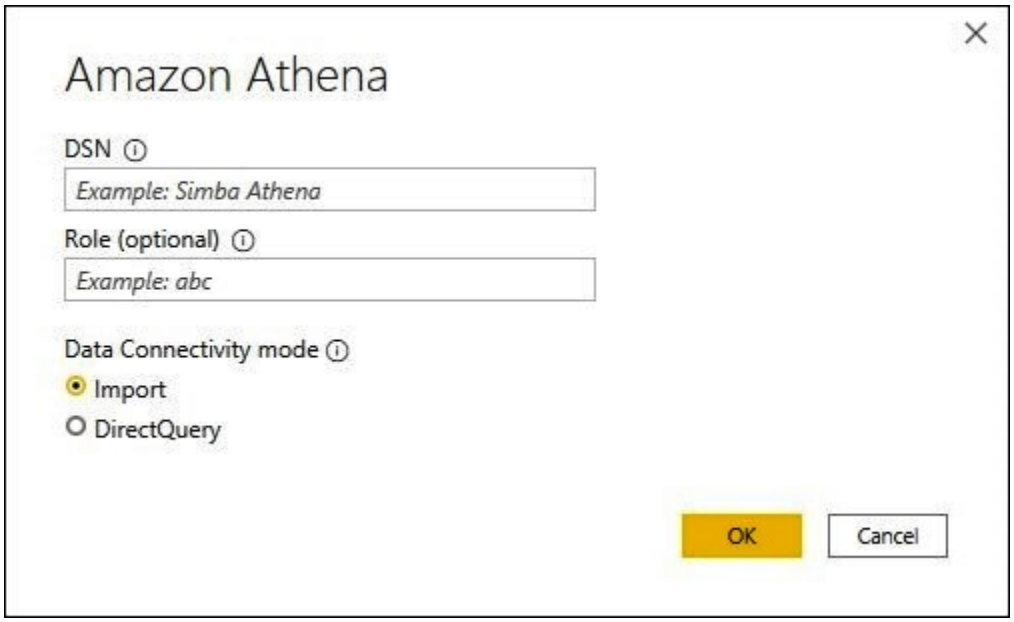
Untuk menyambung ke data Athena dari desktop power BI

1. Luncurkan Power BI Desktop.
2. Lakukan salah satu dari berikut:
   - PilihBerkas,Dapatkan Data
   - DariRumahpita, pilihDapatkan Data.
3. Dalam kotak pencarian, masukkan Athena.
4. PilihAmazon Athena, lalu pilihHubungkan.

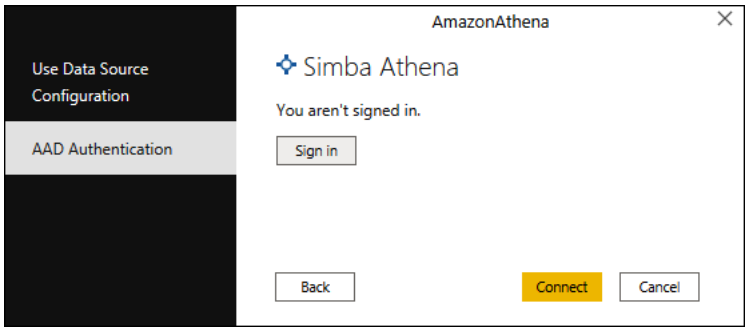


5. PadaAmazon Athena, masukkan informasi berikut.
   - UntukDSN, masukkan nama DSN ODBC yang ingin Anda gunakan. Untuk petunjuk tentang mengonfigurasi DSN Anda, lihatDriver ODBC.
   - UntukMode Konektivitas Data, pilih mode yang sesuai untuk kasus penggunaan Anda, mengikuti panduan umum berikut:
     - Untuk set data yang lebih kecil, pilihImpor. Saat menggunakan mode Impor, Power BI bekerja dengan Athena untuk mengimpor isi seluruh set data untuk digunakan dalam visualisasi Anda.

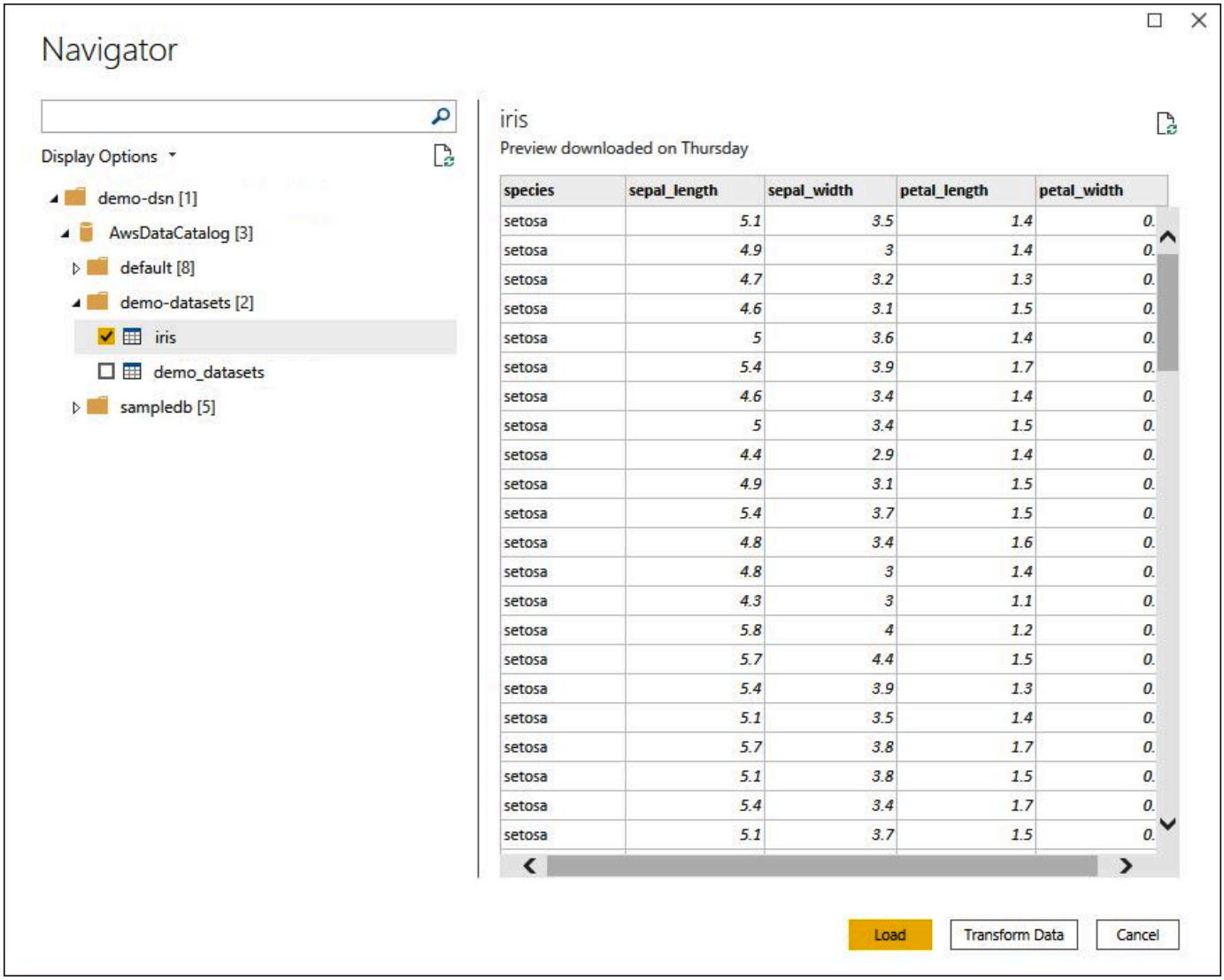
- Untuk kumpulan data yang lebih besar, pilih. DirectQuery Dalam DirectQuery mode, tidak ada data yang diunduh ke workstation Anda. Saat Anda membuat atau berinteraksi dengan visualisasi, Microsoft Power BI bekerja sama dengan Athena untuk secara dinamis meminta sumber data dasar sehingga Anda selalu melihat data saat ini. Untuk informasi selengkapnya DirectQuery, lihat Menggunakan DirectQuery desktop Power BI di dokumentasi Microsoft.



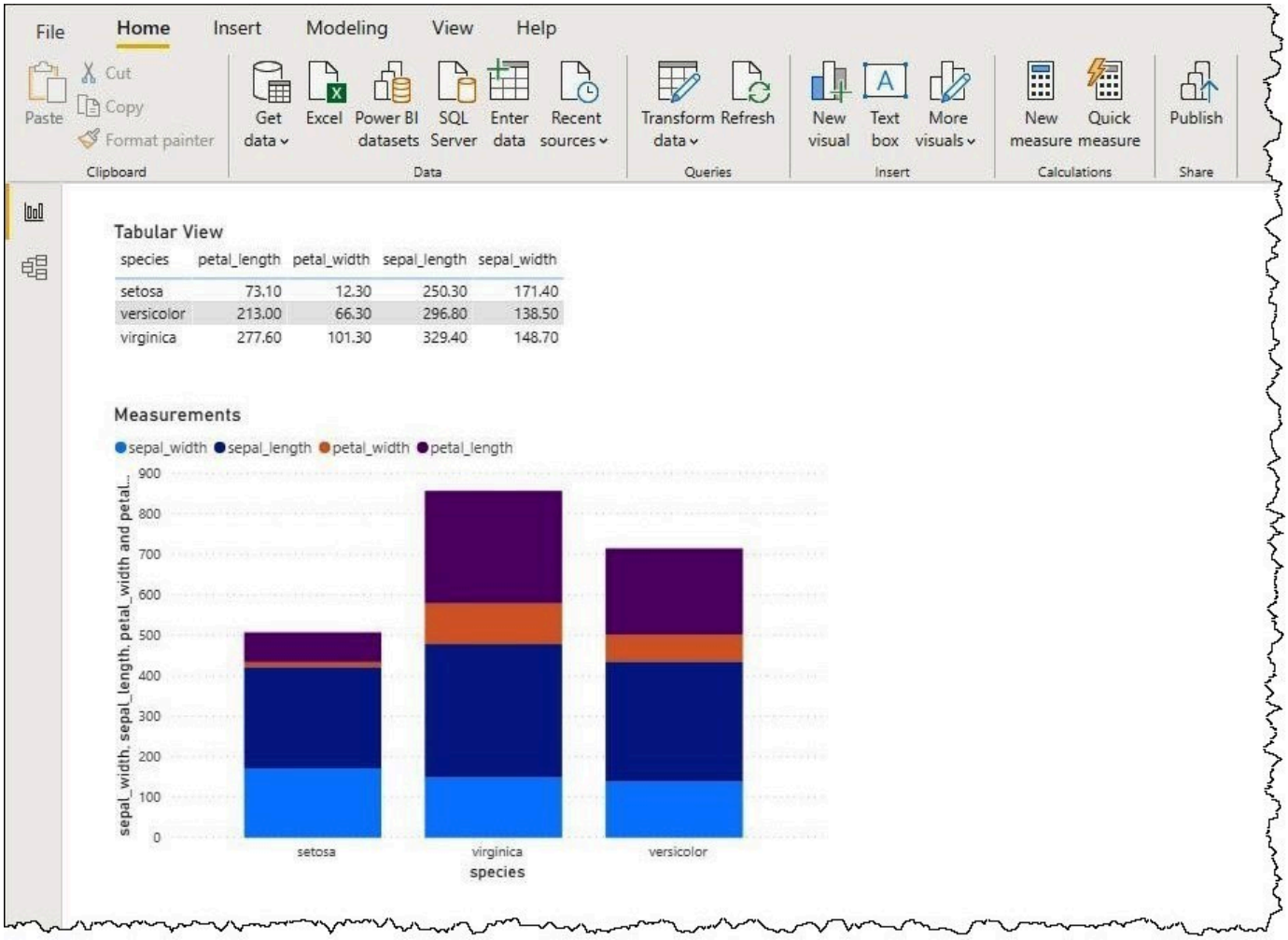
6. Pilih OK.
7. Pada prompt untuk mengonfigurasi otentikasi sumber data, pilih salah satuGunakan Konfigurasi Sumber DataatauAutentikasi AAD, lalu pilihHubungkan.



Katalog data Anda, basis data, dan tabel muncul dinavigatorkotak dialog.



8. DiOpsi tampilan, pilih kotak centang untuk set data yang ingin Anda gunakan.
9. Jika Anda ingin mengubah set data sebelum mengimpornya, buka bagian bawah kotak dialog dan pilihMengubah Data. Ini akan membuka Power Kueri Editor sehingga Anda dapat memfilter dan memperbaiki set data yang ingin Anda gunakan.
10. Pilih Muat . Setelah beban selesai, Anda dapat membuat visualisasi seperti yang di gambar berikut. Jika Anda memilih DirectQuerysebagai mode impor, Power BI mengeluarkan kueri ke Athena untuk visualisasi yang Anda minta.



Menyiapkan gateway on-premise

Anda dapat mempublikasikan dashboard dan set data ke layanan Power BI sehingga pengguna lain dapat berinteraksi dengan mereka melalui web, mobile, dan aplikasi tertanam. Untuk melihat data Anda di Microsoft Power BI Service, Anda menginstal gateway data on-premise Microsoft Power BI di Akun AWS. Gateway bekerja seperti jembatan antara Microsoft Power BI Service dan Athena.

Untuk mengunduh, menginstal, dan menguji gateway data on-premise

1. Kunjungi halaman unduhan gateway Microsoft power BI dan pilih mode pribadi atau mode standar. Mode pribadi berguna untuk menguji konektor Athena secara on-premise. Mode standar sesuai dalam pengaturan produksi multiuser.
2. Untuk menginstal gateway on-premise (baik mode pribadi maupun standar), lihatMenginstal gateway data di tempatdalam dokumentasi Microsoft.

3. Untuk menguji gateway, ikuti langkah-langkah diMenggunakan konektor data khusus dengan gateway data lokaldalam dokumentasi Microsoft.

Untuk informasi selengkapnya tentang gateway data on-premise, lihat sumber daya Microsoft berikut.

- Apa yang dimaksud dengan gateway data lokal?
- Panduan untuk menerapkan gateway data untuk power BI

Untuk contoh mengonfigurasi Power BI Gateway untuk digunakan dengan Athena, lihat AWS artikel Big Data Blog Membuat dasbor dengan cepat di Microsoft power BI menggunakan amazon Athena.

Membuat database dan tabel

Amazon Athena mendukung subset dari bahasa definisi data (DDL) pernyataan dan fungsi ANSI SQL dan operator untuk menentukan dan kueri tabel eksternal tempat data berada di Amazon Simple Storage Service.

Jika Anda membuat basis data dan tabel di Athena, Anda menggambarkan skema dan lokasi data, membuat data dalam tabel yang disiapkan untuk kueri waktu nyata.

Untuk meningkatkan kinerja kueri dan mengurangi biaya, kami sarankan Anda mempartisi data Anda dan menggunakan format kolom open source untuk penyimpanan di Amazon S3, seperti parket Apache atau ORC.

Topik

- Membuat database di Athena
- Membuat tabel di Athena
- Nama untuk tabel, database, dan kolom
- Kata kunci terpesan
- Lokasi tabel di Amazon S3
- Format penyimpanan kolumnar
- Mengonversi ke format kolumnar
- Partisi data di Athena
- Proyeksi partisi dengan Amazon Athena

Membuat database di Athena

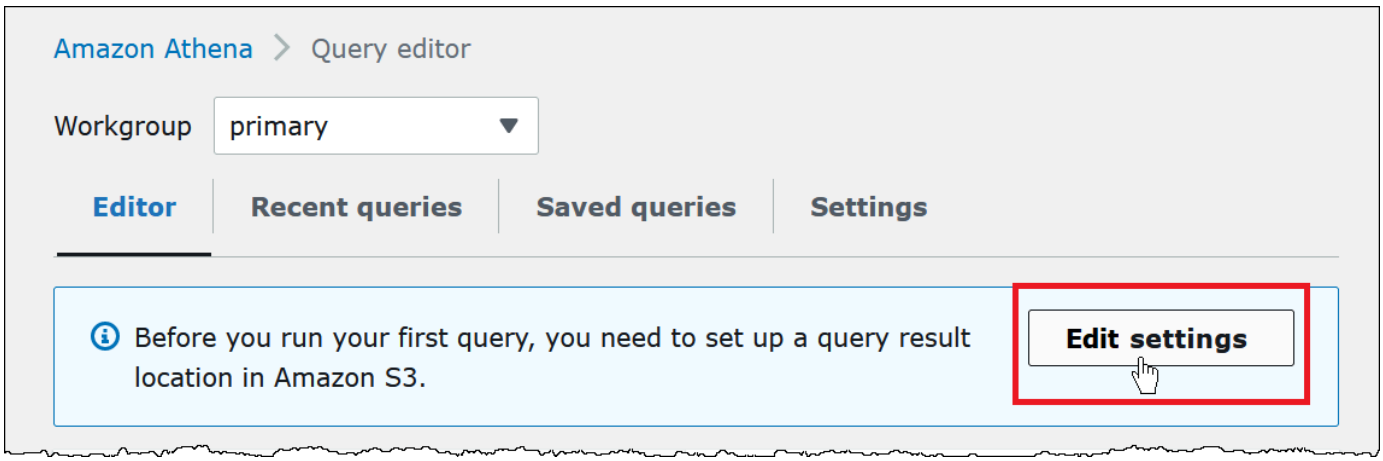
Sebuah basis data di Athena adalah pengelompokan logis untuk tabel yang Anda buat di dalamnya.

Prasyarat

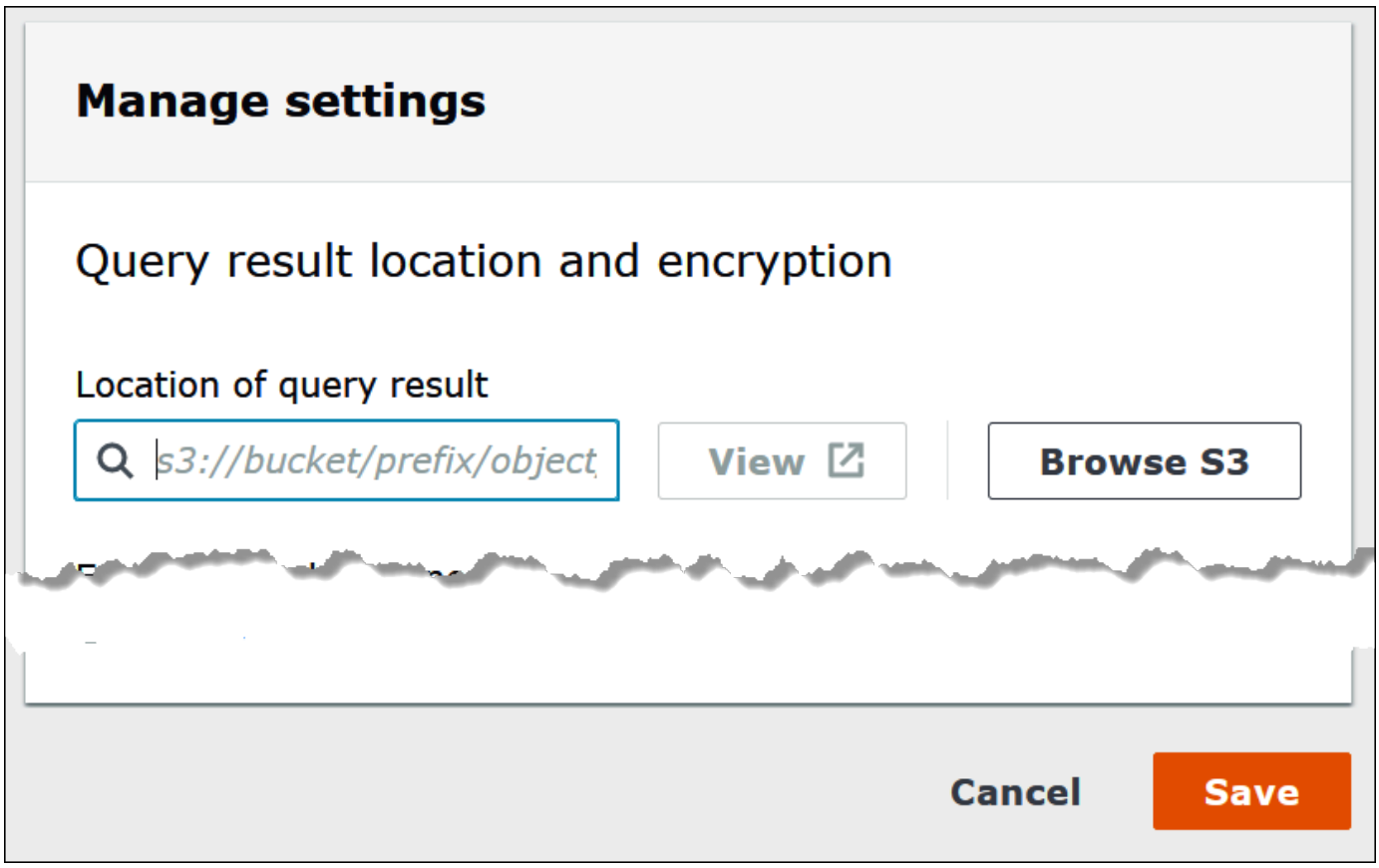
Jika Anda belum menyiapkan lokasi keluaran kueri di Amazon S3, lakukan langkah-langkah prasyarat berikut untuk melakukannya.

Untuk membuat lokasi keluaran kueri

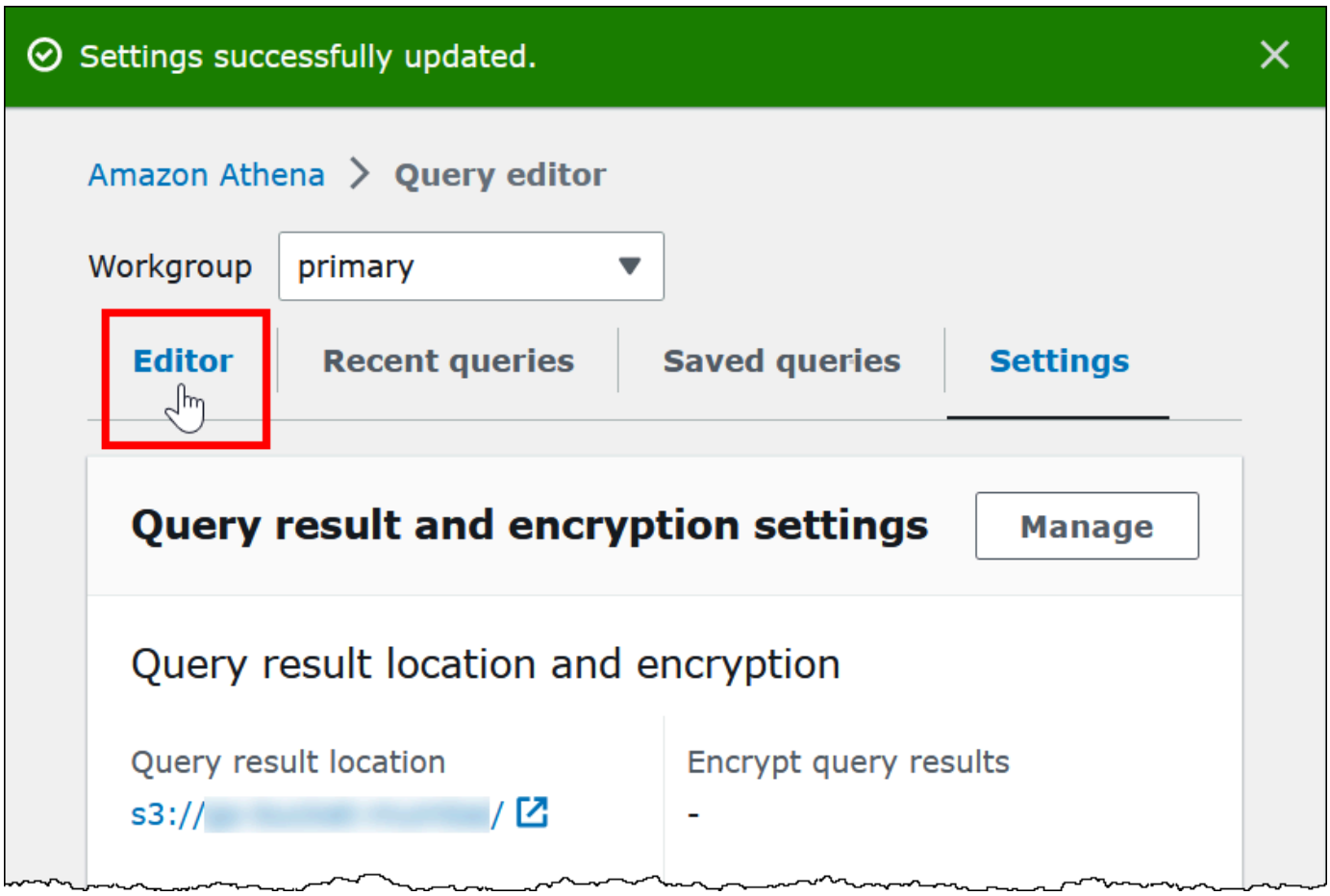
1. Menggunakan yang samaWilayah AWSdan akun yang Anda gunakan untuk Athena, ikuti langkah-langkahnya (misalnya, dengan menggunakan konsol Amazon S3)membuat bucket di Amazon S3untuk menahan hasil kueri Athena Anda. Anda akan mengkonfigurasi bucket ini menjadi lokasi keluaran kueri Anda.
2. Buka konsol Athena di https://console.aws.amazon.com/athena/.
3. Jika ini adalah pertama kalinya Anda mengunjungi konsol Athena dalam hal iniWilayah AWS, pilihJelajahi editor kueriuntuk membuka editor kueri. Kalau tidak, Athena terbuka di editor kueri.
4. PilihEdit Pengaturanuntuk menyiapkan lokasi hasil kueri di Amazon S3.



5. UntukMengelola pengaturan, lakukan salah satu dari berikut ini:
   - DalamLokasi hasil kuerikotak, masukkan jalur ke bucket yang Anda buat di Amazon S3 untuk hasil kueri Anda. Awalan jalan dengan`s3://`.
   - PilihJelajahi S3, pilih bucket Amazon S3 yang Anda buat untuk Wilayah Anda saat ini, lalu pilihPilih.



6. Pilih Save (Simpan).
7. PilihPenyuntinguntuk beralih ke editor kueri.



Membuat basis data

Setelah Anda menyiapkan lokasi hasil kueri, membuat database di editor kueri konsol Athena sangatlah mudah.

Untuk membuat database menggunakan editor kueri Athena

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. PadaPenyuntingtab, di editor query, masukkan bahasa definisi data Hive (DDL) perintah`CREATE DATABASE myDataBase`. Ganti`MyDatabase`dengan nama yang ingin Anda gunakan. Untuk pembatasan nama database, lihatNama untuk tabel, database, dan kolom.
3. PilihJalankanatau tekan**`Ctrl+ENTER`**.
4. Untuk membuat database Anda database saat ini, pilih dariBasis datamenu di sebelah kiri editor kueri.

Untuk informasi tentang mengendalikan izin ke database Athena, lihatAkses berbutir halus ke database dan tabel di AWS Glue Data Catalog.

Membuat tabel di Athena

Anda dapat menjalankan pernyataan DDL di konsol Athena, menggunakan JDBC atau driver ODBC, atau menggunakan formulir tabel Athena Create.

Saat Anda membuat skema tabel baru di Athena, Athena akan menyimpan skema tersebut dalam katalog data dan menggunakannya saat Anda menjalankan kueri.

Athena menggunakan pendekatan yang dikenal sebagai schema-on-read, yang berarti skema diproyeksikan ke data Anda pada saat Anda menjalankan kueri. Ini menghilangkan kebutuhan akan pemuatan data atau transformasi.

Athena tidak mengubah data Anda di Amazon S3.

Athena menggunakan Apache Hive untuk mendefinisikan tabel dan membuat basis data, yang pada dasarnya adalah namespace logis dari tabel.

Jika Anda membuat basis data dan tabel di Athena, Anda hanya menggambarkan skema dan lokasi tempat data tabel terletak di Amazon S3 untuk mengkueri read-time. Basis data dan tabel, oleh karena itu, memiliki arti yang sedikit berbeda dari yang arti mereka untuk sistem basis data relasional tradisional karena data tidak disimpan bersama dengan definisi skema untuk basis data dan tabel.

Saat Anda mengkueri, Anda mengkueri tabel menggunakan SQL standar dan data dibaca pada waktu itu. Anda dapat menemukan panduan untuk cara membuat basis data dan tabel menggunakan Dokumentasi Apache Hive, tetapi berikut memberikan bimbingan khusus untuk Athena.

Panjang string kueri maksimum adalah 256 KB.

Hive mendukung beberapa format data melalui penggunaan pustaka serializer-deserializer (). SerDe Anda juga dapat menentukan skema kompleks menggunakan ekspresi reguler. Untuk daftar SerDe pustaka yang didukung, lihatFormat yang didukung SerDes dan data.

Pertimbangan dan batasan

Berikut adalah beberapa batasan dan pertimbangan penting untuk tabel di Athena.

Persyaratan untuk tabel di Athena dan data di Amazon S3

Saat Anda membuat tabel, Anda menentukan lokasi bucket Amazon S3 untuk data dasar menggunakan klausa `LOCATION`. Pertimbangkan hal berikut:

- Athena hanya dapat mengkueri versi terbaru dari data pada bucket Amazon S3 berversi, dan tidak dapat mengkueri versi sebelumnya dari data.
- Anda harus memiliki izin yang sesuai untuk bekerja dengan data di lokasi Amazon S3. Untuk informasi selengkapnya, lihat Akses ke Amazon S3 dari Athena.
- Athena mendukung kueri objek yang disimpan dengan beberapa kelas penyimpanan dalam bucket yang sama ditentukan oleh klausa `LOCATION`. Misalnya, Anda dapat mengkueri data dalam objek yang disimpan di kelas Penyimpanan yang berbeda (Standard, Standar-IA dan Intelligent-Tiering) di Amazon S3.
- Athena mendukung ember Requester Pays. Untuk informasi cara mengaktifkan Requester Pays untuk bucket dengan data sumber yang ingin Anda kueri di Athena, lihat. Buat grup kerja
- Athena tidak mendukung kueri data dalam pengambilan fleksibel S3 Glacier atau kelas penyimpanan S3 Glacier Deep Archive. Objek di kelas penyimpanan S3 Glacier Flexible Retrieval dan S3 Glacier Deep Archive diabaikan. Sebagai alternatif, Anda dapat menggunakan kelas penyimpanan Amazon S3 Glacier Instant Retrieval, yang dapat ditanyakan oleh Athena. Untuk informasi selengkapnya, lihat kelas penyimpanan pengambilan instan Amazon S3 Glacier.

  Untuk informasi tentang kelas penyimpanan, lihat Kelas penyimpanan, Mengubah kelas penyimpanan objek di amazon S3, Transisi ke kelas penyimpanan GLACIER (arsip objek), dan Bucket Requester Pays di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.
- Jika Anda mengeluarkan kueri terhadap bucket Amazon S3 dengan sejumlah besar objek dan data tidak dipartisi, kueri tersebut dapat memengaruhi batas kecepatan Dapatkan permintaan di Amazon S3 dan menyebabkan pengecualian Amazon S3. Untuk mencegah kesalahan, partisi data Anda. Selain itu, pertimbangkan untuk menyetel tingkat permintaan Amazon S3 Anda. Untuk informasi selengkapnya, lihat Pertimbangan tingkat permintaan dan kinerja.
- Jika Anda menggunakan operasi AWS Glue CreateTableAPI atau AWS CloudFormation `AWS::Glue::Table`template untuk membuat tabel untuk digunakan di Athena tanpa menentukan `TableType` properti dan kemudian menjalankan kueri DDL seperti `SHOW CREATE TABLE` atau`MSCK REPAIR TABLE`, Anda dapat menerima pesan kesalahan GAGAL: NullPointerException Nama adalah null.

Untuk mengatasi kesalahan, tentukan nilai TableInput`TableType`atribut sebagai bagian dari panggilan AWS Glue `CreateTable` API atau AWS CloudFormation templat. Nilai yang mungkin untuk `TableType` include `EXTERNAL_TABLE` atau`VIRTUAL_VIEW`.

Persyaratan ini hanya berlaku ketika Anda membuat tabel menggunakan operasi AWS Glue `CreateTable` API atau `AWS::Glue::Table` template. Jika Anda membuat tabel untuk Athena menggunakan pernyataan DDL atau AWS Glue crawler, `TableType` properti didefinisikan untuk Anda secara otomatis.

Fungsi yang didukung

Fungsi yang didukung dalam kueri Athena sesuai dengan yang ada di Trino dan Presto. Untuk informasi tentang fungsi individual, lihat bagian fungsi dan operator di dokumentasi Trino atau Presto.

Transformasi data transaksional tidak didukung

Athena tidak mendukung operasi berbasis transaksi (seperti yang ditemukan di Hive atau Presto) pada data tabel. Untuk daftar lengkap kata kunci yang tidak didukung, lihat: DDL Tidak Didukung.

Operasi yang mengubah status tabel adalah ACID

Saat Anda membuat, memperbarui, atau menghapus tabel, operasi tersebut dijamin mematuhi ASID. Sebagai contoh, jika beberapa pengguna atau klien mencoba untuk membuat atau mengubah tabel yang ada pada saat yang sama, hanya satu akan berhasil.

Tabel adalah EKSTERNAL

Kecuali saat membuat tabel Iceberg, selalu gunakan kata kunci. `EXTERNAL` Jika Anda menggunakan `CREATE TABLE` tanpa `EXTERNAL` kata kunci untuk tabel non-Iceberg, Athena mengeluarkan kesalahan. Saat Anda menjatuhkan tabel di Athena, hanya metadata tabel dihapus; data tetap di Amazon S3.

Membuat tabel menggunakan AWS Glue atau konsol Athena

Anda dapat membuat tabel di Athena dengan menggunakan AWS Glue, menambahkan formulir tabel, atau dengan menjalankan pernyataan DDL di editor kueri Athena.

Untuk membuat tabel menggunakan AWS Glue crawler

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.

2. Di editor kueri, di samping Tabel dan tampilan, pilih Buat, lalu pilih AWS Glue crawler.
3. Ikuti langkah-langkah di halaman Add crawler AWS Glue konsol untuk menambahkan crawler.

   Untuk informasi selengkapnya, lihat Menggunakan AWS Glue crawler.

Untuk membuat tabel menggunakan Athena membuat formulir tabel

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Di editor kueri, di samping Tabel dan tampilan, pilih Buat, lalu pilih data bucket S3.
3. Dalam formulir data bucket Create Table From S3, masukkan informasi untuk membuat tabel, lalu pilih Create table. Untuk informasi lebih lanjut tentang bidang dalam formulir, lihatMenambahkan tabel menggunakan formulir.

Untuk membuat tabel menggunakan Hive DDL

1. Dari menu Basis Data, pilih basis data yang ingin Anda buatkan tabel. Jika Anda tidak menentukan database dalam `CREATE TABLE` pernyataan Anda, tabel dibuat dalam database yang saat ini dipilih di editor kueri.
2. Masukkan pernyataan seperti berikut ini di editor kueri, lalu pilih Jalankan, atau tekan**`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/';
```

Menampilkan informasi tabel

Setelah Anda membuat tabel di Athena, namanya ditampilkan dalam daftar Tabel di sebelah kiri. Untuk menampilkan informasi tentang tabel dan mengelolanya, pilih tiga titik vertikal di sebelah nama tabel di konsol Athena.

- Tabel pratinjau - Menampilkan 10 baris pertama dari semua kolom dengan menjalankan `SELECT * FROM "database_name"."table_name" LIMIT 10` pernyataan di editor kueri Athena.
- Menghasilkan tabel DDL - Menghasilkan pernyataan DDL yang dapat Anda gunakan untuk membuat ulang tabel dengan menjalankan `SHOW CREATE TABLE` *pernyataan table_name* di editor kueri Athena.
- Memuat partisi - Menjalankan `MSCK REPAIR TABLE` *table_name* pernyataan di editor kueri Athena. Opsi ini hanya tersedia jika tabel memiliki partisi.
- Sisipkan ke editor - Menyisipkan nama tabel ke editor kueri di lokasi pengeditan saat ini.
- Hapus tabel - Menampilkan kotak dialog konfirmasi menanyakan apakah Anda ingin menghapus tabel. Jika Anda setuju, jalankan `DROP TABLE` *table_name* pernyataan di editor kueri Athena.
- Properti tabel - Menampilkan nama tabel, nama basis data, waktu dibuat, dan apakah tabel memiliki data terenkripsi.

Nama untuk tabel, database, dan kolom

Gunakan tips ini untuk penamaan objek database di Athena.

Persyaratan nama database, tabel, dan kolom

- Karakter yang dapat diterima untuk nama database, nama tabel, dan nama kolom AWS Glue harus berupa string UTF-8. Panjang string tidak boleh kurang dari 1 atau lebih dari 255 byte. Melebihi batas ini menghasilkan kesalahan seperti Nilai pada 'nama' gagal memenuhi kendala: Anggota harus memiliki panjang kurang dari atau sama dengan 255. Karakter yang dapat digunakan termasuk spasi dan ditentukan oleh pola string baris tunggal berikut:

```
[\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\t]*
```

- Saat ini, pola AWS Glue regex memungkinkan spasi utama ditambahkan ke awal nama. Karena ruang terdepan ini sulit dideteksi dan dapat menyebabkan masalah kegunaan setelah pembuatan, hindari membuat nama objek yang memiliki spasi terdepan.
- Jika Anda menggunakan AWS::Glue::Database AWS CloudFormation template untuk membuat AWS Glue database dan tidak menentukan nama database, AWS Glue secara otomatis menghasilkan nama database dalam format *resource_name–random_string* yang tidak kompatibel dengan Athena.
- Anda dapat menggunakan Pengelola AWS Glue Katalog untuk mengganti nama kolom, tetapi bukan nama tabel atau nama database. Untuk mengatasi batasan ini, Anda harus menggunakan definisi database lama untuk membuat database dengan nama baru. Kemudian Anda menggunakan definisi tabel dari database lama untuk membuat ulang tabel dalam database baru. Untuk melakukan ini, Anda dapat menggunakan AWS CLI atau AWS Glue SDK. Untuk langkah, lihat Menggunakan AWS CLI untuk membuat ulang AWS Glue database dan tabelnya.

Gunakan huruf kecil untuk nama tabel dan nama kolom tabel di Athena

Athena menerima kasus campuran dalam kueri DDL dan DHTML, tetapi huruf kecil nama saat mengeksekusi kueri. Untuk alasan ini, hindari menggunakan kotak campuran untuk nama tabel atau kolom, dan jangan mengandalkan casing saja di Athena untuk membedakan nama-nama tersebut. Misalnya, jika Anda menggunakan pernyataan DDL untuk membuat kolom bernama`Castle`, kolom yang dibuat akan diturunkan ke. `castle` Jika Anda kemudian menentukan nama kolom dalam kueri DHTML sebagai `Castle` atau`CASTLE`, Athena akan huruf kecil nama bagi Anda untuk menjalankan kueri, tetapi menampilkan judul kolom menggunakan casing yang Anda pilih dalam kueri.

Nama database, tabel, dan kolom harus kurang dari atau sama dengan 255 karakter.

Nama yang dimulai dengan garis bawah

Saat membuat tabel, gunakan backticks untuk melampirkan nama tabel, tampilan, atau kolom yang dimulai dengan garis bawah. Sebagai contoh:

```
CREATE EXTERNAL TABLE IF NOT EXISTS `_myunderscoretable`(
  `_id` string, `_index` string)
LOCATION 's3://DOC-EXAMPLE-BUCKET/'
```

Nama tabel, tampilan, atau kolom yang dimulai dengan angka

Saat menjalankan`SELECT`,`CTAS`, atau `VIEW` kueri, letakkan tanda kutip di sekitar pengidentifikasi seperti nama tabel, tampilan, atau kolom yang dimulai dengan digit. Sebagai contoh:

```
CREATE OR REPLACE VIEW "123view" AS
SELECT "123columnone", "123columntwo"
FROM "234table"
```

Nama kolom dan tipe kompleks

Untuk tipe kompleks, hanya karakter alfanumerik, garis bawah (_), dan periode (`.`) yang diizinkan dalam nama kolom. Untuk membuat tabel dan pemetaan untuk kunci yang memiliki karakter terbatas, Anda dapat menggunakan pernyataan DDL kustom. Untuk informasi selengkapnya, lihat artikel Membuat tabel di Amazon Athena dari JSON bersarang dan pemetaan menggunakan SerDe JSON di Blog Big Data.AWS

Kata yang dicadangkan

Kata-kata pendiam tertentu di Athena harus lolos. Untuk melepaskan kata kunci cadangan dalam pernyataan DDL, sertakan kata kunci tersebut dalam backtick (`). Untuk melepaskan kata kunci cadangan dalam pernyataan `SELECT` SQL dan dalam kueri pada tampilan, sertakan kata kunci dalam tanda kutip ganda (“).

Untuk informasi selengkapnya, lihat Kata kunci terpesan.

Sumber daya tambahan

Untuk database lengkap dan sintaks pembuatan tabel, lihat halaman berikut.

- CREATE DATABASE
- CREATE TABLE

Untuk informasi selengkapnya tentang database dan tabel AWS Glue, lihat Database dan Tabel di Panduan AWS Glue Pengembang.

Kata kunci terpesan

Saat Anda menjalankan kueri di Athena yang mencakup kata kunci cadangan, Anda harus melepaskannya dengan menyertakan mereka dalam karakter khusus. Gunakan daftar dalam topik ini untuk memeriksa kata kunci mana yang dicadangkan di Athena.

Untuk melepaskan kata kunci cadangan dalam pernyataan DDL, sertakan kata kunci tersebut dalam backtick (`). Untuk melepaskan kata kunci cadangan dalam pernyataan SELECT SQL dan dalam kueri pada tampilan, sertakan kata kunci dalam tanda kutip ganda (“).

- Daftar kata kunci yang dicadangkan dalam pernyataan DDL
- Daftar kata kunci yang dicadangkan dalam pernyataan SQL SELECT
- Contoh kueri dengan kata-kata yang dicadangkan

Daftar kata kunci yang dicadangkan dalam pernyataan DDL

Athena menggunakan daftar kata kunci cadangan dalam pernyataan DDL-nya. Jika Anda menggunakan kata kunci cadangan tanpa melepaskannya, Athena akan mengeluarkan kesalahan. Untuk melepaskan kata kunci, sertakan kata kunci dalam backtick (`).

Anda tidak dapat menggunakan kata kunci cadangan DDL sebagai nama pengidentifikasi dalam pernyataan DDL tanpa menyertakannya dalam backtick (`).

```
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
```

Daftar kata kunci yang dicadangkan dalam pernyataan SQL SELECT

Athena akan menggunakan daftar kata kunci cadangan berikut dalam pernyataan `SELECT` SQL dan dalam kueri di tampilan.

Jika Anda menggunakan kata kunci ini sebagai pengidentifikasi, Anda harus menyertakan mereka dalam tanda kutip ganda (“) dalam pernyataan kueri Anda.

```
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
```

Contoh kueri dengan kata-kata yang dicadangkan

Kueri dalam contoh berikut menggunakan backtick (`) untuk melepaskan partisi dan tanggal kata kunci cadangan terkait DDL yang digunakan untuk nama tabel dan salah satu nama kolom:

```
CREATE EXTERNAL TABLE `partition` (
`date` INT,
col2 STRING
)
PARTITIONED BY (year STRING)
STORED AS TEXTFILE
LOCATION 's3://DOC-EXAMPLE-BUCKET/test_examples/';
```

Contoh kueri berikut termasuk nama kolom yang berisi kata kunci cadangan terkait DDL di `ALTER TABLE ADD PARTITION` dan `ALTER TABLE DROP PARTITION`. Kata kunci cadangan DDL disertakan dalam backtick (`):

```
ALTER TABLE test_table
ADD PARTITION (`date` = '2018-05-14')
```

```
ALTER TABLE test_table
DROP PARTITION (`partition` = 'test_partition_value')
```

Contoh kueri berikut mencakup kata kunci cadangan (akhir) sebagai pengidentifikasi dalam `SELECT`. Kata kunci dilepaskan dalam tanda kutip ganda:

```
SELECT *
FROM TestTable
WHERE "end" != nil;
```

Contoh kueri berikut mencakup kata kunci cadangan (pertama) dalam `SELECT`. Kata kunci dilepaskan dalam tanda kutip ganda:

```
SELECT "itemId"."first"
FROM testTable
LIMIT 10;
```

Lokasi tabel di Amazon S3

Saat Anda menjalankan `CREATE TABLE` kueri di Athena, Athena mendaftarkan tabel Anda dengan Katalog AWS Glue Data, tempat Athena menyimpan metadata Anda.

Untuk menentukan jalur ke data Anda di Amazon S3, gunakan `LOCATION` properti, seperti yang ditunjukkan pada contoh berikut:

```
CREATE EXTERNAL TABLE `test_table`(
...
)
ROW FORMAT ...
STORED AS INPUTFORMAT ...
OUTPUTFORMAT ...
LOCATION s3://DOC-EXAMPLE-BUCKET/folder/
```

- Untuk informasi tentang penamaan bucket, lihat Pembatasan dan batasan Bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

- Untuk informasi tentang menggunakan folder di Amazon S3, lihat Menggunakan folder di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

`LOCATION`Di Amazon S3 menentukan semua file yang mewakili tabel Anda.

> **Important**
>
> Athena membaca semua data yang disimpan di folder Amazon S3 yang Anda tentukan. Jika Anda memiliki data yang tidak ingin Athena baca, jangan simpan data itu di folder Amazon S3 yang sama dengan data yang Anda ingin Athena baca. Jika Anda memanfaatkan partisi, untuk memastikan Athena memindai data dalam partisi, filter Anda `WHERE` harus menyertakan partisi. Untuk informasi selengkapnya, lihat Lokasi tabel dan partisi.

Saat Anda menentukan `LOCATION` dalam `CREATE TABLE` pernyataan, gunakan pedoman berikut:

- Gunakan garis miring.
- Anda dapat menggunakan jalur ke folder Amazon S3 atau alias jalur akses Amazon S3. Untuk informasi tentang alias jalur akses Amazon S3, lihat Menggunakan alias gaya ember untuk titik akses Anda di Panduan Pengguna Amazon S3.

Gunakan:

```
s3://DOC-EXAMPLE-BUCKET/folder/
```

```
s3://DOC-EXAMPLE-BUCKET-metadata-s3alias/folder/
```

Jangan gunakan salah satu item berikut `LOCATION` untuk menentukan data Anda.

- Jangan gunakan nama file, garis bawah, wildcard, atau pola glob untuk menentukan lokasi file.
- Jangan menambahkan notasi HTTP lengkap, seperti `s3.amazon.com` ke jalur bucket Amazon S3.
- Jangan gunakan folder kosong seperti `//` di jalur, sebagai berikut:`S3://DOC-EXAMPLE-BUCKET/folder//folder/`. Meskipun ini adalah jalur Amazon S3 yang valid, Athena tidak mengizinkannya dan mengubahnya menjadi`s3://DOC-EXAMPLE-BUCKET/folder/folder/`, menghapus ekstra. `/`

  Jangan gunakan:

```
s3://DOC-EXAMPLE-BUCKET
s3://DOC-EXAMPLE-BUCKET/*
s3://DOC-EXAMPLE-BUCKET/mySpecialFile.dat
s3://DOC-EXAMPLE-BUCKET/prefix/filename.csv
s3://DOC-EXAMPLE-BUCKET.s3.amazon.com
S3://DOC-EXAMPLE-BUCKET/prefix//prefix/
arn:aws:s3:::DOC-EXAMPLE-BUCKET/prefix
s3://arn:aws:s3:<region>:<account_id>:accesspoint/<accesspointname>
https://<accesspointname>-<number>.s3-accesspoint.<region>.amazonaws.com
```

Lokasi tabel dan partisi

Data sumber Anda dapat dikelompokkan ke dalam folder Amazon S3 yang disebut partisi berdasarkan sekumpulan kolom. Misalnya, kolom-kolom ini dapat mewakili tahun, bulan, dan hari rekaman tertentu dibuat.

Saat Anda membuat tabel, Anda dapat memilih untuk membuatnya dipartisi. Ketika Athena menjalankan query SQL terhadap tabel non-partisi, ia menggunakan `LOCATION` properti dari definisi tabel sebagai jalur dasar untuk daftar dan kemudian memindai semua file yang tersedia. Namun, sebelum tabel yang dipartisi dapat ditanyakan, Anda harus memperbarui Katalog AWS Glue Data dengan informasi partisi. Informasi ini mewakili skema file dalam partisi tertentu dan file di Amazon S3 untuk partisi. `LOCATION`

- Untuk mempelajari cara AWS Glue crawler menambahkan partisi, lihat Bagaimana crawler menentukan kapan harus membuat partisi? di Panduan AWS Glue Pengembang.
- Untuk mempelajari cara mengkonfigurasi crawler sehingga membuat tabel untuk data di partisi yang ada, lihat. Menggunakan beberapa sumber data dengan crawler
- Anda juga dapat membuat partisi dalam tabel langsung di Athena. Untuk informasi selengkapnya, lihat Partisi data di Athena.

Ketika Athena menjalankan kueri pada tabel yang dipartisi, Athena memeriksa untuk melihat apakah kolom yang dipartisi digunakan dalam klausa kueri. `WHERE` Jika kolom yang dipartisi digunakan, Athena meminta Katalog AWS Glue Data untuk mengembalikan spesifikasi partisi yang cocok dengan kolom partisi yang ditentukan. Spesifikasi partisi mencakup `LOCATION` properti yang memberi tahu Athena awalan Amazon S3 mana yang akan digunakan saat membaca data. Dalam

hal ini, hanya data yang disimpan dalam awalan ini yang dipindai. Jika Anda tidak menggunakan kolom yang dipartisi dalam `WHERE` klausa, Athena memindai semua file milik partisi tabel.

Untuk contoh menggunakan partisi dengan Athena untuk meningkatkan kinerja kueri dan mengurangi biaya kueri, lihat 10 kiat penyetelan kinerja teratas untuk Amazon Athena.

Format penyimpanan kolumnar

Apache Parquet dan ORC adalah format penyimpanan kolumnar yang dioptimalkan untuk pengambilan data dengan cepat dan digunakan dalam aplikasi analitis. AWS

Format penyimpanan kolom memiliki karakteristik berikut yang membuatnya cocok untuk digunakan dengan Athena:

- Kompresi berdasarkan kolom, dengan algoritme kompresi yang dipilih untuk tipe data kolom untuk menghemat ruang penyimpanan di Amazon S3 dan mengurangi ruang disk dan I/O selama pemrosesan kueri.
- Predikat pushdown di Parquet dan ORC memungkinkan kueri Athena untuk mengambil hanya blok yang dibutuhkan, meningkatkan performa kueri. Saat kueri Athena memperoleh nilai kolom tertentu dari data Anda, kueri tersebut menggunakan statistik dari predikat blok data, seperti nilai maks/min nilai, untuk menentukan apakah akan membaca atau melewatkan blok.
- Memisahkan data di Parquet dan ORC memungkinkan Athena membagi pembacaan data ke beberapa pembaca dan meningkatkan paralelisme selama pemrosesan kueri.

Untuk mengonversi data mentah yang ada dari format penyimpanan lain ke Parket atau ORC, Anda dapat menjalankan kueri CREATE TABLE AS SELECT (CTAS) di Athena dan menentukan format penyimpanan data sebagai Parket atau ORC, atau menggunakan Crawler. AWS Glue

Memilih antara Parket dan ORC

Pilihan antara ORC (Optimized Row Columnar) dan Parquet tergantung pada kebutuhan penggunaan spesifik Anda.

Apache Parquet menyediakan skema kompresi dan pengkodean data yang efisien dan sangat ideal untuk menjalankan kueri kompleks dan memproses data dalam jumlah besar. Parket dioptimalkan untuk digunakan dengan Apache Arrow, yang dapat menguntungkan jika Anda menggunakan alat yang terkait dengan Arrow.

ORC menyediakan cara yang efisien untuk menyimpan data Hive. File ORC seringkali lebih kecil dari file Parket, dan indeks ORC dapat membuat kueri lebih cepat. Selain itu, ORC mendukung tipe kompleks seperti struct, peta, dan daftar.

Saat memilih antara Parket dan ORC, pertimbangkan hal berikut:

Kinerja kueri — Karena Parket mendukung berbagai jenis kueri yang lebih luas, Parket mungkin menjadi pilihan yang lebih baik jika Anda berencana untuk melakukan kueri yang kompleks.

Tipe data yang kompleks — Jika Anda menggunakan tipe data yang kompleks, ORC mungkin menjadi pilihan yang lebih baik karena mendukung berbagai tipe data kompleks yang lebih luas.

Ukuran file — Jika ruang disk menjadi perhatian, ORC biasanya menghasilkan file yang lebih kecil, yang dapat mengurangi biaya penyimpanan.

Kompresi - Parket dan ORC memberikan kompresi yang baik, tetapi format terbaik untuk Anda dapat bergantung pada kasus penggunaan spesifik Anda.

Evolusi - Parket dan ORC mendukung evolusi skema, yang berarti Anda dapat menambah, menghapus, atau memodifikasi kolom dari waktu ke waktu.

Baik Parket dan ORC adalah pilihan yang baik untuk aplikasi data besar, tetapi pertimbangkan persyaratan skenario Anda sebelum memilih. Anda mungkin ingin melakukan tolok ukur pada data dan kueri untuk melihat format mana yang berkinerja lebih baik untuk kasus penggunaan Anda.

Mengonversi ke format kolumnar

Kinerja kueri Amazon Athena Anda meningkat jika Anda mengonversi data menjadi format kolom sumber terbuka, seperti parket Apache atau ORC.

Opsi untuk dengan mudah mengonversi data sumber seperti JSON atau CSV ke dalam format kolom termasuk menggunakan kueri CREATE TABLE AS atau menjalankan pekerjaan di. AWS Glue

- Anda dapat menggunakan kueri `CREATE TABLE AS` (CTAS) untuk mengonversi data menjadi Parket atau ORC dalam satu langkah. Sebagai contoh, lihat Contoh: Menulis hasil kueri ke format yang berbeda pada Contoh kueri CTAS halaman.
- Untuk informasi tentang menjalankan AWS Glue pekerjaan untuk mengubah data CSV menjadi Parket, lihat bagian “Ubah data dari format CSV ke Parket” di posting blog AWS Big Data Membangun fondasi danau data dengan dan Amazon AWS Glue S3. AWS Glue mendukung

menggunakan teknik yang sama untuk mengonversi data CSV ke ORC, atau data JSON ke Parquet atau ORC.

Partisi data di Athena

Dengan mempartisi data, Anda dapat membatasi jumlah data yang dipindai oleh setiap kueri, sehingga meningkatkan performa dan mengurangi biaya. Anda dapat mempartisi data Anda dengan kunci apa pun. Sebuah praktik umum adalah untuk partisi data berdasarkan waktu, sering mengarah ke skema partisi multi-level. Misalnya, pelanggan yang memiliki data datang setiap jam mungkin memutuskan untuk mempartisi menurut tahun, bulan, tanggal, dan jam. Pelanggan lain, yang memiliki data yang berasal dari berbagai sumber tetapi yang dimuat hanya sekali per hari, mungkin dipartisi oleh pengenal sumber data dan tanggal.

Athena dapat menggunakan partisi gaya Apache Hive, yang jalur datanya berisi pasangan nilai kunci yang dihubungkan dengan tanda yang sama (misalnya, atau). `country=us/...` `year=2021/month=01/day=26/...` Dengan demikian, jalur mencakup nama kunci partisi dan nilai yang diwakili oleh setiap jalur. Untuk memuat partisi Hive baru ke dalam tabel yang dipartisi, Anda dapat menggunakan MSCK REPAIR TABLE perintah, yang hanya berfungsi dengan partisi gaya Hive.

Athena juga dapat menggunakan skema partisi gaya non-sarang. Misalnya, CloudTrail log dan aliran pengiriman Firehose menggunakan komponen jalur terpisah untuk bagian tanggal seperti. `data/2021/01/26/us/6fc7845e.json` Untuk partisi gaya non-sarang seperti itu, Anda gunakan ALTER TABLE ADD PARTITION untuk menambahkan partisi secara manual.

Pertimbangan dan batasan

Saat menggunakan partisi, ingatlah hal-hal berikut:

- Jika Anda mengkueri tabel dipartisi dan menentukan partisi di `WHERE`, Athena memindai data hanya dari partisi itu. Untuk informasi selengkapnya, lihat Lokasi tabel dan partisi.
- Jika Anda mengeluarkan kueri terhadap bucket Amazon S3 dengan begitu banyak objek dan data yang tidak dipartisi, kueri tersebut dapat memengaruhi batas tingkat permintaan `GET` di Amazon S3 dan menyebabkan pengecualian Amazon S3. Untuk mencegah kesalahan, partisi data Anda. Selain itu, pertimbangkan untuk menyetel tingkat permintaan Amazon S3 Anda. Untuk informasi selengkapnya, lihat Pola desain praktik terbaik: Mengoptimalkan kinerja Amazon S3.
- Lokasi partisi yang akan digunakan dengan Athena harus menggunakan protokol `s3` (misalnya, `s3://DOC-EXAMPLE-BUCKET/folder/`). Di Athena, lokasi yang menggunakan protokol

lain (contohnya,`s3a://DOC-EXAMPLE-BUCKET/folder/`) akan mengakibatkan kegagalan permintaan ketika`MSCK REPAIR TABLE`query dijalankan pada tabel yang mengandung.

- Pastikan bahwa jalur Amazon S3 dalam huruf kecil bukan camel case (misalnya, `userid` sebagai gantinya `userId`). Jika jalur S3 dalam kasus unta,`MSCK REPAIR TABLE`tidak menambahkan partisi ke AWS Glue Data Catalog. Untuk informasi selengkapnya, lihat MSCK REPAIR TABLE.
- Karena `MSCK REPAIR TABLE` memindai folder dan subfoldernya untuk menemukan skema partisi yang cocok, pastikan untuk menyimpan data untuk tabel terpisah dalam hierarki folder terpisah. Misalnya, Anda memiliki data untuk tabel 1 in `s3://DOC-EXAMPLE-BUCKET1` dan data untuk tabel 2 in`s3://DOC-EXAMPLE-BUCKET1/table-2-data`. Jika kedua tabel dipartisi oleh string, `MSCK REPAIR TABLE` akan menambahkan partisi untuk tabel 2 ke tabel 1. Untuk menghindari hal ini, gunakan struktur folder terpisah seperti `s3://DOC-EXAMPLE-BUCKET1` dan `s3://DOC-EXAMPLE-BUCKET2` sebagai gantinya. Perhatikan bahwa perilaku ini konsisten dengan Amazon EMR dan Apache Hive.
- Jika Anda menggunakan AWS Glue Data Catalog with Athena, lihat AWS Glue titik akhir dan kuota untuk kuota layanan pada partisi per akun dan per tabel.
  - Meskipun Athena mendukung AWS Glue tabel kueri yang memiliki 10 juta partisi, Athena tidak dapat membaca lebih dari 1 juta partisi dalam satu pemindaian. Dalam skenario seperti itu, pengindeksan partisi dapat bermanfaat. Untuk informasi selengkapnya, lihat artikel AWS Big Data Blog Meningkatkan kinerja kueri Amazon Athena menggunakan indeks AWS Glue Data Catalog partisi.
- Untuk meminta peningkatan kuota partisi jika Anda menggunakan AWS Glue Data Catalog, kunjungi konsol Service Quotas untuk. AWS Glue

Membuat dan memuat tabel dengan data yang dipartisi

Untuk membuat tabel yang menggunakan partisi, gunakan `PARTITIONED BY` klausa dalam pernyataan AndaCREATE TABLE. `PARTITIONED BY`Klausa mendefinisikan kunci untuk mempartisi data, seperti pada contoh berikut. `LOCATION`Klausa menentukan lokasi root dari data yang dipartisi.

```
CREATE EXTERNAL TABLE users (
first string,
last string,
username string
)
PARTITIONED BY (id string)
STORED AS parquet
```

```
LOCATION 's3://DOC-EXAMPLE-BUCKET'
```

Setelah Anda membuat tabel, Anda memuat data di partisi untuk kueri. Untuk partisi gaya Hive, Anda lari. MSCK REPAIR TABLE Untuk partisi gaya non-sarang, Anda gunakan ALTER TABLE ADD PARTITION untuk menambahkan partisi secara manual.

Mempersiapkan gaya Hive dan data gaya non-Hive untuk kueri

Bagian berikut menunjukkan bagaimana mempersiapkan gaya Hive dan data gaya non-Hive untuk query di Athena.

Skenario 1: Data disimpan di Amazon S3 dalam format Hive

Dalam skenario ini, partisi disimpan dalam folder terpisah di Amazon S3. Misalnya, berikut adalah daftar sebagian untuk contoh tayangan iklan yang dihasilkan oleh `aws s3 ls`perintah, yang mencantumkan objek S3 di bawah awalan yang ditentukan:

```
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/
```

Di sini, log disimpan dengan nama kolom (dt) ditetapkan sama dengan tanggal, jam, dan kenaikan menit. Saat Anda memberikan DDL dengan lokasi folder induk, skema, dan nama kolom dipartisi, Athena dapat mengkueri data dalam subfolder tersebut.

Buat tabel

Untuk membuat tabel dari data ini, buat partisi bersama 'dt' seperti pada pernyataan Athena DDL berikut:

```
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/' ;
```

Tabel ini menggunakan Hive asli JSON serializer-deserializer untuk membaca data JSON yang disimpan di Amazon S3. Untuk informasi selengkapnya tentang format yang didukung, lihat Format yang didukung SerDes dan data.

Jalankan MSCK REPAIR TABLE

Setelah Anda menjalankan `CREATE TABLE` query, jalankan `MSCK REPAIR TABLE` perintah di Athena query editor untuk memuat partisi, seperti pada contoh berikut.

```
MSCK REPAIR TABLE impressions
```

Setelah Anda menjalankan perintah ini, data siap untuk query.

Kueri data

Kueri data dari tabel tayangan menggunakan kolom partisi. Inilah contohnya:

```
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
```

Kueri ini harus menunjukkan hasil yang mirip dengan berikut ini:

```
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
```

Skenario 2: Data tidak dipartisi dalam format Hive

Dalam contoh berikut, `aws s3 ls` perintah menunjukkan log ELB yang disimpan di Amazon S3. Perhatikan bagaimana tata letak data tidak menggunakan `key=value` pasangan dan oleh karena itu tidak dalam format Hive. (`--recursive`Opsi untuk `aws s3 ls` perintah menentukan bahwa semua file atau objek di bawah direktori atau awalan yang ditentukan dicantumkan.)

```
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_$folder$
```

Jalankan ALTER TABLE ADD PARTITION

Karena data tidak dalam format Hive, Anda tidak dapat menggunakan `MSCK REPAIR TABLE` perintah untuk menambahkan partisi ke tabel setelah Anda membuatnya. Sebagai gantinya, Anda dapat menggunakan ALTER TABLE ADD PARTITION perintah untuk menambahkan setiap partisi secara manual. Misalnya, untuk memuat data di s3://athena-examples - *myregion* /elb/plaintext/2015/01/01/, Anda dapat menjalankan kueri berikut. Perhatikan bahwa kolom partisi terpisah untuk setiap folder Amazon S3 tidak diperlukan, dan bahwa nilai kunci partisi dapat berbeda dari kunci Amazon S3.

```
ALTER TABLE elb_logs_raw_native_part ADD PARTITION (dt='2015-01-01') location 's3://
athena-examples-us-west-1/elb/plaintext/2015/01/01/'
```

Jika partisi sudah ada, Anda menerima kesalahan Partisi sudah ada. Untuk menghindari kesalahan ini, Anda dapat menggunakan klausa `IF NOT EXISTS`. Untuk informasi selengkapnya, lihat ALTER

TABLE ADD PARTITION. Untuk menghapus partisi, Anda dapat menggunakanALTER TABLE DROP PARTITION.

Proyeksi partisi

Untuk menghindari keharusan mengelola partisi, Anda dapat menggunakan proyeksi partisi. Proyeksi partisi adalah opsi untuk tabel yang sangat dipartisi yang strukturnya diketahui sebelumnya. Dalam proyeksi partisi, nilai partisi dan lokasi dihitung dari properti tabel yang Anda konfigurasikan daripada dibaca dari repositori metadata. Karena perhitungan dalam memori lebih cepat daripada pencarian jarak jauh, penggunaan proyeksi partisi dapat secara signifikan mengurangi runtime kueri.

Untuk informasi selengkapnya, lihat Proyeksi partisi dengan Amazon Athena.

Sumber daya tambahan

- Untuk informasi tentang opsi partisi untuk data Firehose, lihat. Contoh Amazon Data Firehose
- Anda dapat mengotomatiskan penambahan partisi dengan menggunakan driver JDBC.
- Anda dapat menggunakan CTAS dan INSERT INTO untuk partisi set data. Untuk informasi selengkapnya, lihat Menggunakan CTAS dan INSERT INTO untuk ETL dan analisis data.

Proyeksi partisi dengan Amazon Athena

Anda dapat menggunakan proyeksi partisi di Athena untuk mempercepat pemrosesan kueri tabel yang sangat dipartisi dan mengotomatiskan manajemen partisi.

Dalam proyeksi partisi, Athena menghitung nilai partisi dan lokasi menggunakan properti tabel yang Anda konfigurasikan langsung pada tabel Anda. AWS Glue Properti tabel memungkinkan Athena untuk 'memproyeksikan', atau menentukan, informasi partisi yang diperlukan alih-alih harus melakukan pencarian metadata yang lebih memakan waktu di. AWS Glue Data Catalog Karena operasi dalam memori sering lebih cepat daripada operasi jarak jauh, proyeksi partisi dapat mengurangi runtime kueri terhadap tabel yang sangat dipartisi. Tergantung pada karakteristik spesifik dari kueri dan data yang mendasari, partisi proyeksi dapat secara signifikan mengurangi permintaan runtime untuk mengkueri yang dibatasi pada partisi metadata pengambilan.

Pemangkasan dan proyeksi untuk tabel yang dipartisi berat

Pemangkasan partisi mengumpulkan metadata dan “plum” untuk hanya partisi yang berlaku untuk permintaan Anda. Ini sering mempercepat kueri. Athena menggunakan pemangkasan partisi untuk semua tabel dengan kolom partisi, termasuk tabel yang dikonfigurasi untuk proyeksi partisi.

Biasanya, saat memproses kueri, Athena menelepon sebelum melakukan pemangkasan AWS Glue Data Catalog partisi. `GetPartitions` Jika tabel memiliki sejumlah besar partisi, menggunakan`GetPartitions`dapat mempengaruhi performa negatif. Untuk menghindarinya, Anda bisa menggunakan proyeksi partisi. Proyeksi partisi memungkinkan Athena untuk menghindari panggilan`GetPartitions`karena konfigurasi proyeksi partisi memberikan Athena semua informasi yang diperlukan untuk membangun partisi itu sendiri.

Menggunakan proyeksi partisi

Untuk menggunakan proyeksi partisi, Anda menentukan rentang nilai partisi dan jenis proyeksi untuk setiap kolom partisi dalam properti tabel di AWS Glue Data Catalog atau di metastore Hive eksternal Anda. Properti kustom ini di atas tabel memungkinkan Athena untuk mengetahui pola partisi apa yang diharapkan saat menjalankan kueri di atas tabel. Selama eksekusi kueri, Athena menggunakan informasi ini untuk memproyeksikan nilai partisi alih-alih mengambilnya dari metastore Hive AWS Glue Data Catalog atau eksternal. Ini tidak hanya mengurangi waktu eksekusi kueri tetapi juga mengotomatiskan manajemen partisi karena menghilangkan kebutuhan untuk secara manual membuat partisi di Athena, AWS Glue, atau metastore Hive eksternal Anda.

> **Important**
>
> Mengaktifkan proyeksi partisi pada tabel menyebabkan Athena mengabaikan metadata partisi apa pun yang terdaftar ke tabel di metastore or Hive. AWS Glue Data Catalog

Kasus penggunaan

Skenario tempat proyeksi partisi berguna meliputi:

- Kueri terhadap tabel yang sangat dipartisi tidak lengkap secepat yang Anda inginkan.
- Anda secara teratur menambahkan partisi ke tabel sebagai partisi tanggal atau waktu baru dibuat dalam data Anda. Dengan proyeksi partisi, Anda mengonfigurasi rentang tanggal relatif yang dapat digunakan sebagai data baru tiba.
- Anda memiliki data yang sangat dipartisi di Amazon S3. Data tidak praktis untuk dimodelkan di metastore Anda AWS Glue Data Catalog atau Hive, dan kueri Anda hanya membaca sebagian kecil saja.

Struktur partisi yang dapat diproyeksikan

Proyeksi partisi paling mudah dikonfigurasi saat partisi Anda mengikuti pola yang dapat diprediksi seperti, tetapi tidak terbatas pada, berikut:

- Bilangan bulat— Setiap urutan kontinu bilangan bulat seperti`[1, 2, 3, 4, ..., 1000]`atau`[0500, 0550, 0600, ..., 2500]`.
- Tanggal— Setiap urutan kontinu tanggal atau datetime seperti`[20200101, 20200102, ..., 20201231]`atau`[1-1-2020 00:00:00, 1-1-2020 01:00:00, ..., 12-31-2020 23:00:00]`.
- Nilai yang disebutkan - Satu set terbatas dari nilai yang disebutkan seperti kode bandara atau. Wilayah AWS
- Layanan AWS log — Layanan AWS log biasanya memiliki struktur yang diketahui yang skema partisinya dapat Anda tentukan AWS Glue dan karena itu Athena dapat digunakan untuk proyeksi partisi.

Menyesuaikan template jalur partisi

Secara default, Athena membangun lokasi partisi menggunakan formulir`s3://DOC-EXAMPLE-BUCKET/<table-root>/partition-col-1=<partition-col-1-val>/partition-col-2=<partition-col-2-val>/`, tetapi jika data Anda diatur secara berbeda, Athena menawarkan mekanisme untuk menyesuaikan templat jalur ini. Untuk langkah, lihat Menentukan lokasi penyimpanan S3 kustom.

Pertimbangan dan batasan

Pertimbangan berikut berlaku untuk instans Mac:

- Proyeksi partisi menghilangkan kebutuhan untuk menentukan partisi secara manual di AWS Glue atau metastore Hive eksternal.
- Saat Anda mengaktifkan proyeksi partisi di atas meja, Athena mengabaikan metadata partisi apa pun di metastore Hive atau eksternal AWS Glue Data Catalog untuk tabel itu.
- Jika partisi yang diproyeksikan tidak ada di Amazon S3, Athena masih akan memproyeksikan partisi. Athena tidak membuang kesalahan, tapi tidak ada data dikembalikan. Namun, jika terlalu banyak partisi Anda kosong, kinerjanya bisa lebih lambat dibandingkan dengan AWS Glue partisi tradisional. Jika lebih dari separuh partisi yang diproyeksikan kosong, sebaiknya gunakan partisi tradisional.

- Kueri untuk nilai-nilai yang berada di luar batas rentang yang ditetapkan untuk proyeksi partisi tidak mengembalikan kesalahan. Sebaliknya, kueri berjalan, tetapi kembali nol baris. Misalnya, jika Anda memiliki data terkait waktu yang dimulai pada tahun 2020 dan didefinisikan sebagai`'projection.timestamp.range'='2020/01/01,NOW'`, kueri seperti`SELECT * FROM table-name WHERE timestamp = '2019/02/02'` akan selesai dengan sukses, tapi kembali nol baris.
- Proyeksi partisi hanya dapat digunakan jika tabel dipertanyakan melalui Athena. Jika tabel yang sama dibaca melalui layanan lain seperti Amazon Redshift Spectrum, Athena for Spark, atau Amazon EMR, metadata partisi standar digunakan.
- Karena proyeksi partisi adalah fitur DML-only, `SHOW PARTITIONS` tidak mencantumkan partisi yang diproyeksikan oleh Athena tetapi tidak terdaftar dalam katalog atau metastore Hive eksternal. AWS Glue
- Athena tidak menggunakan properti tabel tampilan sebagai konfigurasi untuk proyeksi partisi. Untuk mengatasi keterbatasan ini, mengonfigurasi dan mengaktifkan partisi proyeksi dalam tabel properti untuk tabel yang tampilan referensi.
- Filter data Lake Formation tidak dapat digunakan dengan proyeksi partisi di mesin Athena versi 2.

Video

Video berikut menunjukkan bagaimana menggunakan proyeksi partisi untuk meningkatkan performa kueri Anda di Athena.

Proyeksi partisi dengan Amazon Athena

Topik

- Menyiapkan proyeksi partisi
- Jenis yang didukung untuk proyeksi partisi
- Partisi ID dinamis
- Contoh Amazon Data Firehose

Menyiapkan proyeksi partisi

Menyiapkan proyeksi partisi di properti tabel adalah proses dua langkah:

1. Tentukan rentang data dan pola yang relevan untuk setiap kolom partisi, atau gunakan templat kustom.

2. Aktifkan proyeksi partisi untuk tabel.

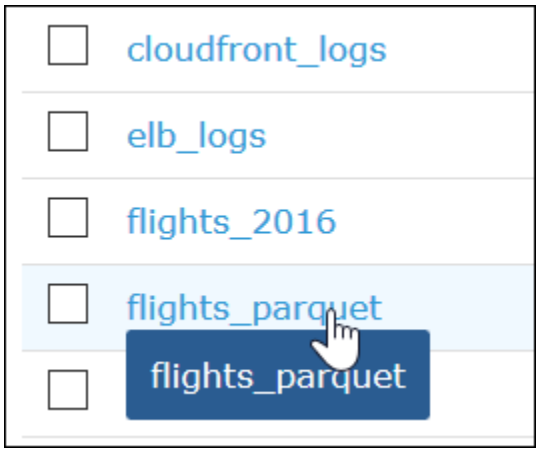
> **Note**
>
> Sebelum Anda menambahkan properti proyeksi partisi ke tabel yang ada, kolom partisi yang Anda siapkan properti proyeksi partisi harus sudah ada dalam skema tabel. Jika kolom partisi belum ada, Anda harus menambahkan kolom partisi ke tabel yang ada secara manual. AWS Glue tidak melakukan langkah ini untuk Anda secara otomatis.

Bagian ini menunjukkan cara mengatur properti tabel untuk AWS Glue. Untuk mengaturnya, Anda dapat menggunakan AWS Glue konsol, CREATE TABLE kueri Athena, atau operasi. AWS Glue API Prosedur berikut menunjukkan cara mengatur properti di AWS Glue konsol.

Untuk mengkonfigurasi dan mengaktifkan proyeksi partisi menggunakan konsol AWS Glue

1. Masuk ke AWS Management Console dan buka AWS Glue konsol di https://console.aws.amazon.com/glue/.
2. Pilih tab Tabel.

   Pada tab Tabel, Anda dapat mengedit tabel yang ada, atau memilih Tambahkan tabel untuk membuat yang baru. Untuk informasi tentang menambahkan tabel secara manual atau dengan crawler, lihat Bekerja dengan tabel di AWS Glue konsol di Panduan AWS Glue Pengembang.
3. Dalam daftar tabel, pilih tautan untuk tabel yang ingin Anda edit.



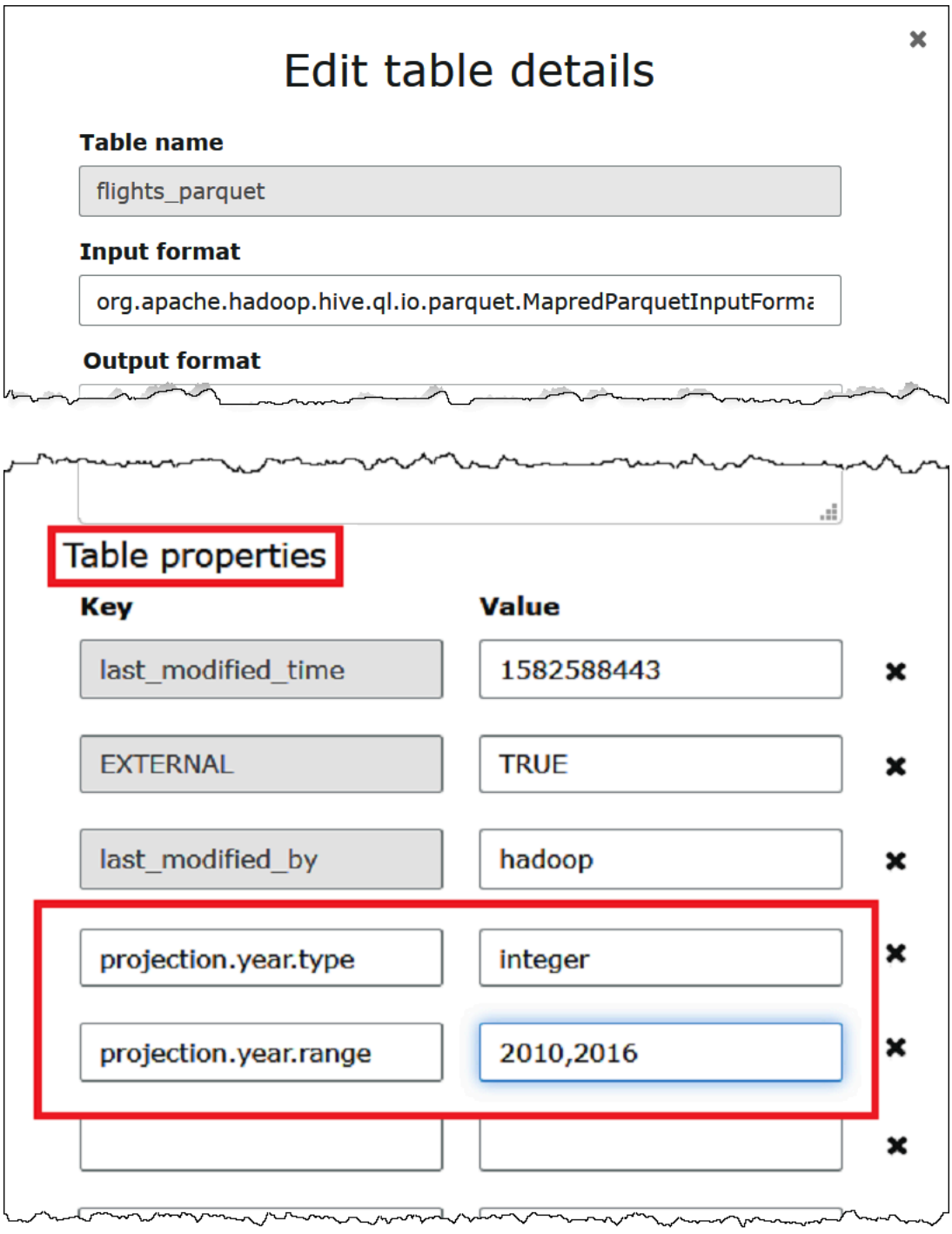
4. Pilih Tindakan, Edit tabel.
5. Pada halaman Edit tabel, di bagian properti Tabel, untuk setiap kolom yang dipartisi, tambahkan pasangan kunci-nilai berikut:

a. Untuk Kunci, tambahkan`projection.`*`columnName`*`.type`.

b. Untuk Nilai, tambahkan salah satu tipe yang didukung: `enum`, `integer`, `date`, atau `injected`. Untuk informasi selengkapnya, lihat Jenis yang didukung untuk proyeksi partisi.

6. Mengikuti petunjuk di Jenis yang didukung untuk proyeksi partisi , tambahkan pasangan kunci-nilai tambahan sesuai dengan kebutuhan konfigurasi Anda.

   Contoh konfigurasi tabel berikut mengkonfigurasi `year` kolom untuk proyeksi partisi, membatasi nilai-nilai yang dapat dikembalikan ke rentang dari 2010 hingga 2016.

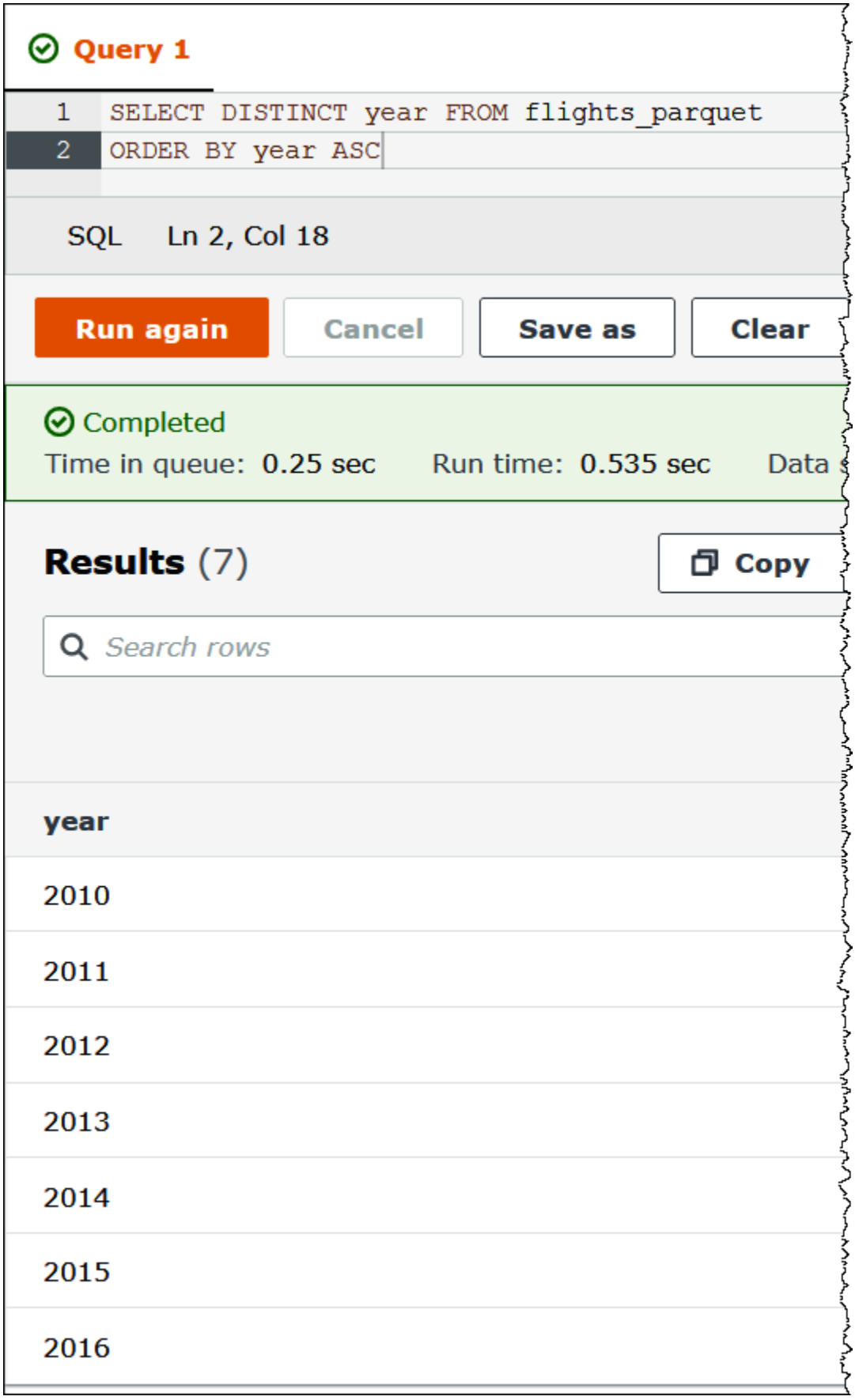


7. Tambahkan pasangan nilai kunci untuk mengaktifkan proyeksi partisi. Untuk Kunci, masukkan `projection.enabled`, dan untuk Nilai, masukkan `true`.

> **Note**
>
> Anda dapat menonaktifkan proyeksi partisi pada tabel ini kapan saja dengan menetapkan `projection.enabled` ke `false`.

8. Setelah selesai, pilih Simpan.
9. Di Editor Kueri Athena, uji kueri kolom yang Anda konfigurasikan untuk tabel.

   Contoh kueri berikut menggunakan`SELECT DISTINCT`untuk mengembalikan nilai-nilai unik dari`year`kolom. Basis data berisi data dari 1987 hingga 2016, tetapi `projection.year.range` properti membatasi nilai yang dikembalikan ke tahun 2010 hingga 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**
>
> Jika Anda mengaturprojection.enabledketruetetapi gagal untuk mengonfigurasi satu atau lebih partisi kolom, Anda menerima pesan kesalahan seperti berikut: `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).`

Menentukan lokasi penyimpanan S3 kustom

Saat mengedit properti tabel AWS Glue, Anda juga dapat menentukan templat jalur Amazon S3 khusus untuk partisi yang diproyeksikan. Template kustom memungkinkan Athena untuk benar memetakan nilai partisi ke lokasi file Amazon S3 kustom yang tidak mengikuti`.../column=value/...`Pola.

Menggunakan templat kustom adalah opsional. Namun, jika Anda menggunakan templat kustom, templat harus berisi placeholder untuk setiap kolom partisi. Lokasi template harus diakhiri dengan garis miring ke depan sehingga file data yang dipartisi hidup dalam “folder” per partisi.

Untuk menentukan templat lokasi partisi kustom

1. Mengikuti langkah-langkah untuk mengkonfigurasi dan mengaktifkan proyeksi partisi menggunakan AWS Glue konsol, tambahkan tambahan pasangan kunci-nilai yang menentukan template kustom sebagai berikut:

   a. Untuk Kunci, masukkan `storage.location.template`.

   b. UntukNilai, tentukan lokasi yang mencakup placeholder untuk setiap kolom partisi. Pastikan bahwa setiap placeholder (dan jalur S3 itu sendiri) diakhiri dengan satu garis miring ke depan.

      Nilai templat contoh berikut mengasumsikan tabel dengan kolom partisia,b, danc.

      ```
      s3://DOC-EXAMPLE-BUCKET/table_root/a=${a}/${b}/some_static_subdirectory/${c}/
      ```

```
s3://DOC-EXAMPLE-BUCKET/table_root/c=${c}/${b}/some_static_subdirectory/${a}/
${b}/${c}/${c}/
```

Untuk tabel yang sama, nilai templat contoh berikut tidak valid karena mengandung tidak ada tempat untuk kolom`c`.

```
s3://DOC-EXAMPLE-BUCKET/table_root/a=${a}/${b}/some_static_subdirectory/
```

2. Pilih Terapkan.

Jenis yang didukung untuk proyeksi partisi

Sebuah tabel dapat memiliki kombinasi`enum`,`integer`,`date`,atau`injected`jenis kolom partisi.

Jenis enum

Gunakan `enum` tipe untuk kolom partisi yang nilainya adalah anggota dari kumpulan yang disebutkan (misalnya, kode bandara atau). Wilayah AWS

Tentukan properti partisi dalam tabel sebagai berikut:

Nama properti	Contoh nilai	Deskripsi
`projection.` *`columnName`* `.type`	`enum`	Wajib. Jenis proyeksi yang akan digunakan untuk kolom*`columnName`* . Nilai harus`enum`(case sensitive) untuk memberi sinyal penggunaan tipe enum. Memimpin dan mengikuti ruang putih diperbolehkan.
`projection.` *`columnName`* `.values`	`A,B,C,D,E,F,G,Unkn own`	Wajib. Sebuah daftar dipisahkan koma dari nilai partisi enumerasi untuk kolom*`columnName`* . Setiap

Nama properti	Contoh nilai	Deskripsi
		ruang putih dianggap sebagai bagian dari nilai enum.

> **Note**
>
> Sebagai praktik terbaik kami sarankan untuk membatasi penggunaan`enum`berdasarkan proyeksi partisi untuk beberapa lusin atau kurang. Meskipun tidak ada batasan spesifik untuk `enum` proyeksi, ukuran total metadata tabel Anda tidak dapat melebihi AWS Glue batas sekitar 1 MB saat gzip dikompresi. Perhatikan bahwa batas ini dibagi di bagian kunci dari tabel Anda seperti nama kolom, lokasi, format penyimpanan, dan lain-lain. Jika Anda menemukan diri Anda menggunakan lebih dari beberapa lusin ID unik di`enum`proyeksi, mempertimbangkan pendekatan alternatif seperti bucketing ke sejumlah kecil dari nilai-nilai unik di bidang pengganti. Dengan melakukan trading di luar kardinalitas, Anda dapat mengontrol jumlah nilai unik di`enum`Bidang.

Jenis bilangan bulat

Gunakan tipe integer untuk kolom partisi yang nilainya mungkin ditafsirkan sebagai bilangan bulat dalam kisaran yang ditentukan. Kolom bilangan bulat yang diproyeksikan saat ini terbatas pada kisaran Java yang ditandatangani panjang (-2^{63}ke $2^{63}-1$ inklusif).

Nama properti	Contoh nilai	Deskripsi
`projection.` *`columnName`* `.type`	`integer`	Wajib. Jenis proyeksi yang akan digunakan untuk kolom*`columnName`* . Nilai harus berupa`integer`(case sensitive) untuk sinyal penggunaan tipe integer. Memimpin dan mengikuti ruang putih diperbolehkan.
`projection.` *`columnName`* `.range`	`0,10`	Wajib. Sebuah daftar dua elemen dipisahkan koma

Nama properti	Contoh nilai	Deskripsi

	`-1,8675309`
	
	`0001,9999`
		yang menyediakan nilai-nilai rentang minimum dan maksimum yang akan dikembalikan oleh kueri pada kolom*`columnName`* . Perhatikan bahwa nilai harus dipisahkan dengan koma, bukan tanda hubung. Nilai-nilai ini inklusif, bisa negatif, dan dapat memiliki nol terkemuka . Memimpin dan mengikuti ruang putih diperbolehkan.

`projection.` *`columnName`* `.interval`	`1`
	
	`5`
		Tidak wajib. Sebuah bilangan bulat positif yang menentukan interval antara nilai-nilai partisi berturut-turut untuk kolom*`columnName`* . Contohnya,`range`nilai “1,3" dengan`interval`Nilai “1" menghasilkan nilai 1, 2, dan 3. Sama`range`Nilai dengan`interval`nilai “2" menghasilkan nilai-nilai 1 dan 3, melewatkan 2. Memimpin dan mengikuti ruang putih diperbolehkan. Default-nya adalah 1.

Nama properti	Contoh nilai	Deskripsi

`projection.`*`columnName`*`.digits`	`1`
	
	`5`
		Tidak wajib. Sebuah bilangan bulat positif yang menentukan jumlah digit untuk dimasukkan dalam representasi akhir nilai partisi untuk kolom*`columnName`* . Contohnya,`range`nilai “1,3" yang memiliki`digits`Nilai “1" menghasilkan nilai 1, 2, dan 3. Sama`range`Nilai dengan`digits`nilai “2" menghasilkan nilai-nilai 01, 02, dan 03. Memimpin dan mengikuti ruang putih diperbolehkan. Default adalah tidak ada jumlah statis digit dan tidak ada nol terkemuka.

Jenis tanggal

Gunakan tipe tanggal untuk kolom partisi yang nilainya dapat diinterpretasikan sebagai tanggal (dengan waktu opsional) dalam rentang yang ditetapkan.

> **Important**
>
> Kolom tanggal yang Diproyeksikan dihasilkan dalam Waktu Universal Terkoordinasi (UTC) pada waktu eksekusi kueri.

Nama properti	Contoh nilai	Deskripsi
`projection.`*`columnName`*`.type`	`date`	Wajib. Jenis proyeksi yang akan digunakan untuk kolom*`columnName`* . Nilai harus berupa`date`(case insensitive) untuk memberi

Nama properti	Contoh nilai	Deskripsi
		sinyal penggunaan tipe tanggal. Memimpin dan mengikuti ruang putih diperbolehkan.

`projection.` *`columnName`* `.range`	`201701,201812`
	
	`01-01-2010,12-31-2018`
	
	`NOW-3YEARS,NOW`
	
	`201801,NOW+1MONTH`
		Wajib. Sebuah dua elemen, daftar dipisahkan koma yang menyediakan minimum dan maksimum`range`nilai untuk kolom*`columnName`* . Nilai-nilai ini inklusif dan dapat menggunakan format yang kompatibel dengan Java`java.time.*` Jenis tanggal. Kedua nilai minimum dan maksimum harus menggunakan format yang sama. Format yang ditentukan dalam`.format`properti harus format yang digunakan untuk nilai-nilai ini.
		
		Kolom ini juga dapat berisi string tanggal relatif, diformat dalam pola ekspresi reguler ini:
		
		`\s*NOW\s*(([\+\-])\s*([0-9]+)\s*(YEARS?|MONTHS?|WEEKS?|DAYS?|HOURS?|MINUTES?|SECONDS?)\s*)?`
		
		Ruang putih diperbolehkan, tetapi dalam literal tanggal dianggap bagian dari string tanggal sendiri.

`projection.` *`columnName`* `.format`	`yyyyMM`
	
	`dd-MM-yyyy`
	
	`dd-MM-yyyy-HH-mm-ss`
		Wajib. String format tanggal berdasarkan format tanggal Java DateTimeFormatter. Dapat didukung`Java.time.*` Jenis.

Nama properti	Contoh nilai	Deskripsi

`projection.`*`columnNam e`*`.interval`	`1`
	
	`5`
		Sebuah bilangan bulat positif yang menentukan interval antara nilai-nilai partisi berturut-turut untuk kolom*`columnNam e`* . Contohnya,`rangeNilai2017-01,2 018-12` dengan`intervalNilai1`dan sebuah`interval.unit` Nilai`MONTHS`menghasilkan nilai 2017-01, 2017-02, 2017-03, dan seterusnya. Sama`rangeNilai` dengan`intervalNilai2`dan sebuah`interval.unit` Nilai`MONTHS`menghasilkan nilai 2017-01, 2017-03, 2017-05, dan seterusnya. Spasi di awal dan akhir kalimat tidak diperbolehkan.
		
		Saat tanggal yang diberikan berada pada presisi satu hari atau satu bulan,`interval` adalah opsional dan default untuk 1 hari atau 1 bulan, masing-masing. Sebaliknya, `interval` diperlukan.

`projection.`*`columnNam e`*`.interval.unit`	`YEARS`
	
	`MONTHS`
	
	`WEEKS`
	
	`DAYS`
	
	`HOURS`
	
	`MINUTES`
	
	`SECONDS`
	
	`MILLIS`
		Kata satuan waktu yang mewakili bentuk serial dari a ChronoUnit. Kemungkinan nilai adalah `YEARS`, `MONTHS`, `WEEKS`, `DAYS`, `HOURS`, `MINUTES`, `SECONDS`, dan `MILLIS`. Nilai-nilai ini tidak peka huruf besar.
		
		Saat tanggal yang diberikan berada pada presisi satu hari atau satu bulan,`interval.unit` adalah opsional dan default untuk 1 hari atau 1 bulan, masing-masing. Sebaliknya, `interval.unit` diperlukan.

Jenis yang disuntikkan

Gunakan jenis disuntikkan untuk partisi kolom dengan nilai-nilai yang mungkin tidak prosedural dihasilkan dalam beberapa kisaran logis tetapi yang disediakan dalam query`WHERE`klausul sebagai nilai tunggal.

Penting untuk diingat hal-hal berikut:

- Kueri pada kolom disuntikkan gagal jika ekspresi filter tidak disediakan untuk setiap kolom disuntikkan.
- Kueri dengan beberapa nilai untuk ekspresi filter pada kolom yang disuntikkan hanya berhasil jika nilainya terpisah.
- Hanya kolom`string`didukung.

Nama properti	Nilai	Deskripsi
`projection.` *`columnNam e`* `.type`	`inje`	Wajib. Jenis proyeksi yang akan digunakan untuk kolom*`columnName`* . Hanya`string`didukung. Nilai yang ditentukan harus`injected`(kasus tidak sensitif). Memimpin dan mengikuti ruang putih diperbolehkan.

Untuk informasi selengkapnya, lihat Menggunakan tipe `injected` proyeksi.

Partisi ID dinamis

Ketika data Anda dipartisi oleh properti dengan kardinalitas tinggi atau ketika nilai tidak dapat diketahui sebelumnya, Anda dapat menggunakan jenis proyeksi. `injected` Contoh properti tersebut adalah nama pengguna, dan ID perangkat atau produk. Bila Anda menggunakan jenis `injected` proyeksi untuk mengkonfigurasi kunci partisi, Athena menggunakan nilai-nilai dari query itu sendiri untuk menghitung set partisi yang akan dibaca.

Agar Athena dapat menjalankan kueri pada tabel yang memiliki kunci partisi yang dikonfigurasi dengan jenis `injected` proyeksi, berikut ini harus benar:

- Kueri Anda harus menyertakan setidaknya satu nilai untuk kunci partisi.
- Nilai harus literal atau ekspresi yang dapat dievaluasi tanpa membaca data apa pun.

Jika salah satu kriteria ini tidak terpenuhi, kueri Anda gagal dengan kesalahan berikut:

CONSTRAINT_VIOLATION: Untuk kolom partisi proyeksi yang disuntikkan *column_name*, klausa WHERE harus berisi hanya kondisi kesetaraan statis, dan setidaknya satu kondisi seperti itu harus ada.

Menggunakan tipe **injected** proyeksi

Bayangkan Anda memiliki kumpulan data yang terdiri dari peristiwa dari perangkat IoT, yang dipartisi pada ID perangkat. Kumpulan data ini memiliki karakteristik sebagai berikut:

- ID perangkat dihasilkan secara acak.
- Perangkat baru sering disediakan.
- Saat ini ada ratusan ribu perangkat, dan di masa depan akan ada jutaan.

Kumpulan data ini sulit dikelola menggunakan metastores tradisional. Sulit untuk menjaga partisi tetap sinkron antara penyimpanan data dan metastore, dan partisi penyaringan bisa lambat selama perencanaan kueri. Tetapi jika Anda mengonfigurasi tabel untuk menggunakan proyeksi partisi dan menggunakan jenis `injected` proyeksi, Anda memiliki dua keuntungan: Anda tidak perlu mengelola partisi di metastore, dan kueri Anda tidak harus mencari metadata partisi.

`CREATE TABLE`Contoh berikut membuat tabel untuk set data peristiwa perangkat yang baru saja dijelaskan. Tabel menggunakan jenis proyeksi yang disuntikkan.

```
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}"
)
```

Contoh kueri berikut mencari jumlah peristiwa yang diterima dari tiga perangkat tertentu selama 12 jam.

```
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
```

Ketika Anda menjalankan query ini, Athena melihat tiga nilai untuk kunci `device_id` partisi dan menggunakannya untuk menghitung lokasi partisi. Athena menggunakan nilai `storage.location.template` properti untuk menghasilkan lokasi berikut:

- `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`

Jika Anda meninggalkan `storage.location.template` properti dari konfigurasi proyeksi partisi, Athena menggunakan partisi gaya HIVE untuk memproyeksikan lokasi partisi berdasarkan nilai `LOCATION` dalam (misalnya,). `s3://DOC-EXAMPLE-BUCKET/prefix/device_id=4a770164-0392-4a41-8565-40ed8cec737e`

Contoh Amazon Data Firehose

Saat Anda menggunakan Firehose untuk mengirimkan data ke Amazon S3, konfigurasi default akan menulis objek dengan kunci yang terlihat seperti contoh berikut:

```
s3://DOC-EXAMPLE-BUCKET/prefix/yyyy/MM/dd/HH/file.extension
```

Untuk membuat tabel Athena yang menemukan partisi secara otomatis pada waktu kueri, alih-alih harus menambahkannya ke AWS Glue Data Catalog saat data baru tiba, Anda dapat menggunakan proyeksi partisi.

`CREATE TABLE`Contoh berikut menggunakan konfigurasi Firehose default.

```
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}/"
)
```

`TBLPROPERTIES`Klausul dalam `CREATE TABLE` pernyataan tersebut memberi tahu Athena sebagai berikut:

- Gunakan proyeksi partisi saat menanyakan tabel
- Kunci partisi `datehour` adalah tipe `date` (yang mencakup waktu opsional)
- Bagaimana tanggal diformat
- Kisaran waktu tanggal. Perhatikan bahwa nilai harus dipisahkan dengan koma, bukan tanda hubung.
- Di mana menemukan data di Amazon S3.

Saat Anda menanyakan tabel, Athena menghitung nilai `datehour` dan menggunakan templat lokasi penyimpanan untuk menghasilkan daftar lokasi partisi.

Menggunakan **`date`** tipe

Bila Anda menggunakan `date` tipe untuk kunci partisi yang diproyeksikan, Anda harus menentukan rentang. Karena Anda tidak memiliki data untuk tanggal sebelum aliran pengiriman Firehose dibuat, Anda dapat menggunakan tanggal pembuatan sebagai awal. Dan karena Anda tidak memiliki data untuk tanggal di masa depan, Anda dapat menggunakan token khusus `NOW` sebagai akhir.

Dalam `CREATE TABLE` contoh, tanggal mulai ditentukan sebagai 1 Januari 2021 pada tengah malam UTC.

> **Note**
>
> Konfigurasikan rentang yang cocok dengan data Anda sedekat mungkin sehingga Athena hanya mencari partisi yang ada.

Ketika kueri dijalankan pada tabel sampel, Athena menggunakan kondisi pada tombol `datehour` partisi dalam kombinasi dengan rentang untuk menghasilkan nilai. Pertimbangkan kueri berikut:

```
SELECT *
FROM my_ingested_data
WHERE datehour >= '2020/12/15/00'
AND datehour < '2021/02/03/15'
```

Kondisi pertama dalam `SELECT` kueri menggunakan tanggal yang sebelum dimulainya rentang tanggal yang ditentukan oleh `CREATE TABLE` pernyataan. Karena konfigurasi proyeksi partisi tidak menetapkan partisi untuk tanggal sebelum 1 Januari 2021, Athena mencari data hanya di lokasi berikut, dan mengabaikan tanggal sebelumnya dalam kueri.

```
s3://DOC-EXAMPLE-BUCKET/prefix/2021/01/01/00/
s3://DOC-EXAMPLE-BUCKET/prefix/2021/01/01/01/
s3://DOC-EXAMPLE-BUCKET/prefix/2021/01/01/02/
...
s3://DOC-EXAMPLE-BUCKET/prefix/2021/02/03/12/
s3://DOC-EXAMPLE-BUCKET/prefix/2021/02/03/13/
s3://DOC-EXAMPLE-BUCKET/prefix/2021/02/03/14/
```

Demikian pula, jika kueri berjalan pada tanggal dan waktu sebelum 3 Februari 2021 pukul 15:00, partisi terakhir akan mencerminkan tanggal dan waktu saat ini, bukan tanggal dan waktu dalam kondisi kueri.

Jika Anda ingin meminta data terbaru, Anda dapat memanfaatkan fakta bahwa Athena tidak menghasilkan tanggal masa depan dan hanya menentukan awal`datehour`, seperti pada contoh berikut.

```
SELECT *
```

```
FROM my_ingested_data
WHERE datehour >= '2021/11/09/00'
```

Memilih kunci partisi

Anda dapat menentukan bagaimana proyeksi partisi memetakan lokasi partisi ke kunci partisi. Dalam `CREATE TABLE` contoh di bagian sebelumnya, tanggal dan jam digabungkan menjadi satu kunci partisi yang disebut datehour, tetapi skema lain dimungkinkan. Misalnya, Anda juga dapat mengonfigurasi tabel dengan kunci partisi terpisah untuk tahun, bulan, hari, dan jam.

Namun, membagi tanggal menjadi tahun, bulan, dan hari berarti bahwa jenis proyeksi `date` partisi tidak dapat digunakan. Alternatifnya adalah memisahkan tanggal dari jam untuk tetap memanfaatkan jenis proyeksi `date` partisi, tetapi membuat kueri yang menentukan rentang jam lebih mudah dibaca.

Dengan mengingat hal itu, `CREATE TABLE` contoh berikut memisahkan tanggal dari jam. Karena `date` adalah kata cadangan dalam SQL, contoh menggunakan `day` sebagai nama untuk kunci partisi yang mewakili tanggal.

```
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}/"
)
```

Dalam `CREATE TABLE` pernyataan contoh, jam adalah kunci partisi terpisah, dikonfigurasi sebagai bilangan bulat. Konfigurasi untuk kunci partisi jam menentukan rentang 0 hingga 23, dan jam harus diformat dengan dua digit saat Athena menghasilkan lokasi partisi.

Kueri untuk `my_ingested_data2` tabel mungkin terlihat seperti ini:

```
SELECT *
FROM my_ingested_data2
WHERE day = '2021/11/09'
AND hour > 3
```

Jenis kunci partisi dan jenis proyeksi partisi

Perhatikan bahwa `datehour` kunci dalam `CREATE TABLE` contoh pertama dikonfigurasi seperti `date` pada konfigurasi proyeksi partisi, tetapi jenis kunci partisi adalah`string`. Hal yang sama `day` berlaku untuk contoh kedua. Jenis dalam konfigurasi proyeksi partisi hanya memberi tahu Athena cara memformat nilai saat menghasilkan lokasi partisi. Jenis yang Anda tentukan tidak mengubah jenis kunci partisi — dalam kueri, `datehour` dan `day` bertipe`string`.

Ketika kueri menyertakan kondisi seperti`day = '2021/11/09'`, Athena mem-parsing string di sisi kanan ekspresi menggunakan format tanggal yang ditentukan dalam konfigurasi proyeksi partisi. Setelah Athena memverifikasi bahwa tanggal berada dalam rentang yang dikonfigurasi, ia menggunakan format tanggal lagi untuk memasukkan tanggal sebagai string ke dalam template lokasi penyimpanan.

Demikian pula, untuk kondisi kueri seperti`day > '2021/11/09'`, Athena mem-parsing sisi kanan dan menghasilkan daftar semua tanggal yang cocok dalam rentang yang dikonfigurasi. Kemudian menggunakan format tanggal untuk memasukkan setiap tanggal ke dalam template lokasi penyimpanan untuk membuat daftar lokasi partisi.

Menulis kondisi yang sama dengan `day > '2021-11-09'` atau `day > DATE '2021-11-09'` tidak berfungsi. Dalam kasus pertama, format tanggal tidak cocok (perhatikan tanda hubung alih-alih garis miring ke depan), dan dalam kasus kedua, tipe data tidak cocok.

Menggunakan awalan kustom dan partisi dinamis

Firehose dapat dikonfigurasi dengan awalan khusus dan partisi dinamis. Dengan menggunakan fitur-fitur ini, Anda dapat mengonfigurasi kunci Amazon S3 dan menyiapkan skema partisi yang lebih mendukung kasus penggunaan Anda. Anda juga dapat menggunakan proyeksi partisi dengan skema partisi ini dan mengkonfigurasinya sesuai dengan itu.

Misalnya, Anda dapat menggunakan fitur awalan khusus untuk mendapatkan kunci Amazon S3 yang memiliki tanggal berformat ISO, bukan skema default. `yyyy/MM/dd/HH`

Anda juga dapat menggabungkan awalan kustom dengan partisi dinamis untuk mengekstrak properti seperti dari pesan `customer_id` Firehose, seperti pada contoh berikut.

```
prefix/!{timestamp:yyyy}-!{timestamp:MM}-!{timestamp:dd}/!
{partitionKeyFromQuery:customer_id}/
```

Dengan awalan Amazon S3 itu, aliran pengiriman Firehose akan menulis objek ke kunci seperti. `s3://DOC-EXAMPLE-BUCKET/prefix/2021-11-01/customer-1234/file.extension` Untuk properti seperti`customer_id`, di mana nilainya mungkin tidak diketahui sebelumnya, Anda dapat menggunakan jenis proyeksi partisi `injected` dan menggunakan `CREATE TABLE` pernyataan seperti berikut:

```
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}/"
)
```

Saat Anda menanyakan tabel yang memiliki jenis kunci partisi`injected`, kueri Anda harus menyertakan nilai untuk kunci partisi tersebut. Kueri untuk `my_ingested_data3` tabel mungkin terlihat seperti ini:

```
SELECT *
```

```
FROM my_ingested_data3
WHERE day BETWEEN '2021-11-01' AND '2021-11-30'
AND customer_id = 'customer-1234'
```

Tanggal berformat ISO

Karena nilai untuk kunci `day` partisi diformat ISO, Anda juga dapat menggunakan `DATE` tipe untuk kunci partisi hari alih-alih`STRING`, seperti pada contoh berikut:

```
PARTITIONED BY (day DATE, customer_id STRING)
```

Saat Anda melakukan kueri, strategi ini memungkinkan Anda untuk menggunakan fungsi tanggal pada kunci partisi tanpa parsing atau casting, seperti pada contoh berikut:

```
SELECT *
FROM my_ingested_data3
WHERE day > CURRENT_DATE - INTERVAL '7' DAY
AND customer_id = 'customer-1234'
```

Note

Menentukan kunci partisi dari `DATE` jenis mengasumsikan bahwa Anda telah menggunakan fitur awalan khusus untuk membuat kunci Amazon S3 yang memiliki tanggal berformat ISO. Jika Anda menggunakan format Firehose default`yyyy/MM/dd/HH`, Anda harus menentukan kunci partisi sebagai tipe `string` meskipun properti tabel yang sesuai adalah tipe`date`, seperti pada contoh berikut:

```
PARTITIONED BY (
  `mydate` string)
TBLPROPERTIES (
  'projection.enabled'='true',
   ...
  'projection.mydate.type'='date',
  'storage.location.template'='s3://DOC-EXAMPLE-BUCKET/prefix/${mydate}')
```

Membuat tabel dari hasil query (CTAS)

`ACREATE TABLE AS SELECT`(CTAS) kueri menciptakan tabel baru di Athena dari hasil`SELECT`pernyataan dari kueri lain. Athena menyimpan file data yang dibuat oleh pernyataan CTAS di lokasi tertentu di Amazon S3. Untuk sintaks, lihat CREATE TABLE AS.

`CREATE TABLE AS`menggabungkan pernyataan `CREATE TABLE` DDL dengan pernyataan `SELECT` DHTML dan oleh karena itu secara teknis berisi DDL dan DML. Namun, perhatikan bahwa untuk tujuan Service Quotas, kueri CTAS di Athena diperlakukan sebagai DHTML. Untuk informasi tentang Service Quotas di Athena, lihat. Service Quotas

Gunakan kueri CTAS untuk:

- Membuat tabel dari hasil kueri dalam satu langkah, tanpa berulang kali kueri set data mentah. Ini memudahkan untuk bekerja dengan set data mentah.
- Mengubah hasil kueri dan memigrasikan tabel ke dalam format tabel lain seperti Apache Iceberg. Ini meningkatkan performa kueri dan mengurangi biaya permintaan di Athena. Untuk informasi, lihat Membuat tabel Iceberg.
- Ubah hasil kueri menjadi format penyimpanan seperti Parket dan ORC. Ini meningkatkan performa kueri dan mengurangi biaya permintaan di Athena. Untuk informasi, lihat Format penyimpanan kolumnar.
- Buat salinan tabel yang ada yang hanya berisi data yang Anda butuhkan.

Topik

- Pertimbangan dan batasan untuk kueri CTAS
- Menjalankan kueri CTAS di konsol
- Partisi dan bucketing di Athena
- Contoh kueri CTAS
- Menggunakan CTAS dan INSERT INTO untuk ETL dan analisis data
- Menggunakan CTAS dan INSERT INTO untuk bekerja di sekitar batas partisi 100

Pertimbangan dan batasan untuk kueri CTAS

Bagian berikut menjelaskan pertimbangan dan batasan yang perlu diingat saat Anda menggunakan kueri `CREATE TABLE AS SELECT` (CTAS) di Athena.

Kueri sintaks CTAS

Sintaks permintaan CTAS berbeda dari sintaks`CREATE [EXTERNAL] TABLE`digunakan untuk membuat tabel. Lihat CREATE TABLE AS.

CTAS kueri vs tampilan

kueri CTAS menulis data baru ke lokasi yang ditentukan di Amazon S3, sedangkan tampilan tidak menulis data apapun.

Lokasi hasil kueri CTAS

Jika grup kerja AndaMengabaikan pengaturan sisi klienuntuk lokasi hasil kueri, Athena membuat tabel Anda di lokasi`s3://DOC-EXAMPLE-BUCKET/tables/<query-id>/`. Untuk melihat lokasi hasil kueri yang ditentukan untuk grup kerja,lihat detail grup kerja.

Jika grup kerja Anda tidak menimpa lokasi hasil kueri, Anda dapat menggunakan sintaks`WITH (external_location ='s3://DOC-EXAMPLE-BUCKET/')`dalam permintaan CTAS Anda untuk menentukan tempat hasil kueri CTAS Anda disimpan.

> **Note**
>
> Parameter`external_location`properti harus menentukan lokasi yang kosong. Permintaan CTAS memeriksa bahwa lokasi jalur (awalan) dalam bucket kosong dan tidak pernah menimpa data jika lokasi sudah memiliki data di dalamnya. Untuk menggunakan lokasi yang sama lagi, hapus data di lokasi key prefix dalam bucket.

Jika Anda menghilangkan`external_location`sintaks dan tidak menggunakan pengaturan grup kerja, Athena menggunakanPengaturan sisi klienuntuk lokasi hasil kueri dan membuat tabel Anda di lokasi`s3://DOC-EXAMPLE-BUCKET/<Unsaved-or-query-name>/<year>/<month/<date>/tables/<query-id>/`.

Mencari File Yatim Piatu

Jika`CTAS`atau`INSERT INTO`pernyataan gagal, ada kemungkinan bahwa data yatim yang tersisa di lokasi data. Karena Athena dalam beberapa kasus tidak menghapus data atau sebagian data dari bucket Anda, Anda mungkin dapat membaca sebagian data ini di kueri berikutnya. Untuk menemukan file yatim piatu untuk pemeriksaan atau penghapusan, Anda dapat menggunakan file

manifest data yang disediakan Athena untuk melacak daftar file yang akan ditulis. Untuk informasi selengkapnya, lihat Mengidentifikasi file keluaran kueri dan DataManifestLocation.

Klausa ORDER BY diabaikan

Dalam kueri CTAS, Athena `ORDER BY` mengabaikan klausa di `SELECT` bagian kueri.

Menurut spesifikasi SQL (ISO 9075 Bagian 2), urutan baris tabel yang ditentukan oleh ekspresi kueri dijamin hanya untuk ekspresi kueri yang segera berisi klausa. `ORDER BY` Tabel dalam SQL dalam hal apa pun secara inheren tidak berurutan, dan mengimplementasikan klausa sub kueri `ORDER BY` in akan menyebabkan kueri berkinerja buruk dan tidak menghasilkan keluaran yang diurutkan. Dengan demikian, dalam kueri CTAS Athena, tidak ada jaminan bahwa pesanan yang ditentukan oleh `ORDER BY` klausa akan dipertahankan ketika data ditulis.

Format untuk menyimpan hasil kueri

Hasil kueri CTAS disimpan di Parquet secara default jika Anda tidak menentukan format penyimpanan data. Anda dapat menyimpan hasil CTAS di`PARQUET`,`ORC`,`AVRO`,`JSON`, dan`TEXTFILE`. Pembatas multi-karakter tidak didukung untuk CTAS`TEXTFILE`format. Kueri CTAS tidak memerlukan menentukan a SerDe untuk menafsirkan transformasi format. Lihat Example: Writing query results to a different format.

Format kompresi baru

`GZIP`kompresi digunakan untuk hasil query CTAS dalam format JSON dan TEXTFILE. Untuk Parket, Anda dapat menggunakan `GZIP` atau`SNAPPY`, dan defaultnya adalah`GZIP`. Untuk ORC, Anda dapat menggunakan`LZ4`,,, atau `SNAPPY` `ZLIBZSTD`, dan defaultnya adalah`ZLIB`. Untuk contoh CTAS yang menentukan kompresi, lihatExample: Specifying data storage and compression formats. Untuk informasi lebih lanjut tentang kompresi di Athena, lihat. Dukungan kompresi Athena

Batas partisi dan bucket

Anda dapat partisi dan bucket data hasil kueri CTAS. Untuk informasi selengkapnya, lihat Partisi dan bucketing di Athena. Saat membuat tabel yang dipartisi menggunakan CTAS, Athena memiliki batas penulisan 100 partisi.

Sertakan predikat partisi dan bucketing pada akhir`WITH`klausa yang menentukan properti dari tabel tujuan. Untuk informasi selengkapnya, lihat Example: Creating bucketed and partitioned tables dan Partisi dan bucketing di Athena.

Untuk informasi tentang bekerja di sekitar 100 partisi pembatasan, lihatMenggunakan CTAS dan INSERT INTO untuk bekerja di sekitar batas partisi 100.

Enkripsi

Anda dapat mengenkripsi hasil kueri CTAS di Amazon S3, mirip dengan cara Anda mengenkripsi hasil kueri lainnya di Athena. Untuk informasi selengkapnya, lihat Mengenkripsi hasil kueri Athena yang disimpan di Amazon S3.

Pemilik ember yang diharapkan

Untuk pernyataan CTAS, setelan pemilik bucket yang diharapkan tidak berlaku untuk lokasi tabel tujuan di Amazon S3. Setelan pemilik bucket yang diharapkan hanya berlaku untuk lokasi keluaran Amazon S3 yang Anda tentukan untuk hasil kueri Athena. Untuk informasi selengkapnya, lihat Menentukan lokasi hasil kueri menggunakan konsol Athena.

Jenis data

jenis data kolom untuk kueri CTAS adalah sama seperti yang ditentukan untuk kueri asli.

Menjalankan kueri CTAS di konsol

Di konsol Athena, Anda dapat membuat kueri CTAS dari kueri lain.

Untuk membuat kueri CTAS dari kueri lain

1. Jalankan kueri di editor kueri konsol Athena.
2. Di bagian bawah editor kueri, pilih opsi Buat, lalu pilih Tabel dari kueri.
3. Dalam Buat tabel sebagai formulir pilih, lengkapi bidang sebagai berikut:

    a. Untuk nama Tabel, masukkan nama untuk tabel baru Anda. Gunakan hanya huruf kecil dan garis bawah, sepertimy_select_query_parquet.

    b. Untuk konfigurasi Database, gunakan opsi untuk memilih database yang ada atau membuat database.

    c. (Opsional) Dalam konfigurasi Hasil, untuk hasil kueri Lokasi CTAS, jika pengaturan lokasi hasil kueri grup kerja Anda tidak mengganti opsi ini, lakukan salah satu hal berikut:

        - Masukkan jalur ke lokasi S3 yang ada di kotak pencarian, atau pilih Jelajahi S3 untuk memilih lokasi dari daftar.

- Pilih Lihat untuk membuka halaman Bucket di konsol Amazon S3 tempat Anda dapat melihat informasi selengkapnya tentang bucket yang ada dan memilih atau membuat bucket dengan pengaturan Anda sendiri.

Anda harus menentukan lokasi kosong di Amazon S3 di mana data akan dikeluarkan. Jika data sudah ada di lokasi yang Anda tentukan, kueri gagal dengan kesalahan.

Jika pengaturan lokasi hasil kueri grup kerja Anda mengesampingkan setelan lokasi ini, Athena akan membuat tabel Anda di lokasi `s3://DOC-EXAMPLE-BUCKET/tables/query_id/`

d. Untuk format Data, tentukan format data Anda.

- Jenis tabel - Jenis tabel default di Athena adalah Apache Hive.
- Format file - Pilih di antara opsi seperti CSV, TSV, JSON, Parket, atau ORC. Untuk informasi tentang format Parket dan ORC, lihat. Format penyimpanan kolumnar
- Tulis kompresi - (Opsional) Pilih format kompresi. Athena mendukung berbagai format kompresi untuk membaca dan menulis data, termasuk membaca dari tabel yang menggunakan beberapa format kompresi. Misalnya, Athena berhasil membaca data dalam tabel yang menggunakan format file Parket ketika beberapa file Parket dikompresi dengan Snappy dan file Parket lainnya dikompresi dengan GZIP. Prinsip yang sama berlaku untuk ORC, file teks, dan format penyimpanan JSON. Untuk informasi selengkapnya, lihat Dukungan kompresi Athena.
- Partisi - (Opsional) Pilih kolom yang ingin Anda partisi. Mempartisi data Anda membatasi jumlah data yang dipindai oleh setiap kueri, sehingga meningkatkan kinerja dan mengurangi biaya. Anda dapat mempartisi data Anda dengan kunci apa pun. Untuk informasi selengkapnya, lihat Partisi data di Athena.
- Bucket - (Opsional) Pilih kolom yang ingin Anda ember. Bucketing adalah teknik yang mengelompokkan data berdasarkan kolom tertentu bersama-sama dalam satu partisi. Kolom ini dikenal sebagai kunci ember. Dengan mengelompokkan data terkait ke dalam satu bucket (file dalam partisi), Anda secara signifikan mengurangi jumlah data yang dipindai oleh Athena, sehingga meningkatkan kinerja kueri dan mengurangi biaya. Untuk informasi selengkapnya, lihat Partisi dan bucketing di Athena.

e. Untuk kueri tabel Pratinjau, tinjau kueri Anda. Untuk sintaks kueri, lihatCREATE TABLE AS.

f. Pilih Buat tabel.

Untuk membuat kueri CTAS menggunakan template SQL

Gunakan `CREATE TABLE AS SELECT` template untuk membuat kueri CTAS di editor kueri.

1. Di konsol Athena, di samping Tabel dan tampilan, pilih Buat tabel, lalu pilih CREATE TABLE AS SELECT. Ini mengisi editor kueri dengan kueri CTAS dengan nilai placeholder.
2. Di editor kueri, edit kueri sesuai kebutuhan. Untuk sintaks kueri, lihatCREATE TABLE AS.
3. Pilih Jalankan.

Sebagai contoh, lihat Contoh kueri CTAS.

Partisi dan bucketing di Athena

Partisi dan bucketing adalah dua cara untuk mengurangi jumlah data Athena harus memindai ketika Anda menjalankan kueri. Partisi dan bucketing saling melengkapi dan dapat digunakan bersama. Mengurangi jumlah data yang dipindai mengarah pada peningkatan kinerja dan biaya yang lebih rendah. Untuk panduan umum tentang performa kueri Athena, lihat 10 kiat penyetelan kinerja teratas untuk Amazon Athena.

Apa itu partisi?

Partisi berarti mengatur data ke dalam direktori (atau “awalan”) di Amazon S3 berdasarkan properti data tertentu. Properti seperti itu disebut kunci partisi. Kunci partisi umum adalah tanggal atau satuan waktu lainnya seperti tahun atau bulan. Namun, dataset dapat dipartisi oleh lebih dari satu kunci. Misalnya, data tentang penjualan produk dapat dipartisi berdasarkan tanggal, kategori produk, dan pasar.

Memutuskan cara partisi

Kandidat yang baik untuk kunci partisi adalah properti yang selalu atau sering digunakan dalam kueri dan memiliki kardinalitas rendah. Ada trade-off antara memiliki terlalu banyak partisi dan memiliki terlalu sedikit. Dengan terlalu banyak partisi, peningkatan jumlah file menciptakan overhead. Ada juga beberapa overhead dari penyaringan partisi itu sendiri. Dengan terlalu sedikit partisi, kueri seringkali harus memindai lebih banyak data.

Membuat tabel yang dipartisi

Saat kumpulan data dipartisi, Anda dapat membuat tabel yang dipartisi di Athena. Tabel yang dipartisi adalah tabel yang memiliki kunci partisi. Saat Anda menggunakan`CREATE TABLE`, Anda

menambahkan partisi ke tabel. Saat Anda menggunakan`CREATE TABLE AS`, partisi yang dibuat di Amazon S3 secara otomatis ditambahkan ke tabel.

Dalam sebuah `CREATE TABLE` pernyataan, Anda menentukan kunci partisi dalam `PARTITIONED BY (`*`column_name data_type`*`)` klausa. Dalam sebuah `CREATE TABLE AS` pernyataan, Anda menentukan kunci partisi dalam `WITH (partitioned_by = ARRAY['`*`partition_key`*`'])` klausa, atau `WITH (partitioning = ARRAY['`*`partition_key`*`'])` untuk tabel Iceberg. Untuk alasan kinerja, kunci partisi harus selalu bertipe`STRING`. Untuk informasi selengkapnya, lihat Gunakan string sebagai tipe data untuk kunci partisi.

Untuk detail tambahan `CREATE TABLE` dan `CREATE TABLE AS` sintaks, lihat CREATE TABLE dan Properti tabel CTAS.

Menanyakan tabel yang dipartisi

Saat Anda menanyakan tabel yang dipartisi, Athena menggunakan predikat dalam kueri untuk memfilter daftar partisi. Kemudian menggunakan lokasi partisi yang cocok untuk memproses file yang ditemukan. Athena dapat secara efisien mengurangi jumlah data yang dipindai hanya dengan tidak membaca data di partisi yang tidak cocok dengan predikat kueri.

Contoh

Misalkan Anda memiliki tabel yang dipartisi oleh `sales_date` `product_category` dan dan ingin mengetahui total pendapatan selama seminggu dalam kategori tertentu. Anda menyertakan `sales_date` predikat pada `product_category` kolom dan untuk memastikan bahwa Athena hanya memindai jumlah minimum data, seperti pada contoh berikut.

```
SELECT SUM(amount) AS total_revenue
FROM sales
WHERE sales_date BETWEEN '2023-02-27' AND '2023-03-05'
AND product_category = 'Toys'
```

Misalkan Anda memiliki kumpulan data yang dipartisi berdasarkan tanggal tetapi juga memiliki stempel waktu berbutir halus.

Dengan tabel Iceberg, Anda dapat mendeklarasikan kunci partisi untuk memiliki hubungan dengan kolom, tetapi dengan tabel Hive mesin kueri tidak memiliki pengetahuan tentang hubungan antara kolom dan kunci partisi. Untuk alasan ini, Anda harus menyertakan predikat pada kolom dan kunci partisi dalam kueri Anda untuk memastikan kueri tidak memindai lebih banyak data daripada yang diperlukan.

Misalnya, `sales` tabel pada contoh sebelumnya juga memiliki `sold_at` kolom tipe `TIMESTAMP` data. Jika Anda menginginkan pendapatan hanya untuk rentang waktu tertentu, Anda akan menulis kueri seperti ini:

```
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'
```

Untuk informasi selengkapnya tentang perbedaan antara menanyakan tabel Hive dan Iceberg, lihat. [Cara menulis kueri untuk bidang stempel waktu yang juga dipartisi waktu]

Apa itu bucketing?

Bucketing adalah cara untuk mengatur catatan dataset ke dalam kategori yang disebut bucket.

Arti bucket dan bucketing ini berbeda dari, dan jangan bingung dengan, ember Amazon S3. Dalam bucketing data, catatan yang memiliki nilai yang sama untuk properti masuk ke bucket yang sama. Catatan didistribusikan secara merata di antara ember sehingga setiap bucket memiliki jumlah data yang kira-kira sama.

Dalam praktiknya, bucket adalah file, dan fungsi hash menentukan bucket tempat rekaman masuk. Dataset berember akan memiliki satu atau lebih file per ember per partisi. Bucket yang dimiliki file dikodekan dalam nama file.

Manfaat Bucketing

Bucketing berguna ketika kumpulan data diselimuti oleh properti tertentu dan Anda ingin mengambil catatan di mana properti tersebut memiliki nilai tertentu. Karena datanya diselimuti, Athena dapat menggunakan nilainya untuk menentukan file mana yang akan dilihat. Misalnya, dataset diselimuti oleh `customer_id` dan Anda ingin menemukan semua catatan untuk pelanggan tertentu. Athena menentukan ember yang berisi catatan itu dan hanya membaca file di ember itu.

Kandidat yang baik untuk bucketing terjadi ketika Anda memiliki kolom yang memiliki kardinalitas tinggi (yaitu, memiliki banyak nilai berbeda), terdistribusi secara seragam, dan Anda sering meminta nilai tertentu.

> **Note**
>
> Athena tidak mendukung penggunaan `INSERT INTO` untuk menambahkan catatan baru ke tabel berember.

Tipe data yang didukung untuk pemfilteran pada kolom berember

Anda dapat menambahkan filter pada kolom berember dengan tipe data tertentu. Athena mendukung penyaringan pada kolom berember dengan tipe data berikut:

- BOOLEAN
- BYTE
- DATE
- DOUBLE
- FLOAT
- INT
- LONG
- SHORT
- STRING
- VARCHAR

Dukungan Hive dan Spark

Mesin Athena versi 2 mendukung kumpulan data yang diselimuti menggunakan algoritma bucket Hive, dan mesin Athena versi 3 juga mendukung algoritma bucketing Apache Spark. Bucketing sarang adalah default. Jika kumpulan data Anda diselimuti menggunakan algoritma Spark, gunakan `TBLPROPERTIES` klausa untuk menyetel nilai properti. `bucketing_format spark`

> **Note**
>
> Athena memiliki batas 100 partisi dalam kueri `CREATE TABLE AS SELECT` (CTAS). Demikian pula, Anda hanya dapat menambahkan maksimum 100 partisi ke tabel tujuan dengan INSERT INTO pernyataan. Batas 100 ini hanya berlaku ketika tabel diselimuti dan dipartisi.

Jika Anda melebihi batasan ini, Anda mungkin menerima pesan kesalahan HIVE_TOO_MANY_OPEN_PARTITIONS: Melebihi batas 100 penulis terbuka untuk partisi/ember. Untuk mengatasi batasan ini, Anda dapat menggunakan pernyataan CTAS dan serangkaian `INSERT INTO` pernyataan yang membuat atau menyisipkan hingga 100 partisi masing-masing. Untuk informasi selengkapnya, lihat Menggunakan CTAS dan INSERT INTO untuk bekerja di sekitar batas partisi 100.

Bucketing CREATE TABLE contoh

Untuk membuat tabel untuk kumpulan data berember yang ada, gunakan klausa yang diikuti oleh `CLUSTERED BY (`*`column`*`)` klausa. `INTO` *`N`* `BUCKETS INTO` *`N`* `BUCKETS`Klausa menentukan jumlah ember yang dimasukkan ke dalam data.

Dalam `CREATE TABLE` contoh berikut, `sales` kumpulan data dimasukkan `customer_id` ke dalam 8 ember menggunakan algoritma Spark. `CREATE TABLE`Pernyataan menggunakan `TBLPROPERTIES` klausa `CLUSTERED BY` and untuk mengatur properti yang sesuai.

```
CREATE EXTERNAL TABLE sales (...)
...
CLUSTERED BY (`customer_id`) INTO 8 BUCKETS
...
TBLPROPERTIES (
  'bucketing_format' = 'spark'
)
```

Bucketing CREATE TABLE AS (CTAS) contoh

Untuk menentukan ember dengan`CREATE TABLE AS`, gunakan `bucket_count` parameter `bucketed_by` dan, seperti pada contoh berikut.

```
CREATE TABLE sales
WITH (
  ...
  bucketed_by = ARRAY['customer_id'],
  bucket_count = 8
)
AS SELECT ...
```

Contoh kueri bucketing

Contoh kueri berikut mencari nama-nama produk yang dibeli pelanggan tertentu selama seminggu.

```
SELECT DISTINCT product_name
FROM sales
WHERE sales_date BETWEEN '2023-02-27' AND '2023-03-05'
AND customer_id = 'c123'
```

Jika tabel ini dipartisi oleh `sales_date` dan dikantongi oleh`customer_id`, Athena dapat menghitung ember tempat catatan pelanggan berada. Paling-paling, Athena membaca satu file per partisi.

Sumber daya tambahan

- Untuk `CREATE TABLE AS` contoh yang membuat tabel berember dan dipartisi, lihat Contoh: Membuat tabel berember dan dipartisi.
- Untuk informasi tentang penerapan bucketing pada AWS data lake, termasuk menggunakan pernyataan Athena CTAS, AWS Glue untuk Apache Spark, dan bucketing untuk tabel Apache Iceberg, lihat posting AWS Blog Big Data Optimalkan tata letak data dengan menggunakan Amazon Athena dan untuk mempercepat kueri hilir. AWS Glue

Contoh kueri CTAS

Gunakan contoh berikut untuk membuat permintaan CTAS. Untuk informasi tentang sintaks aturan, lihat CREATE TABLE AS.

Di bagian ini:

- Example: Duplicating a table by selecting all columns
- Example: Selecting specific columns from one or more tables
- Example: Creating an empty copy of an existing table
- Example: Specifying data storage and compression formats
- Example: Writing query results to a different format
- Example: Creating unpartitioned tables
- Example: Creating partitioned tables
- Example: Creating bucketed and partitioned tables

- Example: Creating an Iceberg table with Parquet data
- Example: Creating an Iceberg table with Avro data

Example Contoh: Menduplikasi tabel dengan memilih semua kolom

Contoh berikut membuat tabel dengan menyalin semua kolom dari tabel:

```
CREATE TABLE new_table AS
SELECT *
FROM old_table;
```

Dalam variasi berikut dari contoh yang sama, Anda`SELECT`pernyataan juga mencakup`WHERE`Klausul. Dalam kasus ini, kueri memilih hanya baris dari tabel yang memenuhi`WHERE`Klausul:

```
CREATE TABLE new_table AS
SELECT *
FROM old_table
WHERE condition;
```

Example Contoh: Memilih kolom tertentu dari satu atau lebih tabel

Contoh berikut membuat kueri baru yang berjalan pada satu set kolom dari tabel lain:

```
CREATE TABLE new_table AS
SELECT column_1, column_2, ... column_n
FROM old_table;
```

Variasi ini dari contoh yang sama menciptakan tabel baru dari kolom tertentu dari beberapa tabel:

```
CREATE TABLE new_table AS
SELECT column_1, column_2, ... column_n
FROM old_table_1, old_table_2, ... old_table_n;
```

Example Contoh: Membuat salinan kosong dari tabel yang ada

Contoh berikut menggunakan`WITH NO DATA`untuk membuat tabel baru yang kosong dan memiliki skema yang sama seperti tabel asli:

```
CREATE TABLE new_table
```

```
AS SELECT *
FROM old_table
WITH NO DATA;
```

Example Contoh: Menentukan penyimpanan data dan format kompresi

Dengan CTAS, Anda dapat menggunakan tabel sumber dalam satu format penyimpanan untuk membuat tabel lain dalam format penyimpanan yang berbeda.

Gunakan `format` properti untuk menentukan`ORC`,`PARQUET`,`AVRO`,`JSON`, atau `TEXTFILE` sebagai format penyimpanan untuk tabel baru.

Untuk format `PARQUET`ORC,`TEXTFILE`,, dan `JSON` penyimpanan, gunakan `write_compression` properti untuk menentukan format kompresi untuk data tabel baru. Untuk informasi tentang format kompresi yang didukung oleh setiap format file, lihatDukungan kompresi Athena.

Contoh berikut menentukan bahwa data dalam tabel `new_table` disimpan dalam format Parket dan menggunakan kompresi Snappy. Kompresi default untuk Parket adalah`GZIP`.

```
CREATE TABLE new_table
WITH (
      format = 'Parquet',
      write_compression = 'SNAPPY')
AS SELECT *
FROM old_table;
```

Contoh berikut menentukan bahwa data dalam tabel `new_table` disimpan dalam format ORC menggunakan kompresi Snappy. Kompresi default untuk ORC adalah ZLIB.

```
CREATE TABLE new_table
WITH (format = 'ORC',
      write_compression = 'SNAPPY')
AS SELECT *
FROM old_table ;
```

Contoh berikut menentukan bahwa data dalam tabel `new_table` disimpan dalam format textfile menggunakan kompresi Snappy. Kompresi default untuk format textfile dan JSON adalah GZIP.

```
CREATE TABLE new_table
WITH (format = 'TEXTFILE',
      write_compression = 'SNAPPY')
```

```
AS SELECT *
FROM old_table ;
```

Example Contoh: Menulis hasil kueri ke format yang berbeda

Kueri CTAS berikut memilih semua catatan dario`ld_table`, yang dapat disimpan dalam CSV atau format lain, dan membuat tabel baru dengan data dasar yang disimpan ke Amazon S3 dalam format ORC:

```
CREATE TABLE my_orc_ctas_table
WITH (
      external_location = 's3://DOC-EXAMPLE-BUCKET/my_orc_stas_table/',
      format = 'ORC')
AS SELECT *
FROM old_table;
```

Example Contoh: Membuat tabel yang tidak dipartisi

Contoh berikut membuat tabel yang tidak dipartisi. Data tabel disimpan dalam format yang berbeda. Beberapa contoh ini menentukan lokasi eksternal.

Contoh berikut membuat kueri CTAS yang menyimpan hasil sebagai file teks:

```
CREATE TABLE ctas_csv_unpartitioned
WITH (
     format = 'TEXTFILE',
     external_location = 's3://DOC-EXAMPLE-BUCKET/ctas_csv_unpartitioned/')
AS SELECT key1, name1, address1, comment1
FROM table1;
```

Pada contoh berikut, hasil disimpan di Parquet, dan lokasi hasil default digunakan:

```
CREATE TABLE ctas_parquet_unpartitioned
WITH (format = 'PARQUET')
AS SELECT key1, name1, comment1
FROM table1;
```

Dalam kueri berikut, tabel disimpan dalam JSON, dan kolom tertentu dipilih dari hasil tabel asli ini:

```
CREATE TABLE ctas_json_unpartitioned
WITH (
```

```
     format = 'JSON',
     external_location = 's3://DOC-EXAMPLE-BUCKET/ctas_json_unpartitioned/')
AS SELECT key1, name1, address1, comment1
FROM table1;
```

Berikut ini adalah contoh file .

```
CREATE TABLE ctas_orc_unpartitioned
WITH (
     format = 'ORC')
AS SELECT key1, name1, comment1
FROM table1;
```

Berikut ini adalah contoh file .

```
CREATE TABLE ctas_avro_unpartitioned
WITH (
     format = 'AVRO',
     external_location = 's3://DOC-EXAMPLE-BUCKET/ctas_avro_unpartitioned/')
AS SELECT key1, name1, comment1
FROM table1;
```

Example Contoh: Membuat tabel yang dipartisi

Contoh berikut menunjukkan`CREATE TABLE AS SELECT`query untuk tabel dipartisi dalam format penyimpanan yang berbeda, menggunakan`partitioned_by`, dan properti lainnya di`WITH`Klausul. Untuk sintaks, lihatProperti tabel CTAS. Untuk informasi selengkapnya tentang memilih kolom untuk partisi, lihatPartisi dan bucketing di Athena.

> **Note**
>
> Daftar kolom partisi pada akhir daftar kolom di`SELECT`. Anda dapat mempartisi dengan lebih dari satu kolom, dan memiliki hingga 100 partisi unik dan kombinasi bucket. Misalnya, Anda dapat memiliki 100 partisi jika tidak ada bucket yang ditentukan.

```
CREATE TABLE ctas_csv_partitioned
WITH (
     format = 'TEXTFILE',
     external_location = 's3://DOC-EXAMPLE-BUCKET/ctas_csv_partitioned/',
```

```
     partitioned_by = ARRAY['key1'])
AS SELECT name1, address1, comment1, key1
FROM tables1;
```

```
CREATE TABLE ctas_json_partitioned
WITH (
     format = 'JSON',
     external_location = 's3://DOC-EXAMPLE-BUCKET/ctas_json_partitioned/',
     partitioned_by = ARRAY['key1'])
AS select name1, address1, comment1, key1
FROM table1;
```

Example Contoh: Membuat tabel berember dan dipartisi

Contoh berikut menunjukkan`CREATE TABLE AS SELECT`query yang menggunakan partisi dan bucketing untuk menyimpan hasil kueri di Amazon S3. Hasil tabel dipartisi dan bucketed oleh kolom yang berbeda. Athena mendukung maksimal 100 kombinasi bucket dan partisi yang unik. Misalnya, jika Anda membuat tabel dengan lima bucket, 20 partisi dengan lima bucket masing-masing didukung. Untuk sintaks, lihatProperti tabel CTAS.

Untuk informasi tentang memilih kolom untuk bucketing, lihatPartisi dan bucketing di Athena.

```
CREATE TABLE ctas_avro_bucketed
WITH (
      format = 'AVRO',
      external_location = 's3://DOC-EXAMPLE-BUCKET/ctas_avro_bucketed/',
      partitioned_by = ARRAY['nationkey'],
      bucketed_by = ARRAY['mktsegment'],
      bucket_count = 3)
AS SELECT key1, name1, address1, phone1, acctbal, mktsegment, comment1, nationkey
FROM table1;
```

Example Contoh: Membuat tabel Gunung Es dengan data Parket

Contoh berikut membuat tabel Iceberg dengan file data Parket. File dipartisi berdasarkan bulan menggunakan `dt` kolom di. `table1` Contoh memperbarui properti retensi pada tabel sehingga 10 snapshot dipertahankan secara default pada setiap cabang dalam tabel. Snapshot dalam 7 hari terakhir juga dipertahankan. Untuk informasi lebih lanjut tentang properti tabel Gunung Es di Athena, lihat. Properti tabel

```
CREATE TABLE ctas_iceberg_parquet
```

```
WITH (table_type = 'ICEBERG',
      format = 'PARQUET',
      location = 's3://DOC-EXAMPLE-BUCKET/ctas_iceberg_parquet/',
      is_external = false,
      partitioning = ARRAY['month(dt)'],
      vacuum_min_snapshots_to_keep = 10,
      vacuum_max_snapshot_age_seconds = 604800
   )
AS SELECT key1, name1, dt FROM table1;
```

Example Contoh: Membuat tabel Iceberg dengan data Avro

Contoh berikut membuat tabel Iceberg dengan file data Avro dipartisi oleh. `key1`

```
CREATE TABLE ctas_iceberg_avro
WITH ( format = 'AVRO',
       location = 's3://DOC-EXAMPLE-BUCKET/ctas_iceberg_avro/',
       is_external = false,
       table_type = 'ICEBERG',
       partitioning = ARRAY['key1'])
AS SELECT key1, name1, date FROM table1;
```

Menggunakan CTAS dan INSERT INTO untuk ETL dan analisis data

Anda dapat menggunakan pernyataan Create Table as Select (CTAS) dan INSERT INTO di Athena untuk extract, transform, and load (ETL) data ke Amazon S3 untuk pemrosesan data. Topik ini menunjukkan cara menggunakan pernyataan ini untuk partisi dan mengkonversi set data ke format data columnar untuk mengoptimalkan untuk analisis data.

Pernyataan CTAS menggunakan kueri SELECT standar untuk membuat tabel baru. Anda dapat menggunakan pernyataan CTAS untuk membuat subset dari data Anda untuk analisis. Dalam satu pernyataan CTAS, Anda dapat partisi data, menentukan kompresi, dan mengkonversi data ke dalam format columnar seperti Apache Parquet atau Apache ORC. Saat Anda menjalankan permintaan CTAS, tabel dan partisi yang dibuat secara otomatis ditambahkan keAWS Glue Data Catalog. Ini membuat tabel baru dan partisi yang menciptakan segera tersedia untuk kueri berikutnya.

INSERT INTO pernyataan memasukkan baris baru ke dalam tabel tujuan berdasarkan pernyataan kueri SELECT yang berjalan pada tabel sumber. Anda dapat menggunakan INSERT INTO pernyataan untuk mengubah dan memuat data tabel sumber dalam format CSV ke data tabel tujuan menggunakan semua transformasi yang didukung CTAS.

Gambaran Umum

Di Athena, menggunakan pernyataan CTAS untuk melakukan konversi batch awal data. Kemudian gunakan beberapa INSERT INTO pernyataan untuk membuat update inkremental ke tabel yang dibuat oleh pernyataan CTAS.

Langkah-langkah

- Langkah 1: Buat tabel berdasarkan dataset asli
- Langkah 2: Gunakan CTAS untuk mempartisi, mengonversi, dan mengompres data
- Langkah 3: Gunakan INSERT INTO untuk menambahkan data
- Langkah 4: Ukur perbedaan kinerja dan biaya

Langkah 1: Buat tabel berdasarkan dataset asli

Contoh dalam topik ini menggunakan subset Amazon S3 yang dapat dibaca dari kumpulan data harian jaringan klimatologi historis global NOAA (GHCN-D) yang tersedia untuk umum. Data di Amazon S3 memiliki karakteristik sebagai berikut.

```
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
```

Data asli disimpan di Amazon S3 tanpa partisi. Data dalam format CSV dalam fail seperti berikut.

```
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
```

Ukuran file dalam sampel ini relatif kecil. Dengan menggabungkan mereka ke dalam file yang lebih besar, Anda dapat mengurangi jumlah total file, memungkinkan performa kueri yang lebih baik. Anda dapat menggunakan CTAS dan INSERT INTO pernyataan untuk meningkatkan performa kueri.

Untuk membuat basis data dan tabel berdasarkan set data sampel

1. Di konsol Athena, pilih US East (Virginia N.). Wilayah AWS Pastikan untuk menjalankan semua pertanyaan dalam tutorial ini di`us-east-1`.
2. Dalam editor permintaan Athena, jalankan perintahBUAT BASIS DATAUntuk membuat basis data.

```
CREATE DATABASE blogdb
```

3. Jalankan pernyataan berikut untukUntuk membuat tabel.

```
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/'
```

Langkah 2: Gunakan CTAS untuk mempartisi, mengonversi, dan mengompres data

Setelah Anda membuat tabel, Anda dapat menggunakanCTASuntuk mengkonversi data ke format Parquet dengan kompresi Snappy dan partisi data per tahun.

Tabel yang Anda buat di Langkah 1 memilikidatelapangan dengan tanggal diformat sebagaiYYYYMMDD(misalnya,`20100104`). Karena tabel baru akan dipartisi padayear, contoh pernyataan dalam prosedur berikut menggunakan fungsi Prestosubstr("date",1,4)untuk mengekstrakyearnilai daridateBidang.

Untuk mengonversi data ke format parket dengan kompresi tajam, partisi berdasarkan tahun

- Menjalankan pernyataan CTAS berikut, mengganti*bucket Anda*dengan lokasi bucket Amazon S3.

```
CREATE table new_parquet
WITH (format='PARQUET',
parquet_compression='SNAPPY',
partitioned_by=array['year'],
external_location = 's3://DOC-EXAMPLE-BUCKET/optimized-data/')
AS
SELECT id,
         date,
         element,
         datavalue,
         mflag,
         qflag,
         sflag,
         obstime,
         substr("date",1,4) AS year
FROM original_csv
WHERE cast(substr("date",1,4) AS bigint) >= 2015
        AND cast(substr("date",1,4) AS bigint) <= 2019
```

> **Note**
>
> Dalam contoh ini, tabel yang Anda buat hanya mencakup data dari 2015 hingga 2019. Pada Langkah 3, Anda menambahkan data baru ke tabel ini menggunakan INSERT INTO perintah.

Saat permintaan selesai, gunakan prosedur berikut untuk memverifikasi output di lokasi Amazon S3 yang Anda tentukan dalam pernyataan CTAS.

Untuk melihat partisi dan file Parquet yang dibuat oleh pernyataan CTAS

1. Untuk menampilkan partisi yang dibuat, jalankan AWS CLI perintah berikut. Pastikan untuk menyertakan garis miring ke depan akhir (/).

```
aws s3 ls s3://DOC-EXAMPLE-BUCKET/optimized-data/
```

Output menunjukkan partisi.

```
      PRE year=2015/
      PRE year=2016/
      PRE year=2017/
      PRE year=2018/
      PRE year=2019/
```

2. Untuk melihat file Parquet, jalankan perintah berikut. Perhatikan bahwa opsi | head-5, yang membatasi output untuk lima hasil pertama, tidak tersedia pada Windows.

```
aws s3 ls s3://DOC-EXAMPLE-BUCKET/optimized-data/ --recursive --human-readable |
 head -5
```

Output menyerupai berikut.

```
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
```

Langkah 3: Gunakan INSERT INTO untuk menambahkan data

Pada langkah 2, Anda menggunakan CTAS untuk membuat tabel dengan partisi untuk tahun 2015 hingga 2019. Namun, set data asli juga berisi data untuk tahun 2010 hingga 2014. Sekarang Anda menambahkan data yang menggunakanSISIPAN KE.

Untuk menambahkan data ke tabel menggunakan satu atau lebih INSERT INTO pernyataan

1. Jalankan perintah INSERT INTO berikut, menentukan tahun sebelum 2015 di klausa WHERE.

```
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. Jalankan`aws s3 ls`Perintah lagi, menggunakan sintaks berikut.

```
aws s3 ls s3://DOC-EXAMPLE-BUCKET/optimized-data/
```

Output menunjukkan partisi baru.

```
      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. Untuk melihat pengurangan ukuran set data yang diperoleh dengan menggunakan kompresi dan penyimpanan kolumnar dalam format Parquet, jalankan perintah berikut.

```
aws s3 ls s3://DOC-EXAMPLE-BUCKET/optimized-data/ --recursive --human-readable --
summarize
```

Hasil berikut menunjukkan bahwa ukuran set data setelah Parquet dengan kompresi Snappy adalah 1,2 GB.

```
...
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. Jika lebih banyak data CSV ditambahkan ke tabel asli, Anda dapat menambahkan data tersebut ke tabel Parquet dengan menggunakan INSERT INTO pernyataan. Sebagai contoh, jika Anda memiliki data baru untuk tahun 2020, Anda bisa menjalankan berikut INSERT INTO pernyataan. Pernyataan itu menambahkan data dan partisi yang relevan untuk`new_parquet`Tabel.

```
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**
>
> Pernyataan INSERT INTO mendukung menulis maksimal 100 partisi ke tabel tujuan. Namun, untuk menambahkan lebih dari 100 partisi, Anda dapat menjalankan beberapa INSERT INTO pernyataan. Untuk informasi selengkapnya, lihat Menggunakan CTAS dan INSERT INTO untuk bekerja di sekitar batas partisi 100.

Langkah 4: Ukur perbedaan kinerja dan biaya

Setelah Anda mengubah data, Anda dapat mengukur performa keuntungan dan penghematan biaya dengan menjalankan kueri yang sama pada tabel baru dan lama dan membandingkan hasil.

> **Note**
>
> Untuk informasi biaya per permintaan Athena, lihatHarga Amazon Athena.

Untuk mengukur keuntungan performa dan perbedaan biaya

1. Menjalankan kueri berikut pada tabel asli. kueri menemukan jumlah ID yang berbeda untuk setiap nilai tahun.

```
SELECT substr("date",1,4) as year,
       COUNT(DISTINCT id)
FROM original_csv
GROUP BY 1 ORDER BY 1 DESC
```

2. Perhatikan waktu yang permintaan berlari dan jumlah data yang dipindai.
3. Menjalankan kueri yang sama pada tabel baru, mencatat runtime kueri dan jumlah data yang dipindai.

```
SELECT year,
  COUNT(DISTINCT id)
FROM new_parquet
GROUP BY 1 ORDER BY 1 DESC
```

4. Bandingkan hasilnya dan hitung perbedaan performa dan biaya. Hasil contoh berikut menunjukkan bahwa permintaan tes di tabel baru lebih cepat dan lebih murah daripada permintaan di tabel tua.

Tabel	Waktu Aktif	Data dipindai
Asal	16,88 detik	11,35 GB
Baru	3,79 detik	482,05 MB

5. Menjalankan kueri contoh berikut pada tabel asli. Kueri menghitung suhu maksimum rata-rata (Celcius), suhu minimum rata-rata (Celcius), dan curah hujan rata-rata (mm) untuk Bumi pada tahun 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. Perhatikan waktu yang permintaan berlari dan jumlah data yang dipindai.

7. Menjalankan kueri yang sama pada tabel baru, mencatat runtime kueri dan jumlah data yang dipindai.

```
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. Bandingkan hasilnya dan hitung perbedaan performa dan biaya. Hasil contoh berikut menunjukkan bahwa permintaan tes di tabel baru lebih cepat dan lebih murah daripada permintaan di tabel tua.

Tabel	Waktu Aktif	Data dipindai
Asal	18,65 detik	11,35 GB
Baru	1,92 detik	68 MB

Ringkasan

Topik ini menunjukkan cara untuk melakukan operasi ETL menggunakan CTAS dan INSERT INTO pernyataan di Athena. Anda melakukan rangkaian transformasi pertama menggunakan pernyataan CTAS yang mengubah data ke format Parquet dengan kompresi Snappy. Pernyataan CTAS juga dikonversi set data dari non-dipartisi untuk dipartisi. Ini mengurangi ukurannya dan menurunkan biaya menjalankan kueri. Saat data baru menjadi tersedia, Anda dapat menggunakan INSERT INTO pernyataan untuk mengubah dan memuat data ke dalam tabel yang Anda buat dengan pernyataan CTAS.

Menggunakan CTAS dan INSERT INTO untuk bekerja di sekitar batas partisi 100

Athena memiliki batas 100 partisi per `CREATE TABLE AS SELECT` (CTAS) kueri. Demikian pula, Anda dapat menambahkan maksimal 100 partisi ke tabel tujuan dengan SISIPAN KE.

Jika Anda melebihi batasan ini, Anda mungkin menerima pesan kesalahan HIVE_TOO_MANY_OPEN_PARTITIONS: Melebihi batas 100 penulis terbuka untuk partisi/ember. Untuk mengatasi batasan ini, Anda dapat menggunakan pernyataan CTAS dan serangkaian `INSERT INTO` pernyataan yang membuat atau menyisipkan hingga 100 partisi masing-masing.

Contoh dalam topik ini menggunakan database bernama `tpch100` yang datanya berada di lokasi bucket Amazon S3 s3://DOC-EXAMPLE-BUCKET/.

Untuk menggunakan CTAS dan INSERT INTO untuk membuat tabel lebih dari 100 partisi

1. Menggunakan`CREATE EXTERNAL TABLE`pernyataan untuk membuat tabel dipartisi pada bidang yang Anda inginkan.

   Contoh pernyataan berikut partisi data dengan kolom`l_shipdate`. Tabel ini memiliki 2525 partisi.

   ```
   CREATE EXTERNAL TABLE `tpch100.lineitem_parq_partitioned`(
     `l_orderkey` int,
     `l_partkey` int,
     `l_suppkey` int,
     `l_linenumber` int,
     `l_quantity` double,
     `l_extendedprice` double,
     `l_discount` double,
     `l_tax` double,
     `l_returnflag` string,
     `l_linestatus` string,
     `l_commitdate` string,
     `l_receiptdate` string,
     `l_shipinstruct` string,
     `l_comment` string)
   PARTITIONED BY (
     `l_shipdate` string)
   ROW FORMAT SERDE
     'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe' STORED AS
    INPUTFORMAT
     'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat' OUTPUTFORMAT
     'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat' LOCATION
    's3://DOC-EXAMPLE-BUCKET/lineitem/'
   ```

2. Jalankan`SHOW PARTITIONS <table_name>`perintah seperti berikut untuk daftar partisi.

   ```
   SHOW PARTITIONS lineitem_parq_partitioned
   ```

   Berikut ini adalah hasil sampel parsial.

   ```
   /*
   ```

```
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. Menjalankan kueri CTAS untuk membuat tabel dipartisi.

   Contoh berikut membuat tabel yang disebut`my_lineitem_parq_partitioned`dan menggunakan`WHERE` klausa untuk membatasi`DATE`ke awal dari`1992-02-01`. Karena set data sampel dimulai dengan Januari 1992, hanya partisi untuk Januari 1992 dibuat.

```
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. JalankanSHOW PARTITIONSperintah untuk memverifikasi bahwa tabel berisi partisi yang Anda inginkan.

```
SHOW PARTITIONS my_lineitem_parq_partitioned;
```

Partisi dalam contoh adalah dari Januari 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. MenggunakanINSERT INTOpernyataan untuk menambahkan partisi ke tabel.

Contoh berikut menambahkan partisi untuk tanggal dari bulan Februari 1992.

```
INSERT INTO my_lineitem_parq_partitioned
SELECT l_orderkey,
         l_partkey,
         l_suppkey,
         l_linenumber,
         l_quantity,
         l_extendedprice,
         l_discount,
         l_tax,
         l_returnflag,
         l_linestatus,
         l_commitdate,
         l_receiptdate,
         l_shipinstruct,
         l_comment,
         l_shipdate
FROM tpch100.lineitem_parq_partitioned
WHERE cast(l_shipdate as timestamp) >= DATE ('1992-02-01')
AND cast(l_shipdate as timestamp) < DATE ('1992-03-01');
```

6. Jalankan lagi `SHOW PARTITIONS`.

```
SHOW PARTITIONS my_lineitem_parq_partitioned;
```

Tabel contoh sekarang memiliki partisi dari Januari dan Februari 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. Lanjutkan menggunakan `INSERT INTO` pernyataan yang membaca dan menambahkan tidak lebih dari 100 partisi masing-masing. Lanjutkan sampai Anda mencapai jumlah partisi yang Anda butuhkan.

> **Important**
>
> Saat menyetel`WHERE`kondisi, pastikan bahwa permintaan tidak tumpang tindih. Jika tidak, beberapa partisi mungkin memiliki data yang diduplikasi.

Referensi SerDe

Athena mendukung beberapa`SerDe`perpustakaan untuk parsing data dari format data yang berbeda, seperti CSV, JSON, Parquet, dan ORC. Athena tidak mendukung kebiasaan`SerDes`.

Topik

- Menggunakan SerDe
- Format yang didukung SerDes dan data

Menggunakan SerDe

A SerDe (Serializer/Deserializer) adalah cara Athena berinteraksi dengan data dalam berbagai format.

Ini adalah yang SerDe Anda tentukan, dan bukan DDL, yang mendefinisikan skema tabel. Dengan kata lain, SerDe dapat mengganti konfigurasi DDL yang Anda tentukan di Athena saat Anda membuat tabel Anda.

Untuk menggunakan a SerDe dalam kueri

Untuk menggunakan SerDe saat membuat tabel di Athena, gunakan salah satu metode berikut:

- Tentukan `ROW FORMAT DELIMITED` dan kemudian gunakan pernyataan DDL untuk menentukan pembatas bidang, seperti pada contoh berikut. Saat Anda menentukan`ROW FORMAT DELIMITED`, Athena menggunakan secara LazySimpleSerDe default.

```
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
ESCAPED BY '\\'
COLLECTION ITEMS TERMINATED BY '|'
MAP KEYS TERMINATED BY ':'
```

  Untuk contoh`ROW FORMAT DELIMITED`, lihat topik berikut:

  LazySimpleSerDe untuk CSV, TSV, dan file yang dibatasi khusus

  Menanyakan log Amazon CloudFront

  Menanyakan log EMR Amazon

  Menanyakan log aliran VPC Amazon

  Menggunakan CTAS dan INSERT INTO untuk ETL dan analisis data

- Gunakan `ROW FORMAT SERDE` untuk secara eksplisit menentukan jenis SerDe Athena yang harus digunakan ketika membaca dan menulis data ke tabel. Contoh berikut menentukan. LazySimpleSerDe Untuk menentukan pembatas, gunakan. `WITH SERDEPROPERTIES` Properti yang ditentukan oleh `WITH SERDEPROPERTIES` sesuai dengan pernyataan terpisah (seperti`FIELDS TERMINATED BY`) dalam `ROW FORMAT DELIMITED` contoh.

```
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
WITH SERDEPROPERTIES (
'serialization.format' = ',',
'field.delim' = ',',
'collection.delim' = '|',
'mapkey.delim' = ':',
'escape.delim' = '\\'
)
```

  Untuk contoh`ROW FORMAT SERDE`, lihat topik berikut:

  AvroSerDe

Grok SerDe

Perpustakaan JSON SerDe

OpenCSV untuk SerDe memproses CSV

Regex SerDe

Format yang didukung SerDes dan data

Athena mendukung pembuatan tabel dan kueri data dari format CSV, TSV, custom-delimited, dan JSON; data dari format terkait Hadoop: ORC, Apache Avro dan Parquet; log dari Logstash, log, dan log Apache. AWS CloudTrail WebServer

> **Note**
>
> Format yang tercantum dalam bagian ini digunakan oleh Athena untuk membaca data. Untuk informasi tentang format yang digunakan Athena untuk menulis data saat menjalankan kueri CTAS, lihat. Membuat tabel dari hasil query (CTAS)

Untuk membuat tabel dan data kueri dalam format ini di Athena, tentukan kelas serializer-deserializer (SerDe) sehingga Athena tahu format mana yang digunakan dan cara mengurai data.

Tabel ini mencantumkan format data yang didukung di Athena dan pustaka yang sesuai SerDe.

A SerDe adalah pustaka khusus yang memberi tahu katalog data yang digunakan oleh Athena cara menangani data. Anda menentukan SerDe jenis dengan mencantumkannya secara eksplisit di `ROW FORMAT` bagian `CREATE TABLE` pernyataan Anda di Athena. Dalam beberapa kasus, Anda dapat menghilangkan SerDe nama karena Athena menggunakan SerDe beberapa jenis secara default untuk jenis format data tertentu.

Format data yang didukung dan SerDes

Format data	Deskripsi	SerDe jenis yang didukung di Athena
Amazon Ion	Amazon Ion adalah format data yang kaya ketik dan	Gunakan Sarang Ion Amazon SerDe.

Format data	Deskripsi	SerDe jenis yang didukung di Athena
	mendeskripsikan diri yang merupakan superset JSON, dikembangkan dan bersumber terbuka oleh Amazon.	
Apache Avro	Format untuk menyimpan data di Hadoop yang menggunakan skema berbasis JSON untuk nilai rekaman.	Gunakan AvroSerDe.
Apache Parquet	Sebuah format untuk penyimpanan kolumnar data di Hadoop.	Gunakan kompresi Parket SerDe dan SNAPPY.
Log Apache WebServer	Format untuk menyimpan log di Apache WebServer.	Gunakan Grok SerDe atauRegex SerDe.
CloudTrail log	Format untuk menyimpan log masuk CloudTrail.	• Gunakan Sarang JSON SerDe. Untuk informasi selengkapnya, lihat Meminta log AWS CloudTrail.

Format data	Deskripsi	SerDe jenis yang didukung di Athena
CSV (Nilai Dipisahkan Koma)	Untuk data dalam CSV, setiap baris mewakili catatan data, dan setiap catatan terdiri dari satu atau lebih bidang, dipisahkan dengan koma.	• Gunakan LazySimpleSerDe untuk CSV, TSV, dan file yang dibatasi khusus jika data Anda tidak menyertakan nilai yang terlampir dalam tanda kutip atau jika menggunakan `java.sql.Timestamp` format. • Gunakan OpenCSV untuk SerDe memproses CSV ketika data Anda menyertakan tanda kutip dalam nilai atau menggunakan format numerik UNIX untuk `TIMESTAMP` (misalnya ,`1564610311`).
Dibatasi Khusus	Untuk data dalam format ini, setiap baris mewakili catatan data, dan catatan dipisahkan oleh pembatas karakter tunggal kustom.	Gunakan LazySimpleSerDe untuk CSV, TSV, dan file yang dibatasi khusus dan tentukan pembatas karakter tunggal kustom.
JSON (Notasi JavaScript Objek)	Untuk data JSON, setiap baris mewakili catatan data, dan setiap catatan terdiri dari pasangan nilai atribut dan array, dipisahkan dengan koma.	• Gunakan Sarang JSON SerDe. • Gunakan OpenX JSON SerDe.
Log logstash	Format untuk menyimpan log di Logstash.	Gunakan Grok SerDe.

Format data	Deskripsi	SerDe jenis yang didukung di Athena
ORC (Kolom Baris yang Dioptimalkan)	Format untuk penyimpan an kolumnar data Hive yang dioptimalkan.	Gunakan kompresi ORC SerDe dan ZLIB.
TSV (Nilai Dipisahkan Tab)	Untuk data di TSV, setiap baris mewakili catatan data, dan setiap catatan terdiri dari satu atau lebih bidang, dipisahkan oleh tab.	Gunakan LazySimpleSerDe untuk CSV, TSV, dan file yang dibatasi khusus dan tentukan karakter pemisah sebagai`FIELDS TERMINATE D BY '\t' .`

Topik

- Sarang Ion Amazon SerDe
- AvroSerDe
- Grok SerDe
- Perpustakaan JSON SerDe
- LazySimpleSerDe untuk CSV, TSV, dan file yang dibatasi khusus
- OpenCSV untuk SerDe memproses CSV
- ORC SerDe
- Parket SerDe
- Regex SerDe

Sarang Ion Amazon SerDe

Anda dapat menggunakan Amazon Ion Hive SerDe untuk menanyakan data yang disimpan dalam format Amazon Ion. Amazon Ion adalah format data sumber terbuka yang diketik dengan kaya, menggambarkan sendiri. Format Amazon Ion digunakan oleh layanan seperti Amazon Quantum Ledger Database (Amazon QLDB) dan dalam bahasa kueri SQL open source PartiQL.

Amazon Ion memiliki format biner dan teks yang dapat dipertukarkan. Fitur ini menggabungkan kemudahan penggunaan teks dengan efisiensi pengkodean biner.

Untuk menanyakan data Amazon Ion dari Athena, Anda dapat menggunakan Amazon Ion Hive SerDe, yang membuat serial dan deserialisasi data Amazon Ion. Deserialisasi memungkinkan Anda untuk menjalankan kueri pada data Amazon Ion atau membacanya untuk menulis ke dalam format yang berbeda seperti Parket atau ORC. Serialisasi memungkinkan Anda menghasilkan data dalam format Amazon Ion dengan menggunakan `CREATE TABLE AS SELECT` (CTAS) atau `INSERT INTO` kueri untuk menyalin data dari tabel yang ada.

> **Note**
>
> Karena Amazon Ion adalah superset dari JSON, Anda dapat menggunakan Amazon Ion Hive SerDe untuk menanyakan kumpulan data JSON non-Amazon Ion. Tidak seperti SerDeperpustakaan JSON lainnya, Amazon Ion SerDe tidak mengharapkan setiap baris data berada pada satu baris. Fitur ini berguna jika Anda ingin menanyakan kumpulan data JSON yang dalam format "cetak cantik" atau memecah bidang berturut-turut dengan karakter baris baru.

Untuk informasi tambahan dan contoh kueri Amazon Ion dengan Athena, lihat Menganalisis kumpulan data Amazon Ion menggunakan Amazon Athena.

SerDe nama

- com.amazon.ionhiveserde. IonHiveSerDe

Pertimbangan dan batasan

- Bidang duplikat - Struct Amazon Ion diurutkan dan mendukung bidang duplikat, sedangkan Hive dan tidak`STRUCT<>`. `MAP<>` Jadi, ketika Anda deserialisasi bidang duplikat dari struct Amazon Ion, satu nilai dipilih secara non deterministik, dan yang lainnya diabaikan.
- Tabel simbol eksternal tidak didukung — Saat ini, Athena tidak mendukung tabel simbol eksternal atau properti Amazon Ion SerDe Hive berikut:
  - `ion.catalog.class`
  - `ion.catalog.file`
  - `ion.catalog.url`
  - `ion.symbol_table_imports`
- Ekstensi file - Amazon Ion menggunakan ekstensi file untuk menentukan codec kompresi mana yang akan digunakan untuk deserialisasi file Amazon Ion. Dengan demikian, file terkompresi harus

memiliki ekstensi file yang sesuai dengan algoritma kompresi yang digunakan. Misalnya, jika ZSTD digunakan, file yang sesuai harus memiliki ekstensi. `.zst`

- Data homogen — Amazon Ion tidak memiliki batasan pada tipe data yang dapat digunakan untuk nilai di bidang tertentu. Misalnya, dua dokumen Amazon Ion yang berbeda mungkin memiliki bidang dengan nama yang sama yang memiliki tipe data berbeda. Namun, karena Hive menggunakan skema, semua nilai yang Anda ekstrak ke kolom Hive tunggal harus memiliki tipe data yang sama.
- Pembatasan jenis kunci peta — Saat Anda membuat serial data dari format lain ke Amazon Ion, pastikan bahwa jenis kunci peta adalah salah satu dari`STRING`,`VARCHAR`, atau`CHAR`. Meskipun Hive memungkinkan Anda untuk menggunakan tipe data primitif apa pun sebagai kunci peta, simbol Amazon Ion harus berupa tipe string.
- Jenis serikat - Athena saat ini tidak mendukung jenis serikat Hive.
- Tipe data ganda — Amazon Ion saat ini tidak mendukung tipe `double` data.

Topik

- Menggunakan CREATE TABLE untuk membuat tabel Amazon Ion
- Menggunakan CTAS dan INSERT INTO untuk membuat tabel Amazon Ion
- Menggunakan Amazon IonSerDeciri
- Menggunakan ekstraktor jalur

Menggunakan CREATE TABLE untuk membuat tabel Amazon Ion

Untuk membuat tabel di Athena dari data yang disimpan dalam format Amazon Ion, Anda dapat menggunakan salah satu teknik berikut dalam pernyataan CREATE TABLE:

- Tentukan `STORED AS ION`. Dalam penggunaan ini, Anda tidak perlu menentukan Amazon Ion Hive SerDe secara eksplisit. Pilihan ini adalah opsi yang lebih mudah.
- Tentukan jalur kelas Amazon Ion di `OUTPUTFORMAT` bidang `ROW FORMAT SERDEINPUTFORMAT`,, dan.

Anda juga dapat menggunakan pernyataan `CREATE TABLE AS SELECT` (CTAS) untuk membuat tabel Amazon Ion di Athena. Untuk informasi, lihat Menggunakan CTAS dan INSERT INTO untuk membuat tabel Amazon Ion.

Menentukan DISIMPAN SEBAGAI ION

`CREATE TABLE`Pernyataan contoh berikut menggunakan `STORED AS ION` sebelum `LOCATION` klausa untuk membuat tabel berdasarkan data penerbangan dalam format Amazon Ion. `LOCATION`Klausa menentukan bucket atau folder tempat file input dalam format Ion berada. Semua file di lokasi yang ditentukan dipindai.

```
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/'
```

Menentukan jalur kelas Amazon Ion

Alih-alih menggunakan `STORED AS ION` sintaks, Anda dapat secara eksplisit menentukan nilai jalur kelas Ion untuk`ROW FORMAT SERDE`,`INPUTFORMAT`, dan `OUTPUTFORMAT` klausa sebagai berikut.

Parameter	Jalur kelas ion
`ROW FORMAT SERDE`	`'com.amazon.ionhiveserde.IonHiveSerDe'`
`STORED AS INPUTFORMAT`	`'com.amazon.ionhiveserde.formats.IonInputFormat'`
`OUTPUTFORMAT`	`'com.amazon.ionhiveserde.formats.IonOutputFormat'`

Query DDL berikut menggunakan teknik ini untuk membuat tabel eksternal yang sama seperti pada contoh sebelumnya.

```
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/'
```

Untuk informasi tentang SerDe properti untuk `CREATE TABLE` pernyataan di Athena, lihat. Menggunakan Amazon IonSerDeciri

Menggunakan CTAS dan INSERT INTO untuk membuat tabel Amazon Ion

Anda dapat menggunakan`CREATE TABLE AS SELECT`(CTAS) dan`INSERT INTO`pernyataan untuk menyalin atau menyisipkan data dari tabel ke tabel baru dalam format Amazon Ion di Athena.

Dalam kueri CTAS, tentukan`format='ION'`di dalam`WITH`klausa, seperti pada contoh berikut.

```
CREATE TABLE new_table
WITH (format='ION')
AS SELECT * from existing_table
```

Secara default, Athena membuat serial hasil Amazon Ion diFormat biner ion, tetapi Anda juga dapat menggunakan format teks. Untuk menggunakan format teks, tentukan`ion_encoding = 'TEXT'`di CTAS`WITH`klausa, seperti pada contoh berikut.

```
CREATE TABLE new_table
WITH (format='ION', ion_encoding = 'TEXT')
AS SELECT * from existing_table
```

Untuk informasi selengkapnya tentang properti spesifik Amazon Ion di CTAS`WITH`klausa, lihat bagian berikut.

CTAS DENGAN klausa properti Amazon Ion

Dalam kueri CTAS, Anda dapat menggunakan`WITH`klausa untuk menentukan format Amazon Ion dan secara opsional menentukan pengkodean Amazon Ion dan/atau algoritma kompresi tulis yang akan digunakan.

format

Anda dapat menentukan`ION`kata kunci sebagai opsi format di`WITH`klausul dari query CTAS. Saat Anda melakukannya, tabel yang Anda buat menggunakan format yang Anda tentukan`IonInputFormat`untuk membaca, dan serializes data dalam format yang Anda tentukan untuk`IonOutputFormat`.

Contoh berikut menetapkan bahwa kueri CTAS menggunakan format Amazon Ion.

```
WITH (format='ION')
```

ion_pengkodean

Opsional

Default: `BINARY`

Nilai:`BINARY`,`TEXT`

Menentukan apakah data diserialkan dalam format biner Amazon Ion atau format teks Amazon Ion. Contoh berikut menentukan format teks Amazon Ion.

```
WITH (format='ION', ion_encoding='TEXT')
```

write_compression

Opsional

Default: `GZIP`

Nilai:`GZIP`,`ZSTD`,`BZIP2`,`SNAPPY`,`NONE`

Menentukan algoritma kompresi untuk digunakan untuk kompres file output.

Contoh berikut menetapkan bahwa kueri CTAS menulis outputnya dalam format Amazon Ion menggunakan[Zstandard](#)algoritma kompresi.

```
WITH (format='ION', write_compression = 'ZSTD')
```

Untuk informasi tentang penggunaan kompresi di Athena, lihatDukungan kompresi Athena.

Untuk informasi tentang properti CTAS lainnya di Athena, lihatProperti tabel CTAS.

Menggunakan Amazon IonSerDeciri

Topik ini berisi informasi tentangSerDeproperti untukCREATE TABLEpernyataan di Athena. Untuk informasi lebih lanjut dan contoh Amazon IonSerDepenggunaan properti, lihatSerDeciridi Amazon Ion HiveSerDedokumentasi padaGitHub.

Menentukan Amazon IonSerDeciri

Untuk menentukan properti untuk Amazon Ion HiveSerDedi dalam kamuCREATE TABLEpernyataan, gunakanWITH SERDEPROPERTIESklausa. KarenaWITH SERDEPROPERTIESadalah subfield dariROW FORMAT SERDEklausa, Anda harus menentukanROW FORMAT SERDEdan Amazon Ion HiveSerDepath kelas pertama, sebagai sintaks berikut menunjukkan.

```
...
ROW FORMAT SERDE
 'com.amazon.ionhiveserde.IonHiveSerDe'
WITH SERDEPROPERTIES (
 'property' = 'value',
 'property' = 'value',
...
)
```

Perhatikan bahwa meskipunROW FORMAT SERDEklausa diperlukan jika Anda ingin menggunakanWITH SERDEPROPERTIES, Anda dapat menggunakanSTORED AS IONatau semakin lamaINPUTFORMATdanOUTPUTFORMATsintaks untuk menentukan format Amazon Ion.

Ion AmazonSerDeciri

Berikut ini adalah Amazon IonSerDesifat yang dapat digunakan dalamCREATE TABLEpernyataan di Athena.

ion.pengkodean

Opsional

Default: `BINARY`

Nilai-nilai:`BINARY`,`TEXT`

Properti ini menyatakan apakah nilai-nilai baru ditambahkan serial sebagaiBiner Amazon Ionatau format teks Amazon Ion.

Berikut iniSerDecontoh properti menentukan format teks Amazon Ion.

```
'ion.encoding' = 'TEXT'
```

ion.fail_on_overflow

Opsional

Default: `true`

Nilai-nilai:`true`,`false`

Amazon Ion memungkinkan untuk jenis numerik besar sewenang-wenang sementara Hive tidak. Secara default,SerDegagal jika nilai Amazon Ion tidak sesuai dengan kolom Hive, tetapi Anda dapat menggunakan`fail_on_overflow`opsi konfigurasi untuk membiarkan nilai meluap bukannya gagal.

Properti ini dapat diatur baik pada tingkat tabel atau kolom. Untuk menentukannya di tingkat tabel, tentukan`ion.fail_on_overflow`seperti pada contoh berikut. Ini menetapkan perilaku default untuk semua kolom.

```
'ion.fail_on_overflow' = 'true'
```

Untuk mengontrol kolom tertentu, tentukan nama kolom antara`ion`dan`fail_on_overflow`, dibatasi oleh periode, seperti pada contoh berikut.

```
'ion.<column>.fail_on_overflow' = 'false'
```

ion.path_extractor.case_sensitive

Opsional

Default: `false`

Nilai-nilai:`true`,`false`

Menentukan apakah akan memperlakukan nama bidang Amazon Ion sebagai peka huruf besar. Kapan`false`, yangSerDemengabaikan kasus parsing nama bidang Amazon Ion.

Sebagai contoh, misalkan Anda memiliki skema tabel Hive yang mendefinisikan bidang`aliases`dalam huruf kecil dan dokumen Amazon Ion dengan keduanya`aliases`bidang dan`ALIAS`lapangan, seperti pada contoh berikut.

```
-- Hive Table Schema
alias: STRING

-- Amazon Ion Document
{ 'ALIAS': 'value1'}
{ 'alias': 'value2'}
```

Contoh berikut menunjukkanSerDeproperti dan tabel diekstraksi yang dihasilkan ketika sensitivitas kasus diatur ke`false`:

```
-- Serde properties
'ion.alias.path_extractor' = '(alias)'
'ion.path_extractor.case_sensitive' = 'false'

--Extracted Table
| alias    |
|----------|
| "value1" |
| "value2" |
```

Contoh berikut menunjukkanSerDeproperti dan tabel diekstraksi yang dihasilkan ketika sensitivitas kasus diatur ke`true`:

```
-- Serde properties
'ion.alias.path_extractor' = '(alias)'
'ion.path_extractor.case_sensitive' = 'true'

--Extracted Table
| alias    |
|----------|
| "value2" |
```

Dalam kasus kedua,`value1`untuk`ALIAS`bidang diabaikan ketika sensitivitas kasus diatur ke`true`dan ekstraktor jalur ditentukan sebagai`alias`.

ion.*<column>*.path_extractor

Opsional

Bawaan: NA

Nilai: String dengan jalur pencarian

Menciptakan extractor jalan dengan jalur pencarian yang ditentukan untuk kolom yang diberikan. Ekstraktor jalur memetakan bidang Amazon Ion ke kolom Hive. Jika tidak ada ekstraktor jalur yang ditentukan, Athena secara dinamis membuat ekstraktor jalur pada waktu berjalan berdasarkan nama kolom.

Contoh jalur extractor berikut memetakan`example_ion_field`kepada`example_hive_column`.

```
'ion.example_hive_column.path_extractor' = '(example_ion_field)'
```

Untuk informasi selengkapnya tentang ekstraktor jalur dan jalur pencarian, lihatMenggunakan ekstraktor jalur.

ion.timestamp.serialization_offset

Opsional

Default: `'Z'`

Nilai-nilai:`OFFSET`, dimana`OFFSET` direpresentasikan sebagai*<signal>*`hh:mm`. Contoh nilai:`01:00`,`+01:00`,`-09:30`,`Z`(UTC, sama seperti 00:00)

Tidak seperti Apache Hivecap waktu, yang tidak memiliki zona waktu bawaan dan disimpan sebagai offset dari zaman UNIX, stempel waktu Amazon Ion memang memiliki offset. Gunakan properti ini untuk menentukan offset saat Anda membuat serial ke Amazon Ion.

Contoh berikut menambahkan offset satu jam.

```
'ion.timestamp.serialization_offset' = '+01:00'
```

ion.serialize_null

Opsional

Default: `OMIT`

Nilai-nilai:`OMIT`,`UNTYPED`,`TYPED`

Ion AmazonSerDedapat dikonfigurasi untuk baik serialize atau menghilangkan kolom yang memiliki nilai null. Anda dapat memilih untuk menulis nulls diketik kuat (`TYPED`) atau nulls tidak diketik (`UNTYPED`). Null yang diketik dengan kuat ditentukan berdasarkan pemetaan tipe Amazon Ion to Hive default.

Contoh berikut menentukan nulls sangat diketik.

```
'ion.serialize_null'='TYPED'
```

ion.ignore_cacat

Opsional

Default: `false`

Nilai-nilai:`true`,`false`

Kapan`true`, mengabaikan entri cacat atau seluruh file jikaSerDetidak dapat membacanya. Untuk informasi lebih lanjut, lihatAbaikan cacatdalam dokumentasiGitHub.

ion.***<column>***.serialize_as

Opsional

Default: Tipe default untuk kolom.

Nilai: String yang berisi tipe Amazon Ion

Menentukan tipe data Amazon Ion di mana nilai diserialkan. Karena jenis Amazon Ion dan Hive tidak selalu memiliki pemetaan langsung, beberapa jenis Hive memiliki beberapa tipe data yang valid untuk serialisasi. Untuk membuat serial data sebagai tipe data non-default, gunakan properti ini. Untuk informasi selengkapnya tentang pemetaan tipe, lihat Amazon IonJenis pemetaanhalaman padaGitHub.

Secara default, kolom Hive biner diserialkan sebagai gumpalan Amazon Ion, tetapi mereka juga dapat diserialkan sebagaiAmazon Ion gumpalan(karakter objek besar). Contoh berikut serializes kolom`example_hive_binary_column`sebagai gumpalan.

```
'ion.example_hive_binary_column.serialize_as' = 'clob'
```

Menggunakan ekstraktor jalur

Amazon Ion adalah format file gaya dokumen, tetapi Apache Hive adalah format kolom datar. Anda dapat menggunakan SerDe properti Amazon Ion khusus yang dipanggil `path extractors` untuk memetakan antara dua format. Ekstraktor jalur meratakan format Amazon Ion hierarkis, memetakan nilai Amazon Ion ke kolom Hive, dan dapat digunakan untuk mengganti nama bidang.

Athena dapat menghasilkan ekstraktor untuk Anda, tetapi Anda juga dapat menentukan ekstraktor Anda sendiri jika perlu.

Ekstraktor jalur yang dihasilkan

Secara default, Athena mencari nilai Amazon Ion tingkat atas yang cocok dengan nama kolom Hive dan membuat ekstraktor jalur saat runtime berdasarkan nilai yang cocok ini. Jika format data Amazon Ion Anda cocok dengan skema tabel Hive, Athena secara dinamis menghasilkan ekstraktor untuk Anda, dan Anda tidak perlu menambahkan ekstraktor jalur tambahan apa pun. Ekstraktor jalur default ini tidak disimpan dalam metadata tabel.

Contoh berikut menunjukkan bagaimana Athena menghasilkan ekstraktor berdasarkan nama kolom.

```
-- 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/'
```

Contoh ekstraktor berikut dihasilkan oleh Athena. Yang pertama mengekstrak `identification` bidang ke `identification` kolom, dan yang kedua mengekstrak `alias` bidang ke `alias` kolom.

```
'ion.identification.path_extractor' = '(identification)'
```

```
'ion.alias.path_extractor' = '(alias)'
```

Contoh berikut menunjukkan tabel yang diekstraksi.

```
|                  identification                    |  alias   |
|----------------------------------------------------|----------|
|{["name", "driver_license"],["John Smith", "XXXX"]} | "Johnny" |
```

Menentukan ekstraktor jalur Anda sendiri

Jika bidang Amazon Ion Anda tidak dipetakan dengan rapi ke kolom Hive, Anda dapat menentukan ekstraktor jalur Anda sendiri. Dalam `WITH SERDEPROPERTIES` klausa `CREATE TABLE` pernyataan Anda, gunakan sintaks berikut.

```
WITH SERDEPROPERTIES (
   "ion.path_extractor.case_sensitive" = "<Boolean>",
   "ion.<column_name>.path_extractor" = "<path_extractor_expression>"
)
```

> **Note**
>
> Secara default, ekstraktor jalur tidak peka huruf besar/kecil. Untuk mengganti setelan ini, setel ion.path_extractor.case_sensitive SerDe properti `ketrue`.

Menggunakan jalur pencarian di ekstraktor jalur

<path_extractor_expression>Sintaks SerDe properti untuk path extractor berisi:

```
"ion.<column_name>.path_extractor" = "<path_extractor_expression>"
```

Anda dapat menggunakan <path_extractor_expression>untuk menentukan jalur pencarian yang mem-parsing dokumen Amazon Ion dan menemukan data yang cocok. Jalur pencarian tertutup dalam tanda kurung dan dapat berisi satu atau lebih komponen berikut yang dipisahkan oleh spasi.

- Wild card — Cocokkan semua nilai.
- Indeks - Cocokkan nilai pada indeks numerik yang ditentukan. Indeks berbasis nol.
- Teks - Cocokkan semua nilai yang cocok dengan nama bidangnya setara dengan teks yang ditentukan.

- Anotasi - Mencocokkan nilai yang ditentukan oleh komponen jalur terbungkus yang memiliki anotasi yang ditentukan.

Contoh berikut menunjukkan dokumen Amazon Ion dan beberapa contoh jalur pencarian.

```
-- 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"
```

Contoh ekstraktor

Meratakan dan mengganti nama bidang

Contoh berikut menunjukkan satu set jalur pencarian yang meratakan dan mengganti nama bidang. Contoh menggunakan jalur pencarian untuk melakukan hal berikut:

- Petakan `nickname` kolom ke `alias` bidang
- Petakan `name` kolom ke `name` subbidang yang terletak di `identification` struct.

Berikut ini adalah contoh dokumen Amazon Ion.

```
-- Example Amazon Ion Document
{
    identification: {
        name: "John Smith",
        driver_license: "XXXX"
    },

    alias: "Johnny"
```

```
}
```

Berikut ini adalah contoh `CREATE TABLE` pernyataan yang mendefinisikan ekstraktor jalur.

```
-- 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/'
```

Contoh berikut menunjukkan data yang diekstraksi.

```
-- Extracted Table
| name         |   nickname   |
|--------------|--------------|
| "John Smith" |  "Johnny"    |
```

Untuk informasi selengkapnya tentang jalur penelusuran dan contoh jalur penelusuran tambahan, lihat halaman Ekstraksi Jalur Ion Java GitHub.

Mengekstrak data penerbangan ke format teks

Contoh `CREATE TABLE` kueri berikut digunakan `WITH SERDEPROPERTIES` untuk menambahkan ekstraktor jalur untuk mengekstrak data penerbangan dan menentukan pengkodean output sebagai teks Amazon Ion. Contoh menggunakan `STORED AS ION` sintaks.

```
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/'
```

AvroSerDe

SerDe nama

AvroSerDe

Nama perpustakaan

org.apache.hadoop.hive.serde2.avro.AvroSerDe

Contoh

Athena tidak mendukung penggunaan`avro.schema.url`untuk menentukan skema tabel untuk alasan keamanan. Gunakan `avro.schema.literal`. Untuk mengekstrak skema dari data dalam format Avro, gunakan Apache`avro-tools-<version>.jar`dengan`getschema`parameter. Ini mengembalikan skema yang dapat Anda gunakan dalam`WITH SERDEPROPERTIES`pernyataan. Misalnya:

```
java -jar avro-tools-1.8.2.jar getschema my_data.avro
```

Yang`avro-tools-<version>.jar`terletak di`java`subdirektori dari rilis Avro Anda diinstal. Untuk mengunduh Avro, lihatRilis Apache Avro. Untuk mengunduh Apache Avro Tools secara langsung, lihatApache Avro alat Maven repositori.

Setelah Anda mendapatkan skema, gunakan`CREATE TABLE`pernyataan untuk membuat tabel Athena berdasarkan data Avro yang mendasari yang disimpan di Amazon S3. Untuk menentukan AvroSerDe, gunakan`ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'`. Seperti yang ditunjukkan dalam

contoh berikut, Anda harus menentukan skema menggunakan`WITH SERDEPROPERTIES`klausa selain menentukan nama kolom dan jenis data yang sesuai untuk tabel.

> **Note**
>
> Ganti*myregion*dis`s3://athena-examples-`*myregion*`/path/to/data/`dengan pengidentifikasi wilayah tempat Anda menjalankan Athena, misalnya,`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/';
```

Jalankan`MSCK REPAIR TABLE`pernyataan di atas meja untuk me-refresh metadata partisi.

```
MSCK REPAIR TABLE flights_avro_example;
```

Query atas 10 kota keberangkatan dengan jumlah total keberangkatan.

```
SELECT origin, count(*) AS total_departures
FROM flights_avro_example
WHERE year >= '2000'
GROUP BY origin
ORDER BY total_departures DESC
LIMIT 10;
```

> **Note**
>
> Data tabel penerbangan berasal dariPenerbangandisediakan oleh Departemen Perhubungan AS,Biro Statistik Transportasi. Desaturated dari aslinya.

Grok SerDe

Logstash Grok SerDe adalah perpustakaan dengan serangkaian pola khusus untuk deserialisasi data teks tidak terstruktur, biasanya log. Setiap pola Grok adalah ekspresi reguler bernama. Anda dapat mengidentifikasi dan menggunakan kembali pola deserialisasi ini sesuai kebutuhan. Ini membuatnya lebih mudah untuk menggunakan Grok dibandingkan dengan menggunakan ekspresi reguler. Grok menyediakan satu set pola yang telah ditentukan sebelumnya. Anda juga dapat membuat pola khusus.

Untuk menentukan Grok SerDe saat membuat tabel di Athena, gunakan `ROW FORMAT SERDE 'com.amazonaws.glue.serde.GrokSerDe'` klausa, diikuti dengan klausa `WITH SERDEPROPERTIES` yang menentukan pola yang cocok dalam data Anda, di mana:

- `input.format`Ekspresi mendefinisikan pola yang cocok dalam data. Hal ini diperlukan.
- `input.grokCustomPatterns`Ekspresi mendefinisikan pola kustom bernama, yang selanjutnya dapat Anda gunakan dalam `input.format` ekspresi. Ini opsional. Untuk menyertakan beberapa entri pola ke dalam `input.grokCustomPatterns` ekspresi, gunakan karakter escape baris baru (\n) untuk memisahkannya, sebagai berikut:. `'input.grokCustomPatterns'='INSIDE_QS ([^\"]*)\nINSIDE_BRACKETS ([^\\]]*)')`
- `OUTPUTFORMAT`Klausul `STORED AS INPUTFORMAT` dan diperlukan.
- `LOCATION`Klausa menentukan bucket Amazon S3, yang dapat berisi beberapa objek data. Semua objek data dalam bucket dideserialisasi untuk membuat tabel.

Contoh

Contoh-contoh ini bergantung pada daftar pola Grok yang telah ditentukan. Lihat pola yang telah ditentukan sebelumnya.

Contoh 1

Contoh ini menggunakan data sumber dari entri maillog Postfix yang disimpan di. `s3://DOC-EXAMPLE-BUCKET/groksample/`

```
Feb  9 07:15:00 m4eastmail postfix/smtpd[19305]: B88C4120838: connect from
 unknown[192.168.55.4]
Feb  9 07:15:00 m4eastmail postfix/smtpd[20444]: B58C4330038:
 client=unknown[192.168.55.4]
Feb  9 07:15:03 m4eastmail postfix/cleanup[22835]: BDC22A77854: message-
id=<31221401257553.5004389LCBF@m4eastmail.example.com>
```

Pernyataan berikut membuat tabel di Athena dipanggil `mygroktable` dari sumber data, menggunakan pola kustom dan pola standar yang Anda tentukan:

```
CREATE EXTERNAL TABLE `mygroktable`(
   syslogbase string,
   queue_id string,
   syslog_message string
   )
ROW FORMAT SERDE
   'com.amazonaws.glue.serde.GrokSerDe'
WITH SERDEPROPERTIES (
   'input.grokCustomPatterns' = 'POSTFIX_QUEUEID [0-9A-F]{7,12}',
   'input.format'='%{SYSLOGBASE} %{POSTFIX_QUEUEID:queue_id}:
 %{GREEDYDATA:syslog_message}'
   )
STORED AS INPUTFORMAT
   'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
   'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
   's3://DOC-EXAMPLE-BUCKET/groksample/';
```

Mulailah dengan pola sederhana, seperti`%{NOTSPACE:column}`, untuk mendapatkan kolom dipetakan terlebih dahulu dan kemudian mengkhususkan kolom jika diperlukan.

Contoh 2

Dalam contoh berikut, Anda membuat kueri untuk log Log4j. Contoh log memiliki entri dalam format ini:

```
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
```

Untuk menanyakan data log ini:

- Tambahkan pola Grok ke `input.format` untuk setiap kolom. Misalnya, untuk`timestamp`, tambahkan`%{TIMESTAMP_ISO8601:timestamp}`. Untuk`loglevel`, tambahkan`%{LOGLEVEL:loglevel}`.
- Pastikan pola di `input.format` cocok dengan format log persis, dengan memetakan tanda hubung (`-`) dan koma yang memisahkan entri dalam format log.

```
CREATE EXTERNAL TABLE bltest (
 timestamp STRING,
 loglevel STRING,
 processtype STRING,
 processid STRING,
 status STRING,
 threadid STRING,
 executiontime INT,
 tenantid INT,
 userid STRING,
 shard STRING,
 jobid STRING,
 messagetype STRING,
 action STRING,
 hostname STRING
 )
ROW FORMAT SERDE 'com.amazonaws.glue.serde.GrokSerDe'
WITH SERDEPROPERTIES (
"input.grokCustomPatterns" = 'C_ACTION receive|send',
```

```
"input.format" = "%{TIMESTAMP_ISO8601:timestamp} %{LOGLEVEL:loglevel} - processType=
%{NOTSPACE:processtype}, processId=%{NOTSPACE:processid}, status=%{NOTSPACE:status},
 threadId=%{NOTSPACE:threadid}, executionTime=%{POSINT:executiontime}, tenantId=
%{POSINT:tenantid}, userId=%{NOTSPACE:userid}, shard=%{NOTSPACE:shard}, jobId=
%{NOTSPACE:jobid}, messageType=%{NOTSPACE:messagetype}, action=%{C_ACTION:action},
 hostname=%{HOST:hostname}"
) STORED AS INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://DOC-EXAMPLE-BUCKET/samples/';
```

Contoh 3

Contoh kueri log Amazon S3 berikut menunjukkan `'input.grokCustomPatterns'` ekspresi yang berisi dua entri pola, dipisahkan oleh karakter escape baris baru \n (), seperti yang ditunjukkan dalam cuplikan ini dari contoh kueri:. `'input.grokCustomPatterns'='INSIDE_QS ([^ \"]*)\nINSIDE_BRACKETS ([^\\]]*)')`

```
CREATE EXTERNAL TABLE `s3_access_auto_raw_02`(
  `bucket_owner` string COMMENT 'from deserializer',
  `bucket` string COMMENT 'from deserializer',
  `time` string COMMENT 'from deserializer',
  `remote_ip` string COMMENT 'from deserializer',
  `requester` string COMMENT 'from deserializer',
  `request_id` string COMMENT 'from deserializer',
  `operation` string COMMENT 'from deserializer',
  `key` string COMMENT 'from deserializer',
  `request_uri` string COMMENT 'from deserializer',
  `http_status` string COMMENT 'from deserializer',
  `error_code` string COMMENT 'from deserializer',
  `bytes_sent` string COMMENT 'from deserializer',
  `object_size` string COMMENT 'from deserializer',
  `total_time` string COMMENT 'from deserializer',
  `turnaround_time` string COMMENT 'from deserializer',
  `referrer` string COMMENT 'from deserializer',
  `user_agent` string COMMENT 'from deserializer',
  `version_id` string COMMENT 'from deserializer')
ROW FORMAT SERDE
  'com.amazonaws.glue.serde.GrokSerDe'
WITH SERDEPROPERTIES (
  'input.format'='%{NOTSPACE:bucket_owner} %{NOTSPACE:bucket} \
\[%{INSIDE_BRACKETS:time}\\] %{NOTSPACE:remote_ip} %{NOTSPACE:requester}
 %{NOTSPACE:request_id} %{NOTSPACE:operation} %{NOTSPACE:key} \"?
```

```
%{INSIDE_QS:request_uri}\"? %{NOTSPACE:http_status} %{NOTSPACE:error_code}
 %{NOTSPACE:bytes_sent} %{NOTSPACE:object_size} %{NOTSPACE:total_time}
 %{NOTSPACE:turnaround_time} \"?%{INSIDE_QS:referrer}\"? \"?%{INSIDE_QS:user_agent}\"?
 %{NOTSPACE:version_id}',
  'input.grokCustomPatterns'='INSIDE_QS ([^\"]*)\nINSIDE_BRACKETS ([^\\]]*)')
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET'
```

Perpustakaan JSON SerDe

Di Athena, Anda dapat menggunakan SerDe pustaka untuk deserialisasi data JSON. Deserialisasi mengubah data JSON sehingga dapat diserialisasi (ditulis) ke dalam format yang berbeda seperti Parket atau ORC.

- Penduduk asli Sarang JSON SerDe
- Sebuah OpenX JSON SerDe
- Sebuah Sarang Ion Amazon SerDe

> **Note**
>
> Pustaka Hive dan OpenX mengharapkan data JSON berada pada satu baris (tidak diformat), dengan catatan dipisahkan oleh karakter baris baru. Amazon Ion Hive SerDe tidak memiliki persyaratan itu dan dapat digunakan sebagai alternatif karena format data Ion adalah superset dari JSON.

Nama perpustakaan

Gunakan salah satu langkah berikut:

org.apache.hive.hcatalog.data. JsonSerDe

org.openx.data.jsonserde. JsonSerDe

com.amazon.ionhiveserde. IonHiveSerDe

Sarang JSON SerDe

Hive JSON biasanya SerDe digunakan untuk memproses data JSON seperti peristiwa. Peristiwa ini direpresentasikan sebagai string baris tunggal dari teks yang disandikan JSON yang dipisahkan oleh baris baru. Hive JSON SerDe tidak mengizinkan kunci duplikat dalam `map` atau `struct` nama kunci.

> **Note**
>
> SerDe Mengharapkan setiap dokumen JSON berada pada satu baris teks tanpa karakter penghentian baris yang memisahkan bidang dalam catatan. Jika teks JSON dalam format cetak cantik, Anda mungkin menerima pesan kesalahan seperti HIVE_CURSOR_ERROR: Row is not a valid JSON Object or HIVE_CURSOR_ERROR:: Unexpected JsonParseException end-of-input: expected close marker for OBJECT saat Anda mencoba menanyakan tabel setelah Anda membuatnya. Untuk informasi selengkapnya, lihat File Data JSON di dokumentasi SerDe OpenX. GitHub

Contoh pernyataan DDL berikut menggunakan Hive JSON SerDe untuk membuat tabel berdasarkan sampel data iklan online. Dalam `LOCATION` klausa, ganti *`myregion`* `s3://DOC-EXAMPLE-BUCKET.elasticmapreduce/samples/hive-ads/tables/impressions` dengan pengidentifikasi wilayah tempat Anda menjalankan Athena (misalnya,). `s3://us-west-2.elasticmapreduce/samples/hive-ads/tables/impressions`

```
CREATE EXTERNAL TABLE impressions (
    requestbegintime string,
    adid string,
    impressionid string,
    referrer string,
    useragent string,
    usercookie string,
    ip string,
    number string,
    processid string,
    browsercookie string,
    requestendtime string,
    timers struct
                <
                 modellookup:string,
                 requesttime:string
                >,
    threadid string,
```

```
    hostname string,
    sessionid string
)
PARTITIONED BY (dt string)
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
LOCATION 's3://DOC-EXAMPLE-BUCKET.elasticmapreduce/samples/hive-ads/tables/
impressions';
```

Menentukan format stempel waktu dengan Hive JSON SerDe

Untuk mengurai nilai stempel waktu dari string, Anda dapat menambahkan `WITH SERDEPROPERTIES` subfield ke `ROW FORMAT SERDE` klausa dan menggunakannya untuk menentukan parameter. `timestamp.formats` Dalam parameter, tentukan daftar terpisah koma dari satu atau beberapa pola stempel waktu, seperti pada contoh berikut:

```
...
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")
...
```

Untuk informasi selengkapnya, lihat Stempel waktu dalam dokumentasi Apache Hive.

Memuat tabel untuk kueri

Setelah Anda membuat tabel, jalankan MSCK REPAIR TABLE untuk memuat tabel dan membuatnya dapat ditanyakan dari Athena:

```
MSCK REPAIR TABLE impressions
```

Meminta log CloudTrail

Anda dapat menggunakan Hive JSON SerDe untuk query CloudTrail log. Untuk informasi lebih lanjut dan contoh `CREATE TABLE` pernyataan, lihatMeminta log AWS CloudTrail.

OpenX JSON SerDe

Seperti Hive JSON SerDe, Anda dapat menggunakan OpenX JSON untuk memproses data JSON. Data juga direpresentasikan sebagai string baris tunggal dari teks yang dikodekan JSON yang dipisahkan oleh baris baru. Seperti Hive JSON SerDe, OpenX JSON SerDe tidak mengizinkan kunci duplikat masuk atau nama kunci. `map` `struct`

> **Note**
>
> SerDe Mengharapkan setiap dokumen JSON berada pada satu baris teks tanpa karakter penghentian baris yang memisahkan bidang dalam catatan. Jika teks JSON dalam format cetak cantik, Anda mungkin menerima pesan kesalahan seperti HIVE_CURSOR_ERROR: Row is not a valid JSON Object or HIVE_CURSOR_ERROR:: Unexpected JsonParseException end-of-input: expected close marker for OBJECT saat Anda mencoba menanyakan tabel setelah Anda membuatnya. Untuk informasi selengkapnya, lihat File Data JSON dalam dokumentasi SerDe OpenX pada. GitHub

Properti opsional

Berbeda dengan Hive JSON SerDe, OpenX JSON SerDe juga memiliki SerDe properti opsional berikut yang dapat berguna untuk mengatasi inkonsistensi dalam data.

abaikan.malformed.json

Tidak wajib. Saat disetel ke`TRUE`, memungkinkan Anda melewati sintaks JSON yang salah bentuk. Nilai default-nya `FALSE`.

dots.in.keys

Tidak wajib. Nilai default-nya `FALSE`. Ketika diatur ke`TRUE`, memungkinkan SerDe untuk mengganti titik-titik dalam nama kunci dengan garis bawah. Sebagai contoh, jika set data JSON berisi kunci dengan nama`"a.b"`, Anda dapat menggunakan properti ini untuk menentukan nama kolom menjadi`"a_b"`di Athena. Secara default (tanpa ini SerDe), Athena tidak mengizinkan titik dalam nama kolom.

kasus.tidak sensitif

Tidak wajib. Nilai default-nya `TRUE`. Ketika diatur ke`TRUE`, SerDe mengubah semua kolom huruf besar menjadi huruf kecil.

Untuk menggunakan nama kunci peka huruf besar/kecil dalam data Anda, gunakan. `WITH SERDEPROPERTIES ("case.insensitive"= FALSE;)` Kemudian, untuk setiap kunci yang belum semua huruf kecil, berikan pemetaan dari nama kolom ke nama properti menggunakan sintaks berikut:

```
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES ("case.insensitive" = "FALSE", "mapping.userid" = "userId")
```

Jika Anda memiliki dua kunci seperti URL dan Url itu sama ketika mereka dalam huruf kecil, kesalahan seperti berikut dapat terjadi:

HIVE_CURSOR_ERROR: Baris bukan Objek JSON yang valid - JSONException: Kunci duplikat “url”

Untuk mengatasi hal ini, atur case.insensitive properti ke FALSE dan petakan kunci ke nama yang berbeda, seperti pada contoh berikut:

```
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES ("case.insensitive" = "FALSE", "mapping.url1" = "URL",
 "mapping.url2" = "Url")
```

pemetaan

Tidak wajib. Memetakan nama kolom ke kunci JSON yang tidak identik dengan nama kolom. mappingParameter ini berguna ketika data JSON berisi kunci yang merupakan kata kunci. Misalnya, jika Anda memiliki kunci JSON bernamatimestamp, gunakan sintaks berikut untuk memetakan kunci ke kolom bernama: ts

```
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES ("mapping.ts" = "timestamp")
```

Memetakan nama bidang bersarang dengan titik dua ke nama yang kompatibel dengan HIVE

Jika Anda memiliki nama bidang dengan titik dua di dalamstruct, Anda dapat menggunakan mapping properti untuk memetakan bidang ke nama yang kompatibel dengan HIVE. Misalnya, jika definisi jenis kolom Anda berisimy:struct:field:string, Anda dapat memetakan definisi tersebut my_struct_field:string dengan menyertakan entri berikut diWITH SERDEPROPERTIES:

```
("mapping.my_struct_field" = "my:struct:field")
```

Contoh berikut menunjukkan CREATE TABLE pernyataan yang sesuai.

```
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")
```

Contoh: data iklan

Contoh pernyataan DDL berikut menggunakan OpenX SerDe JSON untuk membuat tabel berdasarkan sampel data iklan online yang sama yang digunakan dalam contoh untuk Hive JSON. SerDe Dalam `LOCATION` klausa, ganti *`myregion`* dengan pengidentifikasi wilayah tempat Anda menjalankan Athena.

```
CREATE EXTERNAL TABLE impressions (
    requestbegintime string,
    adid string,
    impressionId string,
    referrer string,
    useragent string,
    usercookie string,
    ip string,
    number string,
    processid string,
    browsercokie string,
    requestendtime string,
    timers struct<
       modellookup:string,
       requesttime:string>,
    threadid string,
    hostname string,
    sessionid string
)   PARTITIONED BY (dt string)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 's3://DOC-EXAMPLE-BUCKET.elasticmapreduce/samples/hive-ads/tables/
impressions';
```

Contoh: deserialisasi JSON bersarang

Anda dapat menggunakan JSON SerDes untuk mengurai data yang dikodekan JSON yang lebih kompleks. Ini membutuhkan penggunaan `CREATE TABLE` pernyataan yang menggunakan `struct` dan `array` elemen untuk mewakili struktur bersarang.

Contoh berikut membuat tabel Athena dari data JSON yang memiliki struktur bersarang. Contohnya memiliki struktur sebagai berikut:

```
{
"DocId": "AWS",
"User": {
```

```
        "Id": 1234,
        "Username": "carlos_salazar",
        "Name": "Carlos",
"ShippingAddress": {
"Address1": "123 Main St.",
"Address2": null,
"City": "Anytown",
"State": "CA"
   },
"Orders": [
   {
     "ItemId": 6789,
     "OrderDate": "11/11/2022"
   },
   {
     "ItemId": 4352,
     "OrderDate": "12/12/2022"
   }
  ]
 }
}
```

Ingat bahwa OpenX SerDe mengharapkan setiap catatan JSON berada pada satu baris teks. Saat disimpan di Amazon S3, semua data dalam contoh sebelumnya harus dalam satu baris, seperti ini:

```
{"DocId":"AWS","User":
{"Id":1234,"Username":"carlos_salazar","Name":"Carlos","ShippingAddress" ...
```

`CREATE TABLE`Pernyataan berikut menggunakan Openx- JsonSerDe dengan `struct` dan `array` mengumpulkan tipe data untuk menetapkan kelompok objek untuk data contoh.

```
CREATE external TABLE complex_json (
   docid string,
   `user` struct<
               id:INT,
               username:string,
               name:string,
               shippingaddress:struct<
                                  address1:string,
                                  address2:string,
                                  city:string,
                                  state:string
```

```
                                    >,
                orders:array<
                            struct<
                                 itemid:INT,
                                  orderdate:string
                                  >
                              >
                >
   )
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 's3://DOC-EXAMPLE-BUCKET/myjsondata/';
```

Untuk menanyakan tabel, gunakan SELECT pernyataan seperti berikut ini.

```
SELECT
 user.name as Name,
 user.shippingaddress.address1 as Address,
 user.shippingaddress.city as City,
 o.itemid as Item_ID, o.orderdate as Order_date
FROM complex_json, UNNEST(user.orders) as temp_table (o)
```

Untuk mengakses bidang data di dalam struct, kueri sampel menggunakan notasi titik (misalnya,`user.name`). Untuk mengakses data di dalam array struct (seperti pada `orders` bidang), Anda dapat menggunakan UNNEST fungsi. UNNESTFungsi meratakan array ke dalam tabel sementara (dalam hal ini disebuto). Ini memungkinkan Anda menggunakan notasi titik seperti yang Anda lakukan dengan struct untuk mengakses elemen array yang tidak bersarang (misalnya,). `o.itemid` Nama`temp_table`, yang digunakan dalam contoh untuk tujuan ilustrasi, sering disingkat sebagai. `t`

Tabel berikut menunjukkan hasil query.

#	Nama	Alamat	Kota	Item_id	Pesanan_tanggal
1	Carlos	123 Utama St.	Kota Anytown	6789	11/11/2022
2	Carlos	123 Utama St.	Kota Anytown	4352	12/12/2022

Sumber daya tambahan

Untuk informasi selengkapnya tentang bekerja dengan JSON dan JSON bersarang di Athena, lihat sumber daya berikut:

- Buat tabel di Amazon Athena dari JSON bersarang dan pemetaan menggunakan JSON (Big Data Blog) SerDe AWS
- Saya mendapatkan kesalahan ketika saya mencoba membaca data JSON di Amazon Athena AWS (artikel Pusat Pengetahuan)
- hive-json-schema(GitHub) — Alat yang ditulis dalam Java yang menghasilkan `CREATE TABLE` pernyataan dari contoh dokumen JSON. `CREATE TABLE`Pernyataan yang dihasilkan menggunakan OpenX JSON Serde.

LazySimpleSerDe untuk CSV, TSV, dan file yang dibatasi khusus

Menentukan ini SerDe adalah opsional. Ini adalah SerDe untuk data dalam CSV, TSV, dan format yang dibatasi khusus yang digunakan Athena secara default. SerDe Ini digunakan jika Anda tidak menentukan SerDe dan hanya menentukan`ROW FORMAT DELIMITED`. Gunakan ini SerDe jika data Anda tidak memiliki nilai yang terlampir dalam tanda kutip.

Untuk dokumentasi referensi tentang LazySimpleSerDe, lihat SerDe bagian Hive dari Panduan Pengembang Apache Hive.

Nama perpustakaan

Nama perpustakaan Kelas untuk LazySimpleSerDe adalah`org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe`. Untuk informasi tentang LazySimpleSerDe kelas, lihat LazySimpleSerDe.java GitHub di.com.

Mengabaikan header

Untuk mengabaikan header dalam data Anda ketika Anda mendefinisikan tabel, Anda dapat menggunakan properti `skip.header.line.count` tabel, seperti pada contoh berikut.

```
TBLPROPERTIES ("skip.header.line.count"="1")
```

Sebagai contoh, lihat `CREATE TABLE` pernyataan di Menanyakan log aliran VPC Amazon danMenanyakan log Amazon CloudFront .

Contoh CSV

Contoh berikut menunjukkan bagaimana menggunakan `LazySimpleSerDe` untuk membuat tabel di Athena dari data CSV. Untuk deserialisasi file yang dibatasi khusus menggunakan ini SerDe, ikuti pola dalam contoh tetapi gunakan `FIELDS TERMINATED BY` klausa untuk menentukan pembatas karakter tunggal yang berbeda. LazySimpleSerDe tidak mendukung pembatas multi-karakter.

> **Note**
>
> Ganti *`myregion`* `s3://athena-examples-`*`myregion`*`/path/to/data/` dengan pengidentifikasi wilayah tempat Anda menjalankan Athena, misalnya,. `s3://athena-examples-us-west-1/path/to/data/`

Gunakan `CREATE TABLE` pernyataan untuk membuat tabel Athena dari data dasar CSV yang disimpan di Amazon S3.

```
CREATE EXTERNAL TABLE flight_delays_csv (
    yr INT,
    quarter INT,
    month INT,
    dayofmonth INT,
    dayofweek INT,
    flightdate STRING,
    uniquecarrier STRING,
    airlineid INT,
    carrier STRING,
    tailnum STRING,
    flightnum STRING,
    originairportid INT,
    originairportseqid INT,
    origincitymarketid INT,
    origin STRING,
    origincityname STRING,
    originstate STRING,
    originstatefips STRING,
    originstatename STRING,
    originwac INT,
    destairportid INT,
    destairportseqid INT,
    destcitymarketid INT,
    dest STRING,
```

```
    destcityname STRING,
    deststate STRING,
    deststatefips STRING,
    deststatename STRING,
    destwac INT,
    crsdeptime STRING,
    deptime STRING,
    depdelay INT,
    depdelayminutes INT,
    depdel15 INT,
    departuredelaygroups INT,
    deptimeblk STRING,
    taxiout INT,
    wheelsoff STRING,
    wheelson STRING,
    taxiin INT,
    crsarrtime INT,
    arrtime STRING,
    arrdelay INT,
    arrdelayminutes INT,
    arrdel15 INT,
    arrivaldelaygroups INT,
    arrtimeblk STRING,
    cancelled INT,
    cancellationcode STRING,
    diverted INT,
    crselapsedtime INT,
    actualelapsedtime INT,
    airtime INT,
    flights INT,
    distance INT,
    distancegroup INT,
    carrierdelay INT,
    weatherdelay INT,
    nasdelay INT,
    securitydelay INT,
    lateaircraftdelay INT,
    firstdeptime STRING,
    totaladdgtime INT,
    longestaddgtime INT,
    divairportlandings INT,
    divreacheddest INT,
    divactualelapsedtime INT,
    divarrdelay INT,
```

```
    divdistance INT,
    div1airport STRING,
    div1airportid INT,
    div1airportseqid INT,
    div1wheelson STRING,
    div1totalgtime INT,
    div1longestgtime INT,
    div1wheelsoff STRING,
    div1tailnum STRING,
    div2airport STRING,
    div2airportid INT,
    div2airportseqid INT,
    div2wheelson STRING,
    div2totalgtime INT,
    div2longestgtime INT,
    div2wheelsoff STRING,
    div2tailnum STRING,
    div3airport STRING,
    div3airportid INT,
    div3airportseqid INT,
    div3wheelson STRING,
    div3totalgtime INT,
    div3longestgtime INT,
    div3wheelsoff STRING,
    div3tailnum STRING,
    div4airport STRING,
    div4airportid INT,
    div4airportseqid INT,
    div4wheelson STRING,
    div4totalgtime INT,
    div4longestgtime INT,
    div4wheelsoff STRING,
    div4tailnum STRING,
    div5airport STRING,
    div5airportid INT,
    div5airportseqid INT,
    div5wheelson STRING,
    div5totalgtime INT,
    div5longestgtime INT,
    div5wheelsoff STRING,
    div5tailnum STRING
)
    PARTITIONED BY (year STRING)
    ROW FORMAT DELIMITED
```

```
    FIELDS TERMINATED BY ','
    ESCAPED BY '\\'
    LINES TERMINATED BY '\n'
  LOCATION 's3://athena-examples-myregion/flight/csv/';
```

Jalankan `MSCK REPAIR TABLE` untuk menyegarkan metadata partisi setiap kali partisi baru ditambahkan ke tabel ini:

```
MSCK REPAIR TABLE flight_delays_csv;
```

Kueri 10 rute teratas yang tertunda lebih dari 1 jam:

```
SELECT origin, dest, count(*) as delays
FROM flight_delays_csv
WHERE depdelayminutes > 60
GROUP BY origin, dest
ORDER BY 3 DESC
LIMIT 10;
```

> **Note**
>
> Data tabel penerbangan berasal dari Penerbangan yang disediakan oleh Departemen Perhubungan AS, Biro Statistik Transportasi. Desaturasi dari aslinya.

Contoh TSV

Untuk membuat tabel Athena dari data TSV yang disimpan di Amazon S3, gunakan `ROW FORMAT DELIMITED` dan tentukan sebagai pembatas bidang tab, `\t` sebagai pemisah garis, dan `\n` sebagai karakter escape. \ Kutipan berikut menunjukkan sintaks ini. Tidak ada contoh data penerbangan TSV yang tersedia di `athena-examples` lokasi, tetapi seperti tabel CSV, Anda akan menjalankan `MSCK REPAIR TABLE` untuk menyegarkan metadata partisi setiap kali partisi baru ditambahkan.

```
...
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
ESCAPED BY '\\'
LINES TERMINATED BY '\n'
...
```

OpenCSV untuk SerDe memproses CSV

Saat Anda membuat tabel Athena untuk data CSV, tentukan tabel yang SerDe akan digunakan berdasarkan jenis nilai yang terkandung dalam data Anda:

- Jika data Anda berisi nilai yang diapit tanda kutip ganda ('), Anda dapat menggunakan SerDeOpenCSV untuk deserialisasi nilai di Athena. Jika data Anda tidak berisi nilai yang terlampir dalam tanda kutip ganda ('), Anda dapat menghilangkan penentuannya. SerDe Dalam hal ini, Athena menggunakan default. `LazySimpleSerDe` Untuk informasi, lihat LazySimpleSerDe untuk CSV, TSV, dan file yang dibatasi khusus.
- Jika data Anda memiliki `TIMESTAMP` nilai numerik UNIX (misalnya,`1579059880000`), gunakan OpenCSV. SerDe Jika data Anda menggunakan `java.sql.Timestamp` format, gunakan file LazySimpleSerDe.

CSV SerDe (SerDeOpenCSV)

SerDeOpenCSV memiliki karakteristik berikut untuk data string:

- Menggunakan tanda kutip ganda (') sebagai karakter kutipan default, dan memungkinkan Anda menentukan karakter pemisah, kutipan, dan pelarian, seperti:

```
WITH SERDEPROPERTIES ("separatorChar" = ",", "quoteChar" = "`", "escapeChar" = "\\" )
```

- Tidak dapat melarikan diri `\t` atau `\n` secara langsung. Untuk melarikan diri dari mereka, gunakan`"escapeChar" = "\\"`. Lihat contoh dalam topik ini.
- Tidak mendukung jeda baris yang disematkan dalam file CSV.

Untuk tipe data selain`STRING`, SerDe OpenCSV berperilaku sebagai berikut:

- Mengenali`BOOLEAN`,`BIGINT`,`INT`, dan tipe `DOUBLE` data.
- Tidak mengenali nilai kosong atau nol dalam kolom yang didefinisikan sebagai tipe data numerik, meninggalkannya sebagai. `string` Salah satu solusinya adalah membuat kolom dengan nilai nol sebagai `string` dan kemudian digunakan `CAST` untuk mengonversi bidang dalam kueri menjadi tipe data numerik, memberikan nilai default untuk nol. `0` Untuk informasi lebih lanjut, lihat Ketika saya menanyakan data CSV di Athena, saya mendapatkan kesalahan HIVE_BAD_DATA: Error parsing field value in the Knowledge Center. AWS

- Untuk kolom yang ditentukan dengan tipe `timestamp` data dalam `CREATE TABLE` pernyataan Anda, mengenali `TIMESTAMP` data jika ditentukan dalam format numerik UNIX dalam milidetik, seperti. `1579059880000`
  - SerDe OpenCSV tidak `TIMESTAMP` mendukung dalam format yang `java.sql.Timestamp` sesuai dengan JDBC, seperti (presisi tempat desimal 9). `"YYYY-MM-DD HH:MM:SS.fffffffff"`
- Untuk kolom yang ditentukan dengan tipe `DATE` data dalam `CREATE TABLE` pernyataan Anda, kenali nilai sebagai tanggal jika nilai mewakili jumlah hari yang telah berlalu sejak 1 Januari 1970. Misalnya, nilai `18276` dalam kolom dengan tipe `date` data dirender seperti `2020-01-15` saat ditanyakan. Dalam format UNIX ini, setiap hari dianggap memiliki 86.400 detik.
  - SerDe OpenCSV tidak `DATE` mendukung dalam format lain secara langsung. Untuk memproses data stempel waktu dalam format lain, Anda dapat menentukan kolom sebagai `string` dan kemudian menggunakan fungsi konversi waktu untuk mengembalikan hasil yang diinginkan dalam kueri Anda`SELECT`. Untuk informasi lebih lanjut, lihat artikel Saat saya menanyakan tabel di Amazon Athena, hasil TIMESTAMP kosong di pusat pengetahuan.AWS
- Untuk mengonversi kolom lebih lanjut ke jenis yang diinginkan dalam tabel, Anda dapat membuat tampilan di atas tabel dan menggunakannya `CAST` untuk mengonversi ke jenis yang diinginkan.

Example Contoh: Menggunakan tipe TIMESTAMP dan tipe DATE yang ditentukan dalam format numerik UNIX.

Pertimbangkan tiga kolom berikut dari data yang dipisahkan koma. Nilai-nilai di setiap kolom diapit tanda kutip ganda.

```
"unixvalue creationdate 18276 creationdatetime 1579059880000","18276","1579059880000"
```

Pernyataan berikut membuat tabel di Athena dari lokasi bucket Amazon S3 yang ditentukan.

```
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'
```

Selanjutnya, jalankan query berikut:

```
SELECT * FROM testtimestamp1
```

Query mengembalikan hasil sebagai berikut, menunjukkan tanggal dan waktu data:

```
profile_id                                                          creationdate
 creationdatetime
unixvalue creationdate 18276 creationdatetime 1579146280000        2020-01-15
 2020-01-15 03:44:40.000
```

Example Contoh: Melarikan diri **\t** atau **\n**

Pertimbangkan data uji berikut:

```
" \\t\\t\\n 123 \\t\\t\\n ",abc
" 456 ",xyz
```

Pernyataan berikut membuat tabel di Athena, menentukan itu. `"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/'
```

Selanjutnya, jalankan query berikut:

```
SELECT * FROM test1;
```

Ia mengembalikan hasil ini, melarikan diri dengan benar `\t` atau`\n`:

```
f1            s2
\t\t\n 123 \t\t\n            abc
456                          xyz
```

SerDe nama

CSV SerDe

Nama perpustakaan

Untuk menggunakan ini SerDe, tentukan nama kelas yang sepenuhnya memenuhi syarat setelahnya`ROW FORMAT SERDE`. Juga tentukan pembatas di dalamnya`SERDEPROPERTIES`, sebagai berikut:

```
...
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
  "separatorChar" = ",",
  "quoteChar"     = "`",
  "escapeChar"    = "\\"
)
```

Mengabaikan header

Untuk mengabaikan header dalam data Anda ketika Anda mendefinisikan tabel, Anda dapat menggunakan properti `skip.header.line.count` tabel, seperti pada contoh berikut.

```
TBLPROPERTIES ("skip.header.line.count"="1")
```

Sebagai contoh, lihat `CREATE TABLE` pernyataan di Menanyakan log aliran VPC Amazon dan Menanyakan log Amazon CloudFront .

Contoh

Contoh ini menganggap data dalam CSV disimpan `s3://DOC-EXAMPLE-BUCKET/mycsv/` dengan konten berikut:

```
"a1","a2","a3","a4"
"1","2","abc","def"
"a","a1","abc3","ab4"
```

Gunakan `CREATE TABLE` pernyataan untuk membuat tabel Athena berdasarkan data. Referensikan kelas SerDe OpenCSV `ROW FORMAT SERDE` setelah dan tentukan pemisah karakter, karakter kutipan, dan karakter escape `WITH SERDEPROPERTIES` di, seperti pada contoh berikut.

```
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/';
```

Kueri semua nilai dalam tabel:

```
SELECT * FROM myopencsvtable;
```

Query mengembalikan nilai-nilai berikut:

```
col1     col2    col3    col4
-----------------------------
a1       a2      a3      a4
1        2       abc     def
a        a1      abc3    ab4
```

ORC SerDe

SerDe nama

OrcSerDe

Nama perpustakaan

Pustaka ini menggunakan kelas Apache Hive OrcSerde.java untuk data dalam format ORC. Ini meneruskan objek dari ORC ke pembaca dan dari ORC ke penulis.

Contoh

> **Note**
>
> Ganti *myregion* `s3://athena-examples-`*myregion*`/path/to/data/` dengan pengidentifikasi wilayah tempat Anda menjalankan Athena, misalnya,. `s3://athena-examples-us-west-1/path/to/data/`

Contoh berikut membuat tabel untuk data penundaan penerbangan di ORC. Tabel termasuk partisi:

```
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");
```

Jalankan `MSCK REPAIR TABLE` pernyataan di atas meja untuk menyegarkan metadata partisi:

```
MSCK REPAIR TABLE flight_delays_orc;
```

Gunakan kueri ini untuk mendapatkan 10 rute teratas yang tertunda lebih dari 1 jam:

```
SELECT origin, dest, count(*) as delays
FROM flight_delays_orc
WHERE depdelayminutes > 60
GROUP BY origin, dest
ORDER BY 3 DESC
```

```
LIMIT 10;
```

Parket SerDe

SerDe nama

ParquetHiveSerDe digunakan untuk data yang disimpan dalam format Parket.

> **Note**
>
> Untuk mengonversi data ke dalam format Parket, Anda dapat menggunakan kueri CREATE TABLE AS SELECT (CTAS). Untuk informasi lebih lanjut, lihatMembuat tabel dari hasil query (CTAS), Contoh kueri CTAS danMenggunakan CTAS dan INSERT INTO untuk ETL dan analisis data.

Nama perpustakaan

Athena menggunakan kelas berikut ketika perlu deserialisasi data yang disimpan di Parket: `org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe`

Contoh: Menanyakan file yang disimpan di parket

> **Note**
>
> Ganti *`myregion`* `s3://athena-examples-`*`myregion`*`/path/to/data/` dengan pengidentifikasi wilayah tempat Anda menjalankan Athena, misalnya,. `s3://athena-examples-us-west-1/path/to/data/`

Gunakan `CREATE TABLE` pernyataan berikut untuk membuat tabel Athena dari data dasar yang disimpan dalam format Parket di Amazon S3:

```
CREATE EXTERNAL TABLE flight_delays_pq (
    yr INT,
    quarter INT,
    month INT,
    dayofmonth INT,
    dayofweek INT,
    flightdate STRING,
    uniquecarrier STRING,
```

```
    airlineid INT,
    carrier STRING,
    tailnum STRING,
    flightnum STRING,
    originairportid INT,
    originairportseqid INT,
    origincitymarketid INT,
    origin STRING,
    origincityname STRING,
    originstate STRING,
    originstatefips STRING,
    originstatename STRING,
    originwac INT,
    destairportid INT,
    destairportseqid INT,
    destcitymarketid INT,
    dest STRING,
    destcityname STRING,
    deststate STRING,
    deststatefips STRING,
    deststatename STRING,
    destwac INT,
    crsdeptime STRING,
    deptime STRING,
    depdelay INT,
    depdelayminutes INT,
    depdel15 INT,
    departuredelaygroups INT,
    deptimeblk STRING,
    taxiout INT,
    wheelsoff STRING,
    wheelson STRING,
    taxiin INT,
    crsarrtime INT,
    arrtime STRING,
    arrdelay INT,
    arrdelayminutes INT,
    arrdel15 INT,
    arrivaldelaygroups INT,
    arrtimeblk STRING,
    cancelled INT,
    cancellationcode STRING,
    diverted INT,
    crselapsedtime INT,
```

```
    actualelapsedtime INT,
    airtime INT,
    flights INT,
    distance INT,
    distancegroup INT,
    carrierdelay INT,
    weatherdelay INT,
    nasdelay INT,
    securitydelay INT,
    lateaircraftdelay INT,
    firstdeptime STRING,
    totaladdgtime INT,
    longestaddgtime INT,
    divairportlandings INT,
    divreacheddest INT,
    divactualelapsedtime INT,
    divarrdelay INT,
    divdistance INT,
    div1airport STRING,
    div1airportid INT,
    div1airportseqid INT,
    div1wheelson STRING,
    div1totalgtime INT,
    div1longestgtime INT,
    div1wheelsoff STRING,
    div1tailnum STRING,
    div2airport STRING,
    div2airportid INT,
    div2airportseqid INT,
    div2wheelson STRING,
    div2totalgtime INT,
    div2longestgtime INT,
    div2wheelsoff STRING,
    div2tailnum STRING,
    div3airport STRING,
    div3airportid INT,
    div3airportseqid INT,
    div3wheelson STRING,
    div3totalgtime INT,
    div3longestgtime INT,
    div3wheelsoff STRING,
    div3tailnum STRING,
    div4airport STRING,
    div4airportid INT,
```

```
    div4airportseqid INT,
    div4wheelson STRING,
    div4totalgtime INT,
    div4longestgtime INT,
    div4wheelsoff STRING,
    div4tailnum STRING,
    div5airport STRING,
    div5airportid INT,
    div5airportseqid INT,
    div5wheelson STRING,
    div5totalgtime INT,
    div5longestgtime INT,
    div5wheelsoff STRING,
    div5tailnum STRING
)
PARTITIONED BY (year STRING)
STORED AS PARQUET
LOCATION 's3://athena-examples-myregion/flight/parquet/'
tblproperties ("parquet.compression"="SNAPPY");
```

Jalankan `MSCK REPAIR TABLE` pernyataan di atas meja untuk menyegarkan metadata partisi:

```
MSCK REPAIR TABLE flight_delays_pq;
```

Kueri 10 rute teratas yang tertunda lebih dari 1 jam:

```
SELECT origin, dest, count(*) as delays
FROM flight_delays_pq
WHERE depdelayminutes > 60
GROUP BY origin, dest
ORDER BY 3 DESC
LIMIT 10;
```

> **Note**
>
> Data tabel penerbangan berasal dari Penerbangan yang disediakan oleh Departemen Perhubungan AS, Biro Statistik Transportasi. Desaturasi dari aslinya.

Mengabaikan statistik Parket

Ketika Anda membaca data Parket, Anda mungkin menerima pesan kesalahan seperti berikut:

```
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.
```

Untuk mengatasi masalah ini, gunakan ALTER TABLE SET TBLPROPERTIES pernyataan CREATE TABLE or untuk menyetel SerDe `parquet.ignore.statistics` properti Parket`true`, seperti pada contoh berikut.

CREATE TABLE contoh

```
...
ROW FORMAT SERDE
'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
WITH SERDEPROPERTIES (
'parquet.ignore.statistics'='true')
STORED AS PARQUET
...
```

Contoh ALTER TABLE

```
ALTER TABLE ... SET TBLPROPERTIES ('parquet.ignore.statistics'='true')
```

Regex SerDe

Regex SerDe menggunakan ekspresi reguler (regex) untuk deserialisasi data dengan mengekstrak grup regex ke dalam kolom tabel.

Jika baris dalam data tidak cocok dengan regex, maka semua kolom di baris dikembalikan sebagai. `NULL` Jika baris cocok dengan regex tetapi memiliki lebih sedikit grup dari yang diharapkan, grup yang hilang adalah. `NULL` Jika baris dalam data cocok dengan regex tetapi memiliki lebih banyak kolom daripada grup di regex, kolom tambahan akan diabaikan.

Untuk informasi selengkapnya, lihat Kelas RegexSerDe dalam dokumentasi Apache Hive.

SerDe nama

RegexSerDe

Nama perpustakaan

RegexSerDe

Contoh

Contoh berikut membuat tabel dari CloudFront log menggunakan RegExSerDe. Ganti *myregion* `s3://athena-examples-`*myregion*`/cloudfront/plaintext/` dengan pengidentifikasi wilayah tempat Anda menjalankan Athena (misalnya,). `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+[^\(]+[\(]([^\;]+).*\%20([^\/]+)[\/]
(.*)$"
 )
LOCATION 's3://athena-examples-myregion/cloudfront/plaintext/';
```

Menjalankan kueri SQL menggunakan Amazon Athena

Anda dapat menjalankan kueri SQL menggunakan Amazon Athena pada sumber data yang terdaftar dengan AWS Glue Data Catalog dan sumber data seperti metastores Hive dan Amazon DocumentDB contoh yang Anda terhubung ke menggunakan fitur Kueri Gabungan Athena. Untuk informasi selengkapnya tentang bekerja dengan sumber data, lihat Menghubungkan ke sumber data. Saat Anda menjalankan kueri Data Definition Language (DDL) yang memodifikasi skema,

Athena menulis metadata metastore yang terkait dengan sumber data. Selain itu, beberapa kueri, seperti`CREATE TABLE AS`dan`INSERT INTO`dapat menulis catatan ke set data–misalnya, menambahkan catatan CSV ke lokasi Amazon S3. Saat Anda menjalankan kueri, Athena menyimpan hasil kueri di lokasi hasil kueri yang Anda tentukan. Hal ini memungkinkan Anda untuk melihat riwayat kueri dan untuk mengunduh dan melihat hasil kueri set.

Bagian ini menyediakan panduan untuk menjalankan kueri Athena pada sumber data umum dan tipe data menggunakan berbagai pernyataan SQL. Panduan umum disediakan untuk bekerja dengan struktur umum dan operator–misalnya, bekerja dengan larik, concatenating, pemfilteran, perataan, dan pemilahan. Contoh lain termasuk kueri untuk data dalam tabel dengan struktur dan peta bersarang, tabel berdasarkan kumpulan data yang disandikan JSON, dan kumpulan data yang terkait dengan seperti log dan log EMR Amazon. Layanan AWS AWS CloudTrail Cakupan komprehensif penggunaan SQL standar berada di luar cakupan dokumentasi ini. Untuk informasi lebih lanjut tentang SQL, lihat referensi bahasa Trino dan Presto.

Topik

- Meminta log Layanan AWS
- Menanyakan log server web yang disimpan di Amazon S3

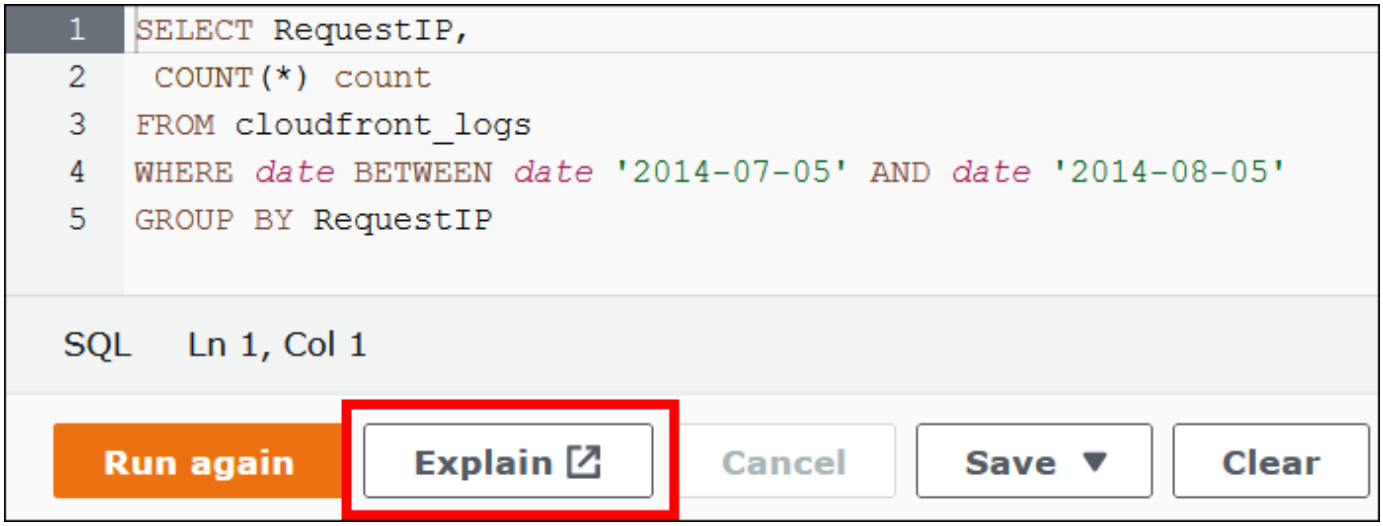
Untuk pertimbangan dan batasan, lihat Pertimbangan dan batasan untuk kueri SQL di Amazon Athena.

Melihat rencana eksekusi untuk kueri SQL

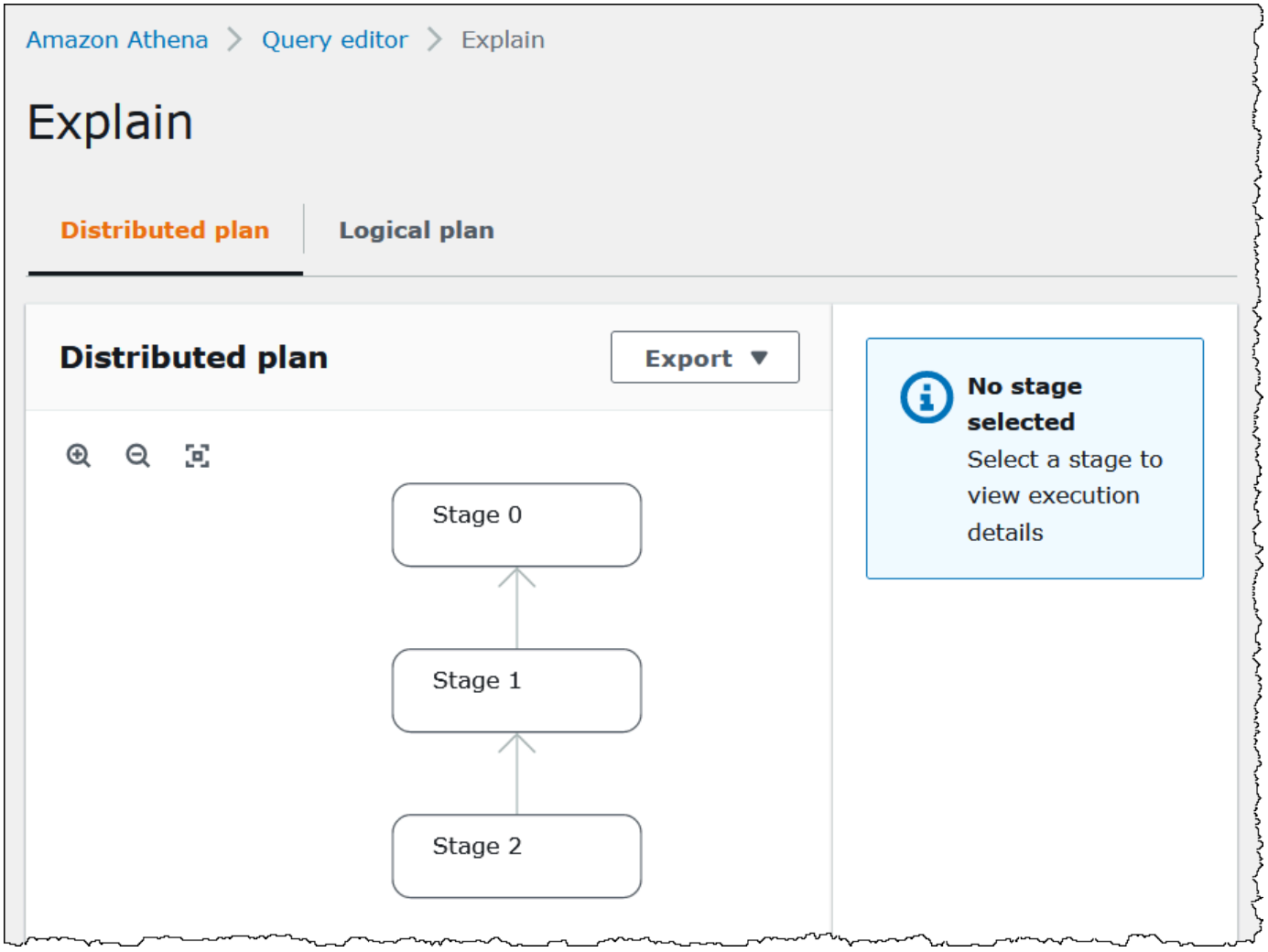
Anda dapat menggunakan editor kueri Athena untuk melihat representasi grafis tentang bagaimana kueri Anda akan dijalankan. Saat Anda memasukkan kueri di editor dan memilih opsi Jelaskan, Athena menggunakan pernyataan EXPLOW SQL pada kueri Anda untuk membuat dua grafik yang sesuai: rencana eksekusi terdistribusi dan rencana eksekusi logis. Anda dapat menggunakan grafik ini untuk menganalisis, memecahkan masalah, dan meningkatkan efisiensi kueri Anda.

Untuk melihat rencana eksekusi untuk kueri

1. Masukkan kueri Anda di editor kueri Athena, lalu pilih Jelaskan.



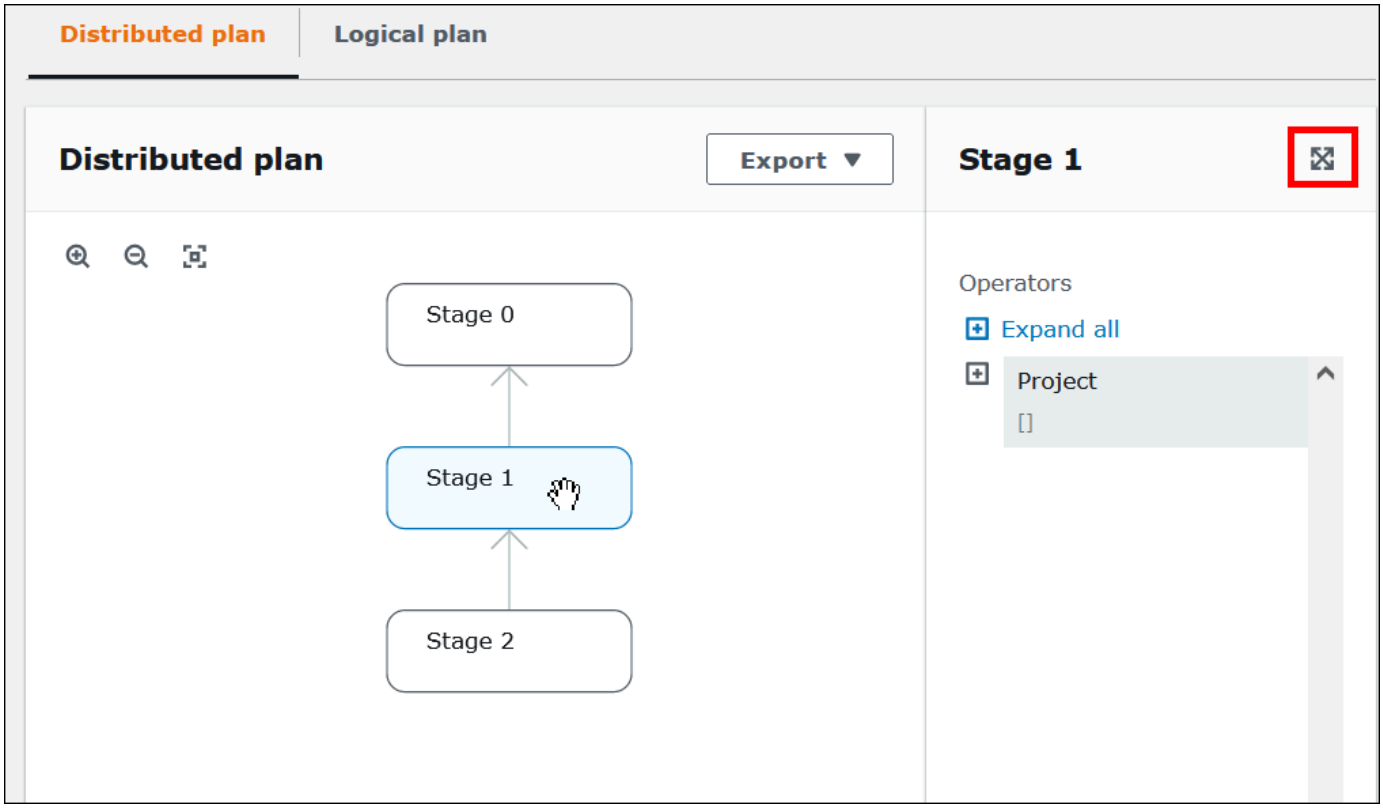
   Tab Paket terdistribusi menunjukkan rencana eksekusi untuk kueri Anda di lingkungan terdistribusi. Rencana terdistribusi memiliki fragmen atau tahapan pemrosesan. Setiap tahap memiliki nomor indeks berbasis nol dan diproses oleh satu atau lebih node. Data dapat dipertukarkan antar node.



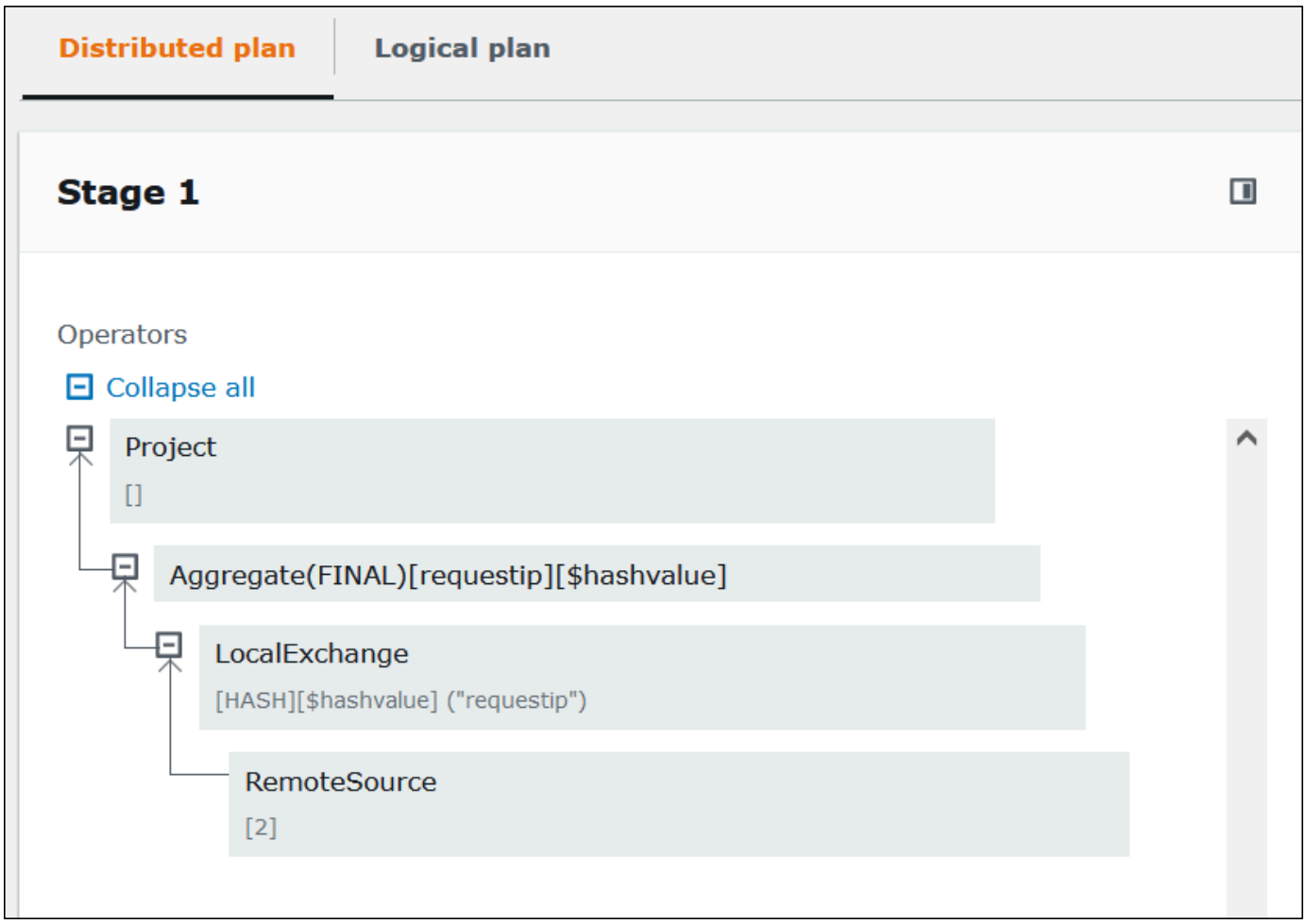
2. Untuk menavigasi grafik, gunakan opsi berikut:

   - Untuk memperbesar atau memperkecil, gulir mouse, atau gunakan ikon pembesar.
   - Untuk menyesuaikan grafik agar sesuai dengan layar, pilih ikon Zoom to fit.
   - Untuk memindahkan grafik, seret penunjuk mouse.

3. Untuk melihat detail panggung, pilih panggung.

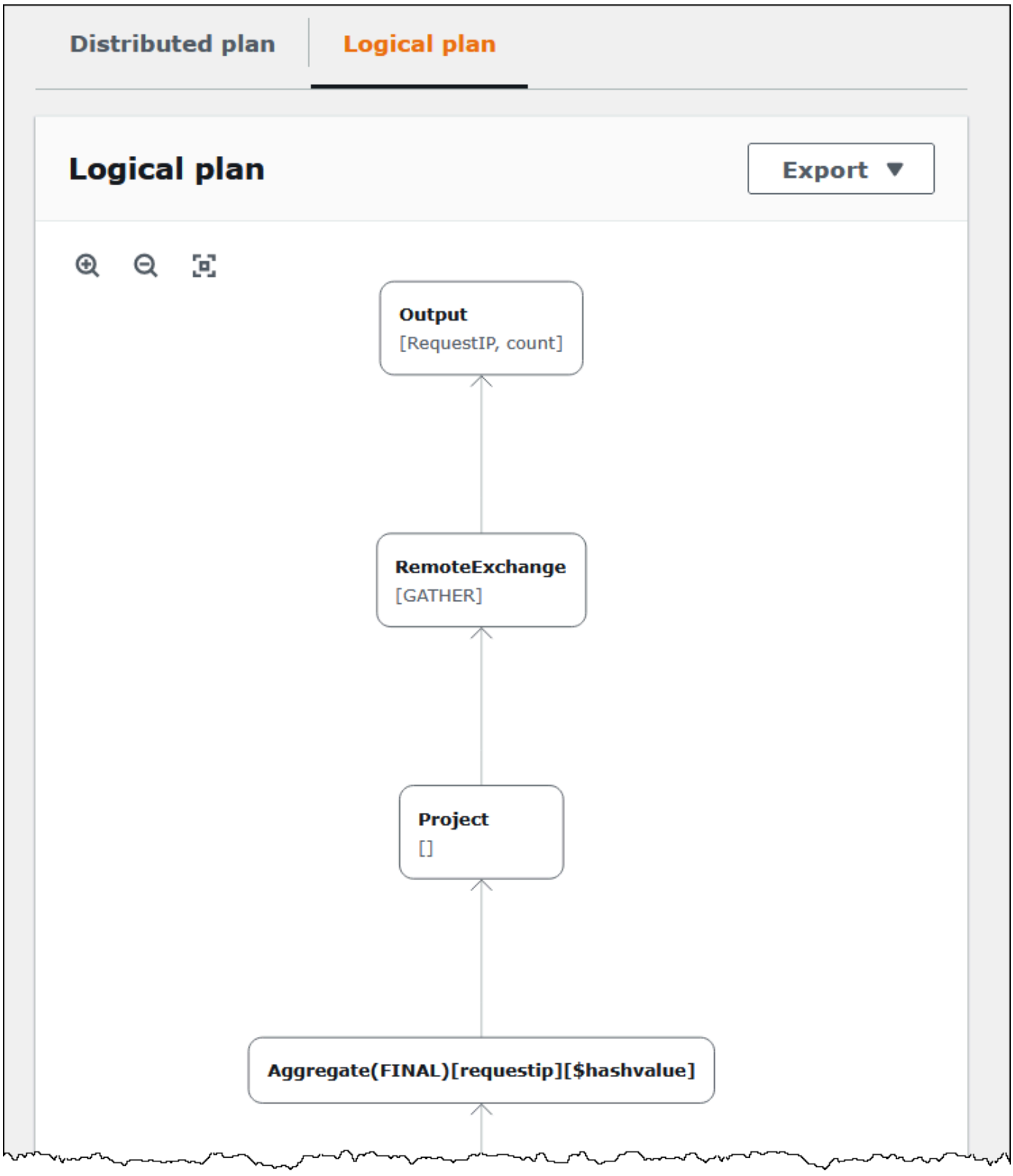


4. Untuk melihat detail panggung lebar penuh, pilih ikon perluas di kanan atas panel detail.
5. Untuk melihat lebih detail, perluas satu atau beberapa item di pohon operator. Untuk informasi tentang fragmen rencana terdistribusi, lihatJELASKAN jenis keluaran pernyataan.



> **Important**
>
> Saat ini, beberapa filter partisi mungkin tidak terlihat di grafik pohon operator bersarang meskipun Athena menerapkannya ke kueri Anda. Untuk memverifikasi efek filter tersebut, jalankan EXPLOW atau EXPLY ANALYSIS pada kueri Anda dan lihat hasilnya.

6. Pilih tab Rencana logis. Grafik menunjukkan rencana logis untuk menjalankan kueri Anda. Untuk informasi tentang persyaratan operasional, lihatMemahami Athena MENJELASKAN hasil pernyataan.



7. Untuk mengekspor paket sebagai gambar SVG atau PNG, atau sebagai teks JSON, pilih Ekspor.

Sumber daya tambahan

Untuk informasi selengkapnya, lihat sumber daya berikut ini.

Menggunakan JELASKAN dan JELASKAN ANALISIS di Athena

Memahami Athena MENJELASKAN hasil pernyataan

Melihat statistik dan detail eksekusi untuk kueri yang diselesaikan

Bekerja dengan hasil kueri, kueri terbaru, dan file keluaran

Amazon Athena secara otomatis menyimpan hasil kueri dan informasi metadata untuk setiap kueri yang berjalan dalamlokasi hasil kueriAnda dapat menentukan di Amazon S3. Jika perlu, Anda dapat mengakses file di lokasi ini untuk bekerja dengan mereka. Anda juga dapat mengunduh file hasil kueri langsung dari konsol Athena.

Untuk mengatur lokasi hasil kueri Amazon S3 untuk pertama kalinya, lihat Menentukan lokasi hasil kueri menggunakan konsol Athena.

File output disimpan secara otomatis untuk setiap kueri yang berjalan. Untuk mengakses dan melihat file keluaran kueri menggunakan konsol Athena, kepala sekolah IAM (pengguna dan peran) memerlukan izin untuk tindakan Amazon S3 untuk lokasi hasil kueri, serta izin untuk GetObjecttindakan Athena. GetQueryResults Lokasi hasil kueri dapat dienkripsi. Jika lokasi dienkripsi, pengguna harus memiliki izin kunci yang sesuai untuk mengenkripsi dan mendekripsi lokasi hasil permintaan.

> **Important**
>
> IAM utama dengan izin ke Amazon S3`GetObject`tindakan untuk lokasi hasil kueri dapat mengambil hasil kueri dari Amazon S3 bahkan jika izin ke Athena`GetQueryResults`tindakan ditolak.

Menentukan lokasi hasil query

Lokasi hasil kueri yang digunakan Athena ditentukan oleh kombinasi pengaturan grup kerja danPengaturan sisi klien. Pengaturan sisi klien didasarkan pada bagaimana Anda menjalankan kueri.

- Jika Anda menjalankan kueri menggunakan konsol Athena,Lokasi hasil kueridimasukkan di bawahPengaturandi bilah navigasi menentukan pengaturan sisi klien.

- Jika Anda menjalankan kueri menggunakan Athena API, `OutputLocation` parameter StartQueryExecutiontindakan menentukan setelan sisi klien.
- Jika Anda menggunakan ODBC atau JDBC driver untuk menjalankan kueri,`S3OutputLocation` properti yang ditentukan dalam URL koneksi menentukan pengaturan sisi klien.

> **Important**
>
> Saat Anda menjalankan kueri menggunakan API atau menggunakan ODBC atau JDBC driver, pengaturan konsol tidak berlaku.

Setiap konfigurasi grup kerja memilikiMenimpa pengaturan sisi klienpilihan yang dapat diaktifkan. Saat opsi ini diaktifkan, pengaturan grup kerja diutamakan atas pengaturan sisi klien berlaku saat utama IAM terkait dengan kelompok kerja yang menjalankan kueri.

Menentukan lokasi hasil kueri menggunakan konsol Athena

Sebelum Anda dapat menjalankan kueri, kueri hasil bucket lokasi di Amazon S3 harus ditentukan, atau Anda harus menggunakan grup kerja yang telah ditentukan bucket dan konfigurasi yang menimpa pengaturan klien.

Untuk menentukan lokasi hasil kueri pengaturan sisi klien menggunakan konsol Athena

1. Beralih ke workgroup yang ingin Anda tentukan lokasi hasil kueri. Nama workgroup default adalah primary.
2. Dari bilah navigasi, pilih Pengaturan.
3. Dari bilah navigasi, pilih Kelola.
4. Untuk Mengelola pengaturan, lakukan salah satu hal berikut:
   - Di kotak Lokasi hasil kueri, masukkan jalur ke bucket yang Anda buat di Amazon S3 untuk hasil kueri. Awalan jalur dengan`s3://`.
   - Pilih Browse S3, pilih bucket Amazon S3 yang Anda buat untuk Wilayah Anda saat ini, lalu pilih Pilih.

> **Note**
>
> Jika Anda menggunakan workgroup yang menentukan lokasi hasil kueri untuk semua pengguna workgroup, opsi untuk mengubah lokasi hasil kueri tidak tersedia.

5. (Opsional) Pilih Lihat konfigurasi siklus hidup untuk melihat dan mengonfigurasi aturan siklus hidup Amazon S3 pada bucket hasil kueri Anda. Aturan siklus hidup Amazon S3 yang Anda buat dapat berupa aturan kedaluwarsa atau aturan transisi. Aturan kedaluwarsa secara otomatis menghapus hasil kueri setelah jangka waktu tertentu. Aturan transisi memindahkannya ke tingkat penyimpanan Amazon S3 lainnya. Untuk informasi selengkapnya, lihat Menyetel konfigurasi siklus hidup pada bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.
6. (Opsional) Untuk pemilik bucket yang Diharapkan, masukkan ID Akun AWS yang Anda harapkan sebagai pemilik bucket lokasi keluaran. Ini adalah langkah keamanan tambahan. Jika ID akun pemilik bucket tidak cocok dengan ID yang Anda tentukan di sini, upaya untuk menampilkan ke bucket akan gagal. Untuk informasi lebih lanjut, lihat Memverifikasi kepemilikan bucket dengan kondisi pemilik bucket di Panduan Pengguna Amazon S3.

> **Note**
>
> Setelan pemilik bucket yang diharapkan hanya berlaku untuk lokasi keluaran Amazon S3 yang Anda tentukan untuk hasil kueri Athena. Ini tidak berlaku untuk lokasi Amazon S3 lainnya seperti lokasi sumber data di bucket Amazon S3 eksternal`CTAS`, `INSERT INTO` dan lokasi tabel tujuan, lokasi keluaran pernyataan`UNLOAD`, operasi untuk menumpahkan bucket untuk kueri federasi, `SELECT` atau kueri yang dijalankan terhadap tabel di akun lain.

7. (Opsional) Pilih Enkripsi hasil kueri jika Anda ingin mengenkripsi hasil kueri yang disimpan di Amazon S3. Untuk informasi lebih lanjut tentang enkripsi di Athena, lihat. Enkripsi diam
8. (Opsional) Pilih Tetapkan kontrol penuh pemilik bucket atas hasil kueri untuk memberikan akses kontrol penuh atas hasil kueri kepada pemilik bucket saat ACL diaktifkan untuk bucket hasil kueri. Misalnya, jika lokasi hasil kueri Anda dimiliki oleh akun lain, Anda dapat memberikan kepemilikan dan kontrol penuh atas hasil kueri Anda ke akun lain. Untuk informasi selengkapnya, lihat Mengontrol kepemilikan objek dan menonaktifkan ACL untuk bucket Anda di Panduan Pengguna Amazon S3.
9. Pilih Simpan.

Lokasi default yang dibuat sebelumnya

Sebelumnya di Athena, jika Anda menjalankan kueri tanpa menentukan nilai untuk lokasi hasil Kueri, dan pengaturan lokasi hasil kueri tidak diganti oleh grup kerja, Athena membuat lokasi default untuk Anda. Lokasi default adalah`aws-athena-query-results-`*`MyAcctID`*`-`*`MyRegion`*, di mana *`MyAcctID adalah ID`* akun Amazon Web Services dari prinsipal IAM yang menjalankan kueri, dan *`MyRegion`*merupakan wilayah tempat kueri dijalankan (misalnya,`us-west-1`.)

Sekarang, sebelum Anda dapat menjalankan kueri Athena di wilayah tempat akun Anda belum digunakan Athena sebelumnya, Anda harus menentukan lokasi hasil kueri, atau menggunakan grup kerja yang menimpa pengaturan lokasi hasil kueri. Meski Athena tidak lagi membuat lokasi hasil kueri default untuk Anda, default yang sebelumnya dibuat lokasi `aws-athena-query-results-`*`MyAcctID`*`-`*`MyRegion`* tetap valid dan Anda dapat terus menggunakannya.

Menentukan lokasi hasil kueri menggunakan workgroup

Anda menentukan lokasi hasil kueri dalam konfigurasi grup kerja menggunakan AWS Management Console, AWS CLI, atau API Athena.

Saat menggunakan AWS CLI, tentukan lokasi hasil kueri menggunakan `OutputLocation` parameter `--configuration` opsi saat Anda menjalankan `aws athena update-work-group`perintah `aws athena create-work-group`or.

Untuk menentukan lokasi hasil permintaan untuk kelompok kerja menggunakan konsol Athena

1. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.

2. Di panel navigasi, pilih Workgroups.

3. Dalam daftar workgroup, pilih link workgroup yang ingin Anda edit.
4. Pilih Edit.
5. Untuk lokasi dan enkripsi hasil Query, lakukan salah satu hal berikut:
   - Di kotak Lokasi hasil kueri, masukkan jalur ke bucket di Amazon S3 untuk hasil kueri Anda. Awalan jalur dengan`s3://`.
   - Pilih Browse S3, pilih bucket Amazon S3 untuk Wilayah saat ini yang ingin Anda gunakan, lalu pilih Pilih.
6. (Opsional) Untuk pemilik bucket yang Diharapkan, masukkan ID Akun AWS yang Anda harapkan sebagai pemilik bucket lokasi keluaran. Ini adalah langkah keamanan tambahan. Jika ID akun pemilik bucket tidak cocok dengan ID yang Anda tentukan di sini, upaya untuk menampilkan ke bucket akan gagal. Untuk informasi lebih lanjut, lihat Memverifikasi kepemilikan bucket dengan kondisi pemilik bucket di Panduan Pengguna Amazon S3.

   > **Note**
   >
   > Setelan pemilik bucket yang diharapkan hanya berlaku untuk lokasi keluaran Amazon S3 yang Anda tentukan untuk hasil kueri Athena. Ini tidak berlaku untuk lokasi Amazon S3 lainnya seperti lokasi sumber data di bucket Amazon S3 eksternal`CTAS`, `INSERT INTO` dan lokasi tabel tujuan, lokasi keluaran pernyataan`UNLOAD`, operasi untuk menumpahkan bucket untuk kueri federasi, `SELECT` atau kueri yang dijalankan terhadap tabel di akun lain.

7. (Opsional) Pilih Enkripsi hasil kueri jika Anda ingin mengenkripsi hasil kueri yang disimpan di Amazon S3. Untuk informasi lebih lanjut tentang enkripsi di Athena, lihat. Enkripsi diam
8. (Opsional) Pilih Tetapkan kontrol penuh pemilik bucket atas hasil kueri untuk memberikan akses kontrol penuh atas hasil kueri kepada pemilik bucket saat ACL diaktifkan untuk bucket hasil kueri. Misalnya, jika lokasi hasil kueri Anda dimiliki oleh akun lain, Anda dapat memberikan kepemilikan dan kontrol penuh atas hasil kueri Anda ke akun lain.

   Jika setelan Kepemilikan Objek S3 bucket lebih disukai pemilik Bucket, pemilik bucket juga memiliki semua objek hasil kueri yang ditulis dari workgroup ini. Misalnya, jika grup kerja akun eksternal mengaktifkan opsi ini dan menetapkan lokasi hasil kueri ke bucket Amazon S3 akun Anda yang memiliki pengaturan Kepemilikan Objek S3 dari pemilik Bucket yang lebih disukai, Anda memiliki dan memiliki akses kontrol penuh atas hasil kueri grup kerja eksternal.

Memilih opsi ini ketika setelan Kepemilikan Objek S3 bucket hasil kueri diberlakukan oleh pemilik Bucket tidak berpengaruh. Untuk informasi selengkapnya, lihat Mengontrol kepemilikan objek dan menonaktifkan ACL untuk bucket Anda di Panduan Pengguna Amazon S3.

9. Jika Anda ingin meminta semua pengguna workgroup untuk menggunakan lokasi hasil kueri yang Anda tentukan, gulir ke bawah ke bagian Pengaturan dan pilih Ganti pengaturan sisi klien.
10. Pilih Simpan perubahan.

Mengunduh file hasil kueri menggunakan konsol Athena

Anda dapat mengunduh file CSV hasil kueri dari panel kueri segera setelah Anda menjalankan kueri. Anda juga dapat mengunduh hasil kueri dari kueri terbaru dari tab Kueri terbaru.

> **Note**
>
> File hasil kueri Athena adalah file data yang berisi informasi yang dapat dikonfigurasi oleh pengguna individu. Beberapa program yang membaca dan menganalisis data ini berpotensi menafsirkan beberapa data sebagai perintah (CSV injection). Untuk alasan ini, saat Anda mengimpor data CSV hasil kueri ke program spreadsheet, program tersebut mungkin akan memperingatkan Anda tentang masalah keamanan. Untuk menjaga sistem Anda aman, Anda harus selalu memilih untuk menonaktifkan tautan atau makro dari hasil kueri yang diunduh.

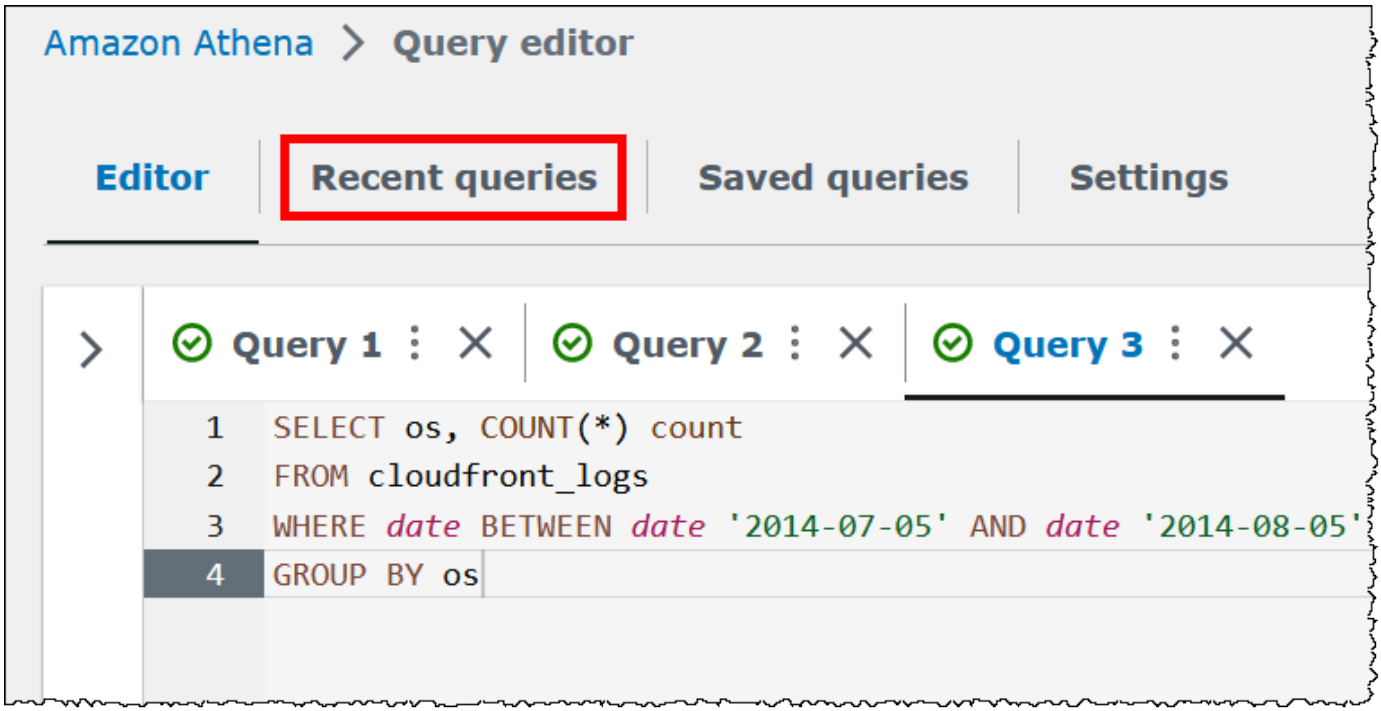
Untuk menjalankan kueri dan mengunduh hasil kueri

1. Masukkan kueri Anda di editor kueri dan kemudian pilih Jalankan.

   Saat kueri selesai berjalan,Hasilmenampilkan hasil kueri.
2. Untuk mengunduh file CSV hasil kueri, pilih Unduh hasil di atas panel hasil kueri. Bergantung pada konfigurasi browser dan browser Anda, Anda mungkin perlu mengonfirmasi unduhan.

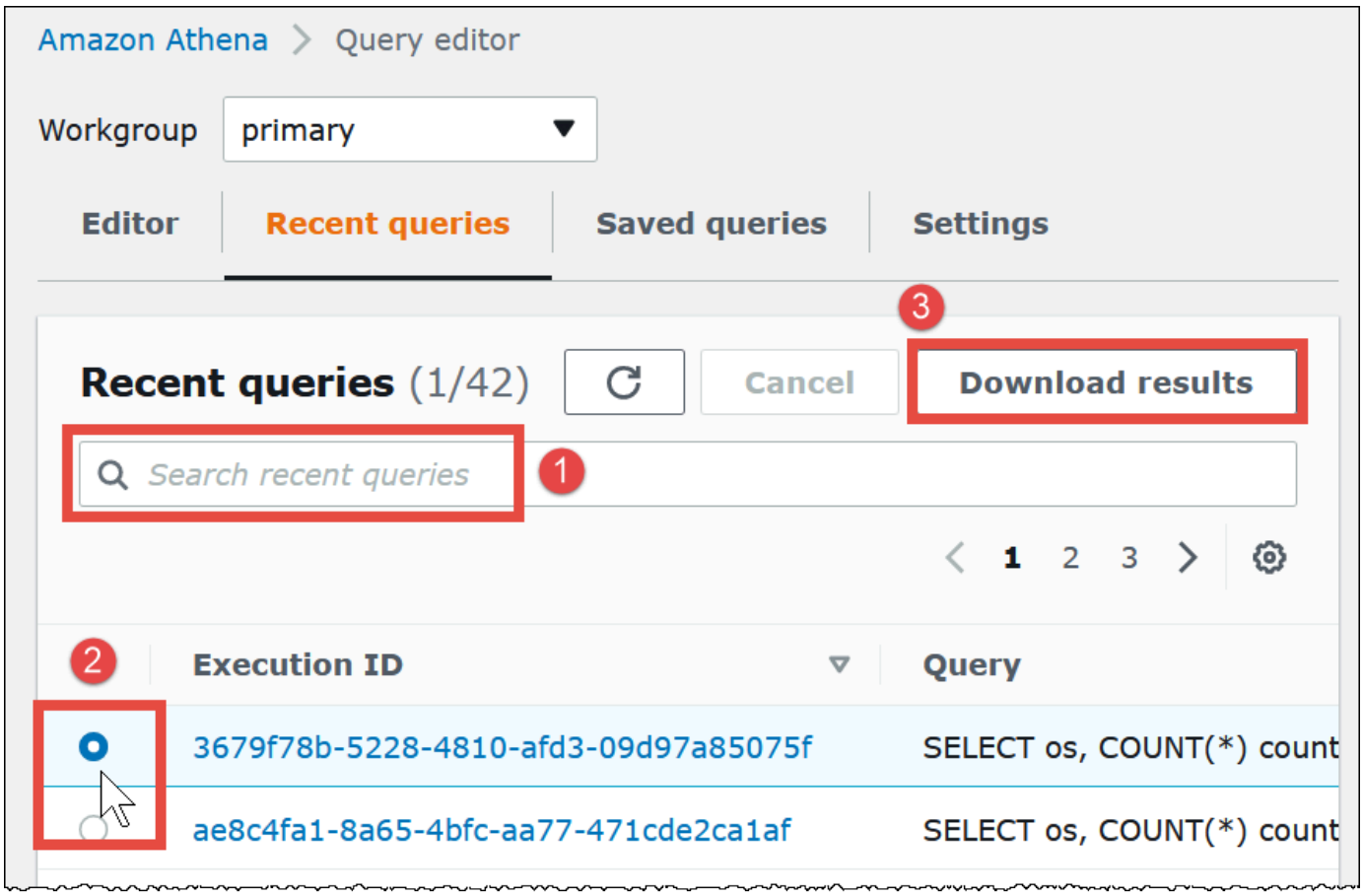
Untuk mengunduh file hasil kueri untuk mengkueri sebelumnya

1. Pilih Kueri terbaru.



2. Gunakan kotak pencarian untuk menemukan kueri, pilih kueri, lalu pilih Unduh hasil.

> **Note**
>
> Anda tidak dapat menggunakan opsi Unduh hasil untuk mengambil hasil kueri yang telah dihapus secara manual, atau mengambil hasil kueri yang telah dihapus atau dipindahkan ke lokasi lain oleh aturan siklus hidup Amazon S3.

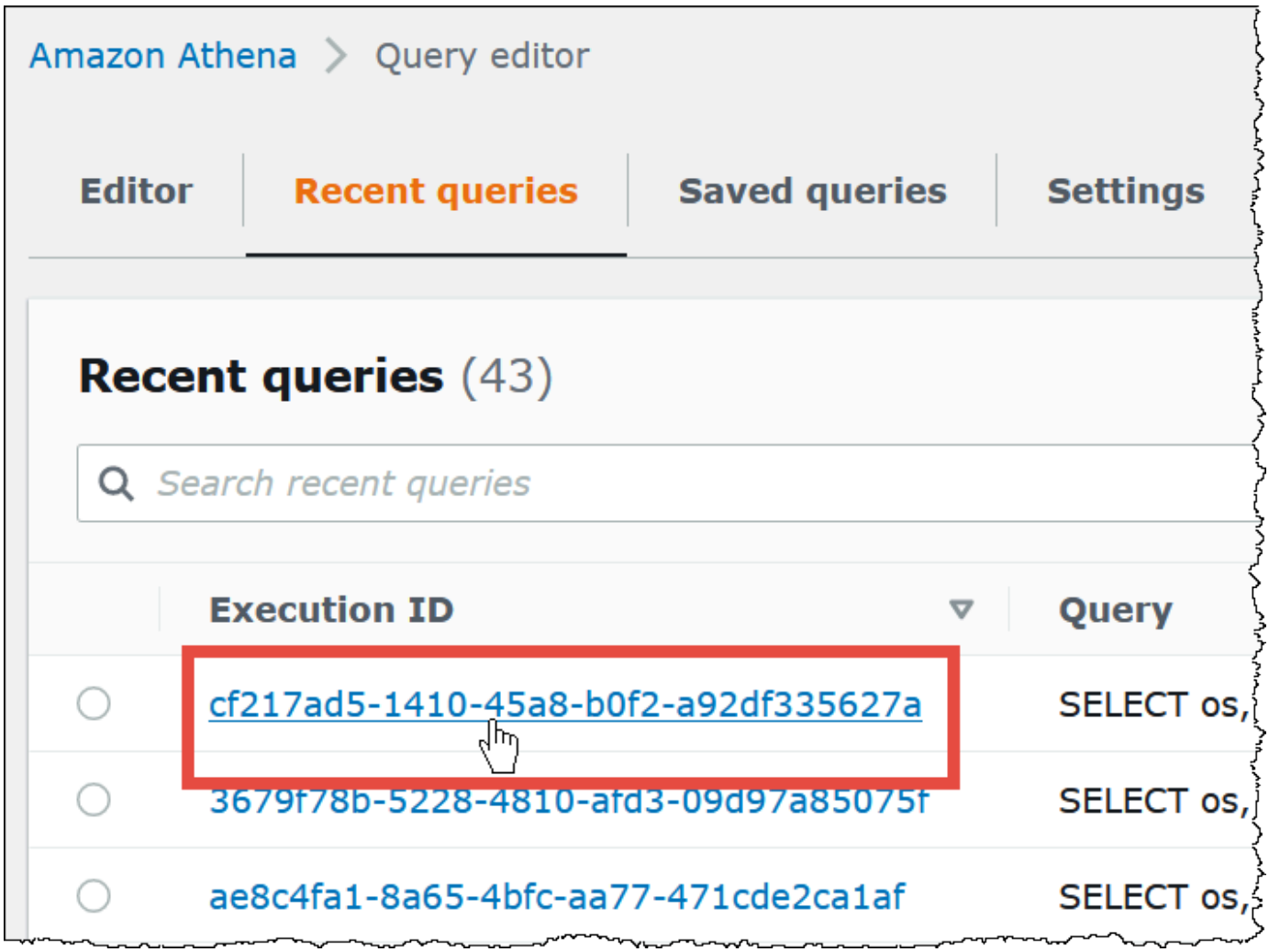


Melihat kueri terbaru

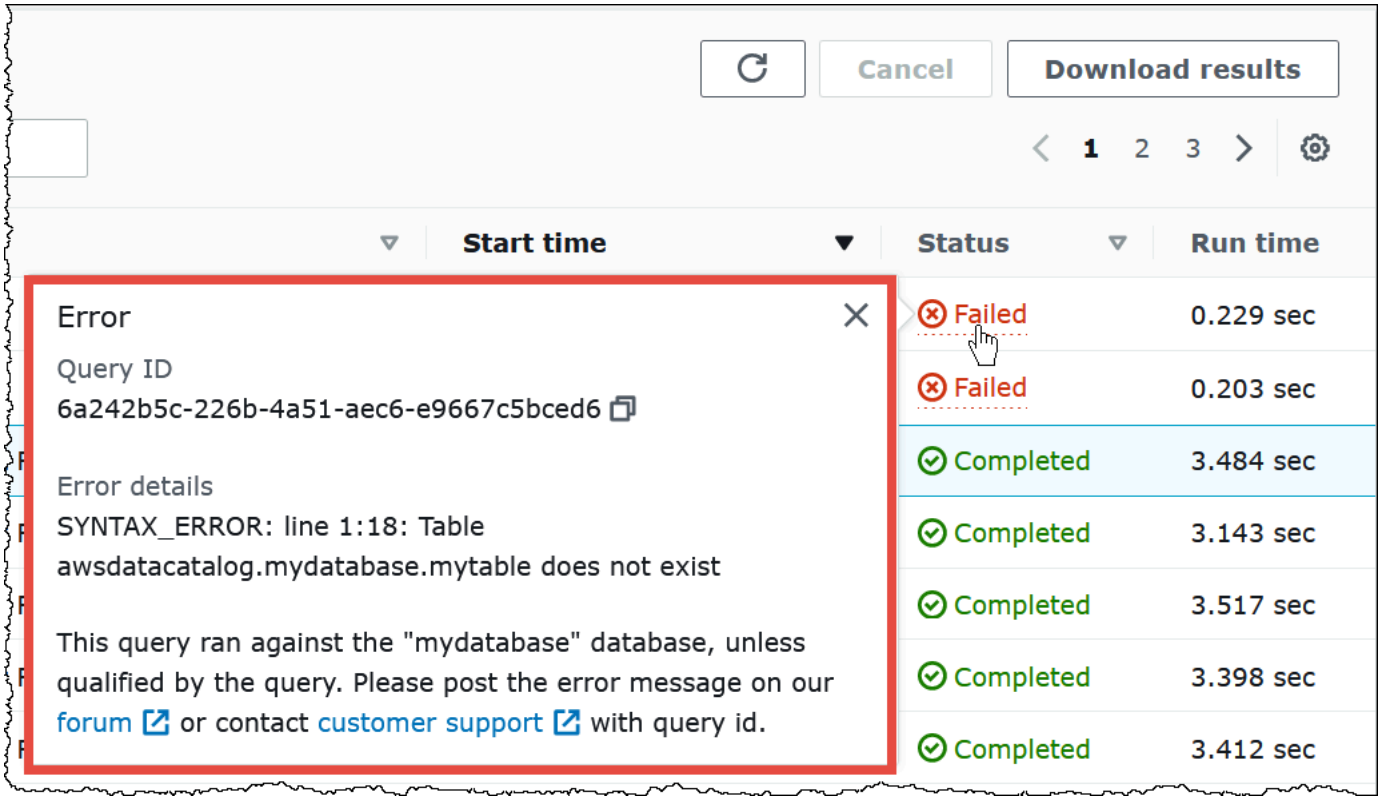
Anda dapat menggunakan konsol Athena untuk melihat kueri mana yang berhasil atau gagal, dan melihat detail kesalahan untuk kueri yang gagal. Athena menyimpan sejarah pertanyaan selama 45 hari.

Untuk melihat kueri terbaru di konsol Athena

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Pilih Kueri terbaru. Tab Kueri terbaru menampilkan informasi tentang setiap kueri yang dijalankan.
3. Untuk membuka pernyataan kueri di editor kueri, pilih ID eksekusi kueri.



4. Untuk melihat detail kueri yang gagal, pilih tautan Gagal untuk kueri.

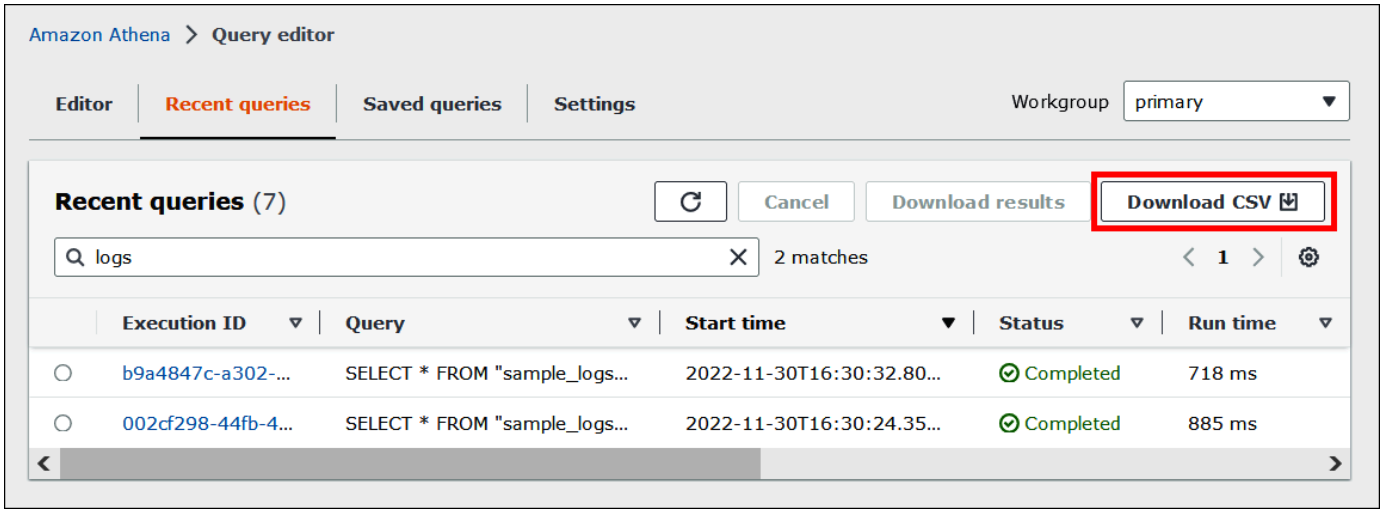


Mengunduh beberapa kueri terbaru ke file CSV

Anda dapat menggunakan tab Kueri terbaru dari konsol Athena untuk mengekspor satu atau beberapa kueri terbaru ke file CSV untuk melihatnya dalam format tabel. File yang diunduh tidak berisi hasil kueri, tetapi string kueri SQL itu sendiri dan informasi lain tentang kueri. Bidang yang diekspor mencakup ID eksekusi, konten string kueri, waktu mulai kueri, status, waktu berjalan, jumlah data yang dipindai, versi mesin kueri yang digunakan, dan metode enkripsi. Anda dapat mengekspor maksimal 500 kueri terbaru, atau maksimum 500 kueri yang difilter menggunakan kriteria yang Anda masukkan di kotak pencarian.

Untuk mengekspor satu atau beberapa kueri terbaru ke file CSV

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Pilih Kueri terbaru.
3. (Opsional) Gunakan kotak pencarian untuk memfilter kueri terbaru yang ingin Anda unduh.
4. Pilih Unduh CSV.



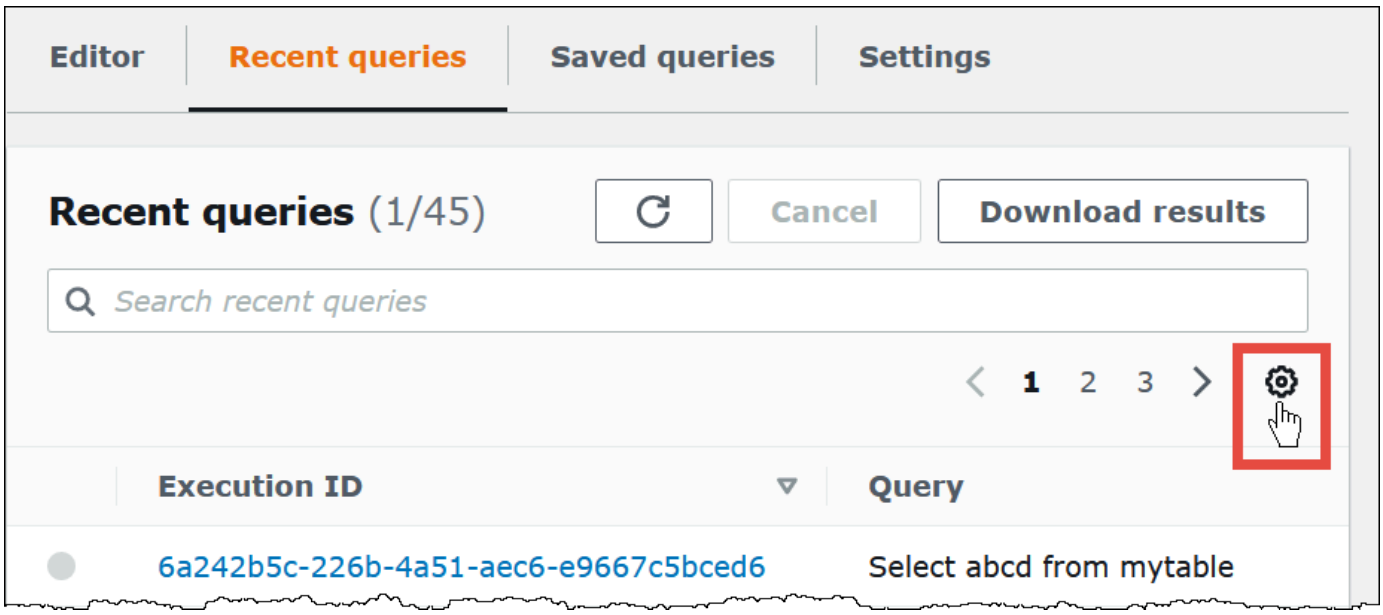
5. Pada prompt penyimpanan file, pilih Simpan. Nama file default `Recent Queries` diikuti oleh stempel waktu (misalnya,) `Recent Queries 2022-12-05T16 04 27.352-08 00.csv`

Mengkonfigurasi opsi tampilan kueri terbaru

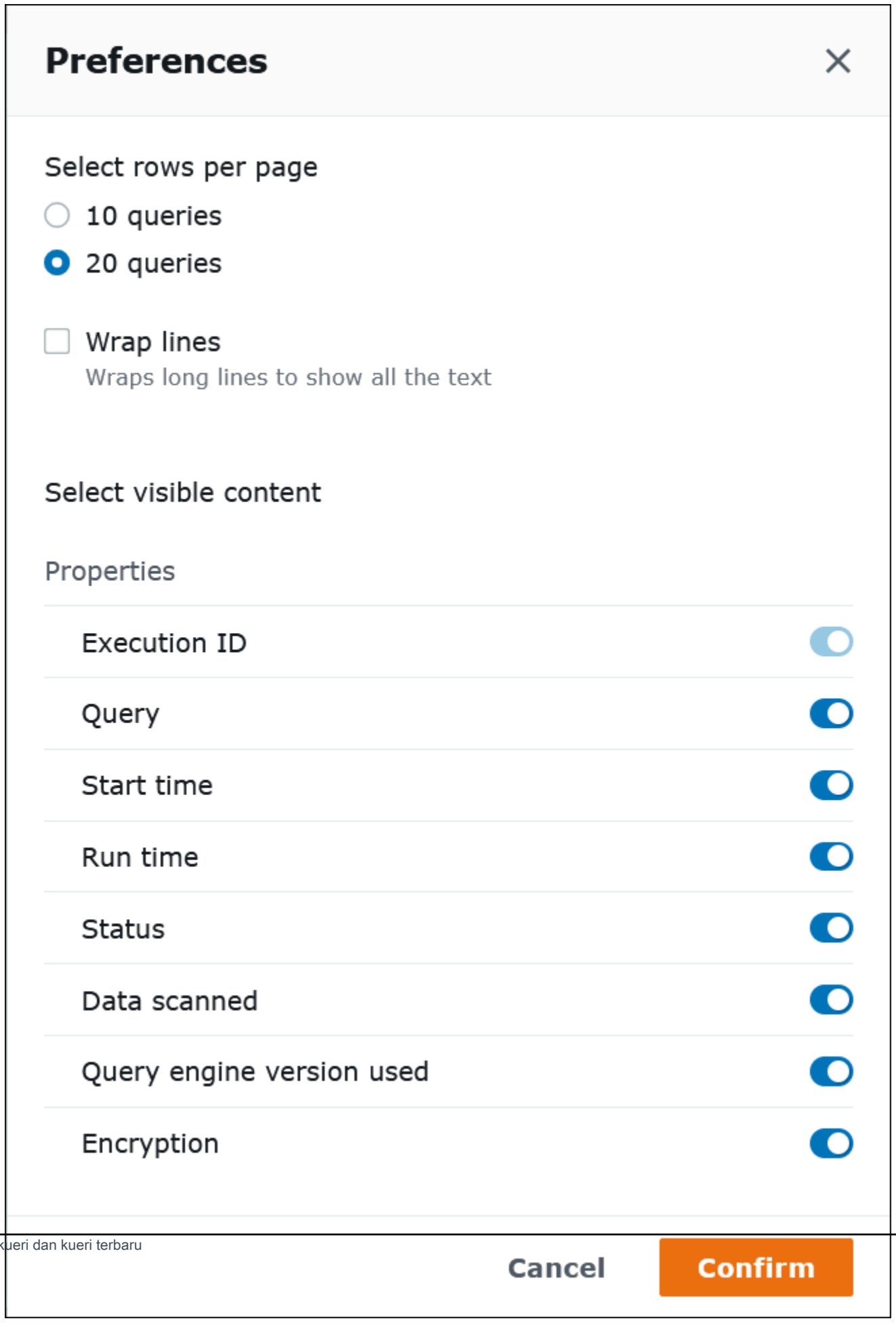
Anda dapat mengonfigurasi opsi untuk tab Kueri terbaru seperti kolom untuk ditampilkan dan pembungkus teks.

Untuk mengonfigurasi opsi untuk tab Kueri terbaru

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Pilih Kueri terbaru.
3. Pilih tombol opsi (ikon roda gigi).



4. Di kotak dialog Preferensi, pilih jumlah baris per halaman, perilaku pembungkus baris, dan kolom yang akan ditampilkan.


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. Pilih Konfirmasi.

Menyimpan riwayat kueri lebih dari 45 hari

Jika Anda ingin menyimpan riwayat kueri lebih lama dari 45 hari, Anda dapat mengambil riwayat kueri dan menyimpannya ke toko data seperti Amazon S3. Untuk mengotomatiskan proses ini, Anda dapat menggunakan Athena dan Amazon S3 API tindakan dan perintah CLI. Prosedur berikut merangkum langkah-langkah ini.

Untuk mengambil dan menyimpan sejarah permintaan pemrograman

1. Gunakan tindakan Athena ListQueryExecutionsAPI atau perintah list-query-executionsCLI untuk mengambil ID kueri.
2. Gunakan tindakan Athena GetQueryExecutionAPI atau perintah get-query-executionCLI untuk mengambil informasi tentang setiap kueri berdasarkan ID-nya.
3. Gunakan tindakan Amazon S3 PutObjectAPI atau perintah CLI put-object untuk menyimpan informasi di Amazon S3.

Menemukan file keluaran kueri di Amazon S3

File output permintaan disimpan dalam sub-folder di Amazon S3 dalam pola jalur berikut kecuali permintaan terjadi dalam kelompok kerja yang konfigurasi menimpa pengaturan sisi klien. Saat konfigurasi grup kerja menimpa pengaturan sisi klien, permintaan menggunakan jalur hasil yang ditentukan oleh kelompok kerja.

```
QueryResultsLocationInS3/[QueryName|Unsaved/yyyy/mm/dd/]
```

- *QueryResultsLocationInS3* adalah lokasi hasil kueri yang ditentukan baik oleh pengaturan workgroup atau pengaturan sisi klien. Untuk informasi selengkapnya, lihat the section called “Menentukan lokasi hasil query” nanti dalam dokumen ini.
- Sub-folder berikut dibuat hanya untuk permintaan menjalankan dari konsol jalur hasil yang belum ditimpa oleh konfigurasi grup kerja. *Kueri yang dijalankan dari AWS CLI atau menggunakan Athena API disimpan langsung ke QueryResultsLocationIn S3.*
  - *QueryName*adalah nama kueri yang hasilnya disimpan. Jika permintaan berjalan tetapi tidak disimpan,Unsaveddigunakan.
  - *yyyy/mm/dd*adalah tanggal bahwa kueri berlari.

File yang diasosiasikan dengan`CREATE TABLE AS SELECT`permintaan disimpan dalam`tables`sub-folder dari pola di atas.

Mengidentifikasi file keluaran kueri

File disimpan ke lokasi hasil kueri di Amazon S3 berdasarkan nama kueri, ID kueri, dan tanggal permintaan berlari. File untuk setiap kueri diberi nama menggunakan*`queryID`*, yang merupakan pengenal unik yang diberikan Athena untuk setiap kueri saat berjalan.

Tipe fail berikut disimpan:

Tipe file	Pola penamaan file	Deskripsi
File hasil kueri	*`QueryID`*`.csv` *`QueryID`*`.txt`	File hasil kueri DDLL disimpan dalam format nilai yang dipisahkan koma (CSV). Hasil kueri DDL disimpan sebagai file teks biasa. Anda dapat mengunduh berkas hasil dari konsol dariHasilsaat menggunakan konsol atau dari queryriwayat. Untuk informasi selengkap nya, lihat Mengunduh file hasil kueri menggunakan konsol Athena.
Kueri file metadata	*`QueryID`*`.csv.metadata` *`QueryID`*`.txt.metadata`	DDLL dan DDL kueri file metadata disimpan dalam format biner dan tidak dapat dibaca manusia. Ekstensi file sesuai dengan file hasil kueri terkait. Athena menggunakan metadata saat membaca hasil kueri menggunakan`GetQueryResults` Tindakan. Meski file-

Tipe file	Pola penamaan file	Deskripsi
		file ini dapat dihapus, kami tidak merekomendasikanny a karena informasi penting tentang kueri hilang.
File manifes data	*QueryID*-manifest.csv	File manifest data dibuat untuk melacak file yang dibuat Athena di lokasi sumber data Amazon S3 saatINSERT INTOberjalan kueri. Jika permintaan gagal, manifest juga melacak file yang permintaan dimaksudk an untuk menulis. Manifest berguna untuk mengident ifikasi file yatim piatu yang dihasilkan dari permintaan gagal.

Menggunakan AWS CLI untuk mengidentifikasi lokasi dan file keluaran kueri

Untuk menggunakan AWS CLI untuk mengidentifikasi lokasi output query dan file hasil, jalankan `aws athena get-query-execution` perintah, seperti pada contoh berikut. Ganti*abc1234d-5efg-67hi-jklm-89n0op12qr34*dengan ID kueri.

```
aws athena get-query-execution --query-execution-id abc1234d-5efg-67hi-
jklm-89n0op12qr34
```

Perintah ini menghasilkan output serupa dengan berikut: Untuk deskripsi dari setiap parameter output, lihat get-query-executiondi AWS CLI Command Reference.

```
{
    "QueryExecution": {
        "Status": {
            "SubmissionDateTime": 1565649050.175,
            "State": "SUCCEEDED",
```

```
            "CompletionDateTime": 1565649056.6229999
        },
        "Statistics": {
            "DataScannedInBytes": 5944497,
            "DataManifestLocation": "s3://DOC-EXAMPLE-BUCKET/athena-query-
results-123456789012-us-west-1/MyInsertQuery/2019/08/12/abc1234d-5efg-67hi-
jklm-89n0op12qr34-manifest.csv",
            "EngineExecutionTimeInMillis": 5209
        },
        "ResultConfiguration": {
            "EncryptionConfiguration": {
                "EncryptionOption": "SSE_S3"
            },
            "OutputLocation": "s3://DOC-EXAMPLE-BUCKET/athena-query-
results-123456789012-us-west-1/MyInsertQuery/2019/08/12/abc1234d-5efg-67hi-
jklm-89n0op12qr34"
        },
        "QueryExecutionId": "abc1234d-5efg-67hi-jklm-89n0op12qr34",
        "QueryExecutionContext": {},
        "Query": "INSERT INTO mydb.elb_log_backup SELECT * FROM mydb.elb_logs LIMIT
 100",
        "StatementType": "DML",
        "WorkGroup": "primary"
    }
}
```

Menggunakan kembali hasil kueri

Saat Anda menjalankan ulang kueri di Athena, Anda dapat memilih untuk menggunakan kembali hasil kueri terakhir yang disimpan. Opsi ini dapat meningkatkan kinerja dan mengurangi biaya dalam hal jumlah byte yang dipindai. Menggunakan kembali hasil kueri berguna jika, misalnya, Anda tahu bahwa hasilnya tidak akan berubah dalam jangka waktu tertentu. Anda dapat menentukan usia maksimum untuk menggunakan kembali hasil kueri. Athena menggunakan hasil yang disimpan selama tidak lebih tua dari usia yang Anda tentukan. Untuk informasi selengkapnya, lihat Mengurangi biaya dan meningkatkan kinerja kueri dengan Amazon Athena di Blog AWS Big Data.

> **Note**
>
> Fitur penggunaan kembali hasil kueri memerlukan mesin Athena versi 3. Untuk informasi tentang mengubah versi mesin, lihatMengubah versi mesin Athena.

Fitur utama

- Menggunakan kembali hasil kueri adalah fitur keikutsertaan per kueri. Anda dapat mengaktifkan penggunaan kembali hasil kueri berdasarkan per kueri.
- Usia maksimum untuk menggunakan kembali hasil kueri dapat ditentukan dalam hitungan menit, jam, atau hari. Usia maksimum yang ditentukan adalah setara dengan 7 hari terlepas dari satuan waktu yang digunakan. Default-nya adalah 60 menit.
- Saat Anda mengaktifkan penggunaan kembali hasil untuk kueri, Athena mencari eksekusi kueri sebelumnya dalam grup kerja yang sama. Jika Athena menemukan hasil kueri tersimpan yang sesuai, itu tidak menjalankan kembali kueri, tetapi menunjuk ke lokasi hasil sebelumnya atau mengambil data darinya.

- Untuk kueri apa pun yang memungkinkan opsi penggunaan kembali hasil, Athena menggunakan kembali hasil kueri terakhir yang disimpan ke folder workgroup hanya jika semua kondisi berikut benar:
  - String kueri adalah sama persis.
  - Database dan nama katalog cocok.
  - Hasil sebelumnya tidak lebih dari usia maksimum yang ditentukan, atau tidak lebih dari 60 menit jika usia maksimum belum ditentukan.
  - Athena hanya menggunakan kembali eksekusi yang memiliki konfigurasi hasil yang sama persis dengan eksekusi saat ini.
  - Anda memiliki akses ke semua tabel yang direferensikan dalam kueri.
  - Anda memiliki akses ke lokasi file S3 tempat hasil sebelumnya disimpan.

Jika salah satu kondisi ini tidak terpenuhi, Athena menjalankan kueri tanpa menggunakan hasil cache.

Pertimbangan dan batasan

Saat menggunakan fitur penggunaan kembali hasil kueri, ingatlah hal-hal berikut:

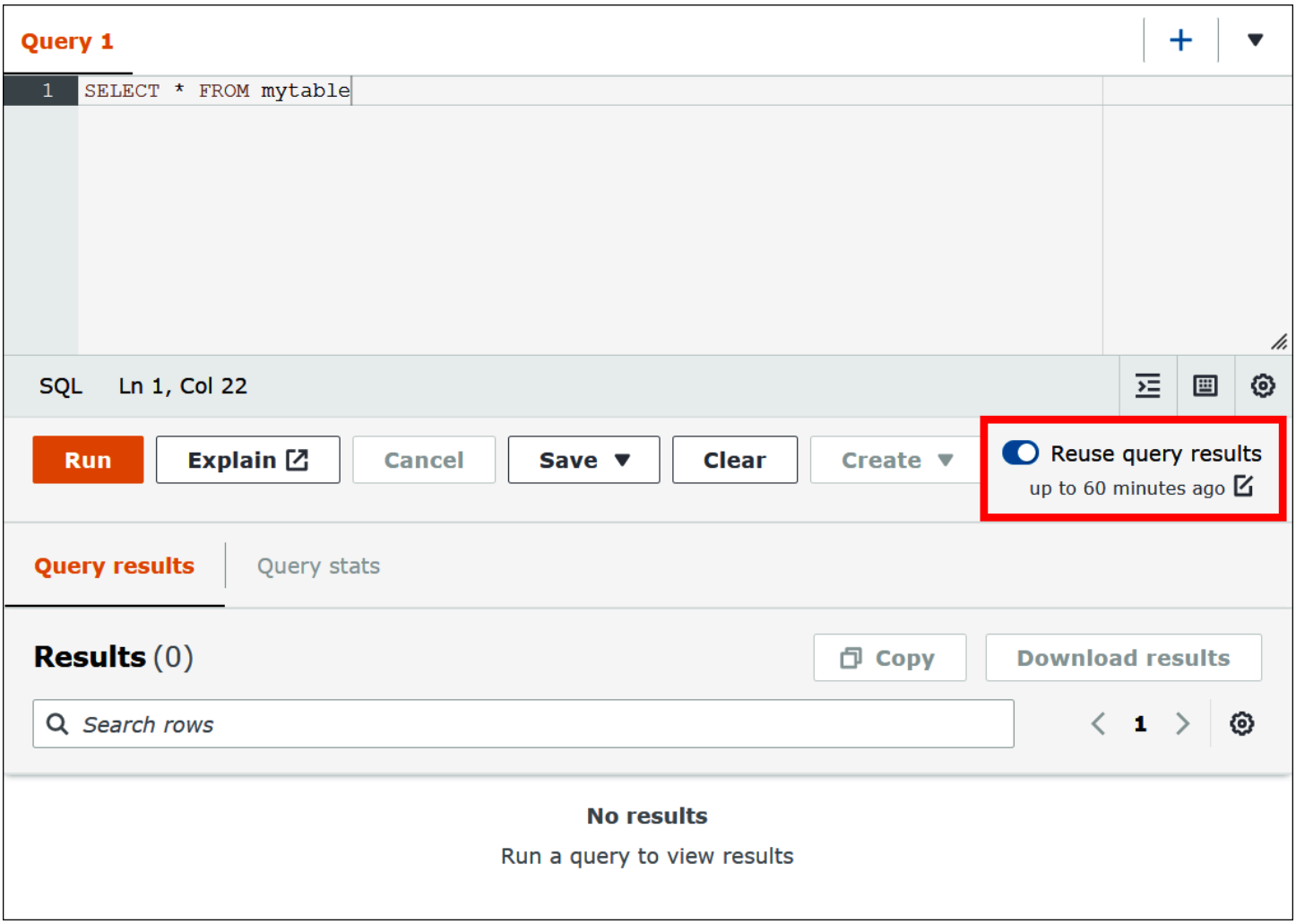
- Athena menggunakan kembali hasil kueri hanya dalam kelompok kerja yang sama.
- Fitur hasil kueri penggunaan kembali menghormati konfigurasi workgroup. Jika Anda mengganti konfigurasi hasil untuk kueri, fitur tersebut dinonaktifkan.

- Apache Hive, Apache Hudi, Apache Iceberg, dan Linux Foundation Delta Lake tabel terdaftar dengan didukung. AWS Glue Metastor Sarang Eksternal tidak didukung.
- Kueri yang mereferensikan katalog federasi atau metastore Hive eksternal tidak didukung.
- Penggunaan kembali hasil kueri tidak didukung untuk tabel yang diatur Lake Formation.
- Penggunaan kembali hasil kueri tidak didukung saat lokasi Amazon S3 dari sumber tabel terdaftar sebagai lokasi data di Lake Formation.
- Tabel dengan izin baris dan kolom tidak didukung.
- Tabel yang memiliki kontrol akses berbutir halus (misalnya, pemfilteran kolom atau baris) tidak didukung.
- Setiap kueri yang mereferensikan tabel yang tidak didukung tidak memenuhi syarat untuk digunakan kembali hasil kueri.
- Athena mengharuskan Anda memiliki izin baca Amazon S3 untuk file keluaran yang dihasilkan sebelumnya untuk digunakan kembali.
- Fitur hasil kueri penggunaan kembali mengasumsikan bahwa konten hasil sebelumnya belum dimodifikasi. Athena tidak memeriksa integritas hasil sebelumnya sebelum menggunakannya.
- Jika hasil kueri dari eksekusi sebelumnya telah dihapus atau dipindahkan ke lokasi yang berbeda di Amazon S3, eksekusi kueri yang sama berikutnya tidak akan menggunakan kembali hasil kueri.
- Hasil yang berpotensi basi dapat dikembalikan. Athena tidak memeriksa perubahan data sumber hingga usia penggunaan ulang maksimum yang Anda tentukan telah tercapai.
- Jika beberapa hasil tersedia untuk digunakan kembali, Athena menggunakan hasil terbaru.
- Kueri yang menggunakan operator non-deterministik atau fungsi seperti `rand()` atau `shuffle()` tidak menggunakan hasil cache. Misalnya, `LIMIT` tanpa `ORDER BY` non-deterministik dan tidak di-cache, tetapi `LIMIT` dengan deterministik dan `ORDER BY` di-cache.
- Penggunaan kembali hasil kueri didukung di konsol Athena, di Athena API, dan di driver JDBC. Saat ini, dukungan driver ODBC untuk penggunaan kembali hasil kueri hanya tersedia untuk Windows.
- Untuk menggunakan fitur penggunaan kembali hasil kueri dengan JDBC, versi driver minimum yang diperlukan adalah 2.0.34.1000. Untuk ODBC, versi driver minimum yang diperlukan adalah 1.1.19.1002. Untuk informasi unduhan driver, lihatMenghubungkan ke Amazon Athena dengan driver ODBC dan JDBC.
- Penggunaan kembali hasil kueri tidak didukung untuk kueri yang menggunakan lebih dari satu katalog data.

- Penggunaan kembali hasil kueri tidak didukung untuk kueri yang menyertakan lebih dari 20 tabel.

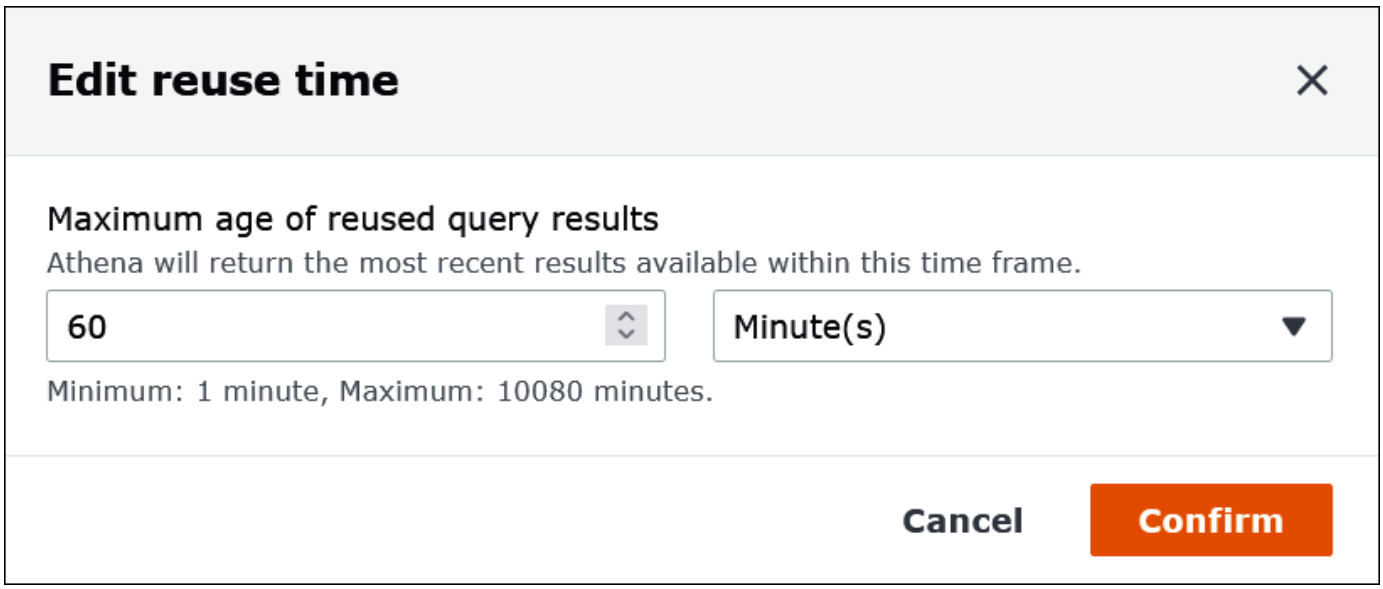
Menggunakan kembali hasil kueri di konsol Athena

Untuk menggunakan fitur ini, aktifkan opsi Gunakan kembali hasil kueri di editor kueri Athena.

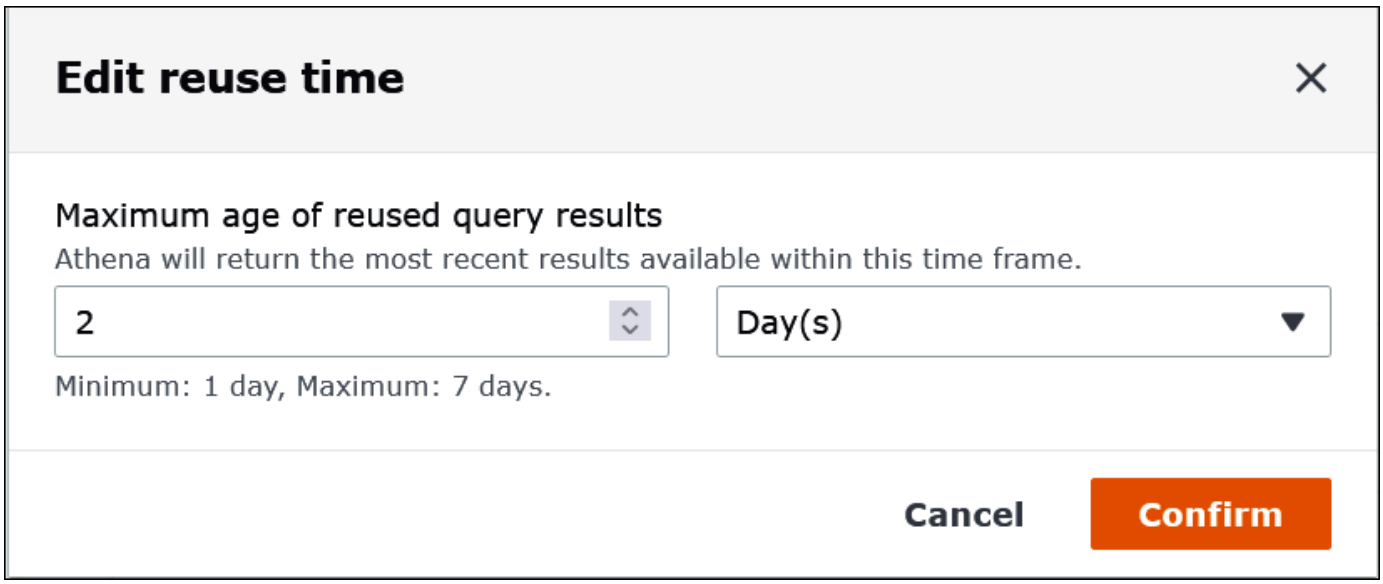


Untuk mengonfigurasi fitur hasil kueri penggunaan kembali

1. Di editor kueri Athena, di bawah opsi Gunakan kembali hasil kueri, pilih ikon edit di samping hingga 60 menit yang lalu.
2. Dalam kotak dialog Edit waktu penggunaan kembali, dari kotak di sebelah kanan, pilih unit waktu (menit, jam, atau hari).
3. Di kotak di sebelah kiri, masukkan atau pilih jumlah satuan waktu yang ingin Anda tentukan. Waktu maksimum yang dapat Anda masukkan setara dengan tujuh hari terlepas dari unit waktu yang dipilih.

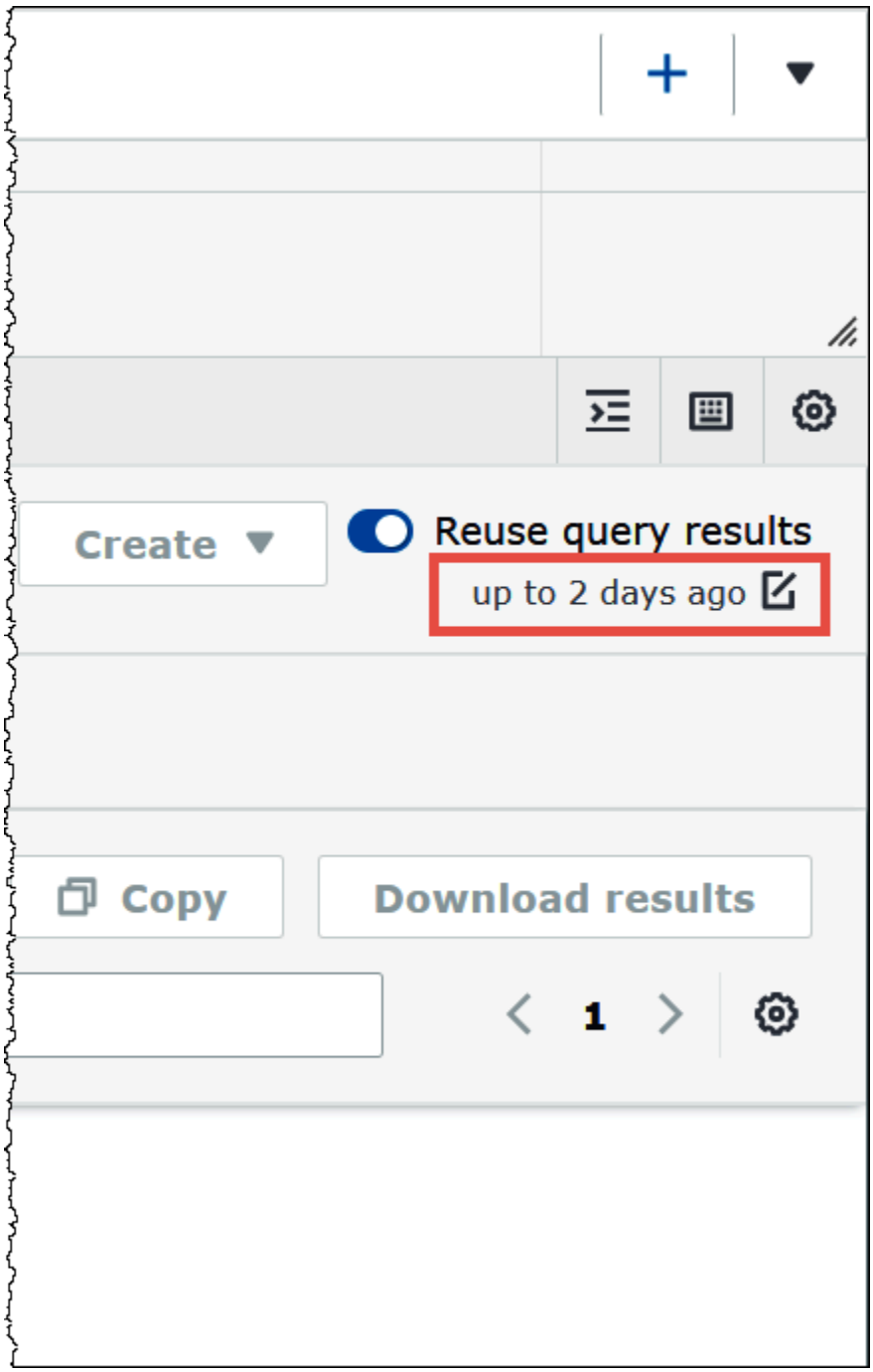


Contoh berikut menentukan waktu penggunaan kembali maksimum dua hari.



4. Pilih Konfirmasi.

   Spanduk mengonfirmasi perubahan konfigurasi Anda, dan opsi Reuse query results menampilkan pengaturan baru Anda.

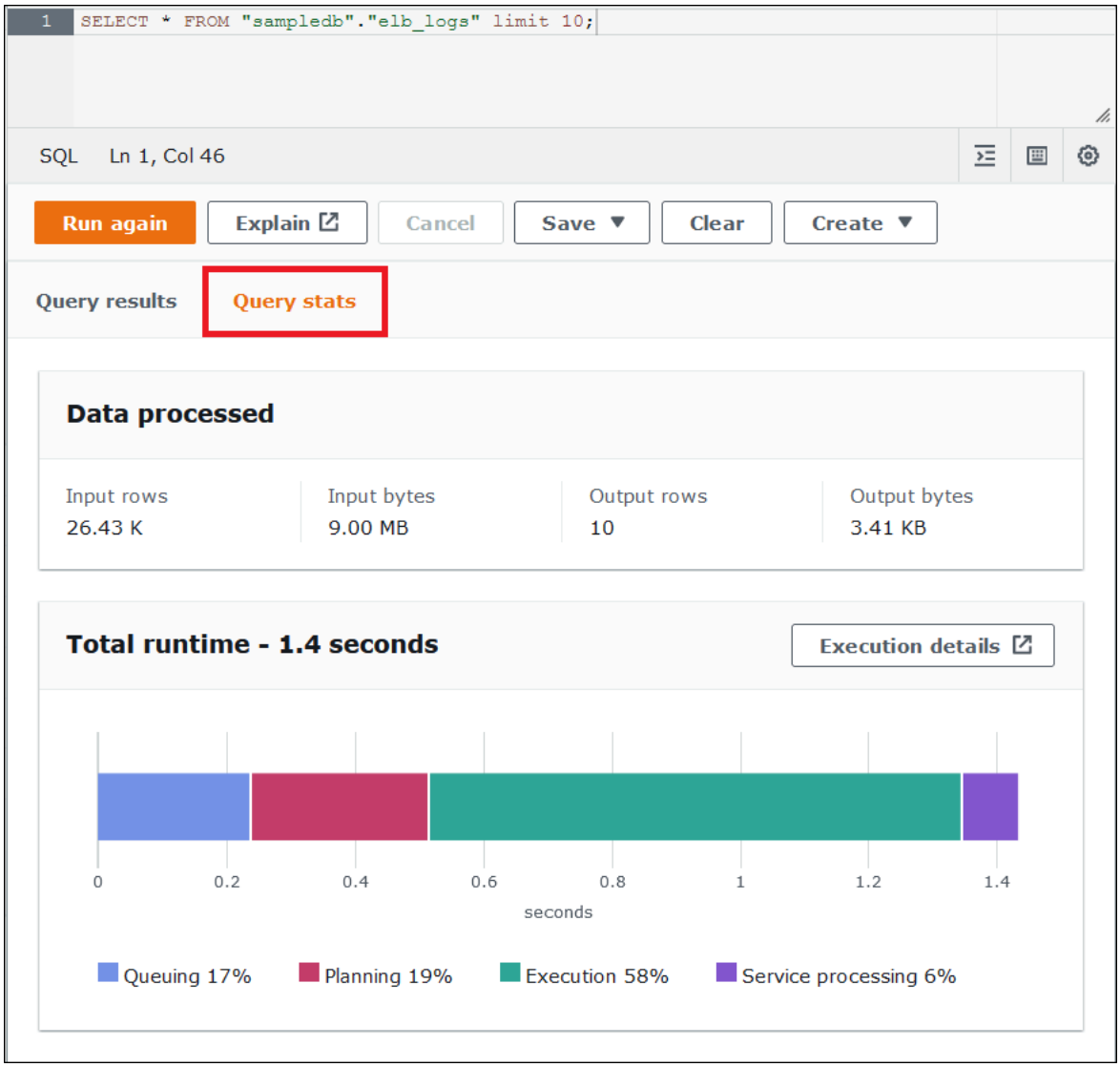


Melihat statistik dan detail eksekusi untuk kueri yang diselesaikan

Setelah menjalankan kueri, Anda bisa mendapatkan statistik tentang data input dan output yang diproses, melihat representasi grafis dari waktu yang dibutuhkan untuk setiap fase kueri, dan mengeksplorasi detail eksekusi secara interaktif.

Untuk melihat statistik kueri untuk kueri yang telah selesai

1. Setelah Anda menjalankan kueri di editor kueri Athena, pilih tab Statistik kueri.



Tab Query stats menyediakan informasi berikut:

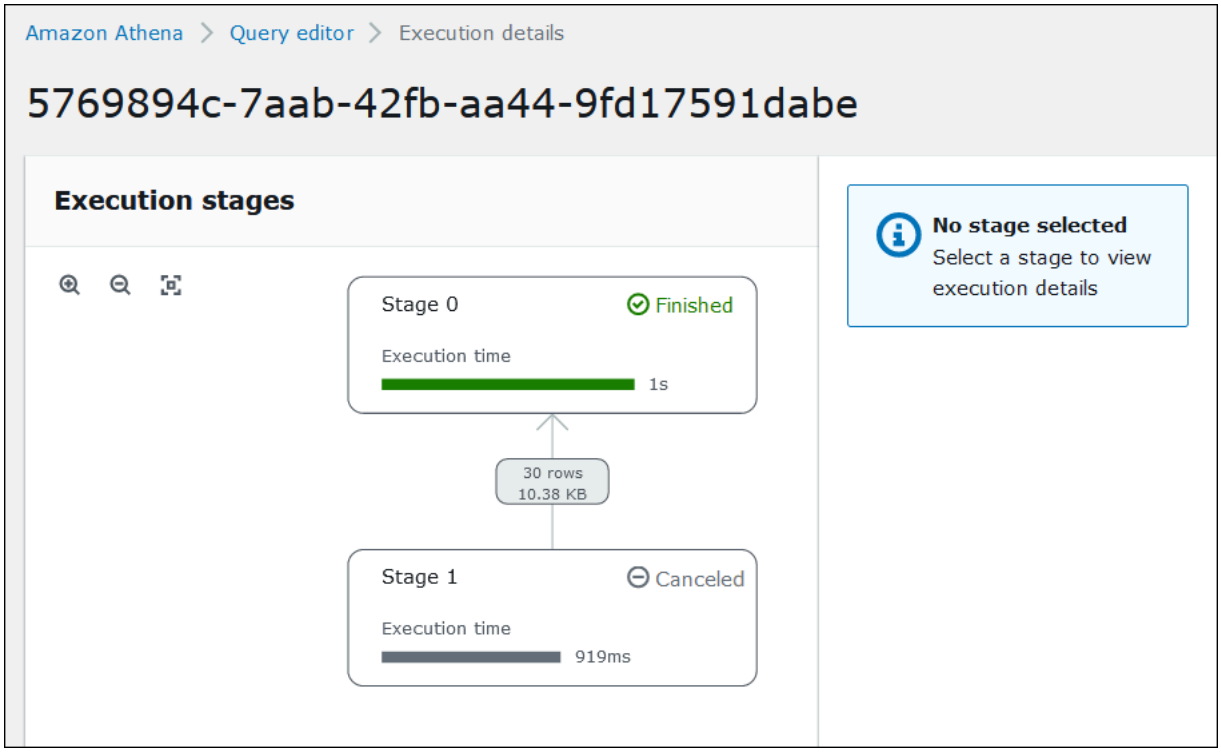
- Data diproses - Menunjukkan jumlah baris input dan byte yang diproses, dan jumlah baris dan byte output.
- Total runtime — Menunjukkan jumlah total waktu yang dibutuhkan kueri untuk dijalankan dan representasi grafis dari berapa banyak waktu yang dihabiskan untuk antrian, perencanaan, eksekusi, dan pemrosesan layanan.

> **Note**
>
> Masukan tingkat tahap dan jumlah baris keluaran dan informasi ukuran data tidak ditampilkan ketika kueri memiliki filter tingkat baris yang ditentukan dalam Lake Formation.

2. Untuk mengeksplorasi informasi secara interaktif tentang bagaimana kueri dijalankan, pilih Detail eksekusi.

Halaman detail Eksekusi menunjukkan ID eksekusi untuk kueri dan grafik tahapan berbasis nol dalam kueri. Tahapan yang diperintahkan mulai selesai dari bawah ke atas. Label setiap tahap menunjukkan jumlah waktu yang dibutuhkan panggung untuk berjalan.

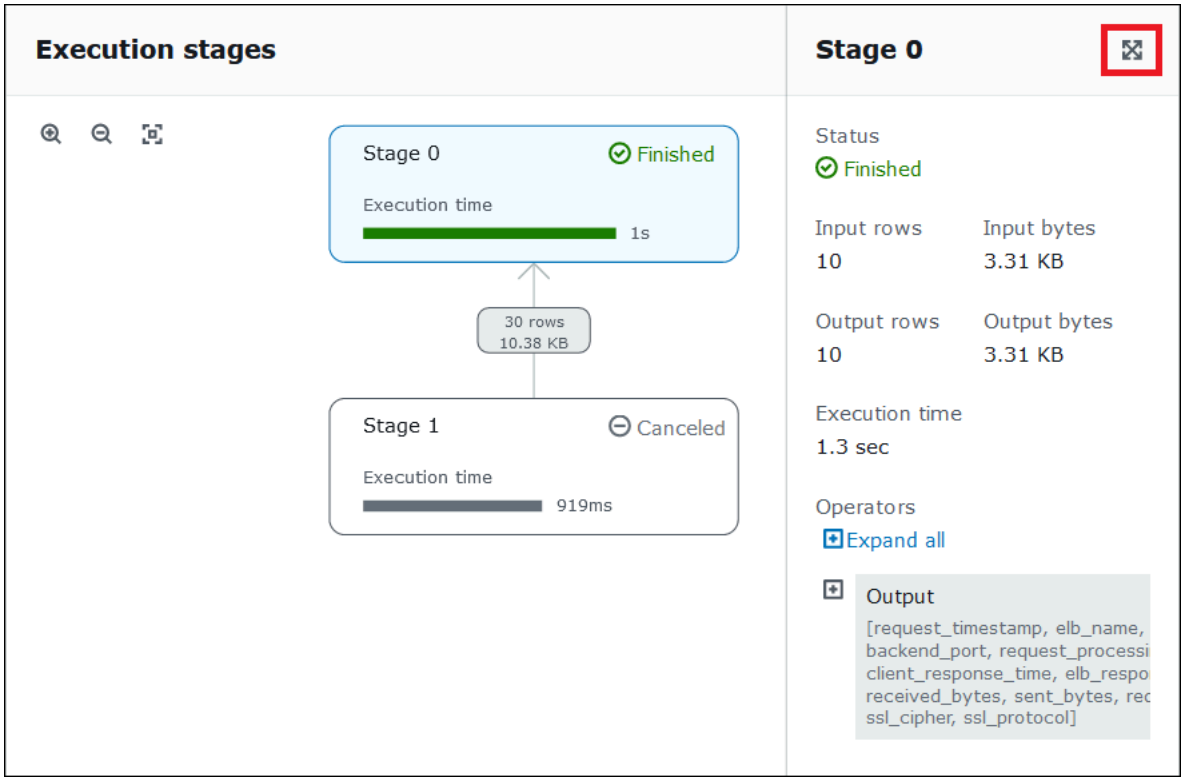
> **Note**
>
> Total runtime dan waktu tahap eksekusi kueri seringkali berbeda secara signifikan. Misalnya, kueri dengan total runtime dalam menit dapat menunjukkan waktu eksekusi untuk tahap dalam jam. Karena tahap adalah unit logis komputasi yang dieksekusi secara paralel di banyak tugas, waktu eksekusi tahap adalah waktu eksekusi agregat dari semua tugasnya. Terlepas dari perbedaan ini, waktu eksekusi tahap dapat berguna sebagai indikator relatif tahap mana yang paling intensif secara komputasi dalam kueri.



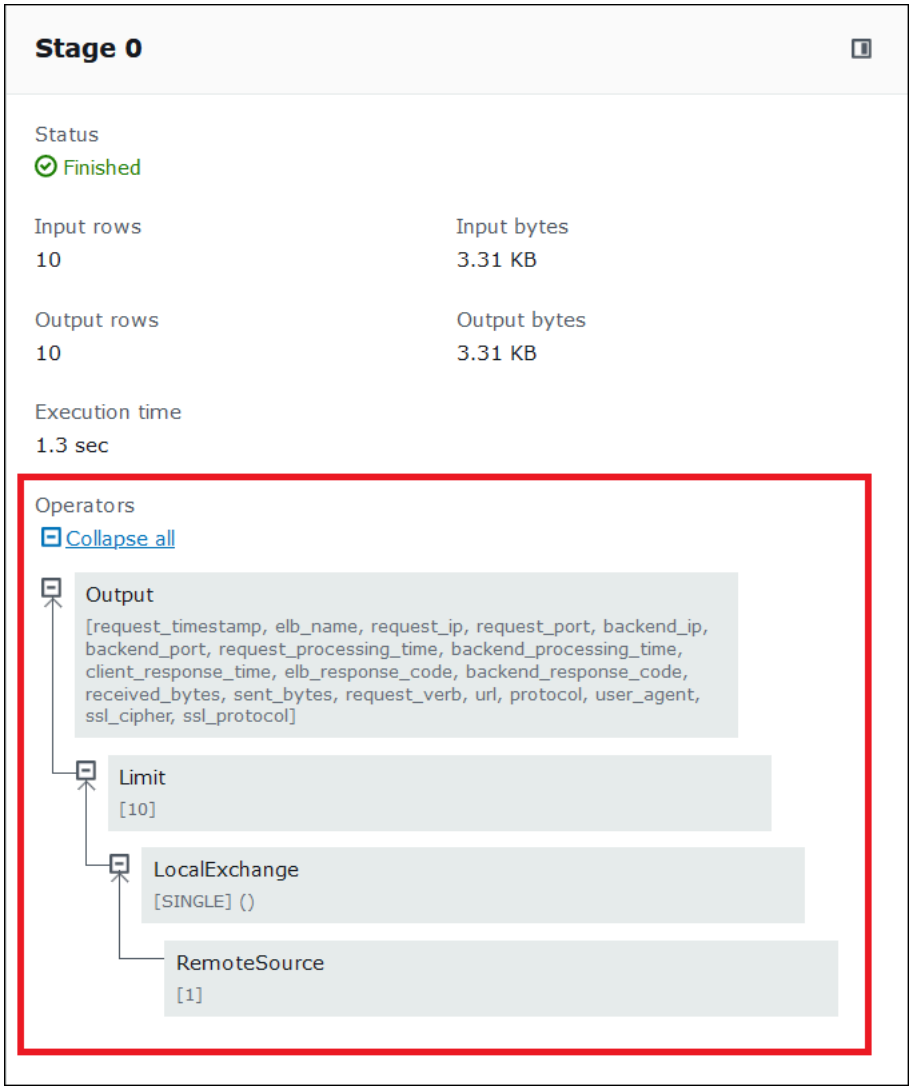
Untuk menavigasi grafik, gunakan opsi berikut:

- Untuk memperbesar atau memperkecil, gulir mouse, atau gunakan ikon pembesar.
- Untuk menyesuaikan grafik agar sesuai dengan layar, pilih ikon Zoom to fit.
- Untuk memindahkan grafik, seret penunjuk mouse.

3. Untuk melihat detail lebih lanjut untuk sebuah panggung, pilih panggung. Panel detail tahap di sebelah kanan menunjukkan jumlah baris dan byte input dan output, dan pohon operator.



4. Untuk melihat detail panggung lebar penuh, pilih ikon perluas di kanan atas panel detail.
5. Untuk mendapatkan informasi tentang bagian-bagian panggung, perluas satu atau lebih item di pohon operator.



Untuk informasi selengkapnya tentang detail eksekusi, lihatMemahami Athena MENJELASKAN hasil pernyataan.

Sumber daya tambahan

Untuk informasi selengkapnya, lihat sumber daya berikut ini.

Melihat rencana eksekusi untuk kueri SQL

Menggunakan JELASKAN dan JELASKAN ANALISIS di Athena

Bekerja dengan pandangan

Pemandangan di Amazon Athena adalah tabel logis, bukan tabel fisik. Kueri yang mendefinisikan tampilan berjalan setiap kali tampilan direferensikan dalam kueri.

Anda dapat membuat tampilan dari`SELECT`query kemudian referensi tampilan ini di masa depan kueri. Untuk informasi selengkapnya, lihat CREATE VIEW.

Topik

Kapan menggunakan tampilan?

Anda mungkin ingin membuat tampilan untuk:

- Kueri subset data. Misalnya, Anda dapat membuat tampilan dengan subset kolom dari tabel asli untuk menyederhanakan data kueri.
- Menggabungkan beberapa tabel dalam satu kueri. Jika Anda memiliki beberapa tabel dan ingin menggabungkannya dengan`UNION ALL`, Anda dapat membuat tampilan dengan ekspresi yang untuk menyederhanakan kueri terhadap tabel gabungan.
- Menyembunyikan kompleksitas kueri dasar yang ada dan menyederhanakan kueri yang dijalankan oleh pengguna. kueri dasar sering termasuk bergabung antara tabel, ekspresi dalam daftar kolom, dan sintaks SQL lain yang membuatnya sulit untuk memahami dan debug mereka. Anda dapat membuat tampilan yang menyembunyikan kompleksitas dan menyederhanakan kueri.
- Bereksperimenlah dengan teknik optimasi dan buat kueri. Misalnya, jika Anda menemukan kombinasi`WHERE`Kondisi,`JOIN`pesanan, atau ekspresi lain yang menunjukkan performa terbaik, Anda dapat membuat tampilan dengan klausa dan ekspresi ini. Aplikasi kemudian dapat membuat kueri yang relatif sederhana terhadap tampilan ini. Jika nanti Anda menemukan cara yang lebih

baik untuk mengoptimalkan kueri asli, saat Anda membuat ulang tampilan, semua aplikasi segera memanfaatkan kueri dasar yang dioptimalkan.

- Menyembunyikan mendasari tabel dan kolom nama, dan meminimalkan masalah pemeliharaanjika nama-nama itu berubah. Dalam hal ini, Anda menciptakan tampilan menggunakan nama baru. Semua kueri yang menggunakan tampilan daripada tabel yang mendasari tetap berjalan dengan tidak ada perubahan.

Tindakan yang didukung untuk tampilan di Athena

Athena mendukung tindakan berikut untuk dilihat. Anda dapat menjalankan perintah ini di editor kueri.

Pernyataan	Deskripsi
CREATE VIEW	Menciptakan tampilan baru dari yang ditentukan`SELECT`Query. Untuk informasi selengkapnya, lihat Membuat tampilan. Opsional`OR REPLACE`klausa memungkinkan Anda memperbarui tampilan yang ada dengan menggantinya.
DESCRIBE VIEW	Menunjukkan daftar kolom untuk tampilan bernama. Ini memungkinkan Anda untuk memeriksa atribut dari tampilan yang kompleks.
DROP VIEW	Menghapus tampilan yang ada. Opsional`IF EXISTS`klausa menekan kesalahan jika tampilan tidak ada.
SHOW CREATE VIEW	Menunjukkan pernyataan SQL yang menciptakan tampilan tertentu.
SHOW VIEWS	Daftar tampilan dalam basis data tertentu, atau dalam basis data saat ini jika Anda menghilangkan nama basis data. Gunakan pilihan`LIKE`klausul dengan ekspresi reguler untuk membatasi daftar nama tampilan. Anda juga dapat melihat daftar tampilan di panel kiri di konsol.
SHOW COLUMNS	Daftar kolom dalam skema untuk tampilan.

Pertimbangan untuk tampilan

Pertimbangan berikut berlaku untuk membuat dan menggunakan tampilan di Athena:

- Di Athena, Anda dapat melihat pratinjau dan bekerja dengan tampilan yang dibuat di Konsol Athena, di AWS Glue Data Catalog, jika Anda telah bermigrasi untuk menggunakannya, atau dengan Presto berjalan di klaster EMR Amazon yang terhubung ke katalog yang sama. Anda tidak dapat melihat pratinjau atau menambahkan ke tampilan Athena yang dibuat dengan cara lain.
- Jika Anda telah membuat tampilan Athena dalam Katalog Data, Katalog Data memperlakukan tampilan sebagai tabel. Anda dapat menggunakan tingkat tabel kontrol akses halus dalam Katalog Data untukMembatasi aksesuntuk tampilan ini.
- Athena mencegah Anda menjalankan tampilan rekursif dan menampilkan pesan kesalahan dalam kasus tersebut. Tampilan rekursif adalah tampilan kueri yang referensi itu sendiri.
- Athena menampilkan pesan kesalahan saat mendeteksi tampilan basi. Tampilan basi dilaporkan saat salah satu hal berikut terjadi:
  - Tampilan referensi tabel atau basis data yang tidak ada.
  - Perubahan skema atau metadata dibuat dalam tabel direferensikan.
  - Sebuah tabel direferensikan dijatuhkan dan diciptakan dengan skema yang berbeda atau konfigurasi.
- Anda dapat membuat dan menjalankan tampilan nest selama kueri di balik tampilan nest berlaku dan tabel dan basis data ada.

Batasan untuk tampilan

- Nama tampilan Athena tidak dapat berisi karakter khusus, selain garis bawah(`_`). Untuk informasi selengkapnya, lihat Nama untuk tabel, database, dan kolom.
- Hindari menggunakan kata kunci reserved untuk penamaan tampilan. Jika Anda menggunakan kata kunci yang dipesan, gunakan tanda kutip ganda untuk menyertakan kata kunci yang dicadangkan di kueri Anda pada tampilan. Lihat Kata kunci terpesan.
- Anda tidak dapat menggunakan tampilan yang dibuat di Athena dengan metastor Hive eksternal atau UDF. Untuk informasi tentang bekerja dengan tampilan yang dibuat secara eksternal di Hive, lihat. Bekerja dengan tampilan Hive
- Anda tidak dapat menggunakan tampilan dengan fungsi geospasial.
- Anda tidak dapat menggunakan tampilan untuk mengelola kontrol akses pada data di Amazon S3. Untuk kueri tampilan, Anda perlu izin untuk mengakses data yang tersimpan di Amazon S3. Untuk informasi selengkapnya, lihat Akses ke Amazon S3 dari Athena.
- Meskipun kueri tampilan di seluruh akun didukung di mesin Athena versi 2 dan mesin Athena versi 3, Anda tidak dapat membuat tampilan yang menyertakan akun silang. AWS Glue Data Catalog

Untuk informasi tentang akses katalog data lintas akun, lihatAkses lintas akun ke katalog AWS Glue data.

- Kolom metadata tersembunyi Sarang atau Gunung Es`$bucket`,, `$file_modified_time``$file_size`, dan tidak didukung untuk tampilan `$partition` di Athena. Untuk informasi tentang menggunakan kolom `$path` metadata di Athena, lihat. Mendapatkan lokasi file untuk data sumber di Amazon S3

Bekerja dengan tampilan di konsol

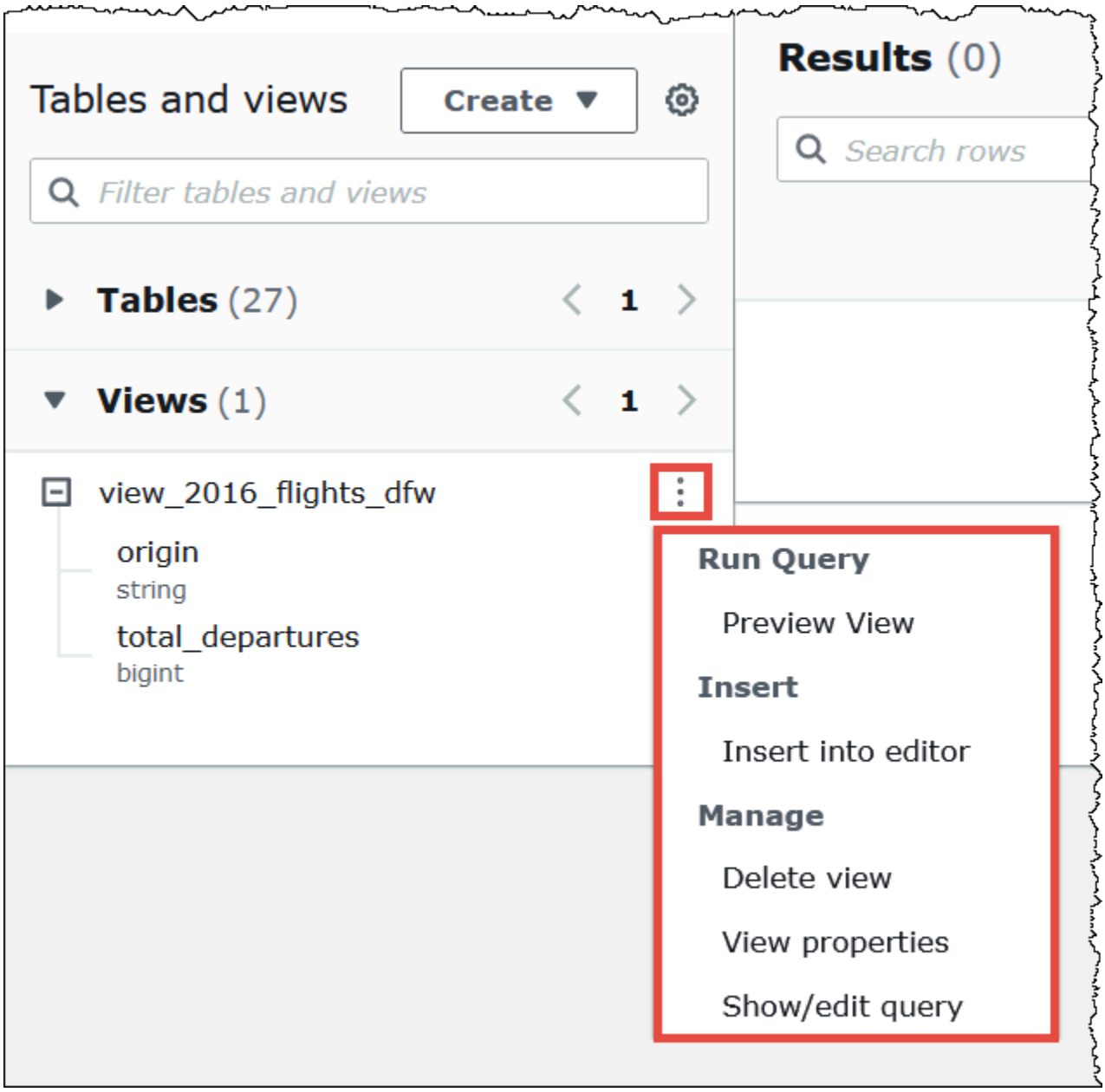
Di konsol Athena, Anda dapat:

- Cari semua tampilan di sebelah kiri, tempat tabel tercantum.
- Tampilan filter.
- Pratinjau tampilan, menampilkan propertinya, mengeditnya, atau menghapusnya.

Untuk menampilkan tindakan untuk tampilan

Tampilan ditampilkan di konsol hanya jika Anda telah membuatnya.

1. Di konsol Athena, pilih Tampilan, lalu pilih tampilan untuk memperluasnya dan tampilkan kolom dalam tampilan.
2. Pilih tiga titik vertikal di sebelah tampilan untuk menampilkan daftar tindakan untuk tampilan.



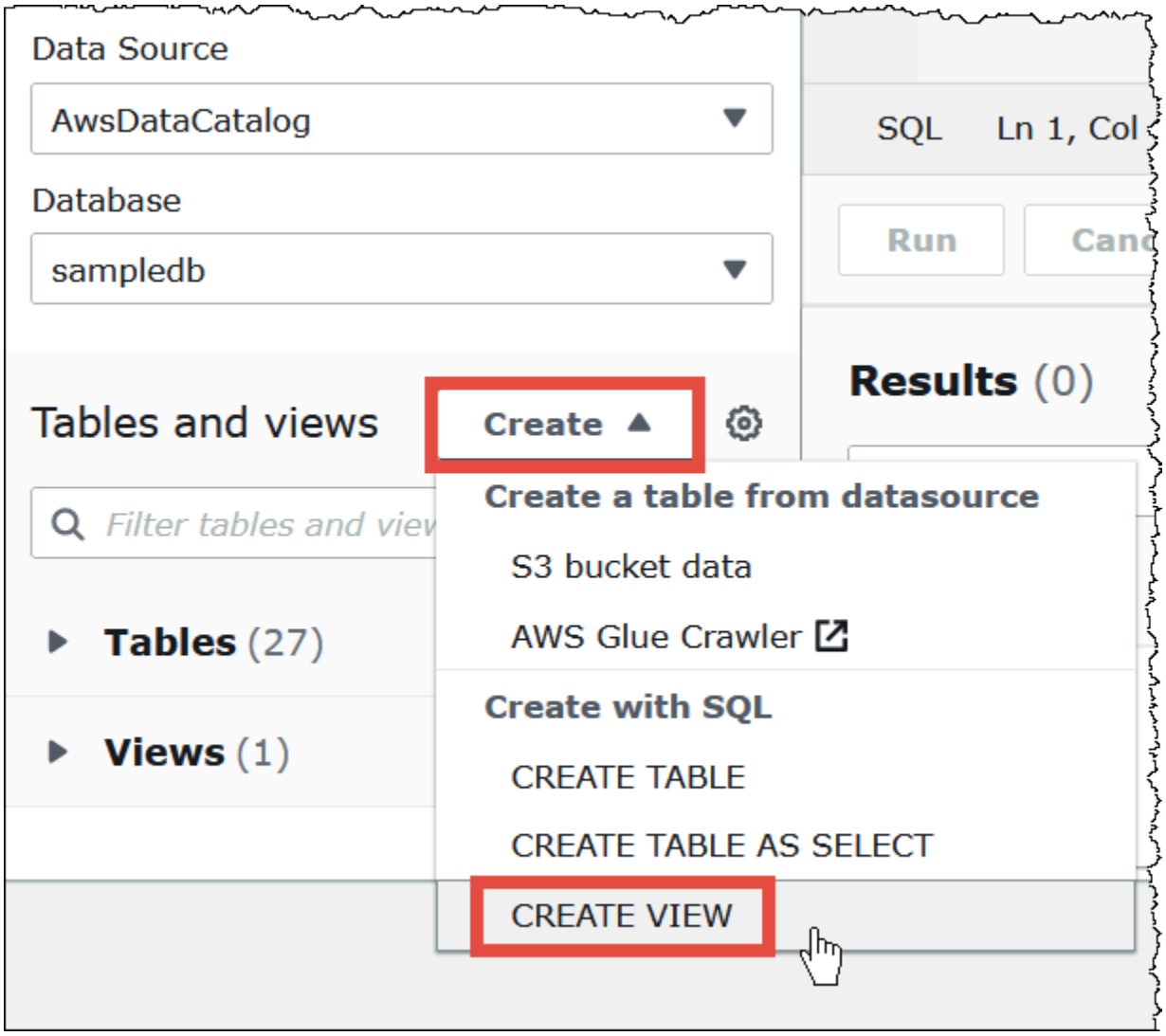
3. Pilih tindakan untuk melihat pratinjau tampilan, masukkan nama tampilan ke editor kueri, hapus tampilan, lihat properti tampilan, atau tampilkan dan edit tampilan di editor kueri.

Membuat tampilan

Anda dapat membuat tampilan di konsol Athena dengan menggunakan templat atau dengan menjalankan kueri yang ada.

Untuk menggunakan template untuk membuat tampilan

1. Di konsol Athena, di samping Tabel dan tampilan, pilih Buat, lalu pilih Buat tampilan.



   Tindakan ini menempatkan template tampilan yang dapat diedit ke dalam editor kueri.

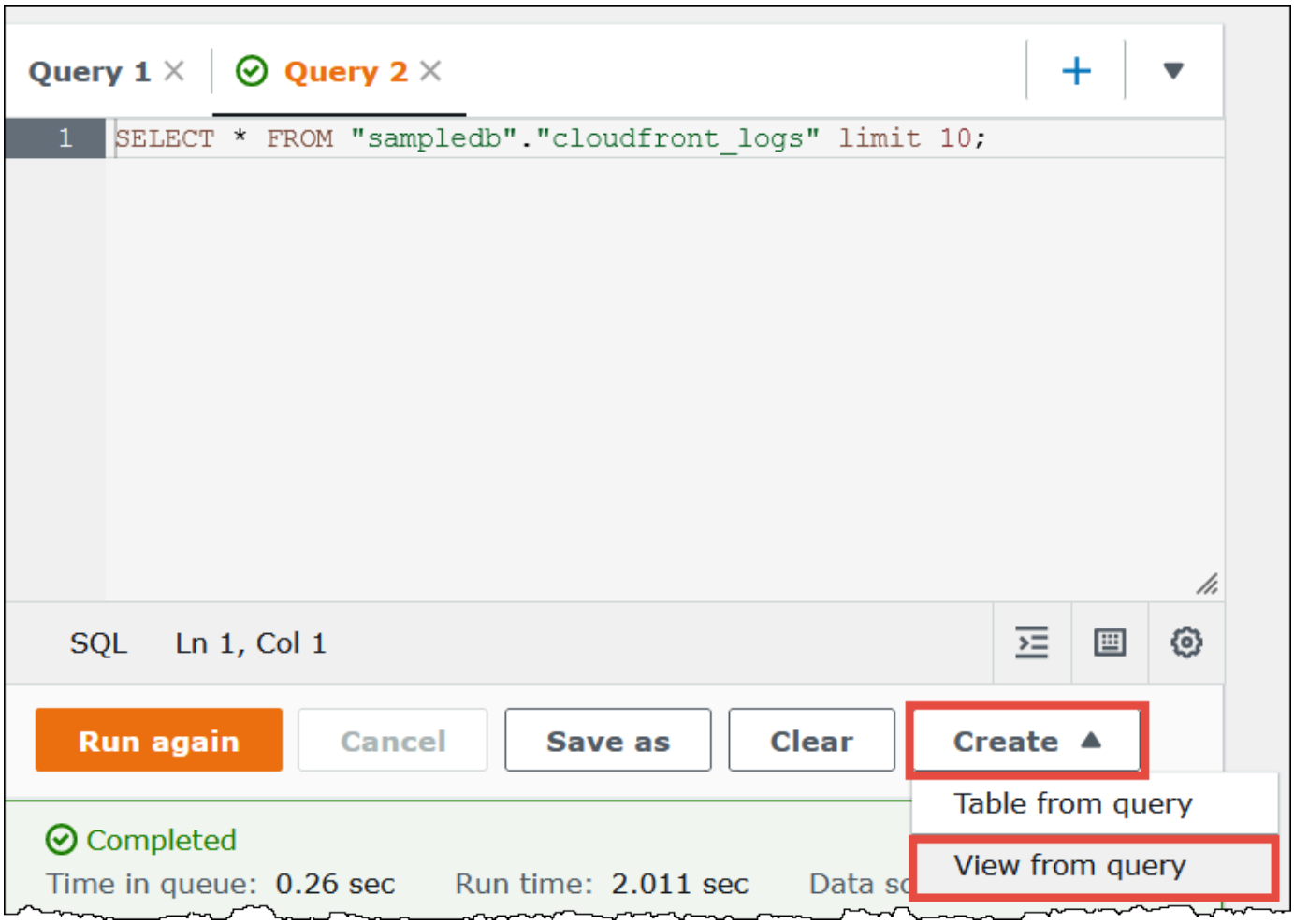
2. Edit template tampilan sesuai dengan kebutuhan Anda. Saat Anda memasukkan nama untuk tampilan dalam pernyataan, ingatlah bahwa nama tampilan tidak dapat berisi karakter khusus selain garis bawah (`_`). Lihat Nama untuk tabel, database, dan kolom. Hindari penggunaan Kata kunci terpesan untuk memberi nama tampilan.

   Untuk informasi selengkapnya tentang membuat tampilan, lihat CREATE VIEW danContoh tampilan.

3. Pilih Jalankan untuk membuat tampilan. Tampilan muncul dalam daftar tampilan di konsol Athena.

Untuk membuat tampilan dari kueri yang ada

1. Gunakan editor kueri Athena untuk menjalankan kueri yang ada.
2. Di bawah jendela editor kueri, pilih Buat, lalu pilih Lihat dari kueri.



3. Di kotak dialog Buat Tampilan, masukkan nama untuk tampilan, lalu pilih Buat. Nama tampilan tidak dapat berisi karakter khusus selain garis bawah(_). Lihat Nama untuk tabel, database, dan kolom. Hindari penggunaan Kata kunci terpesan untuk memberi nama tampilan.

   Athena menambahkan tampilan ke daftar tampilan di konsol dan menampilkan `CREATE VIEW` pernyataan untuk tampilan di editor kueri.

Catatan

- Jika Anda menghapus tabel yang menjadi dasar tabel dan kemudian mencoba menjalankan tampilan, Athena menampilkan pesan kesalahan.

- Anda dapat membuat tampilan nest, yang merupakan tampilan di atas tampilan yang ada. Athena mencegah Anda menjalankan tampilan rekursif yang referensi itu sendiri.

Contoh tampilan

Untuk menampilkan sintaks tampilan kueri, gunakanSHOW CREATE VIEW.

Example Contoh 1

Pertimbangkan dua tabel berikut: tabelemployeesdengan dua kolom,iddanname, dan tabelsalaries, dengan dua kolom,iddansalary.

Dalam contoh ini, kami membuat tampilan bernamaname_salarysebagaiSELECTpermintaan yang memperoleh daftar ID dipetakan ke gaji dari tabelemployeesdansalaries:

```
CREATE VIEW name_salary AS
SELECT
 employees.name,
 salaries.salary
FROM employees, salaries
WHERE employees.id = salaries.id
```

Example Contoh 2

Dalam contoh berikut, kami membuat tampilan bernamaview1yang memungkinkan Anda untuk menyembunyikan sintaks kueri yang lebih kompleks.

Tampilan ini berjalan di atas dua tabel,table1dantable2, tempat setiap tabel adalah berbedaSELECTQuery. Tampilan memilih kolom daritable1dan bergabung dengan hasil dengantable2. Gabung didasarkan pada kolomayang hadir di kedua tabel.

```
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;
```

Untuk informasi tentang menanyakan tampilan gabungan, lihat. Menanyakan pandangan federasi

Menggunakan AWS Glue Data Catalog tampilan

Fitur ini dalam rilis pratinjau dan dapat berubah. Untuk informasi selengkapnya, lihat bagian Beta dan Pratinjau di dokumen Ketentuan AWS Layanan.

Gunakan AWS Glue Data Catalog tampilan saat Anda menginginkan satu tampilan umum Layanan AWS seperti Amazon Athena dan Amazon Redshift. Dalam tampilan Katalog Data, izin akses ditentukan oleh pengguna yang membuat tampilan, bukan pengguna yang menanyakan tampilan. Metode pemberian izin ini disebut semantik definer.

Kasus penggunaan berikut menunjukkan bagaimana Anda dapat menggunakan tampilan Katalog Data.

- Kontrol akses yang lebih besar — Anda membuat tampilan yang membatasi akses data berdasarkan tingkat izin yang dibutuhkan pengguna. Misalnya, Anda dapat menggunakan tampilan Katalog Data untuk mencegah karyawan yang tidak bekerja di departemen sumber daya manusia (SDM) melihat informasi yang dapat diidentifikasi secara pribadi.
- Pastikan catatan lengkap — Dengan menerapkan filter tertentu ke tampilan Katalog Data Anda, Anda memastikan bahwa catatan data dalam tampilan Katalog Data selalu lengkap.
- Keamanan yang ditingkatkan — Dalam tampilan Katalog Data, definisi kueri yang membuat tampilan harus utuh agar tampilan dibuat. Hal ini membuat tampilan Data Catalog kurang rentan terhadap perintah SQL dari aktor jahat.
- Mencegah akses ke tabel yang mendasari — Semantik definer memungkinkan pengguna mengakses tampilan tanpa membuat tabel yang mendasarinya tersedia bagi mereka. Hanya pengguna yang mendefinisikan tampilan yang memerlukan akses ke tabel.

Definisi tampilan Katalog Data disimpan dalam format AWS Glue Data Catalog. Ini berarti Anda dapat menggunakan AWS Lake Formation untuk memberikan akses melalui hibah sumber daya,

hibah kolom, atau kontrol akses berbasis tag. Untuk informasi selengkapnya tentang pemberian dan pencabutan akses di Lake Formation, lihat Memberikan dan mencabut izin pada sumber daya Katalog Data di Panduan Pengembang.AWS Lake Formation

Izin

Tampilan Katalog Data memerlukan tiga peran:`Lake Formation Admin`,`Definer`, dan`Invoker`.

- **`Lake Formation Admin`**— Memiliki akses untuk mengonfigurasi semua izin Lake Formation.
- **`Definer`**— Membuat tampilan Katalog Data. `Definer`Peran harus memiliki `SELECT` izin penuh yang dapat diberikan pada semua tabel yang mendasari referensi definisi tampilan.
- **`Invoker`**— Dapat menanyakan tampilan Katalog Data atau memeriksa metadatanya. Untuk menampilkan pemanggil kueri, Anda dapat menggunakan fungsi `invoker_principal()` DHTML. Untuk informasi selengkapnya, lihat invoker_principal ().

Hubungan kepercayaan `Definer` peran harus memungkinkan `sts:AssumeRole` tindakan untuk kepala layanan AWS Glue dan Lake Formation, seperti pada contoh berikut.

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

Izin IAM untuk akses Athena juga diperlukan. Untuk informasi selengkapnya, lihat AWS kebijakan terkelola untuk Amazon Athena.

Batasan

- Tampilan Katalog Data tidak dapat mereferensikan tampilan lain.

- Anda dapat mereferensikan hingga 10 tabel dalam definisi tampilan.
- Tabel yang mendasari harus terdaftar di Lake Formation.
- `DEFINER`Kepala sekolah hanya dapat menjadi peran IAM.
- `DEFINER`Peran harus memiliki izin penuh `SELECT` (dapat diberikan) pada tabel yang mendasarinya.
- `UNPROTECTED`Tampilan Katalog Data tidak didukung.
- Fungsi yang ditentukan pengguna (UDF) tidak didukung dalam definisi tampilan.
- Sumber data federasi Athena tidak dapat digunakan dalam tampilan Katalog Data.
- Tampilan Katalog Data tidak didukung untuk Hive metastor eksternal.
- Athena menampilkan pesan kesalahan saat mendeteksi tampilan basi. Tampilan basi dilaporkan saat salah satu hal berikut terjadi:
  - Tampilan referensi tabel atau basis data yang tidak ada.
  - Perubahan skema atau metadata dibuat dalam tabel direferensikan.
  - Sebuah tabel direferensikan dijatuhkan dan diciptakan dengan skema yang berbeda atau konfigurasi.

Membuat tampilan Katalog Data

Contoh sintaks berikut menunjukkan bagaimana pengguna `Definer` peran menciptakan tampilan `orders_by_date` Data Catalog. Contoh mengasumsikan bahwa `Definer` peran memiliki `SELECT` izin penuh pada `orders` tabel dalam database. `default`

```
CREATE PROTECTED MULTI DIALECT VIEW orders_by_date
SECURITY DEFINER
AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
WHERE order_city = 'SEATTLE'
GROUP BY orderdate
```

Menanyakan tampilan Katalog Data

Setelah tampilan dibuat, `Lake Formation Admin` dapat memberikan `SELECT` izin pada tampilan Katalog Data ke `Invoker` prinsipal. `Invoker`Prinsipal kemudian dapat menanyakan tampilan tanpa memiliki akses ke tabel dasar yang mendasari yang direferensikan oleh tampilan. Berikut ini adalah contoh `Invoker` query.

```
SELECT * from orders_by_date where price > 5000
```

Memperbarui tampilan Katalog Data

`Lake Formation AdminAtau Definer` dapat menggunakan `ALTER VIEW UPDATE DIALECT` sintaks untuk memperbarui definisi tampilan. Contoh berikut memodifikasi definisi tampilan untuk memilih kolom dari `returns` tabel bukan `orders` tabel.

```
ALTER VIEW orders_by_date UPDATE DIALECT
AS
SELECT return_date, sum(totalprice) AS price
FROM returns
WHERE order_city = 'SEATTLE'
GROUP BY orderdate
```

Untuk informasi selengkapnya tentang sintaks untuk membuat dan mengelola tampilan Katalog Data, lihatGlue Data Catalog tampilan sintaks.

Glue Data Catalog tampilan sintaks

> Fitur ini dalam rilis pratinjau dan dapat berubah. Untuk informasi selengkapnya, lihat bagian Beta dan Pratinjau di dokumen Ketentuan AWS Layanan.

Bagian ini menjelaskan perintah bahasa definisi data (DDL) untuk membuat dan mengelola AWS Glue Data Catalog tampilan.

MENGUBAH TAMPILAN DIALEK

Anda dapat memperbarui tampilan Katalog Data dengan menambahkan dialek mesin atau dengan memperbarui atau menjatuhkan dialek mesin yang ada. Hanya `Lake Formation Admin` dan `Definer` (pengguna yang membuat tampilan) memiliki izin untuk menggunakan `ALTER VIEW DIALECT` pernyataan pada tampilan Katalog Data.

Sintaks

```
ALTER VIEW name [ FORCE ] [ ADD|UPDATE ] DIALECT AS query
```

```
ALTER VIEW name [ DROP ] DIALECT
```

KEKUATAN

`FORCE`Kata kunci menyebabkan informasi dialek mesin yang bertentangan dalam pandangan yang akan ditimpa dengan definisi baru. `FORCE`Kata kunci berguna ketika pembaruan ke tampilan Katalog Data menghasilkan definisi tampilan yang bertentangan di seluruh dialek mesin yang ada. Misalkan tampilan Katalog Data memiliki dialek Athena dan Amazon Redshift dan pembaruan menghasilkan konflik dengan Amazon Redshift dalam definisi tampilan. Dalam hal ini, Anda dapat menggunakan `FORCE` kata kunci untuk memungkinkan pembaruan selesai dan menandai dialek Amazon Redshift sebagai basi. Ketika mesin ditandai sebagai kueri basi tampilan, kueri gagal. Mesin memberikan pengecualian untuk melarang hasil basi. Untuk memperbaikinya, perbarui dialek basi dalam tampilan.

MENAMBAHKAN

Menambahkan dialek mesin baru ke tampilan Katalog Data. Mesin yang ditentukan harus belum ada dalam tampilan Katalog Data.

UPDATE

Memperbarui dialek mesin yang sudah ada di tampilan Katalog Data.

MENJATUHKAN

Menjatuhkan dialek mesin yang ada dari tampilan Katalog Data. Setelah Anda menjatuhkan mesin dari tampilan Katalog Data, tampilan Katalog Data tidak dapat ditanyakan oleh mesin yang dijatuhkan. Dialek mesin lain dalam tampilan masih dapat menanyakan tampilan.

DIALEK SEBAGAI

Memperkenalkan kueri SQL khusus mesin.

Contoh

```
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
```

BUAT TAMPILAN MULTI DIALEK YANG DILINDUNGI

Membuat tampilan Katalog Data di AWS Glue Data Catalog. Tampilan Katalog Data adalah skema tampilan tunggal yang bekerja dengan mulus di Athena dan mesin SQL lainnya seperti Amazon Redshift dan Amazon EMR.

Sintaks

```
CREATE [ OR REPLACE ] PROTECTED MULTI DIALECT VIEW view_name
 [ SECURITY DEFINER ]
AS query
```

DIJAGA

Kata kunci yang diperlukan. Menentukan bahwa tampilan dilindungi terhadap kebocoran data. Tampilan Katalog Data hanya dapat dibuat sebagai `PROTECTED` tampilan.

MULTI DIALEK

Menentukan bahwa tampilan mendukung dialek SQL dari mesin query yang berbeda dan karena itu dapat dibaca oleh mesin tersebut.

PENENTU KEAMANAN

Menentukan bahwa semantik definer berlaku untuk tampilan ini. Semantik definer berarti bahwa izin baca efektif pada tabel yang mendasarinya termasuk dalam prinsip atau peran yang mendefinisikan tampilan daripada prinsipal yang melakukan pembacaan aktual.

ATAU GANTI

Tampilan Katalog Data tidak dapat diganti jika dialek SQL dari mesin lain ada dalam tampilan. Jika mesin pemanggil memiliki satu-satunya dialek SQL yang ada dalam tampilan, tampilan dapat diganti.

Contoh

Contoh berikut membuat tampilan `orders_by_date` Data Catalog berdasarkan query pada `orders` tabel.

```
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
```

MENJELASKAN

Menampilkan daftar kolom untuk tampilan Katalog Data yang ditentukan. `DESCRIBE`Pernyataan itu mirip dengan `DESCRIBE` pernyataan untuk pandangan Athena. Tidak seperti pandangan Athena, output dari pernyataan tersebut dikendalikan melalui kontrol akses Lake Formation. Oleh karena itu, output dari kueri ini tidak semua kolom tampilan, tetapi hanya kolom yang dapat diakses oleh pemanggil.

Sintaks

```
DESCRIBE [db_name.]view_name
```

Contoh

```
DESCRIBE orders
```

TAMPILAN DROP

Menjatuhkan tampilan Katalog Data hanya jika dialek mesin pemanggil hadir dalam tampilan Katalog Data. Misalnya, jika pengguna menelepon `DROP VIEW` dari Athena, tampilan akan dihapus hanya jika dialek Athena ada dalam tampilan. Jika tidak, operasi gagal. Hanya Admin Lake Formation dan penentu tampilan yang memiliki izin untuk menggunakan `DROP VIEW` pernyataan pada tampilan Katalog Data.

Sintaks

```
DROP VIEW [ IF EXISTS ] view_name
```

Contoh

```
DROP VIEW orders_by_date
```

```
DROP FORCE VIEW IF EXISTS orders_by_date
```

Opsional`IF EXISTS`klausa menyebabkan kesalahan yang akan ditekan jika tampilan tidak ada.

TAMPILKAN KOLOM

Hanya menampilkan nama kolom untuk satu tampilan Katalog Data tertentu. `SHOW COLUMNS`Pernyataan itu mirip dengan `SHOW COLUMNS` pernyataan untuk pandangan Athena. Tidak seperti pandangan Athena, output dari pernyataan tersebut dikendalikan melalui kontrol akses Lake Formation. Oleh karena itu, output dari kueri ini tidak semua kolom tampilan, tetapi hanya kolom yang dapat diakses oleh pemanggil.

Sintaks

```
SHOW COLUMNS {FROM|IN} database_name.view_name
```

```
SHOW COLUMNS {FROM|IN} view_name [{FROM|IN} database_name]
```

TAMPILKAN BUAT TAMPILAN

Menampilkan sintaks SQL yang menciptakan tampilan Data Catalog. SQL yang dikembalikan menunjukkan sintaks create view yang digunakan di Athena. Hanya Admin Lake Formation dan prinsipal penentu tampilan yang diizinkan untuk memanggil tampilan Katalog `SHOW CREATE VIEW` Data.

Sintaks

```
SHOW CREATE VIEW view_name
```

Contoh

```
SHOW CREATE VIEW orders_by_date
```

TAMPILKAN TAMPILAN

Daftar nama semua tampilan dalam database. Semua tampilan Katalog Data dalam database yang memiliki dialek SQL mesin Athena terdaftar. Tampilan Katalog Data lainnya yang tidak memiliki dialek mesin Athena yang ada dalam tampilan disaring.

Sintaks

```
SHOW VIEWS [IN database_name] [LIKE 'regular_expression']
```

Contoh

```
SHOW VIEWS
```

```
SHOW VIEWS IN marketing_analytics LIKE 'orders*'
```

Menggunakan kueri yang disimpan

Anda dapat menggunakan konsol Athena untuk menyimpan, mengedit, menjalankan, mengganti nama, dan menghapus kueri yang Anda buat di editor kueri.

Pertimbangan dan batasan

- Anda dapat memperbarui nama, deskripsi, dan teks kueri kueri yang disimpan.
- Anda hanya dapat memperbarui kueri di akun Anda sendiri.
- Anda tidak dapat mengubah workgroup atau database yang query milik.
- Athena tidak menyimpan riwayat modifikasi query. Jika Anda ingin menyimpan versi kueri tertentu, simpan dengan nama yang berbeda.

Bekerja dengan kueri yang disimpan di konsol Athena

Untuk menyimpan kueri dan memberinya nama

1. Di editor kueri konsol Athena, masukkan atau jalankan kueri.
2. Di atas jendela editor kueri, pada tab untuk kueri, pilih tiga titik vertikal, lalu pilihSimpan sebagai.
3. DalamSimpan kuerikotak dialog, masukkan nama untuk query dan deskripsi opsional. Anda dapat menggunakan yang dapat diperluasPratinjau kueri SQLwindow untuk memverifikasi isi kueri sebelum Anda menyimpannya.
4. PilihSimpan kueri.

   Di editor kueri, tab untuk kueri menunjukkan nama yang Anda tentukan.

Untuk menjalankan kueri yang disimpan

1. Di konsol Athena, pilihKueri tersimpantab.
2. DalamKueri tersimpanlist, pilih ID kueri yang ingin Anda jalankan.

   Editor kueri menampilkan kueri yang Anda pilih.
3. Memilih Jalankan.

Untuk mengedit kueri yang disimpan

1. Di konsol Athena, pilihKueri tersimpantab.
2. DalamKueri tersimpanlist, pilih ID kueri yang ingin Anda edit.
3. Edit kueri di editor kueri.
4. Lakukan salah satu langkah berikut:

   - Untuk menjalankan kueri, pilihJalankan.
   - Untuk menyimpan kueri, pilih tiga titik vertikal pada tab untuk kueri, lalu pilihSimpan.
   - Untuk menyimpan kueri dengan nama yang berbeda, pilih tiga titik vertikal pada tab untuk kueri, lalu pilihSimpan sebagai.

Untuk mengganti nama atau menghapus kueri tersimpan yang sudah ditampilkan di editor kueri

1. Pilih tiga titik vertikal pada tab untuk kueri, lalu pilihGanti namaatauMenghapus.
2. Ikuti petunjuk untuk mengganti nama atau menghapus kueri.

Untuk mengganti nama kueri tersimpan yang tidak ditampilkan di editor kueri

1. Di konsol Athena, pilihKueri tersimpantab.
2. Pilih kotak centang untuk kueri yang ingin Anda ganti nama.
3. PilihGanti nama.
4. DalamGanti nama kuerikotak dialog, mengedit nama query dan deskripsi query. Anda dapat menggunakan yang dapat diperluasPratinjau kueri SQLwindow untuk memverifikasi isi kueri sebelum Anda mengganti namanya.
5. PilihGanti nama kueri.

Kueri berganti nama muncul diKueri tersimpandaftar.

Untuk menghapus kueri tersimpan yang tidak ditampilkan di editor kueri

1. Di konsol Athena, pilihKueri tersimpantab.
2. Pilih satu atau beberapa kotak centang untuk kueri yang ingin Anda hapus.
3. Pilih Delete (Hapus).
4. Pada prompt konfirmasi, pilihMenghapus.

   Satu atau lebih kueri dihapus dariKueri tersimpandaftar.

Menggunakan API Athena untuk memperbarui kueri yang disimpan

Untuk informasi tentang menggunakan API Athena untuk memperbarui kueri yang disimpan, lihatUpdateNamedQuerytindakan di Athena API Referensi.

Menggunakan kueri berparameter

Anda dapat menggunakan kueri berparameter Athena untuk menjalankan kembali kueri yang sama dengan nilai parameter yang berbeda pada waktu eksekusi dan membantu mencegah serangan injeksi SQL. Di Athena, query parameterized dapat mengambil bentuk parameter eksekusi dalam setiap query DHTML atau pernyataan SQL disiapkan.

- Kueri dengan parameter eksekusi dapat dilakukan dalam satu langkah dan tidak spesifik kelompok kerja. Anda menempatkan tanda tanya dalam kueri DHTML apa pun untuk nilai yang ingin Anda parameterisasi. Saat Anda menjalankan kueri, Anda mendeklarasikan nilai parameter eksekusi secara berurutan. Deklarasi parameter dan penetapan nilai untuk parameter dapat dilakukan dalam kueri yang sama, tetapi dengan cara dipisahkan. Tidak seperti pernyataan yang disiapkan, Anda dapat memilih workgroup saat Anda mengirimkan kueri dengan parameter eksekusi.
- Pernyataan disiapkan membutuhkan dua pernyataan SQL terpisah: `PREPARE` dan`EXECUTE`. Pertama, Anda menentukan parameter dalam `PREPARE` pernyataan. Kemudian, Anda menjalankan `EXECUTE` pernyataan yang memasok nilai untuk parameter yang Anda tentukan. Pernyataan yang disiapkan adalah spesifik kelompok kerja; Anda tidak dapat menjalankannya di luar konteks kelompok kerja tempat mereka berada.

Pertimbangan dan batasan

- Kueri parameter didukung di mesin Athena versi 2 dan versi yang lebih baru. Untuk informasi selengkapnya tentang versi mesin Aurora, lihat Pembuatan versi mesin Athena.
- Saat ini, kueri parameter hanya didukung untuk pernyataan `SELECT`, `INSERT INTO`, `CTAS`, dan `UNLOAD`.
- Dalam kueri berparameter, parameter bersifat posisional dan dilambangkan dengan. `?` Parameter diberikan nilai berdasarkan urutannya dalam kueri. Parameter bernama tidak didukung.
- Saat ini, `?` parameter hanya dapat ditempatkan di `WHERE` klausa. Sintaks seperti `SELECT ? FROM table` tidak didukung.
- Parameter tanda tanya tidak dapat ditempatkan dalam tanda kutip ganda atau tunggal (yaitu, `'?'` dan `"?"` bukan sintaks yang valid).
- Agar parameter eksekusi SQL diperlakukan sebagai string, mereka harus diapit dalam tanda kutip tunggal daripada tanda kutip ganda.
- Jika perlu, Anda dapat menggunakan `CAST` fungsi saat memasukkan nilai untuk istilah parameter. Misalnya, jika Anda memiliki kolom `date` tipe yang telah Anda parameterisasi dalam kueri dan Anda ingin menanyakan tanggal tersebut`2014-07-05`, memasukkan `CAST('2014-07-05' AS DATE)` nilai parameter akan mengembalikan hasilnya.
- Pernyataan yang disiapkan adalah kelompok kerja khusus, dan nama pernyataan yang disiapkan harus unik dalam kelompok kerja.
- Izin IAM untuk pernyataan yang disiapkan diperlukan. Untuk informasi selengkapnya, lihat Izinkan akses ke pernyataan yang disiapkan.
- Kueri dengan parameter eksekusi di konsol Athena dibatasi hingga maksimum 25 tanda tanya.

Query menggunakan parameter eksekusi

Anda dapat menggunakan placeholder tanda tanya dalam kueri DHTML apa pun untuk membuat kueri berparameter tanpa membuat pernyataan yang disiapkan terlebih dahulu. Untuk menjalankan kueri ini, Anda dapat menggunakan konsol Athena, atau menggunakan AWS CLI AWS SDK atau mendeklarasikan variabel dalam argumen. `execution-parameters`

Menjalankan kueri dengan parameter eksekusi di konsol Athena

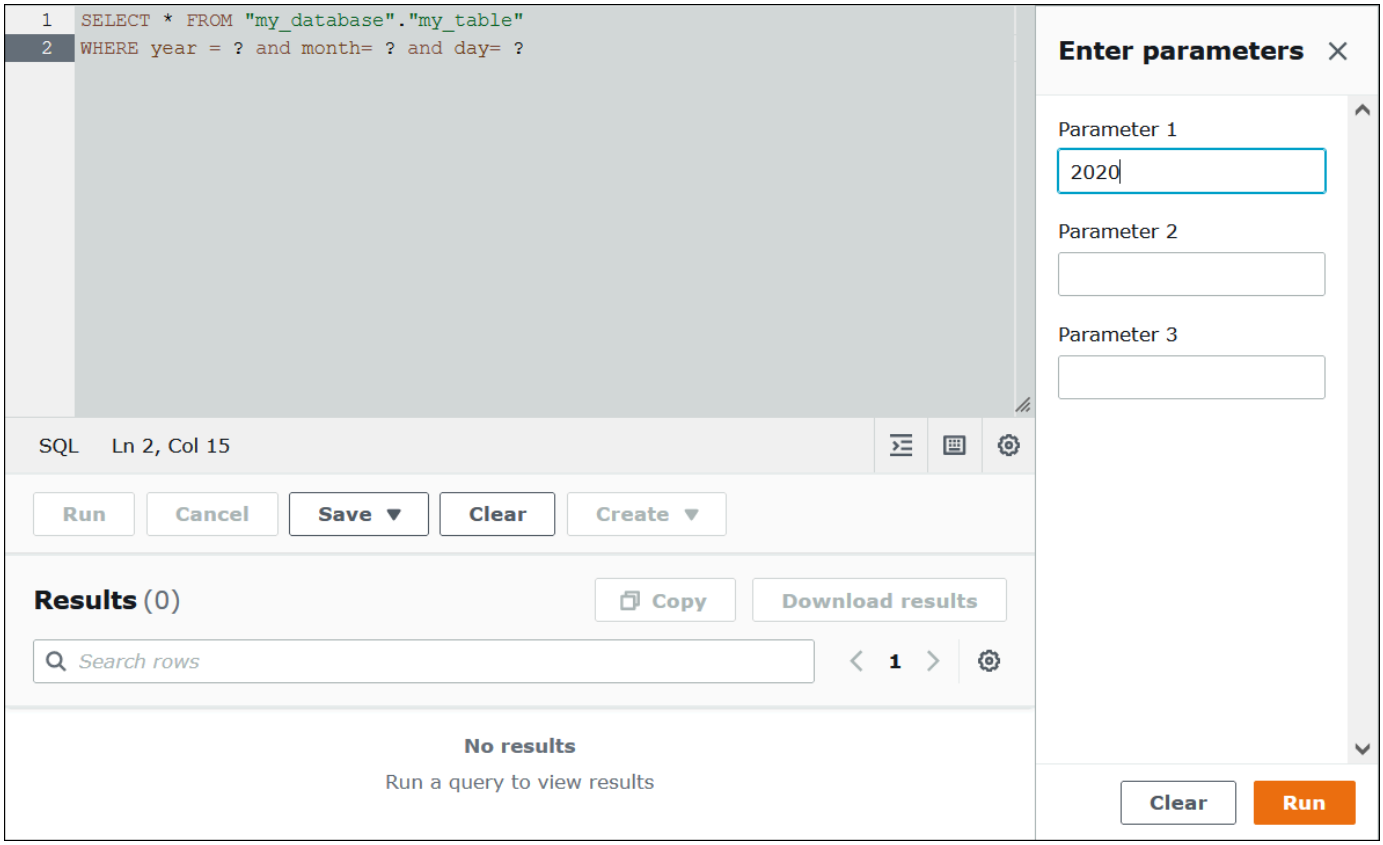
Saat Anda menjalankan kueri berparameter yang memiliki parameter eksekusi (tanda tanya) di konsol Athena, Anda akan diminta untuk nilai dalam urutan tanda tanya muncul dalam kueri.

Untuk menjalankan kueri yang memiliki parameter eksekusi

1. Masukkan kueri dengan placeholder tanda tanya di editor Athena, seperti pada contoh berikut.

```
SELECT * FROM "my_database"."my_table"
WHERE year = ? and month= ? and day= ?
```

2. Pilih Jalankan.
3. Dalam kotak dialog Masukkan parameter, masukkan nilai sesuai urutan untuk setiap tanda tanya dalam kueri.



4. Setelah Anda selesai memasukkan parameter, pilih Jalankan. Editor menampilkan hasil kueri untuk nilai parameter yang Anda masukkan.

Pada titik ini, Anda dapat melakukan salah satu dari yang berikut:

- Masukkan nilai parameter yang berbeda untuk kueri yang sama, lalu pilih Jalankan lagi.
- Untuk menghapus semua nilai yang Anda masukkan sekaligus, pilih Hapus.
- Untuk mengedit kueri secara langsung (misalnya, untuk menambah atau menghapus tanda tanya), tutup kotak dialog Enter parameter terlebih dahulu.

- Untuk menyimpan kueri berparameter untuk digunakan nanti, pilih Simpan atau Simpan sebagai, lalu beri nama kueri. Untuk informasi selengkapnya tentang menggunakan kueri tersimpan, lihatMenggunakan kueri yang disimpan.

Sebagai kenyamanan, kotak dialog Enter parameter mengingat nilai yang Anda masukkan sebelumnya untuk kueri selama Anda menggunakan tab yang sama di editor kueri.

Menjalankan kueri dengan parameter eksekusi menggunakan AWS CLI

Untuk menggunakan AWS CLI untuk menjalankan kueri dengan parameter eksekusi, gunakan `start-query-execution` perintah dan berikan kueri parameter dalam argumen. `query-string` Kemudian, dalam `execution-parameters` argumen, berikan nilai untuk parameter eksekusi. Contoh berikut menggambarkan teknik ini.

```
aws athena start-query-execution
--query-string "SELECT * FROM table WHERE x = ? AND y = ?"
--query-execution-context "Database"="default"
--result-configuration "OutputLocation"="s3://DOC-EXAMPLE-BUCKET;/..."
--execution-parameters "1" "2"
```

Menanyakan dengan pernyataan yang disiapkan

Anda dapat menggunakan pernyataan yang disiapkan untuk eksekusi berulang dari kueri yang sama dengan parameter kueri yang berbeda. Sebuah pernyataan yang disiapkan berisi placeholder parameter yang nilainya disuplai pada waktu eksekusi.

> **Note**
>
> Jumlah maksimum pernyataan yang disiapkan dalam kelompok kerja adalah 1000.

Pernyataan SQL

Anda dapat menggunakan pernyataan`PREPARE`, `EXECUTE` dan `DEALLOCATE PREPARE` SQL untuk menjalankan kueri berparameter di editor kueri konsol Athena.

- Untuk menentukan parameter tempat Anda biasanya akan menggunakan nilai-nilai literal, gunakan tanda tanya dalam pernyataan `PREPARE`.

- Untuk mengganti parameter dengan nilai-nilai saat Anda menjalankan kueri, gunakan klausa `USING` dalam pernyataan `EXECUTE`.
- Untuk menghapus pernyataan yang disiapkan dari pernyataan yang disiapkan dalam kelompok kerja, gunakan `DEALLOCATE PREPARE` pernyataan tersebut.

Bagian berikut memberikan detail tambahan tentang masing-masing pernyataan ini.

MEMPERSIAPKAN

Mempersiapkan pernyataan untuk dijalankan di lain waktu. Pernyataan yang disiapkan disimpan dalam grup kerja saat ini dengan nama yang Anda tentukan. Pernyataan tersebut dapat menyertakan parameter di tempat literal untuk diganti saat kueri dijalankan. Parameter yang akan diganti dengan nilai dilambangkan dengan tanda tanya.

Sintaks

```
PREPARE statement_name FROM statement
```

Tabel berikut mendeskripsikan parameter ini.

Parameter	Deskripsi
statement_name	Nama pernyataan yang harus dipersiapkan. Nama dalam buket harus unik.
pernyataan	Kueri `SELECT`, `CTAS`, atau `INSERT INTO`.

SIAPKAN contoh

Contoh berikut menunjukkan penggunaan `PREPARE`. Tanda tanya melambangkan nilai yang akan disuplai oleh pernyataan `EXECUTE` saat kueri dijalankan.

```
PREPARE my_select1 FROM
SELECT * FROM nation
```

```
PREPARE my_select2 FROM
```

```
SELECT * FROM "my_database"."my_table" WHERE year = ?
```

```
PREPARE my_select3 FROM
SELECT order FROM orders WHERE productid = ? and quantity < ?
```

```
PREPARE my_insert FROM
INSERT INTO cities_usa (city, state)
SELECT city, state
FROM cities_world
WHERE country = ?
```

```
PREPARE my_unload FROM
UNLOAD (SELECT * FROM table1 WHERE productid < ?)
TO 's3://DOC-EXAMPLE-BUCKET/'
WITH (format='PARQUET')
```

EXECUTE

Menjalankan pernyataan yang disiapkan. Nilai untuk parameter ditentukan dalam klausa `USING`.

Sintaks

```
EXECUTE statement_name [USING value1 [ ,value2, ... ] ]
```

`statement_name` adalah nama pernyataan yang disiapkan. *`value1`* dan *`value2`* adalah nilai-nilai yang akan ditentukan untuk parameter dalam pernyataan.

CONTOH EKSEKUSI

Contoh berikut menjalankan pernyataan yang disiapkan `my_select1`, yang tidak berisi parameter.

```
EXECUTE my_select1
```

Contoh berikut menjalankan pernyataan yang disiapkan `my_select2`, yang berisi parameter tunggal.

```
EXECUTE my_select2 USING 2012
```

Contoh berikut menjalankan pernyataan yang disiapkan `my_select3`, yang memiliki dua parameter.

```
EXECUTE my_select3 USING 346078, 12
```

Contoh berikut menyuplai nilai string untuk parameter dalam pernyataan yang disiapkan `my_insert`.

```
EXECUTE my_insert USING 'usa'
```

Contoh berikut memasok nilai numerik untuk `productid` parameter dalam pernyataan `my_unload` disiapkan.

```
EXECUTE my_unload USING 12
```

DEALLOCATE PREPARE

Menghapus pernyataan yang disiapkan dengan nama tertentu dari daftar pernyataan yang disiapkan dalam grup kerja saat ini.

Sintaks

```
DEALLOCATE PREPARE statement_name
```

`statement_name` adalah nama pernyataan yang disiapkan yang akan dihapus.

Contoh

Contoh berikut menghapus pernyataan yang disiapkan `my_select1` dari grup kerja saat ini.

```
DEALLOCATE PREPARE my_select1
```

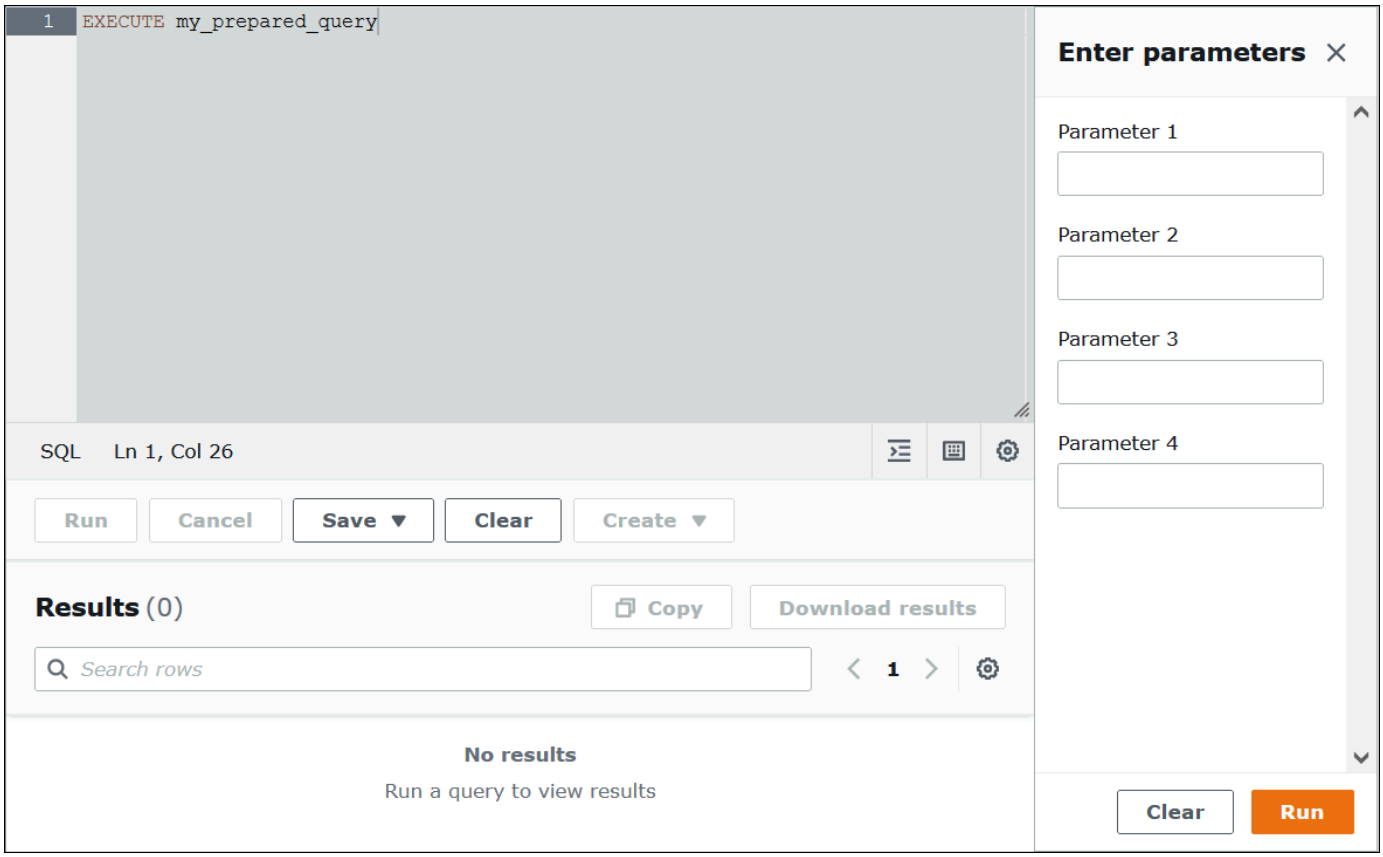
Menjalankan pernyataan yang disiapkan tanpa klausa USING di konsol Athena

Jika Anda menjalankan pernyataan siap yang ada dengan sintaks `EXECUTE` *`prepared_statement`* di editor kueri, Athena membuka kotak dialog Enter parameter sehingga Anda dapat memasukkan nilai yang biasanya masuk dalam klausa pernyataan. `USING` `EXECUTE ... USING`

Untuk menjalankan pernyataan yang disiapkan menggunakan kotak dialog Enter parameter

1. *Di editor kueri, alih-alih menggunakan sintaks EXECUTE prepared_statement USING value1 , value2 ..., gunakan sintaks prepared_statement. EXECUTE*

2. Pilih Jalankan. Kotak dialog Enter parameter muncul.



3. Masukkan nilai secara berurutan di kotak dialog Parameter eksekusi. Karena teks asli kueri tidak terlihat, Anda harus mengingat arti dari setiap parameter posisi atau memiliki pernyataan yang disiapkan tersedia untuk referensi.
4. Pilih Jalankan.

Membuat pernyataan yang disiapkan menggunakan AWS CLI

Untuk menggunakan AWS CLI untuk membuat pernyataan siap, Anda dapat menggunakan salah satu `athena` perintah berikut:

- Gunakan `create-prepared-statement` perintah dan berikan pernyataan kueri yang memiliki parameter eksekusi.
- Gunakan `start-query-execution` perintah dan berikan string kueri yang menggunakan `PREPARE` sintaks.

Menggunakan create-prepared-statement

Dalam sebuah `create-prepared-statement` perintah, tentukan teks query dalam `query-statement` argumen, seperti pada contoh berikut.

```
aws athena create-prepared-statement
--statement-name PreparedStatement1
--query-statement "SELECT * FROM table WHERE x = ?"
--work-group athena-engine-v2
```

Menggunakan start-query-execution dan sintaks PREPARE

Gunakan perintah `start-query-execution`. Masukkan `PREPARE` pernyataan dalam `query-string` argumen, seperti pada contoh berikut:

```
aws athena start-query-execution
--query-string "PREPARE PreparedStatement1 FROM SELECT * FROM table WHERE x = ?"
--query-execution-context '{"Database": "default"}'
--result-configuration '{"OutputLocation": "s3://DOC-EXAMPLE-BUCKET/..."}'
```

Melaksanakan pernyataan yang disiapkan menggunakan AWS CLI

Untuk mengeksekusi pernyataan yang disiapkan dengan AWS CLI, Anda dapat memberikan nilai untuk parameter dengan menggunakan salah satu metode berikut:

- Gunakan `execution-parameters` argumennya.
- Gunakan sintaks `EXECUTE ... USING` SQL dalam argumen. `query-string`

Menggunakan argumen eksekusi-parameter

Dalam pendekatan ini, Anda menggunakan `start-query-execution` perintah dan memberikan nama pernyataan siap yang ada dalam `query-string` argumen. Kemudian, dalam `execution-parameters` argumen, Anda memberikan nilai untuk parameter eksekusi. Contoh berikut menunjukkan metode ini.

```
aws athena start-query-execution
--query-string "Execute PreparedStatement1"
--query-execution-context "Database"="default"
--result-configuration "OutputLocation"="s3://DOC-EXAMPLE-BUCKET/..."
```

```
--execution-parameters "1" "2"
```

Menggunakan EXECUTE... MENGGUNAKAN sintaks SQL

Untuk menjalankan pernyataan siap yang ada menggunakan `EXECUTE ... USING` sintaks, Anda menggunakan `start-query-execution` perintah dan menempatkan kedua nama pernyataan yang disiapkan dan nilai parameter dalam `query-string` argumen, seperti pada contoh berikut:

```
aws athena start-query-execution
--query-string "EXECUTE PreparedStatement1 USING 1"
--query-execution-context '{"Database": "default"}'
--result-configuration '{"OutputLocation": "s3://DOC-EXAMPLE-BUCKET/..."}'
```

Daftar pernyataan yang disiapkan

Untuk membuat daftar pernyataan yang disiapkan untuk grup kerja tertentu, Anda dapat menggunakan perintah list-prepared-statements AWS CLI Athena atau tindakan ListPreparedStatementsAthena API. parameter `--work-group` diperlukan.

```
aws athena list-prepared-statements --work-group primary
```

Sumber daya tambahan

Lihat posting terkait berikut di Blog AWS Big Data.

- Meningkatkan penggunaan kembali dan keamanan menggunakan kueri parameter Amazon Athena
- Gunakan kueri parameter Amazon Athena untuk menyediakan data sebagai layanan

Menggunakan pengoptimal berbasis biaya

Anda dapat menggunakan fitur pengoptimal berbasis biaya (CBO) di Athena SQL untuk mengoptimalkan kueri Anda. Anda dapat meminta Athena mengumpulkan statistik tingkat tabel atau kolom untuk salah satu tabel Anda. AWS Glue Jika semua tabel dalam kueri Anda memiliki statistik, Athena menggunakan statistik untuk membuat rencana eksekusi yang ditentukannya sebagai yang paling berkinerja. Pengoptimal kueri menghitung rencana alternatif berdasarkan model statistik dan kemudian memilih salah satu yang kemungkinan akan menjadi yang tercepat untuk menjalankan kueri.

Statistik pada AWS Glue tabel dikumpulkan dan disimpan di AWS Glue Data Catalog dan disediakan untuk Athena untuk perencanaan dan pelaksanaan kueri yang lebih baik. Statistik ini adalah statistik tingkat kolom seperti jumlah yang berbeda, jumlah nilai nol, maks, dan min pada jenis file seperti Parket, ORC, JSON, ION, CSV, dan XHTML. Amazon Athena menggunakan statistik ini untuk mengoptimalkan kueri dengan menerapkan filter paling ketat sedini mungkin dalam pemrosesan kueri. Pemfilteran ini membatasi penggunaan memori dan jumlah catatan yang harus dibaca untuk memberikan hasil kueri.

Dalam hubungannya dengan CBO, Athena menggunakan fitur yang disebut rule-based optimizer (RBO). RBO secara mekanis menerapkan aturan yang diharapkan dapat meningkatkan kinerja kueri. RBO umumnya bermanfaat karena transformasinya bertujuan untuk menyederhanakan rencana kueri. Namun, karena RBO tidak melakukan perhitungan biaya atau perbandingan rencana, kueri yang lebih rumit menyulitkan RBO untuk membuat rencana yang optimal.

Untuk alasan ini, Athena menggunakan RBO dan CBO untuk mengoptimalkan kueri Anda. Setelah Athena mengidentifikasi peluang untuk meningkatkan eksekusi kueri, Athena menciptakan rencana yang optimal. Untuk informasi tentang detail rencana eksekusi, lihatMelihat rencana eksekusi untuk kueri SQL. Untuk diskusi mendetail tentang cara kerja CBO, lihat Mempercepat kueri dengan pengoptimal berbasis biaya di Amazon Athena di Blog Big Data. AWS

Untuk menghasilkan statistik untuk tabel AWS Glue Katalog, Anda dapat menggunakan konsol Athena, AWS Glue Konsol, atau AWS Glue API. Karena Athena terintegrasi dengan AWS Glue Catalog, Anda secara otomatis mendapatkan peningkatan kinerja kueri yang sesuai saat menjalankan kueri dari Amazon Athena.

Pertimbangan dan batasan

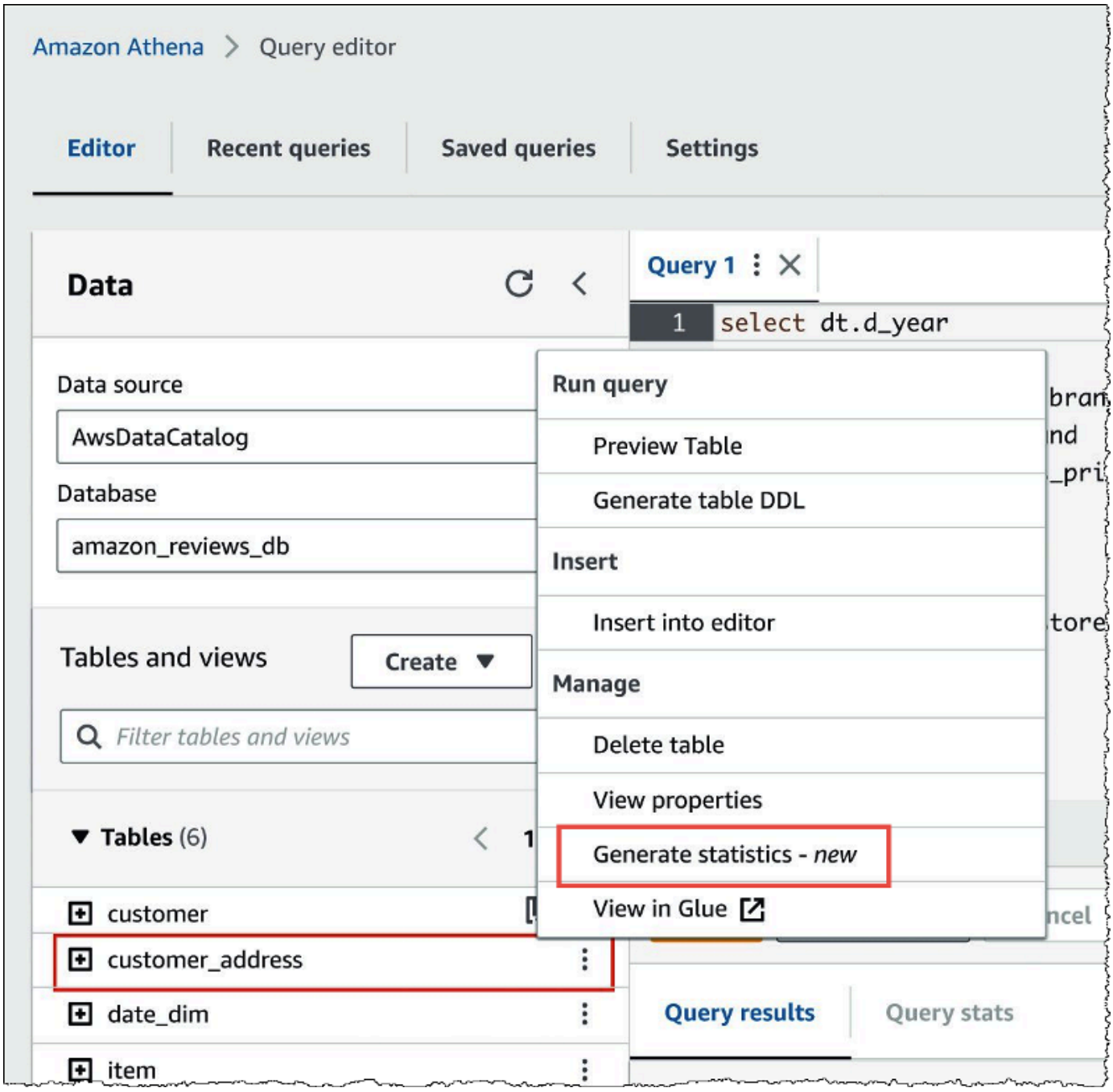
- Jenis tabel — Saat ini, fitur CBO di Athena hanya mendukung tabel Hive yang ada di. AWS Glue Data Catalog
- Athena for Spark - Fitur CBO tidak tersedia di Athena untuk Spark.
- Harga — Untuk informasi harga, kunjungi halaman AWS Glue harga.

Menghasilkan statistik tabel menggunakan konsol Athena

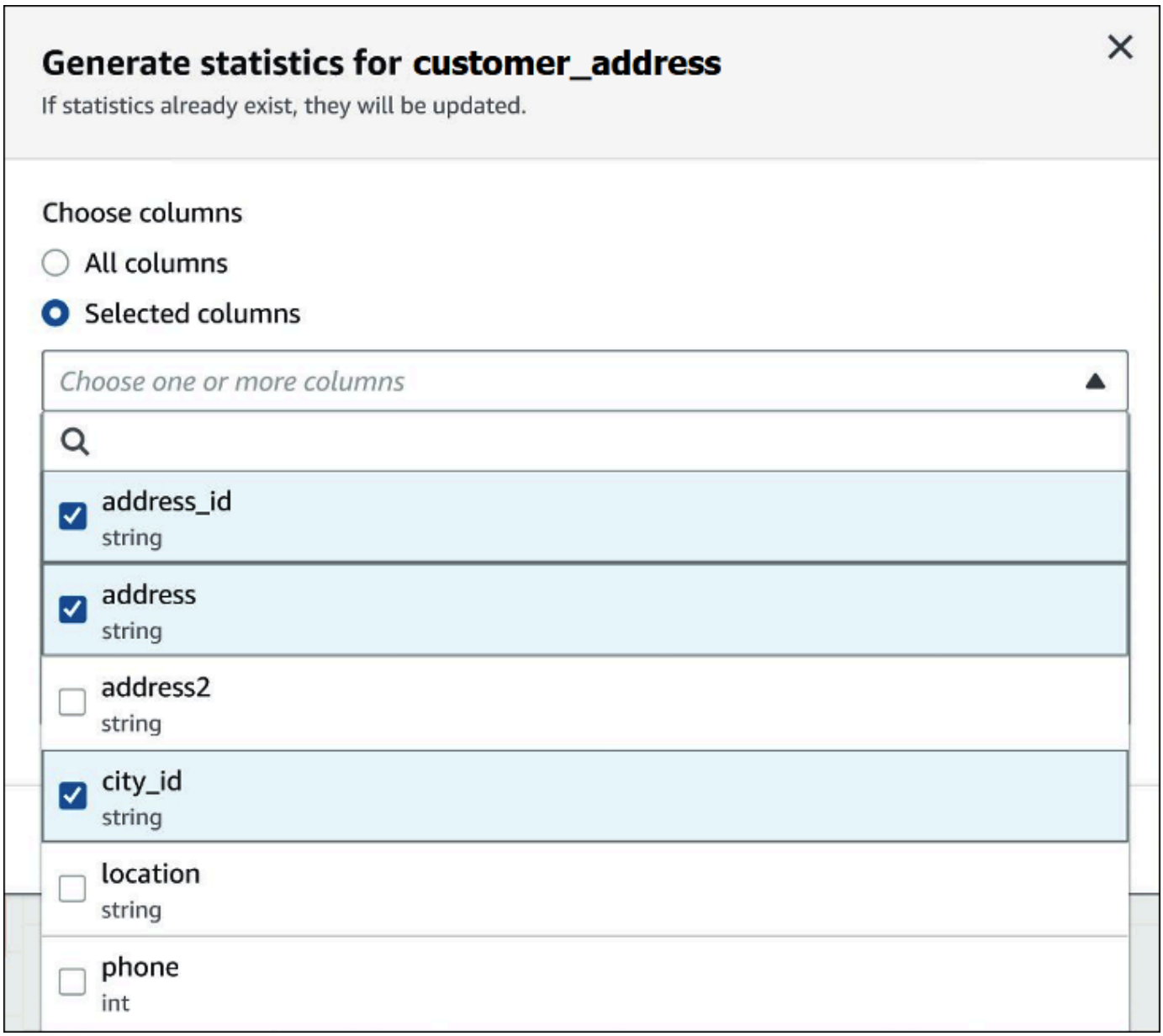
Bagian ini menjelaskan cara menggunakan konsol Athena untuk menghasilkan statistik tingkat tabel atau kolom untuk tabel di. AWS Glue Untuk informasi tentang penggunaan AWS Glue untuk menghasilkan statistik tabel, lihat Bekerja dengan statistik kolom di Panduan AWS Glue Pengembang.

Untuk menghasilkan statistik untuk tabel menggunakan konsol Athena

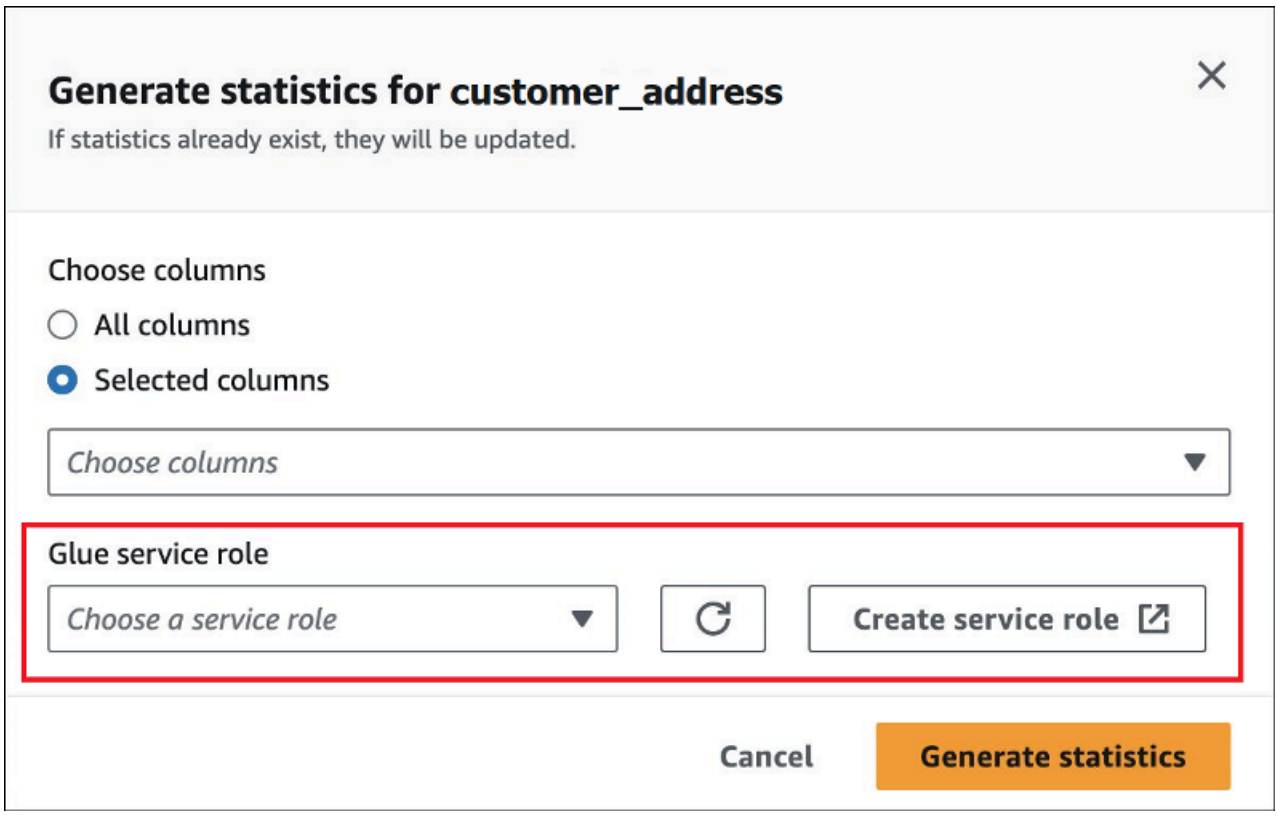
1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Dalam daftar Tabel editor kueri Athena, pilih tiga titik vertikal untuk tabel yang Anda inginkan, lalu pilih Hasilkan statistik.



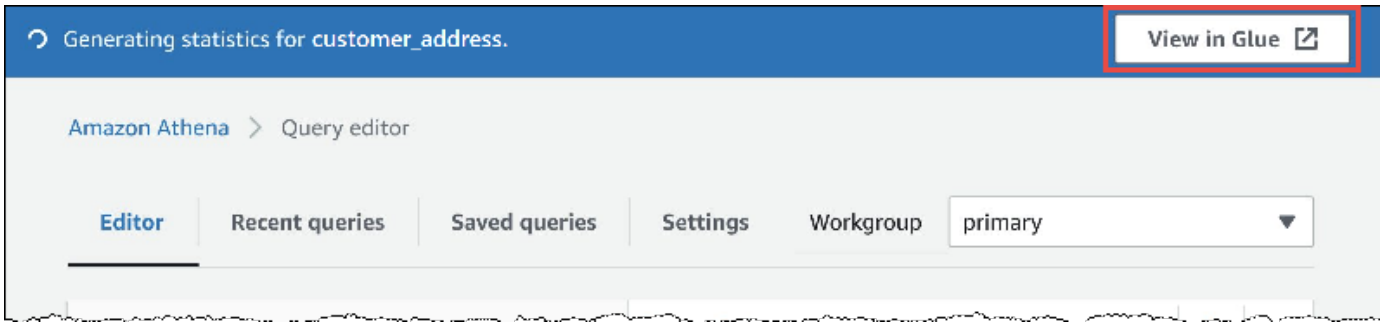
3. Dalam kotak dialog Hasilkan statistik, pilih Semua kolom untuk menghasilkan statistik untuk semua kolom dalam tabel, atau pilih Kolom yang dipilih untuk memilih kolom tertentu. Semua kolom adalah pengaturan default.



4. Untuk peran AWS Glue layanan, buat atau pilih peran layanan yang ada untuk memberikan izin AWS Glue untuk menghasilkan statistik. Peran AWS Glue layanan juga memerlukan `S3:GetObject`izin ke bucket Amazon S3 yang berisi data tabel.



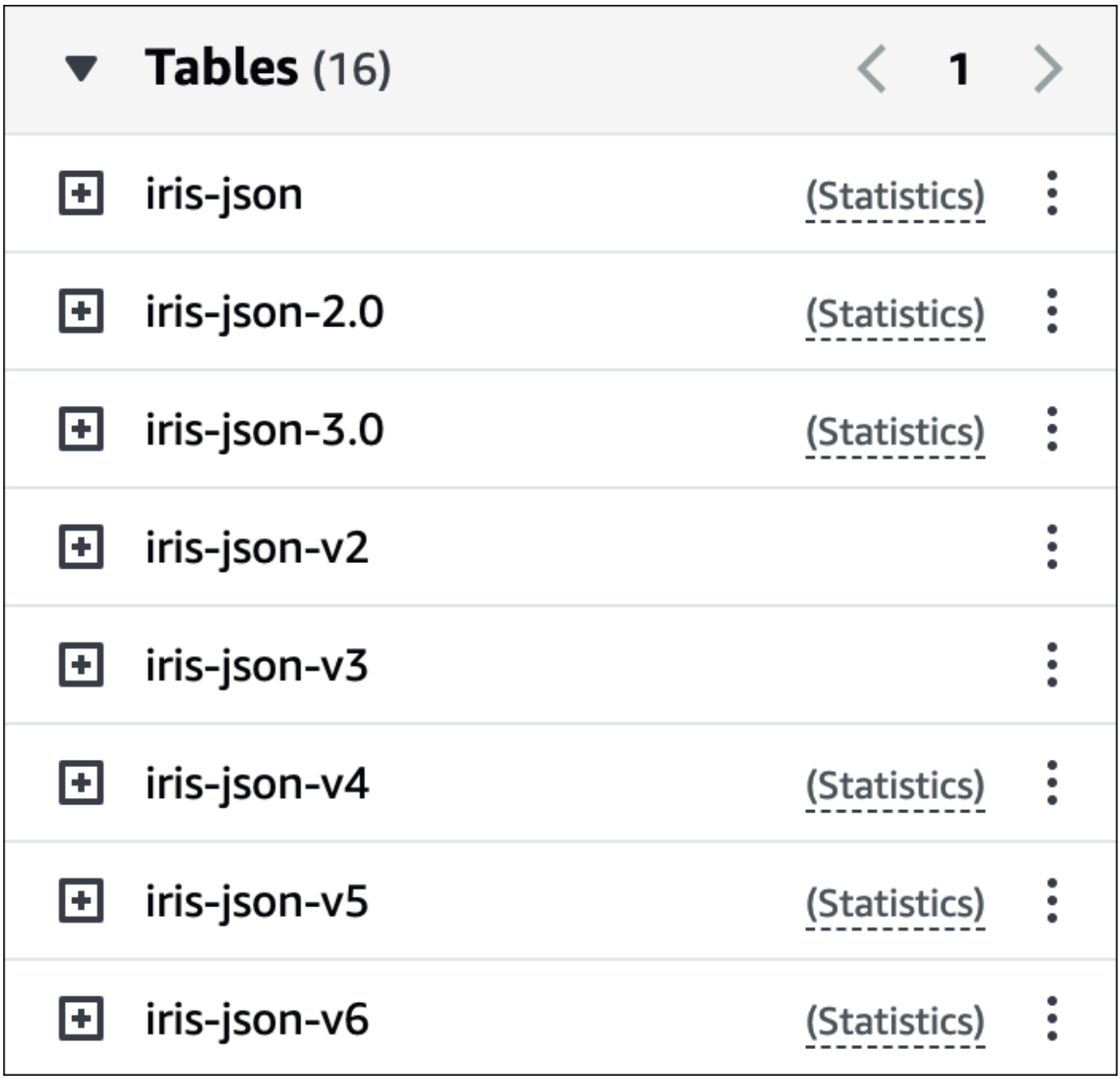
5. Pilih Hasilkan statistik. Sebuah Menghasilkan statistik untuk banner notifikasi ***`table_name`*** menampilkan status tugas.



6. Untuk melihat detail di AWS Glue konsol, pilih Lihat di Glue.

   Untuk informasi tentang melihat statistik di AWS Glue konsol, lihat Melihat statistik kolom di Panduan AWS Glue Pengembang.

7. Setelah statistik dihasilkan, tabel dan kolom yang memiliki statistik menunjukkan kata Statistik dalam tanda kurung, seperti pada gambar berikut.



Sekarang ketika Anda menjalankan kueri Anda, Athena akan melakukan optimasi berbasis biaya pada tabel dan kolom yang statistiknya dihasilkan.

Sumber daya tambahan

Untuk informasi tambahan, lihat sumber daya berikut.

Meminta data S3 Express One Zone

Kelas penyimpanan Amazon S3 Express One Zone adalah kelas penyimpanan Amazon S3 berkinerja tinggi yang menyediakan waktu respons milidetik satu digit. Dengan demikian, ini berguna untuk aplikasi yang sering mengakses data dengan ratusan ribu permintaan per detik.

S3 Express One Zone mereplikasi dan menyimpan data dalam Availability Zone yang sama untuk mengoptimalkan kecepatan dan biaya. Ini berbeda dari kelas penyimpanan Regional Amazon S3, yang secara otomatis mereplikasi data di minimal tiga AWS Availability Zone dalam file. Wilayah AWS

Untuk informasi selengkapnya, lihat Apa itu S3 Express One Zone? di Panduan Pengguna Amazon S3.

Prasyarat

Konfirmasikan bahwa kondisi berikut terpenuhi sebelum Anda mulai:

- Mesin Athena versi 3 - Untuk menggunakan S3 Express One Zone dengan Athena SQL, workgroup Anda harus dikonfigurasi untuk menggunakan mesin Athena versi 3.
- Izin S3 Express One Zone — Saat S3 Express One Zone memanggil tindakan seperti`GET`,`LIST`, atau `PUT` pada objek Amazon S3, kelas penyimpanan memanggil atas nama Anda. `CreateSession` Untuk alasan ini, kebijakan IAM Anda harus mengizinkan `s3express:CreateSession` tindakan, yang memungkinkan Athena untuk menjalankan operasi API yang sesuai.

Pertimbangan dan batasan

Saat Anda menanyakan S3 Express One Zone dengan Athena, pertimbangkan poin-poin berikut.

- Bucket S3 Express One Zone hanya `SSE_S3` mendukung enkripsi. Hasil kueri Athena ditulis menggunakan `SSE_S3` enkripsi terlepas dari opsi yang Anda pilih dalam pengaturan workgroup untuk mengenkripsi hasil kueri. Batasan ini mencakup semua skenario di mana Athena menulis data ke bucket S3 Express One Zone, termasuk `CREATE TABLE AS` (CTAS) dan pernyataan. `INSERT INTO`
- AWS Glue Crawler tidak didukung untuk membuat tabel pada data S3 Express One Zone.
- `MSCK REPAIR TABLE`Pernyataan ini tidak didukung. Sebagai solusinya, gunakan. ALTER TABLE ADD PARTITION

- `ALTER TABLE ADD PARTITION`,`ALTER TABLE DROP PARTITION`, dan tidak `ALTER TABLE RENAME PARTITION` didukung untuk tabel Iceberg di S3 Express One Zone.
- Format file dan tabel berikut tidak didukung atau memiliki dukungan terbatas. Jika format tidak terdaftar, tetapi didukung untuk Athena (seperti Parket, ORC, dan JSON), maka format tersebut juga didukung untuk digunakan dengan penyimpanan S3 Express One Zone.

Format file atau tabel	Batasan
Apache Avro	Tidak didukung
CloudTrail log	Tidak didukung
Apache Hudi	Tidak didukung
Amazon Ion	Tidak didukung
Log logstash	Tidak didukung
Log Apache WebServer	Tidak didukung
Danau Delta	DDL tidak didukung. Untuk informasi tentang membuat tabel Delta Lake menggunakan skema dummy, lihat. Sinkronisasi metadata Delta Lake `SELECT`kueri terhadap tabel didukung.

Memulai

Menanyakan data S3 Express One Zone dengan Athena sangatlah mudah. Untuk memulai, gunakan prosedur berikut.

Untuk menggunakan Athena SQL untuk menanyakan data S3 Express One Zone

1. Transisi data Anda ke penyimpanan S3 Express One Zone. Untuk informasi selengkapnya, lihat Menyetel kelas penyimpanan objek di Panduan Pengguna Amazon S3.
2. Gunakan CREATE TABLE pernyataan di Athena untuk membuat katalog data Anda. AWS Glue Data Catalog Untuk informasi tentang membuat tabel di Athena, lihat Membuat tabel di Athena dan pernyataannya. CREATE TABLE

3. (Opsional) Konfigurasikan lokasi hasil kueri workgroup Athena Anda untuk menggunakan bucket direktori Amazon S3. Bucket direktori Amazon S3 lebih berkinerja daripada bucket umum dan dirancang untuk beban kerja atau aplikasi penting kinerja yang memerlukan latensi milidetik satu digit yang konsisten. Untuk informasi selengkapnya, lihat ikhtisar bucket direktori di Panduan Pengguna Amazon S3.

Menanyakan objek Amazon S3 Glacier yang dipulihkan

Anda dapat menggunakan Athena untuk menanyakan objek yang dipulihkan dari kelas penyimpanan Amazon S3 Glacier Flexible Retrieval (sebelumnya Glacier) dan S3 Glacier Deep Archive kelas penyimpanan Amazon S3. Anda harus mengaktifkan kemampuan ini berdasarkan per tabel. Jika Anda tidak mengaktifkan fitur pada tabel sebelum menjalankan kueri, Athena melewatkan semua objek S3 Glacier Flexible Retrieval dan S3 Glacier Deep Archive tabel selama eksekusi kueri.

Pertimbangan dan batasan

- Menanyakan objek Amazon S3 Glacier yang dipulihkan hanya didukung pada mesin Athena versi 3.
- Fitur ini hanya didukung untuk tabel Apache Hive.
- Anda harus memulihkan objek Anda sebelum Anda menanyakan data Anda; Athena tidak mengembalikan objek untuk Anda.

Mengkonfigurasi tabel untuk menggunakan objek yang dipulihkan

Untuk mengonfigurasi tabel Athena Anda untuk menyertakan objek yang dipulihkan dalam kueri Anda, Anda harus mengatur properti `read_restored_glacier_objects` tabelnya. `true` Untuk melakukan ini, Anda dapat menggunakan editor kueri Athena atau konsol. AWS Glue Anda juga dapat menggunakan AWS Glue CLI, AWS Glue API, atau SDK AWS Glue .

Menggunakan editor kueri Athena

Di Athena, Anda dapat menggunakan ALTER TABLE SET TBLPROPERTIES perintah untuk mengatur properti tabel, seperti pada contoh berikut.

```
ALTER TABLE table_name SET TBLPROPERTIES ('read_restored_glacier_objects' = 'true')
```

Menggunakan konsol AWS Glue

Di AWS Glue konsol, lakukan langkah-langkah berikut untuk menambahkan properti `read_restored_glacier_objects` tabel.

Untuk mengkonfigurasi properti tabel di AWS Glue konsol

1. Masuk ke AWS Management Console dan buka AWS Glue konsol di https://console.aws.amazon.com/glue/.
2. Lakukan salah satu tindakan berikut:
   - Pilih Buka Katalog Data.
   - Di panel navigasi, pilih tabel Katalog Data.
3. Pada halaman Tabel, dalam daftar tabel, pilih tautan untuk tabel yang ingin Anda edit.
4. Pilih Tindakan, Edit tabel.
5. Pada halaman Edit tabel, di bagian properti Tabel, tambahkan pasangan kunci-nilai berikut.
   - Untuk Kunci, tambahkan`read_restored_glacier_objects`.
   - Untuk Nilai, masukkan `true`.
6. Pilih Simpan.

Menggunakan AWS CLI

Dalam AWS CLI, Anda dapat menggunakan perintah AWS Glue update-table dan `--table-input` argumennya untuk mendefinisikan ulang tabel dan dengan demikian menambahkan properti. `read_restored_glacier_objects` Dalam `--table-input` argumen, gunakan `Parameters` struktur untuk menentukan `read_restored_glacier_objects` properti dan nilai`true`. Perhatikan bahwa argumen untuk tidak `--table-input` harus memiliki spasi dan harus menggunakan garis miring terbalik untuk menghindari tanda kutip ganda. Dalam contoh berikut, ganti *`my_database`* dan *`my_table`* dengan nama database dan tabel Anda.

```
aws glue update-table \
   --database-name my_database \
   --table-input={\"Name\":\"my_table\",\"Parameters\":{\"read_restored_glacier_objects
\":\"true\"}}
```

> **Important**
>
> AWS Glue `update-table`Perintah bekerja dalam mode overwrite, yang berarti menggantikan definisi tabel yang ada dengan definisi baru yang ditentukan oleh parameter. `table-input` Untuk alasan ini, pastikan untuk juga menentukan semua bidang yang Anda inginkan dalam tabel Anda dalam `table-input` parameter ketika Anda menambahkan `read_restored_glacier_objects` properti.

Menangani pembaruan skema

Bagian ini menyediakan panduan tentang penanganan pembaruan skema untuk berbagai format data. Athena adalah mesin schema-on-read kueri. Ini berarti bahwa saat Anda membuat tabel di Athena, itu berlaku skema saat membaca data. Itu tidak mengubah atau menulis ulang data yang mendasari.

Jika Anda mengantisipasi perubahan dalam skema tabel, pertimbangkan untuk membuat mereka dalam format data yang cocok untuk kebutuhan Anda. Tujuan Anda adalah untuk menggunakan kembali pertanyaan Athena yang ada terhadap skema yang berkembang, dan menghindari kesalahan ketidakcocokan skema saat kueri tabel dengan partisi.

Untuk mencapai tujuan ini, pilih format data tabel berdasarkan tabel dalam topik berikut.

Topik

Ringkasan: Pembaruan dan format data di Athena

Tabel berikut merangkum format penyimpanan data dan manipulasi skema mereka didukung. Gunakan tabel ini untuk membantu Anda memilih format yang akan memungkinkan Anda untuk terus menggunakan Athena kueri bahkan saat skema Anda berubah dari waktu ke waktu.

Dalam tabel ini, mengamati bahwa Parquet dan ORC adalah format columnar dengan metode akses kolom default yang berbeda. Secara default, Parquet akan mengakses kolom berdasarkan nama dan ORC berdasarkan indeks (nilai ordinal). Oleh karena itu, Athena menyediakan SerDe properti yang

didefinisikan saat membuat tabel untuk beralih metode akses kolom default yang memungkinkan fleksibilitas yang lebih besar dengan evolusi skema.

Untuk Parket, `parquet.column.index.access` properti dapat diatur `ketrue`, yang menetapkan metode akses kolom untuk menggunakan nomor urut kolom. Menetapkan properti ini `kefalse`akan mengubah metode akses kolom untuk menggunakan nama kolom. Demikian pula, untuk ORC menggunakan`orc.column.index.access`properti untuk mengontrol metode akses kolom. Untuk informasi selengkapnya, lihat Akses indeks di ORC dan parket.

CSV dan TSV memungkinkan Anda untuk melakukan semua manipulasi skema kecuali penataan ulang kolom, atau menambahkan kolom di awal tabel. Misalnya, jika evolusi skema Anda hanya memerlukan penggantian nama kolom tetapi tidak menghapusnya, Anda dapat memilih untuk membuat tabel Anda di CSV atau TSV. Jika Anda memerlukan menghapus kolom, jangan gunakan CSV atau TSV, dan sebagai gantinya gunakan format lain yang didukung, sebaiknya, format kolumnar, seperti Parquet atau ORC.

Pembaruan skema dan format data di Athena

Jenis pembaruan skema yang diharapkan	Ringkasan	CSV (dengan dan tanpa tajuk) dan TSV	JSO	AVF	PARQUE Baca dengan nama (default)	PARQUE Dibaca dengan indeks	ORC: Dibaca berdasarkan indeks (default)	ORC: Baca dengan nama
Ganti nama kolom	Simpan data Anda di CSV dan TSV, atau di ORC dan Parquet jika dibaca berdasarkan indeks.	T	T	T	T	T	T	T
Tambahkan kolom di awal atau di tengah tabel	Simpan data Anda di JSON, AVRO, atau di Parquet dan ORC jika mereka dibaca berdasarkan	T	T	Y	T	T	T	T

Jenis pembaruan skema yang diharapkan	Ringkasan	CSV (dengan dan tanpa tajuk) dan TSV	JSO	AVF	PARQUE Baca dengan nama (default)	PARQUE Dibaca dengan indeks	ORC: Dibaca berdasarkan indeks (default)	ORC: Baca dengan nama
	nama. Jangan gunakan CSV dan TSV.							
Tambahkan kolom di akhir tabel	Simpan data Anda di CSV atau TSV, JSON, AVRO, ORC, atau Parquet.	T	Y	Y	Y	Y	Y	T
Hapus kolom	Simpan data Anda di JSON, AVRO, atau Parquet dan ORC, jika mereka dibaca berdasarkan nama. Jangan gunakan CSV dan TSV.	T	T	Y	T	T	T	T
Susun ulang kolom	Simpan data Anda di AVRO, JSON atau ORC dan Parquet jika dibaca berdasarkan nama.	T	T	Y	T	T	T	T

Jenis pembaruan skema yang diharapkan	Ringkasan	CSV (dengan dan tanpa tajuk) dan TSV	JSO	AVF	PARQUE Baca dengan nama (default)	PARQUE Dibaca dengan indeks	ORC: Dibaca berdasarkan indeks (default)	ORC: Baca dengan nama
Mengubah tipe data kolom	Simpan data Anda dalam format apa pun, tetapi uji kueri Anda di Athena untuk memastikan jenis data kompatibel. Untuk Parquet dan ORC, mengubah tipe data bekerja hanya untuk tabel dipartisi.	T	Y	Y	Y	Y	Y	T

Akses indeks di ORC dan parket

Parquet dan ORC adalah format penyimpanan data kolumnar yang dapat dibaca berdasarkan indeks, atau dengan nama. Menyimpan data Anda dalam salah satu format ini memungkinkan Anda melakukan semua operasi pada skema dan menjalankan kueri Athena tanpa kesalahan skema ketidakcocokan.

- Athenamembaca ORC berdasarkan indeks secara baku, sebagaimana didefinisikan dalam`SERDEPROPERTIES ( 'orc.column.index.access'='true')`. Untuk informasi selengkapnya, lihat ORC: Dibaca dengan indeks.
- Athena membacaParket dengan nama secara lalai, sebagaimana didefinisikan dalam`SERDEPROPERTIES ( 'parquet.column.index.access'='false')`. Untuk informasi selengkapnya, lihat Parket: Baca dengan nama.

Karena ini adalah default, menentukan SerDe properti ini dalam `CREATE TABLE` kueri Anda adalah opsional, mereka digunakan secara implisit. Saat digunakan, mereka memungkinkan Anda untuk menjalankan beberapa operasi update skema sementara mencegah operasi seperti lainnya. Untuk mengaktifkan operasi tersebut, jalankan `CREATE TABLE` kueri lain dan ubah SerDe pengaturan.

> **Note**
>
> SerDe Properti tidak secara otomatis disebarkan ke setiap partisi. Gunakan `ALTER TABLE ADD PARTITION` pernyataan untuk mengatur SerDe properti untuk setiap partisi. Untuk mengotomatiskan proses ini, menulis script yang berjalan`ALTER TABLE ADD PARTITION`.

Bagian berikut akan menjelaskan kasus ini secara terperinci.

ORC: Dibaca dengan indeks

Tabel di`ORC` dibaca oleh indeksSecara default. Ini didefinisikan oleh sintaks berikut:

```
WITH SERDEPROPERTIES (
  'orc.column.index.access'='true')
```

Membaca berdasarkan indeksmemungkinkan Anda untuk mengubah nama kolom. Tapi kemudian Anda kehilangan kemampuan untuk menghapus kolom atau menambahkannya di tengah tabel.

Untuk membuat ORC dibaca dengan nama, yang akan memungkinkan Anda untuk menambahkan kolom di tengah tabel atau menghapus kolom di ORC, atur SerDe properti `orc.column.index.access` ke `false` dalam pernyataan. `CREATE TABLE` Dalam konfigurasi ini, Anda akan kehilangan kemampuan untuk mengubah nama kolom.

> **Note**
>
> Di mesin Athena versi 2, ketika tabel ORC diatur untuk dibaca berdasarkan nama, Athena mengharuskan semua nama kolom dalam file ORC berada dalam huruf kecil. Karena Apache Spark tidak nama field huruf kecil saat menghasilkan file ORC, Athena mungkin tidak dapat membaca data sehingga dihasilkan. Solusinya adalah mengganti nama kolom menjadi huruf kecil, atau menggunakan mesin Athena versi 3.

Contoh berikut menggambarkan bagaimana mengubah ORC untuk membuatnya dibaca dengan nama:

```
CREATE EXTERNAL TABLE orders_orc_read_by_name (
   `o_comment` string,
   `o_orderkey` int,
   `o_custkey` int,
   `o_orderpriority` string,
   `o_orderstatus` string,
   `o_clerk` string,
   `o_shippriority` int,
   `o_orderdate` string
)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.orc.OrcSerde'
WITH SERDEPROPERTIES (
  'orc.column.index.access'='false')
STORED AS INPUTFORMAT
  'org.apache.hadoop.hive.ql.io.orc.OrcInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat'
LOCATION 's3://DOC-EXAMPLE-BUCKET/orders_orc/';
```

Parket: Baca dengan nama

Tabel diParket dibaca dengan namaSecara default. Ini didefinisikan oleh sintaks berikut:

```
WITH SERDEPROPERTIES (
  'parquet.column.index.access'='false')
```

Membaca dengan namamemungkinkan Anda untuk menambahkan kolom di tengah tabel dan menghapus kolom. Tapi kemudian Anda kehilangan kemampuan untuk mengganti nama kolom.

Untuk membuat Parket dibaca oleh indeks, yang akan memungkinkan Anda untuk mengganti nama kolom, Anda harus membuat tabel dengan `parquet.column.index.access` SerDe properti diatur ke. `true`

Jenis pembaruan

Topik ini menjelaskan beberapa perubahan yang dapat Anda buat pada skema dalam `CREATE TABLE` pernyataan tanpa benar-benar mengubah data Anda. Kami meninjau setiap jenis pembaruan

skema dan menentukan format data mana yang memungkinkan mereka di Athena. Untuk memperbarui skema, Anda dapat dalam beberapa kasus menggunakan `ALTER TABLE` perintah, tetapi dalam kasus lain Anda tidak benar-benar memodifikasi tabel yang ada. Sebagai gantinya, Anda membuat tabel dengan nama baru yang memodifikasi skema yang Anda gunakan dalam pernyataan asli `CREATE TABLE` Anda.

- Menambahkan kolom di awal atau di tengah tabel
- Menambahkan kolom di akhir tabel
- Menghapus kolom
- Mengganti nama kolom
- Menata ulang kolom
- Mengubah tipe data kolom

Tergantung pada bagaimana Anda mengharapkan skema Anda untuk berkembang, untuk terus menggunakan Athena kueri, memilih format data yang kompatibel.

Pertimbangkan aplikasi yang membaca informasi pesanan dari `orders` tabel yang ada dalam dua format: CSV dan Parket.

Contoh berikut akan membuat tabel di Parquet.

```
CREATE EXTERNAL TABLE orders_parquet (
   `orderkey` int,
   `orderstatus` string,
   `totalprice` double,
   `orderdate` string,
   `orderpriority` string,
   `clerk` string,
   `shippriority` int
) STORED AS PARQUET
LOCATION 's3://DOC-EXAMPLE-BUCKET/orders_ parquet/';
```

Contoh berikut membuat tabel yang sama di CSV:

```
CREATE EXTERNAL TABLE orders_csv (
   `orderkey` int,
   `orderstatus` string,
   `totalprice` double,
   `orderdate` string,
```

```
   `orderpriority` string,
   `clerk` string,
   `shippriority` int
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION 's3://DOC-EXAMPLE-BUCKET/orders_csv/';
```

Pada bagian berikut, kami meninjau bagaimana pembaruan untuk tabel ini mempengaruhi permintaan Athena.

Menambahkan kolom di awal atau di tengah tabel

Menambahkan kolom adalah salah satu perubahan skema yang paling sering. Misalnya, Anda dapat menambahkan kolom baru untuk memperkaya tabel dengan data baru. Atau, Anda dapat menambahkan kolom baru jika sumber untuk kolom yang ada telah berubah, dan menjaga versi sebelumnya dari kolom ini, untuk menyesuaikan aplikasi yang bergantung pada mereka.

Untuk menambahkan kolom di awal atau di tengah tabel, dan terus menjalankan kueri terhadap tabel yang ada, gunakan AVRO, JSON, dan Parket dan ORC jika SerDe properti mereka diatur untuk dibaca berdasarkan nama. Untuk informasi, lihat Akses indeks di ORC dan parket.

Jangan menambahkan kolom di awal atau di tengah tabel di CSV dan TSV, karena format ini bergantung pada pemesanan. Menambahkan kolom dalam kasus tersebut akan menyebabkan kesalahan ketidakcocokan skema saat skema partisi berubah.

Contoh berikut membuat tabel baru yang menambahkan `o_comment` kolom di tengah tabel berdasarkan data JSON.

```
CREATE EXTERNAL TABLE orders_json_column_addition (
   `o_orderkey` int,
   `o_custkey` int,
   `o_orderstatus` string,
   `o_comment` string,
   `o_totalprice` double,
   `o_orderdate` string,
   `o_orderpriority` string,
   `o_clerk` string,
   `o_shippriority` int,
)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 's3://DOC-EXAMPLE-BUCKET/orders_json/';
```

Menambahkan kolom di akhir tabel

Jika Anda membuat tabel dalam salah satu format yang didukung Athena, seperti Parquet, ORC, Avro, JSON, CSV, dan TSV, Anda dapat menggunakan`ALTER TABLE ADD COLUMNS`pernyataan untuk menambahkan kolom setelah kolom yang ada tapi sebelum kolom partisi.

Contoh berikut menambahkan`comment`kolom pada akhir`orders_parquet`tabel sebelum kolom partisi:

```
ALTER TABLE orders_parquet ADD COLUMNS (comment string)
```

> **Note**
>
> Untuk melihat kolom tabel baru di Athena Kueri Editor setelah Anda menjalankan`ALTER TABLE ADD COLUMNS`, secara manual refresh daftar tabel di editor, dan kemudian memperluas tabel lagi.

Menghapus kolom

Anda mungkin perlu menghapus kolom dari tabel jika mereka tidak lagi berisi data, atau untuk membatasi akses ke data di dalamnya.

- Anda dapat menghapus kolom dari tabel di JSON, Avro, dan Parquet dan ORC jika mereka dibaca oleh nama. Untuk informasi, lihat Akses indeks di ORC dan parket.
- Kami tidak menyarankan untuk menghapus kolom dari tabel di CSV dan TSV jika Anda ingin mempertahankan tabel yang telah Anda buat di Athena. Menghapus kolom istirahat skema dan mengharuskan Anda membuat tabel tanpa kolom dihapus.

Dalam contoh ini, menghapus kolom`` `totalprice` ``dari tabel di Parquet dan menjalankan kueri. Di Athena, Parquet dibaca dengan nama secara default, ini adalah mengapa kita menghilangkan konfigurasi SERDEPROPERTIES yang menentukan membaca dengan nama. Perhatikan bahwa kueri berikut berhasil, meski Anda mengubah skema:

```
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/';
```

Mengganti nama kolom

Anda mungkin ingin mengubah nama kolom dalam tabel Anda untuk memperbaiki ejaan, membuat nama kolom lebih deskriptif, atau untuk menggunakan kembali kolom yang ada untuk menghindari penataan ulang kolom.

Anda dapat mengubah nama kolom jika Anda menyimpan data Anda di CSV dan TSV, atau di Parquet dan ORC yang dikonfigurasi untuk dibaca berdasarkan indeks. Untuk informasi, lihat Akses indeks di ORC dan parket.

Athena membaca data dalam CSV dan TSV dalam urutan kolom dalam skema dan mengembalikan mereka dalam urutan yang sama. Ini tidak menggunakan nama kolom untuk pemetaan data ke kolom, itulah sebabnya Anda dapat mengubah nama kolom di CSV atau TSV tanpa melanggar permintaan Athena.

Salah satu strategi untuk mengubah nama kolom adalah untuk membuat tabel baru berdasarkan data yang mendasari sama, tetapi menggunakan nama kolom baru. Contoh berikut akan membuat `orders_parquet` baru menggunakan `orders_parquet_column_renamed`. Contoh mengubah nama `` `o_totalprice` `` kolom ke `` `o_total_price` ``, kemudian menjalankan kueri di Athena:

```
CREATE EXTERNAL TABLE orders_parquet_column_renamed (
    `o_orderkey` int,
    `o_custkey` int,
    `o_orderstatus` string,
    `o_total_price` double,
    `o_orderdate` string,
    `o_orderpriority` string,
    `o_clerk` string,
    `o_shippriority` int,
    `o_comment` string
)
STORED AS PARQUET
```

```
LOCATION 's3://DOC-EXAMPLE-BUCKET/orders_parquet/';
```

Dalam kasus tabel Parquet, kueri berikut berjalan, tetapi kolom berganti nama tidak menunjukkan data karena kolom sedang diakses oleh nama (default di Parquet) bukan oleh indeks:

```
SELECT *
FROM orders_parquet_column_renamed;
```

Kueri dengan tabel di CSV terlihat serupa:

```
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/';
```

Dalam kasus tabel CSV, kueri berikut berjalan dan menampilkan data di semua kolom, termasuk salah satu yang diubah namanya:

```
SELECT *
FROM orders_csv_column_renamed;
```

Menata ulang kolom

Anda dapat menyusun ulang kolom hanya untuk tabel dengan data dalam format yang dibaca berdasarkan nama, seperti JSON atau Parquet, yang berbunyi berdasarkan nama secara default. Anda juga dapat membuat ORC dibaca berdasarkan nama, jika diperlukan. Untuk informasi, lihat Akses indeks di ORC dan parket.

Contoh berikut membuat tabel baru dengan kolom dalam urutan yang berbeda:

```
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/';
```

Mengubah tipe data kolom

Anda mungkin ingin menggunakan jenis kolom yang berbeda ketika tipe yang ada tidak dapat lagi menyimpan jumlah informasi yang diperlukan. Misalnya, nilai kolom ID mungkin melebihi ukuran tipe `INT` data dan memerlukan penggunaan tipe `BIGINT` data.

Saat berencana menggunakan tipe data yang berbeda untuk kolom, pertimbangkan hal-hal berikut:

- Dalam kebanyakan kasus, Anda tidak dapat mengubah tipe data kolom secara langsung. Sebagai gantinya, Anda membuat ulang tabel Athena dan menentukan kolom dengan tipe data baru.
- Hanya tipe data tertentu yang dapat dibaca sebagai tipe data lainnya. Lihat tabel di bagian ini untuk tipe data yang dapat diperlakukan sedemikian rupa.
- Untuk data di Parket dan ORC, Anda tidak dapat menggunakan tipe data yang berbeda untuk kolom jika tabel tidak dipartisi.
- Untuk tabel yang dipartisi di Parquet dan ORC, tipe kolom partisi dapat berbeda dari tipe kolom partisi lain, dan Athena akan`CAST`ke jenis yang diinginkan, jika memungkinkan. Untuk informasi, lihat Menghindari kesalahan ketidakcocokan skema untuk tabel dengan partisi.
- Untuk tabel yang dibuat menggunakan LazySimpleSerDesatu-satunya, dimungkinkan untuk menggunakan `ALTER TABLE REPLACE COLUMNS` pernyataan untuk mengganti kolom yang ada dengan tipe data yang berbeda, tetapi semua kolom yang ada yang ingin Anda simpan juga harus didefinisikan ulang dalam pernyataan, atau mereka akan dihapus. Untuk informasi selengkapnya, lihat ALTER TABLE REPLACE COLUMNS.
- Hanya untuk tabel Apache Iceberg, Anda dapat menggunakan MENGUBAH KOLOM PERUBAHAN TABEL pernyataan untuk mengubah tipe data kolom. `ALTER TABLE REPLACE COLUMNS`tidak didukung untuk tabel Iceberg. Untuk informasi selengkapnya, lihat Skema tabel Gunung Es yang Berkembang.

> **Important**
>
> Kami sangat menyarankan agar Anda menguji dan memverifikasi pertanyaan Anda sebelum melakukan terjemahan tipe data. Jika Athena tidak dapat menggunakan tipe data target, `CREATE TABLE` kueri mungkin gagal.

Tabel berikut mencantumkan tipe data yang diperlakukan sebagai tipe data lainnya:

Tipe data yang kompatibel

Tipe data asli	Tipe data target yang tersedia
`STRING`	`BYTE`, `TINYINT`, `SMALLINT`, `INT`, `BIGINT`
`BYTE`	`TINYINT`, `SMALLINT`, `INT`, `BIGINT`
`TINYINT`	`SMALLINT`, `INT`, `BIGINT`
`SMALLINT`	`INT`, `BIGINT`
`INT`	`BIGINT`
`FLOAT`	`DOUBLE`

Contoh berikut menggunakan `CREATE TABLE` pernyataan untuk tabel asli untuk membuat `orders_json` tabel baru yang disebut`orders_json_bigint`. Tabel baru menggunakan `BIGINT` bukan `INT` sebagai tipe data untuk `` `o_shippriority` `` kolom.

```
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';
```

Kueri berikut berjalan dengan sukses, mirip dengan aslinya`SELECT`query, sebelum perubahan tipe data:

```
Select * from orders_json
LIMIT 10;
```

Pembaruan dalam tabel dengan partisi

Di Athena, tabel dan partisi yang harus menggunakan format data yang sama tetapi skema mereka mungkin berbeda. Saat Anda membuat partisi baru, partisi yang biasanya mewarisi skema tabel. Seiring waktu, skema mungkin mulai berbeda. Alasan meliputi:

- Jika skema tabel Anda perubahan, skema untuk partisi tidak diperbarui untuk tetap sinkron dengan tabel skema.
- AWS Glue Crawler memungkinkan Anda menemukan data dalam partisi dengan skema yang berbeda. Ini berarti bahwa jika Anda membuat tabel di Athena dengan AWS Glue, setelah crawler selesai diproses, skema untuk tabel dan partisinya mungkin berbeda.
- Jika Anda menambahkan partisi secara langsung menggunakan AWS API.

Athena memproses tabel dengan partisi berhasil jika mereka memenuhi kendala berikut. Jika kendala ini tidak terpenuhi, Athena mengeluarkan`HIVE_PARTITION_SCHEMA_MISMATCH`kesalahan.

- Setiap skema partisi kompatibel dengan skema tabel.
- Format data tabel memungkinkan jenis pembaruan yang ingin Anda lakukan: menambah, menghapus, menyusun ulang kolom, atau mengubah jenis data kolom.

  Misalnya, untuk format CSV dan TSV, Anda dapat mengganti nama kolom, menambahkan kolom baru di ujung tabel, dan mengubah jenis data kolom jika jenisnya kompatibel, tetapi Anda tidak dapat menghapus kolom. Untuk format lain, Anda dapat menambahkan atau menghapus kolom, atau mengubah jenis data kolom yang lain jika jenis kompatibel. Untuk informasi selengkapnya, lihat Pembaruan dan Format Data di Athena.

Menghindari kesalahan ketidakcocokan skema untuk tabel dengan partisi

Pada awal eksekusi kueri, Athena memverifikasi skema tabel dengan memeriksa bahwa setiap jenis data kolom kompatibel antara tabel dan partisi.

- Untuk jenis penyimpanan data Parquet dan ORC, Athena bergantung pada nama kolom dan menggunakannya untuk verifikasi skema berbasis nama kolom. Ini menghilangkan`HIVE_PARTITION_SCHEMA_MISMATCH`kesalahan untuk tabel dengan partisi di Parquet dan ORC. (Ini berlaku untuk ORC jika SerDe properti diatur untuk mengakses indeks dengan nama:`orc.column.index.access=FALSE`. Parket membaca indeks dengan nama secara default).
- Untuk CSV, JSON, dan Avro, Athena menggunakan verifikasi skema berbasis indeks. Ini berarti bahwa jika Anda mengalami kesalahan ketidakcocokan skema, Anda harus drop partisi yang menyebabkan ketidakcocokan skema dan menciptakan itu, sehingga Athena dapat mengkueri tanpa gagal.

Athena membandingkan skema tabel untuk skema partisi. Jika Anda membuat tabel di CSV, JSON, dan AVRO di Athena dengan AWS Glue Crawler, setelah Crawler selesai diproses, skema untuk tabel dan partisinya mungkin berbeda. Jika ada ketidakcocokan antara skema tabel dan skema partisi, pertanyaan Anda gagal di Athena karena kesalahan verifikasi skema yang mirip dengan ini:'crawler_test.click_avro' dinyatakan sebagai tipe 'string', tetapi partisi 'partition_0=2017-01-17' menyatakan kolom 'col68' sebagai tipe 'double'.”

Solusi khas untuk kesalahan tersebut adalah untuk menjatuhkan partisi yang menyebabkan kesalahan dan menciptakan itu. Lihat informasi yang lebih lengkap di ALTER TABLE DROP PARTITION dan ALTER TABLE ADD PARTITION.

Permintaan array

Amazon Athena memungkinkan Anda membuat larik, menggabungkan mereka, mengonversi mereka ke tipe data yang berbeda, kemudian memfilter, meratakan, dan mengurutkan mereka.

Topik

- Array penyaringan
- Menyortir array
- Menggunakan fungsi agregasi dengan array
- Mengonversi array ke string
- Menggunakan array untuk membuat peta
- Query array dengan tipe kompleks dan struktur bersarang

Membuat array

Untuk membangun sebuah larik literal di Athena, gunakanARRAYkata kunci, diikuti dengan tanda kurung`[ ]`, dan termasuk elemen larik dipisahkan dengan koma.

Contoh

Kueri ini menciptakan satu larik dengan empat elemen.

```
SELECT ARRAY [1,2,3,4] AS items
```

Ini menghasilkan:

```
+-----------+
| items     |
+-----------+
| [1,2,3,4] |
+-----------+
```

Kueri ini menciptakan dua larik.

```
SELECT ARRAY[ ARRAY[1,2], ARRAY[3,4] ] AS items
```

Ini menghasilkan:

```
+--------------------+
| items              |
+--------------------+
| [[1, 2], [3, 4]]   |
+--------------------+
```

Untuk membuat larik dari kolom yang dipilih dari tipe yang kompatibel, gunakan kueri, seperti dalam contoh ini:

```
WITH
dataset AS (
  SELECT 1 AS x, 2 AS y, 3 AS z
)
SELECT ARRAY [x,y,z] AS items FROM dataset
```

Kueri ini kembali:

```
+-----------+
| items     |
+-----------+
| [1,2,3]   |
+-----------+
```

Pada contoh berikut, dua larik dipilih dan dikembalikan sebagai pesan selamat datang.

```
WITH
dataset AS (
  SELECT
    ARRAY ['hello', 'amazon', 'athena'] AS words,
    ARRAY ['hi', 'alexa'] AS alexa
)
SELECT ARRAY[words, alexa] AS welcome_msg
FROM dataset
```

Kueri ini kembali:

```
+----------------------------------------+
| welcome_msg                            |
+----------------------------------------+
| [[hello, amazon, athena], [hi, alexa]] |
+----------------------------------------+
```

Untuk membuat larik pasangan nilai kunci, gunakan`MAP`operator yang mengambil larik kunci diikuti oleh larik nilai, seperti dalam contoh ini:

```
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
```

Kueri ini menghasilkan:

```
+-----------------------------------------------------------------------------------------------
+
| people
               |
+-----------------------------------------------------------------------------------------------
+
| [{last=Smith, first=Bob, age=40}, {last=Doe, first=Jane, age=30}, {last=Smith,
 first=Billy, age=8}] |
+-----------------------------------------------------------------------------------------------
+
```

Menggabungan string dan array

Senar penggabungan

Untuk menggabungkan dua string, Anda dapat menggunakan alur ganda | | operator, seperti dalam contoh berikut.

```
SELECT 'This' || ' is' || ' a' || ' test.' AS Concatenated_String
```

Kueri ini kembali:

#	Concatenated_String
1	This is a test.

Anda dapat menggunakan`concat()`fungsi untuk mencapai hasil yang sama.

```
SELECT concat('This', ' is', ' a', ' test.') AS Concatenated_String
```

Kueri ini menghasilkan:

#	Concatenated_String
1	This is a test.

Anda dapat menggunakan `concat_ws()` fungsi untuk menggabungkan string dengan pemisah yang ditentukan dalam argumen pertama.

```
SELECT concat_ws(' ', 'This', 'is', 'a', 'test.') as Concatenated_String
```

Kueri ini menghasilkan:

#	Concatenated_String
1	This is a test.

Untuk menggabungkan dua kolom tipe data string menggunakan titik, rujuk dua kolom menggunakan tanda kutip ganda, dan lampirkan titik dalam tanda kutip tunggal sebagai string hard-code. Jika kolom bukan tipe data string, Anda dapat menggunakan `CAST("column_name" as VARCHAR)` untuk mentransmisikan kolom terlebih dahulu.

```
SELECT "col1" || '.' || "col2" as Concatenated_String
FROM my_table
```

Kueri ini menghasilkan:

#	Concatenated_String
1	col1_string_value .col2_string_value

Array gabungan

Anda dapat menggunakan teknik yang sama untuk menggabungkan larik.

Untuk menggabungkan beberapa larik, menggunakan alur ganda | | Operator.

```
SELECT ARRAY [4,5] || ARRAY[ ARRAY[1,2], ARRAY[3,4] ] AS items
```

Kueri ini menghasilkan:

#	item
1	[[4, 5], [1, 2], [3, 4]]

Untuk menggabungkan beberapa larik ke dalam satu larik, menggunakan operator alur ganda atau`concat()`fungsi.

```
WITH
dataset AS (
  SELECT
    ARRAY ['Hello', 'Amazon', 'Athena'] AS words,
    ARRAY ['Hi', 'Alexa'] AS alexa
)
SELECT concat(words, alexa) AS welcome_msg
FROM dataset
```

Kueri ini menghasilkan:

#	selamat datang_msg
1	[Hello, Amazon, Athena, Hi, Alexa]

Untuk informasi selengkapnya tentang fungsi string `concat()` lainnya, lihat Fungsi dan operator String dalam dokumentasi Trino.

Mengonversi tipe data array

Untuk mengonversi data dalam larik untuk tipe data yang didukung, gunakanCASTOperator, sebagai`CAST(value AS type)`. Athena mendukung semua tipe data Presto asli.

```
SELECT
   ARRAY [CAST(4 AS VARCHAR), CAST(5 AS VARCHAR)]
AS items
```

Kueri ini kembali:

```
+-------+
| items |
+-------+
| [4,5] |
+-------+
```

Buat dua larik dengan elemen pasangan kunci-nilai, mengonversi mereka ke JSON, dan menggabungkan, seperti dalam contoh ini:

```
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
```

Kueri ini menghasilkan:

```
+-------------------------------------------------+
| items                                           |
+-------------------------------------------------+
| [{"a1":1,"a2":2,"a3":3}, {"b1":4,"b2":5,"b3":6}] |
+-------------------------------------------------+
```

Menemukan panjang

Fungsi `cardinality` mengembalikan panjang larik, seperti dalam contoh ini:

```
SELECT cardinality(ARRAY[1,2,3,4]) AS item_count
```

Kueri ini mengembalikan:

```
+------------+
| item_count |
+------------+
| 4          |
+------------+
```

Mengakses elemen array

Untuk mengakses elemen larik, gunakan`[]`operator, dengan 1 menentukan elemen pertama, 2 menentukan elemen kedua, dan seterusnya, seperti dalam contoh ini:

```
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
```

Kueri ini kembali:

```
+------------------------+
| item                   |
+------------------------+
| {"a1":1,"a2":2,"a3":3} |
+------------------------+
```

Untuk mengakses elemen larik pada posisi tertentu (dikenal sebagai posisi indeks), gunakan`element_at()`fungsi dan menentukan nama larik dan posisi indeks:

- Jika indeks lebih besar dari 0,`element_at()`mengembalikan elemen yang Anda tentukan, menghitung dari awal sampai akhir larik. Berperilaku sebagai`[]`Operator.
- Jika indeks kurang dari 0,`element_at()`mengembalikan elemen menghitung dari akhir ke awal larik.

Kueri berikut membuat larik`words`, dan memilih elemen pertama`hello`dari itu sebagai`first_word`, elemen kedua`amazon`(menghitung dari akhir larik) sebagai`middle_word`, dan elemen ketiga`athena`, sebagai`last_word`.

```
WITH dataset AS (
  SELECT ARRAY ['hello', 'amazon', 'athena'] AS words
)
SELECT
  element_at(words, 1) AS first_word,
  element_at(words, -2) AS middle_word,
  element_at(words, cardinality(words)) AS last_word
FROM dataset
```

Kueri ini menghasilkan:

```
+----------------------------------------+
```

```
| first_word  | middle_word | last_word  |
+---------------------------------------+
| hello       | amazon      | athena     |
+---------------------------------------+
```

Meratakan array bersarang

Saat bekerja dengan larik nest, Anda sering perlu memperluas elemen larik nest ke dalam satu larik, atau memperluas larik menjadi beberapa baris.

Contoh

Untuk meratakan elemen larik nest ini ke dalam satu larik nilai, gunakan`flatten`fungsi. Kueri ini mengembalikan baris untuk setiap elemen dalam larik.

```
SELECT flatten(ARRAY[ ARRAY[1,2], ARRAY[3,4] ]) AS items
```

Kueri ini kembali:

```
+-----------+
| items     |
+-----------+
| [1,2,3,4] |
+-----------+
```

Untuk meratakan larik menjadi beberapa baris, gunakan`CROSS JOIN`dalam hubungannya dengan`UNNEST`operator, seperti dalam contoh ini:

```
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)
```

Kueri ini kembali:

```
+----------------------+
| department  | names  |
```

```
+----------------------+
| engineering | Sharon |
+----------------------|
| engineering | John   |
+----------------------|
| engineering | Bob    |
+----------------------|
| engineering | Sally  |
+----------------------+
```

Untuk meratakan larik pasangan kunci-nilai, transpose kunci yang dipilih ke dalam kolom, seperti dalam contoh ini:

```
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)
```

Kueri ini kembali:

```
+-------------------------------------+
| first_name | last_name | department  |
+-------------------------------------+
| Bob        | Smith     | engineering |
| Jane       | Doe       | engineering |
| Billy      | Smith     | engineering |
+-------------------------------------+
```

Dari daftar karyawan, pilih karyawan dengan skor gabungan tertinggi.`UNNEST`dapat digunakan dalam`FROM`klausul tanpa sebelumnya`CROSS JOIN`karena merupakan default bergabung operator dan karena itu tersirat.

```
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
```

Kueri ini kembali:

```
+---------------------------------+
| name | department | total_score |
+---------------------------------+
| Amy  | devops     | 54          |
+---------------------------------+
```

Dari daftar karyawan, pilih karyawan dengan skor individu tertinggi.

```
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
```

Kueri ini menghasilkan:

```
+--------------+
| name | score |
+--------------+
| Amy  | 15    |
+--------------+
```

Pertimbangan dan batasan

Jika UNNEST digunakan pada satu atau lebih array dalam query, dan salah satu array adalahNULL, query tidak mengembalikan baris. Jika UNNEST digunakan pada array yang merupakan string kosong, string kosong dikembalikan.

Misalnya, dalam query berikut, karena array kedua adalah null, query tidak mengembalikan baris.

```
SELECT
    col1,
    col2
FROM UNNEST (ARRAY ['apples','oranges','lemons']) AS t(col1)
CROSS JOIN UNNEST (ARRAY []) AS t(col2)
```

Dalam contoh berikutnya, array kedua dimodifikasi untuk berisi string kosong. Untuk setiap baris, query mengembalikan nilai dalam col1 dan string kosong untuk nilai dalamcol2. String kosong dalam array kedua diperlukan agar nilai-nilai dalam array pertama dikembalikan.

```
SELECT
    col1,
```

```
    col2
FROM UNNEST (ARRAY ['apples','oranges','lemons']) AS t(col1)
CROSS JOIN UNNEST (ARRAY ['']) AS t(col2)
```

Membuat array dari subquery

Buat larik dari koleksi baris.

```
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)
```

Kueri ini kembali:

```
+-----------------+
| array_items     |
+-----------------+
| [1, 2, 3, 4, 5] |
+-----------------+
```

Untuk membuat larik nilai unik dari satu set baris, gunakan`distinct`kata kunci.

```
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)
```

Kueri ini akan mengembalikan hasil berikut. Perhatikan bahwa pemesanan tidak dijamin.

```
+-----------------+
| array_items     |
+-----------------+
| [1, 2, 3, 4, 5] |
+-----------------+
```

Untuk informasi selengkapnya tentang penggunaan `array_agg` fungsi, lihat Fungsi agregat dalam dokumentasi Trino.

Array penyaringan

Membuat larik dari koleksi baris jika mereka cocok dengan kriteria filter.

```
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
```

Kueri ini kembali:

```
+-------------+
| array_items |
+-------------+
| [4, 5]      |
+-------------+
```

Filter larik berdasarkan apakah salah satu unsur-unsurnya mengandung nilai tertentu, seperti 2, seperti dalam contoh ini:

```
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)
```

Kueri ini kembali:

```
+--------------+
| array_items  |
+--------------+
| [1, 2, 3, 4] |
+--------------+
```

Fungsi **`filter`**

```
 filter(ARRAY [list_of_values], boolean_function)
```

Anda dapat menggunakan filter fungsi pada ARRAY ekspresi untuk membuat array baru yang merupakan bagian dari item dalam list_of_values yang boolean_function adalah true. Fungsi `filter` dapat berguna dalam kasus saat Anda tidak dapat menggunakan fungsi *UNNEST*.

Contoh berikut filter untuk nilai yang lebih besar dari nol dalam array`[1,0,5,-1]`.

```
SELECT filter(ARRAY [1,0,5,-1], x -> x>0)
```

Hasil

`[1,5]`

Contoh berikut filter untuk nilai-nilai non-null dalam array. `[-1, NULL, 10, NULL]`

```
SELECT filter(ARRAY [-1, NULL, 10, NULL], q -> q IS NOT NULL)
```

Hasil

`[-1,10]`

Menyortir array

Untuk membuat larik diurutkan nilai-nilai unik dari satu set baris, Anda dapat menggunakanarray_sort, seperti dalam contoh berikut.

```
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)
```

Kueri ini menghasilkan:

```
+--------------------+
| array_items        |
+--------------------+
| [1, 2, 3, 4, 5, 6] |
+--------------------+
```

Untuk informasi tentang memperluas array menjadi beberapa baris, lihatMeratakan array bersarang.

Menggunakan fungsi agregasi dengan array

- Untuk menambahkan nilai dalam larik, gunakan`SUM`, seperti dalam contoh berikut.
- Untuk agregat beberapa baris dalam larik, gunakan`array_agg`. Untuk informasi, lihat Membuat array dari subquery.

> **Note**
>
> `ORDER BY`didukung untuk fungsi agregasi dimulai pada mesin Athena versi 2.

```
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;
```

Pada akhirnyaSELECTpernyataan, bukan menggunakansum( )danUNNEST, Anda dapat menggunakanreduce( )untuk mengurangi waktu pemrosesan dan transfer data, seperti dalam contoh berikut.

```
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;
```

DKueri berikut mengembalikan hasil ini. Urutan hasil yang dikembalikan tidak dijamin.

```
+----------------------+
| array_items  | total |
+----------------------+
| [1, 2, 3, 4] | 10    |
| [5, 6, 7, 8] | 26    |
| [9, 0]       | 9     |
+----------------------+
```

Mengonversi array ke string

Untuk mengonversi larik menjadi string tunggal, gunakanarray_joinfungsi. Contoh mandiri berikut membuat tabel yang disebutdatasetyang berisi sebuah larik alias disebutwords. Penggunaan kueriarray_joinuntuk bergabung dengan elemen larik diwords, memisahkan mereka dengan spasi, dan mengembalikan string yang dihasilkan dalam kolom alias disebutwelcome_msg.

```
WITH
dataset AS (
```

```
  SELECT ARRAY ['hello', 'amazon', 'athena'] AS words
)
SELECT array_join(words, ' ') AS welcome_msg
FROM dataset
```

Kueri ini menghasilkan:

```
+---------------------+
| welcome_msg         |
+---------------------+
| hello amazon athena |
+---------------------+
```

Menggunakan array untuk membuat peta

Peta adalah pasangan kunci-nilai yang terdiri dari tipe data yang tersedia di Athena. Untuk membuat peta, gunakanMAPoperator dan lulus dua larik: yang pertama adalah kolom (kunci) nama, dan yang kedua adalah nilai-nilai. Semua nilai dalam larik harus dari tipe yang sama. Jika salah satu elemen nilai larik peta harus dari berbagai tipe, Anda dapat mengonversi mereka nanti.

Contoh

Contoh ini memilih pengguna dari set data. MenggunakanMAPoperator dan melewati dua larik. Larik pertama mencakup nilai-nilai untuk nama kolom, seperti “pertama”, “terakhir”, dan “usia”. Larik kedua terdiri dari nilai untuk masing-masing kolom ini, seperti “Bob”, “Smith”, “35".

```
WITH dataset AS (
  SELECT MAP(
    ARRAY['first', 'last', 'age'],
    ARRAY['Bob', 'Smith', '35']
  ) AS user
)
SELECT user FROM dataset
```

Kueri ini kembali:

```
+--------------------------------+
| user                           |
+--------------------------------+
| {last=Smith, first=Bob, age=35} |
+--------------------------------+
```

Anda dapat mengambilMapnilai dengan memilih nama field diikuti oleh[key_name], seperti dalam contoh ini:

```
WITH dataset AS (
 SELECT MAP(
   ARRAY['first', 'last', 'age'],
   ARRAY['Bob', 'Smith', '35']
 ) AS user
)
SELECT user['first'] AS first_name FROM dataset
```

Kueri ini menghasilkan:

```
+------------+
| first_name |
+------------+
| Bob        |
+------------+
```

Query array dengan tipe kompleks dan struktur bersarang

Data sumber Anda sering berisi larik dengan tipe data yang kompleks dan struktur nest. Contoh dalam bagian ini menunjukkan bagaimana mengubah tipe data elemen, menemukan elemen dalam larik, dan menemukan kata kunci menggunakan kueri Athena.

- MembuatROW
- Mengubah nama field dalam array menggunakanCAST
- Penyaringan array menggunakan.notasi
- Penyaringan array dengan nilai bersarang
- Penyaringan array menggunakanUNNEST
- Menemukan kata kunci dalam array menggunakanregexp_like

Membuat**ROW**

> **Note**
>
> Contoh dalam bagian ini menggunakanROWsebagai sarana untuk membuat data sampel untuk bekerja dengan. Saat Anda kueri tabel dalam Athena, Anda tidak perlu

membuatROWjenis data, karena mereka telah dibuat dari sumber data Anda. Saat Anda menggunakanCREATE_TABLE, Athena mendefinisikanSTRUCTdi dalamnya, mengisi dengan data, dan menciptakanROWtipe data untuk Anda, untuk setiap baris dalam set data. Yang mendasariROWtipe data terdiri dari bidang bernama dari setiap tipe data SQL yang didukung.

```
WITH dataset AS (
 SELECT
   ROW('Bob', 38) AS users
 )
SELECT * FROM dataset
```

Kueri ini kembali:

```
+-------------------------+
| users                   |
+-------------------------+
| {field0=Bob, field1=38} |
+-------------------------+
```

Mengubah nama field dalam array menggunakan**CAST**

Untuk mengubah nama field dalam larik yang berisiROWnilai, Anda dapatCASTyangROWdeklarasi:

```
WITH dataset AS (
  SELECT
    CAST(
      ROW('Bob', 38) AS ROW(name VARCHAR, age INTEGER)
    ) AS users
)
SELECT * FROM dataset
```

Kueri ini kembali:

```
+--------------------+
| users              |
+--------------------+
| {NAME=Bob, AGE=38} |
+--------------------+
```

> **Note**
>
> Pada contoh di atas, Anda menyatakan`name`sebagai`VARCHAR`karena ini adalah jenisnya di Presto. Jika Anda menyatakan hal ini`STRUCT`di dalam`CREATE TABLE`pernyataan, gunakan`String`jenis karena Hive mendefinisikan tipe data ini sebagai`String`.

Penyaringan array menggunakan`.`notasi

Pada contoh berikut, pilih`accountId`bidang dari`userIdentity`kolomAWS CloudTraillog tabel dengan menggunakan titik`.`notasi. Untuk informasi selengkapnya, lihat, Mengkueri AWS CloudTrailLog.

```
SELECT
  CAST(useridentity.accountid AS bigint) as newid
FROM cloudtrail_logs
LIMIT 2;
```

Kueri ini kembali:

```
+--------------+
| newid        |
+--------------+
| 112233445566 |
+--------------+
| 998877665544 |
+--------------+
```

Untuk kueri larik nilai, masalah kueri ini:

```
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
```

Ini mengembalikan hasil ini:

```
+-----------------------------------------------------------------+
| users                                                           |
+-----------------------------------------------------------------+
| [{NAME=Bob, AGE=38}, {NAME=Alice, AGE=35}, {NAME=Jane, AGE=27}] |
+-----------------------------------------------------------------+
```

Penyaringan array dengan nilai bersarang

Larik besar sering mengandung struktur nest, dan Anda harus dapat memfilter, atau mencari, untuk nilai-nilai di dalamnya.

Untuk menentukan set data untuk larik nilai yang mencakup bersarang`BOOLEAN`nilai, masalah kueri ini:

```
WITH dataset AS (
  SELECT
    CAST(
      ROW('aws.amazon.com', ROW(true)) AS ROW(hostname VARCHAR, flaggedActivity
 ROW(isNew BOOLEAN))
    ) AS sites
)
SELECT * FROM dataset
```

Ini mengembalikan hasil ini:

```
+-------------------------------------------------------------+
| sites                                                       |
+-------------------------------------------------------------+
| {HOSTNAME=aws.amazon.com, FLAGGEDACTIVITY={ISNEW=true}}  |
+-------------------------------------------------------------+
```

Selanjutnya, untuk memfilter dan mengakses`BOOLEAN`nilai dari elemen itu, terus menggunakan titik`.`notasi.

```
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
```

Kueri ini memilih bidang nest dan mengembalikan hasil ini:

```
+------------------------+
| hostname       | isnew |
+------------------------+
| aws.amazon.com | true  |
+------------------------+
```

Penyaringan array menggunakan**UNNEST**

Untuk memfilter larik yang mencakup struktur nest oleh salah satu elemen anaknya, mengeluarkan kueri denganUNNESTOperator. Untuk informasi selengkapnya tentangUNNEST, lihatPerataan Larik Nest.

Misalnya, kueri ini menemukan nama host situs di dataset.

```
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
```

Ini menghasilkan:

```
+------------------------+
| hostname       | isnew |
+------------------------+
| aws.amazon.com | true  |
+------------------------+
```

Menemukan kata kunci dalam array menggunakan**regexp_like**

Contoh berikut menggambarkan cara mencari set data untuk kata kunci dalam elemen dalam larik, menggunakanregexp_likefungsi. Dibutuhkan sebagai masukan pola ekspresi reguler untuk mengevaluasi, atau daftar istilah dipisahkan oleh alur (|), mengevaluasi pola, dan menentukan apakah string yang ditentukan berisi itu.

Pola ekspresi reguler perlu terkandung dalam string, dan tidak harus mencocokkannya. Untuk mencocokkan seluruh string, lampirkan pola dengan ^ di awal itu, dan $ di akhir, seperti`'^pattern $'`.

Pertimbangkan array situs yang berisi nama host mereka, dan`flaggedActivity`elemen. Elemen ini mencakup`ARRAY`, berisi beberapa`MAP`elemen, setiap daftar kata kunci populer yang berbeda dan jumlah popularitas mereka. Asumsikan Anda ingin menemukan kata kunci tertentu di dalam `MAP` di larik ini.

Untuk mencari set data ini untuk situs dengan kata kunci tertentu, kami menggunakan`regexp_like`bukan SQL yang sama`LIKE`operator, karena mencari sejumlah besar kata kunci lebih efisien dengan`regexp_like`.

Example Contoh 1: Menggunakan**regexp_like**

Kueri dalam contoh ini menggunakan fungsi `regexp_like` untuk mencari istilah `'politics|bigdata'`, ditemukan dalam nilai-nilai dalam larik:

```
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)
```

Kueri ini mengembalikan dua situs:

```
+----------------+
| hostname       |
+----------------+
| aws.amazon.com |
+----------------+
| news.cnn.com   |
+----------------+
```

Example Contoh 2: Menggunakan**`regexp_like`**

Kueri dalam contoh berikut menambahkan skor popularitas total untuk situs yang cocok dengan istilah pencarian Anda dengan`regexp_like`fungsi, kemudian perintah mereka dari tertinggi ke terendah.

```
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
```

Kueri ini mengembalikan dua situs:

```
+-----------------------------------+
| hostname       | terms    | total  |
+----------------+------------------+
| news.cnn.com   | politics |  241   |
+----------------+------------------+
| aws.amazon.com | bigdata |  10    |
+----------------+------------------+
```

Menanyakan data geospasial

Data geospasial berisi pengidentifikasi yang menentukan posisi geografis untuk suatu objek. Contoh jenis data ini termasuk laporan cuaca, arah peta, tweet dengan posisi geografis, lokasi toko, dan rute maskapai penerbangan. Data geospasial memainkan peran penting dalam analisis bisnis, pelaporan, dan peramalan.

Pengidentifikasi geospasial, seperti garis lintang dan bujur, memungkinkan Anda mengonversi alamat surat apa pun menjadi satu set koordinat geografis.

Topik

Apa itu kueri geospasial?

kueri geospasial adalah jenis khusus dari kueri SQL didukung di Athena. Mereka berbeda dari kueri SQL non-spasial dengan cara berikut:

- Menggunakan jenis data geometri khusus berikut:`point`,`line`,`multiline`,`polygon`, dan`multipolygon`.
- Mengekspresikan hubungan antara tipe data geometri, seperti`distance`,`equals`,`crosses`,`touches`,`overlaps`,`disjoint`, dan lainnya.

Dengan menggunakan kueri geospasial di Athena, Anda dapat menjalankan operasi ini dan operasi serupa lainnya:

- Temukan jarak antara dua titik.
- Periksa apakah satu wilayah (poligon) berisi lain.
- Periksa apakah satu baris melintasi atau menyentuh garis atau poligon lain.

Misalnya, untuk mendapatkan`point`tipe data geometri dari nilai-nilai tipe`double`untuk koordinat geografis Gunung Rainier di Athena, gunakan`ST_Point (longitude, latitude)`Fungsi geospasial, seperti dalam contoh berikut.

```
ST_Point(-121.7602, 46.8527)
```

Format data input dan tipe data geometri

Untuk menggunakan fungsi geospasial di Athena, masukkan data Anda dalam format WKT, atau gunakan Hive JSON. SerDe Anda juga dapat menggunakan jenis data geometri yang didukung di Athena.

Format data masukan

Untuk menangani kueri geospasial, Athena mendukung input data dalam format data berikut:

- WKT (Teks Terkenal). Di Athena, WKT direpresentasikan sebagai tipe data atau`varchar(x)`. `string`
- Data geospasial yang dikodekan JSON. Untuk mengurai file JSON dengan data geospasial dan membuat tabel untuknya, Athena menggunakan Hive JSON. SerDe Untuk informasi lebih lanjut tentang menggunakan ini SerDe di Athena, lihat. Perpustakaan JSON SerDe

Jenis data geometri

Untuk menangani kueri geospasial, Athena mendukung tipe data geometri khusus ini:

- `point`
- `line`
- `polygon`
- `multiline`
- `multipolygon`

Fungsi geospasial yang didukung

Fungsi geospasial yang tersedia di Athena bergantung pada versi mesin yang Anda gunakan.

- Untuk informasi tentang fungsi geospasial di mesin Athena versi 3, lihat Fungsi geospasial dalam dokumentasi Trino.
- Untuk daftar perubahan nama fungsi dan fungsi baru pada mesin Athena versi 2, lihat. Perubahan nama fungsi geospasial dan fungsi baru di mesin Athena versi 2

Untuk informasi tentang versi mesin Athena, lihat Pembuatan versi mesin Athena.

Topik

- Fungsi geospasial di mesin Athena versi 3
- Fungsi geospasial di mesin Athena versi 2

Fungsi geospasial di mesin Athena versi 3

Untuk informasi tentang fungsi geospasial di mesin Athena versi 3, lihatFungsi geospasialdalam dokumentasi Trino.

Fungsi geospasial di mesin Athena versi 2

Topik ini mencantumkan fungsi geospasial ESRI yang didukung mulai dari mesin Athena versi 2. Untuk informasi selengkapnya tentang versi mesin Athena, lihat Pembuatan versi mesin Athena.

Perubahan mesin Athena versi 2

- Jenis input dan output untuk beberapa fungsi telah berubah. Terutama,`VARBINARY`tidak lagi didukung secara langsung untuk input. Untuk informasi selengkapnya, lihat Perubahan fungsi geospasial.

- Nama-nama beberapa fungsi geospasial telah berubah. Untuk informasi selengkapnya, lihat Perubahan nama fungsi geospasial di mesin Athena versi 2.
- Fungsi baru telah ditambahkan. Untuk informasi selengkapnya, lihat Fungsi geospasial baru di mesin Athena versi 2.

Athena mendukung jenis fungsi geospasial berikut:

- Fungsi konstruktor
- Fungsi hubungan geospasial
- Fungsi operasi
- Fungsi pengakses
- Fungsi agregasi
- Fungsi ubin Bing

Fungsi konstruktor

Gunakan fungsi konstruktor untuk mendapatkan representasi biner dari`point`,`line`, atau`polygon`Jenis data geometri Anda juga dapat menggunakan fungsi-fungsi ini untuk mengkonversi data biner untuk teks, dan mendapatkan nilai-nilai biner untuk data geometri yang dinyatakan sebagai teks terkenal (WKT).

`ST_AsBinary(geometry)`

Mengembalikan tipe data varbinary yang berisi representasi WKB dari geometri yang ditentukan. Contoh:

```
SELECT ST_AsBinary(ST_Point(-158.54, 61.56))
```

`ST_AsText(geometry)`

Mengkonversi masing-masing yang ditentukanJenis data geometrike teks. Mengembalikan nilai dalam tipe data varchar, yang merupakan representasi WKT dari tipe data geometri. Contoh:

```
SELECT ST_AsText(ST_Point(-158.54, 61.56))
```

`ST_GeomAsLegacyBinary(geometry)`

Mengembalikan varbinary warisan dari geometri tertentu. Contoh:

```
SELECT ST_GeomAsLegacyBinary(ST_Point(-158.54, 61.56)
```

ST_GeometryFromText(varchar)

Mengkonversi teks dalam format WKT menjadi tipe data geometri. Mengembalikan nilai dalam tipe data geometri. Contoh:

```
SELECT ST_GeometryFromText(ST_AsText(ST_Point(1, 2)))
```

ST_GeomFromBinary(varbinary)

Mengembalikan jenis objek geometri dari representasi WKB. Contoh:

```
SELECT ST_GeomFromBinary(ST_AsBinary(ST_Point(-158.54, 61.56)))
```

ST_GeomFromLegacyBinary(varbinary)

Mengembalikan objek tipe geometri dari jenis varbinary warisan. Contoh:

```
SELECT ST_GeomFromLegacyBinary(ST_GeomAsLegacyBinary(ST_Point(-158.54, 61.56)))
```

ST_LineFromText(varchar)

Mengembalikan nilai dalamJenis data geometri `line`. Contoh:

```
SELECT ST_Line('linestring(1 1, 2 2, 3 3)')
```

ST_LineString(array(point))

Pengembalian`LineString`jenis geometri yang terbentuk dari larik jenis titik geometri. Jika ada kurang dari dua titik tidak kosong dalam larik yang ditentukan, kosong`LineString`dikembalikan. Melempar pengecualian jika ada elemen dalam larik adalah nol, kosong, atau sama dengan yang sebelumnya. Geometri yang dikembalikan mungkin tidak sederhana. Tergantung pada input specfied, geometri kembali dapat diri berpotongan atau mengandung duplikat vertexes. Contoh:

```
SELECT ST_LineString(ARRAY[ST_Point(-158.54, 61.56), ST_Point(-158.55, 61.56)])
```

ST_MultiPoint(array(point))

PengembalianMultiPointobjek geometri terbentuk dari titik-titik tertentu. Mengembalikan null jika larik yang ditentukan kosong. Melempar pengecualian jika ada elemen dalam larik adalah nol atau kosong. Geometri kembali mungkin tidak sederhana dan dapat berisi duplikat poin jika larik yang ditentukan memiliki duplikat. Contoh:

```
SELECT ST_MultiPoint(ARRAY[ST_Point(-158.54, 61.56), ST_Point(-158.55, 61.56)])
```

ST_Point(double, double)

Mengembalikan jenis geometripointobjek. Untuk nilai input data ke fungsi ini, gunakan nilai geometrik, seperti nilai dalam sistem koordinat Cartesian Universal Transverse Mercator (UTM), atau unit peta geografis (bujur dan lintang) dalam derajat desimal. Nilai bujur dan lintang menggunakan Sistem Geodetik Dunia, juga dikenal sebagai WGS 1984, atau EPSG:4326. WGS 1984 adalah sistem koordinat yang digunakan oleh Global Positioning System (GPS).

Sebagai contoh, dalam notasi berikut, koordinat peta ditentukan dalam bujur dan lintang, dan nilai.072284, yang merupakan jarak buffer, ditentukan dalam unit sudut sebagai derajat desimal:

```
SELECT ST_Buffer(ST_Point(-74.006801, 40.705220), .072284)
```

Sintaksis:

```
SELECT ST_Point(longitude, latitude) FROM earthquakes LIMIT 1
```

Contoh berikut menggunakan koordinat bujur dan lintang tertentu:

```
SELECT ST_Point(-158.54, 61.56)
FROM earthquakes
LIMIT 1
```

Contoh berikutnya menggunakan koordinat bujur dan lintang tertentu:

```
SELECT ST_Point(-74.006801, 40.705220)
```

Contoh berikut menggunakanST_AsTextfungsi untuk mendapatkan geometri dari WKT:

```
SELECT ST_AsText(ST_Point(-74.006801, 40.705220)) AS WKT
```

ST_Polygon(varchar)

Menggunakan urutan ordinat yang disediakan searah jarum jam, kiri ke kanan, mengembalikanJenis data geometri `polygon`. Mulai dari mesin Athena versi 2, hanya poligon yang diterima sebagai input. Contoh:

```
SELECT ST_Polygon('polygon ((1 1, 1 4, 4 4, 4 1))')
```

to_geometry(sphericalGeography)

Mengembalikan objek geometri dari objek geografi bola yang ditentukan. Contoh:

```
SELECT to_geometry(to_spherical_geography(ST_Point(-158.54, 61.56)))
```

to_spherical_geography(geometry)

Mengembalikan objek geografi bola dari geometri yang ditentukan. Gunakan fungsi ini untuk mengubah objek geometri menjadi objek geografi bola pada bola radius Bumi. Fungsi ini hanya dapat digunakan pada`POINT`,`MULTIPOINT`,`LINESTRING`,`MULTILINESTRING`,`POLYGON`, dan`MULTIPOLYGON`geometri didefinisikan dalam ruang 2D atau`GEOMETRYCOLLECTION`geometri seperti itu. Untuk setiap titik geometri yang ditentukan, fungsi memverifikasi bahwa`point.x`berada di dalam`[-180.0, 180.0]`dan`point.y`berada di dalam`[-90.0, 90.0]`. Fungsi ini menggunakan titik-titik ini sebagai derajat bujur dan lintang untuk membangun bentuk`sphericalGeography`hyhasil.

Contoh:

```
SELECT to_spherical_geography(ST_Point(-158.54, 61.56))
```

Fungsi hubungan geospasial

Fungsi-fungsi berikut mengungkapkan hubungan antara dua geometri yang berbeda yang Anda tentukan sebagai masukan dan hasil kembali dari jenis`boolean`. Urutan tempat Anda menentukan sepasang geometri penting: nilai geometri pertama disebut geometri kiri, nilai geometri kedua disebut geometri kanan.

Fungsi ini kembali:

- `TRUE`jika dan hanya jika hubungan yang dijelaskan oleh fungsi puas.
- `FALSE`jika dan hanya jika hubungan yang dijelaskan oleh fungsi tidak puas.

ST_Contains(geometry, geometry)

PengembalianTRUEjika dan hanya jika geometri kiri berisi geometri yang tepat. Contoh:

```
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)

PengembalianTRUEjika dan hanya jika geometri kiri melintasi geometri yang tepat. Contoh:

```
SELECT ST_Crosses(ST_Line('linestring(1 1, 2 2 )'), ST_Line('linestring(0 1, 2 2)'))
```

ST_Disjoint(geometry, geometry)

PengembalianTRUEjika dan hanya jika persimpangan geometri kiri dan geometri kanan kosong. Contoh:

```
SELECT ST_Disjoint(ST_Line('linestring(0 0, 0 1)'), ST_Line('linestring(1 1, 1 0)'))
```

ST_Equals(geometry, geometry)

PengembalianTRUEjika dan hanya jika geometri kiri sama dengan geometri yang tepat. Contoh:

```
SELECT ST_Equals(ST_Line('linestring( 0 0, 1 1)'), ST_Line('linestring(1 3, 2 2)'))
```

ST_Intersects(geometry, geometry)

PengembalianTRUEjika dan hanya jika geometri kiri memotong geometri yang tepat. Contoh:

```
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)

PengembalianTRUEjika dan hanya jika geometri kiri tumpang tindih geometri yang tepat. Contoh:

```
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)

PengembalianTRUEjika dan hanya jika geometri kiri memiliki dimensi diperpanjang model sembilan persimpangan yang ditentukan (DE-9IM) hubungan dengan geometri yang betul. Yang ketiga (`varchar`) masukan mengambil hubungan. Contoh:

```
SELECT ST_Relate(ST_Line('linestring(0 0, 3 3)'), ST_Line('linestring(1 1, 4 4)'),
 'T********')
```

ST_Touches(geometry, geometry)

PengembalianTRUEjika dan hanya jika geometri kiri menyentuh geometri yang betul.

Contoh:

```
SELECT ST_Touches(ST_Point(8, 8), ST_Polygon('polygon((1  1, 1  4, 4  4, 4 1))'))
```

ST_Within(geometry, geometry)

PengembalianTRUEjika dan hanya jika geometri kiri berada dalam geometri yang tepat.

Contoh:

```
SELECT ST_Within(ST_Point(8, 8), ST_Polygon('polygon((1  1, 1  4, 4  4, 4 1))'))
```

Fungsi operasi

Gunakan fungsi operasi untuk melakukan operasi pada nilai-nilai tipe data geometri. Misalnya, Anda dapat memperoleh batas-batas tipe data geometri tunggal; persimpangan antara dua jenis data geometri; perbedaan antara geometri kiri dan kanan, tempat masing-masing adalah dari jenis data geometri yang sama; atau buffer eksterior atau cincin di sekitar tipe data geometri tertentu.

geometry_union(array(geometry))

Mengembalikan geometri yang mewakili titik set keunit geometri yang ditentukan. Contoh:

```
SELECT geometry_union(ARRAY[ST_Point(-158.54, 61.56), ST_Point(-158.55, 61.56)])
```

ST_Boundary(geometry)

Membawa sebagai masukan salah satu tipe data geometri dan mengembalikan`boundary`Jenis data geometri

Contoh:

```
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)

Membawa sebagai masukan salah satu tipe data geometri, seperti titik, garis, poligon, multiline, atau multipoligon, dan jarak sebagai tipe`double`). Mengembalikan tipe data geometri buffered oleh jarak yang ditentukan (atau radius). Contoh:

```
SELECT ST_Buffer(ST_Point(1, 2), 2.0)
```

Pada contoh berikut, koordinat peta ditentukan dalam bujur dan lintang, dan nilai`.072284`, yang merupakan jarak buffer, ditentukan dalam unit sudut sebagai derajat desimal:

```
SELECT ST_Buffer(ST_Point(-74.006801, 40.705220), .072284)
```

ST_Difference(geometry, geometry)

Mengembalikan geometri perbedaan antara geometri kiri dan geometri kanan. Contoh:

```
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)

Membawa sebagai masukan`line`,`polygon`,`multiline`, dan`multipolygon`Jenis data geometri Tidak mendukung`point`Jenis data geometri Mengembalikan amplop sebagai geometri, tempat amplop adalah persegi panjang di sekitar geometri tipe data yang ditentukan. Contoh:

```
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)

Mengembalikan larik dari dua titik yang mewakili sudut kiri bawah dan kanan atas dari sebuah geometri melompat-lompat poligon persegi panjang. Mengembalikan null jika geometri yang ditentukan kosong. Contoh:

```
SELECT ST_EnvelopeAsPts(ST_Point(-158.54, 61.56))
```

ST_ExteriorRing(geometry)

Mengembalikan geometri cincin eksterior dari jenis input`polygon`. Mulai dari mesin Athena versi 2, poligon adalah satu-satunya geometri yang diterima sebagai input. Contoh:

```
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)

Mengembalikan geometri persimpangan geometri kiri dan geometri kanan. Contoh:

```
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)

Mengembalikan geometri perbedaan geometris simetris antara geometri kiri dan geometri yang tepat. Contoh:

```
SELECT ST_AsText(ST_SymDifference(ST_Line('linestring(0 2, 2 2)'),
 ST_Line('linestring(1 2, 3 2)')))
```

ST_Union(geometry, geometry)

Mengembalikan tipe data geometri yang mewakili titik set keunit geometri yang ditentukan. Contoh:

```
SELECT ST_Union(ST_Point(-158.54, 61.56),ST_LineString(array[ST_Point(1,2),
 ST_Point(3,4)]))
```

Fungsi pengakses

fungsi accessor berguna untuk mendapatkan nilai-nilai dalam jenis`varchar`,`bigint`, atau`double`dari yang berbeda`geometry`Jenis data, tempat`geometry`adalah salah satu tipe data geometri didukung di Athena:`point`,`line`,`polygon`,`multiline`, dan`multipolygon`. Misalnya, Anda dapat memperoleh area`polygon`tipe data geometri, maksimum dan minimum X dan Y nilai untuk tipe data geometri tertentu, mendapatkan panjang`line`, atau menerima jumlah poin dalam tipe data geometri tertentu.

geometry_invalid_reason(geometry)

Pengembalian, dalam tipe data varchar, alasan mengapa geometri yang ditentukan tidak valid atau tidak sederhana. Jika geometri yang ditentukan tidak valid atau sederhana, mengembalikan alasan mengapa tidak valid. Jika geometri yang ditentukan adalah valid dan sederhana, mengembalikan null. Contoh:

```
SELECT geometry_invalid_reason(ST_Point(-158.54, 61.56))
```

great_circle_distance(latitude1, longitude1, latitude2, longitude2)

Mengembalikan, sebagai ganda, jarak lingkaran besar antara dua titik di permukaan bumi dalam kilometer. Contoh:

```
SELECT great_circle_distance(36.12, -86.67, 33.94, -118.40)
```

line_locate_point(lineString, point)

Mengembalikan ganda antara 0 dan 1 yang mewakili lokasi titik terdekat pada baris string yang ditentukan ke titik yang ditentukan sebagai fraksi dari total panjang garis 2d.

Mengembalikan null jika garis tertentu string atau titik kosong atau null. Contoh:

```
SELECT line_locate_point(ST_GeometryFromText('LINESTRING (0 0, 0 1)'), ST_Point(0,
 0.2))
```

simplify_geometry(geometry, double)

Menggunakan amer-douglas-peucker algoritma R untuk mengembalikan tipe data geometri yang merupakan versi sederhana dari geometri yang ditentukan. Menghindari menciptakan geometri yang berasal (khususnya, poligon) yang tidak valid. Contoh:

```
SELECT simplify_geometry(ST_GeometryFromText('POLYGON ((1 0, 2 1, 3 1, 3 1, 4 1, 1
 0))'), 1.5)
```

ST_Area(geometry)

Membawa sebagai masukan tipe data geometri dan mengembalikan area dalam tipe`double`. Contoh:

```
SELECT ST_Area(ST_Polygon('polygon((1 1, 4 1, 4 4, 1 4))'))
```

ST_Centroid(geometry)

Membawa sebagai masukanJenis data geometri `polygon`, dan mengembalikan`point`tipe data geometri yang merupakan pusat amplop poligon ini. Contoh:

```
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)

Mengembalikan tipe data geometri yang merupakan geometri cembung terkecil yang membungkus semua geometri dalam input tertentu. Contoh:

```
SELECT ST_ConvexHull(ST_Point(-158.54, 61.56))
```

ST_CoordDim(geometry)

Membawa sebagai masukan salah satu yang didukungJenis data geometri, dan mengembalikan jumlah komponen koordinat dalam jenis`tinyint`. Contoh:

```
SELECT ST_CoordDim(ST_Point(1.5,2.5))
```

ST_Dimension(geometry)

Membawa sebagai masukan salah satu yang didukungJenis data geometri, dan mengembalikan dimensi spasial geometri dalam jenistinyint. Contoh:

```
SELECT ST_Dimension(ST_Polygon('polygon((1 1, 4 1, 4 4, 1 4))'))
```

ST_Distance(geometry, geometry)

Pengembalian, berdasarkan ref spasial, ganda yang berisi dua dimensi minimum jarak Cartesian antara dua geometri di unit diproyeksikan. Mulai di mesin Athena versi 2, mengembalikan null jika salah satu input adalah geometri kosong. Contoh:

```
SELECT ST_Distance(ST_Point(0.0,0.0), ST_Point(3.0,4.0))
```

ST_Distance(sphericalGeography, sphericalGeography)

Pengembalian, sebagai ganda, jarak lingkaran besar antara dua titik geografi bola dalam meter. Contoh:

```
SELECT ST_Distance(to_spherical_geography(ST_Point(61.56,
 -86.67)),to_spherical_geography(ST_Point(61.56, -86.68)))
```

ST_EndPoint(geometry)

Mengembalikan titik terakhir darilinetipe data geometri dalampointJenis data geometri Contoh:

```
SELECT ST_EndPoint(ST_Line('linestring(0 2, 2 2)'))
```

ST_Geometries(geometry)

Mengembalikan larik geometri dalam koleksi tertentu. Jika geometri yang ditentukan bukan multi-geometri, mengembalikan larik satu elemen. Jika geometri yang ditentukan kosong, mengembalikan null.

Sebagai contoh, diberikanMultiLineStringobjek,ST_Geometriesmenciptakan sebuah arrayLineStringobjek. DiberikanGeometryCollectionobjek,ST_Geometriesmengembalikan larik un-rata dari konstituennya. Contoh:

```
SELECT ST_Geometries(GEOMETRYCOLLECTION(MULTIPOINT(0 0, 1 1),
 GEOMETRYCOLLECTION(MULTILINESTRING((2 2, 3 3)))))
```

Hasil:

```
array[MULTIPOINT(0 0, 1 1),GEOMETRYCOLLECTION(MULTILINESTRING((2 2, 3 3)))]
```

ST_GeometryN(geometry, index)

Pengembalian, sebagai tipe data geometri, elemen geometri pada indeks integer tertentu. Indeks mulai dari 1. Jika geometri yang ditentukan adalah kumpulan geometri (misalnya,`GEOMETRYCOLLECTION`atau`MULTI*`objek), mengembalikan geometri pada indeks tertentu. Jika indeks tertentu kurang dari 1 atau lebih besar dari jumlah elemen dalam koleksi, mengembalikan null. Untuk menemukan jumlah elemen, gunakan`ST_NumGeometries`. Geometri tunggal (misalnya,`POINT`,`LINESTRING`, atau`POLYGON`), diperlakukan sebagai koleksi satu elemen. geometri kosong diperlakukan sebagai koleksi kosong. Contoh:

```
SELECT ST_GeometryN(ST_Point(-158.54, 61.56),1)
```

ST_GeometryType(geometry)

Pengembalian, sebagai varchar, jenis geometri. Contoh:

```
SELECT ST_GeometryType(ST_Point(-158.54, 61.56))
```

ST_InteriorRingN(geometry, index)

Mengembalikan elemen cincin interior pada indeks tertentu (indeks mulai dari 1). Jika indeks yang diberikan kurang dari 1 atau lebih besar dari jumlah total cincin interior dalam geometri tertentu, mengembalikan null. Melempar kesalahan jika geometri yang ditentukan bukan poligon. Untuk menemukan jumlah elemen, gunakan`ST_NumInteriorRing`. Contoh:

```
SELECT ST_InteriorRingN(st_polygon('polygon ((0 0, 1 0, 1 1, 0 1, 0 0))'),1)
```

ST_InteriorRings(geometry)

Mengembalikan larik geometri dari semua cincin interior ditemukan dalam geometri tertentu, atau larik kosong jika poligon tidak memiliki cincin interior. Jika geometri yang ditentukan kosong, mengembalikan null. Jika geometri yang ditentukan bukan poligon, melempar kesalahan. Contoh:

```
SELECT ST_InteriorRings(st_polygon('polygon ((0 0, 1 0, 1 1, 0 1, 0 0))'))
```

ST_IsClosed(geometry)

Dibutuhkan sebagai masukan sajalinedanmultiline Jenis data geometri.
PengembalianTRUE(jenisboolean) jika dan hanya jika garis ditutup. Contoh:

```
SELECT ST_IsClosed(ST_Line('linestring(0 2, 2 2)'))
```

ST_IsEmpty(geometry)

Dibutuhkan sebagai masukan sajalinedanmultiline Jenis data geometri.
PengembalianTRUE(jenisboolean) jika dan hanya jika geometri tertentu kosong, dengan kata lain, ketikalineNilai mulai dan berakhir bertepatan. Contoh:

```
SELECT ST_IsEmpty(ST_Point(1.5, 2.5))
```

ST_IsRing(geometry)

PengembalianTRUE(jenisboolean) jika dan hanya jikalineditutup dan sederhana. Contoh:

```
SELECT ST_IsRing(ST_Line('linestring(0 2, 2 2)'))
```

ST_IsSimple(geometry)

Mengembalikan nilai true jika geometri yang ditentukan tidak memiliki titik geometris anomali (misalnya, persimpangan diri atau tangency diri). Untuk menentukan mengapa geometri tidak mudah, gunakangeometry_invalid_reason(). Contoh:

```
SELECT ST_IsSimple(ST_LineString(array[ST_Point(1,2), ST_Point(3,4)]))
```

ST_IsValid(geometry)

Mengembalikan nilai true jika dan hanya jika geometri yang ditentukan terbentuk dengan baik. Untuk menentukan mengapa geometri tidak terbentuk dengan baik, gunakangeometry_invalid_reason(). Contoh:

```
SELECT ST_IsValid(ST_Point(61.56, -86.68))
```

ST_Length(geometry)

Mengembalikan panjanglineJenisdouble. Contoh:

```
SELECT ST_Length(ST_Line('linestring(0 2, 2 2)'))
```

ST_NumGeometries(geometry)

Pengembalian, sebagai integer, jumlah geometri dalam koleksi. Jika geometri adalah kumpulan geometri (misalnya,GEOMETRYCOLLECTIONatauMULTI*objek), mengembalikan jumlah geometri. geometri tunggal kembali 1; geometri kosong kembali 0. Geometri kosong dalamGEOMETRYCOLLECTIONobjek dianggap sebagai salah satu geometri. Sebagai contoh, yang berikut bernilai SALAH:

```
ST_NumGeometries(ST_GeometryFromText('GEOMETRYCOLLECTION(MULTIPOINT EMPTY)'))
```

ST_NumInteriorRing(geometry)

Mengembalikan jumlah cincin interior dipolygongeometri dalam jenisbigint. Contoh:

```
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)

Mengembalikan jumlah poin dalam geometri dalam tipebigint. Contoh:

```
SELECT ST_NumPoints(ST_Point(1.5, 2.5))
```

ST_PointN(lineString, index)

Pengembalian, sebagai titik geometri tipe data, titik dari baris string yang ditentukan pada indeks integer yang ditentukan. Indeks mulai dari 1. Jika indeks yang diberikan kurang dari 1 atau lebih besar dari jumlah elemen dalam koleksi, mengembalikan null. Untuk menemukan jumlah elemen, gunakanST_NumPoints. Contoh:

```
SELECT ST_PointN(ST_LineString(array[ST_Point(1,2), ST_Point(3,4)]),1)
```

ST_Points(geometry)

Mengembalikan larik poin dari baris yang ditentukan objek geometri string. Contoh:

```
SELECT ST_Points(ST_LineString(array[ST_Point(1,2), ST_Point(3,4)]))
```

ST_StartPoint(geometry)

Mengembalikan titik pertama dari`linestring`tipe data geometri dalam`point`Jenis data geometri Contoh:

```
SELECT ST_StartPoint(ST_Line('linestring(0 2, 2 2)'))
```

ST_X(point)

Mengembalikan koordinat X dari titik dalam tipe`double`. Contoh:

```
SELECT ST_X(ST_Point(1.5, 2.5))
```

ST_XMax(geometry)

Mengembalikan X koordinat maksimum geometri dalam tipe`double`. Contoh:

```
SELECT ST_XMax(ST_Line('linestring(0 2, 2 2)'))
```

ST_XMin(geometry)

Mengembalikan minimal X koordinat geometri dalam tipe`double`. Contoh:

```
SELECT ST_XMin(ST_Line('linestring(0 2, 2 2)'))
```

ST_Y(point)

Mengembalikan koordinat Y dari titik dalam tipe`double`. Contoh:

```
SELECT ST_Y(ST_Point(1.5, 2.5))
```

ST_YMax(geometry)

Mengembalikan Y koordinat maksimum geometri dalam tipe`double`. Contoh:

```
SELECT ST_YMax(ST_Line('linestring(0 2, 2 2)'))
```

ST_YMin(geometry)

Mengembalikan Y koordinat minimum geometri dalam tipe`double`. Contoh:

```
SELECT ST_YMin(ST_Line('linestring(0 2, 2 2)'))
```

Fungsi agregasi

convex_hull_agg(geometry)

Mengembalikan geometri cembung minimum yang membungkus semua geometri berlalu sebagai masukan.

geometry_union_agg(geometry)

Mengembalikan geometri yang mewakili titik set keunit semua geometri berlalu sebagai masukan.

Fungsi ubin Bing

Fungsi berikut mengkonversi antara geometri dan ubin dalam sistem ubin peta Microsoft Bing.

bing_tile(x, y, zoom_level)

Mengembalikan sebuah objek ubin Bing dari koordinat integer`x`dan`y`dan level zoom yang ditentukan. Level zoom harus integer dari 1 sampai 23. Contoh:

```
SELECT bing_tile(10, 20, 12)
```

bing_tile(quadKey)

Mengembalikan sebuah objek ubin Bing dari quadkey. Contoh:

```
SELECT bing_tile(bing_tile_quadkey(bing_tile(10, 20, 12)))
```

bing_tile_at(latitude, longitude, zoom_level)

Mengembalikan sebuah objek ubin Bing di lintang, bujur, dan level zoom yang ditentukan. Garis lintang harus antara -85.05112878 dan 85.05112878. Garis bujur harus antara -180 dan 180. Parameter`latitude`dan`longitude`nilai harus`double`dan`zoom_level`integer. Contoh:

```
SELECT bing_tile_at(37.431944, -122.166111, 12)
```

bing_tiles_around(latitude, longitude, zoom_level)

Mengembalikan larik ubin Bing yang mengelilingi titik lintang dan bujur yang ditentukan pada level zoom yang ditentukan. Contoh:

```
SELECT bing_tiles_around(47.265511, -122.465691, 14)
```

bing_tiles_around(latitude, longitude, zoom_level, radius_in_km)

Pengembalian, pada level zoom yang ditentukan, larik ubin Bing. Larik berisi set minimum ubin Bing yang mencakup lingkaran radius yang ditentukan dalam kilometer di sekitar garis lintang dan bujur yang ditentukan. Parameter`latitude`,`longitude`, dan`radius_in_km`Nilai yang`double`; level zoom adalah`integer`. Contoh:

```
SELECT bing_tiles_around(37.8475, 112.596667, 10, .5)
```

bing_tile_coordinates(tile)

Pengembalian`x`dan`y`koordinat ubin Bing yang ditentukan. Contoh:

```
SELECT bing_tile_coordinates(bing_tile_at(37.431944, -122.166111, 12))
```

bing_tile_polygon(tile)

Mengembalikan representasi poligon dari ubin Bing yang ditentukan. Contoh:

```
SELECT bing_tile_polygon(bing_tile_at(47.265511, -122.465691, 4))
```

bing_tile_quadkey(tile)

Mengembalikan quadkey dari ubin Bing yang ditentukan. Contoh:

```
SELECT bing_tile_quadkey(bing_tile(52, 143, 10))
```

bing_tile_zoom_level(tile)

Mengembalikan level zoom dari ubin Bing ditentukan sebagai integer. Contoh:

```
SELECT bing_tile_zoom_level(bing_tile(52, 143, 10))
```

`geometry_to_bing_tiles(geometry, zoom_level)`

Mengembalikan set minimum ubin Bing yang sepenuhnya mencakup geometri yang ditentukan pada level zoom yang ditentukan. Level zoom dari 1 hingga 23 didukung. Contoh:

```
SELECT geometry_to_bing_tiles(ST_Point(61.56, 58.54), 10)
```

Perubahan nama fungsi geospasial dan fungsi baru di mesin Athena versi 2

Bagian ini berisi daftar perubahan nama fungsi geospasial dan fungsi geospasial yang baru di Athena mesin versi 2. Untuk informasi tentang perubahan lain pada mesin Athena versi 2, lihat. Versi mesin Athena 2

Untuk informasi tentang versioning mesin Athena, lihatPembuatan versi mesin Athena.

Perubahan nama fungsi geospasial di mesin Athena versi 2

Nama-nama fungsi berikut telah berubah. Dalam beberapa kasus, jenis input dan output juga berubah. Untuk informasi selengkapnya, lihat topik terkait.

Nama fungsi sebelumnya	Nama fungsi dimulai di mesin Athena versi 2
st_coordinate_dimension	ST_ CoordDim
st_end_point	ST_ EndPoint
st_exterior_ring	ST_ ExteriorRing
st_interior_ring_number	ST_ NumInteriorRing
st_geometry_from_text	ST_ GeometryFromText
st_is_closed	ST_ IsClosed
st_is_empty	ST_ IsEmpty
st_is_ring	ST_ IsRing

Nama fungsi sebelumnya	Nama fungsi dimulai di mesin Athena versi 2
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

Fungsi geospasial baru di mesin Athena versi 2

Fungsi geospasial berikut ini baru pada mesin Athena versi 2. Untuk informasi selengkapnya, lihat topik terkait.

Fungsi konstruktor

- ST_ AsBinary
- ST_ GeomAsLegacyBinary
- ST_ GeomFromBinary
- ST_ GeomFromLegacyBinary
- ST_ LineString
- ST_ MultiPoint
- to_geometri
- to_spherical_geografi

Fungsi operasi

- geometri_union
- ST_ EnvelopeAsPts
- ST_Union

Fungsi pengakses

- geometry_invalid_reason
- great_circle_distance
- line_locate_point
- sederhana_geometri
- ST_ ConvexHull
- ST_distance (geografi bola)
- ST_Geometri
- ST_Geometri
- ST_ GeometryType
- ST_ N InteriorRing
- ST_ InteriorRings
- ST_ IsSimple
- ST_ IsValid
- ST_ NumGeometries
- St_Pointn
- ST_poin

Fungsi agregasi

- convex_hull_agg
- geometri_union_agg

Fungsi ubin Bing

- bing_tile
- bing_tile (quadkey)
- bing_tile_at
- bing_tiles_around
- bing_tiles_around (radius)
- bing_tile_koordinat

- bing_tile_poligon
- bing_tile_quadkey
- bing_tile_zoom_level
- geometri_to_bing_tiles

Contoh: Kueri geospasial

Contoh dalam topik ini membuat dua tabel dari data sampel yang tersedia GitHub dan menanyakan tabel berdasarkan data. Data contoh, yang hanya untuk tujuan ilustrasi dan tidak dijamin akurat, adalah dalam file berikut:

- **`earthquakes.csv`**— Daftar gempa bumi yang terjadi di California. Contoh`earthquakes`tabel menggunakan bidang dari data ini.
- **`california-counties.json`**— Daftar data county untuk negara bagian California diFormat GeoJSON sesuai dengan ESRI. Data mencakup banyak bidang sepertiAREA,PERIMETER,STATE,COUNTY, danNAME, tetapi contoh`counties`tabel hanya menggunakan dua:Name(string), dan`BoundaryShape`(biner).

> **Note**
>
> Athena menggunakan`com.esri.json.hadoop.EnclosedEsriJsonInputFormat`untuk mengkonversi data JSON ke format biner geospasial.

Contoh berikut membuat direktori yang disebut `earthquakes`:

```
CREATE external TABLE earthquakes
(
 earthquake_date string,
 latitude double,
 longitude double,
 depth double,
 magnitude double,
 magtype string,
 mbstations string,
 gap string,
 distance string,
```

```
 rms string,
 source string,
 eventid string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE LOCATION 's3://DOC-EXAMPLE-BUCKET/my-query-log/csv/';
```

Contoh berikut membuat direktori yang disebut `counties`:

```
CREATE external TABLE IF NOT EXISTS counties
 (
 Name string,
 BoundaryShape binary
 )
ROW FORMAT SERDE 'com.esri.hadoop.hive.serde.EsriJsonSerDe'
STORED AS INPUTFORMAT 'com.esri.json.hadoop.EnclosedEsriJsonInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://DOC-EXAMPLE-BUCKET/my-query-log/json/';
```

Contoh kueri berikut menggunakan`CROSS JOIN`fungsi pada`counties`dan`earthquake`tabel. Contoh menggunakan`ST_CONTAINS`untuk kueri untuk kabupaten yang batas-batasnya termasuk lokasi gempa bumi, yang ditentukan dengan`ST_POINT`. Grup kueri kabupaten tersebut dengan nama, memerintahkannya dengan hitungan, dan mengembalikannya dalam urutan menurun.

> **Note**
>
> Mulai dari mesin Athena versi 2, fungsi seperti `ST_CONTAINS` tidak lagi mendukung `VARBINARY` tipe sebagai input. Untuk alasan ini, contoh menggunakan`ST_GeomFromLegacyBinary(varbinary)`fungsi untuk mengkonversi`boundaryshape`nilai biner menjadi geometri. Untuk informasi selengkapnya, lihat Perubahan fungsi geospasial di referensi Versi mesin Athena 2.

```
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
```

Kueri ini menghasilkan:

```
+------------------------+
| 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   |
+------------------------+
```

Sumber daya tambahan

Untuk contoh tambahan kueri geospasial, lihat postingan blog berikut:

- Perluas kueri geospasial di Amazon Athena dengan UDF dan AWS Lambda
- Visualisasikan lebih dari 200 tahun data iklim global menggunakan Amazon Athena dan Amazon. QuickSight
- Pertanyaan OpenStreetMap dengan Amazon Athena

Mengkueri JSON

Amazon Athena memungkinkan Anda menanyakan data yang disandikan JSON, mengekstrak data dari JSON bersarang, mencari nilai, dan menemukan panjang dan ukuran array JSON. Untuk mempelajari dasar-dasar kueri data JSON di Athena, pertimbangkan contoh data planet berikut:

```
{name:"Mercury",distanceFromSun:0.39,orbitalPeriod:0.24,dayLength:58.65}
{name:"Venus",distanceFromSun:0.72,orbitalPeriod:0.62,dayLength:243.02}
{name:"Earth",distanceFromSun:1.00,orbitalPeriod:1.00,dayLength:1.00}
{name:"Mars",distanceFromSun:1.52,orbitalPeriod:1.88,dayLength:1.03}
```

Perhatikan bagaimana setiap catatan (pada dasarnya, setiap baris dalam tabel) berada pada baris terpisah. Untuk menanyakan data JSON ini, Anda dapat menggunakan `CREATE TABLE` pernyataan seperti berikut:

```
CREATE EXTERNAL TABLE `planets_json`(
  `name` string,
  `distancefromsun` double,
  `orbitalperiod` double,
  `daylength` double)
ROW FORMAT SERDE
  'org.openx.data.jsonserde.JsonSerDe'
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/json/'
```

Untuk query data, gunakan `SELECT` pernyataan sederhana seperti contoh berikut.

```
SELECT * FROM planets_json
```

Hasil kueri terlihat seperti berikut ini.

#	name	jarakdarimatahari	periode orbital	panjang hari
1	Merkuri	0,39	0,24	58.65
2	Venus	0,72	0,62	243.02
3	Bumi	1.0	1.0	1.0
4	Mars	1,52	1,88	1.03

Perhatikan bagaimana `CREATE TABLE` pernyataan menggunakanOpenX JSON SerDe, yang mengharuskan setiap catatan JSON berada pada baris terpisah. Jika JSON dalam format cetak cantik, atau jika semua catatan berada pada satu baris, data tidak akan dibaca dengan benar.

Untuk kueri data JSON yang dalam format cetak cantik, Anda dapat menggunakan Sarang Ion Amazon SerDe bukan SerDe OpenX JSON. Pertimbangkan data sebelumnya yang disimpan dalam format cetak cantik:

```
{
  name:"Mercury",
  distanceFromSun:0.39,
  orbitalPeriod:0.24,
  dayLength:58.65
}
{
  name:"Venus",
  distanceFromSun:0.72,
  orbitalPeriod:0.62,
  dayLength:243.02
}
{
  name:"Earth",
  distanceFromSun:1.00,
  orbitalPeriod:1.00,
  dayLength:1.00
}
{
  name:"Mars",
  distanceFromSun:1.52,
  orbitalPeriod:1.88,
  dayLength:1.03
```

```
}
```

Untuk menanyakan data ini tanpa memformat ulang, Anda dapat menggunakan `CREATE TABLE` pernyataan seperti berikut ini. Perhatikan bahwa, alih-alih menentukan OpenX SerDe JSON, pernyataan menentukan. `STORED AS ION`

```
CREATE EXTERNAL TABLE `planets_ion`(
  `name` string,
  `distancefromsun` DECIMAL(10, 2),
  `orbitalperiod` DECIMAL(10, 2),
  `daylength` DECIMAL(10, 2))
STORED AS ION
LOCATION
  's3://DOC-EXAMPLE-BUCKET/json-ion/'
```

Kueri `SELECT * FROM planets_ion` menghasilkan hasil yang sama seperti sebelumnya. Untuk informasi selengkapnya tentang membuat tabel dengan cara ini menggunakan Amazon Ion Hive SerDe, lihatMenggunakan CREATE TABLE untuk membuat tabel Amazon Ion.

Contoh data JSON sebelumnya tidak berisi tipe data yang kompleks seperti array bersarang atau struct. Untuk informasi selengkapnya tentang menanyakan data JSON bersarang, lihat. Contoh: deserialisasi JSON bersarang

Topik

- Praktik terbaik untuk membaca data JSON
- Mengekstrak data JSON dari string
- Mencari nilai dalam array JSON
- Memperoleh panjang dan ukuran array JSON
- Memecahkan masalah kueri JSON

Praktik terbaik untuk membaca data JSON

JavaScript Object Notation (JSON) adalah metode umum untuk encoding struktur data sebagai teks. Banyak aplikasi dan alat output data yang dikodekan JSON-.

Di Amazon Athena, Anda dapat membuat tabel dari data eksternal dan menyertakan data dikodekan JSON di dalamnya. Untuk tipe data sumber seperti itu, gunakan Athena bersama Perpustakaan JSON SerDe .

Gunakan kiat berikut untuk membaca data yang dikodekan JSON:

- Pilih yang benar SerDe, JSON asli SerDe`org.apache.hive.hcatalog.data.JsonSerDe`, atau SerDe `org.openx.data.jsonserde.JsonSerDe` OpenX,. Untuk informasi selengkapnya, lihat Perpustakaan JSON SerDe .
- Pastikan bahwa setiap catatan yang disandikan JSON diwakili pada baris terpisah, tidak dicetak dengan cantik.

> **Note**
>
> SerDe Mengharapkan setiap dokumen JSON berada pada satu baris teks tanpa karakter penghentian baris yang memisahkan bidang dalam catatan. Jika teks JSON dalam format cetak cantik, Anda mungkin menerima pesan kesalahan seperti HIVE_CURSOR_ERROR: Row is not a valid JSON Object or HIVE_CURSOR_ERROR:: Unexpected JsonParseException end-of-input: expected close marker for OBJECT saat Anda mencoba menanyakan tabel setelah Anda membuatnya. Untuk informasi selengkapnya, lihat File Data JSON di dokumentasi SerDe OpenX. GitHub

- Hasilkan data yang dikodekan JSON Anda dalam kolom peka huruf.
- Beri opsi untuk mengabaikan catatan dengan bentuk yang salah, seperti dalam contoh ini.

```
CREATE EXTERNAL TABLE json_table (
  column_a string,
  column_b int
 )
 ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
 WITH SERDEPROPERTIES ('ignore.malformed.json' = 'true')
 LOCATION 's3://DOC-EXAMPLE-BUCKET/path/';
```

- Konversi bidang dalam data sumber yang memiliki skema yang belum ditentukan ke string yang dikodekan JSON di Athena.

Saat Athena membuat tabel yang didukung oleh data JSON, Athena akan menguraikan data berdasarkan skema yang ada dan yang ditentukan sebelumnya. Namun, tidak semua data Anda mungkin memiliki skema yang ditentukan sebelumnya. Untuk menyederhanakan manajemen skema dalam kasus tersebut, ini sering berguna untuk mengonversi bidang dalam sumber data yang memiliki skema yang belum ditentukan untuk string JSON di Athena, kemudian menggunakan Perpustakaan JSON SerDe .

Sebagai contoh, pertimbangkan sebuah aplikasi IOT yang mempublikasikan peristiwa dengan bidang umum dari sensor yang berbeda. Salah satu bidang tersebut harus menyimpan muatan kustom yang unik untuk sensor yang mengirim peristiwa. Dalam hal ini, karena Anda tidak tahu skema, kami sarankan Anda menyimpan informasi sebagai string dikodekan JSON. Untuk melakukan ini, konversi data dalam tabel Athena Anda ke JSON, seperti dalam contoh berikut. Anda juga dapat mengonversi data yang dikodekan JSON ke tipe data Athena.

- Mengonversi tipe data Athena ke JSON
- Mengonversi tipe data JSON ke Athena

Mengonversi tipe data Athena ke JSON

Untuk mengonversi tipe data Athena ke JSON, gunakan `CAST`.

```
WITH dataset AS (
  SELECT
    CAST('HELLO ATHENA' AS JSON) AS hello_msg,
    CAST(12345 AS JSON) AS some_int,
    CAST(MAP(ARRAY['a', 'b'], ARRAY[1,2]) AS JSON) AS some_map
)
SELECT * FROM dataset
```

Kueri ini menghasilkan:

```
+-----------------------------------------+
| hello_msg      | some_int | some_map      |
+-----------------------------------------+
| "HELLO ATHENA" | 12345    | {"a":1,"b":2} |
+-----------------------------------------+
```

Mengonversi tipe data JSON ke Athena

Untuk mengonversi tipe data JSON ke Athena, gunakan `CAST`.

> **Note**
>
> Dalam contoh ini, untuk menunjukkan string sebagai dikodekan JSON, mulai dengan `JSON` dan gunakan tanda kutip tunggal, seperti `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
```

Kueri ini menghasilkan:

```
+-------------------------------------+
| hello_msg    | some_int | some_map  |
+-------------------------------------+
| HELLO ATHENA | 12345    | {a:1,b:2} |
+-------------------------------------+
```

Mengekstrak data JSON dari string

Anda mungkin memiliki data sumber yang berisi string yang dikodekan JSON yang tidak perlu Anda deserialisasi ke dalam tabel di Athena. Dalam hal ini, Anda masih dapat menjalankan operasi SQL pada data ini, menggunakan fungsi JSON yang tersedia di Presto.

Pertimbangkan string JSON ini sebagai contoh set data.

```
{"name": "Susan Smith",
"org": "engineering",
"projects":
    [
     {"name":"project1", "completed":false},
     {"name":"project2", "completed":true}
    ]
}
```

Contoh: Mengekstrak properti

Untuk mengekstraksi properti `name` dan `projects` dari string JSON, gunakan fungsi `json_extract` seperti pada contoh berikut. Fungsi `json_extract` mengambil kolom yang berisi string JSON, dan mencarinya menggunakan eksprei seperti `JSONPath` dengan notasi titik`.`.

> **Note**
>
> `JSONPath` melakukan traversal pohon sederhana. Ini menggunakan tanda $ untuk menunjukkan root dari dokumen JSON, diikuti oleh titik dan elemen mest langsung di bawah root, seperti `$.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
```

Nilai yang dihasilkan adalah string dikodekan JSON, dan bukan tipe data Athena asli.

```
+-----------------------------------------------------------------------------------------------
+
| name           | projects
         |
+-----------------------------------------------------------------------------------------------
+
| "Susan Smith"  | [{"name":"project1","completed":false},
{"name":"project2","completed":true}] |
+-----------------------------------------------------------------------------------------------
+
```

Untuk mengekstraksi nilai skalar dari string JSON, gunakan fungsi `json_extract_scalar`. Ini mirip dengan `json_extract`, tetapi mengembalikan hanya nilai skalar (Boolean, angka, atau string).

> **Note**
>
> Jangan gunakan fungsi `json_extract_scalar` pada larik, peta, atau struct.

```
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
```

Kueri ini menghasilkan

```
+---------------------------+
| name           | projects |
+---------------------------+
| Susan Smith    |          |
+---------------------------+
```

Untuk mendapatkan elemen pertama dari properti `projects` dalam larik contoh, gunakan fungsi `json_array_get` dan tentukan posisi indeks.

```
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
```

Ini mengembalikan nilai pada posisi indeks yang ditentukan dalam larik dikodekan JSON.

```
+--------------------------------------+
| item                                 |
+--------------------------------------+
| {"name":"project1","completed":false} |
+--------------------------------------+
```

Untuk menghasilkan tipe string Athena, gunakan operator `[]` dalam ekspresi `JSONPath`, kemudian gunakan fungsi `json_extract_scalar`. Untuk informasi selengkapnya tentang `[]`, lihat Mengakses elemen array.

```
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
```

Ini mengembalikan hasil ini:

```
+--------------+
| project_name |
+--------------+
| project1     |
+--------------+
```

Mencari nilai dalam array JSON

Untuk menentukan apakah nilai tertentu ada dalam larik dikodekan JSON, gunakan fungsi `json_array_contains`.

Kueri berikut mencantumkan nama-nama pengguna yang berpartisipasi dalam “project2".

```
WITH dataset AS (
  SELECT * FROM (VALUES
    (JSON '{"name": "Bob Smith", "org": "legal", "projects": ["project1"]}'),
    (JSON '{"name": "Susan Smith", "org": "engineering", "projects": ["project1",
 "project2", "project3"]}'),
    (JSON '{"name": "Jane Smith", "org": "finance", "projects": ["project1",
 "project2"]}')
  ) AS t (users)
)
SELECT json_extract_scalar(users, '$.name') AS user
FROM dataset
WHERE json_array_contains(json_extract(users, '$.projects'), 'project2')
```

Kueri ini mengembalikan daftar pengguna.

```
+-------------+
| user        |
+-------------+
| Susan Smith |
+-------------+
| Jane Smith  |
+-------------+
```

Contoh kueri berikut mencantumkan nama pengguna yang telah menyelesaikan proyek bersama dengan jumlah total proyek yang telah selesai. Ini melakukan tindakan ini:

- Menggunakan pernyataan nested `SELECT` untuk kejelasan.
- Mengekstraksi larik proyek.
- Mengonversi larik ke larik asli pasangan nilai kunci menggunakan `CAST`.
- Mengekstraksi setiap elemen larik menggunakan operator `UNNEST`.
- Filter memperoleh nilai dengan menyelesaikan proyek dan menghitungnya.

> **Note**
>
> Saat menggunakan `CAST` untuk `MAP`, Anda dapat menentukan elemen kunci sebagai `VARCHAR` (String asli di Presto), tetapi meninggalkan nilai sebagai JSON, karena nilai-nilai dalam `MAP`adalah dari berbagai tipe: String untuk pasangan nilai kunci pertama, dan Boolean untuk kedua.

```
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
```

Kueri berikut mengembalikan hasil ini:

```
+----------------------------------+
| name        | completed_projects |
+----------------------------------+
| Susan Smith | 2                  |
+----------------------------------+
| Jane Smith  | 1                  |
+----------------------------------+
```

Memperoleh panjang dan ukuran array JSON

Contoh: **`json_array_length`**

Untuk mendapatkan panjang larik dikodekan JSON, gunakan fungsi `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
```

Kueri ini mengembalikan hasil ini:

```
+---------------------+
| name        | count |
+---------------------+
| Susan Smith | 2     |
+---------------------+
| Bob Smith   | 1     |
+---------------------+
| Jane Smith  | 1     |
+---------------------+
```

Contoh: **`json_size`**

Untuk mendapatkan ukuran larik atau objek dikodekan JSON, gunakan fungsi `json_size` dan tentukan kolom yang berisi string JSON dan ekspresi `JSONPath` untuk larik atau objek.

```
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
```

Kueri ini mengembalikan hasil ini:

```
+---------------------+
| name        | count |
+---------------------+
| Susan Smith | 2     |
+---------------------+
| Bob Smith   | 1     |
+---------------------+
| Jane Smith  | 1     |
+---------------------+
```

Memecahkan masalah kueri JSON

Untuk bantuan pemecahan masalah dengan kueri terkait JSON, lihat Kesalahan terkait JSON atau lihat sumber daya berikut:

- Saya mendapatkan kesalahan ketika saya mencoba membaca data JSON di Amazon Athena
- Bagaimana cara mengatasi “HIVE_CURSOR_ERROR: Baris bukan objek JSON yang valid - JSONException: Kunci duplikat” saat membaca file dari Athena? AWS Config
- Kueri SELECT COUNT di Amazon Athena hanya mengembalikan satu catatan meskipun file JSON input memiliki banyak catatan
- Bagaimana saya bisa melihat file sumber Amazon S3 untuk satu baris di tabel Athena?

Lihat juga Pertimbangan dan batasan untuk kueri SQL di Amazon Athena.

Menggunakan Machine Learning (ML) dengan Amazon Athena

Machine Learning (ML) dengan Amazon Athena memungkinkan Anda menggunakan Athena untuk menulis pernyataan SQL yang menjalankan inferensi Machine Learning (ML) menggunakan Amazon. SageMaker Fitur ini menyederhanakan akses ke model ML untuk analisis data, menghilangkan kebutuhan untuk menggunakan metode pemrograman yang kompleks untuk menjalankan inferensi.

Untuk menggunakan ML dengan Athena, Anda mendefinisikan sebuah ML dengan fungsi Athena dengan klausa `USING EXTERNAL FUNCTION`. Fungsi menunjuk ke titik akhir SageMaker model

yang ingin Anda gunakan dan menentukan nama variabel dan tipe data untuk diteruskan ke model. Klausa berikutnya dalam kueri mereferensikan fungsi untuk meneruskan nilai ke model. Model ini menjalankan inferensi berdasarkan nilai-nilai yang diteruskan oleh kueri, kemudian mengembalikan hasil inferensi. Untuk informasi selengkapnya tentang SageMaker dan cara kerja SageMaker titik akhir, lihat Panduan SageMaker Pengembang Amazon.

Untuk contoh yang menggunakan ML dengan Athena dan SageMaker inferensi untuk mendeteksi nilai anomali dalam kumpulan hasil, lihat artikel Blog AWS Big Data Mendeteksi nilai anomali dengan menjalankan fungsi inferensi pembelajaran mesin Amazon Athena.

Pertimbangan dan batasan

- Wilayah yang Tersedia - Fitur Athena MLadalah fitur di mana mesin Wilayah AWS Athena versi 2 atau yang lebih baru didukung.
- SageMaker titik akhir model harus menerima dan mengembalikan **`text/csv`** — Untuk informasi selengkapnya tentang format data, lihat Format data umum untuk inferensi di Panduan SageMaker Pengembang Amazon.
- Athena tidak mengirim header CSV - Jika SageMaker titik akhir Andatext/csv, penangan input Anda tidak boleh berasumsi bahwa baris pertama input adalah header CSV. Karena Athena tidak mengirim header CSV, output yang dikembalikan ke Athena akan berisi satu baris lebih sedikit dari yang diharapkan Athena dan menyebabkan kesalahan.
- SageMaker penskalaan titik akhir — Pastikan titik akhir SageMaker model yang direferensikan cukup ditingkatkan untuk panggilan Athena ke titik akhir. Untuk informasi selengkapnya, lihat Menskalakan SageMaker model secara otomatis di Panduan SageMaker Pengembang Amazon dan CreateEndpointConfigdi Referensi SageMaker API Amazon.
- Izin IAM — Untuk menjalankan kueri yang menentukan fungsi HTML dengan Athena, prinsipal IAM yang menjalankan kueri harus diizinkan untuk melakukan `sagemaker:InvokeEndpoint` tindakan untuk titik akhir model yang direferensikan. SageMaker Untuk informasi selengkapnya, lihat Mengizinkan akses untuk ML dengan Athena.
- ML dengan fungsi Athena tidak dapat digunakan dalam klausa secara langsung **`GROUP BY`**

ML dengan sintaks Athena

Klausa `USING EXTERNAL FUNCTION` menentukan ML dengan fungsi Athena atau beberapa fungsi yang dapat dijadikan referensi oleh pernyataan `SELECT` berikutnya dalam kueri. Anda menentukan nama fungsi, nama variabel, dan tipe data untuk variabel dan nilai kembali.

Sinopsis

Sintaks berikut menunjukkan klausa `USING EXTERNAL FUNCTION` yang menentukan sebuah ML dengan fungsi Athena.

```
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 mendefinisikan nama fungsi, yang dapat digunakan dalam klausa kueri berikutnya. Setiap *variabel data_type* menentukan variabel bernama dan tipe data yang sesuai yang diterima SageMaker model sebagai input. Tipe data yang ditentukan harus berupa tipe data Athena yang didukung.

RETURNS ***data_type***

data_type menentukan tipe data SQL yang *ml_function_name* kembali ke query sebagai output dari model. SageMaker

SAGEMAKER '***sagemaker_endpoint***'

sagemaker_endpoint menentukan titik akhir model. SageMaker

SELECT [...] ***ml_function_name***(***expression***) [...]

Query SELECT yang meneruskan nilai ke variabel fungsi dan SageMaker model untuk mengembalikan hasil. *ml_function_name* menentukan fungsi didefinisikan sebelumnya dalam query, diikuti oleh *ekspresi* yang dievaluasi untuk lulus nilai. Nilai-nilai yang diteruskan dan dihasilkan harus cocok dengan tipe data yang sesuai yang ditentukan untuk fungsi dalam klausa `USING EXTERNAL FUNCTION`.

Contoh

Contoh berikut menunjukkan kueri menggunakan ML dengan 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;
```

Contoh penggunaan pelanggan

Video berikut, yang menggunakan versi Pratinjau Machine Learning (ML) dengan Amazon Athena, menampilkan cara-cara yang dapat Anda gunakan dengan SageMaker Athena.

Memprediksi churn pelanggan

Video berikut menunjukkan cara menggabungkan Athena dengan kemampuan pembelajaran mesin Amazon SageMaker untuk memprediksi churn pelanggan.

Memprediksi churn pelanggan menggunakan Amazon Athena dan Amazon SageMaker

Mendeteksi botnet

Video berikut menunjukkan bagaimana satu perusahaan menggunakan Amazon Athena dan Amazon SageMaker untuk mendeteksi botnet.

Mendeteksi botnet menggunakan Amazon Athena dan Amazon SageMaker

Query dengan fungsi yang ditentukan pengguna

User Defined Functions (UDF) di Amazon Athena memungkinkan Anda untuk membuat fungsi kustom untuk memproses catatan atau kelompok catatan. Sebuah UDF menerima parameter, melakukan tugas, kemudian mengembalikan hasilnya.

Untuk menggunakan UDF di Athena, Anda menulis klausa `USING EXTERNAL FUNCTION` sebelum pernyataan `SELECT` dalam kueri SQL. Pernyataan `SELECT` mereferensikan UDF dan mendefinisikan variabel yang diteruskan ke UDF saat kueri berjalan. Kueri SQL memanggil fungsi Lambda menggunakan runtime Java saat memanggil UDF. UDFS didefinisikan dalam fungsi Lambda sebagai metode dalam paket deployment Java. Beberapa UDFS dapat didefinisikan dalam paket deployment Java yang sama untuk fungsi Lambda. Anda juga menentukan nama fungsi Lambda di klausa `USING EXTERNAL FUNCTION`.

Anda memiliki dua pilihan untuk men-deploy fungsi Lambda untuk Athena UDFS. Anda dapat men-deploy fungsi langsung menggunakan Lambda, atau Anda dapat menggunakan AWS Serverless Application Repository. Untuk menemukan fungsi Lambda yang ada untuk UDF, Anda dapat mencari repositori publik AWS Serverless Application Repository atau pribadi Anda dan kemudian menyebarkan ke Lambda. Anda juga dapat membuat atau memodifikasi kode sumber Java, mengemasnya ke dalam file JAR, dan men-deploy menggunakan Lambda atau AWS Serverless Application Repository. Misalnya kode sumber Java dan paket untuk memulai, lihat Membuat dan menerapkan UDF menggunakan Lambda. Untuk informasi selengkapnya tentang Lambda, lihat AWS Lambda Panduan Developer. Untuk informasi selengkapnya AWS Serverless Application Repository, lihat Panduan AWS Serverless Application Repository Pengembang.

Untuk contoh yang menggunakan UDF dengan Athena untuk menerjemahkan dan menganalisis teks, lihat artikel Machine AWS Learning Blog Terjemahkan dan analisis teks menggunakan fungsi SQL dengan Amazon Athena, Amazon Translate, dan Amazon Comprehend, atau tonton. video

Untuk contoh menggunakan UDF untuk memperluas kueri geospasial di Amazon Athena, lihat Memperluas kueri geospasial di Amazon Athena dengan UDF dan di Blog Big Data. AWS LambdaAWS

Pertimbangan dan batasan

- Fungsi Athena bawaan - Fungsi bawaan di Athena dirancang agar berkinerja tinggi. Kami merekomendasikan bahwa Anda menggunakan bawaan fungsi atas UDFS jika memungkinkan. Untuk informasi selengkapnya tentang fungsi bawaan, lihat Fungsi di Amazon Athena.
- UDF skalar saja– Athena hanya mendukung UDFS skalar, yang memproses satu baris pada satu waktu dan mengembalikan nilai kolom tunggal. Athena melewati batch baris, berpotensi secara paralel, untuk UDF setiap kali memanggil Lambda. Saat merancang UDFS dan kueri, berhati-hati dari potensi dampak terhadap lalu lintas jaringan pemrosesan ini.
- Fungsi handler UDF menggunakan format yang disingkat - Gunakan format singkat (bukan format penuh), untuk fungsi UDF Anda (misalnya, bukan). `package.Class` `package.Class::method`
- Metode UDF harus huruf kecil — metode UDF harus dalam huruf kecil; kasus unta tidak diizinkan.
- Metode UDF memerlukan parameter — Metode UDF harus memiliki setidaknya satu parameter input. Mencoba memanggil UDF yang ditentukan tanpa parameter input menyebabkan pengecualian runtime. UDF dimaksudkan untuk melakukan fungsi terhadap catatan data, tetapi UDF tanpa argumen tidak mengambil data, sehingga pengecualian terjadi.

- Dukungan runtime Java– Saat ini, Athena UDFS mendukung runtime Java 8 dan Java 11 untuk Lambda. Untuk informasi selengkapnya, lihat Membangun fungsi Lambda dengan Java dalam AWS Lambda Panduan Developer.
- Izin IAM– Untuk menjalankan dan membuat pernyataan permintaan UDF di Athena, IAM utama menjalankan kueri harus diizinkan untuk melakukan tindakan selain fungsi Athena. Untuk informasi selengkapnya, lihat Contoh kebijakan izin IAM untuk mengizinkan Amazon Athena User Defined Functions (UDF).
- Kuota Lambda– Kuota Lambda berlaku untuk UDFS. Untuk informasi selengkapnya, lihat Kuota Lambda dalam Panduan Pengembang AWS Lambda .
- Pemfilteran tingkat baris — Pemfilteran tingkat baris Lake Formation tidak didukung untuk UDF.
- Tampilan — Anda tidak dapat menggunakan tampilan dengan UDF.
- Masalah yang diketahui — Untuk sebagian besar up-to-date daftar masalah yang diketahui, lihat Batasan dan masalah di bagian awslabs/aws-athena-query-federation. GitHub

Video

Tonton video berikut untuk mempelajari selengkapnya tentang penggunaan UDFS di Athena.

Video: Memperkenalkan Fungsi Ditetapkan Pengguna (UDFs) di Amazon Athena

Video berikut menunjukkan bagaimana Anda dapat menggunakan UDFS di Amazon Athena untuk menyunting informasi sensitif.

> **Note**
>
> Sintaks dalam video ini adalah pra-rilis, tetapi konsepnya sama. Gunakan Athena tanpa kelompok kerja. `AmazonAthenaPreviewFunctionality`

Memperkenalkan fungsi yang ditentukan pengguna (UDF) di Amazon Athena

Video: Translate, menganalisis, dan redact bidang teks menggunakan kueri SQL di Amazon Athena

Video berikut menunjukkan bagaimana Anda dapat menggunakan UDF di Amazon Athena bersama dengan yang Layanan AWS lain untuk menerjemahkan dan menganalisis teks.

> **Note**
>
> Sintaks dalam video ini adalah pra-rilis, tetapi konsepnya sama. Untuk sintaks yang benar, lihat postingan blog terkait Menerjemahkan, menyunting, dan menganalisis teks menggunakan fungsi SQL dengan Amazon Athena, Amazon Translate, dan Amazon Comprehend pada AWS Blog Machine Learning.

Terjemahkan, analisis, dan edit bidang teks menggunakan kueri SQL di Amazon Athena

Sintaks kueri UDF

Klausa `USING EXTERNAL FUNCTION` menentukan UDF atau beberapa UDF yang dapat direferensikan oleh pernyataan `SELECT` dalam kueri. Anda perlu nama metode untuk UDF dan nama fungsi Lambda yang host UDF. Di tempat nama fungsi Lambda, Anda dapat menggunakan Lambda ARN. Dalam skenario lintas akun, Lambda ARN diperlukan.

Sinopsis

```
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

MENGGUNAKAN FUNGSI EKSTERNAL***UDF_NAME***(***variabel1 tipe_data***[,***variabel2 tipe_data***] [,...])

*UDF_NAME*menentukan nama UDF, yang harus sesuai dengan metode Java dalam fungsi Lambda direferensikan. Setiap*tipe_data variabel*menentukan variabel bernama dan tipe data yang sesuai bahwa UDF menerima sebagai masukan. *data_type* harus merupakan salah satu tipe data Athena didukung tercantum yang dalam tabel berikut dan dipetakan uke tipe data Java yang sesuai.

Tipe data Athena	Tipe data Java
TIMESTAMP	java.waktu. LocalDateTime (UTC)
DATE	java.waktu. LocalDate (UTC)
TINYINT	java.lang.Byte
SMALLINT	java.lang.Short
REAL	java.lang.Float
DOUBLE	java.lang.Double
DECIMAL (lihatRETURNSCat	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
BARIS	java.util.map<String, Object>

PENGEMBALIAN*tipe_data*

`data_type`menentukan tipe data SQL bahwa UDF kembali sebagai output. Tipe data Athena yang tercantum dalam tabel di atas didukung. Untuk`DECIMAL`tipe data, gunakan sintaks`RETURNS DECIMAL(`*`precision`*`, `*`scale`*`)`Di mana*presisi*dan*skala*Bilangan bulat.

LAMBDA '*lambda_function*'

*lambda_function*menentukan nama fungsi Lambda yang akan dipanggil saat menjalankan UDF.

PILIH [...]***UDF_NAME***(***expression***) [...]

Kueri `SELECT` yang meneruskan nilai ke UDF dan mengembalikan hasil. *UDF_NAME* menentukan UDF yang akan digunakan, diikuti dengan *ekspresi* yang dievaluasi untuk meneruskan nilai. Nilai-nilai yang dilewatkan dan dikembalikan harus sesuai dengan tipe data yang sesuai ditentukan untuk UDF di`USING EXTERNAL FUNCTION`Klausul.

Contoh

Misalnya kueri berdasarkan kode AthenaUDFHandler.java GitHub, lihat halaman konektor UDF GitHub Amazon Athena.

Membuat dan menerapkan UDF menggunakan Lambda

Untuk membuat UDF kustom, Anda membuat kelas Java baru dengan memperluas`UserDefinedFunctionHandler`kelas. Kode sumber untuk UserDefinedFunctionHandler.java di SDK tersedia GitHub di awslabs/aws-athena-query-federation/athena-federation-sdk repositori, bersama dengan contoh implementasi UDF yang dapat Anda periksa dan modifikasi untuk membuat UDF khusus.

Langkah-langkah dalam bagian ini menunjukkan penulisan dan membangun file UDF Jar kustom menggunakanApache Mavendari baris perintah dan menerapkan.

Langkah-langkah untuk membuat UDF Custom untuk Athena menggunakan Maven

- Kloning SDK dan siapkan lingkungan pengembangan Anda
- Buat proyek Maven Anda
- Tambahkan dependensi dan plugin ke proyek Maven Anda
- Menulis kode Java untuk UDF
- Membangun file JAR
- Menyebarkan JAR ke AWS Lambda

Kloning SDK dan siapkan lingkungan pengembangan Anda

Sebelum memulai, pastikan bahwa git diinstal pada sistem Anda menggunakan`sudo yum install git -y`.

Untuk menginstal SDK federasi AWS kueri

- Masukkan yang berikut pada baris perintah untuk mengkloning repositori SDK. Repositori ini mencakup SDK, contoh dan rangkaian konektor sumber data. Untuk informasi selengkapnya tentang konektor sumber data, lihat Menggunakan Amazon Athena.

```
git clone https://github.com/awslabs/aws-athena-query-federation.git
```

Untuk menginstal prasyarat untuk prosedur ini

Jika Anda sedang mengerjakan mesin pengembangan yang sudah memiliki Apache Maven, the AWS CLI, dan alat AWS Serverless Application Model build yang diinstal, Anda dapat melewati langkah ini.

1. Dari akar`aws-athena-query-federation`direktori yang Anda buat saat Anda kloning, jalankanprepare_dev_env.shyang mempersiapkan lingkungan pengembangan Anda.
2. Perbarui shell Anda untuk sumber variabel baru yang dibuat oleh proses instalasi atau restart sesi terminal Anda.

```
source ~/.profile
```

> **Important**
>
> Jika Anda melewati langkah ini, Anda akan mendapatkan kesalahan nanti tentang AWS CLI atau AWS SAM membangun alat yang tidak dapat mempublikasikan fungsi Lambda Anda.

Buat proyek Maven Anda

Jalankan perintah berikut untuk membuat proyek Maven Anda. Ganti *`GroupId`* dengan ID unik organisasi Anda, dan ganti *`my-athena-udf`*dengan nama aplikasi Anda Untuk informasi selengkapnya, lihat Bagaimana cara membuat proyek Maven pertama saya? dalam dokumentasi Apache Maven.

```
mvn -B archetype:generate \
-DarchetypeGroupId=org.apache.maven.archetypes \
-DgroupId=groupId \
-DartifactId=my-athena-udfs
```

Tambahkan dependensi dan plugin ke proyek Maven Anda

Tambahkan konfigurasi berikut ke file proyek Maven `pom.xml` Anda. Sebagai contoh, lihat file pom.xml di GitHub.

```
<properties>
    <aws-athena-federation-sdk.version>2022.47.1</aws-athena-federation-sdk.version>
</properties>

<dependencies>
    <dependency>
        <groupId>com.amazonaws</groupId>
        <artifactId>aws-athena-federation-sdk</artifactId>
        <version>${aws-athena-federation-sdk.version}</version>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-shade-plugin</artifactId>
            <version>3.2.1</version>
            <configuration>
                <createDependencyReducedPom>false</createDependencyReducedPom>
                <filters>
                    <filter>
                        <artifact>*:*</artifact>
                        <excludes>
                            <exclude>META-INF/*.SF</exclude>
                            <exclude>META-INF/*.DSA</exclude>
                            <exclude>META-INF/*.RSA</exclude>
                        </excludes>
                    </filter>
                </filters>
            </configuration>
            <executions>
                <execution>
```

```
                    <phase>package</phase>
                    <goals>
                        <goal>shade</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

Menulis kode Java untuk UDF

Buat kelas baru dengan memperluas UserDefinedFunctionHandler.java. Tulis UDFS Anda di dalam kelas.

Pada contoh berikut, dua metode Java untuk UDFS,`compress()`dan`decompress()`, dibuat di dalam kelas`MyUserDefinedFunctions`.

```
*package *com.mycompany.athena.udfs;

public class MyUserDefinedFunctions
        extends UserDefinedFunctionHandler
{
    private static final String SOURCE_TYPE = "MyCompany";

    public MyUserDefinedFunctions()
    {
        super(SOURCE_TYPE);
    }

    /**
     * Compresses a valid UTF-8 String using the zlib compression library.
     * Encodes bytes with Base64 encoding scheme.
     *
     * @param input the String to be compressed
     * @return the compressed String
     */
    public String compress(String input)
    {
        byte[] inputBytes = input.getBytes(StandardCharsets.UTF_8);

        // create compressor
        Deflater compressor = new Deflater();
```

```
        compressor.setInput(inputBytes);
        compressor.finish();

        // compress bytes to output stream
        byte[] buffer = new byte[4096];
        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(inputBytes.length);
        while (!compressor.finished()) {
            int bytes = compressor.deflate(buffer);
            byteArrayOutputStream.write(buffer, 0, bytes);
        }

        try {
            byteArrayOutputStream.close();
        }
        catch (IOException e) {
            throw new RuntimeException("Failed to close ByteArrayOutputStream", e);
        }

        // return encoded string
        byte[] compressedBytes = byteArrayOutputStream.toByteArray();
        return Base64.getEncoder().encodeToString(compressedBytes);
    }

    /**
     * Decompresses a valid String that has been compressed using the zlib compression
library.
     * Decodes bytes with Base64 decoding scheme.
     *
     * @param input the String to be decompressed
     * @return the decompressed String
     */
    public String decompress(String input)
    {
        byte[] inputBytes = Base64.getDecoder().decode((input));

        // create decompressor
        Inflater decompressor = new Inflater();
        decompressor.setInput(inputBytes, 0, inputBytes.length);

        // decompress bytes to output stream
        byte[] buffer = new byte[4096];
        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(inputBytes.length);
```

```
        try {
            while (!decompressor.finished()) {
                int bytes = decompressor.inflate(buffer);
                if (bytes == 0 && decompressor.needsInput()) {
                    throw new DataFormatException("Input is truncated");
                }
                byteArrayOutputStream.write(buffer, 0, bytes);
            }
        }
        catch (DataFormatException e) {
            throw new RuntimeException("Failed to decompress string", e);
        }

        try {
            byteArrayOutputStream.close();
        }
        catch (IOException e) {
            throw new RuntimeException("Failed to close ByteArrayOutputStream", e);
        }

        // return decoded string
        byte[] decompressedBytes = byteArrayOutputStream.toByteArray();
        return new String(decompressedBytes, StandardCharsets.UTF_8);
    }
}
```

Membangun file JAR

Jalankan`mvn clean install`untuk membangun proyek Anda. Setelah berhasil membangun, file JAR dibuat di`target`dari proyek Anda bernama*`artifactId`*`-`*`version`*`.jar`, tempat*`artifactId`*adalah nama yang Anda berikan dalam proyek Maven, misalnya,`my-athena-udfs`.

Menyebarkan JAR ke AWS Lambda

Anda memiliki dua pilihan untuk men-deploy kode Anda untuk Lambda:

- Menyebarkan Menggunakan AWS Serverless Application Repository (Disarankan)
- Buat Fungsi Lambda dari file JAR

Opsi 1: Menyebarkan ke AWS Serverless Application Repository

Ketika Anda menyebarkan file JAR Anda ke AWS Serverless Application Repository, Anda membuat file YAMM AWS SAM template yang mewakili arsitektur aplikasi Anda. Anda kemudian menentukan file YAML ini dan bucket Amazon S3 tempat artifact untuk aplikasi Anda di-upload dan dibuat tersedia untuk AWS Serverless Application Repository. Prosedur di bawah ini menggunakanpublish.shskrip yang terletak di`athena-query-federation/tools`dari Athena Kueri Federation SDK yang Anda kloning sebelumnya.

Untuk informasi dan persyaratan selengkapnya, lihat Menerbitkan aplikasi di Panduan AWS Serverless Application Repository Pengembang, konsep AWS SAM templat di Panduan AWS Serverless Application Model Pengembang, dan Menerbitkan aplikasi tanpa server menggunakan CLI AWS SAM.

Contoh berikut menunjukkan parameter dalam file YAML. Tambahkan parameter serupa ke file YAML dan simpan di direktori proyek Anda. Lihat athena-udf.yaml untuk contoh lengkapnya. 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
```

Salinpublish.shscript ke direktori proyek tempat Anda menyimpan file YAML Anda, dan jalankan perintah berikut:

```
./publish.sh MyS3Location MyYamlFile
```

Misalnya, jika lokasi bucket Andas3://DOC-EXAMPLE-BUCKET/mysarapps/athenaudfdan file YAML Anda disimpan sebagaimy-athena-udfs.yaml:

```
./publish.sh DOC-EXAMPLE-BUCKET/mysarapps/athenaudf my-athena-udfs
```

Untuk membuat fungsi Lambda

1. Buka konsol Lambda dihttps://console.aws.amazon.com/lambda/, pilihBuat fungsi, lalu pilihJelajahi repositori aplikasi tanpa server
2. PilihAplikasi privat, temukan aplikasi Anda dalam daftar, atau cari menggunakan kata-kata kunci, dan pilih.
3. Tinjau dan berikan detail aplikasi, lalu pilihTerapkan.

   Anda sekarang dapat menggunakan nama metode yang didefinisikan dalam file JAR fungsi Lambda Anda sebagai UDFS di Athena.

Opsi 2: Membuat fungsi Lambda secara langsung

Anda juga dapat membuat fungsi Lambda secara langsung menggunakan konsol atau. AWS CLI Contoh berikut menunjukkan menggunakan Lambdacreate-functionPerintah CLI.

```
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
```

Menanyakan lintas wilayah

Athena mendukung kemampuan untuk menanyakan data Amazon S3 Wilayah AWS yang berbeda dari Wilayah tempat Anda menggunakan Athena. Kueri di seluruh Wilayah dapat menjadi pilihan saat memindahkan data tidak praktis atau diizinkan, atau jika Anda ingin melakukan kueri data di beberapa wilayah. Bahkan jika Athena tidak tersedia di Wilayah tertentu, data dari Wilayah tersebut dapat ditanyakan dari Wilayah lain di mana Athena tersedia.

Untuk menanyakan data di Wilayah, akun Anda harus diaktifkan di Wilayah tersebut meskipun data Amazon S3 bukan milik akun Anda. Untuk beberapa wilayah seperti US East (Ohio), akses Anda ke Wilayah secara otomatis diaktifkan saat akun Anda dibuat. Wilayah lain mengharuskan akun Anda “ikut serta” ke Wilayah sebelum Anda dapat menggunakannya. Untuk daftar Wilayah yang memerlukan keikutsertaan, lihat Wilayah yang tersedia di Panduan Pengguna Amazon EC2. Untuk petunjuk spesifik tentang memilih masuk ke Wilayah, lihat Mengelola AWS wilayah di. Referensi Umum Amazon Web Services

Pertimbangan dan batasan

- Izin akses data — Agar berhasil menanyakan data Amazon S3 dari Athena di seluruh Wilayah, akun Anda harus memiliki izin untuk membaca data. Jika data yang ingin Anda kueri milik akun lain, akun lain harus memberi Anda akses ke lokasi Amazon S3 yang berisi data.
- Biaya transfer data — Biaya transfer data Amazon S3 berlaku untuk kueri lintas wilayah. Menjalankan kueri dapat menghasilkan lebih banyak data yang ditransfer daripada ukuran kumpulan data. Kami menyarankan Anda memulai dengan menguji kueri Anda pada subset data dan meninjau biayanya. AWS Cost Explorer
- AWS Glue— Anda dapat menggunakan AWS Glue di seluruh Wilayah. Biaya tambahan mungkin berlaku untuk lalu lintas AWS Glue lintas wilayah. Untuk informasi selengkapnya, lihat Membuat AWS Glue koneksi lintas akun dan lintas wilayah di Blog AWS Big Data.
- Opsi enkripsi Amazon S3 — Opsi enkripsi SSE-S3 dan SSE-KMS didukung untuk kueri di seluruh Wilayah; CSE-KMS tidak. Untuk informasi selengkapnya, lihat Opsi enkripsi Amazon S3 yang didukung.
- Kueri federasi — Menggunakan kueri federasi di seluruh Wilayah AWS tidak didukung.

- Wilayah China - Pertanyaan lintas wilayah tidak didukung di Wilayah China.

Asalkan kondisi di atas terpenuhi, Anda dapat membuat tabel Athena yang menunjuk ke `LOCATION` nilai yang Anda tentukan dan kueri data secara transparan. Tidak diperlukan sintaks khusus. Untuk informasi tentang membuat tabel Athena, lihat. Membuat tabel di Athena

Mengkueri AWS Glue Data Catalog

Karena AWS Glue Data Catalog digunakan oleh banyak orang Layanan AWS sebagai repositori metadata pusat mereka, Anda mungkin ingin menanyakan metadata Katalog Data. Untuk melakukannya, Anda dapat menggunakan kueri SQL di Athena. Anda dapat menggunakan Athena untuk mengkueri katalog metadata AWS Glue seperti basis data, tabel, partisi, dan kolom.

Untuk mendapatkan metadata Katalog AWS Glue, Anda mengkueri basis data `information_schema` pada backend Athena. Contoh kueri dalam topik ini menunjukkan bagaimana menggunakan Athena untuk mengkueri metadata Katalog AWS Glue untuk kasus penggunaan umum.

Topik

Pertimbangan dan batasan

- Alih-alih menanyakan `information_schema` database, dimungkinkan untuk menggunakan perintah Apache Hive DDL individu untuk mengekstrak informasi metadata untuk database, tabel, tampilan, partisi, dan kolom tertentu dari Athena. Namun, outputnya dalam format non-tabular.
- Kueri paling `information_schema` berkinerja jika Anda memiliki jumlah metadata kecil hingga sedang. AWS Glue Jika Anda memiliki sejumlah besar metadata, kesalahan dapat terjadi.
- Anda tidak dapat menggunakan `CREATE VIEW`untuk membuat tampilan pada basis data `information_schema`.

Daftar database dan mencari database tertentu

Contoh dalam bagian ini menunjukkan cara mencantumkan basis data dalam metadata berdasarkan nama skema.

Example — Daftar database

Contoh kueri berikut mencantumkan basis data dari tabel `information_schema.schemata`.

```
SELECT schema_name
FROM   information_schema.schemata
LIMIT  10;
```

Tabel berikut menunjukkan hasil sampel.

6	alb-databas1
7	alb_original_cust
8	alblogsdatabase
9	athena_db_test
10	athena_ddl_db

Example — Mencari database tertentu

Pada kueri contoh berikut, `rdspostgresql` adalah basis data sampel.

```
SELECT schema_name
FROM   information_schema.schemata
WHERE  schema_name = 'rdspostgresql'
```

Tabel berikut menunjukkan hasil sampel.

	schema_name
1	rdspostgresql

Daftar tabel dalam database tertentu dan mencari tabel dengan nama

Untuk mencantumkan metadata untuk tabel, Anda dapat mengkueri menurut skema tabel atau menurut nama tabel.

Example — Daftar tabel berdasarkan skema

Kueri berikut mencantumkan tabel yang menggunakan skema tabel `rdspostgresql`.

```
SELECT table_schema,
       table_name,
       table_type
FROM   information_schema.tables
WHERE  table_schema = 'rdspostgresql'
```

Tabel berikut menunjukkan hasil sampel.

	table_schema	table_name	table_type
1	rdspostgresql	rdspostgresqldb1_public_account	TABEL DASAR

Example — Mencari tabel dengan nama

Kueri berikut memperoleh informasi metadata untuk tabel`athena1`.

```
SELECT table_schema,
       table_name,
       table_type
FROM   information_schema.tables
WHERE  table_name = 'athena1'
```

Tabel berikut menunjukkan hasil sampel.

	table_schema	table_name	table_type
1	default	athena1	TABEL DASAR

Daftar partisi untuk tabel tertentu

Anda dapat menggunakan `SHOW PARTITIONS` *`table_name`* untuk mencantumkan partisi untuk tabel tertentu, seperti dalam contoh berikut.

```
SHOW PARTITIONS cloudtrail_logs_test2
```

Anda juga dapat menggunakan kueri `$partitions` metadata untuk mencantumkan nomor partisi dan nilai partisi untuk tabel tertentu.

Example — Menanyakan partisi untuk tabel menggunakan sintaks $partisi

Contoh query berikut mencantumkan partisi untuk tabel `cloudtrail_logs_test2` menggunakan `$partitions` sintaks.

```
SELECT * FROM default."cloudtrail_logs_test2$partitions" ORDER BY partition_number
```

Tabel berikut menunjukkan hasil sampel.

	table_catalog	table_schema	table_name	Tahun	Bulan	Hari
1	awsdatacatalog	default	cloudtrail_logs_test2	2020	08	10
2	awsdatacatalog	default	cloudtrail_logs_test2	2020	08	11
3	awsdatacatalog	default	cloudtrail_logs_test2	2020	08	12

Daftar semua kolom untuk semua tabel

Anda dapat mencantumkan semua kolom untuk semua tabel di `AwsDataCatalog` atau untuk semua tabel dalam database tertentu di`AwsDataCatalog`.

- Untuk mencantumkan semua kolom untuk semua database`AwsDataCatalog`, gunakan kueri`SELECT * FROM information_schema.columns`.

- Untuk membatasi hasil ke database tertentu, gunakan `table_schema='database_name'` dalam `WHERE` klausa.

Example - Daftar semua kolom untuk semua tabel dalam database tertentu

Contoh query berikut mencantumkan semua kolom untuk semua tabel dalam database`webdata`.

```
SELECT * FROM information_schema.columns WHERE table_schema = 'webdata'
```

Daftar kolom yang memiliki kesamaan tabel tertentu

Anda dapat membuat daftar kolom yang memiliki kesamaan tabel tertentu dalam database.

- Gunakan sintaks`SELECT column_name FROM information_schema.columns`.
- Untuk `WHERE` klausa, gunakan sintaks`WHERE table_name IN ('table1', 'table2')`.

Example — Daftar kolom umum untuk dua tabel dalam database yang sama

Contoh query berikut mencantumkan kolom yang tabel `table1` dan `table2` memiliki kesamaan.

```
SELECT column_name
FROM information_schema.columns
WHERE table_name IN ('table1', 'table2')
GROUP BY column_name
HAVING COUNT(*) > 1;
```

Daftar atau mencari kolom untuk tabel atau tampilan tertentu

Anda dapat mencantumkan semua kolom untuk tabel, semua kolom untuk tampilan, atau mencari kolom dengan nama dalam basis data dan tabel tertentu.

Untuk daftar kolom, gunakan kueri `SELECT *`. Di klausa `FROM`, tentukan `information_schema.columns`. Di klausa `WHERE`, gunakan `table_schema='database_name'` untuk menentukan basis data dan`table_name = 'table_name'` untuk menentukan tabel atau tampilan yang memiliki kolom yang ingin Anda cantumkan.

Example - Daftar semua kolom untuk tabel tertentu

Contoh kueri berikut mencantumkan semua kolom untuk tabel `rdspostgresqldb1_public_account`.

```
SELECT *
FROM   information_schema.columns
WHERE  table_schema = 'rdspostgresql'
       AND table_name = 'rdspostgresqldb1_public_account'
```

Tabel berikut menunjukkan hasil sampel.

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_t	kome	extra_info
1	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	kata sandi	1		Ya	varcha		
2	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	user_id	2		YA	bilang bulat		
3	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	created_on	3		YA	stempwaktu		
4	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	last_login	4		YA	stempwaktu		
5	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	Email	5		Ya	varcha		

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_t	kome	extra_info
6	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	nama penggun	6		Ya	varcha		

Example - Daftar kolom untuk tampilan tertentu

Contoh kueri berikut mencantumkan semua kolom di basis data `default` untuk tampilan `arrayview`.

```
SELECT *
FROM   information_schema.columns
WHERE  table_schema = 'default'
       AND table_name = 'arrayview'
```

Tabel berikut menunjukkan hasil sampel.

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_typ	kome	extra_info
1	awsdataca talog	default	arrayvie	searchdate	1		Ya	varchar		
2	awsdataca talog	default	arrayvie	sid	2		Ya	varchar		
3	awsdataca talog	default	arrayvie	btid	3		Ya	varchar		
4	awsdataca talog	default	arrayvie	p	4		Ya	varchar		
5	awsdataca talog	default	arrayvie	infantprice	5		Ya	varchar		

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_type	kome	extra_info
6	awsdatacatalog	default	arrayvie	bah	6		Ya	varchar		
7	awsdatacatalog	default	arrayvie	journeymaparray	7		Ya	array (varchar		

Example — Mencari kolom dengan nama dalam database dan tabel tertentu

Contoh berikut mengkueri pencarian untuk metadata kolom `sid` dalam tampilan `arrayview` basis data `default`.

```
SELECT *
FROM   information_schema.columns
WHERE  table_schema = 'default'
       AND table_name = 'arrayview'
       AND column_name='sid'
```

Tabel berikut menunjukkan hasil sampel.

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_t	kome	extra_info
1	awsdatacatalog	default	arrayview	sid	2		Ya	varcha		

Meminta log Layanan AWS

Bagian ini mencakup beberapa prosedur untuk menggunakan Amazon Athena untuk menanyakan kumpulan data populer, seperti log, log Amazon AWS CloudTrail , log Classic Load Balancer, CloudFront log Application Load Balancer, log aliran Amazon VPC, dan log Network Load Balancer.

Tugas di bagian ini menggunakan konsol Athena, tetapi Anda juga dapat menggunakan alat lain seperti driver Athena JDBC, AWS CLIthe, atau Referensi API Amazon Athena.

Untuk informasi tentang penggunaan AWS CloudFormation untuk secara otomatis membuat tabel Layanan AWS log, partisi, dan contoh kueri di Athena, lihat Mengotomatiskan pembuatan tabel Layanan AWS log dan menanyakannya dengan Amazon Athena di Blog Big Data. AWS Untuk informasi tentang penggunaan pustaka Python untuk AWS Glue membuat kerangka kerja umum untuk memproses Layanan AWS log dan menanyakannya di Athena, lihat Kueri log dengan mudah menggunakan Layanan AWS Amazon Athena.

Topik di bagian ini mengasumsikan bahwa Anda telah mengonfigurasi izin yang sesuai untuk mengakses Athena dan bucket Amazon S3 tempat data kueri seharusnya berada. Untuk informasi selengkapnya, lihat Mengatur dan Memulai.

Topik

Meminta log Application Load Balancer

Application Load Balancer adalah opsi load balancing untuk LoElastic Load Balancing yang memungkinkan distribusi lalu lintas dalam deployment mikroservices menggunakan kontainer. Kueri Application Load Balancer log memungkinkan Anda melihat sumber lalu lintas, latency, dan byte yang ditransfer ke dan dari contoh Elastic Load Balancing dan aplikasi backend. Untuk informasi selengkapnya, lihat Log akses untuk Application Load Balancer dan log Koneksi untuk Application Load Balancer di Panduan Pengguna untuk Application Load Balancer.

Topik

Prasyarat

- Aktifkan pencatatan akses atau pencatatan koneksi sehingga log Application Load Balancer dapat disimpan ke bucket Amazon S3 Anda.
- Database untuk menyimpan tabel yang akan Anda buat untuk Athena. Untuk membuat database, Anda dapat menggunakan Athena atau AWS Glue konsol. Untuk informasi selengkapnya, lihat Membuat database di Athena di panduan ini atau Bekerja dengan database di konsol AWS lem di Panduan AWS Glue Pengembang.

Membuat tabel untuk log akses ALB

1. Salin dan tempel `CREATE TABLE` pernyataan berikut ke editor kueri di konsol Athena, lalu modifikasi seperlunya untuk persyaratan entri log Anda sendiri. Untuk informasi tentang memulai dengan konsol Athena, lihat. Memulai Ganti jalur di `LOCATION` klausa dengan lokasi folder log akses Amazon S3 Anda. Untuk informasi selengkapnya tentang lokasi file log akses, lihat Mengakses file log di Panduan Pengguna untuk Penyeimbang Beban Aplikasi.

   Untuk informasi tentang setiap bidang file log, lihat Mengakses entri log di Panduan Pengguna untuk Penyeimbang Beban Aplikasi.

   > **Note**
   >
   > `CREATE TABLE`Pernyataan contoh berikut mencakup kolom yang baru ditambahkan`classification`,`classification_reason`, dan `conn_trace_id` ('ID ketertelusuran', atau TID). Untuk membuat tabel untuk log akses Application Load

Balancer yang tidak berisi entri ini, hapus kolom yang sesuai dari `CREATE TABLE` pernyataan dan modifikasi ekspresi reguler yang sesuai.

```
CREATE EXTERNAL TABLE IF NOT EXISTS alb_access_logs (
             type string,
             time string,
             elb string,
             client_ip string,
             client_port int,
             target_ip string,
             target_port int,
             request_processing_time double,
             target_processing_time double,
             response_processing_time double,
             elb_status_code int,
             target_status_code string,
             received_bytes bigint,
             sent_bytes bigint,
             request_verb string,
             request_url string,
             request_proto string,
             user_agent string,
             ssl_cipher string,
             ssl_protocol string,
             target_group_arn string,
             trace_id string,
             domain_name string,
             chosen_cert_arn string,
             matched_rule_priority string,
             request_creation_time string,
             actions_executed string,
             redirect_url string,
             lambda_error_reason string,
             target_port_list string,
             target_status_code_list string,
             classification string,
             classification_reason string,
             conn_trace_id string
             )
             ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
             WITH SERDEPROPERTIES (
```

```
            'serialization.format' = '1',
            'input.regex' =
        '([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*):([0-9]*) ([^ ]*)[:-]([0-9]*) ([-.0-9]*)
 ([-.0-9]*) ([-.0-9]*) (|[-0-9]*) (-|[-0-9]*) ([-0-9]*) ([-0-9]*) \"([^ ]*) (.*) (-
 |[^ ]*)\" \"([^\"]*)\" ([A-Z0-9-_]+) ([A-Za-z0-9.-]*) ([^ ]*) \"([^\"]*)\" \"([^
\"]*)\" \"([^\"]*)\" ([-.0-9]*) ([^ ]*) \"([^\"]*)\" \"([^\"]*)\" \"([^ ]*)\" \"([^
\s]+?)\" \"([^\s]+)\" \"([^ ]*)\" \"([^ ]*)\" ?([^ ]*)?( .*)?')
            LOCATION 's3://DOC-EXAMPLE-BUCKET/access-log-folder-path/'
```

2. Jalankan kueri di konsol Athena. Setelah kueri selesai, Athena register`alb_access_logs`tabel, membuat data di dalamnya siap bagi Anda untuk mengeluarkan kueri.

Membuat tabel untuk log akses ALB di Athena menggunakan proyeksi partisi

Karena log akses ALB memiliki struktur yang diketahui skema partisi yang dapat Anda tentukan sebelumnya, Anda dapat mengurangi runtime kueri dan mengotomatiskan manajemen partisi dengan menggunakan fitur proyeksi partisi Athena. Proyeksi partisi secara otomatis menambahkan partisi baru saat data baru ditambahkan. Ini menghapus kebutuhan bagi Anda untuk menambahkan partisi secara manual dengan menggunakan`ALTER TABLE ADD PARTITION`.

`CREATE TABLE`Pernyataan contoh berikut secara otomatis menggunakan proyeksi partisi pada log akses ALB dari tanggal tertentu hingga saat ini untuk satu AWS wilayah. Pernyataan ini didasarkan pada contoh di bagian sebelumnya tetapi menambahkan `PARTITIONED BY` dan `TBLPROPERTIES` klausa untuk mengaktifkan proyeksi partisi. Di `storage.location.template` klausa `LOCATION` dan, ganti placeholder dengan nilai yang mengidentifikasi lokasi bucket Amazon S3 dari log akses ALB Anda. Untuk informasi selengkapnya tentang lokasi file log akses, lihat Mengakses file log di Panduan Pengguna untuk Penyeimbang Beban Aplikasi. Untuk`projection.day.range`, ganti *2022*/*01*/*01* dengan tanggal mulai yang ingin Anda gunakan. Setelah Anda menjalankan kueri dengan sukses, Anda dapat meminta tabel. Anda tidak perlu menjalankan `ALTER TABLE ADD PARTITION` untuk memuat partisi. Untuk informasi tentang setiap bidang file log, lihat Mengakses entri log.

```
CREATE EXTERNAL TABLE IF NOT EXISTS alb_access_logs (
            type string,
            time string,
            elb string,
            client_ip string,
            client_port int,
            target_ip string,
            target_port int,
```

```
            request_processing_time double,
            target_processing_time double,
            response_processing_time double,
            elb_status_code int,
            target_status_code string,
            received_bytes bigint,
            sent_bytes bigint,
            request_verb string,
            request_url string,
            request_proto string,
            user_agent string,
            ssl_cipher string,
            ssl_protocol string,
            target_group_arn string,
            trace_id string,
            domain_name string,
            chosen_cert_arn string,
            matched_rule_priority string,
            request_creation_time string,
            actions_executed string,
            redirect_url string,
            lambda_error_reason string,
            target_port_list string,
            target_status_code_list string,
            classification string,
            classification_reason string,
            conn_trace_id string
            )
            PARTITIONED BY
            (
             day STRING
            )
            ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
            WITH SERDEPROPERTIES (
            'serialization.format' = '1',
            'input.regex' =
        '([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*):([0-9]*) ([^ ]*)[:-]([0-9]*) ([-.0-9]*)
 ([-.0-9]*) ([-.0-9]*) (|[-0-9]*) (-|[-0-9]*) ([-0-9]*) ([-0-9]*) \"([^ ]*) (.*) (- |
[^ ]*)\" \"([^\"]*)\" ([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}"
            )
```

Untuk informasi selengkapnya tentang proyek partisi, lihat Proyeksi partisi dengan Amazon Athena.

Contoh kueri untuk log akses ALB

Kueri berikut menghitung jumlah permintaan HTTP GET yang diterima oleh load balancer dikelompokkan berdasarkan alamat IP klien:

```
SELECT COUNT(request_verb) AS
 count,
 request_verb,
 client_ip
FROM alb_access_logs
GROUP BY request_verb, client_ip
LIMIT 100;
```

Kueri lain menunjukkan URL yang dikunjungi oleh pengguna peramban Safari:

```
SELECT request_url
FROM alb_access_logs
WHERE user_agent LIKE '%Safari%'
LIMIT 10;
```

Kueri berikut menunjukkan catatan yang memiliki nilai kode status ELB lebih besar dari atau sama dengan 500.

```
SELECT * FROM alb_access_logs
WHERE elb_status_code >= 500
```

Contoh berikut menunjukkan cara mengurai log dengan`datetime`:

```
SELECT client_ip, sum(received_bytes)
```

```
FROM alb_access_logs
WHERE parse_datetime(time,'yyyy-MM-dd''T''HH:mm:ss.SSSSSS''Z')
     BETWEEN parse_datetime('2018-05-30-12:00:00','yyyy-MM-dd-HH:mm:ss')
     AND parse_datetime('2018-05-31-00:00:00','yyyy-MM-dd-HH:mm:ss')
GROUP BY client_ip;
```

Kueri berikut menanyakan tabel yang menggunakan proyeksi partisi untuk semua log akses ALB dari hari yang ditentukan.

```
SELECT *
FROM alb_access_logs
WHERE day = '2022/02/12'
```

Membuat tabel untuk log koneksi ALB

1. Salin dan tempel `CREATE TABLE` pernyataan contoh berikut ke editor kueri di konsol Athena, lalu modifikasi seperlunya untuk persyaratan entri log Anda sendiri. Untuk informasi tentang memulai dengan konsol Athena, lihat. Memulai Ganti jalur di `LOCATION` klausa dengan lokasi folder log koneksi Amazon S3 Anda. Untuk informasi selengkapnya tentang lokasi file log koneksi, lihat File log koneksi di Panduan Pengguna untuk Penyeimbang Beban Aplikasi. Untuk informasi tentang setiap bidang file log, lihat Entri log koneksi.

   ```
   CREATE EXTERNAL TABLE IF NOT EXISTS alb_connection_logs (
            time string,
            client_ip string,
            client_port int,
            listener_port int,
            tls_protocol string,
            tls_cipher string,
            tls_handshake_latency double,
            leaf_client_cert_subject string,
            leaf_client_cert_validity string,
            leaf_client_cert_serial_number string,
            tls_verify_status string,
            conn_trace_id string
            )
            ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
            WITH SERDEPROPERTIES (
            'serialization.format' = '1',
            'input.regex' =
             '([^ ]*) ([^ ]*) ([0-9]*) ([0-9]*) ([A-Za-z0-9.-]*) ([^ ]*) ([-.0-9]*)
    \"([^\"]*)\" ([^ ]*) ([^ ]*) ([^ ]*) ?([^ ]*)?( .*)?')
   ```

```
        LOCATION 's3://DOC-EXAMPLE-BUCKET/connection-log-folder-path/'
```

2. Jalankan kueri di konsol Athena. Setelah kueri selesai, Athena registeralb_connection_logstabel, membuat data di dalamnya siap bagi Anda untuk mengeluarkan kueri.

Membuat tabel untuk log koneksi ALB di Athena menggunakan proyeksi partisi

Karena log koneksi ALB memiliki struktur yang diketahui yang skema partisi dapat Anda tentukan sebelumnya, Anda dapat mengurangi runtime kueri dan mengotomatiskan manajemen partisi dengan menggunakan fitur proyeksi partisi Athena. Proyeksi partisi secara otomatis menambahkan partisi baru saat data baru ditambahkan. Ini menghapus kebutuhan bagi Anda untuk menambahkan partisi secara manual dengan menggunakan`ALTER TABLE ADD PARTITION`.

`CREATE TABLE`Pernyataan contoh berikut secara otomatis menggunakan proyeksi partisi pada log koneksi ALB dari tanggal tertentu hingga saat ini untuk satu AWS wilayah. Pernyataan ini didasarkan pada contoh di bagian sebelumnya tetapi menambahkan `PARTITIONED BY` dan `TBLPROPERTIES` klausa untuk mengaktifkan proyeksi partisi. Di `storage.location.template` klausa `LOCATION` dan, ganti placeholder dengan nilai yang mengidentifikasi lokasi bucket Amazon S3 dari log koneksi ALB Anda. Untuk informasi selengkapnya tentang lokasi file log koneksi, lihat File log koneksi di Panduan Pengguna untuk Penyeimbang Beban Aplikasi. Untuk`projection.day.range`, ganti *`2023/01/01`* dengan tanggal mulai yang ingin Anda gunakan. Setelah Anda menjalankan kueri dengan sukses, Anda dapat meminta tabel. Anda tidak perlu menjalankan `ALTER TABLE ADD PARTITION` untuk memuat partisi. Untuk informasi tentang setiap bidang file log, lihat Entri log koneksi.

```
CREATE EXTERNAL TABLE IF NOT EXISTS alb_connection_logs (
         time string,
         client_ip string,
         client_port int,
         listener_port int,
         tls_protocol string,
         tls_cipher string,
         tls_handshake_latency double,
         leaf_client_cert_subject string,
         leaf_client_cert_validity string,
         leaf_client_cert_serial_number string,
         tls_verify_status string,
         conn_trace_id string
         )
```

```
            PARTITIONED BY
            (
             day STRING
            )
            ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
            WITH SERDEPROPERTIES (
            'serialization.format' = '1',
            'input.regex' =
             '([^ ]*) ([^ ]*) ([0-9]*) ([0-9]*) ([A-Za-z0-9.-]*) ([^ ]*) ([-.0-9]*)
 \"([^\"]*)\" ([^ ]*) ([^ ]*) ([^ ]*) ?([^ ]*)?( .*)?')
            LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/<ACCOUNT-NUMBER>/
elasticloadbalancing/<REGION>/'
            TBLPROPERTIES
            (
             "projection.enabled" = "true",
             "projection.day.type" = "date",
             "projection.day.range" = "2023/01/01,NOW",
             "projection.day.format" = "yyyy/MM/dd",
             "projection.day.interval" = "1",
             "projection.day.interval.unit" = "DAYS",
             "storage.location.template" = "s3://DOC-EXAMPLE-BUCKET/AWSLogs/<ACCOUNT-
NUMBER>/elasticloadbalancing/<REGION>/${day}"
            )
```

Untuk informasi selengkapnya tentang proyek partisi, lihat Proyeksi partisi dengan Amazon Athena.

Contoh kueri untuk log koneksi ALB

Perhitungan kueri berikut terjadi di mana nilai untuk `tls_verify_status` tidak`'Success'`, dikelompokkan berdasarkan alamat IP klien:

```
SELECT DISTINCT client_ip, count() AS count FROM alb_connection_logs
WHERE tls_verify_status != 'Success'
GROUP BY client_ip
ORDER BY count() DESC;
```

Kueri berikut mencari kejadian di mana nilai untuk `tls_handshake_latency` lebih dari 2 detik dalam rentang waktu yang ditentukan:

```
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);
```

Sumber daya tambahan

- Bagaimana cara menganalisis log akses Application Load Balancer menggunakan Amazon Athena di AWS Pusat Pengetahuan.
- Untuk informasi tentang kode status HTTP di Elastic Load Balancing, lihat Memecahkan masalah load balancer aplikasi Anda di Panduan Pengguna untuk Application Load Balancers.
- Katalog dan analisis log Application Load Balancer lebih efisien dengan pengklasifikasi AWS Glue kustom dan Amazon Athena di Big Data Blog.AWS

Meminta log Classic Load Balancer

Gunakan log Classic Load Balancer untuk menganalisis dan memahami pola lalu lintas ke dan dari contoh Elastic Load Balancing dan aplikasi backend. Anda dapat melihat sumber lalu lintas, latency, dan byte yang telah ditransfer.

Sebelum Anda menganalisis log Elastic Load Balancing, mengonfigurasi mereka untuk menyimpan di tujuan Amazon S3 bucket. Untuk informasi selengkapnya, lihat Mengaktifkan log akses untuk Classic Load Balancer Anda.

- Buat tabel untuk log Elastic Load Balancing
- Contoh pertanyaan Elastic Load Balancing

Untuk membuat tabel untuk log Elastic Load Balancing

1. Salin dan tempel pernyataan DDL berikut ke konsol Athena. PeriksasintaksCatatan log Elastic Load Balancing. Anda mungkin perlu memperbarui kueri berikut untuk menyertakan kolom dan sintaks Regex untuk versi terbaru dari catatan.

   ```
   CREATE EXTERNAL TABLE IF NOT EXISTS elb_logs (
   ```

```
 timestamp string,
 elb_name string,
 request_ip string,
 request_port int,
 backend_ip string,
 backend_port int,
 request_processing_time double,
 backend_processing_time double,
 client_response_time double,
 elb_response_code string,
 backend_response_code string,
 received_bytes bigint,
 sent_bytes bigint,
 request_verb string,
 url string,
 protocol string,
 user_agent string,
 ssl_cipher string,
 ssl_protocol string
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
 'serialization.format' = '1',
 'input.regex' = '([^ ]*) ([^ ]*) ([^ ]*):([0-9]*) ([^ ]*)[:-]([0-9]*) ([-.0-9]*)
 ([-.0-9]*) ([-.0-9]*) (|[-0-9]*) (-|[-0-9]*) ([-0-9]*) ([-0-9]*) \\\"([^ ]*)
 ([^ ]*) (- |[^ ]*)\\\" (\"[^\"]*\") ([A-Z0-9-]+) ([A-Za-z0-9.-]*)$'
)
LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/AWS_account_ID/elasticloadbalancing/';
```

2. MemodifikasiLOCATIONAmazon S3 bucket untuk menentukan tujuan log Elastic Load Balancing Anda.
3. Jalankan kueri di konsol Athena. Setelah kueri selesai, Athena registerelb_logstabel, membuat data di dalamnya siap untuk mengkueri. Untuk informasi selengkapnya, lihat Contoh pertanyaan Elastic Load Balancing.

Contoh pertanyaan Elastic Load Balancing

Gunakan kueri yang mirip dengan contoh berikut. Ini daftar server aplikasi backend yang mengembalikan sebuah4XXatau5XXKode respons kesalahan. MenggunakanLIMITOperator untuk membatasi jumlah log untuk mengkueri pada suatu waktu.

```
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;
```

Gunakan kueri berikutnya untuk meringkas waktu respon dari semua transaksi dikelompokkan berdasarkan alamat IP backend dan nama contoh Elastic Load Balancing.

```
SELECT sum(backend_processing_time) AS
 total_ms,
 elb_name,
 backend_ip
FROM elb_logs WHERE backend_ip <> ''
GROUP BY backend_ip, elb_name
LIMIT 100;
```

Untuk informasi lebih lanjut, lihat Menganalisis data di S3 menggunakan Athena.

Menanyakan log Amazon CloudFront

Anda dapat mengonfigurasi Amazon CloudFront CDN untuk mengekspor log akses distribusi Web ke Amazon Simple Storage Service. Gunakan log ini untuk menjelajahi pola selancar pengguna di seluruh properti web Anda yang dilayani oleh. CloudFront

Sebelum Anda mulai menanyakan log, aktifkan log akses distribusi Web pada distribusi pilihan CloudFront Anda. Untuk selengkapnya, lihat Akses log di Panduan CloudFront Pengembang Amazon. Catat bucket Amazon S3 tempat Anda menyimpan log ini.

- Membuat tabel untuk log CloudFront standar
- Membuat tabel untuk log CloudFront waktu nyata
- Contoh kueri untuk log standar CloudFront

Membuat tabel untuk log CloudFront standar

> **Note**
>
> Prosedur ini berfungsi untuk log akses distribusi Web CloudFront. Ini tidak sahih untuk streaming log dari RTMP distribusi.

Untuk membuat tabel untuk bidang file log CloudFront standar

1. Salin dan tempel contoh pernyataan DDL berikut ke Editor Kueri di konsol Athena. Pernyataan contoh menggunakan bidang file log yang didokumentasikan di bagian bidang file log Standar dari Panduan CloudFront Pengembang Amazon. MemodifikasiLOCATIONuntuk bucket Amazon S3 yang menyimpan log Anda. Untuk informasi selengkapnya, lihat Memulai Penggunaan editor kueri untuk

   Kueri ini menentukan `ROW FORMAT DELIMITED` dan `FIELDS TERMINATED BY '\t'` untuk menunjukkan bahwa bidang dibatasi oleh karakter tab. Untuk`ROW FORMAT DELIMITED`, Athena menggunakan secara LazySimpleSerDedefault. Kolom`date`melarikan diri menggunakan backticks (`) karena itu adalah kata reserved di Athena. Untuk informasi, lihat Kata kunci terpesan.

```
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. Jalankan kueri di konsol Athena. Setelah kueri selesai, Athena registercloudfront_standard_logstabel, membuat data di dalamnya siap bagi Anda untuk mengeluarkan kueri.

Membuat tabel untuk log CloudFront waktu nyata

Untuk membuat tabel untuk bidang file log CloudFront real-time

1. Salin dan tempel contoh pernyataan DDL berikut ke Editor Kueri di konsol Athena. Pernyataan contoh menggunakan bidang file log yang didokumentasikan di bagian log Real-time dari Panduan CloudFront Pengembang Amazon. MemodifikasiLOCATIONuntuk bucket Amazon S3 yang menyimpan log Anda. Untuk informasi selengkapnya, lihat Memulai Penggunaan editor kueri untuk

   Kueri ini menentukan `ROW FORMAT DELIMITED` dan `FIELDS TERMINATED BY '\t'` untuk menunjukkan bahwa bidang dibatasi oleh karakter tab. Untuk`ROW FORMAT DELIMITED`, Athena menggunakan secara LazySimpleSerDedefault. Kolom`timestamp`melarikan diri menggunakan backticks (`) karena itu adalah kata reserved di Athena. Untuk informasi, lihat Kata kunci terpesan.

   Contoh berikut berisi semua bidang yang tersedia. Anda dapat mengomentari atau menghapus bidang yang tidak Anda perlukan.

```
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. Jalankan kueri di konsol Athena. Setelah kueri selesai, Athena registercloudfront_real_time_logstabel, membuat data di dalamnya siap bagi Anda untuk mengeluarkan kueri.

Contoh kueri untuk log standar CloudFront

Kueri berikut menambahkan jumlah byte yang dilayani CloudFront antara 9 Juni dan 11 Juni 2018. Mengelilingi nama kolom tanggal dengan tanda kutip ganda karena merupakan kata reserved.

```
SELECT SUM(bytes) AS total_bytes
FROM cloudfront_standard_logs
WHERE "date" BETWEEN DATE '2018-06-09' AND DATE '2018-06-11'
LIMIT 100;
```

Untuk menghilangkan duplikat baris (misalnya, duplikat baris kosong) dari hasil kueri, Anda dapat menggunakanSELECT DISTINCTpernyataan, seperti dalam contoh berikut.

```
SELECT DISTINCT *
FROM cloudfront_standard_logs
LIMIT 10;
```

Sumber daya tambahan

Untuk informasi lebih lanjut tentang menggunakan Athena untuk menanyakan CloudFront log, lihat posting berikut dari blog data AWS besar.

Kueri Layanan AWS log dengan mudah menggunakan Amazon Athena (29 Mei 2019).

Analisis log CloudFront akses Amazon Anda dalam skala besar (21 Desember 2018).

Bangun arsitektur tanpa server untuk menganalisis log CloudFront akses Amazon menggunakan AWS Lambda, Amazon Athena, dan Amazon Managed Service untuk Apache Flink (26 Mei 2017).

Meminta log AWS CloudTrail

AWS CloudTrail adalah layanan yang merekam panggilan AWS API dan peristiwa untuk akun Amazon Web Services.

CloudTrail log menyertakan detail tentang panggilan API apa pun yang dilakukan ke Anda Layanan AWS, termasuk konsol. CloudTrail menghasilkan file log terenkripsi dan menyimpannya di Amazon S3. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna AWS CloudTrail.

> **Note**
>
> Jika Anda ingin melakukan kueri SQL pada informasi CloudTrail acara di seluruh akun, wilayah, dan tanggal, pertimbangkan untuk menggunakan CloudTrail Lake. CloudTrail Lake adalah AWS alternatif untuk membuat jalur yang mengumpulkan informasi dari suatu perusahaan ke dalam satu penyimpanan data peristiwa yang dapat dicari. Alih-alih menggunakan penyimpanan bucket Amazon S3, ia menyimpan peristiwa di danau data, yang memungkinkan kueri yang lebih kaya dan lebih cepat. Anda dapat menggunakannya untuk membuat kueri SQL yang mencari peristiwa di seluruh organisasi, wilayah, dan dalam rentang waktu khusus. Karena Anda melakukan kueri CloudTrail Danau di dalam CloudTrail konsol itu sendiri, menggunakan CloudTrail Lake tidak memerlukan Athena. Untuk informasi lebih lanjut, lihat dokumentasi CloudTrail Danau.

Menggunakan Athena dengan CloudTrail log adalah cara ampuh untuk meningkatkan analisis aktivitas Anda. Layanan AWS Misalnya, Anda dapat menggunakan kueri untuk mengidentifikasi tren dan mengisolasi aktivitas selengkapnya berdasarkan atribut seperti alamat IP sumber atau pengguna.

Aplikasi umum adalah menggunakan CloudTrail log untuk menganalisis aktivitas operasional untuk keamanan dan kepatuhan. Untuk informasi tentang contoh terperinci, lihat posting Blog AWS Big Data, Menganalisis keamanan, kepatuhan, dan aktivitas operasional menggunakan AWS CloudTrail dan Amazon Athena.

Anda dapat menggunakan Athena untuk mengkueri berkas log ini langsung dari Amazon S3, menentukan`LOCATION`Berkas log. Anda dapat melakukannya dengan salah satu dari dua cara berikut:

- Dengan membuat tabel untuk file CloudTrail log langsung dari CloudTrail konsol.
- Dengan membuat tabel secara manual untuk file CloudTrail log di konsol Athena.

Topik

- Memahami CloudTrail log dan tabel Athena
- Menggunakan CloudTrail konsol untuk membuat tabel Athena untuk log CloudTrail
- Membuat tabel untuk CloudTrail log di Athena menggunakan partisi manual
- Membuat tabel untuk jejak luas organisasi menggunakan partisi manual
- Membuat tabel untuk CloudTrail log di Athena menggunakan proyeksi partisi
- Menanyakan bidang bersarang
- Kueri contoh
- Kiat untuk menanyakan log CloudTrail

Memahami CloudTrail log dan tabel Athena

Sebelum Anda mulai membuat tabel, Anda harus memahami sedikit lebih banyak tentang CloudTrail dan bagaimana menyimpan data. Ini dapat membantu Anda membuat tabel yang Anda butuhkan, apakah Anda membuatnya dari CloudTrail konsol atau dari Athena.

CloudTrail menyimpan log sebagai file teks JSON dalam format gzip terkompresi (*.json.gzip). Lokasi file log tergantung pada cara Anda mengatur jejak, Wilayah AWS atau Wilayah tempat Anda masuk, dan faktor lainnya.

Untuk informasi selengkapnya tentang tempat log disimpan, struktur JSON, dan isi file catatan, lihat topik berikut diAWS CloudTrail Panduan Pengguna:

- Menemukan file CloudTrail log Anda
- CloudTrail Contoh File Log
- CloudTrail isi rekaman
- CloudTrail referensi acara

Untuk mengumpulkan log dan menyimpannya ke Amazon S3, aktifkan CloudTrail dari file. AWS Management Console Untuk informasi selengkapnya, lihat Membuat jejak di Panduan AWS CloudTrail Pengguna.

Perhatikan bucket Amazon S3 tujuan tempat Anda menyimpan log. Ganti `LOCATION` klausa dengan jalur ke lokasi CloudTrail log dan kumpulan objek yang dapat digunakan untuk bekerja. Contoh

menggunakanLOCATIONnilai log untuk akun tertentu, tetapi Anda dapat menggunakan level kekhususan yang sesuai dengan aplikasi Anda.

Sebagai contoh:

- Untuk menganalisis data dari beberapa akun, Anda dapat memutar kembaliLOCATIONspecifier untuk menunjukkan semuaAWSLogsdengan menggunakanLOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/'.
- Untuk menganalisis data dari tanggal, akun, dan Wilayah tertentu, gunakanLOCATION 's3://DOC-EXAMPLE-BUCKET/123456789012/CloudTrail/us-east-1/2016/03/14/'.

Menggunakan level tertinggi dalam hirarki objek memberikan fleksibilitas terbesar saat Anda kueri menggunakan Athena.

Menggunakan CloudTrail konsol untuk membuat tabel Athena untuk log CloudTrail

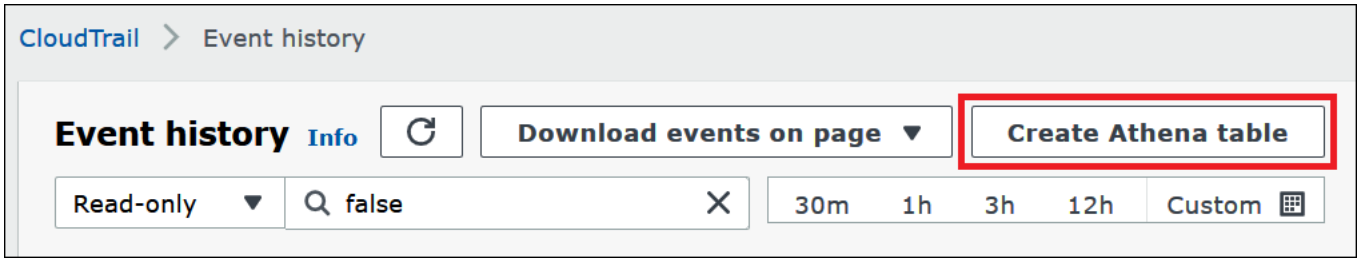
Anda dapat membuat tabel Athena yang tidak dipartisi untuk CloudTrail menanyakan log langsung dari konsol. CloudTrail Membuat tabel Athena dari CloudTrail konsol mengharuskan Anda masuk dengan peran yang memiliki izin yang cukup untuk membuat tabel di Athena.

> **Note**
>
> Anda tidak dapat menggunakan CloudTrail konsol untuk membuat tabel Athena untuk log jejak organisasi. Sebaliknya, buat tabel secara manual menggunakan konsol Athena sehingga Anda dapat menentukan lokasi penyimpanan yang benar. Untuk informasi tentang jalur organisasi, lihatMembuat jejak untuk organisasidiAWS CloudTrail Panduan Pengguna.

- Untuk informasi tentang pengaturan izin untuk Athena, lihatMengatur.
- Untuk informasi tentang membuat tabel dengan partisi, lihatMembuat tabel untuk CloudTrail log di Athena menggunakan partisi manual.

Untuk membuat tabel Athena untuk CloudTrail jejak menggunakan konsol CloudTrail

1. Buka CloudTrail konsol di https://console.aws.amazon.com/cloudtrail/.
2. Pada panel navigasi, pilih Riwayat peristiwa.
3. PilihMembuat tabel Athena.



4. UntukLokasi penyimpanan, gunakan panah bawah untuk memilih bucket Amazon S3 tempat berkas log disimpan untuk jejak untuk mengkueri.

> **Note**
>
> Untuk menemukan nama bucket yang dikaitkan dengan jejak, pilih Trails di panel CloudTrail navigasi dan lihat kolom bucket S3 trail. Untuk melihat lokasi Amazon S3 untuk bucket, pilih tautan untuk bucket diBucket S3kolom. Ini membuka konsol Amazon S3 ke lokasi CloudTrail bucket.

5. Pilih Buat tabel. Tabel dibuat dengan nama default yang mencakup nama bucket Amazon S3.

Membuat tabel untuk CloudTrail log di Athena menggunakan partisi manual

Anda dapat secara manual membuat tabel untuk file CloudTrail log di konsol Athena, dan kemudian menjalankan kueri di Athena.

Untuk membuat tabel Athena untuk CloudTrail jejak menggunakan konsol Athena

1. Salin dan tempel pernyataan DDL berikut ke editor kueri konsol Athena.

```
CREATE EXTERNAL TABLE cloudtrail_logs (
eventversion STRING,
useridentity STRUCT<
               type:STRING,
               principalid:STRING,
               arn:STRING,
               accountid:STRING,
               invokedby:STRING,
               accesskeyid:STRING,
               userName:STRING,
  sessioncontext:STRUCT<
    attributes:STRUCT<
               mfaauthenticated:STRING,
```

```
                creationdate:STRING>,
    sessionissuer:STRUCT<
                type:STRING,
                principalId:STRING,
                arn:STRING,
                accountId:STRING,
                userName:STRING>,
    ec2RoleDelivery:string,
    webIdFederationData: STRUCT<
                federatedProvider: STRING,
                attributes: map<string,string>
    >
  >
>,
eventtime STRING,
eventsource STRING,
eventname STRING,
awsregion STRING,
sourceipaddress STRING,
useragent STRING,
errorcode STRING,
errormessage STRING,
requestparameters STRING,
responseelements STRING,
additionaleventdata STRING,
requestid STRING,
eventid STRING,
resources ARRAY<STRUCT<
                arn:STRING,
                accountid:STRING,
                type:STRING>>,
eventtype STRING,
apiversion STRING,
readonly STRING,
recipientaccountid STRING,
serviceeventdetails STRING,
sharedeventid STRING,
vpcendpointid STRING,
eventCategory STRING,
tlsDetails struct<
  tlsVersion:string,
  cipherSuite:string,
  clientProvidedHostHeader:string>
)
```

```
PARTITIONED BY (region string, year string, month string, day string)
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
STORED AS INPUTFORMAT 'com.amazon.emr.cloudtrail.CloudTrailInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/Account_ID/CloudTrail/';
```

> **Note**
>
> Kami menyarankan menggunakan yang `org.apache.hive.hcatalog.data.JsonSerDe` ditunjukkan dalam contoh. Meskipun `com.amazon.emr.hive.serde.CloudTrailSerde` ada, saat ini tidak menangani beberapa CloudTrail bidang yang lebih baru.

2. (Opsional) Hapus semua bidang yang tidak diperlukan untuk tabel Anda. Jika Anda hanya perlu membaca satu set kolom tertentu, definisi tabel Anda dapat mengecualikan kolom lainnya.
3. Memodifikasi`s3://DOC-EXAMPLE-BUCKET/AWSLogs/Account_ID/CloudTrail/`Untuk menunjuk ke bucket Amazon S3 yang berisi data log Anda.
4. Verifikasi bahwa bidang terdaftar dengan benar. Untuk informasi selengkapnya tentang daftar lengkap bidang dalam CloudTrail catatan, lihat konten CloudTrail rekaman.

   `CREATE TABLE`Pernyataan contoh di Langkah 1 menggunakanSarang JSON SerDe. Dalam contoh, bidang, `requestparametersresponseelements`, dan `additionaleventdata` terdaftar sebagai tipe `STRING` dalam kueri, tetapi tipe `STRUCT` data yang digunakan dalam JSON. Oleh karena itu, untuk mendapatkan data dari bidang ini, gunakan`JSON_EXTRACT`fungsi. Untuk informasi selengkapnya, lihat the section called “Mengekstrak data JSON dari string”. Untuk peningkatan kinerja, contoh mempartisi data berdasarkan Wilayah AWS, tahun, bulan, dan hari.
5. Jalankan `CREATE TABLE` pernyataan di konsol Athena.
6. MenggunakanALTER TABLE ADD PARTITIONperintah untuk memuat partisi sehingga Anda dapat meminta mereka, seperti dalam contoh berikut.

```
ALTER TABLE table_name ADD
   PARTITION (region='us-east-1',
              year='2019',
              month='02',
              day='01')
```

```
   LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/Account_ID/CloudTrail/us-
east-1/2019/02/01/'
```

Membuat tabel untuk jejak luas organisasi menggunakan partisi manual

Untuk membuat tabel untuk file CloudTrail log luas organisasi di Athena, ikuti langkah-langkahnyaMembuat tabel untuk CloudTrail log di Athena menggunakan partisi manual, tetapi buat modifikasi yang dicatat dalam prosedur berikut.

Untuk membuat tabel Athena untuk log lebar organisasi CloudTrail

1. Dalam `CREATE TABLE` pernyataan tersebut, ubah `LOCATION` klausa untuk menyertakan ID organisasi, seperti pada contoh berikut:

   ```
   LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/organization_id/Account_ID/CloudTrail/'
   ```

2. Dalam `PARTITIONED BY` klausa, tambahkan entri untuk ID akun sebagai string, seperti pada contoh berikut:

   ```
   PARTITIONED BY (account string, region string, year string, month string, day
    string)
   ```

   Contoh berikut menunjukkan hasil gabungan:

   ```
   ...

   PARTITIONED BY (account string, region string, year string, month string, day
    string)
   ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
   STORED AS INPUTFORMAT 'com.amazon.emr.cloudtrail.CloudTrailInputFormat'
   OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
   LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/organization_id/Account_ID/CloudTrail/'
   ```

3. Dalam `ADD PARTITION` klausa `ALTER TABLE` pernyataan, sertakan ID akun, seperti pada contoh berikut:

   ```
   ALTER TABLE table_name ADD
   PARTITION (account='111122223333',
   region='us-east-1',
   year='2022',
   ```

```
month='08',
day='08')
```

4. Dalam `LOCATION` klausa `ALTER TABLE` pernyataan, sertakan ID organisasi, ID akun, dan partisi yang ingin Anda tambahkan, seperti pada contoh berikut:

```
LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/organization_id/Account_ID/CloudTrail/us-
east-1/2022/08/08/'
```

`ALTER TABLE`Pernyataan contoh berikut menunjukkan hasil gabungan:

```
ALTER TABLE table_name ADD
PARTITION (account='111122223333',
region='us-east-1',
year='2022',
month='08',
day='08')
LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/organization_id/111122223333/CloudTrail/
us-east-1/2022/08/08/'
```

Membuat tabel untuk CloudTrail log di Athena menggunakan proyeksi partisi

Karena CloudTrail log memiliki struktur yang diketahui yang skema partisi dapat Anda tentukan sebelumnya, Anda dapat mengurangi runtime kueri dan mengotomatiskan manajemen partisi dengan menggunakan fitur proyeksi partisi Athena. Proyeksi partisi secara otomatis menambahkan partisi baru saat data baru ditambahkan. Ini menghapus kebutuhan bagi Anda untuk menambahkan partisi secara manual dengan menggunakan`ALTER TABLE ADD PARTITION`.

`CREATE TABLE`Pernyataan contoh berikut secara otomatis menggunakan proyeksi partisi pada CloudTrail log dari tanggal tertentu hingga saat ini untuk satu Wilayah AWS. Di`LOCATION`dan`storage.location.template`klausa, ganti*bucket*,*akun-id*, dan*aws-region*placeholder dengan nilai-nilai yang sama. Untuk`projection.timestamp.range`, ganti*2020*/*01*/*01*dengan tanggal mulai yang ingin Anda gunakan. Setelah Anda menjalankan kueri dengan sukses, Anda dapat meminta tabel. Anda tidak perlu menjalankan `ALTER TABLE ADD PARTITION` untuk memuat partisi.

```
CREATE EXTERNAL TABLE cloudtrail_logs_pp(
    eventVersion STRING,
    userIdentity STRUCT<
```

```
        type: STRING,
        principalId: STRING,
        arn: STRING,
        accountId: STRING,
        invokedBy: STRING,
        accessKeyId: STRING,
        userName: STRING,
        sessionContext: STRUCT<
            attributes: STRUCT<
                mfaAuthenticated: STRING,
                creationDate: STRING>,
            sessionIssuer: STRUCT<
                type: STRING,
                principalId: STRING,
                arn: STRING,
                accountId: STRING,
                userName: STRING>,
            ec2RoleDelivery:string,
            webIdFederationData: STRUCT<
                federatedProvider: STRING,
                attributes: map<string,string>
            >
        >
    >,
    eventTime STRING,
    eventSource STRING,
    eventName STRING,
    awsRegion STRING,
    sourceIpAddress STRING,
    userAgent STRING,
    errorCode STRING,
    errorMessage STRING,
    requestparameters STRING,
    responseelements STRING,
    additionaleventdata STRING,
    requestId STRING,
    eventId STRING,
    readOnly STRING,
    resources ARRAY<STRUCT<
        arn: STRING,
        accountId: STRING,
        type: STRING>>,
    eventType STRING,
    apiVersion STRING,
```

```
    recipientAccountId STRING,
    serviceEventDetails STRING,
    sharedEventID STRING,
    vpcendpointid STRING,
    eventCategory STRING,
    tlsDetails struct<
        tlsVersion:string,
        cipherSuite:string,
        clientProvidedHostHeader:string>
  )
PARTITIONED BY (
   `timestamp` string)
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
STORED AS INPUTFORMAT 'com.amazon.emr.cloudtrail.CloudTrailInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/AWSLogs/account-id/CloudTrail/aws-region'
TBLPROPERTIES (
  'projection.enabled'='true',
  'projection.timestamp.format'='yyyy/MM/dd',
  'projection.timestamp.interval'='1',
  'projection.timestamp.interval.unit'='DAYS',
  'projection.timestamp.range'='2020/01/01,NOW',
  'projection.timestamp.type'='date',
  'storage.location.template'='s3://DOC-EXAMPLE-BUCKET/AWSLogs/account-id/
CloudTrail/aws-region/${timestamp}')
```

Untuk informasi selengkapnya tentang proyek partisi, lihat Proyeksi partisi dengan Amazon Athena.

Menanyakan bidang bersarang

Karena`userIdentity`dan`resources`adalah tipe data Nest, kueri mereka memerlukan perlakuan khusus.

Parameter`userIdentity`objek terdiri dari bersarang`STRUCT`jenis. Ini dapat bertanya menggunakan titik untuk memisahkan bidang, seperti dalam contoh berikut:

```
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
```

Parameter`resources`bidang adalah sebuah larik dari`STRUCT`objek. Untuk larik ini, gunakan`CROSS JOIN UNNEST`untuk unnest larik sehingga Anda dapat mengkueri objeknya.

Contoh berikut mengembalikan semua baris tempat sumber daya ARN berakhir di`example/datafile.txt`. Untuk dibaca,mengganti fungsi menghapus awal`arn:aws:s3:::`substring dari 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
```

Contoh kueri berikut untuk`DeleteBucket`peristiwa. Kueri mengekstraksi nama bucket dan ID akun yang bucket milik`resources`objek.

```
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
```

Untuk informasi selengkapnya tentang unnesting, lihatArray penyaringan.

Kueri contoh

Contoh berikut menunjukkan sebagian kueri yang mengembalikan semua permintaan anonim (tidak ditandatangani) dari tabel yang dibuat untuk log CloudTrail peristiwa. Kueri ini memilih permintaan

tersebut tempat`useridentity.accountid`adalah anonim, dan`useridentity.arn`tidak ditentukan:

```
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 ]%';
```

Untuk informasi selengkapnya, lihat posting blog AWS Big Data Menganalisis keamanan, kepatuhan, dan aktivitas operasional menggunakan AWS CloudTrail Amazon Athena.

Kiat untuk menanyakan log CloudTrail

Untuk menjelajahi data CloudTrail log, gunakan tips ini:

- Sebelum kueri log, verifikasi bahwa tabel log Anda terlihat sama dengan yang dithe section called “Membuat tabel untuk CloudTrail log di Athena menggunakan partisi manual”. Jika tidak tabel pertama, menghapus tabel yang ada menggunakan perintah berikut:`DROP TABLE cloudtrail_logs`.
- Setelah Anda drop tabel yang ada, kembali membuat itu. Untuk informasi selengkapnya, lihat Membuat tabel untuk CloudTrail log di Athena menggunakan partisi manual.

  Verifikasi bahwa bidang dalam permintaan Athena Anda terdaftar dengan benar. Untuk informasi tentang daftar lengkap bidang dalam CloudTrail catatan, lihat konten CloudTrail rekaman.

  Jika kueri Anda termasuk bidang dalam format JSON, sepertiSTRUCT, ekstrak data dari JSON. Untuk informasi selengkapnya, lihat Mengekstrak data JSON dari string.

  Beberapa saran untuk mengeluarkan kueri terhadap tabel Anda CloudTrail :
- Mulailah dengan melihat pengguna mana yang memanggil operasi API mana dan dari alamat IP sumber mana.
- Gunakan kueri SQL dasar berikut sebagai templat Anda. Tempelkan kueri ke konsol Athena dan jalankannya.

```
SELECT
 useridentity.arn,
```

```
 eventname,
 sourceipaddress,
 eventtime
FROM cloudtrail_logs
LIMIT 100;
```

- Ubah kueri untuk mengeksplorasi data Anda lebih lanjut.
- Untuk meningkatkan performa, sertakan`LIMIT`klausa untuk mengembalikan subset tertentu dari baris.

Menanyakan log EMR Amazon

Amazon EMR dan aplikasi big data yang berjalan di Amazon EMR menghasilkan berkas log. File log ditulis ke simpul utama, dan Anda juga dapat mengonfigurasi Amazon EMR untuk mengarsipkan file log ke Amazon S3 secara otomatis. Anda dapat menggunakan Amazon Athena untuk mengkueri log ini untuk mengidentifikasi peristiwa dan tren untuk aplikasi dan cluster. Untuk informasi selengkapnya tentang jenis file log di Amazon EMR dan menyimpannya ke Amazon S3, lihat Melihat file log di Panduan Manajemen EMR Amazon.

Membuat dan menanyakan tabel dasar berdasarkan file log Amazon EMR

Contoh berikut membuat tabel dasar,`myemrlogs`, berdasarkan berkas log yang disimpan kes3://aws-logs-*123456789012*-*us-west-2*/elasticmapreduce/*j-2ABCDE34F5GH6*/elasticmapreduce/. Lokasi Amazon S3 yang digunakan dalam contoh di bawah ini mencerminkan pola lokasi log default untuk cluster EMR yang dibuat oleh akun Amazon Web Services*123456789012*Wilayah*us-west-2*. Jika Anda menggunakan lokasi khusus, polanya adalah s3://DOC-EXAMPLE-BUCKET/ *clusterId*.

Untuk informasi tentang cara membuat tabel dipartisi untuk berpotensi meningkatkan performa permintaan dan mengurangi transfer data, lihatMembuat dan menanyakan tabel yang dipartisi berdasarkan log EMR Amazon.

```
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'
```

Contoh kueri berikut dapat dijalankan pada`myemrlogs`tabel yang dibuat oleh contoh sebelumnya.

Example — Log langkah kueri untuk terjadinya ERROR, WARN, INFO, EXCEPTION, FATAL, atau 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 — Kueri log contoh tertentu, i-00b3c0a839ece0a9c, untuk ERROR, WARN, INFO, EXCEPTION, FATAL, atau 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 — Kueri log aplikasi presto untuk ERROR, WARN, INFO, EXCEPTION, FATAL, atau 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 — Log aplikasi Query Namenode untuk ERROR, WARN, INFO, EXCEPTION, FATAL, atau 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 — Kueri semua log berdasarkan tanggal dan jam untuk ERROR, WARN, INFO, EXCEPTION, FATAL, atau 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;
```

Membuat dan menanyakan tabel yang dipartisi berdasarkan log EMR Amazon

Contoh ini menggunakan lokasi log yang sama untuk membuat tabel Athena, tetapi tabel dipartisi, dan partisi kemudian dibuat untuk setiap lokasi log. Untuk informasi selengkapnya, lihat Partisi data di Athena.

Kueri berikut membuat tabel dipartisi bernama`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'
```

Pernyataan kueri berikut kemudian membuat partisi tabel berdasarkan sub-direktori untuk jenis log yang berbeda yang Amazon EMR menciptakan di Amazon S3:

```
ALTER TABLE mypartitionedemrlogs ADD
     PARTITION (logtype='containers')
     LOCATION 's3://aws-logs-123456789012-us-west-2/elasticmapreduce/j-2ABCDE34F5GH6/
containers/'
```

```
ALTER TABLE mypartitionedemrlogs ADD
     PARTITION (logtype='hadoop-mapreduce')
```

```
     LOCATION 's3://aws-logs-123456789012-us-west-2/elasticmapreduce/j-2ABCDE34F5GH6/
hadoop-mapreduce/'
```

```
ALTER TABLE mypartitionedemrlogs ADD
     PARTITION (logtype='hadoop-state-pusher')
     LOCATION 's3://aws-logs-123456789012-us-west-2/elasticmapreduce/j-2ABCDE34F5GH6/
hadoop-state-pusher/'
```

```
ALTER TABLE mypartitionedemrlogs ADD
     PARTITION (logtype='node')
     LOCATION 's3://aws-logs-123456789012-us-west-2/elasticmapreduce/j-2ABCDE34F5GH6/
node/'
```

```
ALTER TABLE mypartitionedemrlogs ADD
     PARTITION (logtype='steps')
     LOCATION 's3://aws-logs-123456789012-us-west-2/elasticmapreduce/j-2ABCDE34F5GH6/
steps/'
```

Setelah Anda membuat partisi, Anda dapat menjalankan`SHOW PARTITIONS`query di atas tabel untuk mengkonfirmasi:

```
SHOW PARTITIONS mypartitionedemrlogs;
```

Contoh berikut menunjukkan kueri untuk entri log tertentu menggunakan tabel dan partisi yang dibuat oleh contoh di atas.

Example — Meminta aplikasi application_1561661818238_0002 log di partisi kontainer untuk ERROR atau 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 — Menanyakan partisi Hadoop-mapReduce untuk pekerjaan job_1561661818238_0004 dan gagal mengurangi

```
SELECT data,
```

```
        "$PATH"
FROM "default"."mypartitionedemrlogs"
WHERE logtype='hadoop-mapreduce'
        AND regexp_like(data,'job_1561661818238_0004|Failed Reduces') limit 100;
```

Example — Meminta log Hive di partisi node untuk ID kueri 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 — Meminta log resourcemanager di partisi node untuk aplikasi 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
```

Memerikan log AWS Global Accelerator aliran

Anda dapat menggunakan AWS Global Accelerator untuk membuat akselerator yang mengarahkan lalu lintas jaringan ke titik akhir yang optimal melalui jaringan AWS global. Untuk informasi selengkapnya tentang Global Accelerator, lihat Apa itu AWS Global Accelerator.

Log aliran Global Accelerator memungkinkan Anda menangkap informasi tentang lalu lintas alamat IP yang pergi ke dan dari antarmuka jaringan di akselerator Anda. Data log alur diterbitkan ke Amazon S3, tempat Anda dapat mengambil dan melihat data Anda. Untuk informasi selengkapnya, lihat Flow log in AWS Global Accelerator.

Anda dapat menggunakan Athena untuk mengkueri log alur Global Accelerator Anda dengan membuat tabel yang menentukan lokasi mereka di Amazon S3.

Untuk membuat tabel untuk log alur Global Accelerator

1. Salin dan tempel pernyataan DDL berikut ke konsol Athena. Query ini menentukan ROW FORMAT DELIMITED dan menghilangkan menentukan SerDe, yang berarti bahwa query menggunakan. `LazySimpleSerDe` Dalam kueri ini, bidang diakhiri oleh spasi.

```
CREATE EXTERNAL TABLE IF NOT EXISTS aga_flow_logs (
  version string,
  account string,
  acceleratorid string,
  clientip string,
  clientport int,
  gip string,
  gipport int,
  endpointip string,
  endpointport int,
  protocol string,
  ipaddresstype string,
  numpackets bigint,
  numbytes int,
  starttime int,
  endtime int,
  action string,
  logstatus string,
  agasourceip string,
  agasourceport int,
  endpointregion string,
  agaregion string,
  direction string
)
PARTITIONED BY (dt string)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ' '
LOCATION 's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/account_id/globalaccelerator/
region/'
TBLPROPERTIES ("skip.header.line.count"="1");
```

2. MemodifikasiLOCATIONNilai untuk menunjuk ke bucket Amazon S3 yang berisi data log Anda.

```
's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/account_id/globalaccelerator/region_code/'
```

3. Jalankan kueri di konsol Athena. Setelah kueri selesai, Athena registeraga_flow_logstabel, membuat data di dalamnya tersedia untuk mengkueri.

4. Membuat partisi untuk membaca data, seperti dalam contoh kueri berikut. kueri menciptakan partisi tunggal untuk tanggal yang ditentukan. Ganti placeholder untuk tanggal dan lokasi.

```
ALTER TABLE aga_flow_logs
ADD PARTITION (dt='YYYY-MM-dd')
LOCATION 's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/account_id/
globalaccelerator/region_code/YYYY/MM/dd';
```

Contoh kueri untuk log AWS Global Accelerator aliran

Example — Daftar permintaan yang melewati lokasi tepi tertentu

Contoh kueri berikut mencantumkan permintaan yang melewati lokasi tepi LHR. MenggunakanLIMITOperator untuk membatasi jumlah log untuk mengkueri pada satu waktu.

```
SELECT
  clientip,
  agaregion,
  protocol,
  action
FROM
  aga_flow_logs
WHERE
  agaregion LIKE 'LHR%'
LIMIT
  100;
```

Example — Daftar alamat IP endpoint yang menerima paling banyak permintaan HTTPS

Untuk melihat alamat IP endpoint mana yang menerima jumlah tertinggi permintaan HTTPS, gunakan kueri berikut. Kueri ini menghitung jumlah paket yang diterima pada HTTPS port 443, kelompoknya berdasarkan alamat IP tujuan, dan mengembalikan 10 alamat IP teratas.

```
SELECT
  SUM(numpackets) AS packetcount,
  endpointip
FROM
  aga_flow_logs
WHERE
  endpointport = 443
GROUP BY
```

```
  endpointip
ORDER BY
  packetcount DESC
LIMIT
  10;
```

Menanyakan temuan Amazon GuardDuty

Amazon GuardDuty adalah layanan pemantauan keamanan untuk membantu mengidentifikasi aktivitas yang tidak terduga dan berpotensi tidak sah atau berbahaya di AWS lingkungan Anda. Saat mendeteksi aktivitas yang tidak terduga dan berpotensi berbahaya, GuardDuty hasilkan temuan keamanan yang dapat Anda ekspor ke Amazon S3 untuk penyimpanan dan analisis. Setelah Anda mengekspor temuan Anda ke Amazon S3, Anda dapat menggunakan Athena untuk mengkueri mereka. Artikel ini menunjukkan cara membuat tabel di Athena untuk GuardDuty temuan Anda dan menanyakannya.

Untuk informasi selengkapnya tentang Amazon GuardDuty, lihat Panduan GuardDuty Pengguna Amazon.

Prasyarat

- Aktifkan GuardDuty fitur untuk mengekspor temuan ke Amazon S3. Untuk langkahnya, lihat Mengekspor temuan di Panduan GuardDuty Pengguna Amazon.

Membuat tabel di Athena untuk temuan GuardDuty

Untuk menanyakan GuardDuty temuan Anda dari Athena, Anda harus membuat tabel untuk mereka.

Untuk membuat tabel di Athena untuk temuan GuardDuty

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Paste pernyataan DDL berikut ke konsol Athena. Ubah nilai `LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/account-id/GuardDuty/'` untuk menunjukkan GuardDuty temuan Anda di Amazon S3.

```
CREATE EXTERNAL TABLE `gd_logs` (
  `schemaversion` string,
  `accountid` string,
  `region` string,
  `partition` string,
```

```
  `id` string,
  `arn` string,
  `type` string,
  `resource` string,
  `service` string,
  `severity` string,
  `createdat` string,
  `updatedat` string,
  `title` string,
  `description` string)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
 LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/account-id/GuardDuty/'
 TBLPROPERTIES ('has_encrypted_data'='true')
```

> **Note**
>
> SerDe Mengharapkan setiap dokumen JSON berada pada satu baris teks tanpa karakter penghentian baris yang memisahkan bidang dalam catatan. Jika teks JSON dalam format cetak cantik, Anda mungkin menerima pesan kesalahan seperti HIVE_CURSOR_ERROR: Row is not a valid JSON Object or HIVE_CURSOR_ERROR:: Unexpected JsonParseException end-of-input: expected close marker for OBJECT saat Anda mencoba menanyakan tabel setelah Anda membuatnya. Untuk informasi selengkapnya, lihat File Data JSON di dokumentasi SerDe OpenX. GitHub

3. Menjalankan kueri di konsol Athena untuk mendaftar`gd_logs`tabel. Saat kueri selesai, temuan siap bagi Anda untuk mengkueri dari Athena.

Kueri contoh

Contoh berikut menunjukkan cara menanyakan GuardDuty temuan dari Athena.

Example — Ekskfiltrasi data DNS

Kueri berikut mengembalikan informasi tentang contoh Amazon EC2 yang mungkin exfiltrating data melalui permintaan DNS.

```
SELECT
    title,
    severity,
    type,
    id AS FindingID,
```

```
    accountid,
    region,
    createdat,
    updatedat,
    json_extract_scalar(service, '$.count') AS Count,
    json_extract_scalar(resource, '$.instancedetails.instanceid') AS InstanceID,
    json_extract_scalar(service, '$.action.actiontype') AS DNS_ActionType,
    json_extract_scalar(service, '$.action.dnsrequestaction.domain') AS DomainName,
    json_extract_scalar(service, '$.action.dnsrequestaction.protocol') AS protocol,
    json_extract_scalar(service, '$.action.dnsrequestaction.blocked') AS blocked
FROM gd_logs
WHERE type = 'Trojan:EC2/DNSDataExfiltration'
ORDER BY severity DESC
```

Example — Akses pengguna IAM tidak sah

Kueri berikut mengembalikan semua`UnauthorizedAccess:IAMUser`menemukan jenis untuk IAM Principal dari semua wilayah.

```
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;
```

Kiat untuk menanyakan temuan GuardDuty

Saat Anda membuat kueri Anda, ingatlah hal berikut.

- Untuk mengekstraksi data dari bidang JSON Nest, gunakan Prestojson_extractataujson_extract_scalarfungsi. Untuk informasi selengkapnya, lihat Mengekstrak data JSON dari string.

- Pastikan bahwa semua karakter di bidang JSON dalam huruf kecil.
- Untuk informasi tentang cara mengunduh hasil kueri, lihat Mengunduh file hasil kueri menggunakan konsol Athena.

Memeriksa log AWS Network Firewall

AWS Network Firewall adalah layanan terkelola yang dapat Anda gunakan untuk menerapkan perlindungan jaringan penting untuk instans Amazon Virtual Private Cloud Anda. AWS Network Firewall bekerja sama dengan AWS Firewall Manager sehingga Anda dapat membuat kebijakan berdasarkan AWS Network Firewall aturan dan kemudian menerapkan kebijakan tersebut secara terpusat di seluruh VPC dan akun Anda. Untuk informasi lebih lanjut tentang AWS Network Firewall, lihat AWS Network Firewall.

Anda dapat mengonfigurasi AWS Network Firewall pencatatan untuk lalu lintas yang Anda teruskan ke mesin aturan stateful firewall Anda. Logging memberikan informasi rinci tentang lalu lintas jaringan, termasuk waktu yang mesin stateful menerima paket, informasi rinci tentang paket, dan setiap tindakan aturan stateful diambil terhadap paket. Log diterbitkan ke tujuan log yang telah dikonfigurasi, tempat Anda dapat mengambil dan melihatnya. Untuk informasi selengkapnya, lihat.Log lalu lintas jaringan dari AWS Network FirewalldiAWS Network Firewall Panduan Developer.

Buat tabel untuk log peringatan

1. Ubah contoh pernyataan DDL berikut agar sesuai dengan struktur log peringatan Anda. Anda mungkin perlu memperbarui pernyataan untuk menyertakan kolom untuk versi terbaru dari log. Untuk informasi selengkapnya, lihat.Isi dari log firewalldiAWS Network Firewall Panduan Developer.

```
CREATE EXTERNAL TABLE network_firewall_alert_logs (
  firewall_name string,
  availability_zone string,
  event_timestamp string,
  event struct<
    timestamp:string,
    flow_id:bigint,
    event_type:string,
    src_ip:string,
    src_port:int,
    dest_ip:string,
    dest_port:int,
    proto:string,
```

```
    app_proto:string,
    tls_inspected:boolean,
    alert:struct<
      alert_id:string,
      alert_type:string,
      action:string,
      signature_id:int,
      rev:int,
      signature:string,
      category:string,
      severity:int,
      rule_name:string,
      alert_name:string,
      alert_severity:string,
      alert_description:string,
      file_name:string,
      file_hash:string,
      packet_capture:string,
      reference_links:array<string>
    >,
    src_country:string,
    dest_country:string,
    src_hostname:string,
    dest_hostname:string,
    user_agent:string,
    url:string
   >
)
 ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
 LOCATION 's3://DOC-EXAMPLE-BUCKET/path_to_alert_logs_folder/';
```

2. Ubah `LOCATION` klausa untuk menentukan folder log Anda di Amazon S3.
3. Jalankan `CREATE TABLE` kueri Anda di editor kueri Athena. Setelah kueri selesai, Athena mendaftarkan tabel, membuat data `network_firewall_alert_logs` yang ditunjuknya siap untuk kueri.

Kueri contoh log peringatan

Contoh kueri log peringatan di bagian ini menyaring peristiwa di mana inspeksi TLS dilakukan yang memiliki peringatan dengan tingkat keparahan 2 atau lebih tinggi.

Kueri menggunakan alias untuk membuat judul kolom keluaran yang menunjukkan `struct` bahwa kolom milik. Misalnya, judul kolom untuk `event.alert.category` bidang `event_alert_category` bukan hanya`category`. Untuk menyesuaikan nama kolom lebih lanjut, Anda dapat memodifikasi alias agar sesuai dengan preferensi Anda. Misalnya, Anda dapat menggunakan garis bawah atau pemisah lain untuk membatasi nama dan `struct` nama bidang.

Ingatlah untuk memodifikasi nama kolom dan `struct` referensi berdasarkan definisi tabel Anda dan pada bidang yang Anda inginkan dalam hasil kueri.

```
SELECT
  firewall_name,
  availability_zone,
  event_timestamp,
  event.timestamp AS event_timestamp,
  event.flow_id AS event_flow_id,
  event.event_type AS event_type,
  event.src_ip AS event_src_ip,
  event.src_port AS event_src_port,
  event.dest_ip AS event_dest_ip,
  event.dest_port AS event_dest_port,
  event.proto AS event_protol,
  event.app_proto AS event_app_proto,
  event.tls_inspected AS event_tls_inspected,
  event.alert.alert_id AS event_alert_alert_id,
  event.alert.alert_type AS event_alert_alert_type,
  event.alert.action AS event_alert_action,
  event.alert.signature_id AS event_alert_signature_id,
  event.alert.rev AS event_alert_rev,
  event.alert.signature AS event_alert_signature,
  event.alert.category AS event_alert_category,
  event.alert.severity AS event_alert_severity,
  event.alert.rule_name AS event_alert_rule_name,
  event.alert.alert_name AS event_alert_alert_name,
  event.alert.alert_severity AS event_alert_alert_severity,
  event.alert.alert_description AS event_alert_alert_description,
  event.alert.file_name AS event_alert_file_name,
  event.alert.file_hash AS event_alert_file_hash,
  event.alert.packet_capture AS event_alert_packet_capture,
  event.alert.reference_links AS event_alert_reference_links,
  event.src_country AS event_src_country,
  event.dest_country AS event_dest_country,
  event.src_hostname AS event_src_hostname,
  event.dest_hostname AS event_dest_hostname,
```

```
  event.user_agent AS event_user_agent,
  event.url AS event_url
FROM
  network_firewall_alert_logs
WHERE
  event.alert.severity >= 2
  AND event.tls_inspected = true
LIMIT 10;
```

Buat tabel untuk log netflow

1. Ubah contoh pernyataan DDL berikut agar sesuai dengan struktur log netflow Anda. Anda mungkin perlu memperbarui pernyataan untuk menyertakan kolom untuk versi terbaru dari log. Untuk informasi selengkapnya, lihat.Isi dari log firewalldiAWS Network Firewall Panduan Developer.

   ```
   CREATE EXTERNAL TABLE network_firewall_netflow_logs (
     firewall_name string,
     availability_zone string,
     event_timestamp string,
     event struct<
       timestamp:string,
       flow_id:bigint,
       event_type:string,
       src_ip:string,
       src_port:int,
       dest_ip:string,
       dest_port:int,
       proto:string,
       app_proto:string,
       netflow:struct<
         pkts:int,
         bytes:bigint,
         start:string,
         `end`:string,
         age:int,
         min_ttl:int,
         max_ttl:int,
         tcp_flags:struct<
           syn:boolean,
           fin:boolean,
           rst:boolean,
           psh:boolean,
   ```

```
        ack:boolean,
        urg:boolean
      >,
      tls_inspected:boolean
    >
  >
)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 's3://DOC-EXAMPLE-BUCKET/path_to_netflow_logs_folder/';
```

2. Ubah `LOCATION` klausa untuk menentukan folder log Anda di Amazon S3.
3. Jalankan `CREATE TABLE` kueri di editor kueri Athena. Setelah kueri selesai, Athena mendaftarkan tabel, membuat data `network_firewall_netflow_logs` yang ditunjuknya siap untuk kueri.

Kueri sampel log Netflow

Contoh kueri log netflow di bagian ini menyaring peristiwa di mana inspeksi TLS dilakukan.

Kueri menggunakan alias untuk membuat judul kolom keluaran yang menunjukkan `struct` bahwa kolom milik. Misalnya, judul kolom untuk `event.netflow.bytes` bidang `event_netflow_bytes` bukan hanya`bytes`. Untuk menyesuaikan nama kolom lebih lanjut, Anda dapat memodifikasi alias agar sesuai dengan preferensi Anda. Misalnya, Anda dapat menggunakan garis bawah atau pemisah lain untuk membatasi nama dan `struct` nama bidang.

Ingatlah untuk memodifikasi nama kolom dan `struct` referensi berdasarkan definisi tabel Anda dan pada bidang yang Anda inginkan dalam hasil kueri.

```
SELECT
  event.src_ip AS event_src_ip,
  event.dest_ip AS event_dest_ip,
  event.proto AS event_proto,
  event.app_proto AS event_app_proto,
  event.netflow.pkts AS event_netflow_pkts,
  event.netflow.bytes AS event_netflow_bytes,
  event.netflow.tcp_flags.syn AS event_netflow_tcp_flags_syn,
  event.netflow.tls_inspected AS event_netflow_tls_inspected
FROM network_firewall_netflow_logs
WHERE event.netflow.tls_inspected = true
```

Meminta log Network Load Balancer

Gunakan Athena untuk menganalisis dan memproses log dari Network Load Balancer. Log ini menerima informasi rinci tentang permintaan Transport Layer Security (TLS) yang dikirim ke Network Load Balancer. Anda dapat menggunakan log akses ini untuk menganalisis pola lalu lintas dan memecahkan masalah.

Sebelum Anda menganalisis log akses Network Load Balancer, aktifkan dan konfigurasikan untuk disimpan di bucket Amazon S3 tujuan. Untuk informasi selengkapnya, dan untuk informasi tentang setiap entri log akses Network Load Balancer, lihat Log akses untuk Network Load Balancer Anda.

- Buat tabel untuk log Network Load Balancer
- Contoh query Network Load Balancer

Untuk membuat tabel untuk log Network Load Balancer Anda

1. Salin dan tempel pernyataan DDL berikut ke konsol Athena. Periksasintaksdari catatan log Network Load Balancer. Anda mungkin perlu memperbarui kueri berikut untuk menyertakan kolom dan sintaks Regex untuk versi terbaru dari catatan.

```
CREATE EXTERNAL TABLE IF NOT EXISTS nlb_tls_logs (
            type string,
            version string,
            time string,
            elb string,
            listener_id string,
            client_ip string,
            client_port int,
            target_ip string,
            target_port int,
            tcp_connection_time_ms double,
            tls_handshake_time_ms double,
            received_bytes bigint,
            sent_bytes bigint,
            incoming_tls_alert int,
            cert_arn string,
            certificate_serial string,
            tls_cipher_suite string,
            tls_protocol_version string,
            tls_named_group string,
            domain_name string,
```

```
            alpn_fe_protocol string,
            alpn_be_protocol string,
            alpn_client_preference_list string,
            tls_connection_creation_time string
            )
            ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
            WITH SERDEPROPERTIES (
            'serialization.format' = '1',
            'input.regex' =
        '([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*):([0-9]*) ([^ ]*):([0-9]*)
 ([-.0-9]*) ([-.0-9]*) ([-0-9]*) ([-0-9]*) ([-0-9]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)$')
            LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/AWS_account_ID/
elasticloadbalancing/region';
```

2. MemodifikasiLOCATIONBucket Amazon S3 untuk menentukan tujuan log Network Load Balancer Anda.
3. Jalankan kueri di konsol Athena. Setelah kueri selesai, Athena registernlb_tls_logstabel, membuat data di dalamnya siap untuk mengkueri.

Contoh query Network Load Balancer

Untuk melihat berapa kali sertifikat yang digunakan, gunakan kueri yang mirip dengan contoh ini:

```
SELECT count(*) AS
         ct,
         cert_arn
FROM "nlb_tls_logs"
GROUP BY  cert_arn;
```

Kueri berikut menunjukkan berapa banyak pengguna yang menggunakan versi TLS lebih awal dari 1.3:

```
SELECT tls_protocol_version,
         COUNT(tls_protocol_version) AS
         num_connections,
         client_ip
FROM "nlb_tls_logs"
WHERE tls_protocol_version < 'tlsv13'
GROUP BY tls_protocol_version, client_ip;
```

Gunakan kueri berikut untuk mengidentifikasi koneksi yang mengambil waktu lama TLS jabat tangan:

```
SELECT *
FROM "nlb_tls_logs"
ORDER BY  tls_handshake_time_ms DESC
LIMIT 10;
```

Gunakan kueri berikut untuk mengidentifikasi dan menghitung versi protokol TLS dan cipher suite mana yang telah dinegosiasikan dalam 30 hari terakhir.

```
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;
```

Menanyakan log kueri penyelesai Amazon Route 53

Anda dapat membuat tabel Athena untuk Amazon Route 53 Resolver permintaan log dan kueri mereka dari Athena.

Route 53 penyelesai permintaan penebangan untuk penebangan permintaan DNS yang dibuat oleh sumber daya dalam VPC, lokal sumber daya yang menggunakan inbound resolver endpoint, permintaan yang menggunakan akhir Resolver keluar untuk rekursif DNS resolusi, dan permintaan yang menggunakan Route 53 Resolver DNS firewall aturan untuk memblokir, memungkinkan, atau memantau daftar domain. Untuk informasi selengkapnya, lihat Mencatat Log Kueri DNS dalam Panduan Developer Amazon Route 53. Untuk informasi tentang masing-masing bidang dalam log, lihat Nilai yang muncul di log kueri resolver di Panduan Pengembang Amazon Route 53.

Membuat tabel untuk log kueri resolver

Anda dapat menggunakan Editor Kueri di konsol Athena untuk membuat dan kueri tabel untuk log permintaan Route 53 penyelesai Anda.

Untuk membuat dan menanyakan tabel Athena untuk log kueri resolver Route 53

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.

2. Dalam Editor Kueri Athena, masukkan yang berikut`CREATE TABLE`. Ganti`LOCATION`dengan yang sesuai dengan lokasi log Resolver Anda di Amazon S3.

```
CREATE EXTERNAL TABLE r53_rlogs (
  version string,
  account_id string,
  region string,
  vpc_id string,
  query_timestamp string,
  query_name string,
  query_type string,
  query_class
    string,
  rcode string,
  answers array<
    struct<
      Rdata: string,
      Type: string,
      Class: string>
    >,
  srcaddr string,
  srcport int,
  transport string,
  srcids struct<
    instance: string,
    resolver_endpoint: string
    >,
  firewall_rule_action string,
  firewall_rule_group_id string,
  firewall_domain_list_id string
 )

ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/aws_account_id/vpcdnsquerylogs/{vpc-id}/'
```

Karena data log kueri Resolver dalam format JSON, pernyataan CREATE TABLE menggunakan SerDepustaka JSON untuk menganalisis data.

> **Note**
>
> SerDe Mengharapkan setiap dokumen JSON berada pada satu baris teks tanpa karakter penghentian baris yang memisahkan bidang dalam catatan. Jika teks

JSON dalam format cetak cantik, Anda mungkin menerima pesan kesalahan seperti HIVE_CURSOR_ERROR: Row is not a valid JSON Object or HIVE_CURSOR_ERROR:: Unexpected JsonParseException end-of-input: expected close marker for OBJECT saat Anda mencoba menanyakan tabel setelah Anda membuatnya. Untuk informasi selengkapnya, lihat File Data JSON dalam dokumentasi SerDe OpenX pada. GitHub

3. Pilih Run query (Jalankan kueri). Pernyataan itu menciptakan tabel Athena bernama`r53_rlogs`kolom yang mewakili masing-masing bidang dalam data log Resolver Anda.
4. Di Athena konsol Kueri Editor, jalankan kueri berikut untuk memverifikasi bahwa tabel Anda telah dibuat.

```
SELECT * FROM "r53_rlogs" LIMIT 10
```

Contoh partisi

Contoh berikut menunjukkan `CREATE TABLE` pernyataan untuk log query Resolver yang menggunakan proyeksi partisi dan dipartisi oleh vpc dan tanggal. Untuk informasi selengkapnya tentang proyek partisi, lihat Proyeksi partisi dengan 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}/'
)
```

Kueri contoh

Contoh berikut menunjukkan beberapa pertanyaan yang dapat Anda lakukan dari Athena pada log permintaan Resolver Anda.

Contoh 1 - log kueri dalam urutan query_timestamp menurun

Kueri berikut menampilkan hasil log dalam turun`query_timestamp`Agar.

```
SELECT * FROM "r53_rlogs"
ORDER BY query_timestamp DESC
```

Contoh 2 - log kueri dalam waktu mulai dan akhir yang ditentukan

Kueri kueri berikut log antara tengah malam dan 8 pagi pada 24 September 2020. Gantikan waktu awal dan akhir sesuai dengan kebutuhan Anda sendiri.

```
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
```

Contoh 3 - log kueri berdasarkan pola nama kueri DNS tertentu

Kueri berikut memilih catatan yang nama kueri termasuk string “example.com”.

```
SELECT query_timestamp, srcids.instance, srcaddr, srcport, query_name, rcode, answers
FROM "r53_rlogs"
WHERE query_name LIKE '%example.com%'
ORDER BY query_timestamp DESC
```

Contoh 4 - permintaan log kueri tanpa jawaban

Kueri berikut memilih entri log tempat permintaan tidak menerima jawaban.

```
SELECT query_timestamp, srcids.instance, srcaddr, srcport, query_name, rcode, answers
FROM "r53_rlogs"
WHERE cardinality(answers) = 0
```

Contoh 5 - log kueri dengan jawaban tertentu

Kueri berikut menunjukkan log tempat`answer.Rdata`nilai memiliki alamat IP yang ditentukan.

```
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';
```

Menanyakan log peristiwa Amazon SES

Anda dapat menggunakan Amazon Athena untuk menanyakan log peristiwa Amazon Simple Email Service (Amazon SES).

Amazon SES adalah platform email yang menyediakan cara yang nyaman dan hemat biaya untuk mengirim dan menerima email menggunakan alamat email dan domain Anda sendiri. Anda dapat memantau aktivitas pengiriman Amazon SES Anda pada tingkat terperinci menggunakan peristiwa, metrik, dan statistik.

Berdasarkan karakteristik yang Anda tentukan, Anda dapat mempublikasikan acara Amazon SES ke Amazon CloudWatch, Amazon Data Firehose, atau Amazon Simple Notification Service. Setelah informasi disimpan di Amazon S3, Anda dapat menanyakannya dari Amazon Athena.

Misalnya `CREATE TABLE` pernyataan Athena untuk log Amazon SES, termasuk langkah-langkah tentang cara membuat tampilan dan meratakan array bersarang di data log peristiwa Amazon SES, lihat “Langkah 3: Menggunakan Amazon Athena untuk menanyakan log peristiwa SES” di AWS posting blog Menganalisis data peristiwa Amazon SES dengan Layanan Analytics. AWS

Menanyakan log aliran VPC Amazon

Log aliran Amazon Virtual Private Cloud menangkap informasi tentang lalu lintas IP yang pergi ke dan dari antarmuka jaringan di VPC. Gunakan log untuk menyelidiki pola lalu lintas jaringan dan mengidentifikasi ancaman dan risiko di seluruh jaringan VPC Anda.

Untuk menanyakan log aliran VPC Amazon Anda, Anda memiliki dua opsi:

- Konsol VPC Amazon — Gunakan fitur integrasi Athena di Konsol VPC Amazon untuk membuat template AWS CloudFormation yang membuat database Athena, workgroup, dan tabel flow log dengan partisi untuk Anda. Template ini juga membuat serangkaian kueri log alur yang telah ditentukan sebelumnya yang dapat Anda gunakan untuk mendapatkan wawasan tentang lalu lintas yang mengalir melalui VPC Anda.

  Untuk informasi tentang pendekatan ini, lihat Log alur kueri menggunakan Amazon Athena di Panduan Pengguna Amazon VPC.
- Konsol Amazon Athena — Buat tabel dan kueri langsung di konsol Athena. Untuk informasi lebih lanjut, lanjutkan membaca halaman ini.

Membuat dan menanyakan tabel untuk log aliran VPC kustom

Sebelum Anda mulai mengkueri log di Athena,Aktifkan log alur VPC, dan konfigurasikan untuk disimpan ke bucket Amazon S3. Setelah Anda membuat log, biarkan mereka berjalan selama beberapa menit untuk mengumpulkan beberapa data. Log dibuat dalam format kompresi GZIP yang Athena memungkinkan Anda kueri secara langsung.

Saat Anda membuat log aliran VPC, Anda dapat menggunakan format khusus saat Anda ingin menentukan bidang yang akan dikembalikan dalam log aliran dan urutan bidang yang muncul. Untuk informasi selengkapnya tentang catatan log alur, lihat Catatan log aliran di Panduan Pengguna Amazon VPC.

Pertimbangan umum

Saat Anda membuat tabel di Athena untuk log aliran VPC Amazon, ingat poin-poin berikut:

- Secara default, di Athena, Parket akan mengakses kolom berdasarkan nama. Untuk informasi selengkapnya, lihat Menangani pembaruan skema.
- Gunakan nama dalam catatan log alur untuk nama kolom di Athena. Nama-nama kolom dalam skema Athena harus sama persis dengan nama bidang di log aliran VPC Amazon, dengan perbedaan berikut:
  - Ganti tanda hubung di nama bidang log Amazon VPC dengan garis bawah di nama kolom Athena. Di Athena, satu-satunya karakter yang dapat diterima untuk nama database, nama tabel, dan nama kolom adalah huruf kecil, angka, dan karakter garis bawah. Untuk informasi selengkapnya, lihat Nama database, tabel, dan kolom.
  - Lepaskan nama catatan log alur yang merupakan kata kunci yang dicadangkan di Athena dengan melampirkannya dengan backticks.
- Log aliran VPC bersifat Akun AWS spesifik. Saat Anda memublikasikan file log Anda ke Amazon S3, jalur yang dibuat Amazon VPC di Amazon S3 menyertakan ID yang digunakan untuk Akun AWS membuat log aliran. Untuk informasi selengkapnya, lihat Menerbitkan log alur ke Amazon S3 di Panduan Pengguna Amazon VPC.

Pernyataan CREATE TABLE untuk log aliran Amazon VPC

Prosedur berikut membuat tabel VPC Amazon untuk log aliran VPC Amazon. Saat Anda membuat log alur dengan format kustom, Anda membuat tabel dengan bidang yang cocok dengan bidang yang Anda tentukan saat membuat log alur dalam urutan yang sama dengan yang Anda tentukan.

Untuk membuat tabel Athena untuk log aliran VPC Amazon

1. Masukkan pernyataan DDL seperti berikut ini ke editor kueri konsol Athena, mengikuti pedoman di Pertimbangan umum bagian ini. Pernyataan sampel membuat tabel yang memiliki kolom untuk log aliran VPC Amazon versi 2 hingga 5 seperti yang didokumentasikan dalam catatan log Flow. Jika Anda menggunakan kumpulan kolom atau urutan kolom yang berbeda, ubah pernyataan yang sesuai.

```
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");
```

Perhatikan bidang berikut:

- Query menentukan `ROW FORMAT DELIMITED` dan menghilangkan menentukan. SerDe Ini berarti bahwa kueri menggunakanLazySimpleSerDe untuk CSV, TSV, dan file yang dibatasi khusus. Dalam kueri ini, bidang diakhiri oleh spasi.
- ParameterPARTITIONED BYKlausul menggunakan`date`Jenis. Ini memungkinkan untuk menggunakan operator matematika dalam kueri untuk memilih apa yang lebih tua atau lebih baru dari tanggal tertentu.

> **Note**
>
> Karena `date` merupakan kata kunci yang dicadangkan dalam pernyataan DDL, itu lolos oleh karakter centang kembali. Untuk informasi selengkapnya, lihat Kata kunci terpesan.

- Untuk log aliran VPC dengan format kustom yang berbeda, ubah bidang agar sesuai dengan bidang yang Anda tentukan saat membuat log alur.

2. MemodifikasiLOCATION `'s3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/{account_id}/vpcflowlogs/{region_code}/'`Untuk menunjuk ke bucket Amazon S3 yang berisi data log Anda.
3. Jalankan kueri di konsol Athena. Setelah kueri selesai, Athena register`vpc_flow_logs`tabel, membuat data di dalamnya siap bagi Anda untuk mengeluarkan kueri.
4. Membuat partisi untuk dapat membaca data, seperti dalam contoh kueri berikut. Kueri ini menciptakan partisi tunggal untuk tanggal yang ditentukan. Ganti placeholder untuk tanggal dan lokasi yang diperlukan.

> **Note**
>
> Kueri ini menciptakan satu partisi saja, untuk tanggal yang Anda tentukan. Untuk mengotomatisasi proses, gunakan skrip yang menjalankan query ini dan membuat partisi dengan cara ini untuk`year/month/day`, atau gunakan `CREATE TABLE` pernyataan yang menentukan proyeksi partisi.

```
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';
```

Contoh kueri untuk tabel vpc_flow_logs

Gunakan editor kueri di konsol Athena untuk menjalankan pernyataan SQL pada tabel yang Anda buat. Anda dapat menyimpan kueri, melihat kueri sebelumnya, atau mengunduh hasil kueri dalam format CSV. Dalam contoh berikut, ganti `vpc_flow_logs` dengan nama tabel Anda. Ubah nilai kolom dan variabel lain sesuai dengan kebutuhan Anda.

Contoh kueri berikut mencantumkan maksimum 100 log alur untuk tanggal yang ditentukan.

```
SELECT *
FROM vpc_flow_logs
WHERE date = DATE('2020-05-04')
LIMIT 100;
```

Kueri berikut mencantumkan semua koneksi TCP ditolak dan menggunakan kolom partisi tanggal yang baru dibuat,`date`, untuk mengekstraknya dari hari dalam seminggu dimana peristiwa-peristiwa ini terjadi.

```
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;
```

Untuk melihat server mana yang menerima permintaan HTTPS dengan jumlah tertinggi, gunakan kueri berikut. Ini menghitung jumlah paket yang diterima pada HTTPS port 443, grup mereka berdasarkan alamat IP tujuan, dan mengembalikan 10 besar dari minggu lalu.

```
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;
```

Membuat tabel untuk log aliran dalam format Apache Parquet

Prosedur berikut membuat tabel VPC Amazon untuk log aliran Amazon VPC dalam format Apache Parquet.

Untuk membuat tabel Athena untuk log aliran Amazon VPC dalam format Parket

1. Masukkan pernyataan DDL seperti berikut ini ke editor kueri konsol Athena, mengikuti pedoman di Pertimbangan umum bagian ini. Pernyataan sampel membuat tabel yang memiliki kolom untuk log aliran VPC Amazon versi 2 hingga 5 seperti yang didokumentasikan dalam catatan log Aliran dalam format Parket, Hive dipartisi setiap jam. Jika Anda tidak memiliki partisi per jam, hapus `hour` dari klausa. `PARTITIONED BY`

   ```
   CREATE EXTERNAL TABLE IF NOT EXISTS vpc_flow_logs_parquet (
     version int,
     account_id string,
     interface_id string,
     srcaddr string,
     dstaddr string,
     srcport int,
     dstport int,
     protocol bigint,
     packets bigint,
     bytes bigint,
     start bigint,
     `end` bigint,
     action string,
     log_status string,
     vpc_id string,
     subnet_id string,
     instance_id string,
     tcp_flags int,
     type string,
   ```

```
  pkt_srcaddr string,
  pkt_dstaddr string,
  region string,
  az_id string,
  sublocation_type string,
  sublocation_id string,
  pkt_src_aws_service string,
  pkt_dst_aws_service string,
  flow_direction string,
  traffic_path int
)
PARTITIONED BY (
  `aws-account-id` string,
  `aws-service` string,
  `aws-region` string,
  `year` string,
  `month` string,
  `day` string,
  `hour` string
)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/'
TBLPROPERTIES (
  'EXTERNAL'='true',
  'skip.header.line.count'='1'
  )
```

2. Ubah sampel `LOCATION 's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/'` untuk menunjuk ke jalur Amazon S3 yang berisi data log Anda.
3. Jalankan kueri di konsol Athena.
4. Jika data Anda dalam format yang kompatibel dengan HIVE, jalankan perintah berikut di konsol Athena untuk memperbarui dan memuat partisi Hive di metastore. Setelah kueri selesai, Anda dapat menanyakan data dalam `vpc_flow_logs_parquet` tabel.

```
MSCK REPAIR TABLE vpc_flow_logs_parquet
```

Jika Anda tidak menggunakan data yang kompatibel dengan Hive, jalankan ALTER TABLE ADD PARTITION untuk memuat partisi.

Untuk informasi selengkapnya tentang penggunaan Athena untuk menanyakan log aliran VPC Amazon dalam format Parket, lihat posting Optimalkan kinerja dan kurangi biaya untuk analitik jaringan dengan Log Aliran VPC dalam format Parket Apache di Blog Big Data.AWS

Membuat dan menanyakan tabel untuk log aliran VPC Amazon menggunakan proyeksi partisi

Gunakan `CREATE TABLE` pernyataan seperti berikut ini untuk membuat tabel, partisi tabel, dan mengisi partisi secara otomatis dengan menggunakan proyeksi partisi. Ganti nama tabel `test_table_vpclogs` dalam contoh dengan nama tabel Anda. Edit `LOCATION` klausa untuk menentukan bucket Amazon S3 yang berisi data log VPC Amazon Anda.

`CREATE TABLE`Pernyataan berikut adalah untuk log aliran VPC yang dikirimkan dalam format partisi gaya non-HIVE. Contoh ini memungkinkan agregasi multi-akun. Jika Anda memusatkan log VPC Flow dari beberapa akun ke dalam satu bucket Amazon S3, ID akun harus dimasukkan di jalur Amazon S3.

```
CREATE EXTERNAL TABLE IF NOT EXISTS test_table_vpclogs (
  version int,
  account_id string,
  interface_id string,
  srcaddr string,
  dstaddr string,
  srcport int,
  dstport int,
  protocol bigint,
  packets bigint,
  bytes bigint,
  start bigint,
  `end` bigint,
  action string,
  log_status string,
  vpc_id string,
  subnet_id string,
  instance_id string,
  tcp_flags int,
  type string,
  pkt_srcaddr string,
```

```
  pkt_dstaddr string,
  az_id string,
  sublocation_type string,
  sublocation_id string,
  pkt_src_aws_service string,
  pkt_dst_aws_service string,
  flow_direction string,
  traffic_path int
)
PARTITIONED BY (accid string, region string, day string)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ' '
LOCATION '$LOCATION_OF_LOGS'
TBLPROPERTIES
(
"skip.header.line.count"="1",
"projection.enabled" = "true",
"projection.accid.type" = "enum",
"projection.accid.values" = "$ACCID_1,$ACCID_2",
"projection.region.type" = "enum",
"projection.region.values" = "$REGION_1,$REGION_2,$REGION_3",
"projection.day.type" = "date",
"projection.day.range" = "$START_RANGE,NOW",
"projection.day.format" = "yyyy/MM/dd",
"storage.location.template" = "s3://DOC-EXAMPLE-BUCKET/AWSLogs/${accid}/vpcflowlogs/
${region}/${day}"
)
```

Contoh kueri untuk test_table_vpclogs

Contoh query berikut `test_table_vpclogs` dibuat oleh pernyataan sebelumnya`CREATE TABLE`. Ganti `test_table_vpclogs` dalam kueri dengan nama tabel Anda sendiri. Ubah nilai kolom dan variabel lain sesuai dengan kebutuhan Anda.

Untuk mengembalikan 100 entri log akses pertama dalam urutan kronologis untuk jangka waktu tertentu, jalankan kueri seperti berikut ini.

```
SELECT *
FROM test_table_vpclogs
WHERE day >= '2021/02/01' AND day < '2021/02/28'
ORDER BY day ASC
LIMIT 100
```

Untuk melihat server mana yang menerima sepuluh besar paket HTTP untuk jangka waktu tertentu, jalankan kueri seperti berikut ini. Kueri menghitung jumlah paket yang diterima pada port HTTPS 443, mengelompokkannya berdasarkan alamat IP tujuan, dan mengembalikan 10 entri teratas dari minggu sebelumnya.

```
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
```

Untuk mengembalikan log yang dibuat selama periode waktu tertentu, jalankan kueri seperti berikut ini.

```
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'
```

Untuk mengembalikan log akses untuk alamat IP sumber antara periode waktu tertentu, jalankan kueri seperti berikut ini.

```
SELECT *
FROM test_table_vpclogs
WHERE srcaddr = '10.117.1.22'
  AND day >= '2021/02/01'
  AND day < '2021/02/28'
```

Untuk membuat daftar koneksi TCP yang ditolak, jalankan kueri seperti berikut ini.

```
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
```

Untuk mengembalikan log akses untuk rentang alamat IP yang dimulai dengan`10.117`, jalankan kueri seperti berikut ini.

```
SELECT *
FROM test_table_vpclogs
WHERE split_part(srcaddr,'.', 1)='10'
  AND split_part(srcaddr,'.', 2) ='117'
```

Untuk mengembalikan log akses untuk alamat IP tujuan antara rentang waktu tertentu, jalankan kueri seperti berikut ini.

```
SELECT *
FROM test_table_vpclogs
WHERE dstaddr = '10.0.1.14'
  AND day >= '2021/01/01'
  AND day < '2021/01/31'
```

Membuat tabel untuk log aliran dalam format Apache Parquet menggunakan proyeksi partisi

Pernyataan CREATE TABLE proyeksi partisi berikut untuk log aliran VPC dalam format Apache Parquet, tidak kompatibel dengan Hive, dan dipartisi berdasarkan jam dan tanggal, bukan hari. Ganti nama tabel `test_table_vpclogs_parquet` dalam contoh dengan nama tabel Anda. Edit `LOCATION` klausa untuk menentukan bucket Amazon S3 yang berisi data log VPC Amazon Anda.

```
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}"
)
```

Sumber daya tambahan

Untuk informasi selengkapnya tentang penggunaan Athena untuk menganalisis log aliran VPC, lihat posting blog AWS Big Data berikut:

- Analisis Log Aliran VPC dengan integrasi Amazon point-and-click Athena
- Menganalisis log aliran VPC menggunakan Amazon Athena dan Amazon QuickSight
- Optimalkan kinerja dan kurangi biaya untuk analitik jaringan dengan VPC Flow Logs dalam format Apache Parquet

Meminta log AWS WAF

AWS WAF adalah firewall aplikasi web yang memungkinkan Anda memantau dan mengontrol permintaan HTTP dan HTTPS yang diterima aplikasi web Anda yang dilindungi dari klien. Anda menentukan cara menangani permintaan web dengan mengonfigurasi aturan di dalam daftar kontrol akses AWS WAF web (ACL). Anda kemudian melindungi aplikasi web dengan mengaitkan ACL web ke dalamnya. Contoh sumber daya aplikasi web yang dapat Anda lindungi AWS WAF termasuk CloudFront distribusi Amazon, API REST Amazon API Gateway, dan Application Load Balancer. Untuk informasi selengkapnya AWS WAF, lihat AWS WAFdi panduan AWS WAF pengembang.

AWS WAF log mencakup informasi tentang lalu lintas yang dianalisis oleh ACL web Anda, seperti waktu yang AWS WAF menerima permintaan dari AWS sumber daya Anda, informasi rinci tentang permintaan, dan tindakan untuk aturan yang dicocokkan oleh setiap permintaan.

Anda dapat mengonfigurasi ACL AWS WAF web untuk mempublikasikan log ke salah satu dari beberapa tujuan, tempat Anda dapat menanyakan dan melihatnya. Untuk informasi selengkapnya tentang mengonfigurasi pencatatan ACL web dan konten AWS WAF log, lihat Mencatat lalu lintas ACL AWS WAF web di panduan pengembang.AWS WAF

Untuk contoh cara menggabungkan AWS WAF log ke dalam repositori danau data pusat dan menanyakannya dengan Athena, lihat posting Blog AWS Big Data Menganalisis AWS WAF log dengan Layanan, OpenSearch Amazon Athena, dan Amazon. QuickSight

Topik ini memberikan dua contoh `CREATE TABLE` pernyataan: satu yang menggunakan partisi dan satu yang tidak.

> **Note**
>
> `CREATE TABLE` Pernyataan dalam topik ini dapat digunakan untuk AWS WAF log v1 dan v2. Di v1, `webaclid` bidang berisi ID. Di v2, `webaclid` bidang berisi ARN lengkap. `CREATE TABLE` Pernyataan di sini memperlakukan konten ini secara agnostik dengan menggunakan tipe data. `string`

Topik

- Membuat tabel untuk log AWS WAF S3 di Athena menggunakan proyeksi partisi
- Membuat tabel untuk AWS WAF log tanpa partisi
- Contoh kueri untuk log AWS WAF

Membuat tabel untuk log AWS WAF S3 di Athena menggunakan proyeksi partisi

Karena AWS WAF log memiliki struktur yang diketahui yang skema partisi dapat Anda tentukan sebelumnya, Anda dapat mengurangi runtime kueri dan mengotomatiskan manajemen partisi dengan menggunakan fitur proyeksi partisi Athena. Proyeksi partisi secara otomatis menambahkan partisi baru saat data baru ditambahkan. Ini menghapus kebutuhan bagi Anda untuk menambahkan partisi secara manual dengan menggunakan`ALTER TABLE ADD PARTITION`.

`CREATE TABLE` Pernyataan contoh berikut secara otomatis menggunakan proyeksi partisi pada AWS WAF log dari tanggal tertentu hingga saat ini untuk empat AWS wilayah yang berbeda. `PARTITION BY` Klausul dalam contoh ini partisi berdasarkan wilayah dan tanggal, tetapi Anda dapat memodifikasi ini sesuai dengan kebutuhan Anda. Ubah bidang seperlunya agar sesuai dengan keluaran log Anda. *Di `storage.location.template` klausa `LOCATION` dan, ganti placeholder bucket dan accountID dengan nilai yang mengidentifikasi lokasi bucket Amazon S3 log Anda.* AWS WAF Untuk`projection.day.range`, ganti*2021*/*01*/*01*dengan tanggal mulai yang ingin Anda gunakan. Setelah Anda menjalankan kueri dengan sukses, Anda dapat meminta tabel. Anda tidak perlu menjalankan `ALTER TABLE ADD PARTITION` untuk memuat partisi.

```
CREATE EXTERNAL TABLE `waf_logs`(
  `timestamp` bigint,
  `formatversion` int,
  `webaclid` string,
  `terminatingruleid` string,
  `terminatingruletype` string,
  `action` string,
```

```
  `terminatingrulematchdetails` array <
                                    struct <
                                        conditiontype: string,
                                        sensitivitylevel: string,
                                        location: string,
                                        matcheddata: array < string >
                                          >
                                     >,
  `httpsourcename` string,
  `httpsourceid` string,
  `rulegrouplist` array <
                       struct <
                           rulegroupid: string,
                           terminatingrule: struct <
                                               ruleid: string,
                                               action: string,
                                               rulematchdetails: array <
                                                                    struct <
                                                                        conditiontype:
string,

sensitivitylevel: string,
                                                                        location:
string,
                                                                        matcheddata:
array < string >
                                                                           >
                                                                     >
                                                >,
                           nonterminatingmatchingrules: array <
                                                                struct <
                                                                    ruleid: string,
                                                                    action: string,
                                                                    overriddenaction:
string,
                                                                    rulematchdetails:
array <

struct <

   conditiontype: string,

   sensitivitylevel: string,
```

```
    location: string,

    matcheddata: array < string >

       >
                                                                    >,
                                                                challengeresponse:
 struct <

 responsecode: string,

 solvetimestamp: string
                                                                         >,
                                                                captcharesponse:
 struct <

 responsecode: string,

 solvetimestamp: string
                                                                         >
                                                                  >
                                                              >,
                         excludedrules: string
                           >
                      >,
`ratebasedrulelist` array <
                        struct <
                            ratebasedruleid: string,
                            limitkey: string,
                            maxrateallowed: int
                              >
                         >,
  `nonterminatingmatchingrules` array <
                                    struct <
                                        ruleid: string,
                                        action: string,
                                        rulematchdetails: array <
                                                              struct <
                                            conditiontype: string,
                                            sensitivitylevel:
 string,
                                            location: string,
```

```
                                                                matcheddata: array <
string >
                                                                    >
                                                                >,
                                                challengeresponse: struct <
                                                                responsecode: string,
                                                                solvetimestamp: string
                                                                >,
                                                captcharesponse: struct <
                                                                responsecode: string,
                                                                solvetimestamp: string
                                                                >
                                                  >
                                              >,
 `requestheadersinserted` array <
                                struct <
                                    name: string,
                                    value: string
                                      >
                                 >,
 `responsecodesent` string,
 `httprequest` struct <
                   clientip: string,
                   country: string,
                   headers: array <
                               struct <
                                   name: string,
                                   value: string
                                     >
                                >,
                   uri: string,
                   args: string,
                   httpversion: string,
                   httpmethod: string,
                   requestid: string
                     >,
 `labels` array <
              struct <
                  name: string
                    >
               >,
 `captcharesponse` struct <
                       responsecode: string,
                       solvetimestamp: string,
```

```
                        failureReason: string
                          >,
  `challengeresponse` struct <
                        responsecode: string,
                        solvetimestamp: string,
                        failureReason: string
                        >,
  `ja3Fingerprint` string,
  `oversizefields` string,
  `requestbodysize` int,
  `requestbodysizeinspectedbywaf` int
)
PARTITIONED BY (
`region` string,
`date` string)
ROW FORMAT SERDE
  'org.openx.data.jsonserde.JsonSerDe'
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/AWSLogs/accountID/WAFLogs/region/DOC-EXAMPLE-WEBACL/'
TBLPROPERTIES(
 'projection.enabled' = 'true',
 'projection.region.type' = 'enum',
 'projection.region.values' = 'us-east-1,us-west-2,eu-central-1,eu-west-1',
 'projection.date.type' = 'date',
 'projection.date.range' = '2021/01/01,NOW',
 'projection.date.format' = 'yyyy/MM/dd',
 'projection.date.interval' = '1',
 'projection.date.interval.unit' = 'DAYS',
 'storage.location.template' = 's3://DOC-EXAMPLE-BUCKET/AWSLogs/accountID/WAFLogs/
${region}/DOC-EXAMPLE-WEBACL/${date}/')
```

> **Note**
>
> Format jalur dalam `LOCATION` klausa dalam contoh adalah standar tetapi dapat bervariasi berdasarkan AWS WAF konfigurasi yang telah Anda terapkan. Misalnya, jalur AWS WAF log contoh berikut adalah untuk CloudFront distribusi:

```
s3://DOC-EXAMPLE-BUCKET/AWSLogs/12345678910/WAFLogs/cloudfront/
cloudfronyt/2022/08/08/17/55/
```

Jika Anda mengalami masalah saat membuat atau menanyakan tabel AWS WAF log Anda, konfirmasikan lokasi data log atau kontak AWS Support Anda.

Untuk informasi selengkapnya tentang proyek partisi, lihat Proyeksi partisi dengan Amazon Athena.

Membuat tabel untuk AWS WAF log tanpa partisi

Bagian ini menjelaskan cara membuat tabel untuk AWS WAF log tanpa partisi atau proyeksi partisi.

> **Note**
>
> Untuk alasan kinerja dan biaya, kami tidak menyarankan menggunakan skema non-partisi untuk kueri. Untuk informasi lebih lanjut, lihat 10 Tips Tuning Kinerja Terbaik untuk Amazon Athena di Blog Big AWS Data.

Untuk membuat AWS WAF tabel

1. Salin dan tempel pernyataan DDL berikut ke konsol Athena. Ubah bidang seperlunya agar sesuai dengan keluaran log Anda. `LOCATION`Ubah bucket Amazon S3 agar sesuai dengan bucket yang menyimpan log Anda.

   Kueri ini menggunakanOpenX JSON SerDe.

   > **Note**
   >
   > SerDe Mengharapkan setiap dokumen JSON berada pada satu baris teks tanpa karakter penghentian baris yang memisahkan bidang dalam catatan. Jika teks JSON dalam format cetak cantik, Anda mungkin menerima pesan kesalahan seperti HIVE_CURSOR_ERROR: Row is not a valid JSON Object or HIVE_CURSOR_ERROR:: Unexpected JsonParseException end-of-input: expected close marker for OBJECT saat Anda mencoba menanyakan tabel setelah Anda membuatnya. Untuk informasi selengkapnya, lihat File Data JSON di dokumentasi SerDe OpenX. GitHub

```
CREATE EXTERNAL TABLE `waf_logs`(
  `timestamp` bigint,
  `formatversion` int,
  `webaclid` string,
  `terminatingruleid` string,
  `terminatingruletype` string,
  `action` string,
  `terminatingrulematchdetails` array <
                                  struct <
                                      conditiontype: string,
                                      sensitivitylevel: string,
                                      location: string,
                                      matcheddata: array < string >
                                        >
                                   >,
  `httpsourcename` string,
  `httpsourceid` string,
  `rulegrouplist` array <
                      struct <
                          rulegroupid: string,
                          terminatingrule: struct <
                                           ruleid: string,
                                           action: string,
                                           rulematchdetails: array <
                                                               struct <
                                                                   conditiontype:
 string,

 sensitivitylevel: string,
                                                                   location:
 string,
                                                                   matcheddata:
 array < string >
                                                                      >
                                                              >
                                             >,
                          nonterminatingmatchingrules: array <
                                                         struct <
                                                             ruleid: string,
                                                             action: string,
                                                             overriddenaction:
 string,
```

```
                                                                    rulematchdetails:
array <

 struct <

     conditiontype: string,

     sensitivitylevel: string,

     location: string,

     matcheddata: array < string >

        >
                                                                     >,
                                                                    challengeresponse:
struct <

responsecode: string,

solvetimestamp: string
                                                                                >,
                                                                    captcharesponse:
struct <

responsecode: string,

solvetimestamp: string
                                                                                >
                                                                      >
                                                                >,
                        excludedrules: string
                          >
                     >,
`ratebasedrulelist` array <
                        struct <
                            ratebasedruleid: string,
                            limitkey: string,
                            maxrateallowed: int
                              >
                         >,
  `nonterminatingmatchingrules` array <
                                  struct <
                                      ruleid: string,
```

```
                                        action: string,
                                        rulematchdetails: array <
                                                            struct <
                                                                conditiontype:
string,
                                                                sensitivitylevel:
string,
                                                                location: string,
                                                                matcheddata: array <
string >
                                                                   >
                                                            >,
                                        challengeresponse: struct <
                                                            responsecode: string,
                                                            solvetimestamp: string
                                                            >,
                                        captcharesponse: struct <
                                                            responsecode: string,
                                                            solvetimestamp: string
                                                            >
                                          >
                                     >,
 `requestheadersinserted` array <
                                struct <
                                    name: string,
                                    value: string
                                      >
                                 >,
 `responsecodesent` string,
 `httprequest` struct <
                   clientip: string,
                   country: string,
                   headers: array <
                               struct <
                                   name: string,
                                   value: string
                                     >
                                >,
                   uri: string,
                   args: string,
                   httpversion: string,
                   httpmethod: string,
                   requestid: string
                     >,
```

```
  `labels` array <
              struct <
                  name: string
                    >
               >,
  `captcharesponse` struct <
                      responsecode: string,
                      solvetimestamp: string,
                      failureReason: string
                        >,
  `challengeresponse` struct <
                      responsecode: string,
                      solvetimestamp: string,
                      failureReason: string
                      >,
  `ja3Fingerprint` string,
  `oversizefields` string,
  `requestbodysize` int,
  `requestbodysizeinspectedbywaf` int
)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
STORED AS INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://DOC-EXAMPLE-BUCKET/prefix/'
```

2. Jalankan `CREATE EXTERNAL TABLE` pernyataan di editor kueri konsol Athena. Ini mendaftarkan`waf_logs`tabel dan membuat data di dalamnya tersedia untuk pertanyaan dari Athena.

Contoh kueri untuk log AWS WAF

Banyak contoh query berikut menggunakan tabel proyeksi partisi yang dibuat sebelumnya dalam dokumen ini. Ubah nama tabel, nilai kolom, dan variabel lain dalam contoh sesuai dengan kebutuhan Anda. Untuk meningkatkan performa permintaan Anda dan mengurangi biaya, menambahkan kolom partisi dalam syarat filter.

- 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

Bekerja dengan tanggal dan waktu

- 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

Bekerja dengan permintaan dan alamat yang diblokir

- 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 — Hitung jumlah perujuk yang berisi istilah tertentu

Kueri berikut menghitung jumlah perujuk yang berisi istilah “amazon” untuk rentang tanggal yang ditentukan.

```
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 — Hitung semua alamat IP yang cocok dalam 10 hari terakhir yang telah cocok dengan aturan yang dikecualikan

Kueri berikut menghitung jumlah kali dalam 10 hari terakhir bahwa alamat IP cocok aturan dikecualikan dalam grup aturan.

```
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 — Kelompokkan semua aturan terkelola yang dihitung dengan berapa kali dicocokkan

Jika Anda menetapkan tindakan aturan grup aturan ke Hitung dalam konfigurasi ACL web Anda sebelum 27 Oktober 2022, AWS WAF simpan penggantian Anda di web ACL JSON sebagai. `excludedRules` Sekarang, pengaturan JSON untuk mengganti aturan ke Count ada di `ruleActionOverrides` pengaturan. Untuk informasi selengkapnya, lihat Penggantian tindakan di grup aturan di Panduan AWS WAF Pengembang. Untuk mengekstrak aturan terkelola dalam mode Hitung dari struktur log baru, kueri `nonTerminatingMatchingRules` di `ruleGroupList` bagian alih-alih `excludedRules` bidang, seperti pada contoh berikut.

```
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 — Kelompokkan semua aturan kustom yang dihitung dengan berapa kali cocok

Kelompok kueri berikut semua menghitung aturan kustom dengan berapa kali cocok.

```
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
```

Untuk informasi tentang lokasi log untuk aturan kustom dan grup aturan terkelola, lihat Memantau dan menyetel di Panduan AWS WAF Pengembang.

Bekerja dengan tanggal dan waktu

Example — Kembalikan bidang stempel waktu dalam format ISO 8601 yang dapat dibaca manusia

Kueri berikut menggunakan`from_unixtime`dan`to_iso8601`fungsi untuk mengembalikan`timestamp`dalam format ISO 8601 yang dapat dibaca manusia (misalnya,`2019-12-13T23:40:12.000Z`Sebagai gantinya`1576280412771`). kueri juga mengembalikan nama sumber HTTP, sumber ID, dan permintaan.

```
SELECT to_iso8601(from_unixtime(timestamp / 1000)) as time_ISO_8601,
       httpsourcename,
       httpsourceid,
       httprequest
FROM waf_logs
LIMIT 10;
```

Example — Kembalikan catatan dari 24 jam terakhir

Kueri berikut menggunakan filter di`WHERE`klausa untuk mengembalikan nama sumber HTTP, ID sumber HTTP, dan bidang permintaan HTTP untuk catatan dari 24 jam terakhir.

```
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 — Kembalikan catatan untuk rentang tanggal dan alamat IP yang ditentukan

Kueri berikut mencantumkan catatan dalam rentang tanggal yang ditentukan untuk alamat IP klien tertentu.

```
SELECT *
FROM waf_logs
WHERE httprequest.clientip='53.21.198.66' AND "date" >= '2021/03/01' AND "date" <
 '2021/03/31'
```

Example — Untuk rentang tanggal tertentu, hitung jumlah alamat IP dalam interval lima menit

Kueri berikut menghitung, untuk rentang tanggal tertentu, jumlah alamat IP dalam interval lima menit.

```
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 - Hitung jumlah X-Forwarded-For IP dalam 10 hari terakhir

Kueri berikut memfilter header permintaan dan menghitung jumlah X-Forwarded-For IP dalam 10 hari terakhir.

```
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
```

Untuk informasi selengkapnya tentang fungsi tanggal dan waktu, lihat Fungsi dan operator tanggal dan waktu dalam dokumentasi Trino.

Bekerja dengan permintaan dan alamat yang diblokir

Example — Ekstrak 100 alamat IP teratas yang diblokir oleh jenis aturan tertentu

Kueri berikut ekstrak dan menghitung atas 100 alamat IP yang telah diblokir olehRATE_BASEDmengakhiri aturan selama rentang tanggal yang ditentukan.

```
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 — Hitung berapa kali permintaan dari negara tertentu telah diblokir

Kueri berikut menghitung jumlah kali permintaan telah tiba dari alamat IP milik Irlandia (IE) dan telah diblokir olehRATE_BASEDmengakhiri aturan.

```
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 — Hitung berapa kali permintaan diblokir, dikelompokkan berdasarkan atribut tertentu

Kueri berikut menghitung berapa kali permintaan telah diblokir, dengan hasil dikelompokkan berdasarkan WebACL,, Clientip RuleId, dan HTTP Request URI.

```
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 — Hitung berapa kali ID aturan penghentian tertentu telah dicocokkan

Kueri berikut menghitung berapa kali ID aturan terminating tertentu telah dicocokkan (`WHERE terminatingruleid='e9dd190d-7a43-4c06-bcea-409613d9506e'`). Kueri kemudian grup hasil dengan WebACL, Action, ClientIP, dan HTTP Request 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 — Ambil 100 alamat IP teratas yang diblokir selama rentang tanggal tertentu

Kueri berikut ekstrak atas 100 alamat IP yang telah diblokir untuk rentang tanggal yang ditentukan. Kueri juga mencantumkan berapa kali alamat IP telah diblokir.

```
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
```

Untuk informasi selengkapnya tentang CloudTrail dan Amazon S3, lihat topik berikut:

- Bagaimana cara menganalisis log akses server Amazon S3 menggunakan Athena?di AWS Pusat Pengetahuan
- Menanyakan log akses Amazon S3 untuk permintaan menggunakan Amazon Athena di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon
- Menggunakan AWS CloudTrail untuk mengidentifikasi permintaan Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon

Menanyakan log server web yang disimpan di Amazon S3

Anda dapat menggunakan Athena untuk mengkueri log server Web yang disimpan di Amazon S3. Topik di bagian ini menunjukkan cara membuat tabel di Athena untuk meminta log server Web dalam berbagai format.

Topik

- Menanyakan log Apache yang disimpan di Amazon S3
- Menanyakan log server informasi internet (IIS) yang disimpan di Amazon S3

Menanyakan log Apache yang disimpan di Amazon S3

Anda dapat menggunakan Amazon Athena untuk menanyakan file log Apache HTTP Server yang disimpan di akun Amazon S3 Anda. Topik ini menunjukkan cara membuat skema tabel untuk menanyakan file log Apache Access dalam format log umum.

Bidang dalam format log umum termasuk alamat IP klien, ID klien, ID pengguna, permintaan diterima stempel waktu, teks permintaan klien, kode status server, dan ukuran objek kembali ke klien.

Contoh data berikut menunjukkan Apache format log umum.

```
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
```

Membuat tabel di Athena untuk log Apache

Sebelum Anda dapat meminta log Apache disimpan di Amazon S3, Anda harus membuat skema tabel untuk Athena sehingga dapat membaca data log. Untuk membuat tabel Athena untuk Apache log, Anda dapat menggunakan Grok SerDe. Untuk informasi selengkapnya tentang menggunakan Grok SerDe, lihat Menulis pengklasifikasi kustom grok di Panduan Pengembang.AWS Glue

Untuk membuat tabel di Athena untuk log server web Apache

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Tempelkan pernyataan DDL berikut ke Editor Kueri Athena. Modifikasi nilai di `LOCATION 's3://DOC-EXAMPLE-BUCKET/apache-log-folder/'` untuk mengarahkan ke log Apache Anda di Amazon S3.

```
CREATE EXTERNAL TABLE apache_logs (
  client_ip string,
  client_id string,
  user_id string,
  request_received_time string,
  client_request string,
  server_status string,
  returned_obj_size string
  )
ROW FORMAT SERDE
   'com.amazonaws.glue.serde.GrokSerDe'
WITH SERDEPROPERTIES (
```

```
   'input.format'='^%{IPV4:client_ip} %{DATA:client_id} %{USERNAME:user_id}
 %{GREEDYDATA:request_received_time} %{QUOTEDSTRING:client_request}
 %{DATA:server_status} %{DATA: returned_obj_size}$'
   )
STORED AS INPUTFORMAT
   'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
   'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
   's3://DOC-EXAMPLE-BUCKET/apache-log-folder/';
```

3. jalankan kueri di konsol Athena untuk mendaftarkan tabel `apache_logs`. Saat permintaan selesai, log siap bagi Anda untuk mengkueri dari Athena.

Contoh kueri pilih untuk log Apache

Example – Pemfilteran untuk kesalahan 404

Contoh kueri berikut memilih waktu permintaan diterima, teks permintaan klien, dan kode status server dari tabel `apache_logs`. Klausa `WHERE` memfilter untuk kode status HTTP`404` (halaman tidak ditemukan).

```
SELECT request_received_time, client_request, server_status
FROM apache_logs
WHERE server_status = '404'
```

Gambar berikut menunjukkan hasil kueri di Editor Kueri Athena.

Results

	request_received_time	client_request	server_status
1	[11/Jan/2020:11:40:11 -0700]	GET /image.png HTTP/1.1	404
2	[15/Feb/2019:10:12:22 -0700]	GET /favicon.ico HTTP/1.1	404

Example – Memfilter untuk permintaan yang berhasil

Contoh kueri berikut memilih ID pengguna, waktu permintaan diterima, teks permintaan klien, dan kode status server dari tabel `apache_logs`. Klausa `WHERE` memfilter untuk kode status HTTP `200` (berhasil).

```
SELECT user_id, request_received_time, client_request, server_status
FROM apache_logs
WHERE server_status = '200'
```

Gambar berikut menunjukkan hasil kueri di Editor Kueri Athena.

Results

	user_id	request_received_time	client_request	server_status
1	Li	[10/Oct/2019:13:55:36 -0700]	GET /logo.gif HTTP/1.0	200
2	Jorge	[24/Nov/2019:10:49:52 -0700]	GET /index.html HTTP/1.1	200
3	Mateo	[27/Dec/2019:11:38:12 -0700]	GET /about.html HTTP/1.1	200
4	Saanvi	[14/Mar/2019:11:40:33 -0700]	GET /intro.html HTTP/1.1	200
5	Xiulan	[22/Apr/2019:10:51:34 -0700]	GET /group/index.html HTTP/1.1	200

Example — Penyaringan dengan stempel waktu

Contoh kueri berikut untuk catatan yang permintaannya diterima waktu lebih besar dari stempel waktu yang ditentukan.

```
SELECT * FROM apache_logs WHERE request_received_time > 10/Oct/2023:00:00:00
```

Menanyakan log server informasi internet (IIS) yang disimpan di Amazon S3

Anda dapat menggunakan Amazon Athena untuk mengkueri log server web Microsoft Internet Information Services (IIS) disimpan dalam akun Amazon S3 Anda. Sementara IIS menggunakan berbagai format berkas log, topik ini menunjukkan cara membuat tabel skema untuk mengkueri W3C diperpanjang dan log format berkas log IIS dari Athena.

Karena format file log W3C diperluas dan IIS menggunakan pembatas karakter tunggal (spasi dan koma, masing-masing) dan tidak memiliki nilai terlampir dalam tanda kutip, Anda dapat menggunakan untuk membuat tabel Athena untuk mereka. LazySimpleSerDe

Format file log diperpanjang W3C

Format data berkas log W3C extended memiliki bidang yang dipisahkan dengan spasi. Bidang yang muncul di log diperpanjang W3C ditentukan oleh administrator server web yang memilih bidang log yang akan disertakan. Data log contoh berikut memiliki bidang `date, time`, `c-ip`, `s-ip`, `cs-method`, `cs-uri-stem`, `sc-status`, `sc-bytes`, `cs-bytes`, `time-taken`, dan `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
```

Membuat tabel di Athena untuk log diperpanjang W3C

Sebelum Anda dapat mengkueri log W3C diperpanjang Anda, Anda harus membuat skema tabel sehingga Athena dapat membaca data log.

Untuk membuat tabel di Athena untuk log diperpanjang W3C

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Paste pernyataan DDL seperti berikut ke konsol Athena, mencatat hal-hal berikut:

    a. Menambahkan atau menghapus kolom dalam contoh untuk sesuai dengan bidang di log yang ingin Anda kueri.

    b. Nama kolom dalam format berkas log diperluas W3C berisi tanda hubung (-). Namun, sesuai dengan Konvensi penamaan Athena, contoh pernyataan `CREATE TABLE` menggantikannya dengan garis bawah (_).

    c. Untuk menentukan pembatas ruang, gunakan. `FIELDS TERMINATED BY ' '`

    d. Modifikasi nilai di`LOCATION 's3://DOC-EXAMPLE-BUCKET/w3c-log-folder/'` untuk menunjuk ke W3C extended log Anda di Amazon S3.

```
CREATE EXTERNAL TABLE `iis_w3c_logs`(
  date_col string,
  time_col string,
  c_ip string,
  s_ip string,
  cs_method string,
  cs_uri_stem string,
  sc_status string,
```

```
  sc_bytes string,
  cs_bytes string,
  time_taken string,
  cs_version string
  )
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ' '
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION   's3://DOC-EXAMPLE-BUCKET/w3c-log-folder/'
```

3. Jalankan kueri di konsol Athena untuk mendaftarkan tabel `iis_w3c_logs`. Saat permintaan selesai, log siap bagi Anda untuk mengkueri dari Athena.

Contoh kueri pilih log diperpanjang W3C

Contoh kueri berikut memilih tanggal, waktu, permintaan target, dan waktu yang dibutuhkan untuk permintaan dari tabel `iis_w3c_logs`. Klausa `WHERE` memfilter untuk kasus dengan metode HTTP adalah `GET` dan kode status HTTP adalah `200`(berhasil).

```
SELECT date_col, time_col, cs_uri_stem, time_taken
FROM iis_w3c_logs
WHERE cs_method = 'GET' AND sc_status = '200'
```

Gambar berikut menunjukkan hasil kueri di Editor Kueri Athena.

Results

	date_col	time_col	cs_uri_stem	time_taken
1	2020-01-19	22:48:39	/default.html	157
2	2020-01-19	22:49:40	/index.html	164
3	2020-01-19	22:50:12	/image.gif	358
4	2020-01-19	22:51:44	/faq.html	288

Menggabungkan bidang tanggal dan waktu

Bidang `date` dan `time` yang dibatasi spasi adalah entri terpisah dalam data sumber log, tetapi dapat menggabungkan mereka ke stempel waktu jika Anda ingin. Gunakan fungsi concat () dan date_parse () di kueriSELECT atau CREATE TABLE AS SELECT untuk menggabungkan dan mengonversi tanggal dan waktu kolom ke dalam format stempel waktu. Contoh berikut menggunakan kueri CTAS untuk membuat tabel baru dengan kolom `derived_timestamp`.

```
CREATE TABLE iis_w3c_logs_w_timestamp AS
SELECT
  date_parse(concat(date_col,' ', time_col),'%Y-%m-%d %H:%i:%s') as derived_timestamp,
  c_ip,
  s_ip,
  cs_method,
  cs_uri_stem,
  sc_status,
  sc_bytes,
  cs_bytes,
  time_taken,
  cs_version
FROM iis_w3c_logs
```

Setelah tabel dibuat, Anda dapat mengkueri kolom stempel waktu baru langsung, seperti dalam contoh berikut.

```
SELECT derived_timestamp, cs_uri_stem, time_taken
FROM iis_w3c_logs_w_timestamp
WHERE cs_method = 'GET' AND sc_status = '200'
```

Gambar berikut menunjukkan hasil kueri.

Results

	derived_timestamp	cs_uri_stem	time_taken
1	2020-01-19 22:48:39.000	/default.html	157
2	2020-01-19 22:49:40.000	/index.html	164
3	2020-01-19 22:50:12.000	/image.gif	358
4	2020-01-19 22:51:44.000	/faq.html	288

Format file log IIS

Berbeda dengan format W3C extended, format berkas log IIS memiliki seperangkat bidang tetap dan termasuk koma sebagai pembatas. LazySimpleSerDe Perlakukan koma sebagai pembatas dan spasi setelah koma sebagai awal bidang berikutnya.

Contoh berikut menunjukkan data sampel dalam format berkas log IIS.

```
203.0.113.15, -, 2020-02-24, 22:48:38, W3SVC2, SERVER5, 198.51.100.4, 254, 501, 488,
 200, 0, GET, /index.htm, -,
203.0.113.4, -, 2020-02-24, 22:48:39, W3SVC2, SERVER6, 198.51.100.6, 147, 411, 388,
 200, 0, GET, /about.html, -,
203.0.113.11, -, 2020-02-24, 22:48:40, W3SVC2, SERVER7, 198.51.100.18, 170, 531, 468,
 200, 0, GET, /image.png, -,
203.0.113.8, -, 2020-02-24, 22:48:41, W3SVC2, SERVER8, 198.51.100.14, 125, 711, 868,
 200, 0, GET, /intro.htm, -,
```

Membuat tabel di Athena untuk file log IIS

Untuk kueri log format berkas log IIS Anda di Amazon S3, Anda pertama kali membuat skema tabel sehingga Athena dapat membaca data log.

Untuk membuat tabel di Athena untuk log format berkas log IIS

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Paste pernyataan DDL berikut ke konsol Athena, mencatat poin-poin berikut:
   a. Untuk menentukan pembatas koma, gunakan `FIELDS TERMINATED BY ','`.
   b. Ubah nilai di LOCATION 's3://DOC-EXAMPLE-BUCKET/ *iis-log-file-folder*/' untuk menunjuk ke file log format log IIS Anda di Amazon S3.

```
CREATE EXTERNAL TABLE `iis_format_logs`(
client_ip_address string,
user_name string,
request_date string,
request_time string,
service_and_instance string,
server_name string,
server_ip_address string,
time_taken_millisec string,
client_bytes_sent string,
```

```
server_bytes_sent string,
service_status_code string,
windows_status_code string,
request_type string,
target_of_operation string,
script_parameters string
   )
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/iis-log-file-folder/'
```

3. jalankan kueri di konsol Athena untuk mendaftarkan tabel `iis_format_logs`. Saat permintaan selesai, log siap bagi Anda untuk mengkueri dari Athena.

Contoh kueri pilih format log IIS

Contoh kueri berikut memilih tanggal permintaan, permintaan waktu, permintaan target, dan waktu yang dibutuhkan dalam milidetik dari tabel `iis_format_logs`. Klausa `WHERE` memfilter untuk kasus dengan tipe permintaan adalah `GET` dan kode status HTTP adalah `200` (berhasil). Dalam kueri, perhatikan bahwa spasi utama masuk `' GET'` dan `' 200'` diperlukan untuk membuat kueri berhasil.

```
SELECT request_date, request_time, target_of_operation, time_taken_millisec
FROM iis_format_logs
WHERE request_type = ' GET' AND service_status_code = ' 200'
```

Gambar berikut menunjukkan hasil kueri data sampel.

Results

	request_date	request_time	target_of_operation	time_taken_millisec
1	2020-02-24	22:48:38	/index.htm	254
2	2020-02-24	22:48:39	/about.html	147
3	2020-02-24	22:48:40	/image.png	170
4	2020-02-24	22:48:41	/intro.htm	125

Format file log NCSA

IIS juga menggunakan format logging NCSA, yang memiliki jumlah bidang tetap dalam format teks ASCII yang dipisahkan oleh spasi. Struktur ini mirip dengan format log umum yang digunakan untuk log akses Apache. Kolom dalam format data log umum NCSA termasuk alamat IP klien, ID klien (tidak biasanya digunakan), domain\ user ID, permintaan diterima stempel waktu, teks permintaan klien, kode status server, dan ukuran objek kembali ke klien.

Contoh berikut menunjukkan data dalam format log umum NCSA sebagai didokumentasikan untuk IIS.

```
198.51.100.7 - ExampleCorp\Li [10/Oct/2019:13:55:36 -0700] "GET /logo.gif HTTP/1.0" 200
 232
198.51.100.14 - AnyCompany\Jorge [24/Nov/2019:10:49:52 -0700] "GET /index.html
 HTTP/1.1" 200 2165
198.51.100.22 - ExampleCorp\Mateo [27/Dec/2019:11:38:12 -0700] "GET /about.html
 HTTP/1.1" 200 1287
198.51.100.9 - AnyCompany\Nikki [11/Jan/2020:11:40:11 -0700] "GET /image.png HTTP/1.1"
 404 230
198.51.100.2 - ExampleCorp\Ana [15/Feb/2019:10:12:22 -0700] "GET /favicon.ico HTTP/1.1"
 404 30
198.51.100.13 - AnyCompany\Saanvi [14/Mar/2019:11:40:33 -0700] "GET /intro.html
 HTTP/1.1" 200 1608
198.51.100.11 - ExampleCorp\Xiulan [22/Apr/2019:10:51:34 -0700] "GET /group/index.html
 HTTP/1.1" 200 1344
```

Membuat tabel di Athena untuk log IIS NCSA

Untuk`CREATE TABLE`pernyataan, Anda dapat menggunakanGrok SerDedan pola grok yang mirip dengan yangLog server web Apache. Tidak seperti Apache log, pola grok menggunakan`%{DATA:user_id}`untuk bidang ketiga, bukan`%{USERNAME:user_id}`untuk memperhitungkan adanya backslash di`domain\user_id`. Untuk informasi selengkapnya tentang menggunakan Grok SerDe, lihat Menulis pengklasifikasi kustom grok di Panduan Pengembang.AWS Glue

Untuk membuat tabel di Athena untuk log server web IIS NCSA

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Paste pernyataan DDL berikut ke Editor Kueri Athena. Modifikasi nilai di`LOCATION 's3://DOC-EXAMPLE-BUCKET/iis-ncsa-logs/'`Untuk menunjuk ke log IIS NCSA di Amazon S3.

```
CREATE EXTERNAL TABLE iis_ncsa_logs(
  client_ip string,
  client_id string,
  user_id string,
  request_received_time string,
  client_request string,
  server_status string,
  returned_obj_size string
  )
ROW FORMAT SERDE
   'com.amazonaws.glue.serde.GrokSerDe'
WITH SERDEPROPERTIES (
   'input.format'='^%{IPV4:client_ip} %{DATA:client_id} %{DATA:user_id}
 %{GREEDYDATA:request_received_time} %{QUOTEDSTRING:client_request}
 %{DATA:server_status} %{DATA: returned_obj_size}$'
   )
STORED AS INPUTFORMAT
   'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
   'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
   's3://DOC-EXAMPLE-BUCKET/iis-ncsa-logs/';
```

3. Menjalankan kueri di konsol Athena untuk mendaftar`iis_ncsa_logs`Tabel. Saat permintaan selesai, log siap bagi Anda untuk mengkueri dari Athena.

Contoh kueri pilih untuk log IIS NCSA

Example – Pemfilteran untuk kesalahan 404

Contoh kueri berikut memilih waktu permintaan diterima, teks permintaan klien, dan kode status server dari tabel `iis_ncsa_logs`. Klausa `WHERE` memfilter untuk kode status HTTP`404` (halaman tidak ditemukan).

```
SELECT request_received_time, client_request, server_status
FROM iis_ncsa_logs
WHERE server_status = '404'
```

Gambar berikut menunjukkan hasil kueri di Editor Kueri Athena.

Results

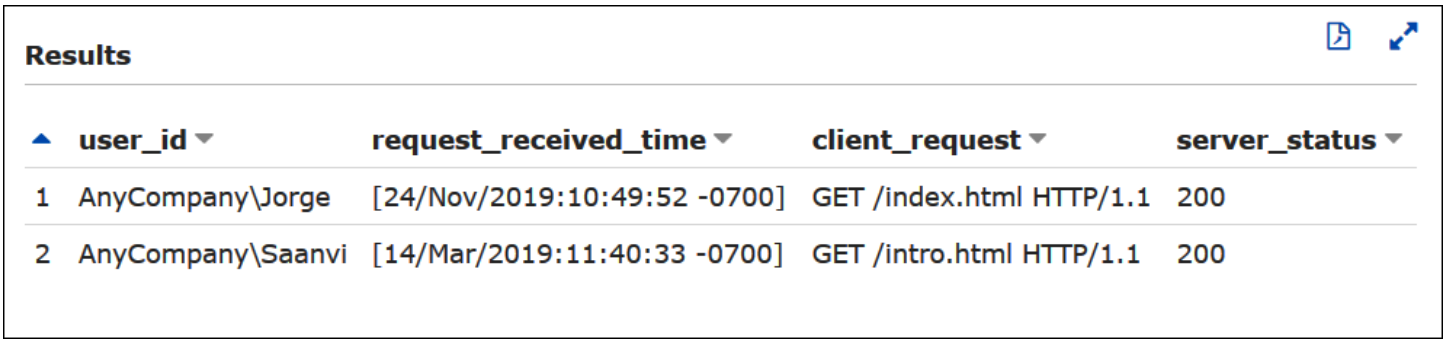
	request_received_time	client_request	server_status
1	[11/Jan/2020:11:40:11 -0700]	GET /image.png HTTP/1.1	404
2	[15/Feb/2019:10:12:22 -0700]	GET /favicon.ico HTTP/1.1	404

Example – Memfilter untuk permintaan sukses dari domain tertentu

Contoh kueri berikut memilih ID pengguna, waktu permintaan diterima, teks permintaan klien, dan kode status server dari tabel `iis_ncsa_logs`. Klausa `WHERE` memfilter untuk permintaan dengan kode status HTTP `200` (berhasil) dari pengguna di domain `AnyCompany`.

```
SELECT user_id, request_received_time, client_request, server_status
FROM iis_ncsa_logs
WHERE server_status = '200' AND user_id LIKE 'AnyCompany%'
```

Gambar berikut menunjukkan hasil kueri di Editor Kueri Athena.



Results

	user_id	request_received_time	client_request	server_status
1	AnyCompany\Jorge	[24/Nov/2019:10:49:52 -0700]	GET /index.html HTTP/1.1	200
2	AnyCompany\Saanvi	[14/Mar/2019:11:40:33 -0700]	GET /intro.html HTTP/1.1	200

Menggunakan transaksi ACID Athena

Istilah “transaksi ACID” mengacu pada seperangkat properti (atomisitas, konsistensi, isolasi, dan daya tahan) yang memastikan integritas data dalam transaksi database. Transaksi ACID memungkinkan beberapa pengguna untuk secara bersamaan dan andal menambah dan menghapus objek Amazon S3 secara atom, sambil mengisolasi kueri yang ada dengan mempertahankan konsistensi baca untuk kueri terhadap data lake. Transaksi Athena ACID menambahkan dukungan tabel tunggal untuk menyisipkan, menghapus, memperbarui, dan operasi perjalanan waktu ke bahasa manipulasi data SQL Athena (DML/bahasa manipulasi data SQL). Anda dan beberapa

pengguna bersamaan dapat menggunakan transaksi Athena ACID untuk membuat modifikasi tingkat baris yang andal pada data Amazon S3. Transaksi Athena secara otomatis mengelola semantik dan koordinasi penguncian dan tidak memerlukan solusi penguncian catatan khusus.

Transaksi Athena ACID dan sintaks SQL yang sudah dikenal menyederhanakan pembaruan data bisnis dan peraturan Anda. Misalnya, untuk menanggapi permintaan penghapusan data, Anda dapat melakukan operasi SQL`DELETE`. Untuk membuat koreksi catatan manual, Anda dapat menggunakan satu `UPDATE` pernyataan. Untuk memulihkan data yang baru saja dihapus, Anda dapat mengeluarkan kueri perjalanan waktu menggunakan `SELECT` pernyataan.

Karena dibangun di atas format tabel bersama, transaksi Athena ACID kompatibel dengan layanan dan mesin lain seperti Amazon EMR dan Apache Spark yang juga mendukung format tabel bersama.

Transaksi Athena tersedia melalui konsol Athena, operasi API, dan driver ODBC dan JDBC.

Topik

- Menanyakan tabel Delta Lake Linux Foundation
- Menggunakan Athena untuk menanyakan kumpulan data Apache Hudi
- Menggunakan tabel Apache Iceberg

Menanyakan tabel Delta Lake Linux Foundation

Linux Foundation Delta Lake adalah format tabel untuk analitik data besar. Anda dapat menggunakan Amazon Athena untuk membaca tabel Delta Lake yang disimpan di Amazon S3 secara langsung tanpa harus membuat file manifes atau menjalankan pernyataan. `MSCK REPAIR`

Format Delta Lake menyimpan nilai minimum dan maksimum per kolom dari setiap file data. Implementasi Athena memanfaatkan informasi ini untuk memungkinkan melewatkan file pada predikat untuk menghilangkan file yang tidak diinginkan dari pertimbangan.

Pertimbangan dan batasan

Dukungan Delta Lake di Athena memiliki pertimbangan dan batasan berikut:

- Tabel dengan AWS Glue katalog saja — Dukungan Danau Delta Asli hanya didukung melalui tabel yang AWS Glue terdaftar. Jika Anda memiliki meja Danau Delta yang terdaftar di metastore lain, Anda masih dapat menyimpannya dan memperlakukannya sebagai metastore utama Anda. Karena metadata Delta Lake disimpan dalam sistem file (misalnya, di Amazon S3) daripada di

metastore, Athena hanya memerlukan properti lokasi untuk membaca dari tabel Delta Lake Anda. AWS Glue

- Hanya mesin V3 - Kueri Delta Lake hanya didukung pada mesin Athena versi 3. Anda harus memastikan bahwa workgroup yang Anda buat dikonfigurasi untuk menggunakan mesin Athena versi 3.
- Versi pembaca Delta Lake — Protokol pembaca Delta Lake hingga versi 3 didukung.
- Pemetaan kolom dan TimestampNTZ — Pemetaan kolom Delta, yang memungkinkan kolom tabel Delta dan kolom file Parket yang mendasarinya menggunakan nama yang berbeda, dan stempel waktu tanpa zona waktu (TimestampNTZ) didukung.
- Tidak ada dukungan perjalanan waktu — Tidak ada dukungan untuk pertanyaan yang menggunakan kemampuan perjalanan waktu Delta Lake.
- Hanya baca - Tulis pernyataan DMLsepertiUPDATE,INSERT, atau tidak DELETE didukung.
- Dukungan Lake Formation — Integrasi Lake Formation tersedia untuk tabel Delta Lake dengan skema mereka yang sinkron. AWS GlueUntuk informasi selengkapnya, lihat Menggunakan AWS Lake Formation Amazon Athena dan Mengatur izin untuk tabel Delta Lake di Panduan Pengembang.AWS Lake Formation
- Dukungan DDL terbatas - Pernyataan DDL berikut didukung:CREATE EXTERNAL TABLE,,SHOW COLUMNS, SHOW TBLPROPERTIES SHOW PARTITIONSSHOW CREATE TABLE, dan. DESCRIBE Untuk informasi tentang menggunakan CREATE EXTERNAL TABLE pernyataan, lihat Memulai bagian.
- Melewati objek S3 Glacier tidak didukung - Jika objek di tabel Delta Lake Linux Foundation berada dalam kelas penyimpanan Amazon S3 Glacier, menyetel properti tabel agar tidak berpengaruh. read_restored_glacier_objects false

  Misalnya, Anda mengeluarkan perintah berikut:

```
ALTER TABLE table_name SET TBLPROPERTIES ('read_restored_glacier_objects' = 'false')
```

  Untuk tabel Iceberg dan Delta Lake, perintah menghasilkan kesalahan Kunci properti tabel Tidak didukung: read_restored_glacier_objects. Untuk tabel Hudi, ALTER TABLE perintah tidak menghasilkan kesalahan, tetapi objek Amazon S3 Glacier masih belum dilewati. Menjalankan SELECT kueri setelah ALTER TABLE perintah terus mengembalikan semua objek.

Tipe data kolom non-partisi yang didukung

Untuk kolom non-partisi, semua tipe data yang didukung Athena CHAR kecuali didukung CHAR (tidak didukung dalam protokol Delta Lake itu sendiri). Tipe data yang didukung meliputi:

```
boolean
tinyint
smallint
integer
bigint
double
float
decimal
varchar
string
binary
date
timestamp
array
map
struct
```

Tipe data kolom partisi yang didukung

Untuk kolom partisi, Athena mendukung tabel dengan tipe data berikut:

```
boolean
integer
smallint
tinyint
bigint
decimal
float
double
date
timestamp
varchar
```

Untuk informasi selengkapnya tentang tipe data di Athena, lihat. Tipe data di Amazon Athena

Memulai

Agar dapat ditanyakan, meja Danau Delta Anda harus ada di. AWS Glue Jika tabel Anda berada di Amazon S3 tetapi tidak di AWS Glue, jalankan `CREATE EXTERNAL TABLE` pernyataan menggunakan sintaks berikut. Jika tabel Anda sudah ada di AWS Glue (misalnya, karena Anda menggunakan Apache Spark atau mesin lain dengan AWS Glue), Anda dapat melewati langkah ini.

```
CREATE EXTERNAL TABLE
  [db_name.]table_name
  LOCATION 's3://DOC-EXAMPLE-BUCKET/your-folder/'
  TBLPROPERTIES ('table_type' = 'DELTA')
```

Perhatikan kelalaian definisi kolom, SerDe pustaka, dan properti tabel lainnya. Tidak seperti tabel Hive tradisional, metadata tabel Delta Lake disimpulkan dari log transaksi Delta Lake dan disinkronkan langsung ke. AWS Glue

> **Note**
>
> Untuk tabel Delta Lake, `CREATE TABLE` pernyataan yang mencakup lebih dari `table_type` properti `LOCATION` dan tidak diperbolehkan.

Membaca tabel Danau Delta

Untuk menanyakan tabel Delta Lake, gunakan sintaks SQL `SELECT` standar:

```
[ 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 ] ]
```

Untuk informasi selengkapnya tentang `SELECT` sintaks, lihat SELECT di dokumentasi Athena.

Format Delta Lake menyimpan nilai minimum dan maksimum per kolom dari setiap file data. Athena memanfaatkan informasi ini untuk memungkinkan file melewatkan predikat untuk menghilangkan file yang tidak perlu dari pertimbangan.

Sinkronisasi metadata Delta Lake

Athena menyinkronkan metadata tabel, termasuk skema, kolom partisi, dan properti tabel, jika AWS Glue Anda menggunakan Athena untuk membuat tabel Delta Lake Anda. Seiring berjalannya waktu, metadata ini dapat kehilangan sinkronisasi dengan metadata tabel yang mendasarinya di log transaksi. Untuk memperbarui tabel Anda, Anda dapat memilih salah satu opsi berikut:

- Gunakan AWS Glue crawler untuk tabel Delta Lake. Untuk informasi selengkapnya, lihat Memperkenalkan dukungan tabel Delta Lake asli dengan AWS Glue crawler di Blog AWS Big Data dan Menjadwalkan AWS Glue crawler di Panduan Pengembang. AWS Glue
- Jatuhkan dan buat ulang tabel di Athena.
- Gunakan SDK, CLI, AWS Glue atau konsol untuk memperbarui skema secara manual. AWS Glue

Perhatikan bahwa fitur berikut mengharuskan AWS Glue skema Anda untuk selalu memiliki skema yang sama dengan log transaksi:

- Lake Formation
- Tampilan
- Filter baris dan kolom

Jika alur kerja Anda tidak memerlukan fungsionalitas ini, dan Anda memilih untuk tidak mempertahankan kompatibilitas ini, Anda dapat menggunakan `CREATE TABLE` DDL di Athena dan kemudian menambahkan jalur Amazon S3 sebagai parameter di. SerDe AWS Glue

Untuk membuat tabel Delta Lake menggunakan AWS Glue Athena dan konsol

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Di editor kueri Athena, gunakan DDL berikut untuk membuat tabel Delta Lake Anda. Perhatikan bahwa saat menggunakan metode ini, nilai untuk `TBLPROPERTIES` harus `'spark.sql.sources.provider' = 'delta'` dan tidak`'table_type' = 'delta'`.

Perhatikan bahwa skema yang sama ini (dengan satu kolom bernama `col` tipe`array<string>`) dimasukkan saat Anda menggunakan Apache Spark (Athena untuk Apache Spark) atau sebagian besar mesin lain untuk membuat tabel Anda.

```
CREATE EXTERNAL TABLE
   [db_name.]table_name(col array<string>)
   LOCATION 's3://DOC-EXAMPLE-BUCKET/your-folder/'
   TBLPROPERTIES ('spark.sql.sources.provider' = 'delta')
```

3. Buka AWS Glue konsol di https://console.aws.amazon.com/glue/.
4. Di panel navigasi, pilih Katalog Data, Tabel.
5. Dalam daftar tabel, pilih tautan untuk tabel Anda.
6. Pada halaman untuk tabel, pilih Tindakan, Edit tabel.
7. Di bagian parameter Serde, tambahkan kunci **`path`** dengan nilai**`s3://DOC-EXAMPLE-BUCKET/your-folder/`**.
8. Pilih Simpan.

Sumber daya tambahan

Untuk diskusi tentang penggunaan tabel Delta Lake dengan AWS Glue dan menanyakannya dengan Athena, lihat Menangani operasi data UPSERT menggunakan Delta Lake sumber terbuka dan di Blog Big Data. AWS GlueAWS

Menggunakan Athena untuk menanyakan kumpulan data Apache Hudi

Apache Hudi adalah kerangka kerja manajemen data sumber terbuka yang menyederhanakan pemrosesan data tambahan. Tindakan sisipan level catatan, pembaruan, upsert, dan hapus diproses jauh lebih terperinci, mengurangi overhead. `Upsert` mengacu pada kemampuan untuk menyisipkan catatan ke dalam set data yang ada jika mereka belum ada atau untuk memperbaruinya jika mereka sudah ada.

Hudi menangani peristiwa penyisipan dan pembaruan data tanpa membuat banyak file kecil yang dapat menyebabkan masalah performa untuk analisis. Apache Hudi secara otomatis melacak perubahan dan menggabungkan file sehingga ukurannya tetap optimal. Ini untuk menghindari kebutuhan untuk membangun solusi kustom yang memantau dan menulis ulang banyak file kecil ke dalam file besar yang lebih sedikit.

Set data Hudi cocok untuk kasus penggunaan berikut:

- Mematuhi peraturan privasi seperti Peraturan Perlindungan Data Umum (GDPR) dan California Consumer Privacy Act (CCPA) yang memberlakukan hak orang untuk menghapus informasi pribadi atau mengubah cara data mereka digunakan.
- Bekerja dengan data streaming dari sensor dan perangkat Internet untuk Segala (IoT) lainnya yang memerlukan penyisipan data dan pembaruan peristiwa tertentu.
- Mengimplementasikan sistem change data capture (CDC).

Kumpulan data yang dikelola oleh Hudi disimpan di Amazon S3 menggunakan format penyimpanan terbuka. Saat ini, Athena dapat membaca set data Hudi yang dipadatkan tetapi tidak menulis data Hudi. Athena mendukung hingga Hudi versi 0.8.0 dengan mesin Athena versi 2, dan Hudi versi 0.14.0 dengan mesin Athena versi 3. Ini dapat berubah. Athena tidak dapat menjamin kompatibilitas baca dengan tabel yang dibuat dengan versi Hudi yang lebih baru. Untuk informasi tentang versi mesin Athena, lihat Pembuatan versi mesin Athena. Untuk informasi selengkapnya tentang fitur dan pembuatan versi Hudi, lihat dokumentasi Hudi di situs web Apache.

Jenis tabel dataset Hudi

Sebuah set data Hudi dapat menjadi salah satu dari tipe berikut:

- Copy on Write (CoW) – Data disimpan dalam format kolom (Parquet), dan setiap pembaruan membuat versi file baru selama penulisan.
- Merge on Read (MoR) – Data disimpan menggunakan kombinasi kolom (Parquet) dan format berbasis baris (Avro). Pembaruan dicatat ke file `delta` berbasis baris dan dipadatkan sesuai kebutuhan untuk membuat file kolom versi baru.

Dengan set data CoW, setiap kali ada pembaruan ke catatan, file yang berisi catatan ditulis ulang dengan nilai yang diperbarui. Dengan set data MoR, setiap kali ada pembaruan, Hudi hanya menulis baris untuk catatan yang berubah. MoR lebih cocok untuk beban kerja tulis atau perubahan berat dengan lebih sedikit pembacaan. CoW lebih cocok untuk beban kerja pembacaan berat pada data yang jarang berubah.

Hudi menyediakan tiga tipe kueri untuk mengakses data:

- Kueri snapshot — Kueri yang melihat snapshot terbaru dari tabel sebagai tindakan komit atau pemadatan yang diberikan. Untuk tabel MoR, kueri snapshot memapar status terbaru tabel dengan menggabungkan file dasar dan delta potongan file terbaru pada pada saat kueri.
- Kueri tambahan — Kueri hanya melihat data baru yang ditulis ke tabel, karena komit/pemadatan yang diberikan. Ini secara efektif menyediakan pengaliran perubahan untuk mengaktifkan data pipeline tambahan.
- Baca kueri yang dioptimalkan — Untuk tabel MoR, kueri melihat data terbaru yang dipadatkan. Untuk tabel CoW, kueri melihat data terbaru yang dikomit.

Tabel berikut menunjukkan kemungkinan tipe kueri Hudi untuk setiap tipe tabel.

Jenis tabel	Kemungkinan jenis kueri Hudi
Copy On Write	Snapshot, tambahan
Merge On Read	snapshot, tambahan, dioptimalkan baca

Saat ini, Athena mendukung kueri snapshot dan dioptimalkan baca, tetapi tidak kueri tambahan. Pada tabel MoR, semua data yang terpapar untuk mengkueri dioptimalkan baca dipadatkan. Ini memberikan performa yang baik tetapi tidak termasuk komit delta terbaru. Kueri snapshot berisi data terbaru tetapi dikenai beberapa overhead komputasi, yang membuat performa kueri ini tidak terlalu baik.

Untuk informasi selengkapnya tentang pengorbanan antara tipe tabel dan kueri, lihat Jenis Tabel & Kueri dalam dokumentasi Apache Hudi.

Perubahan terminologi Hudi: Tampilan sekarang menjadi kueri

Dimulai pada rilis versi 0.5.1, Apache Hudi mengubah beberapa terminologi. Yang sebelumnya tampilan kini disebut kueri dalam rilis berikutnya. Tabel berikut merangkum perubahan antara istilah lama dan baru.

Istilah lama	Istilah baru
CoW: tampilan dioptimalkan baca MoR: tampilan waktu nyata	Kueri snapshot
Tampilan tambahan	Kueri tambahan
Tampilan dioptimalkan baca MoR	Kueri dioptimalkan baca

Tabel dari operasi bootstrap

Mulai di Apache Hudi versi 0.6.0, fitur operasi bootstrap memberikan performa yang lebih baik dengan set data Parquet yang ada. Sebagai ganti menulis ulang set data, operasi bootstrap dapat menghasilkan metadata saja, meninggalkan set data di tempat.

Anda dapat menggunakan Athena untuk mengkueri tabel dari operasi bootstrap seperti tabel lain berdasarkan data di Amazon S3. Di pernyataan `CREATE TABLE`, tentukan jalur tabel Hudi Anda di klausa `LOCATION`.

Untuk informasi lebih lanjut tentang membuat tabel Hudi menggunakan operasi bootstrap di Amazon EMR, lihat artikel Fitur baru dari Apache Hudi tersedia di Amazon AWS EMR di Big Data Blog.

Daftar metadata Hudi

Apache Hudi memiliki tabel metadata yang berisi fitur pengindeksan untuk meningkatkan kinerja seperti daftar file, melewatkan data menggunakan statistik kolom, dan indeks berbasis filter mekar.

Dari fitur-fitur ini, Athena saat ini hanya mendukung indeks daftar file. Indeks daftar file menghilangkan panggilan sistem file seperti “daftar file” dengan mengambil informasi dari indeks yang memelihara partisi ke pemetaan file. Ini menghilangkan kebutuhan untuk daftar secara rekursif setiap partisi di bawah jalur tabel untuk mendapatkan tampilan sistem file. Saat Anda bekerja dengan kumpulan data besar, pengindeksan ini secara drastis mengurangi latensi yang seharusnya terjadi saat mendapatkan daftar file selama penulisan dan kueri. Ini juga menghindari kemacetan seperti pembatasan batas permintaan pada panggilan Amazon S3. `LIST`

> **Note**
>
> Athena tidak mendukung lompatan data atau pengindeksan filter mekar saat ini.

Mengaktifkan tabel metadata Hudi

Daftar file berbasis tabel metadata dinonaktifkan secara default. Untuk mengaktifkan tabel metadata Hudi dan fungsionalitas daftar file terkait, atur properti `hudi.metadata-listing-enabled` tabel ke. `TRUE`

Contoh

`ALTER TABLE SET TBLPROPERTIES`Contoh berikut memungkinkan tabel metadata pada tabel contoh`partition_cow`.

```
ALTER TABLE partition_cow SET TBLPROPERTIES('hudi.metadata-listing-enabled'='TRUE')
```

Pertimbangan dan batasan

- Athena tidak mendukung kueri tambahan.
- Athena tidak mendukung data CTAS atauINSERT INTO Hudi. Jika Anda ingin dukungan Athena untuk menulis set data Hudi, kirim umpan balik ke `<athena-feedback@amazon.com>`.

  Untuk informasi selengkapnya tentang penulisan data Hudi, lihat sumber daya berikut:

  - Bekerja dengan kumpulan data Hudi di Panduan Rilis EMR Amazon.
  - Menulis Data dalam dokumentasi Apache Hudi.
- Menggunakan MSCK REPAIR TABLE pada tabel Hudi Athena tidak didukung. Jika Anda perlu memuat tabel Hudi yang tidak dibuat AWS Glue, gunakanALTER TABLE ADD PARTITION.
- Melewatkan objek S3 Glacier tidak didukung - Jika objek di tabel Apache Hudi berada dalam kelas penyimpanan Amazon S3 Glacier, menyetel properti tabel agar tidak berpengaruh. `read_restored_glacier_objects` `false`

  Misalnya, Anda mengeluarkan perintah berikut:

  ```
  ALTER TABLE table_name SET TBLPROPERTIES ('read_restored_glacier_objects' = 'false')
  ```

Untuk tabel Iceberg dan Delta Lake, perintah menghasilkan kesalahan Kunci properti tabel Tidak didukung: read_restored_glacier_objects. Untuk tabel Hudi, `ALTER TABLE` perintah tidak menghasilkan kesalahan, tetapi objek Amazon S3 Glacier masih belum dilewati. Menjalankan `SELECT` kueri setelah `ALTER TABLE` perintah terus mengembalikan semua objek.

Sumber daya tambahan

Untuk sumber daya tambahan tentang penggunaan Apache Hudi dengan Athena, lihat sumber daya berikut.

Video

Video berikut menunjukkan bagaimana Anda dapat menggunakan Amazon Athena untuk mengkueri set data Apache dioptimalkan baca dalam danau data berbasis Amazon S3 Anda.

Kueri kumpulan data Apache Hudi menggunakan Amazon Athena

Unggahan blog

Posting Blog AWS Big Data berikut mencakup deskripsi tentang bagaimana Anda dapat menggunakan Apache Hudi dengan Athena.

- Gunakan AWS Data Exchange untuk membagikan kumpulan data Apache Hudi dengan mulus
- Membuat data lake near-real-time transaksional berbasis Apache Hudi menggunakan, Amazon AWS DMS Kinesis, AWS Glue streaming ETL, dan visualisasi data menggunakan Amazon QuickSight

Membuat tabel Hudi

Bagian ini memberikan contoh pernyataan CREATE TABLE di Athena untuk tabel data Hudi dipartisi dan tidak dipartisi.

Jika Anda memiliki tabel Hudi yang sudah dibuat AWS Glue, Anda dapat menanyakannya langsung di Athena. Saat Anda membuat tabel Hudi yang dipartisi di Athena, Anda harus menjalankan `ALTER TABLE ADD PARTITION` untuk memuat data Hudi sebelum Anda dapat menanyakannya.

Copy on write (CoW) buat contoh tabel

Tabel CoW yang tidak dipartisi

Contoh berikut membuat tabel CoW tidak dipartisi di Athena.

```
CREATE EXTERNAL TABLE `non_partition_cow`(
  `_hoodie_commit_time` string,
  `_hoodie_commit_seqno` string,
  `_hoodie_record_key` string,
  `_hoodie_partition_path` string,
  `_hoodie_file_name` string,
  `event_id` string,
  `event_time` string,
  `event_name` string,
  `event_guests` int,
  `event_type` string)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
  'org.apache.hudi.hadoop.HoodieParquetInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/folder/non_partition_cow/'
```

Tabel CoW yang dipartisi

Contoh berikut membuat tabel CoW dipartisi di Athena.

```
CREATE EXTERNAL TABLE `partition_cow`(
  `_hoodie_commit_time` string,
  `_hoodie_commit_seqno` string,
  `_hoodie_record_key` string,
  `_hoodie_partition_path` string,
  `_hoodie_file_name` string,
  `event_id` string,
  `event_time` string,
  `event_name` string,
  `event_guests` int)
PARTITIONED BY (
  `event_type` string)
ROW FORMAT SERDE
```

```
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
  'org.apache.hudi.hadoop.HoodieParquetInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/folder/partition_cow/'
```

Contoh `ALTER TABLE ADD PARTITION` berikut menambahkan dua partisi ke contoh tabel `partition_cow`.

```
ALTER TABLE partition_cow ADD
  PARTITION (event_type = 'one') LOCATION 's3://DOC-EXAMPLE-BUCKET/folder/
partition_cow/one/'
  PARTITION (event_type = 'two') LOCATION 's3://DOC-EXAMPLE-BUCKET/folder/
partition_cow/two/'
```

Gabungkan pada read (MoR) buat contoh tabel

Hudi membuat dua tabel di metastore untuk MoR: tabel untuk mengkueri snapshot, dan tabel untuk mengkueri dioptimalkan baca. Kedua tabel dapat dikueri. Dalam versi Hudi sebelum 0.5.1, tabel untuk mengkueri dioptimalkan baca memilini nama yang Anda tentukan saat Anda membuat tabel. Mulai versi Hudi 0.5.1, nama tabel diakhiri dengan `_ro` secara default. Nama tabel untuk mengkueri snapshot adalah nama yang Anda tentukan ditambah `_rt`.

Penggabungan yang tidak dipartisi pada tabel baca (MoR)

Contoh berikut membuat tabel MoR tidak dipartisi di Athena kueri yang dioptimalkan baca. Perhatikan bahwa dioptimalkan baca menggunakan format input `HoodieParquetInputFormat`.

```
CREATE EXTERNAL TABLE `nonpartition_mor`(
  `_hoodie_commit_time` string,
  `_hoodie_commit_seqno` string,
  `_hoodie_record_key` string,
  `_hoodie_partition_path` string,
  `_hoodie_file_name` string,
  `event_id` string,
  `event_time` string,
  `event_name` string,
  `event_guests` int,
  `event_type` string)
```

```
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
  'org.apache.hudi.hadoop.HoodieParquetInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/folder/nonpartition_mor/'
```

Contoh berikut membuat tabel MoR tidak dipartisi di Athena untuk mengkueri snapshot. Untuk kueri snapshot, gunakan format input `HoodieParquetRealtimeInputFormat`.

```
CREATE EXTERNAL TABLE `nonpartition_mor_rt`(
  `_hoodie_commit_time` string,
  `_hoodie_commit_seqno` string,
  `_hoodie_record_key` string,
  `_hoodie_partition_path` string,
  `_hoodie_file_name` string,
  `event_id` string,
  `event_time` string,
  `event_name` string,
  `event_guests` int,
  `event_type` string)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
  'org.apache.hudi.hadoop.realtime.HoodieParquetRealtimeInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/folder/nonpartition_mor/'
```

Penggabungan yang dipartisi pada tabel baca (MoR)

Contoh berikut membuat tabel MoR dipartisi di Athena untuk mengkueri dioptimalkan baca.

```
CREATE EXTERNAL TABLE `partition_mor`(
  `_hoodie_commit_time` string,
  `_hoodie_commit_seqno` string,
  `_hoodie_record_key` string,
  `_hoodie_partition_path` string,
  `_hoodie_file_name` string,
  `event_id` string,
```

```
  `event_time` string,
  `event_name` string,
  `event_guests` int)
PARTITIONED BY (
  `event_type` string)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
  'org.apache.hudi.hadoop.HoodieParquetInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/folder/partition_mor/'
```

Contoh `ALTER TABLE ADD PARTITION` berikut menambahkan dua partisi ke contoh tabel `partition_mor`.

```
ALTER TABLE partition_mor ADD
  PARTITION (event_type = 'one') LOCATION 's3://DOC-EXAMPLE-BUCKET/folder/
partition_mor/one/'
  PARTITION (event_type = 'two') LOCATION 's3://DOC-EXAMPLE-BUCKET/folder/
partition_mor/two/'
```

Contoh berikut membuat tabel MoR dipartisi di Athena untuk mengkueri snapshot.

```
CREATE EXTERNAL TABLE `partition_mor_rt`(
  `_hoodie_commit_time` string,
  `_hoodie_commit_seqno` string,
  `_hoodie_record_key` string,
  `_hoodie_partition_path` string,
  `_hoodie_file_name` string,
  `event_id` string,
  `event_time` string,
  `event_name` string,
  `event_guests` int)
PARTITIONED BY (
  `event_type` string)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
  'org.apache.hudi.hadoop.realtime.HoodieParquetRealtimeInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
```

```
LOCATION
  's3://DOC-EXAMPLE-BUCKET/folder/partition_mor/'
```

Demikian pula, contoh berikut `ALTER TABLE ADD PARTITION` menambahkan dua partisi ke tabel `partition_mor_rt` contoh.

```
ALTER TABLE partition_mor_rt ADD
  PARTITION (event_type = 'one') LOCATION 's3://DOC-EXAMPLE-BUCKET/folder/
partition_mor/one/'
  PARTITION (event_type = 'two') LOCATION 's3://DOC-EXAMPLE-BUCKET/folder/
partition_mor/two/'
```

Sumber daya tambahan

- Untuk informasi tentang penggunaan konektor AWS Glue kustom dan AWS Glue 2.0 pekerjaan untuk membuat tabel Apache Hudi yang dapat Anda kueri dengan Athena, lihat Menulis ke tabel Apache Hudi menggunakan konektor AWS Glue khusus di Blog Big Data. AWS
- Untuk artikel tentang menggunakan Apache Hudi, AWS Glue, dan Amazon Athena untuk membangun kerangka pemrosesan data untuk data lake, lihat Menyederhanakan pemrosesan data operasional di danau data AWS Glue menggunakan dan Apache Hudi di Blog Big Data. AWS

Menggunakan tabel Apache Iceberg

Athena mendukung kueri baca, perjalanan waktu, tulis, dan DDL untuk tabel Apache Iceberg yang menggunakan format Apache Parquet untuk data dan katalog untuk metastore mereka. AWS Glue

Apache Iceberg adalah format tabel terbuka untuk dataset analitik yang sangat besar. Iceberg mengelola koleksi besar file sebagai tabel, dan mendukung operasi danau data analitik modern seperti penyisipan tingkat catatan, pembaruan, penghapusan, dan kueri perjalanan waktu. Spesifikasi Iceberg memungkinkan evolusi tabel yang mulus seperti skema dan evolusi partisi dan dirancang untuk penggunaan yang dioptimalkan di Amazon S3. Iceberg juga membantu menjamin kebenaran data di bawah skenario penulisan bersamaan.

Untuk informasi lebih lanjut tentang Apache Iceberg, lihat https://iceberg.apache.org/.

Pertimbangan dan batasan

Dukungan Athena untuk tabel Iceberg memiliki pertimbangan dan batasan berikut:

- Dukungan versi Iceberg - Athena mendukung Apache Iceberg versi 1.4.2.
- Tabel dengan AWS Glue katalog saja — Hanya tabel Iceberg yang dibuat berdasarkan AWS Glue katalog berdasarkan spesifikasi yang ditentukan oleh implementasi katalog lem open source yang didukung dari Athena.
- Dukungan penguncian meja AWS Glue hanya dengan - Tidak seperti implementasi katalog Glue open source, yang mendukung penguncian khusus plug-in, Athena hanya AWS Glue mendukung penguncian optimis. Menggunakan Athena untuk memodifikasi tabel Iceberg dengan implementasi kunci lainnya akan menyebabkan potensi kehilangan data dan merusak transaksi.
- Format file yang didukung - Dukungan format file Iceberg di Athena tergantung pada versi mesin Athena, seperti yang ditunjukkan pada tabel berikut.

Versi mesin Athena	Parquet	ORC	Avro
2	Ya	Tidak	Tidak
3	Ya	Ya	Ya

- Tabel Iceberg v2 — Athena hanya membuat dan beroperasi pada tabel Iceberg v2. Untuk perbedaan antara tabel v1 dan v2, lihat Format perubahan versi dalam dokumentasi Apache Iceberg.
- Tampilan jenis waktu tanpa zona waktu - Waktu dan stempel waktu tanpa jenis zona waktu ditampilkan di UTC. Jika zona waktu tidak ditentukan dalam ekspresi filter pada kolom waktu, UTC digunakan.
- Presisi data terkait stempel waktu — Meskipun Iceberg mendukung presisi mikrodetik untuk tipe data stempel waktu, Athena hanya mendukung presisi milidetik untuk stempel waktu dalam membaca dan menulis. Untuk data dalam kolom terkait waktu yang ditulis ulang selama operasi pemadatan manual, Athena hanya mempertahankan presisi milidetik.
- Operasi yang tidak didukung - Operasi Athena berikut tidak didukung untuk tabel Iceberg.
  - ALTER TABLE SET LOCATION
- Tampilan — Gunakan `CREATE VIEW` untuk membuat tampilan Athena seperti yang dijelaskan dalam. Bekerja dengan pandangan Jika Anda tertarik menggunakan spesifikasi tampilan Gunung Es untuk membuat tampilan, hubungi athena-feedback@amazon.com.
- Perintah manajemen TTF tidak didukung di AWS Lake Formation - Meskipun Anda dapat menggunakan Lake Formation untuk mengelola izin akses baca untuk TransactionTable Format (TTF) seperti Apache Iceberg, Apache Hudi, dan Linux Foundation Delta Lake, Anda tidak dapat

menggunakan Lake Formation untuk mengelola izin untuk operasi seperti,, atau dengan format tabel ini. `VACUUM` `MERGE` `UPDATE` `OPTIMIZE` Untuk informasi selengkapnya tentang integrasi Lake Formation dengan Athena, lihat Menggunakan AWS Lake Formation dengan Amazon Athena di AWS Lake Formation Panduan Pengembang.

- Partisi dengan bidang bersarang - Partisi dengan bidang bersarang tidak didukung. *Mencoba melakukannya menghasilkan pesan NOT_SUPPORTED: Partisi dengan bidang bersarang tidak didukung: column_name.nested_field_name*.
- Melewati objek S3 Glacier tidak didukung - Jika objek di tabel Apache Iceberg berada dalam kelas penyimpanan Amazon S3 Glacier, menyetel properti tabel agar tidak berpengaruh. `read_restored_glacier_objects` `false`

  Misalnya, Anda mengeluarkan perintah berikut:

  ```
  ALTER TABLE table_name SET TBLPROPERTIES ('read_restored_glacier_objects' = 'false')
  ```

  Untuk tabel Iceberg dan Delta Lake, perintah menghasilkan kesalahan Kunci properti tabel Tidak didukung: read_restored_glacier_objects. Untuk tabel Hudi, `ALTER TABLE` perintah tidak menghasilkan kesalahan, tetapi objek Amazon S3 Glacier masih belum dilewati. Menjalankan `SELECT` kueri setelah `ALTER TABLE` perintah terus mengembalikan semua objek.

Jika Anda ingin Athena mendukung fitur tertentu, kirim umpan balik ke athena-feedback@amazon.com.

Topik

- Membuat tabel Iceberg
- Mengelola tabel Iceberg
- Menanyakan metadata tabel Iceberg
- Skema tabel Gunung Es yang Berkembang
- Menanyakan data tabel Iceberg dan melakukan perjalanan waktu
- Memperbarui data tabel Gunung Es
- Mengoptimalkan tabel Iceberg
- Tipe data yang didukung untuk tabel Iceberg di Athena
- Operasi Athena lainnya di meja Gunung Es
- Sumber daya tambahan

Membuat tabel Iceberg

Untuk membuat tabel Iceberg untuk digunakan di Athena, Anda dapat menggunakan `CREATE TABLE` pernyataan seperti yang didokumentasikan pada halaman ini, atau Anda dapat menggunakan crawler. AWS Glue

Menggunakan pernyataan CREATE TABLE

Athena membuat tabel Iceberg v2. Untuk perbedaan antara tabel v1 dan v2, lihat Format perubahan versi dalam dokumentasi Apache Iceberg.

Athena `CREATE TABLE` membuat tabel Iceberg tanpa data. Anda dapat menanyakan tabel dari sistem eksternal seperti Apache Spark secara langsung jika tabel menggunakan katalog lem sumber terbuka Iceberg. Anda tidak perlu membuat tabel eksternal.

> **Warning**
>
> Menjalankan `CREATE EXTERNAL TABLE` menghasilkan pesan kesalahan Kata kunci eksternal tidak didukung untuk jenis tabel ICEBERG.

Untuk membuat tabel Iceberg dari Athena, atur properti `'table_type'` table ke `'ICEBERG'` dalam `TBLPROPERTIES` klausa, seperti pada ringkasan sintaks berikut.

```
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] )
```

Untuk informasi tentang tipe data yang dapat Anda kueri dalam tabel Iceberg, lihat. Tipe data yang didukung untuk tabel Iceberg di Athena

Partisi

Untuk membuat tabel Iceberg dengan partisi, gunakan sintaks. `PARTITIONED BY` Kolom yang digunakan untuk partisi harus ditentukan dalam deklarasi kolom terlebih dahulu. Dalam `PARTITIONED BY` klausa, jenis kolom tidak boleh disertakan. Anda juga dapat mendefinisikan transformasi partisi dalam `CREATE TABLE` sintaks. Untuk menentukan beberapa kolom untuk partisi, pisahkan kolom dengan karakter koma (`,`), seperti pada contoh berikut.

```
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' )
```

Tabel berikut menunjukkan fungsi transformasi partisi yang tersedia.

Fungsi	Deskripsi	Jenis yang didukung
`year(ts)`	Partisi menurut tahun	`date`, `timestamp`
`month(ts)`	Partisi berdasarkan bulan	`date`, `timestamp`
`day(ts)`	Partisi berdasarkan hari	`date`, `timestamp`
`hour(ts)`	Partisi per jam	`timestamp`
`bucket(N, col)`	Partisi dengan nilai hash mod *`N bucket`*. Ini adalah konsep yang sama dengan hash bucketing untuk tabel Hive.	`int`, `long`, `decimal`, `date`, `timestamp` , `string`, `binary`
`truncate( L, col)`	*Partisi dengan nilai terpotong ke L*	`int`, `long`, `decimal`, `string`

Athena mendukung partisi tersembunyi Iceberg. Untuk informasi selengkapnya, lihat partisi tersembunyi Iceberg dalam dokumentasi Apache Iceberg.

Properti tabel

Bagian ini menjelaskan properti tabel yang dapat Anda tentukan sebagai pasangan kunci-nilai dalam `TBLPROPERTIES` klausa pernyataan. `CREATE TABLE` Athena hanya mengizinkan daftar pasangan

kunci-nilai yang telah ditentukan dalam properti tabel untuk membuat atau mengubah tabel Iceberg. Tabel berikut menunjukkan properti tabel yang dapat Anda tentukan. Untuk informasi selengkapnya tentang opsi pemadatan, lihat Mengoptimalkan tabel Iceberg di dokumentasi ini. Jika Anda ingin Athena mendukung properti konfigurasi tabel open source tertentu, kirim umpan balik ke athena-feedback@amazon.com.

format

Deskripsi	Format data file
Nilai properti yang diizinkan	Format file yang didukung dan kombinasi kompresi bervariasi menurut versi mesin Athena. Untuk informasi selengkapnya, lihat Dukungan kompresi tabel gunung es dengan format file.
Nilai default	parquet

write_compression

Deskripsi	Codec kompresi file
Nilai properti yang diizinkan	Format file yang didukung dan kombinasi kompresi bervariasi menurut versi mesin Athena. Untuk informasi selengkapnya, lihat Dukungan kompresi tabel gunung es dengan format file.
Nilai default	Kompresi tulis default bervariasi menurut versi mesin Athena. Untuk informasi selengkapnya, lihat Dukungan kompresi tabel gunung es dengan format file.

optimize_rewrite_data_file_threshold

Deskripsi	Konfigurasi spesifik pengoptimalan data. Jika ada lebih sedikit file data yang memerlukan optimasi daripada ambang batas yang diberikan, file tidak ditulis ulang. Hal ini memungkinkan akumulasi lebih banyak file data untuk menghasilkan file yang lebih dekat dengan ukuran target dan melewatkan perhitungan yang tidak perlu untuk menghemat biaya.

Nilai properti yang diizinkan	Bilangan positif Harus kurang dari 50.
Nilai default	5

optimize_rewrite_delete_file_threshold

Deskripsi	Konfigurasi spesifik pengoptimalan data. Jika ada lebih sedikit file hapus yang terkait dengan file data daripada ambang batas, file data tidak ditulis ulang. Ini memungkinkan akumulasi lebih banyak file hapus untuk setiap file data untuk menghemat biaya.
Nilai properti yang diizinkan	Bilangan positif Harus kurang dari 50.
Nilai default	2

vacuum_min_snapshots_to_keep

Deskripsi	Jumlah minimum snapshot untuk disimpan di cabang utama tabel. Nilai ini lebih diutamakan daripada properti. `vacuum_max_snapshot_age_seconds` Jika snapshot minimum yang tersisa lebih tua dari usia yang ditentukan oleh`vacuum_max_snapshot_age_seconds` , snapshot disimpan, dan nilai diabaikan`vacuum_max_snapshot_age_seconds` .
Nilai properti yang diizinkan	Bilangan positif
Nilai default	1

vacuum_max_snapshot_age_seconds

Deskripsi	Usia maksimum snapshot untuk dipertahankan di cabang utama. Nilai ini diabaikan jika sisa minimum snapshot yang ditentukan oleh `vacuum_min_snapshots_to_keep` lebih tua dari usia yang ditentukan. Properti perilaku tabel ini sesuai dengan `history.expire.max-snapshot-age-ms` properti dalam konfigurasi Apache Iceberg.
Nilai properti yang diizinkan	Bilangan positif
Nilai default	432000 detik (5 hari)

vacuum_max_metadata_files_to_keep

Deskripsi	Jumlah maksimum file metadata sebelumnya untuk disimpan di cabang utama tabel.
Nilai properti yang diizinkan	Bilangan positif
Nilai default	100

Contoh pernyataan CREATE TABLE

Contoh berikut membuat tabel Iceberg yang memiliki tiga kolom.

```
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'
)
```

BUAT TABEL SEBAGAI PILIH (CTAS)

Untuk informasi tentang membuat tabel Gunung Es menggunakan `CREATE TABLE AS` pernyataan, lihatCREATE TABLE AS, dengan perhatian khusus pada bagian tersebutProperti tabel CTAS.

Menggunakan AWS Glue crawler

Anda dapat menggunakan AWS Glue crawler untuk secara otomatis mendaftarkan tabel Iceberg Anda ke dalam. AWS Glue Data Catalog Jika Anda ingin bermigrasi dari katalog Iceberg lain, Anda dapat membuat dan menjadwalkan AWS Glue crawler dan menyediakan jalur Amazon S3 tempat tabel Iceberg berada. Anda dapat menentukan kedalaman maksimum jalur Amazon S3 yang dapat dilalui AWS Glue crawler. Setelah Anda menjadwalkan AWS Glue crawler, crawler mengekstrak informasi skema dan memperbarui AWS Glue Data Catalog dengan perubahan skema setiap kali dijalankan. AWS Glue Crawler mendukung penggabungan skema di seluruh snapshot dan memperbarui lokasi file metadata terbaru di file. AWS Glue Data Catalog Untuk informasi selengkapnya, lihat Katalog Data dan crawler di AWS Glue.

Mengelola tabel Iceberg

Athena mendukung operasi DDL tabel berikut untuk tabel Iceberg.

UBAH NAMA TABEL

Mengganti nama tabel.

Karena metadata tabel tabel Iceberg disimpan di Amazon S3, Anda dapat memperbarui database dan nama tabel tabel yang dikelola Iceberg tanpa memengaruhi informasi tabel yang mendasarinya.

Sinopsis

```
ALTER TABLE [db_name.]table_name RENAME TO [new_db_name.]new_table_name
```

Contoh

```
ALTER TABLE my_db.my_table RENAME TO my_db2.my_table2
```

MENGUBAH PROPERTI SET TABEL

Menambahkan properti ke tabel Iceberg dan menetapkan nilai yang ditetapkan.

Sesuai dengan spesifikasi Iceberg, properti tabel disimpan dalam file metadata tabel Iceberg daripada di. AWS Glue Athena tidak menerima properti tabel khusus. Lihat Properti tabel bagian untuk pasangan nilai kunci yang diizinkan. Jika Anda ingin Athena mendukung properti konfigurasi tabel open source tertentu, kirim umpan balik ke athena-feedback@amazon.com.

Sinopsis

```
ALTER TABLE [db_name.]table_name SET TBLPROPERTIES ('property_name' =
 'property_value' [ , ... ])
```

Contoh

```
ALTER TABLE iceberg_table SET TBLPROPERTIES (
  'format'='parquet',
  'write_compression'='snappy',
  'optimize_rewrite_delete_file_threshold'='10'
)
```

MENGUBAH PROPERTI TABEL YANG TIDAK DISETEL

Menjatuhkan properti yang ada dari tabel Iceberg.

Sinopsis

```
ALTER TABLE [db_name.]table_name UNSET TBLPROPERTIES ('property_name' [ , ... ])
```

Contoh

```
ALTER TABLE iceberg_table UNSET TBLPROPERTIES ('write_compression')
```

MENGGAMBARKAN TABEL

Menjelaskan informasi tabel.

Sinopsis

```
DESCRIBE [FORMATTED] [db_name.]table_name
```

Ketika `FORMATTED` opsi ditentukan, output menampilkan informasi tambahan seperti lokasi tabel dan properti.

Contoh

```
DESCRIBE iceberg_table
```

MEJA DROP

Menjatuhkan meja Gunung Es.

> **Warning**
>
> Karena tabel Iceberg dianggap tabel terkelola di Athena, menjatuhkan tabel Iceberg juga menghapus semua data dalam tabel.

Sinopsis

```
DROP TABLE [IF EXISTS] [db_name.]table_name
```

Contoh

```
DROP TABLE iceberg_table
```

TAMPILKAN TABEL BUAT

Menampilkan pernyataan `CREATE TABLE` DDL yang dapat digunakan untuk membuat ulang tabel Iceberg di Athena. Jika Athena tidak dapat mereproduksi struktur tabel (misalnya, karena properti tabel kustom ditentukan dalam tabel), kesalahan UNSUPPORTED dilemparkan.

Sinopsis

```
SHOW CREATE TABLE [db_name.]table_name
```

Contoh

```
SHOW CREATE TABLE iceberg_table
```

TAMPILKAN PROPERTI TABEL

Menampilkan satu atau lebih properti tabel dari tabel Iceberg. Hanya properti tabel yang didukung Athena yang ditampilkan.

Sinopsis

```
SHOW TBLPROPERTIES [db_name.]table_name [('property_name')]
```

Contoh

```
SHOW TBLPROPERTIES iceberg_table
```

Menanyakan metadata tabel Iceberg

Dalam SELECT kueri, Anda dapat menggunakan properti berikut setelah *table_name untuk menanyakan* metadata tabel Iceberg:

- $files - Menampilkan file data tabel saat ini.
- $manifests - Menunjukkan manifes file tabel saat ini.
- $history — Menampilkan riwayat tabel.
- $ partisi - Menunjukkan partisi tabel saat ini.
- $ snapshots - Menampilkan snapshot tabel.
- $refs — Menunjukkan referensi tabel.

Sintaksis

Pernyataan berikut mencantumkan file untuk tabel Iceberg.

```
SELECT * FROM "dbname"."tablename$files"
```

Pernyataan berikut mencantumkan manifes untuk tabel Iceberg.

```
SELECT * FROM "dbname"."tablename$manifests"
```

Pernyataan berikut menunjukkan sejarah untuk tabel Gunung Es.

```
SELECT * FROM "dbname"."tablename$history"
```

Contoh berikut menunjukkan partisi untuk tabel Iceberg.

```
SELECT * FROM "dbname"."tablename$partitions"
```

Contoh berikut mencantumkan snapshot untuk tabel Iceberg.

```
SELECT * FROM "dbname"."tablename$snapshots"
```

Contoh berikut menunjukkan referensi untuk tabel Iceberg.

```
SELECT * FROM "dbname"."tablename$refs"
```

Skema tabel Gunung Es yang Berkembang

Pembaruan skema gunung es adalah perubahan khusus metadata. Tidak ada file data yang diubah saat Anda melakukan pembaruan skema.

Format Iceberg mendukung perubahan evolusi skema berikut:

- Tambahkan - Menambahkan kolom baru ke tabel atau ke bersarang`struct`.
- Drop - Menghapus kolom yang ada dari tabel atau bersarang`struct`.
- Ganti nama - Mengganti nama kolom atau bidang yang ada di bersarang. `struct`
- Menyusun ulang - Mengubah urutan kolom.
- Jenis promosi - Memperluas jenis kolom, `struct` bidang, `map` kunci, `map` nilai, atau `list` elemen. Saat ini, kasus berikut didukung untuk tabel Iceberg:
  - bilangan bulat ke bilangan bulat besar
  - mengapung menjadi dua kali lipat
  - meningkatkan presisi tipe desimal

MENGUBAH TABEL TAMBAHKAN KOLOM

Menambahkan satu atau lebih kolom ke tabel Iceberg yang ada.

Sinopsis

```
ALTER TABLE [db_name.]table_name ADD COLUMNS (col_name data_type [,...])
```

Contoh

Contoh berikut menambahkan `comment` kolom tipe `string` ke tabel Iceberg.

```
ALTER TABLE iceberg_table ADD COLUMNS (comment string)
```

Contoh berikut menambahkan `point` kolom tipe `struct` ke tabel Iceberg.

```
ALTER TABLE iceberg_table
ADD COLUMNS (point struct<x: double, y: double>)
```

Contoh berikut menambahkan `points` kolom yang merupakan array struct ke tabel Iceberg.

```
ALTER TABLE iceberg_table
ADD COLUMNS (points array<struct<x: double, y: double>>)
```

MENGUBAH KOLOM DROP TABEL

Menjatuhkan kolom dari tabel Iceberg yang ada.

Sinopsis

```
ALTER TABLE [db_name.]table_name DROP COLUMN col_name
```

Contoh

```
ALTER TABLE iceberg_table DROP COLUMN userid
```

MENGUBAH KOLOM PERUBAHAN TABEL

Mengubah nama, jenis, urutan atau komentar kolom.

> **Note**
>
> `ALTER TABLE REPLACE COLUMNS` tidak didukung. Karena `REPLACE COLUMNS` menghapus semua kolom dan kemudian menambahkan yang baru, itu tidak didukung untuk Iceberg. `CHANGE COLUMN`adalah sintaks yang disukai untuk evolusi skema.

Sinopsis

```
ALTER TABLE [db_name.]table_name
  CHANGE [COLUMN] col_old_name col_new_name column_type
  [COMMENT col_comment] [FIRST|AFTER column_name]
```

Contoh

```
ALTER TABLE iceberg_table CHANGE comment blog_comment string AFTER id
```

TAMPILKAN KOLOM

Menampilkan kolom dalam tabel.

Sinopsis

```
SHOW COLUMNS (FROM|IN) [db_name.]table_name
```

Contoh

```
SHOW COLUMNS FROM iceberg_table
```

Menanyakan data tabel Iceberg dan melakukan perjalanan waktu

Untuk menanyakan dataset Iceberg, gunakan `SELECT` pernyataan standar seperti berikut ini. Kueri mengikuti spesifikasi Apache Iceberg format v2 dan melakukan penghapusan posisi dan merge-on-read kesetaraan.

```
SELECT * FROM [db_name.]table_name [WHERE predicate]
```

Untuk mengoptimalkan waktu kueri, semua predikat didorong ke bawah ke tempat data berada.

Perjalanan waktu dan kueri perjalanan versi

Setiap tabel Apache Iceberg mempertahankan manifes berversi dari objek Amazon S3 yang dikandungnya. Versi manifes sebelumnya dapat digunakan untuk perjalanan waktu dan kueri perjalanan versi.

Kueri perjalanan waktu di Athena menanyakan Amazon S3 untuk data historis dari snapshot yang konsisten pada tanggal dan waktu yang ditentukan. Kueri perjalanan versi di Athena menanyakan Amazon S3 untuk data historis sebagai ID snapshot yang ditentukan.

Pertanyaan perjalanan waktu

Untuk menjalankan kueri perjalanan waktu, gunakan `FOR TIMESTAMP AS OF timestamp` setelah nama tabel dalam `SELECT` pernyataan, seperti pada contoh berikut.

```
SELECT * FROM iceberg_table FOR TIMESTAMP AS OF timestamp
```

Waktu sistem yang akan ditentukan untuk bepergian adalah stempel waktu atau stempel waktu dengan zona waktu. Jika tidak ditentukan, Athena menganggap nilainya sebagai stempel waktu dalam waktu UTC.

Contoh kueri perjalanan waktu berikut memilih CloudTrail data untuk tanggal dan waktu yang ditentukan.

```
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)
```

Kueri perjalanan versi

Untuk menjalankan kueri perjalanan versi (yaitu, melihat snapshot yang konsisten pada versi tertentu), gunakan `FOR VERSION AS OF version` setelah nama tabel dalam `SELECT` pernyataan, seperti pada contoh berikut.

```
SELECT * FROM [db_name.]table_name FOR VERSION AS OF version
```

Parameter *versi* adalah ID `bigint` snapshot yang terkait dengan versi tabel Iceberg.

Contoh kueri perjalanan versi berikut memilih data untuk versi yang ditentukan.

```
SELECT * FROM iceberg_table FOR VERSION AS OF 949530903748831860
```

> **Note**
>
> `FOR SYSTEM_VERSION AS OF`Klausul `FOR SYSTEM_TIME AS OF` dan dalam mesin Athena versi 2 telah digantikan oleh `FOR TIMESTAMP AS OF` klausul `FOR VERSION AS OF` dan di Athena engine versi 3.

Mengambil ID snapshot

Anda dapat menggunakan SnapshotUtilkelas Java yang disediakan oleh Iceberg untuk mengambil ID snapshot Iceberg, seperti pada contoh berikut.

```
import org.apache.iceberg.Table;
import org.apache.iceberg.aws.glue.GlueCatalog;
import org.apache.iceberg.catalog.TableIdentifier;
import org.apache.iceberg.util.SnapshotUtil;

import java.text.SimpleDateFormat;
import java.util.Date;

Catalog catalog = new GlueCatalog();

Map<String, String> properties = new HashMap<String, String>();
properties.put("warehouse", "s3://DOC-EXAMPLE-BUCKET/my-folder");
catalog.initialize("my_catalog", properties);

Date date = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss").parse("2022/01/01 00:00:00");
long millis = date.getTime();

TableIdentifier name = TableIdentifier.of("db", "table");
Table table = catalog.loadTable(name);
long oldestSnapshotIdAfter2022 = SnapshotUtil.oldestAncestorAfter(table, millis);
```

Menggabungkan perjalanan waktu dan versi

Anda dapat menggunakan sintaks perjalanan waktu dan perjalanan versi dalam kueri yang sama untuk menentukan kondisi waktu dan versi yang berbeda, seperti pada contoh berikut.

```
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)
```

Membuat dan menanyakan tampilan dengan tabel Iceberg

Untuk membuat dan menanyakan tampilan Athena pada tabel Iceberg, gunakan `CREATE VIEW` tampilan seperti yang dijelaskan dalam. Bekerja dengan pandangan

Contoh:

```
CREATE VIEW view1 AS SELECT * FROM iceberg_table
```

```
SELECT * FROM view1
```

Jika Anda tertarik menggunakan spesifikasi tampilan Gunung Es untuk membuat tampilan, hubungi athena-feedback@amazon.com.

Bekerja dengan kontrol akses halus Lake Formation

Mesin Athena versi 3 mendukung kontrol akses berbutir halus Lake Formation dengan tabel Gunung Es, termasuk tingkat kolom dan kontrol akses keamanan tingkat baris. Kontrol akses ini berfungsi dengan kueri perjalanan waktu dan dengan tabel yang telah melakukan evolusi skema. Untuk informasi selengkapnya, lihat Kontrol akses halus Formasi Danau dan kelompok kerja Athena.

Jika Anda membuat tabel Gunung Es di luar Athena, gunakan Apache Iceberg SDK versi 0.13.0 atau lebih tinggi sehingga informasi kolom tabel Iceberg Anda terisi dalam. AWS Glue Data Catalog Jika tabel Iceberg Anda tidak berisi informasi kolom AWS Glue, Anda dapat menggunakan MENGUBAH PROPERTI SET TABEL pernyataan Athena atau Iceberg SDK terbaru untuk memperbaiki tabel dan memperbarui informasi kolom di. AWS Glue

Memperbarui data tabel Gunung Es

Data tabel gunung es dapat dikelola langsung di Athena menggunakan`INSERT`,, `UPDATE` dan kueri. `DELETE` Setiap transaksi manajemen data menghasilkan snapshot baru, yang dapat ditanyakan menggunakan perjalanan waktu. `DELETE`Pernyataan `UPDATE` dan mengikuti spesifikasi penghapusan posisi tingkat baris Iceberg format v2 dan menerapkan isolasi snapshot.

Gunakan perintah berikut untuk melakukan operasi manajemen data pada tabel Iceberg.

MASUKKAN KE

Menyisipkan data ke dalam tabel Iceberg. Athena Iceberg dikenakan biaya `INSERT INTO` yang sama dengan `INSERT INTO` kueri saat ini untuk tabel Hive eksternal dengan jumlah data yang dipindai. Untuk menyisipkan data ke dalam tabel Iceberg, gunakan sintaks berikut, di mana *`kueri`* dapat berupa atau. `VALUES (val1, val2, ...)` `SELECT (col1, col2, …) FROM` `[`*`db_name`*`.]`*`table_name`* `WHERE` *`predicate`* Untuk detail sintaks dan semantik SQL, lihat. INSERT INTO

```
INSERT INTO [db_name.]table_name [(col1, col2, …)] query
```

Contoh berikut menyisipkan nilai ke dalam tabel`iceberg_table`.

```
INSERT INTO iceberg_table VALUES (1,'a','c1')
```

```
INSERT INTO iceberg_table (col1, col2, ...) VALUES (val1, val2, ...)
```

```
INSERT INTO iceberg_table SELECT * FROM another_table
```

HAPUS

Athena Iceberg DELETE menulis posisi Iceberg menghapus file ke tabel. Ini dikenal sebagai merge-on-readpenghapusan. Berbeda dengan copy-on-writepenghapusan, merge-on-read penghapusan lebih efisien karena tidak menulis ulang data file. Ketika Athena membaca data Iceberg, ia menggabungkan posisi Iceberg menghapus file dengan file data untuk menghasilkan tampilan terbaru dari tabel. Untuk menghapus file penghapusan posisi ini, Anda dapat menjalankan tindakan pemadatan REWRITE DATA. DELETEoperasi dibebankan oleh jumlah data yang dipindai. Untuk sintaks, lihat HAPUS.

Contoh berikut menghapus baris dari `iceberg_table` yang memiliki `c3` nilai untuk`category`.

```
DELETE FROM iceberg_table WHERE category='c3'
```

UPDATE

Athena Iceberg UPDATE menulis file menghapus posisi Iceberg dan baris yang baru diperbarui sebagai file data dalam transaksi yang sama. UPDATEdapat dibayangkan sebagai kombinasi dari `INSERT INTO` dan`DELETE`. UPDATEoperasi dibebankan oleh jumlah data yang dipindai. Untuk sintaks, lihat PERBARUI.

Contoh berikut memperbarui nilai-nilai yang ditentukan dalam tabel`iceberg_table`.

```
UPDATE iceberg_table SET category='c2' WHERE category='c1'
```

BERGABUNG MENJADI

Secara kondisional memperbarui, menghapus, atau menyisipkan baris ke dalam tabel Iceberg. Sebuah pernyataan tunggal dapat menggabungkan tindakan pembaruan, menghapus, dan menyisipkan. Untuk sintaks, lihat BERGABUNG MENJADI.

> **Note**
>
> `MERGE INTO` bersifat transaksional dan hanya didukung untuk tabel Apache Iceberg di mesin Athena versi 3.

Contoh berikut menghapus semua pelanggan dari tabel `t` yang ada di tabel `s` sumber.

```
MERGE INTO accounts t USING monthly_accounts_update s
ON t.customer = s.customer
WHEN MATCHED
THEN DELETE
```

Contoh berikut memperbarui tabel target `t` dengan informasi pelanggan dari tabel sumber`s`. Untuk baris pelanggan dalam tabel `t` yang memiliki baris pelanggan yang cocok dalam tabel`s`, contoh menambah pembelian dalam tabel t. Jika tabel tidak `t` memiliki kecocokan untuk baris pelanggan dalam tabel`s`, contoh menyisipkan baris pelanggan dari tabel `s` ke dalam tabel`t`.

```
MERGE INTO accounts t USING monthly_accounts_update s
    ON (t.customer = s.customer)
    WHEN MATCHED
        THEN UPDATE SET purchases = s.purchases + t.purchases
    WHEN NOT MATCHED
        THEN INSERT (customer, purchases, address)
              VALUES(s.customer, s.purchases, s.address)
```

Contoh berikut secara kondisional memperbarui tabel target `t` dengan informasi dari tabel `s` sumber. Contoh menghapus setiap baris target yang cocok yang alamat sumbernya adalah Centreville. Untuk semua baris lain yang cocok, contoh menambahkan pembelian sumber dan menetapkan alamat target ke alamat sumber. Jika tidak ada kecocokan dalam tabel target, contoh menyisipkan baris dari tabel sumber.

```
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)
```

Mengoptimalkan tabel Iceberg

Ketika data terakumulasi ke dalam tabel Iceberg, kueri secara bertahap menjadi kurang efisien karena peningkatan waktu pemrosesan yang diperlukan untuk membuka file. Biaya komputasi tambahan dikeluarkan jika tabel berisi file hapus. Di Iceberg, hapus file menyimpan penghapusan tingkat baris, dan mesin harus menerapkan baris yang dihapus ke hasil kueri.

Untuk membantu mengoptimalkan kinerja kueri pada tabel Iceberg, Athena mendukung pemadatan manual sebagai perintah pemeliharaan tabel. Pemadatan mengoptimalkan tata letak struktural tabel tanpa mengubah konten tabel.

MENGOPTIMALKAN

`OPTIMIZE table REWRITE DATA`Tindakan pemadatan menulis ulang file data ke dalam tata letak yang lebih dioptimalkan berdasarkan ukuran dan jumlah file penghapusan terkait. Untuk detail properti sintaks dan tabel, lihatMENGOPTIMALKAN.

Contoh

Contoh berikut menggabungkan menghapus file ke dalam file data dan menghasilkan file di dekat ukuran file yang ditargetkan di mana nilai `category` adalah`c1`.

```
OPTIMIZE iceberg_table REWRITE DATA USING BIN_PACK
  WHERE category = 'c1'
```

VAKUM

`VACUUM`melakukan kedaluwarsa snapshot dan penghapusan file yatim piatu. Tindakan ini mengurangi ukuran metadata dan menghapus file yang tidak dalam keadaan tabel saat ini yang juga lebih tua dari periode retensi yang ditentukan untuk tabel. Untuk detail sintaks, lihatVAKUM.

Contoh

Contoh berikut menggunakan properti tabel untuk mengkonfigurasi tabel `iceberg_table` untuk mempertahankan tiga hari terakhir data, kemudian menggunakan `VACUUM` untuk kedaluwarsa snapshot lama dan menghapus file yatim dari tabel.

```
ALTER TABLE iceberg_table SET TBLPROPERTIES (
  'vacuum_max_snapshot_age_seconds'='259200'
```

```
)

VACUUM iceberg_table
```

Tipe data yang didukung untuk tabel Iceberg di Athena

Athena dapat menanyakan tabel Iceberg yang berisi tipe data berikut:

```
binary
boolean
date
decimal
double
float
int
list
long
map
string
struct
timestamp without time zone
```

Untuk informasi selengkapnya tentang jenis tabel Iceberg, lihat halaman skema untuk Iceberg dalam dokumentasi Apache.

Tabel berikut menunjukkan hubungan antara tipe data Athena dan tipe data tabel Iceberg.

Jenis gunung es	Tipe Athena	Catatan
`boolean`	`boolean`	
-	`tinyint`	Tidak didukung untuk tabel Iceberg di Athena.
-	`smallint`	Tidak didukung untuk tabel Iceberg di Athena.
`int`	`int`	Dalam pernyataan Athena DHTML, tipe ini adalah. `INTEGER`
`long`	`bigint`	
`double`	`double`	

Jenis gunung es	Tipe Athena	Catatan
`float`	`float`	
`decimal(P, S)`	`decimal(P, S)`	Padalah presisi, S adalah skala.
-	`char`	Tidak didukung untuk tabel Iceberg di Athena.
`string`	`string`	Dalam pernyataan Athena DHTML, tipe ini adalah. `VARCHAR`
`binary`	`binary`	
`date`	`date`	
`time`	-	Hanya stempel waktu Gunung Es (tanpa zona waktu) yang didukung untuk pernyataan Athena Iceberg DDL sepertiCREATE TABLE, tetapi semua jenis stempel waktu dapat ditanyakan melalui Athena.
`timestamp`	`timestamp`	
`timestamptz`	`timestamptz`	
`list<E>`	`array`	
`map<K,V>`	`map`	
`struct<...>`	`struct`	
`fixed(L)`	-	`fixed(L)`Jenis ini saat ini tidak didukung di Athena.

Untuk informasi selengkapnya tentang tipe data di Athena, lihat. Tipe data di Amazon Athena

Operasi Athena lainnya di meja Gunung Es

Operasi tingkat basis data

Bila Anda menggunakan DROP DATABASE dengan `CASCADE` opsi, setiap data tabel Iceberg juga dihapus. Operasi DDL berikut tidak berpengaruh pada tabel Iceberg.

- CREATE DATABASE
- ALTER DATABASE SET DBPROPERTIES
- SHOW DATABASES
- SHOW TABLES
- SHOW VIEWS

Operasi terkait partisi

Karena tabel Iceberg menggunakan partisi tersembunyi, Anda tidak harus bekerja dengan partisi fisik secara langsung. Akibatnya, tabel Iceberg di Athena tidak mendukung operasi DDL terkait partisi berikut:

- SHOW PARTITIONS
- ALTER TABLE ADD PARTITION
- ALTER TABLE DROP PARTITION
- ALTER TABLE RENAME PARTITION

Jika Anda ingin melihat evolusi partisi Iceberg di Athena, kirim umpan balik ke athena-feedback@amazon.com.

Membongkar tabel Iceberg

Tabel gunung es dapat diturunkan ke file dalam folder di Amazon S3. Untuk informasi, lihat MEMBONGKAR.

PERBAIKAN MSCK

Karena tabel Iceberg melacak informasi tata letak tabel, berjalan MSCK REPAIR TABLE seperti yang dilakukan dengan tabel Hive tidak diperlukan dan tidak didukung.

Sumber daya tambahan

Artikel berikut ada di dokumentasi Panduan AWS Preskriptif.

- Bekerja dengan tabel Apache Iceberg dengan menggunakan Amazon Athena SQL

Untuk artikel mendalam tentang penggunaan Athena dengan tabel Apache Iceberg, lihat posting berikut di Big Data Blog.AWS

- Menerapkan proses CDC tanpa server dengan Apache Iceberg menggunakan Amazon DynamoDB dan Amazon Athena
- Mempercepat rekayasa fitur ilmu data di danau data transaksional menggunakan Amazon Athena dengan Apache Iceberg
- Bangun data lake Apache Iceberg menggunakan Amazon Athena, Amazon EMR, dan AWS Glue
- Lakukan upserts di danau data menggunakan Amazon Athena dan Apache Iceberg
- Membangun data lake transaksional menggunakan Apache Iceberg, AWS Glue, dan berbagi data lintas akun menggunakan dan Amazon Athena AWS Lake Formation
- Gunakan Apache Iceberg di danau data untuk mendukung pemrosesan data tambahan
- Bangun danau data Apache Iceberg yang selaras dengan GDPR secara real-time
- Otomatiskan replikasi sumber relasional menjadi danau data transaksional dengan Apache Iceberg dan AWS Glue
- Berinteraksi dengan tabel Apache Iceberg menggunakan Amazon Athena dan izin berbutir halus lintas akun menggunakan AWS Lake Formation
- Bangun danau data transaksional tanpa server dengan Apache Iceberg, Amazon EMR Tanpa Server, dan Amazon Athena

Keamanan Amazon Athena

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menggambarkan hal ini sebagai keamanan dari cloud dan keamanan di cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang berjalan Layanan AWS di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Efektivitas keamanan kami diuji dan diverifikasi secara rutin oleh auditor pihak ketiga sebagai bagian dari program kepatuhan AWS. Untuk mempelajari tentang program kepatuhan yang berlaku untuk Athena, lihat AWS layanan dalam cakupan berdasarkan program kepatuhan.
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh Layanan AWS yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, termasuk sensitivitas data, persyaratan perusahaan, serta hukum dan peraturan yang berlaku.

Dokumentasi ini akan membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan Amazon Athena. Topik berikut menunjukkan kepada Anda cara mengonfigurasi Athena untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga akan belajar cara menggunakan Layanan AWS yang lain yang dapat membantu Anda memantau dan mengamankan sumber daya Athena Anda.

Topik

Perlindungan data di Athena

Model tanggung jawab AWS bersama model berlaku untuk perlindungan data di Amazon Athena. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Lihat informasi yang lebih lengkap tentang privasi data dalam Pertanyaan Umum Privasi Data. Lihat informasi tentang perlindungan data di Eropa di pos blog Model Tanggung Jawab Bersama dan GDPR AWS di Blog Keamanan AWS .

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensyal dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan sumber daya. AWS Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.

- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi yang lebih lengkap tentang titik akhir FIPS yang tersedia di Standar Pemrosesan Informasi Federal (FIPS) 140-2.

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan Athena atau lainnya Layanan AWS menggunakan konsol, API AWS CLI, atau AWS SDK. Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Sebagai langkah keamanan tambahan, Anda dapat menggunakan kunci konteks kondisi aws:CalledViaglobal untuk membatasi permintaan hanya yang dibuat dari Athena. Untuk informasi selengkapnya, lihat Menggunakan Athena dengan tombol konteks CalledVia .

Melindungi berbagai jenis data

Beberapa jenis data yang terlibat saat Anda menggunakan Athena untuk membuat basis data dan tabel. Tipe data ini mencakup data sumber yang disimpan di Amazon S3, metadata untuk database, dan tabel yang Anda buat saat menjalankan kueri atau AWS Glue Crawler untuk menemukan data, data hasil kueri, dan riwayat kueri. Bagian ini membahas setiap jenis data dan memberikan panduan tentang melindunginya.

- Sumber data— Anda menyimpan data untuk basis data dan tabel di Amazon S3, dan Athena tidak memodifikasinya. Untuk informasi selengkapnya, lihat Perlindungan data di Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Anda mengontrol akses ke sumber data Anda dan dapat mengenkripsi di Amazon S3. Anda dapat menggunakan Athena untukmembuat tabel berdasarkan set data terenkripsi di Amazon S3.
- Database dan metadata tabel (skema) — Athena menggunakan schema-on-read teknologi, yang berarti bahwa definisi tabel Anda diterapkan ke data Anda di Amazon S3 saat Athena

menjalankan kueri. Skema apa pun yang Anda tentukan akan disimpan secara otomatis kecuali Anda menghapusnya secara eksplisit. Di Athena, Anda dapat memodifikasi metadata Katalog Data menggunakan pernyataan DDL. Anda juga dapat menghapus definisi tabel dan skema tanpa mempengaruhi data yang mendasari disimpan di Amazon S3. Metadata untuk basis data dan tabel yang Anda gunakan di Athena disimpan dalam AWS Glue Data Catalog.

Anda dapat menentukan kebijakan akses berbutir halus ke database dan tabel yang terdaftar di AWS Glue Data Catalog using AWS Identity and Access Management (IAM). Anda juga dapatmengenkripsi metadata di AWS Glue Data Catalog. Jika Anda mengenkripsi metadata, gunakanizin untuk metadata terenkripsiUntuk akses.

- Hasil kueri dan riwayat kueri, termasuk kueri tersimpan— Hasil kueri disimpan di lokasi di Amazon S3 yang dapat Anda pilih untuk menentukan secara global, atau untuk setiap grup kerja. Jika tidak ditentukan, Athena menggunakan lokasi default dalam setiap kasus. Anda mengontrol akses ke bucket Amazon S3 tempat Anda menyimpan hasil kueri dan kueri disimpan. Selain itu, Anda dapat memilih untuk mengenkripsi hasil kueri yang Anda simpan di Amazon S3. Pengguna Anda harus memiliki izin yang sesuai untuk mengakses lokasi Amazon S3 dan mendekripsi file. Untuk informasi lebih lanjut, lihat Mengenkripsi hasil kueri Athena yang disimpan di Amazon S3 di dokumen ini.

  Athena mengekalkan riwayat pertanyaan selama 45 hari. Anda dapat melihat riwayat kueri menggunakan Athena API, di konsol, dan dengan. AWS CLI Untuk menyimpan pertanyaan selama lebih dari 45 hari, menyimpannya. Untuk melindungi akses ke kueri yang disimpan,Gunakan grup kerjadi Athena, membatasi akses ke kueri yang disimpan hanya untuk pengguna yang berwenang untuk melihatnya.

Topik

Enkripsi diam

Anda dapat menjalankan kueri di Amazon Athena pada data terenkripsi di Amazon S3 di Wilayah yang sama dan di sejumlah Wilayah. Anda juga dapat mengenkripsi hasil kueri di Amazon S3 dan data di Katalog Data AWS Glue .

Anda dapat mengenkripsi aset berikut di Athena:

- Hasil dari semua kueri di Amazon S3, yang Athena toko di lokasi yang dikenal sebagai Amazon S3 hasil lokasi. Anda dapat mengenkripsi hasil kueri disimpan di Amazon S3 apakah set data yang mendasari dienkripsi di Amazon S3 atau tidak. Untuk informasi, lihat Mengenkripsi hasil kueri Athena yang disimpan di Amazon S3.
- Data dalam Katalog AWS Glue Data. Untuk informasi, lihat Izin untuk metadata terenkripsi dalam Katalog Data AWS Glue.

> **Note**
>
> Saat Anda menggunakan Athena untuk membaca tabel terenkripsi, Athena menggunakan opsi enkripsi yang ditentukan untuk data tabel, bukan opsi enkripsi untuk hasil kueri. Jika metode atau kunci enkripsi terpisah dikonfigurasi untuk hasil kueri dan data tabel, Athena membaca data tabel tanpa menggunakan opsi enkripsi dan kunci yang digunakan untuk mengenkripsi atau mendekripsi hasil kueri.
> Namun, jika Anda menggunakan Athena untuk menyisipkan data ke dalam tabel yang memiliki data terenkripsi, Athena menggunakan konfigurasi enkripsi yang ditentukan untuk hasil kueri untuk mengenkripsi data yang dimasukkan. Misalnya, jika Anda menentukan `CSE_KMS` enkripsi untuk hasil kueri, Athena menggunakan ID AWS KMS kunci yang sama dengan yang Anda gunakan untuk enkripsi hasil kueri untuk mengenkripsi data tabel yang disisipkan. `CSE_KMS`

Topik

- Opsi enkripsi Amazon S3 yang didukung
- Izin untuk data terenkripsi di Amazon S3
- Izin untuk metadata terenkripsi dalam Katalog Data AWS Glue
- Mengenkripsi hasil kueri Athena yang disimpan di Amazon S3
- Membuat tabel berdasarkan kumpulan data terenkripsi di Amazon S3

Opsi enkripsi Amazon S3 yang didukung

Athena mendukung opsi enkripsi berikut untuk set data dan hasil kueri di Amazon S3.

Jenis enkripsi	Deskripsi	Dukungan Lintas Wilayah
SSE-S3	Enkripsi sisi server dengan kunci yang dikelola Amazon S3 (SSE-S3).	Ya
SSE-KMS	Enkripsi sisi server (SSE) dengan kunci yang dikelola pelanggan. AWS Key Management Service **Note** Dengan jenis enkripsi ini, Athena tidak mengharus kan Anda untuk menunjukkan bahwa data dienkripsi saat Anda membuat tabel.	Ya
CSE-KMS	Enkripsi sisi klien (CSE) dengan kunci yang dikelola AWS KMS pelanggan. Di Athena, opsi ini mengharuskan Anda menggunakan`CREATE TABLE` dengan`TBLPROPERTIES` Klausul yang menentukan`'has_encrypted_data'='true'` . Untuk informasi selengkapnya, lihat Membuat tabel berdasarkan kumpulan data terenkripsi di Amazon S3.	Tidak

Untuk informasi selengkapnya tentang AWS KMS enkripsi dengan Amazon S3, lihat Apa itu AWS Key Management Service dan Bagaimana Amazon Simple Storage Service (Amazon S3) menggunakan Simple Storage Service (Amazon S3) AWS KMS dalam Panduan Pengembang.AWS Key Management Service Untuk informasi selengkapnya tentang penggunaan SSE-KMS atau CSE-KMS dengan Athena, lihat Peluncuran: Amazon Athena menambahkan dukungan untuk kueri data terenkripsi dari Big Data Blog.AWS

Opsi yang tidak didukung

Opsi enkripsi berikut tidak didukung:

- Kunci yang disediakan pelanggan (SSE-C)
- Enkripsi sisi klien menggunakan kunci terkelola sisi klien.
- Kunci asimetris.

Untuk membandingkan opsi enkripsi Amazon S3, lihat Melindungi data menggunakan enkripsi di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Alat untuk enkripsi sisi klien

Untuk enkripsi sisi klien, perhatikan bahwa dua alat yang tersedia:

- Klien enkripsi Amazon S3 — Ini mengenkripsi data hanya untuk Amazon S3 dan didukung oleh Athena.
- AWS Encryption SDKSDK dapat digunakan untuk mengenkripsi data di mana saja AWS tetapi tidak secara langsung didukung oleh Athena.

Alat-alat ini tidak kompatibel, dan data yang dienkripsi menggunakan satu alat tidak dapat didekripsi oleh yang lain. Athena hanya mendukung Amazon S3 Encryption Client secara langsung. Jika Anda menggunakan SDK untuk mengenkripsi data Anda, Anda dapat menjalankan kueri dari Athena, tetapi data tersebut dikembalikan sebagai teks terenkripsi.

Jika Anda ingin menggunakan Athena untuk mengkueri data yang telah dienkripsi dengan AWS Enkripsi SDK, Anda harus mengunduh dan mendekripsi data Anda, dan kemudian mengenkripsi lagi menggunakan Amazon S3 Encryption Client.

Izin untuk data terenkripsi di Amazon S3

Tergantung pada jenis enkripsi yang Anda gunakan di Amazon S3, Anda mungkin perlu menambahkan izin, juga dikenal sebagai tindakan “Izinkan”, ke kebijakan Anda yang digunakan di Athena:

- SSE-S3— Jika Anda menggunakan SSE-S3 untuk enkripsi, pengguna Athena tidak memerlukan izin tambahan dalam kebijakan mereka. Ini cukup untuk memiliki izin Amazon S3 yang sesuai untuk lokasi Amazon S3 yang sesuai dan untuk tindakan Athena. Untuk informasi selengkapnya tentang kebijakan yang mengizinkan izin Athena dan Amazon S3 yang sesuai, lihat dan. AWS kebijakan terkelola untuk Amazon Athena Akses ke Amazon S3 dari Athena
- AWS KMS— Jika Anda menggunakan AWS KMS untuk enkripsi, pengguna Athena harus diizinkan untuk melakukan AWS KMS tindakan tertentu selain izin Athena dan Amazon S3. Anda mengizinkan tindakan ini dengan mengedit kebijakan utama untuk CMK terkelola AWS KMS pelanggan yang digunakan untuk mengenkripsi data di Amazon S3. Untuk menambahkan pengguna kunci ke kebijakan AWS KMS kunci yang sesuai, Anda dapat menggunakan AWS KMS konsol di https://console.aws.amazon.com/kms. Untuk informasi tentang cara menambahkan

pengguna ke kebijakan AWS KMS utama, lihat Mengizinkan pengguna kunci menggunakan CMK di Panduan AWS Key Management Service Pengembang.

> **Note**
>
> Advanced key policy administrator dapat menyesuaikan kebijakan kunci. `kms:Decrypt` adalah tindakan minimum yang diizinkan bagi pengguna Athena untuk bekerja dengan set data terenkripsi. Untuk bekerja dengan hasil kueri terenkripsi, tindakan minimum yang diizinkan `kms:GenerateDataKey` dan `kms:Decrypt`.

Saat menggunakan Athena untuk menanyakan kumpulan data di Amazon S3 dengan sejumlah besar objek yang dienkripsi, dapat membatasi hasil kueri. AWS KMS AWS KMS Ini lebih mungkin terjadi jika ada sejumlah besar objek kecil. Athena mendukung permintaan coba lagi, tetapi kesalahan throttling mungkin masih terjadi. Jika Anda bekerja dengan sejumlah besar objek terenkripsi dan mengalami masalah ini, salah satu opsi adalah mengaktifkan kunci bucket Amazon S3 untuk mengurangi jumlah panggilan ke KMS. Untuk informasi selengkapnya, lihat Mengurangi biaya SSE-KMS dengan kunci Bucket Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Pilihan lain adalah meningkatkan kuota layanan Anda. AWS KMS Untuk informasi selengkapnya, lihat kuota Lambda in the Panduan Developer AWS Key Management Service .

Untuk informasi pemecahan masalah tentang izin saat menggunakan Amazon S3 dengan Athena, lihatIzinBagian dariPemecahan Masalah di Athenatopik.

Izin untuk metadata terenkripsi dalam Katalog Data AWS Glue

Jika Anda mengenkripsi metadata di AWS Glue Data Catalog, Anda harus menambahkan, `"kms:GenerateDataKey""kms:Decrypt"`, dan `"kms:Encrypt"` tindakan ke kebijakan yang Anda gunakan untuk mengakses Athena. Untuk informasi, lihat Akses dari Athena ke metadata terenkripsi di AWS Glue Data Catalog.

Mengenkripsi hasil kueri Athena yang disimpan di Amazon S3

Anda mengatur enkripsi hasil kueri menggunakan konsol Athena atau saat menggunakan JDBC atau ODBC. Grup kerja memungkinkan Anda untuk menegakkan enkripsi hasil kueri.

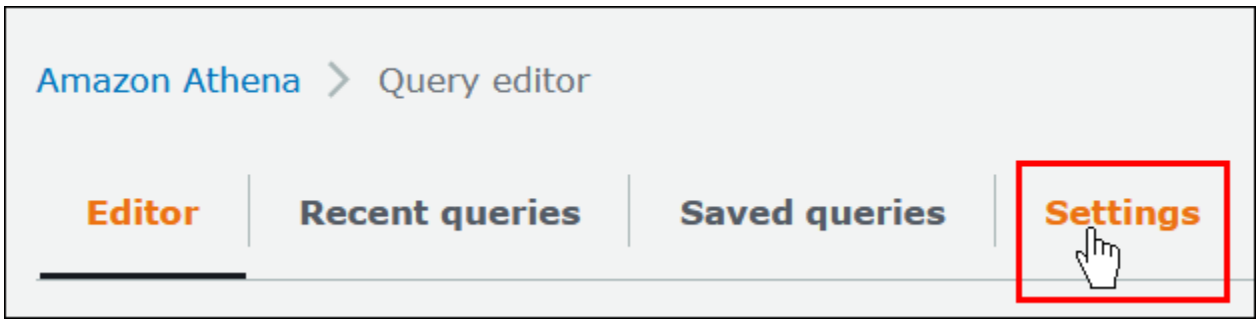
Di konsol, Anda dapat mengonfigurasi pengaturan untuk enkripsi hasil kueri dengan dua cara:

- Pengaturan sisi klien— Saat Anda menggunakanPengaturandi konsol atau operasi API untuk menunjukkan bahwa Anda ingin mengenkripsi hasil kueri, ini dikenal sebagai menggunakan pengaturan sisi klien. Pengaturan sisi klien termasuk permintaan hasil lokasi dan enkripsi. Jika Anda menentukan mereka, mereka digunakan, kecuali mereka ditimpa oleh pengaturan grup kerja.
- Pengaturan Workgroup — Bila Anda membuat atau mengedit workgroup dan memilih bidang Override client-side settings, maka semua query yang berjalan di workgroup ini menggunakan enkripsi workgroup dan pengaturan lokasi hasil kueri. Untuk informasi selengkapnya, lihat Pengaturan Workgroup mengesampingkan setelan sisi klien.

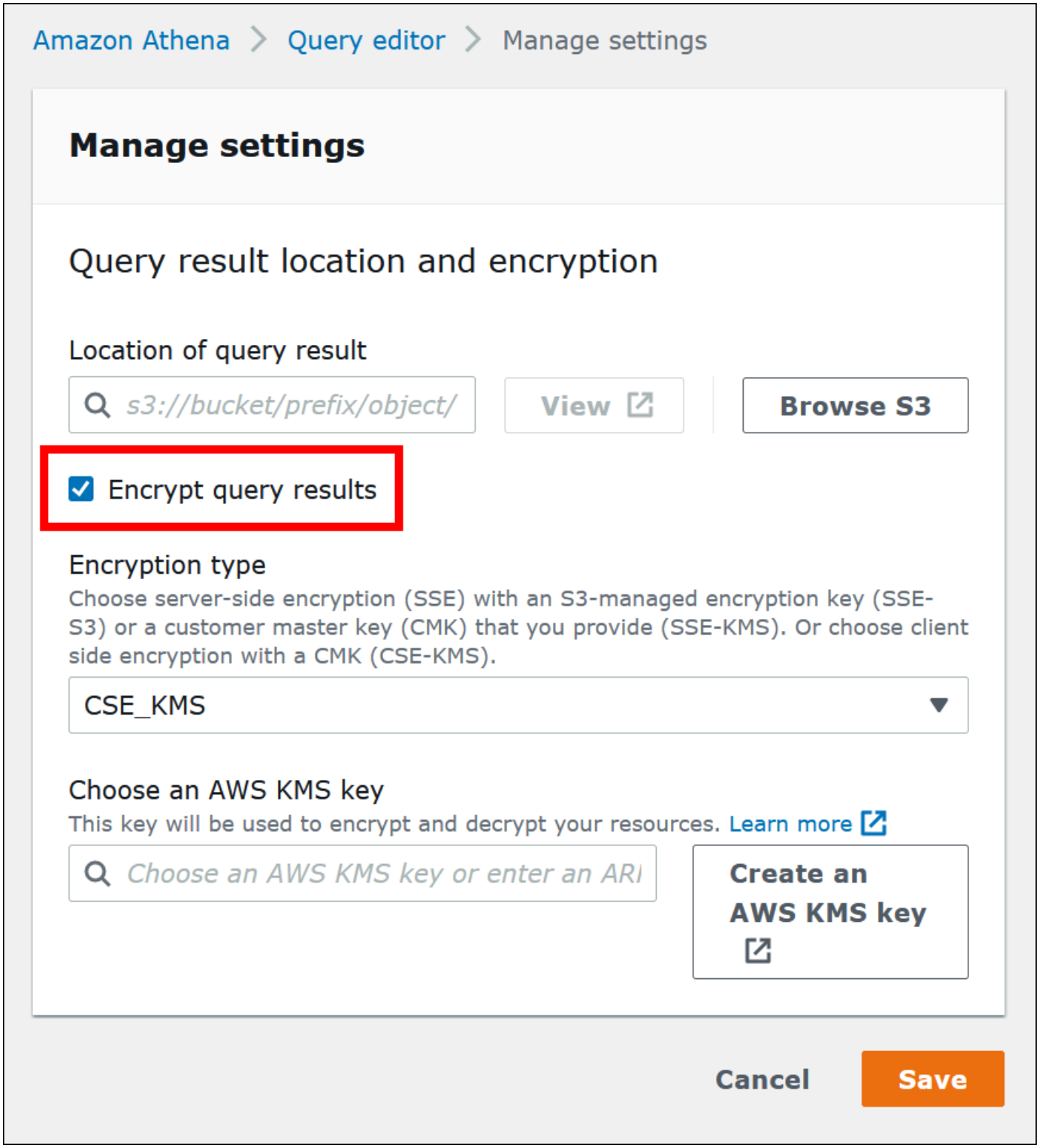
Untuk mengenkripsi hasil kueri yang disimpan di Amazon S3 menggunakan konsol

> **Important**
>
> Jika workgroup Anda memiliki bidang Override setelan sisi klien yang dipilih, maka semua kueri di workgroup menggunakan pengaturan workgroup. Konfigurasi enkripsi dan lokasi hasil kueri yang ditentukan pada tab Pengaturan di konsol Athena, oleh operasi API dan oleh driver JDBC dan ODBC tidak digunakan. Untuk informasi selengkapnya, lihat Pengaturan Workgroup mengesampingkan setelan sisi klien.

1. Dalam konsol Athena, pilih Pengaturan.



2. Pilih Kelola.
3. Untuk Lokasi hasil kueri, masukkan atau pilih jalur Amazon S3. Ini adalah lokasi Amazon S3 tempat hasil kueri disimpan.
4. Pilih Enkripsi hasil kueri.



5. UntukJenis enkripsi, pilihCSE-KMS,SSE-KM, atauSSE-S3. Dari ketiganya, CSE-KMS menawarkan tingkat enkripsi tertinggi dan SSE-S3 terendah.
6. Jika Anda memilih SSE-KMS atau CSE-KMS, tentukan kunci. AWS KMS

- Untuk Pilih AWS KMS kunci, jika akun Anda memiliki akses ke kunci terkelola AWS KMS pelanggan (CMK) yang ada, pilih aliasnya atau masukkan kunci AWS KMS ARN.
- Jika akun Anda tidak memiliki akses ke kunci terkelola pelanggan (CMK) yang ada, pilih Buat AWS KMS kunci, lalu buka AWS KMS konsol. Untuk informasi selengkapnya, lihat Membuat kunci di Panduan AWS Key Management Service Pengembang.

> **Note**
>
> Athena hanya mendukung kunci simetris untuk membaca dan menulis data.

7. Kembali ke konsol Athena dan pilih kunci yang Anda buat dengan alias atau ARN.
8. Pilih Simpan.

Mengenkripsi hasil kueri Athena saat menggunakan JDBC atau ODBC

Jika Anda terhubung menggunakan driver JDBC atau ODBC, Anda mengonfigurasi opsi driver untuk menentukan jenis enkripsi untuk menggunakan dan lokasi direktori pementasan Amazon S3. Untuk mengonfigurasi driver JDBC atau ODBC untuk mengenkripsi hasil kueri Anda menggunakan salah satu protokol enkripsi yang didukung Athena, lihat Menghubungkan ke Amazon Athena dengan driver ODBC dan JDBC.

Membuat tabel berdasarkan kumpulan data terenkripsi di Amazon S3

Saat Anda membuat tabel, menunjukkan kepada Athena bahwa set data dienkripsi di Amazon S3. Ini tidak diperlukan saat menggunakan SSE-KMS. Untuk SSE-S3 dan AWS KMS enkripsi, Athena menentukan cara mendekripsi kumpulan data dan membuat tabel, sehingga Anda tidak perlu memberikan informasi penting.

Pengguna yang menjalankan kueri, termasuk pengguna yang membuat tabel, harus memiliki izin yang dijelaskan sebelumnya dalam topik ini.

> **Important**
>
> Jika Anda menggunakan Amazon EMR bersama dengan EMRFS untuk mengunggah file Parquet terenkripsi, Anda harus menonaktifkan unggahan multipart dengan menetapkan`fs.s3n.multipart.uploads.enabledkefalse`. Jika Anda tidak melakukan ini, Athena tidak dapat menentukan panjang file Parquet

danHIVE_CANNOT_OPEN_SPLITTerjadi kesalahan. Untuk informasi lebih lanjut, lihat Konfigurasi unggahan multipart untuk Amazon S3 di Amazon EMR.

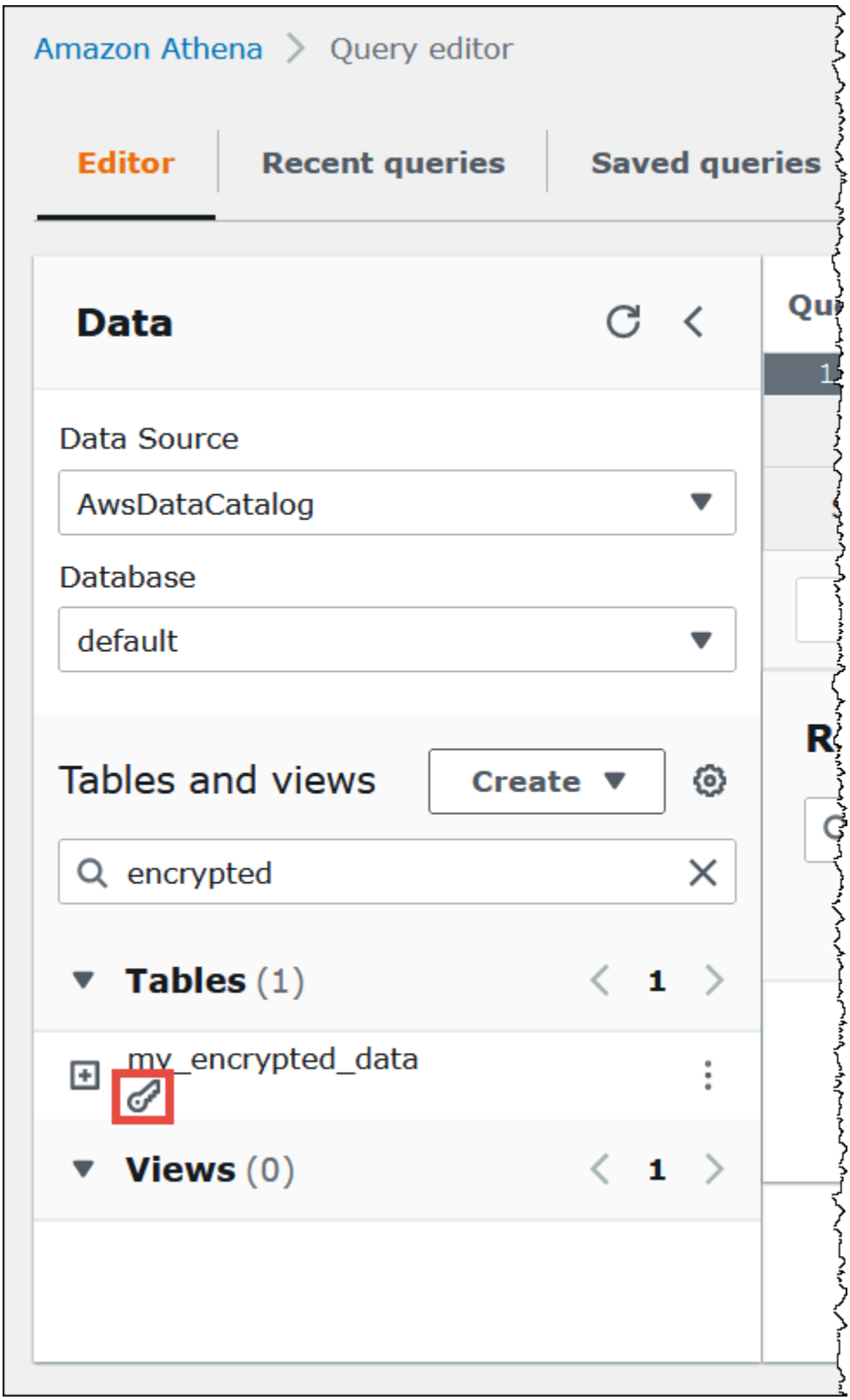
Untuk menunjukkan bahwa set data dienkripsi di Amazon S3, lakukan salah satu langkah berikut. Langkah ini tidak diperlukan jika SSE-KMS digunakan.

- DalamBUAT TABELpernyataan, gunakanTBLPROPERTIESKlausul yang menentukan`'has_encrypted_data'='true'`, seperti dalam contoh berikut.

```
CREATE EXTERNAL TABLE 'my_encrypted_data' (
   `n_nationkey` int,
   `n_name` string,
   `n_regionkey` int,
   `n_comment` string)
ROW FORMAT SERDE
   'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
   'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'
LOCATION
   's3://DOC-EXAMPLE-BUCKET/folder_with_my_encrypted_data/'
TBLPROPERTIES (
   'has_encrypted_data'='true')
```

- GunakanDriver JDBCdan mengaturTBLPROPERTIESnilai seperti yang ditunjukkan dalam contoh sebelumnya saat Anda menggunakan`statement.executeQuery()`Untuk menjalankanBUAT TABEL.
- Saat Anda menggunakan konsol Athena untuk membuat tabel menggunakan formulir dan menentukan lokasi tabel, pilih opsi Kumpulan data terenkripsi.

Dalam daftar tabel konsol Athena, tabel terenkripsi menampilkan ikon berbentuk kunci.


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

Enkripsi bergerak

Selain mengenkripsi data at rest di Amazon S3, Amazon Athena menggunakan enkripsi Transport Layer Security (TLS) untuk data in-transit antara Athena dan Amazon S3, dan antara Athena dan aplikasi pelanggan yang mengaksesnya.

Anda sebaiknya hanya mengizinkan koneksi terenkripsi melalui HTTPS (TLS) menggunakan `aws:SecureTransport condition` di kebijakan IAM bucket Amazon S3.

Hasil kueri yang streaming ke JDBC atau ODBC klien dienkripsi menggunakan TLS. Untuk informasi tentang versi terbaru dari driver JDBC dan ODBC dan dokumentasi mereka, lihatMenghubungkan ke Amazon Athena dengan JDBCdanMenghubungkan ke Amazon Athena dengan ODBC.

Untuk konektor sumber data federasi Athena, dukungan untuk enkripsi dalam perjalanan menggunakan TLS tergantung pada konektor individu. Untuk informasi, lihat dokumentasi untuk konektor sumber data individual.

Manajemen kunci

Amazon Athena mendukung AWS Key Management Service (AWS KMS) untuk mengenkripsi kumpulan data di hasil kueri Amazon S3 dan Athena. AWS KMS menggunakan kunci terkelola pelanggan (CMK) untuk mengenkripsi objek Amazon S3 Anda dan bergantung pada enkripsi amplop.

Di AWS KMS, Anda dapat melakukan tindakan berikut:

- Buat kunci
- Impor materi kunci Anda sendiri untuk CMK baru

> **Note**
>
> Athena hanya mendukung kunci simetris untuk membaca dan menulis data.

Untuk informasi selengkapnya, lihat Apa yang ada AWS Key Management Service di Panduan AWS Key Management Service Pengembang, dan Cara Amazon Simple Storage Service menggunakan AWS KMS. Untuk melihat kunci di akun Anda yang AWS membuat dan mengelola untuk Anda, di panel navigasi, pilih kunci AWS terkelola.

Jika Anda mengunggah atau mengakses objek yang dienkripsi oleh SSE-KMS, gunakan AWS Signature Version 4 untuk keamanan tambahan. Untuk informasi selengkapnya, lihat Menentukan

versi tanda tangan dalam autentikasi permintaan di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Jika beban kerja Athena mengenkripsi sejumlah besar data, Anda dapat menggunakan Amazon S3 Bucket Keys untuk mengurangi biaya. Untuk informasi selengkapnya, lihat Mengurangi biaya SSE-KMS dengan kunci Bucket Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Privasi lalu lintas antar jaringan

Lalu lintas dilindungi baik antara Athena dan aplikasi on-premise dan antara Athena dan Amazon S3. Lalu lintas antara Athena dan layanan lainnya, seperti AWS Glue dan AWS Key Management Service, menggunakan HTTPS secara default.

- Untuk lalu Athena klien dan aplikasi on-premise, hasil kueri yang streaming ke JDBC atau ODBC klien dienkripsi menggunakan Transport Layer Security (TLS).

  Anda memiliki dua opsi konektivitas antara jaringan pribadi Anda dan AWS:

  - Koneksi VPN AWS VPN Site-to-Site. Untuk informasi selengkapnya, lihat Apa itu Site-to-Site VPN AWS VPN di AWS Site-to-Site VPN Panduan Pengguna.
  - AWS Direct Connect Koneksi. Untuk informasi selengkapnya, lihat Apa itu AWS Direct Connect dalam Panduan Pengguna AWS Direct Connect .
- Untuk lalu lintas antara bucket Athena dan Amazon S3, Transport Layer Security (TLS) mengenkripsi objek di-transit antara Athena dan Amazon S3, dan antara Athena dan aplikasi pelanggan yang mengaksesnya, Anda harus mengizinkan hanya koneksi terenkripsi melalui HTTPS (TLS) menggunakan`aws:SecureTransport condition`pada kebijakan Amazon S3 bucket IAM. Meskipun Athena saat ini menggunakan titik akhir publik untuk mengakses data di bucket Amazon S3, ini tidak berarti bahwa data tersebut melintasi internet publik. Semua lalu lintas antara Athena dan Amazon S3 dirutekan melalui jaringan dan dienkripsi AWS menggunakan TLS.
- Program kepatuhan — Amazon Athena mematuhi beberapa program AWS kepatuhan, termasuk SOC, PCI, FedRAMP, dan lainnya. Untuk informasi selengkapnya, lihat Layanan AWS dalam lingkup berdasarkan program kepatuhan.

Manajemen identitas dan akses di Athena

Amazon Athena menggunakan kebijakan AWS Identity and Access Management (IAM) untuk membatasi akses ke operasi Athena. Untuk daftar lengkap izin Athena, lihat Tindakan, sumber daya, dan kunci kondisi untuk Amazon Athena di Referensi Otorisasi Layanan.

Setiap kali Anda menggunakan kebijakan IAM, pastikan bahwa Anda mengikuti praktik terbaik IAM. Untuk informasi selengkapnya tentang administrator, lihat Praktik terbaik keamanan di IAM dalam Panduan Pengguna IAM.

Izin yang diperlukan untuk menjalankan Athena kueri meliputi berikut:

- lokasi Amazon S3 tempat data yang mendasari untuk kueri disimpan. Untuk informasi selengkapnya, lihat Manajemen identitas dan akses di Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.
- Metadata dan sumber daya yang Anda simpan di AWS Glue Data Catalog, seperti database dan tabel, termasuk tindakan tambahan untuk metadata terenkripsi. Untuk informasi selengkapnya, lihat Menyiapkan izin IAM untuk AWS Glue dan Menyiapkan enkripsi AWS Glue di Panduan AWS Glue Pengembang.
- Tindakan API Athena. Untuk daftar tindakan API di Athena, lihat Tindakan dalam Referensi API Amazon Athena.

Topik berikut memberikan informasi selengkapnya tentang izin untuk area tertentu di Athena.

Topik

- Menggunakan Athena dengan tombol konteks CalledVia
- Izinkan akses ke Konektor Data Athena untuk Metastore Sarang Eksternal
- Izinkan akses fungsi Lambda ke metastores Hive eksternal
- Contoh kebijakan izin IAM untuk mengizinkan Kueri Federasi Athena
- Contoh kebijakan izin IAM untuk mengizinkan Amazon Athena User Defined Functions (UDF)
- Mengizinkan akses untuk ML dengan Athena
- Mengaktifkan akses federasi ke Athena API

AWS kebijakan terkelola untuk Amazon Athena

Kebijakan AWS terkelola adalah kebijakan mandiri yang dibuat dan dikelola oleh AWS. AWS Kebijakan terkelola dirancang untuk memberikan izin bagi banyak kasus penggunaan umum sehingga Anda dapat mulai menetapkan izin kepada pengguna, grup, dan peran.

Perlu diingat bahwa kebijakan AWS terkelola mungkin tidak memberikan izin hak istimewa paling sedikit untuk kasus penggunaan spesifik Anda karena tersedia untuk digunakan semua pelanggan. AWS Kami menyarankan Anda untuk mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola pelanggan yang khusus untuk kasus penggunaan Anda.

Anda tidak dapat mengubah izin yang ditentukan dalam kebijakan AWS terkelola. Jika AWS memperbarui izin yang ditentukan dalam kebijakan AWS terkelola, pembaruan akan memengaruhi semua identitas utama (pengguna, grup, dan peran) yang dilampirkan kebijakan tersebut. AWS kemungkinan besar akan memperbarui kebijakan AWS terkelola saat baru Layanan AWS diluncurkan atau operasi API baru tersedia untuk layanan yang ada.

Untuk informasi selengkapnya, lihat AWS kebijakan yang dikelola dalam Panduan Pengguna IAM.

Pertimbangan saat menggunakan kebijakan terkelola dengan Athena

Kebijakan terkelola mudah digunakan dan diperbarui secara otomatis dengan tindakan yang diperlukan saat layanan berkembang. Saat menggunakan kebijakan terkelola dengan Athena, ingatlah hal-hal berikut:

- Untuk mengizinkan atau menolak tindakan layanan Amazon Athena untuk diri sendiri atau pengguna lain yang menggunakan AWS Identity and Access Management (IAM), Anda melampirkan kebijakan berbasis identitas untuk utama, seperti pengguna atau grup.

- Setiap kebijakan berbasis identitas terdiri dari pernyataan yang menentukan tindakan yang diizinkan atau ditolak. Untuk informasi selengkapnya dan step-by-step petunjuk untuk melampirkan kebijakan ke pengguna, lihat Melampirkan kebijakan terkelola di Panduan Pengguna IAM. Untuk daftar tindakan, lihat bagianReferensi API Amazon Athena.
- Dikelola pelanggandaninlinekebijakan berbasis identitas memungkinkan Anda menentukan tindakan Athena yang lebih terperinci dalam kebijakan untuk menyempurnakan akses. Kami menyarankan agar Anda menggunakan`AmazonAthenaFullAccess`kebijakan sebagai titik awal dan kemudian mengizinkan atau menolak tindakan tertentu yang tercantum dalamReferensi API Amazon Athena. Untuk informasi selengkapnya tentang kebijakan sebaris, lihat Kebijakan terkelola dan kebijakan sebaris di Panduan Pengguna IAM.
- Jika Anda juga memiliki utama yang terhubung menggunakan JDBC, Anda harus memberikan mandat driver JDBC untuk aplikasi Anda. Untuk informasi selengkapnya, lihat Akses melalui koneksi JDBC dan ODBC.
- Jika Anda telah mengenkripsi Katalog AWS Glue Data, Anda harus menentukan tindakan tambahan dalam kebijakan IAM berbasis identitas untuk Athena. Untuk informasi selengkapnya, lihat Akses dari Athena ke metadata terenkripsi di AWS Glue Data Catalog.
- Jika Anda membuat dan menggunakan grup kerja, pastikan kebijakan Anda termasuk akses yang relevan ke tindakan grup kerja. Untuk informasi detail, lihat the section called “ Kebijakan IAM untuk mengakses workgroup” dan the section called “Kebijakan contoh kelompok kerja”.

AWS kebijakan terkelola: AmazonAthenaFullAccess

Kebijakan terkelola `AmazonAthenaFullAccess` memberikan akses penuh ke Athena.

Untuk memberikan akses, menambahkan izin ke pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

  Buat rangkaian izin. Ikuti instruksi di Buat rangkaian izin di Panduan Pengguna AWS IAM Identity Center .
- Pengguna yang dikelola di IAM melalui penyedia identitas:

  Buat peran untuk federasi identitas. Ikuti instruksi dalam Membuat peran untuk penyedia identitas pihak ketiga (federasi) di Panduan Pengguna IAM.
- Pengguna IAM:
  - Buat peran yang dapat diambil pengguna Anda. Ikuti instruksi dalam Membuat peran untuk pengguna IAM dalam Panduan Pengguna IAM.

- (Tidak disarankan) Pasang kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti instruksi dalam Menambahkan izin ke pengguna (konsol) dalam Panduan Pengguna IAM.

Pengelompokan izin

Parameter`AmazonAthenaFullAccess`dikelompokkan ke dalam kumpulan izin berikut.

- **`athena`**— Memungkinkan utama akses ke sumber daya Athena.
- **`glue`**— Memungkinkan akses prinsipal ke AWS Glue database, tabel, dan partisi. Ini diperlukan agar kepala sekolah dapat menggunakan AWS Glue Data Catalog dengan Athena.
- **`s3`**— Memungkinkan utama untuk menulis dan membaca hasil kueri dari Amazon S3, untuk membaca tersedia secara publik contoh data Athena yang berada di Amazon S3, dan daftar bucket. Ini diperlukan agar utama dapat menggunakan Athena untuk bekerja dengan Amazon S3.
- **`sns`**— Memungkinkan utama untuk daftar topik Amazon SNS dan mendapatkan atribut topik. Ini memungkinkan utama untuk menggunakan topik Amazon SNS dengan Athena untuk tujuan pemantauan dan peringatan.
- **`cloudwatch`**— Memungkinkan kepala sekolah untuk membuat, membaca, dan menghapus alarm. CloudWatch Untuk informasi selengkapnya, lihat Mengontrol biaya dan memantau kueri dengan CloudWatch metrik dan peristiwa.
- **`lakeformation`**— Memungkinkan kepala sekolah untuk meminta kredensil sementara untuk mengakses data di lokasi danau data yang terdaftar di Lake Formation. Untuk informasi selengkapnya, lihat Kontrol akses data yang mendasari di Panduan Pengembang AWS Lake Formation.
- **`datazone`**— Memungkinkan kepala sekolah untuk mencantumkan DataZone proyek, domain, dan lingkungan Amazon. Untuk informasi tentang penggunaan DataZone di Athena, lihat. Menggunakan Amazon DataZone di Athena
- **`pricing`**— Menyediakan akses ke AWS Billing and Cost Management. Untuk informasi selengkapnya, lihat GetProductsdi Referensi AWS Billing and Cost Management API.

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

```
            "Effect": "Allow",
            "Action": [
                "athena:*"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseGluePermissions",
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:DeleteDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:CreateTable",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:StartColumnStatisticsTaskRun",
                "glue:GetColumnStatisticsTaskRun",
                "glue:GetColumnStatisticsTaskRuns",
                "glue:GetCatalogImportStatus"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseQueryResultsPermissions",
            "Effect": "Allow",
            "Action": [
```

```
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:PutObject",
                "s3:PutBucketPublicAccessBlock"
            ],
            "Resource": [
                "arn:aws:s3:::aws-athena-query-results-*"
            ]
        },
        {
            "Sid": "BaseAthenaExamplesPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::athena-examples*"
            ]
        },
        {
            "Sid": "BaseS3BucketPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:ListAllMyBuckets"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseSNSPermissions",
            "Effect": "Allow",
            "Action": [
                "sns:ListTopics",
                "sns:GetTopicAttributes"
            ],
```

```
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseCloudWatchPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:GetMetricData"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseLakeFormationPermissions",
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseDataZonePermissions",
            "Effect": "Allow",
            "Action": [
                "datazone:ListDomains",
                "datazone:ListProjects",
                "datazone:ListAccountEnvironments"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BasePricingPermissions",
            "Effect": "Allow",
            "Action": [
                "pricing:GetProducts"
```

```
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AWS kebijakan terkelola: AWSQuicksightAthenaAccess

`AWSQuicksightAthenaAccess`memberikan akses ke tindakan yang QuickSight diperlukan Amazon untuk integrasi dengan Athena. Anda dapat melampirkan kebijakan `AWSQuicksightAthenaAccess` ke identitas IAM Anda. Lampirkan kebijakan ini hanya untuk kepala sekolah yang menggunakan Amazon dengan QuickSight Athena. Kebijakan ini mencakup beberapa tindakan untuk Athena yang baik usang dan tidak termasuk dalam API publik saat ini, atau yang digunakan hanya dengan driver JDBC dan ODBC.

Pengelompokan izin

Parameter`AWSQuicksightAthenaAccess`dikelompokkan ke dalam kumpulan izin berikut.

- **`athena`**— Memungkinkan utama untuk menjalankan kueri pada sumber daya Athena.
- **`glue`**— Memungkinkan akses prinsipal ke AWS Glue database, tabel, dan partisi. Ini diperlukan agar kepala sekolah dapat menggunakan AWS Glue Data Catalog dengan Athena.
- **`s3`**— Memungkinkan utama untuk menulis dan membaca hasil kueri dari Amazon S3.
- **`lakeformation`**— Memungkinkan kepala sekolah untuk meminta kredensil sementara untuk mengakses data di lokasi danau data yang terdaftar di Lake Formation. Untuk informasi selengkapnya, lihat Kontrol akses data yang mendasari di Panduan Pengembang 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 memperbarui kebijakan terkelola AWS

Lihat detail tentang pembaruan kebijakan AWS terkelola untuk Athena sejak layanan ini mulai melacak perubahan ini.

Perubahan	Deskripsi	Tanggal
AmazonAthenaFullAccess – Pembaruan ke kebijakan yang sudah ada	Memungkinkan Athena menggunakan AWS Glue `GetCatalogImportStatus` API yang didokumentasikan	Juni 18, 2024

Perubahan	Deskripsi	Tanggal
	secara publik untuk mengambil status impor katalog.	
AmazonAthenaFullAccess – Pembaruan ke kebijakan yang sudah ada	`datazone:ListAccountEnvironments` Izin `datazone:ListDomains` `datazone:ListProjects` ,, dan ditambahkan untuk memungkinkan pengguna Athena bekerja dengan domain, proyek, dan lingkungan DataZone Amazon. Untuk informasi selengkapnya, lihat Menggunakan Amazon DataZone di Athena.	Januari 3, 2024
AmazonAthenaFullAccess – Pembaruan ke kebijakan yang sudah ada	`glue:GetColumnStatisticsTaskRuns` Izin `glue:StartColumnStatisticsTaskRun` `glue:GetColumnStatisticsTaskRun` ,, dan ditambahkan untuk memberi Athena hak AWS Glue menelepon untuk mengambil statistik untuk fitur pengoptimal berbasis biaya. Untuk informasi selengkapnya, lihat Menggunakan pengoptimal berbasis biaya.	Januari 3, 2024

Perubahan	Deskripsi	Tanggal
AmazonAthenaFullAccess – Pembaruan ke kebijakan yang sudah ada	Athena menambahkan `pricing:GetProducts` untuk menyediakan akses ke. AWS Billing and Cost Management Untuk informasi selengkapnya, lihat GetProducts di Referensi AWS Billing and Cost Management API.	Januari 25, 2023
AmazonAthenaFullAccess – Pembaruan ke kebijakan yang sudah ada	Athena ditambahkan `cloudwatch:GetMetricData` untuk mengambil nilai metrik CloudWatch. Untuk informasi selengkapnya, lihat GetMetricData di Referensi Amazon CloudWatch API.	November 14, 2022
AmazonAthenaFullAccess dan AWSQuicksightAthenaAccess— Pembaruan kebijakan yang ada	Athena `s3:PutBucketPublicAccessBlock` untuk mengaktifkan pemblokiran akses publik pada bucket yang dibuat oleh Athena.	7 Juli 2021
Athena mulai melacak perubahan	Athena mulai melacak perubahan untuk kebijakan yang AWS dikelola.	7 Juli 2021

Akses melalui koneksi JDBC dan ODBC

Untuk mendapatkan akses Layanan AWS dan sumber daya, seperti Athena dan bucket Amazon S3, berikan kredenal driver JDBC atau ODBC ke aplikasi Anda. Jika Anda menggunakan driver JDBC atau ODBC, pastikan bahwa kebijakan izin IAM mencakup semua tindakan yang tercantum di dalamnya. AWS kebijakan terkelola: AWSQuicksightAthenaAccess

Setiap kali Anda menggunakan kebijakan IAM, pastikan bahwa Anda mengikuti praktik terbaik IAM. Untuk informasi selengkapnya tentang administrator, lihat Praktik Terbaik IAM dalam Panduan Pengguna IAM.

Metode otentikasi

Athena JDBC dan ODBC driver mendukung otentikasi berbasis SAML 2.0, termasuk penyedia identitas berikut:

- Microsoft Active Directory Federation Services (AD FS)
- Azure direktori aktif (AD)
- Okta
- PingFederate

Untuk informasi lebih lanjut, lihat panduan instalasi dan konfigurasi untuk masing-masing driver, yang dapat diunduh dalam format PDF dari halaman driver JDBC dan ODBC. Untuk informasi tambahan, lihat catatan setelahnya.

- Mengaktifkan akses federasi ke Athena API
- Menggunakan Lake Formation dan Athena JDBC dan ODBC driver untuk akses federasi ke Athena
- Mengkonfigurasi sistem masuk tunggal menggunakan ODBC, SAMP 2.0, dan Penyedia Identitas Okta

Untuk informasi tentang versi terbaru dari driver JDBC dan ODBC dan dokumentasi mereka, lihatMenghubungkan ke Amazon Athena dengan JDBCdanMenghubungkan ke Amazon Athena dengan ODBC.

Akses ke Amazon S3 dari Athena

Anda dapat memberikan akses ke lokasi Amazon S3 menggunakan kebijakan berbasis identitas, kebijakan sumber daya bucket, kebijakan titik akses, atau kombinasi apa pun di atas. Ketika aktor berinteraksi dengan Athena, izin mereka melewati Athena untuk menentukan apa yang dapat diakses Athena. Ini berarti bahwa pengguna harus memiliki izin untuk mengakses bucket Amazon S3 untuk menanyakannya dengan Athena.

Setiap kali Anda menggunakan kebijakan IAM, pastikan bahwa Anda mengikuti praktik terbaik IAM. Untuk informasi selengkapnya tentang administrator, lihat Praktik terbaik keamanan di IAM dalam Panduan Pengguna IAM.

Perhatikan bahwa permintaan ke Amazon S3 berasal dari alamat IPv4 pribadi untuk Athena, bukan IP sumber yang ditentukan. `aws:SourceIp` Untuk alasan ini, Anda tidak dapat menggunakan `aws:SourceIp` kondisi untuk menolak akses ke tindakan Amazon S3 dalam kebijakan IAM tertentu. Anda juga tidak dapat membatasi atau mengizinkan akses ke sumber daya Amazon S3 berdasarkan kunci `aws:SourceVpce` atau `aws:SourceVpc` kondisi.

> **Note**
>
> Kelompok kerja Athena yang menggunakan autentikasi Pusat Identitas IAM mengharuskan Hibah Akses S3 dikonfigurasi untuk menggunakan identitas propagasi identitas tepercaya. Untuk informasi selengkapnya, lihat Hibah Akses S3 dan identitas direktori di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Topik

Mengontrol akses ke bucket Amazon S3 dengan kebijakan berbasis identitas

Kebijakan berbasis identitas terlampir pada pengguna, grup, atau peran IAM. Kebijakan ini memungkinkan Anda menentukan apa yang dapat dilakukan oleh identitas (izinnya). Anda dapat menggunakan kebijakan berbasis identitas untuk mengontrol akses ke bucket Amazon S3 Anda.

Kebijakan berbasis identitas berikut memungkinkan `Read` dan `Write` mengakses objek dalam bucket Amazon S3 tertentu. Untuk menggunakan kebijakan ini, ganti *`teks placeholder yang dicetak miring`* dengan nilai Anda sendiri.

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

```
            "Effect": "Allow",
            "Action": ["s3:ListBucket"],
            "Resource":
                ["arn:aws:s3:::DOC-EXAMPLE-BUCKET"]
        },
        {
            "Sid": "AllObjectActions",
            "Effect": "Allow",
            "Action": "s3:*Object",
            "Resource":
                ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"]
        }
    ]
}
```

Mengontrol akses ke bucket Amazon S3 dengan kebijakan sumber daya bucket

Anda dapat menggunakan kebijakan bucket Amazon S3 untuk mengamankan akses ke objek di bucket sehingga hanya pengguna dengan izin yang sesuai yang dapat mengaksesnya. Untuk panduan cara membuat kebijakan Amazon S3, lihat Menambahkan kebijakan bucket menggunakan konsol Amazon S3 di Panduan Pengguna Amazon S3.

Contoh kebijakan izin berikut membatasi pengguna untuk membaca objek yang memiliki kunci `environment: production` tag dan nilai. Kebijakan contoh menggunakan kunci `s3:ExistingObjectTag` kondisi untuk menentukan kunci tag dan nilai.

```
{
    "Version":"2012-10-17",
    "Statement":
    [
        {
            "Principal":{"AWS":"arn:aws:iam::111122223333:role/JohnDoe"
        },
            "Effect":"Allow",
            "Action": [ "s3:GetObject", "s3:GetObjectVersion" ],
            "Resource":"arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
            "Condition":
                {
                    "StringEquals":{"s3:ExistingObjectTag/environment":"production"
                }
    ]
}
```

Untuk contoh kebijakan bucket lainnya, lihat Contoh kebijakan bucket Amazon S3 di Panduan Pengguna Amazon S3.

Jalur akses Amazon S3, alias titik akses, dan kebijakan titik akses

Jika Anda memiliki kumpulan data bersama di bucket Amazon S3, mempertahankan kebijakan bucket tunggal yang mengelola akses untuk ratusan kasus penggunaan dapat menjadi tantangan.

Titik akses bucket Amazon S3 membantu mengatasi masalah ini. Bucket dapat memiliki beberapa titik akses, masing-masing dengan kebijakan yang mengontrol akses ke bucket dengan cara yang berbeda.

Untuk setiap titik akses yang Anda buat, Amazon S3 menghasilkan alias yang mewakili titik akses. Karena alias dalam format nama bucket Amazon S3, Anda dapat menggunakan alias dalam klausa pernyataan Anda `LOCATION` `CREATE TABLE` di Athena. Akses Athena ke bucket kemudian dikontrol oleh kebijakan untuk titik akses yang diwakili alias.

Untuk informasi selengkapnya, lihat Lokasi tabel di Amazon S3 dan Menggunakan titik akses di Panduan Pengguna Amazon S3.

Menggunakan tombol CalledVia konteks

Untuk keamanan tambahan, Anda dapat menggunakan kunci konteks kondisi aws:CalledViaglobal. Kunci `aws:CalledVia` berisi daftar yang dipesan dari setiap layanan dalam rantai yang membuat permintaan atas nama utama. Dengan menentukan `athena.amazonaws.com` nama utama layanan Athena untuk `aws:CalledVia` kunci konteks, Anda dapat membatasi permintaan hanya yang dibuat dari Athena. Untuk informasi selengkapnya, lihat Menggunakan Athena dengan tombol konteks CalledVia .

Sumber daya tambahan

Untuk informasi detail dan contoh tentang cara memberikan akses Amazon S3, lihat sumber daya berikut:

- Contoh penelusuran: Mengelola akses di Panduan Pengguna Amazon S3.
- Bagaimana saya bisa memberikan akses lintas akun ke objek yang ada di bucket Amazon S3? di pusat AWS pengetahuan.
- Akses lintas akun di Athena ke ember Amazon S3.

Akses lintas akun di Athena ke ember Amazon S3

Skenario Amazon Athena yang umum memberikan akses ke pengguna di akun yang berbeda dari pemilik bucket sehingga mereka dapat melakukan kueri. Dalam kasus ini, gunakan kebijakan bucket untuk memberikan akses.

> **Note**
>
> Untuk informasi tentang akses lintas akun ke katalog AWS Glue data dari Athena, lihat. Akses lintas akun ke katalog AWS Glue data

Contoh kebijakan bucket berikut, dibuat dan diterapkan ke buckets`3://DOC-EXAMPLE-BUCKET`oleh pemilik bucket, memberikan akses ke semua pengguna dalam akun`123456789123`, yang merupakan akun yang berbeda.

```
{
   "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/*"
          ]
       }
    ]
 }
```

Untuk memberikan akses ke pengguna tertentu di akun, gantiPrincipalkunci dengan kunci yang menentukan pengguna bukanroot. Misalnya, untuk profil penggunaDave, gunakanarn:aws:iam::123456789123:user/Dave.

Akses lintas akun ke bucket yang dienkripsi dengan kunci khusus AWS KMS

Jika Anda memiliki bucket Amazon S3 yang dienkripsi dengan kunci kustom AWS Key Management Service (AWS KMS), Anda mungkin perlu memberikan akses ke bucket Amazon S3 dari akun Amazon Web Services lainnya.

Memberikan akses ke bucket AWS KMS-enkripsi di Akun A kepada pengguna di Akun B memerlukan izin berikut:

- Kebijakan bucket di Akun A harus memberikan akses ke peran yang diambil oleh Akun B.
- Kebijakan AWS KMS utama di Akun A harus memberikan akses ke peran yang diambil oleh pengguna di Akun B.
- Peran AWS Identity and Access Management (IAM) yang diasumsikan oleh Akun B harus memberikan akses ke bucket dan kunci di Akun A.

Prosedur berikut menjelaskan cara memberikan izin masing-masing.

Untuk memberikan akses ke bucket di akun a ke pengguna di akun b

- Dari Akun A,meninjau kebijakan bucket S3dan konfirmasikan bahwa ada pernyataan yang memungkinkan akses dari ID akun Akun B.

  Misalnya, kebijakan bucket berikut mengizinkan `s3:GetObject` akses ke ID akun`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"
        ]
      }
    }
  ]
}
```

Untuk memberikan akses ke pengguna di akun b dari kebijakan AWS KMS kunci di akun a

1. Dalam kebijakan AWS KMS utama untuk Akun A, berikan peran yang diasumsikan oleh izin Akun B untuk tindakan berikut:

   - `kms:Encrypt`
   - `kms:Decrypt`
   - `kms:ReEncrypt*`
   - `kms:GenerateDataKey*`
   - `kms:DescribeKey`

   Contoh berikut memberikan akses kunci ke hanya satu peran IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowUseOfTheKey",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/role_name"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

2. Dari Akun A, tinjau kebijakan [utama menggunakan tampilan AWS Management Console kebijakan].
3. Dalam kebijakan kunci, verifikasi bahwa pernyataan berikut mencantumkan akun B sebagai utama.

```
"Sid": "Allow use of the key"
```

4. Jika`"Sid": "Allow use of the key"`Pernyataan tidak ada, lakukan langkah-langkah berikut:

   a. Beralih untuk melihat kebijakan kunci [menggunakan tampilan default konsol].

   b. Tambahkan ID akun akun B sebagai akun eksternal dengan akses ke kunci.

Untuk memberikan akses ke bucket dan kunci di akun a dari peran IAM yang diasumsikan oleh akun b

1. Dari Akun B, buka konsol IAM di[https://console.aws.amazon.com/iam/].
2. Buka peran IAM yang terkait dengan pengguna di Akun B.
3. Tinjau daftar kebijakan izin yang diterapkan pada peran IAM.
4. Pastikan bahwa kebijakan diterapkan yang memberikan akses ke bucket.

   Contoh pernyataan berikut memberikan akses peran IAM ke `s3:GetObject` dan `s3:PutObject` operasi di 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. Pastikan bahwa kebijakan diterapkan yang memberikan akses ke kunci.

> **Note**
>
> Jika peran IAM yang diasumsikan oleh Akun B sudah memiliki akses administrator, maka Anda tidak perlu memberikan akses ke kunci dari kebijakan IAM pengguna.

Pernyataan contoh berikut memberikan akses peran IAM untuk menggunakan kunci. `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"
    }
  ]
}
```

Akses lintas akun ke objek bucket

Objek yang di-upload oleh akun (Account C) selain bucket memiliki akun (Account A) mungkin memerlukan eksplisit objek level ACL yang memberikan akses baca ke akun kueri (Account B). Untuk menghindari persyaratan ini, Akun C harus berperan dalam Akun A sebelum menempatkan objek di

bucket Akun A. Untuk informasi selengkapnya, lihatBagaimana cara memberikan akses lintas akun ke objek yang ada di bucket Amazon S3?.

Akses berbutir halus ke database dan tabel di AWS Glue Data Catalog

Jika Anda menggunakan AWS Glue Data Catalog dengan Amazon Athena, Anda dapat menentukan kebijakan tingkat sumber daya untuk database dan tabel objek Katalog Data yang digunakan di Athena.

> **Note**
>
> Istilah “kontrol akses berbutir halus” di sini mengacu pada keamanan tingkat database dan tabel. Untuk informasi tentang keamanan tingkat kolom, baris, dan sel, lihat Pemfilteran data dan keamanan tingkat sel di Lake Formation.

Anda menentukan izin level sumber daya di kebijakan berbasis identitas IAM.

> **Important**
>
> Bagian ini membahas izin level sumber daya dalam kebijakan berbasis identitas IAM. Kebijakan ini berbeda dengan kebijakan berbasis sumber daya. Untuk informasi selengkapnya tentang perbedaan, lihat Kebijakan berbasis identitas dan kebijakan berbasis sumber daya di Panduan Pengguna IAM.

Lihat topik berikut untuk tugas berikut:

Untuk melakukan tugas ini	Lihat topik berikut:
Membuat kebijakan IAM yang mendefinisikan akses detail ke sumber daya	Membuat kebijakan IAM di Panduan Pengguna IAM.
Pelajari tentang kebijakan berbasis	Kebijakan berbasis identitas (kebijakan IAM) dalam Panduan Pengembang.AWS Glue

Untuk melakukan tugas ini	Lihat topik berikut:
identitas IAM yang digunakan di AWS Glue	

Di bagian ini

Batasan

Pertimbangkan batasan berikut jika menggunakan kontrol akses detail dengan AWS Glue Data Catalog dan Athena:

- Kelompok kerja Athena yang mengaktifkan Pusat Identitas IAM mengharuskan Lake Formation dikonfigurasi untuk menggunakan identitas Pusat Identitas IAM. Untuk informasi selengkapnya, lihat Mengintegrasikan Pusat Identitas IAM di Panduan AWS Lake Formation Pengembang.
- Anda dapat membatasi akses hanya ke basis data dan tabel. Kontrol akses berbutir halus berlaku di level tabel dan Anda tidak dapat membatasi akses ke masing-masing partisi dalam tabel. Untuk informasi selengkapnya, lihat Partisi tabel dan versi di AWS Glue.
- AWS Glue Data Catalog Berisi sumber daya berikut:`CATALOG`,`DATABASE`,`TABLE`, dan`FUNCTION`.

> **Note**
>
> Dari daftar ini, sumber daya yang umum antara Athena dan AWS Glue Data Catalog adalah`TABLE`,`DATABASE`, dan `CATALOG` untuk setiap akun. `Function`khusus untuk AWS Glue. Untuk menghapus tindakan di Athena, Anda harus menyertakan izin untuk AWS Glue Tindakan. Lihat Contoh izin berbutir halus untuk tabel dan database.

Hierarki adalah sebagai berikut:`CATALOG`adalah leluhur dari semua`DATABASES`di setiap akun, dan masing-masing`DATABASE`adalah leluhur untuk semua`TABLES`dan`FUNCTIONS`. Misalnya,

untuk tabel bernama`table_test`yang tergolong dalam basis datadbdalam katalog di akun Anda, nenek moyangnyadbdan katalog di akun Anda. Untukdbdatabase, nenek moyangnya adalah katalog di akun Anda, dan keturunannya adalah tabel dan fungsi. Untuk informasi selengkapnya tentang struktur hirarkis sumber daya, lihatDaftar ARN dalam Katalog DatadiAWS Glue Panduan Developer.

- Untuk tindakan Athena yang tidak dihapus pada sumber daya, seperti`CREATE DATABASE`,`CREATE TABLE`,`SHOW DATABASE`,`SHOW TABLE`, atau`ALTER TABLE`, Anda memerlukan izin untuk memanggil tindakan ini pada sumber daya (tabel atau basis data) dan semua nenek moyang sumber daya dalam Katalog Data. Misalnya, untuk tabel, nenek moyangnya adalah basis data yang menjadi miliknya, dan katalog untuk akun tersebut. Untuk basis data, nenek moyangnya adalah katalog untuk akun. Lihat Contoh izin berbutir halus untuk tabel dan database.
- Untuk tindakan hapus di Athena, seperti `DROP DATABASE` atau`DROP TABLE`, Anda juga memerlukan izin untuk memanggil tindakan hapus pada semua leluhur dan keturunan sumber daya di Katalog Data. Misalnya, untuk menghapus basis data Anda memerlukan izin pada basis data, katalog, yang merupakan nenek moyangnya, dan semua tabel dan fungsi yang ditetapkan pengguna, yang merupakan keturunannya. Meja tidak memiliki keturunan. Untuk menjalankan`DROP TABLE`, Anda memerlukan izin untuk tindakan ini di atas tabel, basis data yang dimiliki, dan katalog. Lihat Contoh izin berbutir halus untuk tabel dan database.

AWS Glue akses ke katalog dan database Anda per Wilayah AWS

Agar Athena dapat bekerja dengan AWS Glue, kebijakan yang memberikan akses ke database Anda dan ke akun Anda per Wilayah AWS diperlukan. AWS Glue Data Catalog Untuk membuat database, `CreateDatabase` izin juga diperlukan. Dalam contoh kebijakan berikut, ganti nama Wilayah AWS, Akun AWS ID, dan database dengan nama Anda sendiri.

```
{
   "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"
   ]
```

```
}
```

Partisi tabel dan versi di AWS Glue

Di AWS Glue, tabel dapat memiliki partisi dan versi. Versi tabel dan partisi tidak dianggap sebagai sumber daya independen di AWS Glue. Akses ke versi tabel dan partisi diberikan dengan memberikan akses di atas tabel dan sumber leluhur untuk tabel.

Untuk tujuan kontrol akses detail, izin akses berikut berlaku:

- Kontrol akses berbutir halus berlaku di level tabel. Anda dapat membatasi akses hanya ke basis data dan tabel. Misalnya, jika Anda mengizinkan akses ke tabel dipartisi, akses ini berlaku untuk semua partisi dalam tabel. Anda tidak dapat membatasi akses ke partisi individu dalam tabel.

  > **Important**
  >
  > Untuk menjalankan tindakan AWS Glue di partisi, izin untuk tindakan partisi diperlukan di tingkat katalog, database, dan tabel. Memiliki akses ke partisi dalam tabel tidak cukup. Misalnya, untuk berjalan `GetPartitions` di atas tabel `myTable` dalam database`myDB`, Anda harus memberikan izin `glue:GetPartitions` untuk katalog, `myDB` database, dan `myTable` sumber daya.

- Kontrol akses berbutir halus tidak berlaku untuk versi tabel. Seperti halnya partisi, akses ke versi tabel sebelumnya diberikan melalui akses ke API versi tabel di AWS Glue atas tabel, dan ke leluhur tabel.

Untuk informasi tentang izin AWS Glue tindakan, lihat Izin AWS Glue API: Referensi tindakan dan sumber daya di Panduan AWS Glue Pengembang.

Contoh izin berbutir halus untuk tabel dan database

Tabel berikut mencantumkan contoh kebijakan berbasis identitas IAM yang memungkinkan akses berbutir halus ke basis data dan tabel di Athena. Kami merekomendasikan bahwa Anda mulai dengan contoh-contoh ini dan, tergantung pada kebutuhan Anda, menyesuaikan mereka untuk mengizinkan atau menolak tindakan tertentu untuk basis data tertentu dan tabel.

Contoh-contoh ini termasuk akses ke database dan katalog sehingga Athena dan dapat bekerja sama. AWS Glue Untuk beberapa AWS Wilayah, sertakan kebijakan serupa untuk setiap database dan katalog Anda, satu baris untuk setiap Wilayah.

Dalam contoh, ganti `example_db` database dan `test` tabel dengan database dan nama tabel Anda sendiri.

Pernyataan DDL	Contoh kebijakan akses IAM yang memberikan akses ke sumber daya
ALTER DATABASE	Memungkinkan Anda untuk mengubah properti untuk`example_db` basis data. `{ "Effect": "Allow", "Action": [ "glue:GetDatabase", "glue:UpdateDatabase" ], "Resource": [ "arn:aws:glue: us-east-1 :123456789012 :catalog", "arn:aws:glue: us-east-1 :123456789012 :database / example_db " ] }`
BUAT BASIS DATA	Memungkinkan Anda untuk membuat basis data bernama`example_db` . `{ "Effect": "Allow", "Action": [ "glue:GetDatabase", "glue:CreateDatabase" ], "Resource": [ "arn:aws:glue: us-east-1 :123456789012 :catalog", "arn:aws:glue: us-east-1 :123456789012 :database / example_db " ] }`
CREATE TABLE	Memungkinkan Anda untuk membuat tabel bernama`test`di`example_db` basis data.

Pernyataan DDL	Contoh kebijakan akses IAM yang memberikan akses ke sumber daya

```
{
   "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_d
b /test"
   ]
}
```

Pernyataan DDL	Contoh kebijakan akses IAM yang memberikan akses ke sumber daya

DROP DATABASE	Memungkinkan Anda untuk menjatuhkan`example_db` database, termasuk semua tabel di dalamnya.
	
	{
	"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* /*"
	]
	}

Pernyataan DDL	Contoh kebijakan akses IAM yang memberikan akses ke sumber daya

MEJA DROP	Memungkinkan Anda untuk menjatuhkan tabel dipartisi bernama`test`di`example_db` basis data. Jika tabel Anda tidak memiliki partisi, jangan sertakan tindakan partisi.
	
	{
	"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*"
	]
	}

Pernyataan DDL	Contoh kebijakan akses IAM yang memberikan akses ke sumber daya
TABEL PERBAIKAN MSCK	Memungkinkan Anda memperbarui metadata katalog setelah menambahkan partisi kompatibel Hive ke tabel yang disebutkan `test` dalam database. `example_db` `{ "Effect": "Allow", "Action": ["glue:GetDatabase", "glue:CreateDatabase", "glue:GetTable", "glue:GetPartitions", "glue:GetPartition", "glue:BatchCreatePartition"], "Resource": ["arn:aws:glue: us-east-1 :123456789012 :catalog", "arn:aws:glue: us-east-1 :123456789012 :database / example_db ", "arn:aws:glue: us-east-1 :123456789 012 :table/example_db /test"] }`
TAMPILKAN DATABASE	Memungkinkan Anda untuk daftar semua basis data di AWS Glue Data Catalog. `{ "Effect": "Allow", "Action": [ "glue:GetDatabase", "glue:GetDatabases" ], "Resource": [ "arn:aws:glue: us-east-1 :123456789012 :catalog", "arn:aws:glue: us-east-1 :123456789012 :database/*" ] }`

Pernyataan DDL	Contoh kebijakan akses IAM yang memberikan akses ke sumber daya

TAMPILKAN TABEL	Memungkinkan Anda untuk daftar semua tabel di`example_db` basis data.
	
	{
	"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* /*"
	]
	}

Akses lintas akun ke katalog AWS Glue data

Anda dapat menggunakan fitur AWS Glue katalog lintas akun Athena untuk mendaftarkan AWS Glue katalog dari akun selain milik Anda. Setelah mengonfigurasi izin IAM yang diperlukan untuk AWS Glue dan mendaftarkan katalog sebagai DataCatalogsumber daya Athena, Anda dapat menggunakan Athena untuk menjalankan kueri lintas akun. Untuk informasi tentang menggunakan konsol Athena untuk mendaftarkan katalog dari akun lain, lihat. Mendaftarkan akun AWS Glue Data Catalog dari akun lain

Untuk informasi selengkapnya tentang akses lintas akun AWS Glue, lihat Memberikan akses lintas akun di Panduan Pengembang.AWS Glue

Sebelum Anda mulai

Karena fitur ini menggunakan Athena yang adaDataCatalogSumber daya API dan fungsionalitas untuk mengaktifkan akses lintas-akun, kami menyarankan agar Anda membaca sumber daya berikut sebelum Anda memulai:

- Menghubungkan ke sumber data- Berisi topik tentang penggunaan Athena dengan AWS Glue, Hive, atau Lambda sumber katalog data.

- Contoh kebijakan Katalog Data- Menunjukkan bagaimana menulis kebijakan yang mengontrol akses ke katalog data.
- Menggunakan AWS CLI metastores with Hive- Menunjukkan cara menggunakan metastores AWS CLI with Hive, tetapi berisi kasus penggunaan yang berlaku untuk sumber data lainnya.

Pertimbangan dan batasan

Saat ini, akses AWS Glue katalog lintas akun Athena memiliki batasan sebagai berikut:

- Fitur ini hanya tersedia di Wilayah AWS mana mesin Athena versi 2 atau yang lebih baru didukung. Untuk informasi selengkapnya tentang versi mesin Aurora, lihat Pembuatan versi mesin Athena. Untuk memutakhirkan versi engine untuk workgroup, lihatMengubah versi mesin Athena.
- Ketika Anda mendaftarkan akun lain AWS Glue Data Catalog di akun Anda, Anda membuat `DataCatalog` sumber daya regional yang ditautkan ke data akun lain di Wilayah tertentu saja.
- Saat ini,`CREATE VIEW`Pernyataan yang menyertakan akun lintas akun AWS Glue katalog tidak didukung.
- Katalog yang dienkripsi menggunakan kunci AWS terkelola tidak dapat ditanyakan di seluruh akun. Untuk katalog yang ingin Anda kueri di seluruh akun, gunakan kunci terkelola pelanggan (`KMS_CMK`) sebagai gantinya. Untuk informasi tentang perbedaan antara kunci yang dikelola pelanggan dan kunci AWS terkelola, lihat Kunci dan AWS kunci pelanggan di Panduan AWS Key Management Service Pengembang.

Memulai

Dalam skenario berikut, akun “peminjam” (666666666666) ingin menjalankan `SELECT` kueri yang mengacu pada AWS Glue katalog milik akun “pemilik” (99999999999999), seperti pada contoh berikut:

```
SELECT * FROM ownerCatalog.tpch1000.customer
```

Dalam prosedur berikut, Langkah 1a dan 1b menunjukkan cara memberikan akses akun peminjam ke AWS Glue sumber daya akun pemilik, baik dari sisi peminjam maupun pemilik. Contoh memberikan akses ke database`tpch1000`dan tabel`customer`. Ubah nama contoh ini agar sesuai dengan kebutuhan Anda.

Langkah 1a: Buat peran peminjam dengan kebijakan untuk mengakses sumber daya pemilik AWS Glue

Untuk membuat peran akun peminjam dengan kebijakan untuk mengakses AWS Glue sumber daya akun pemilik, Anda dapat menggunakan konsol AWS Identity and Access Management (IAM) atau API IAM. Prosedur berikut berlaku jika Anda menggunakan konsol IAM.

Untuk membuat peran dan kebijakan peminjam untuk mengakses sumber daya akun pemilik AWS Glue

1. Masuk ke konsol IAM di https://console.aws.amazon.com/iam/ dari akun peminjam.
2. Di panel navigasi, perluas Manajemen akses, lalu pilih Kebijakan.
3. Pilih Buat kebijakan.
4. Untuk editor Kebijakan, pilih JSON.
5. Di editor kebijakan, masukkan kebijakan berikut, lalu modifikasi sesuai dengan kebutuhan Anda:

```
{
    "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. Pilih Selanjutnya.
7. Pada halaman Tinjau dan buat, untuk nama Kebijakan, masukkan nama untuk kebijakan (misalnya,**`CrossGluePolicyForBorrowerRole`**).
8. Pilih Buat kebijakan.
9. Di panel navigasi, pilih Peran.
10. Pilih Buat peran.
11. Pada halaman Pilih entitas tepercaya, pilih Akun AWS, lalu pilih Berikutnya.

12. Pada halaman Tambahkan izin, masukkan nama kebijakan yang Anda buat ke dalam kotak pencarian (misalnya,**`CrossGluePolicyForBorrowerRole`**).
13. Pilih kotak centang di samping nama kebijakan, lalu pilih Berikutnya.
14. Pada Nama, tinjau, dan buat, untuk Nama peran, masukkan nama untuk peran (misalnya, **`CrossGlueBorrowerRole`**).
15. Pilih Buat peran.

Langkah 1b: Buat kebijakan pemilik untuk memberikan AWS Glue akses ke peminjam

Untuk memberikan AWS Glue akses dari akun pemilik (99999999999999) ke peran peminjam, Anda dapat menggunakan konsol atau operasi API. AWS Glue AWS Glue PutResourcePolicy Prosedur berikut menggunakan AWS Glue konsol.

Untuk memberikan AWS Glue akses ke akun peminjam dari pemilik

1. Masuk ke AWS Glue konsol di https://console.aws.amazon.com/glue/ dari akun pemilik.
2. Di panel navigasi, perluas Katalog Data, lalu pilih Pengaturan katalog.
3. DiIzin, masukkan kebijakan seperti berikut. Untuk *`rolename`*, masukkan peran yang dibuat peminjam di Langkah 1a (misalnya,). **`CrossGlueBorrowerRole`** Jika Anda ingin meningkatkan cakupan izin, Anda dapat menggunakan karakter wild card * untuk jenis sumber daya database dan tabel.

```
{
    "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"
            ]
```

```
        }
    ]
}
```

Setelah selesai, kami sarankan Anda menggunakan AWS Glue API untuk melakukan beberapa pengujian panggilan lintas akun untuk mengonfirmasi bahwa izin dikonfigurasi seperti yang Anda harapkan.

Langkah 2: Peminjam mendaftarkan AWS Glue Data Catalog yang menjadi milik akun pemilik

Prosedur berikut menunjukkan cara menggunakan konsol Athena untuk mengonfigurasi akun Amazon Web Services pemilik sebagai sumber data. AWS Glue Data Catalog Untuk informasi tentang menggunakan operasi API alih-alih konsol untuk mendaftarkan katalog, lihatMenggunakan API untuk mendaftarkan Katalog Data Athena milik akun pemilik.

Untuk mendaftarkan AWS Glue Data Catalog milik akun lain

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.

3. Perluas Administrasi, lalu pilih Sumber data.
4. Di kanan atas, pilih Buat sumber data.
5. Pada halaman Pilih sumber data, untuk Sumber data, pilih S3 - AWS Glue Data Catalog, lalu pilih Berikutnya.

6. Pada halaman Masukkan detail sumber data, di AWS Glue Data Catalogbagian, untuk Pilih AWS Glue Data Catalog, pilih AWS Glue Data Catalog di akun lain.
7. Untuk detail sumber data, masukkan informasi berikut:
   - Nama sumber data — Masukkan nama yang ingin Anda gunakan dalam kueri SQL Anda untuk merujuk ke katalog data di akun lain.
   - Deskripsi— (Opsional) Masukkan deskripsi katalog data di akun lainnya.
   - ID Katalog— Masukkan 12 digit Amazon Web Services akun ID dari akun tempat katalog data berada. Amazon Web Services akun ID adalah ID katalog.
8. (Opsional) Perluas Tag, lalu masukkan pasangan nilai kunci yang ingin Anda kaitkan dengan sumber data. Untuk informasi selengkapnya tentang tag, lihat Menandai sumber daya Athena.
9. Pilih Selanjutnya.
10. Pada halaman Tinjau dan buat, tinjau informasi yang Anda berikan, lalu pilih Buat sumber data. Halaman detail sumber data mencantumkan database dan tag untuk katalog data yang Anda daftarkan.
11. Pilih Sumber data. Katalog data yang Anda daftarkan tercantum di kolom Nama sumber data.
12. Untuk melihat atau mengedit informasi tentang katalog data, pilih katalog, lalu pilih Tindakan, Edit.
13. Untuk menghapus katalog data baru, pilih katalog, lalu pilih Tindakan, Hapus.

Langkah 3: Peminjam mengirimkan kueri

Peminjam mengirimkan kueri yang mereferensikan katalog menggunakan katalog. basis data. sintaks *tabel*, seperti pada contoh berikut:

```
SELECT * FROM ownerCatalog.tpch1000.customer
```

Alih-alih menggunakan sintaks yang sepenuhnya memenuhi syarat, peminjam juga dapat menentukan katalog secara kontekstual dengan meneruskannya melalui. QueryExecutionContext

Izin Amazon S3 tambahan

- Jika akun peminjam menggunakan kueri Athena untuk menulis data baru ke tabel di akun pemilik, pemilik tidak akan secara otomatis memiliki akses ke data ini di Amazon S3, meskipun tabel ada di akun pemilik. Ini karena peminjam adalah pemilik objek informasi di Amazon S3 kecuali

dikonfigurasi lain. Untuk memberikan pemilik akses ke data, atur izin pada objek yang sesuai sebagai langkah tambahan.

- Operasi lintas akun DDL tertentu seperti MSCK REPAIR TABLE memerlukan izin Amazon S3. Misalnya, jika akun peminjam melakukan `MSCK REPAIR` operasi lintas akun terhadap tabel di akun pemilik yang memiliki datanya di bucket S3 akun pemilik, bucket tersebut harus memberikan izin untuk peran yang diambil oleh peminjam agar kueri berhasil.

Untuk informasi tentang pemberian izin bucket, lihat Bagaimana cara menyetel izin bucket ACL? di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Menggunakan katalog secara dinamis

Dalam beberapa kasus Anda mungkin ingin cepat melakukan pengujian terhadap lintas akun AWS Glue katalog tanpa langkah prasyarat mendaftarkannya. Anda dapat secara dinamis melakukan cross-akun kueri tanpa membuat`DataCatalog`objek sumber daya jika izin IAM dan Amazon S3 yang diperlukan dikonfigurasi dengan benar seperti yang dijelaskan sebelumnya dalam dokumen ini.

Untuk secara eksplisit referensi katalog tanpa registrasi, gunakan sintaks dalam contoh berikut:

```
SELECT * FROM "glue:arn:aws:glue:us-east-1:999999999999:catalog".tpch1000.customer
```

Gunakan format"`glue:`*`<arn>`*", tempat*`<arn>`*adalahAWS Glue Data Catalog ARNyang ingin Anda gunakan. Dalam contoh, Athena menggunakan sintaks ini untuk secara dinamis menunjuk ke katalog AWS Glue data akun 999999999999 seolah-olah Anda telah membuat objek secara terpisah untuk itu. `DataCatalog`

Catatan untuk menggunakan katalog dinamis

Jika Anda menggunakan katalog dinamis, ingat poin-poin berikut.

- Penggunaan Katalog dinamis memerlukan izin IAM yang biasanya Anda gunakan untuk operasi API Katalog Data Athena. Perbedaan utama adalah bahwa nama sumber daya Katalog Data mengikuti`glue:*`Konvensi penamaan.
- Katalog ARN harus milik Wilayah yang sama tempat kueri sedang dijalankan.
- Jika menggunakan katalog dinamis dalam kueri DDLL atau tampilan, mengelilinginya dengan lolos tanda kutip ganda (\"). Jika menggunakan katalog dinamis dalam kueri DDL, mengelilingi dengan karakter backtick (`).

Menggunakan API untuk mendaftarkan Katalog Data Athena milik akun pemilik

Alih-alih menggunakan konsol Athena seperti yang dijelaskan pada Langkah 2, Anda dapat menggunakan operasi API untuk mendaftarkan Katalog Data milik akun pemilik.

Pembuat DataCatalogsumber daya Athena harus memiliki izin yang diperlukan untuk menjalankan operasi API Athena. CreateDataCatalog Tergantung pada kebutuhan Anda, akses ke operasi API tambahan mungkin diperlukan. Untuk informasi selengkapnya, lihat Contoh kebijakan Katalog Data.

Badan `CreateDataCatalog` permintaan berikut mendaftarkan AWS Glue katalog untuk akses lintas akun:

```
# 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"
}
```

Kode contoh berikut menggunakan klien Java untuk membuat`DataCatalog`objek.

```
# 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);
```

Setelah langkah-langkah ini, peminjam akan melihat `ownerCatalog` kapan memanggil operasi ListDataCatalogsAPI.

Sumber daya tambahan

- Mendaftarkan akun AWS Glue Data Catalog dari akun lain
- Konfigurasikan akses lintas akun ke berbagi AWS Glue Data Catalog menggunakan Amazon Athena dalam AWS panduan Pola Panduan Preskriptif.
- Kueri lintas akun AWS Glue Data Catalog menggunakan Amazon Athena di Blog Big AWS Data

- Memberikan akses lintas akun di Panduan Pengembang AWS Glue

Akses dari Athena ke metadata terenkripsi di AWS Glue Data Catalog

Jika Anda menggunakan AWS Glue Data Catalog dengan Amazon Athena, Anda dapat mengaktifkan enkripsi di AWS Glue Data Catalog menggunakan AWS Glue konsol atau API. Untuk selengkapnya, lihat Mengenkripsi katalog data Anda di Panduan AWS Glue Pengembang.

Jika AWS Glue Data Catalog dienkripsi, Anda harus menambahkan tindakan berikut ke semua kebijakan yang digunakan untuk mengakses Athena:

```
{
 "Version": "2012-10-17",
 "Statement": {
 "Effect": "Allow",
     "Action": [
           "kms:GenerateDataKey",
           "kms:Decrypt",
           "kms:Encrypt"
      ],
     "Resource": "(arn of the key used to encrypt the catalog)"
   }
}
```

Setiap kali Anda menggunakan kebijakan IAM, pastikan bahwa Anda mengikuti praktik terbaik IAM. Untuk informasi selengkapnya tentang administrator, lihat Praktik Terbaik IAM dalam Panduan Pengguna IAM.

Akses ke grup kerja dan tag

Grup kerja adalah sumber daya yang dikelola oleh Athena. Oleh karena itu, jika kebijakan grup kerja Anda menggunakan tindakan yang mengambil`workgroup`sebagai masukan, Anda harus menentukan ARN grup kerja sebagai berikut, tempat*`workgroup-name`*adalah nama grup kerja Anda:

```
"Resource": [arn:aws:athena:region:AWSAcctID:workgroup/workgroup-name]
```

Misalnya, untuk grup kerja bernama`test_workgroup`di`us-west-2`Wilayah untuk akun Amazon Web Services`123456789012`, menentukan grup kerja sebagai sumber daya yang menggunakan ARN berikut:

```
"Resource":["arn:aws:athena:us-east-2:123456789012:workgroup/test_workgroup"]
```

Untuk mengakses grup kerja yang diaktifkan propagasi identitas tepercaya (TIP), pengguna IAM Identity Center harus ditetapkan ke `IdentityCenterApplicationArn` yang dikembalikan oleh respons tindakan API Athena. GetWorkGroup

- Untuk daftar kebijakan grup kerja, lihatthe section called “Kebijakan contoh kelompok kerja”.
- Untuk daftar kebijakan berbasis tanda untuk grup kerja, lihatKebijakan kontrol akses IAM berbasis tag.
- Untuk informasi selengkapnya tentang cara membuat kebijakan IAM untuk grup kerja, lihat Kebijakan IAM untuk mengakses workgroup.
- Untuk daftar lengkap tindakan Amazon Athena, lihat nama tindakan API diReferensi API Amazon Athena.
- Untuk informasi selengkapnya tentang kebijakan IAM, lihat Membuat kebijakan dengan editor visual di Panduan Pengguna IAM.

Setiap kali Anda menggunakan kebijakan IAM, pastikan bahwa Anda mengikuti praktik terbaik IAM. Untuk informasi selengkapnya tentang administrator, lihat Praktik terbaik keamanan di IAM dalam Panduan Pengguna IAM.

Izinkan akses ke pernyataan yang disiapkan

Topik ini mencakup izin IAM untuk pernyataan yang disiapkan di. Amazon Athena Setiap kali Anda menggunakan kebijakan IAM, pastikan bahwa Anda mengikuti praktik terbaik IAM. Untuk informasi selengkapnya tentang administrator, lihat Praktik Terbaik IAM dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang pernyataan siap, lihatMenggunakan kueri berparameter.

Izin IAM berikut diperlukan untuk membuat, mengelola, dan mengeksekusi pernyataan siap.

```
athena:CreatePreparedStatement
athena:UpdatePreparedStatement
athena:GetPreparedStatement
athena:ListPreparedStatements
athena:DeletePreparedStatement
```

Gunakan izin ini seperti yang ditunjukkan dalam tabel berikut.

Untuk melakukannya:	Gunakan izin ini
JalankanPREPAREkueri	`athena:StartQueryExecution` `athena:CreatePreparedStatement`
Jalankan ulangPREPAREquery untuk memperbarui pernyataan siap yang ada	`athena:StartQueryExecution` `athena:UpdatePreparedStatement`
Jalankan sebuahEXECUTEkueri	`athena:StartQueryExecution` `athena:GetPreparedStatement`
JalankanDEALLOCATE PREPAREkueri	`athena:StartQueryExecution` `athena:DeletePreparedStatement`

Contoh

Contoh berikut IAM kebijakan memberikan izin untuk mengelola dan menjalankan pernyataan siap pada akun tertentu ID dan grup kerja.

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

Menggunakan Athena dengan tombol konteks CalledVia

Ketika kepala sekolah membuat permintaan AWS, AWS mengumpulkan informasi permintaan ke dalam konteks permintaan yang mengevaluasi dan mengotorisasi permintaan. Anda dapat menggunakan elemen `Condition` dari kebijakan JSON untuk membandingkan kunci dalam konteks permintaan dengan nilai kunci yang Anda tentukan dalam kebijakan Anda. Kunci konteks syarat global adalah berbagai kunci syarat dengan prefiks `aws:`.

Kunci aws:CalledVia konteks

Anda dapat menggunakan aws:CalledVia kunci konteks syarat global untuk membandingkan layanan dalam kebijakan dengan layanan yang membuat permintaan atas nama utama IAM (pengguna atau peran). Ketika kepala sekolah membuat permintaan ke Layanan AWS, layanan itu mungkin menggunakan kredensi kepala sekolah untuk membuat permintaan berikutnya ke layanan lain. Kunci `aws:CalledVia` berisi daftar yang dipesan dari setiap layanan dalam rantai yang membuat permintaan atas nama utama.

Dengan menentukan nama utama layanan untuk kunci `aws:CalledVia` konteks, Anda dapat membuat kunci konteks Layanan AWS-spesifik. Misalnya, Anda dapat menggunakan`aws:CalledVia`kondisi kunci untuk membatasi permintaan hanya yang dibuat dari Athena. Untuk menggunakan`aws:CalledVia`kondisi kunci dalam kebijakan dengan Athena, Anda menentukan nama utama layanan Athena`athena.amazonaws.com`, seperti pada contoh berikut.

```
...
   "Condition": {
       "ForAnyValue:StringEquals": {
           "aws:CalledVia": "athena.amazonaws.com"
       }
   }
...
```

Anda dapat menggunakan`aws:CalledVia`untuk memastikan pemanggil hanya memiliki akses ke sumber daya (seperti fungsi Lambda) jika mereka memanggil sumber daya dari Athena.

> **Note**
>
> Kunci `aws:CalledVia` konteks tidak kompatibel dengan fitur propagasi identitas tepercaya.

Tambahkan kunci CalledVia konteks opsional untuk akses berbutir halus ke fungsi Lambda

Athena membutuhkan penelepon untuk memiliki`lambda:InvokeFunction`izin untuk memanggil fungsi Lambda terkait dengan kueri. Pernyataan berikut memungkinkan akses halus ke fungsi Lambda sehingga pengguna dapat menggunakan hanya Athena untuk memanggil fungsi Lambda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor3",
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": "arn:aws:lambda:us-
east-1:111122223333:function:OneAthenaLambdaFunction",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": "athena.amazonaws.com"
                }
            }
        }
    ]
}
```

Contoh berikut menunjukkan penambahan pernyataan sebelumnya ke kebijakan yang memungkinkan pengguna menjalankan dan membaca kueri gabungan. Utama yang diizinkan untuk melakukan tindakan ini dapat menjalankan kueri yang menentukan katalog Athena terkait dengan sumber data gabungan. Namun, utama tidak dapat mengakses fungsi Lambda terkait kecuali fungsi dipanggil melalui Athena.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "athena:GetWorkGroup",
                "s3:PutObject",
                "s3:GetObject",
                "athena:StartQueryExecution",
                "s3:AbortMultipartUpload",
```

```
                "athena:StopQueryExecution",
                "athena:GetQueryExecution",
                "athena:GetQueryResults",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": [
                "arn:aws:athena:*:111122223333:workgroup/WorkGroupName",
                "arn:aws:s3:::MyQueryResultsBucket/*",
                "arn:aws:s3:::MyLambdaSpillBucket/MyLambdaSpillPrefix*"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "athena:ListWorkGroups",
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action":
                [
                "s3:ListBucket",
                "s3:GetBucketLocation"
                ],
            "Resource": "arn:aws:s3:::MyLambdaSpillBucket"
        },
        {
            "Sid": "VisualEditor3",
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": [
                "arn:aws:lambda:*:111122223333:function:OneAthenaLambdaFunction",
                "arn:aws:lambda:*:111122223333:function:AnotherAthenaLambdaFunction"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": "athena.amazonaws.com"
                }
            }
        }
    ]
}
```

Untuk informasi selengkapnya tentang kunci syarat `CalledVia`, lihat AWS kunci konteks syarat global dalam Panduan Pengguna IAM.

Izinkan akses ke Konektor Data Athena untuk Metastore Sarang Eksternal

Contoh kebijakan izin dalam topik ini menunjukkan diperlukan tindakan diperbolehkan dan sumber daya yang mereka diizinkan. Periksa kebijakan ini dengan hati-hati dan memodifikasi mereka sesuai dengan kebutuhan Anda sebelum Anda melampirkan kebijakan izin yang sama untuk identitas IAM.

- Example Policy to Allow an IAM Principal to Query Data Using Athena Data Connector for External Hive Metastore
- Example Policy to Allow an IAM Principal to Create an Athena Data Connector for External Hive Metastore

Example — Izinkan prinsipal IAM untuk menanyakan data menggunakan Athena Data Connector untuk External Hive Metastore

Kebijakan berikut dilampirkan pada utama IAM selainAWS kebijakan terkelola: AmazonAthenaFullAccess, yang memberikan akses penuh ke tindakan Athena.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "lambda:GetFunction",
                "lambda:GetLayerVersion",
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:*:111122223333:function:MyAthenaLambdaFunction",
                "arn:aws:lambda:*:111122223333:function:AnotherAthenaLambdaFunction",
                "arn:aws:lambda:*:111122223333:layer:MyAthenaLambdaLayer:*"
            ]
        },
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": [
```

```
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:PutObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": "arn:aws:s3:::MyLambdaSpillBucket/MyLambdaSpillLocation"
        }
    ]
}
```

Penjelasan perizinan

Tindakan yang diizinkan	Penjelasan
`"s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket", "s3:PutObject", "s3:ListMultipartUploadParts", "s3:AbortMultipartUpload"`	`s3`tindakan memungkinkan membaca dari dan menulis ke sumber daya yang ditentukan sebagai`"arn:aws:s3::: MyLambdaSpillBucket /MyLambdaSpillLocation "`, di mana *MyLambdaSpillLocation* mengidentifikasi keranjang tumpahan yang ditentukan dalam konfigurasi fungsi atau fungsi Lambda yang dipanggil. Pengenal *sumber daya arn:aws:lambda: *: my:layer AWSAcctId :: MyAthenaLambdaLayer * * diperlukan hanya jika Anda menggunakan lapisan Lambda untuk membuat dependensi runtime khusus untuk mengurangi ukuran artefak fungsi pada waktu penerapan. Parameter`*`di posisi terakhir adalah wildcard untuk versi layer.
`"lambda:GetFunction", "lambda:GetLayerVersion", "lambda:InvokeFunction"`	Mengizinkan kueri untuk memanggil AWS Lambda fungsi yang ditentukan dalam blok. `Resource` Misalnya`aarn:aws:lambda:*: MyAWSAcctId :function : MyAthenaLambdaFunction` , di mana *MyAthenaLambdaFunction* menentuka

Tindakan yang diizinkan	Penjelasan
	n nama fungsi Lambda yang akan dipanggil. Beberapa fungsi dapat ditentukan seperti yang ditunjukkan pada contoh.

Example — Izinkan prinsipal IAM untuk membuat Konektor Data Athena untuk Metastore Sarang Eksternal

Kebijakan berikut dilampirkan pada utama IAM selainAWS kebijakan terkelola: AmazonAthenaFullAccess, yang memberikan akses penuh ke tindakan Athena.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "lambda:GetFunction",
                "lambda:ListFunctions",
                "lambda:GetLayerVersion",
                "lambda:InvokeFunction",
                "lambda:CreateFunction",
                "lambda:DeleteFunction",
                "lambda:PublishLayerVersion",
                "lambda:DeleteLayerVersion",
                "lambda:UpdateFunctionConfiguration",
                "lambda:PutFunctionConcurrency",
                "lambda:DeleteFunctionConcurrency"
            ],
            "Resource": "arn:aws:lambda:*:111122223333:
 function: MyAthenaLambdaFunctionsPrefix*"
        }
    ]
}
```

Penjelasan Izin

Memungkinkan kueri untuk memanggil AWS Lambda fungsi untuk AWS Lambda fungsi yang ditentukan dalam blok. `Resource`

Misalnya`arn:aws:lambda:*:MyAWSAcctId:function:MyAthenaLambdaFunction`, di mana *`MyAthenaLambdaFunction`*menentukan nama fungsi Lambda yang akan dipanggil. Beberapa fungsi dapat ditentukan seperti yang ditunjukkan pada contoh.

Izinkan akses fungsi Lambda ke metastores Hive eksternal

Untuk memanggil fungsi Lambda di akun Anda, Anda harus membuat peran yang memiliki izin berikut:

- `AWSLambdaVPCAccessExecutionRole`— Izin peran AWS Lambda eksekusi untuk mengelola antarmuka jaringan elastis yang menghubungkan fungsi Anda ke VPC. Pastikan bahwa Anda memiliki cukup jumlah antarmuka jaringan dan alamat IP yang tersedia.
- `AmazonAthenaFullAccess`— Kebijakan yang AmazonAthenaFullAccessdikelola memberikan akses penuh ke Athena.
- Sebuah kebijakan Amazon S3 untuk memungkinkan fungsi Lambda untuk menulis ke S3 dan untuk memungkinkan Athena untuk membaca dari S3.

Sebagai contoh, kebijakan berikut mendefinisikan izin untuk lokasi tumpahan`s3:\\mybucket\spill`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/spill"
            ]
        }
    ]
}
```

Setiap kali Anda menggunakan kebijakan IAM, pastikan bahwa Anda mengikuti praktik terbaik IAM. Untuk informasi selengkapnya tentang administrator, lihat Praktik Terbaik IAM dalam Panduan Pengguna IAM.

Membuat fungsi Lambda

Untuk membuat fungsi Lambda di akun Anda, izin pengembangan fungsi atau peran `AWSLambdaFullAccess` diperlukan. Untuk informasi selengkapnya, lihat Kebijakan IAM berbasis identitas untuk. AWS Lambda

Karena Athena menggunakan AWS Serverless Application Repository untuk membuat fungsi Lambda, superuser atau administrator yang membuat fungsi Lambda juga harus memiliki kebijakan IAM untuk mengizinkan kueri federasi Athena.

Pendaftaran katalog dan operasi API metadata

Untuk akses ke API pendaftaran katalog dan operasi API metadata, gunakan kebijakan AmazonAthenaFullAccess terkelola. Jika Anda tidak menggunakan kebijakan ini, tambahkan operasi API berikut ke kebijakan Athena Anda:

```
{
    "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": [
                "*"
            ]
        }
    ]
}
```

Doa Lambda Lintas Wilayah

Untuk memanggil fungsi Lambda di wilayah selain wilayah tempat Anda menjalankan kueri Athena, menggunakan ARN penuh dari fungsi Lambda. Secara default, Athena memanggil fungsi Lambda didefinisikan di wilayah yang sama. Jika Anda perlu untuk memanggil fungsi Lambda untuk mengakses metastore Hive di wilayah selain wilayah tempat Anda menjalankan kueri Athena, Anda harus memberikan ARN penuh fungsi Lambda.

Misalnya, anggap Anda menentukan katalogehmsdi Wilayah Eropa (Frankfurt)`eu-central-1`Untuk menggunakan fungsi Lambda berikut di US East (N. Virginia).

```
arn:aws:lambda:us-east-1:111122223333:function:external-hms-service-new
```

Jika Anda menentukan ARN penuh dengan cara ini, Athena dapat memanggil`external-hms-service-new`Fungsi Lambda`us-east-1`untuk mengambil data metastore Hive dari`eu-central-1`.

> **Note**
>
> Katalogehmsharus terdaftar di wilayah yang sama bahwa Anda menjalankan permintaan Athena.

Pemanggilan Lambda lintas akun

Kadang-kadang Anda mungkin memerlukan akses ke metastore Hive dari akun yang berbeda. Misalnya, untuk menjalankan metastore Hive, Anda mungkin meluncurkan cluster EMR dari akun yang berbeda dari salah satu yang Anda gunakan untuk Athena kueri. grup yang berbeda atau tim mungkin menjalankan Hive metastore dengan akun yang berbeda dalam VPC mereka. Atau Anda mungkin ingin mengakses metadata dari metastores Hive berbeda dari grup atau tim yang berbeda.

Athena menggunakanAWS Lambda dukungan untuk akses lintas akununtuk mengaktifkan akses lintas rekening untuk Hive Metastores.

> **Note**
>
> Perhatikan bahwa akses lintas rekening untuk Athena biasanya menyiratkan akses rekening lintas untuk kedua metadata dan data di Amazon S3.

Bayangkan skenario berikut:

- Akun111122223333mengatur fungsi Lambdaexternal-hms-service-newpada kita-timur-1 di Athena untuk mengakses Hive Metastore berjalan pada cluster EMR.
- Akun111122223333ingin memungkinkan akun 444455556666 untuk mengakses data Hive Metastore.

Untuk memberikan 444455556666 akses akun ke fungsi Lambdaexternal-hms-service-new, akun 111122223333 menggunakan perintah berikut AWS CLI add-permission. Perintah telah diformat untuk dibaca.

```
$ 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\"}"
}
```

Untuk memeriksa izin Lambda, gunakanget-policyseperti pada contoh berikut. Perintah telah diformat untuk dibaca.

```
$ 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\"}]}"
}
```

Setelah menambahkan izin, Anda dapat menggunakan ARN lengkap dari fungsi Lambda`us-east-1`seperti berikut saat anda menentukan katalog`ehms`:

```
arn:aws:lambda:us-east-1:111122223333:function:external-hms-service-new
```

Untuk informasi tentang doa lintas wilayah, lihatDoa Lambda Lintas Wilayahsebelumnya dalam topik ini.

Memberikan akses lintas akun ke data

Sebelum Anda dapat menjalankan kueri Athena, Anda harus memberikan akses lintas rekening ke data di Amazon S3. Anda dapat melakukannya dengan salah satu cara berikut:

- Memperbarui kebijakan daftar kontrol akses bucket Amazon S3 denganID pengguna kanonis.
- Menambahkan akses akun lintas ke kebijakan bucket Amazon S3.

Sebagai contoh, tambahkan kebijakan berikut untuk Amazon S3 bucket kebijakan di akun`111122223333`untuk mengizinkan akun`444455556666`Untuk membaca data dari lokasi Amazon S3 yang ditentukan.

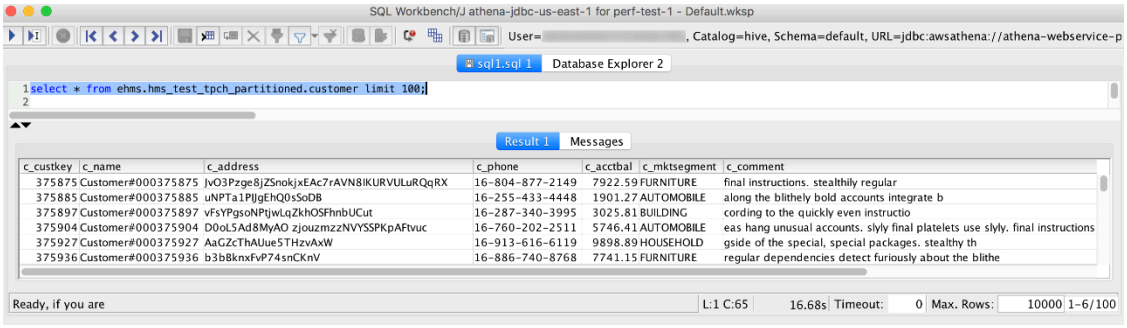
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1234567890123",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::444455556666:user/perf1-test"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::athena-test/lambda/dataset/*"
        }
    ]
```

```
}
```

> **Note**
>
> Anda mungkin perlu untuk memberikan akses rekening lintas ke Amazon S3 tidak hanya untuk data Anda, tetapi juga untuk Amazon S3 lokasi tumpahan Anda. Fungsi Lambda Anda tumpahan data tambahan ke lokasi tumpahan saat ukuran objek respon melebihi ambang batas yang diberikan. Lihat awal topik ini untuk kebijakan sampel.

Dalam contoh saat ini, setelah akses lintas rekening diberikan kepada`444455556666`, `444455556666`dapat menggunakan katalog`ehms`sendiri`account`untuk kueri tabel yang didefinisikan dalam akun`111122223333`.

Pada contoh berikut, profil SQL Workbench`perf-test-1`Untuk akun`444455556666`. kueri menggunakan katalog`ehms`untuk mengakses metastore Hive dan data Amazon S3 pada akun`111122223333`.



Contoh kebijakan izin IAM untuk mengizinkan Kueri Federasi Athena

Contoh kebijakan izin dalam topik ini menunjukkan diperlukan tindakan diperbolehkan dan sumber daya yang mereka diizinkan. Periksa kebijakan ini dengan saksama dan mengubahnya sesuai dengan kebutuhan Anda sebelum melampirkannya ke identitas IAM.

Untuk informasi tentang melampirkan kebijakan ke identitas IAM, lihat Menambahkan dan menghapus izin identitas IAM di Panduan Pengguna IAM.

- Example policy to allow an IAM principal to run and return results using Athena Federated Query
- Example Policy to Allow an IAM Principal to Create a Data Source Connector

Example — Izinkan prinsipal IAM untuk menjalankan dan mengembalikan hasil menggunakan Athena Federated Query

Kebijakan izin berbasis identitas berikut memungkinkan tindakan yang diperlukan pengguna atau pimpinan IAM lainnya untuk menggunakan Kueri Gabungan Athena. Utama yang diizinkan untuk melakukan tindakan ini dapat menjalankan kueri yang menentukan katalog Athena yang terkait dengan sumber data gabungan.

```
{
    "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/*"
            ]
        }
    ]
}
```

Penjelasan perizinan

Tindakan yang diizinkan	Penjelasan
`"athena:GetQueryExecution", "athena:GetQueryResults", "athena:GetWorkGroup", "athena:StartQueryExecution", "athena:StopQueryExecution"`	Athena izin yang diperlukan untuk menjalankan kueri gabungan.
`"athena:GetDataCatalog", "athena:GetQueryExecution," "athena:GetQueryResults", "athena:GetWorkGroup", "athena:StartQueryExecution", "athena:StopQueryExecution"`	Izin Athena yang diperlukan untuk menjalank an kueri tampilan federasi. `GetDataCa talog` Tindakan ini diperlukan untuk tampilan.
`"lambda:InvokeFunction"`	Memungkinkan kueri untuk memanggil AWS Lambda fungsi untuk AWS Lambda fungsi yang ditentukan dalam blok. `Resource` Misalnya`aarn:aws:lambda:*: MyAWSAcct`

Tindakan yang diizinkan	Penjelasan
	Id :function: *MyAthenaLambdaFunction* , di mana *MyAthenaLambdaFunction* menentukan nama fungsi Lambda yang akan dipanggil. Seperti yang ditunjukkan pada contoh, beberapa fungsi dapat ditentukan.
`"s3:AbortMultipartUpload", "s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket", "s3:ListMultipartUploadParts", "s3:PutObject"`	`s3:GetBucketLocation` Izin `s3:ListBucket` dan diperlukan untuk mengakses bucket keluaran kueri untuk prinsipal IAM yang berjalan. `StartQueryExecution` `s3:PutObject` ,`s3:ListMultipartUploadParts` , dan `s3:AbortMultipartUpload` izinkan penulisan hasil kueri ke semua sub-folder bucket hasil kueri seperti yang ditentukan oleh pengidentifikasi `arn:aws:s3::: MyQueryResultsBucket /*` sumber daya, di mana *MyQueryResultsBucket* keranjang hasil kueri Athena. Untuk informasi selengkapnya, lihat Bekerja dengan hasil kueri, kueri terbaru, dan file keluaran. `s3:GetObject` memungkinkan pembacaan hasil kueri dan riwayat kueri untuk sumber daya yang ditentukan sebagai`arn:aws:s3::: MyQueryResultsBucket` , di mana *MyQueryResultsBucket* keranjang hasil kueri Athena. `s3:GetObject` juga memungkinkan membaca dari sumber daya yang ditentukan sebagai`"arn:aws:s3::: MyLambdaSpillBucket /MyLambdaSpillPrefix *"`, di mana *MyLambdaSpillPrefix* ditentukan dalam konfigurasi fungsi Lambda atau fungsi yang dipanggil.

Example — Izinkan prinsipal IAM untuk membuat konektor sumber data

```
{
    "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/*"
        }
    ]
}
```

Penjelasan perizinan

Tindakan yang diizinkan	Penjelasan
`"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",`	Memungkinkan penciptaan dan pengelolaan fungsi Lambda terdaftar sebagai sumber daya. Dalam contoh, awalan nama digunakan dalam pengenal sumber daya`arn:aws:lambda:*:` *MyAWSAcctId* `:function:` *MyAthenaLambdaFunctionsPrefix* `*`, di mana *MyAthenaLambdaFunctionsPrefix* awalan bersama digunakan dalam nama grup fungsi Lambda sehingga tidak perlu ditentukan secara individual sebagai sumber daya. Anda dapat menentukan satu atau lebih sumber daya fungsi Lambda.
`"s3:GetObject"`	Memungkinkan membaca bucket yang AWS Serverless Application Repository membutuhkan seperti yang ditentukan oleh pengidentifikasi `arn:aws:s3:::awsserverlessrepo-changesets-` *1iiv3xa62ln3m* `/*` sumber daya. Bucket ini mungkin spesifik untuk akun Anda.

Tindakan yang diizinkan	Penjelasan
`"cloudformation:*"`	Memungkinkan pembuatan dan pengelolaan AWS CloudFormation tumpukan yang ditentukan oleh sumber daya *`StackPrefixmyCF`* . Tumpukan dan stackset ini adalah cara AWS Serverless Application Repository menyebarkan konektor dan UDF.
`"serverlessrepo:*"`	Memungkinkan pencarian, melihat, menerbitkan, dan memperbarui aplikasi di AWS Serverless Application Repository, yang ditentukan oleh pengidentifikasi `arn:aws:serverlessrepo:*:*:applications/*` sumber daya.

Contoh kebijakan izin IAM untuk mengizinkan Amazon Athena User Defined Functions (UDF)

Contoh kebijakan izin dalam topik ini menunjukkan diperlukan tindakan diperbolehkan dan sumber daya yang mereka diizinkan. Periksa kebijakan ini dengan hati-hati dan memodifikasi mereka sesuai dengan kebutuhan Anda sebelum Anda melampirkan kebijakan izin yang sama untuk identitas IAM.

- Example Policy to Allow an IAM Principal to Run and Return Queries that Contain an Athena UDF Statement
- Example Policy to Allow an IAM Principal to Create an Athena UDF

Example — Izinkan prinsipal IAM untuk menjalankan dan mengembalikan kueri yang berisi pernyataan Athena UDF

Kebijakan izin berbasis identitas berikut memungkinkan tindakan yang pengguna atau utama IAM lainnya memerlukan untuk menjalankan permintaan yang menggunakan pernyataan Athena UDF.

```
{
    "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": "*"
        }
    ]
}
```

Penjelasan izin

Tindakan yang diizinkan	Penjelasan
`"athena:StartQueryExecution", "athena:GetQueryResults", "athena:GetWorkGroup", "athena:StopQueryExecution", "athena:GetQueryExecution",`	Athena izin yang diperlukan untuk menjalankan kueri di`MyAthenaWorkGroup` kelompok kerja.

Tindakan yang diizinkan	Penjelasan

`"s3:PutObject",`
`"s3:GetObject",`
`"s3:AbortMultipartUpload"`
	`s3:PutObject` dan `s3:AbortMultipartUpload` izinkan penulisan hasil kueri ke semua sub-folder bucket hasil kueri seperti yang ditentukan oleh pengidentifikasi `arn:aws:s3::: MyQueryResultsBucket /*` sumber daya, di mana *`MyQueryResultsBucket`* keranjang hasil kueri Athena. Untuk informasi selengkapnya, lihat Bekerja dengan hasil kueri, kueri terbaru, dan file keluaran.
	
	`s3:GetObject` memungkinkan pembacaan hasil kueri dan riwayat kueri untuk sumber daya yang ditentukan sebagai`arn:aws:s3::: MyQueryResultsBucket` , di mana *`MyQueryResultsBucket`* keranjang hasil kueri Athena. Untuk informasi selengkapnya, lihat Bekerja dengan hasil kueri, kueri terbaru, dan file keluaran.
	
	`s3:GetObject` juga memungkinkan membaca dari sumber daya yang ditentukan sebagai`"arn:aws:s3::: MyLambdaSpillBucket /MyLambdaSpillPrefix *"`, di mana *`MyLambdaSpillPrefix`* ditentukan dalam konfigurasi fungsi Lambda atau fungsi yang dipanggil.

Tindakan yang diizinkan	Penjelasan
`"lambda:InvokeFunction"`	Mengizinkan kueri untuk memanggil AWS Lambda fungsi yang ditentukan dalam blok. `Resource` Misalnya`aarn:aws:lambda:*: MyAWSAcctId :function : MyAthenaLambdaFunction` , di mana `MyAthenaLambdaFunction` menentukan nama fungsi Lambda yang akan dipanggil. Beberapa fungsi dapat ditentukan seperti yang ditunjukkan pada contoh.

Example — Izinkan kepala IAM untuk membuat UDF Athena

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "lambda:CreateFunction",
                "lambda:ListVersionsByFunction",
                "iam:CreateRole",
                "lambda:GetFunctionConfiguration",
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy",
                "lambda:PutFunctionConcurrency",
                "iam:PassRole",
                "iam:DetachRolePolicy",
                "lambda:ListTags",
                "iam:ListAttachedRolePolicies",
                "iam:DeleteRolePolicy",
                "lambda:DeleteFunction",
                "lambda:GetAlias",
                "iam:ListRolePolicies",
                "iam:GetRole",
                "iam:GetPolicy",
                "lambda:InvokeFunction",
                "lambda:GetFunction",
```

```
                "lambda:ListAliases",
                "lambda:UpdateFunctionConfiguration",
                "iam:DeleteRole",
                "lambda:UpdateFunctionCode",
                "s3:GetObject",
                "lambda:AddPermission",
                "iam:UpdateRole",
                "lambda:DeleteFunctionConcurrency",
                "lambda:RemovePermission",
                "iam:GetRolePolicy",
                "lambda:GetPolicy"
            ],
            "Resource": [

"arn:aws:lambda:*:111122223333:function:MyAthenaLambdaFunctionsPrefix*",
                "arn:aws:s3:::awsserverlessrepo-changesets-1iiv3xa62ln3m/*",
                "arn:aws:iam::*:role/RoleName",
                "arn:aws:iam::111122223333:policy/*"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateUploadBucket",
                "cloudformation:DescribeStackDriftDetectionStatus",
                "cloudformation:ListExports",
                "cloudformation:ListStacks",
                "cloudformation:ListImports",
                "lambda:ListFunctions",
                "iam:ListRoles",
                "lambda:GetAccountSettings",
                "ec2:DescribeSecurityGroups",
                "cloudformation:EstimateTemplateCost",
                "ec2:DescribeVpcs",
                "lambda:ListEventSourceMappings",
                "cloudformation:DescribeAccountLimits",
                "ec2:DescribeSubnets",
                "cloudformation:CreateStackSet",
                "cloudformation:ValidateTemplate"
            ],
            "Resource": "*"
        },
        {
```

```
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": "cloudformation:*",
            "Resource": [
                "arn:aws:cloudformation:*:111122223333:stack/aws-serverless-
repository-MyCFStackPrefix*/*",
                "arn:aws:cloudformation:*:111122223333:stack/
serverlessrepo-MyCFStackPrefix*/*",
                "arn:aws:cloudformation:*:*:transform/Serverless-*",
                "arn:aws:cloudformation:*:111122223333:stackset/aws-serverless-
repository-MyCFStackPrefix*:*",
                "arn:aws:cloudformation:*:111122223333:stackset/
serverlessrepo-MyCFStackPrefix*:*"
            ]
        },
        {
            "Sid": "VisualEditor3",
            "Effect": "Allow",
            "Action": "serverlessrepo:*",
            "Resource": "arn:aws:serverlessrepo:*:*:applications/*"
        }
    ]
}
```

Penjelasan izin

Tindakan yang diizinkan	Penjelasan
`"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",`	Memungkinkan penciptaan dan pengelola an fungsi Lambda terdaftar sebagai sumber daya. Dalam contoh, awalan nama digunakan dalam pengenal sumber daya`aarn:aws:l ambda:*: MyAWSAcctId :function : MyAthenaLambdaFunctionsPref ix *`, di mana *MyAthenaLambdaFunc tionsPrefix* awalan bersama digunakan dalam nama grup fungsi Lambda sehingga tidak perlu ditentukan secara individual sebagai sumber daya. Anda dapat menentukan satu atau lebih sumber daya fungsi Lambda.

Tindakan yang diizinkan	Penjelasan
`"lambda:RemovePermission", "lambda:GetPolicy" "lambda:GetAccountSettings", "lambda:ListFunctions", "lambda:ListEventSourceMappings",`	
`"s3:GetObject"`	Memungkinkan membaca bucket yang AWS Serverless Application Repository membutuhkan seperti yang ditentukan oleh pengidentifikasi `arn:aws:s3:::awsserverlessrepo-changesets-` *`1iiv3xa62ln3m`* `/*` sumber daya.
`"cloudformation:*"`	Memungkinkan pembuatan dan pengelolaan AWS CloudFormation tumpukan yang ditentukan oleh sumber daya *`StackPrefixmyCF`* . Tumpukan dan stackset ini adalah cara AWS Serverless Application Repository menyebarkan konektor dan UDF.
`"serverlessrepo:*"`	Memungkinkan pencarian, melihat, menerbitkan, dan memperbarui aplikasi di AWS Serverless Application Repository, yang ditentukan oleh pengidentifikasi `arn:aws:serverlessrepo:*:*:applications/*` sumber daya.

Mengizinkan akses untuk ML dengan Athena

IAM utama yang menjalankan kueri ML Athena harus diizinkan menjalankan tindakan `sagemaker:invokeEndpoint` untuk titik akhir Sagemaker yang mereka gunakan. Sertakan pernyataan kebijakan yang mirip dengan berikut dalam kebijakan izin berbasis identitas yang melekat pada identitas pengguna. Selain itu, lampirkanAWS kebijakan terkelola: AmazonAthenaFullAccess, yang memberikan akses penuh ke tindakan Athena, atau kebijakan inline yang dimodifikasi yang memungkinkan subset tindakan.

Gantiarn:aws:sagemaker:*region*:*AWSAcctID*:*ModelEndpoint*dalam contoh dengan ARN atau ARN model endpoint yang akan digunakan dalam kueri. Untuk informasi selengkapnya, lihat Kunci tindakan, sumber daya, dan kondisi SageMaker di Referensi Otorisasi Layanan.

```
{
            "Effect": "Allow",
            "Action": [
                "sagemaker:invokeEndpoint"
            ],
            "Resource": "arn:aws:sagemaker:us-west-2:123456789012:workteam/public-
crowd/default"
}
```

Setiap kali Anda menggunakan kebijakan IAM, pastikan bahwa Anda mengikuti praktik terbaik IAM. Untuk informasi selengkapnya tentang administrator, lihat Praktik terbaik keamanan di IAM dalam Panduan Pengguna IAM.

Mengaktifkan akses federasi ke Athena API

Bagian ini membahas akses gabungan yang memungkinkan pengguna atau aplikasi klien di organisasi Anda untuk menghubungi operasi API Amazon Athena. Dalam hal ini, pengguna organisasi Anda tidak memiliki akses langsung ke Athena. Sebagai gantinya, Anda mengelola kredensi pengguna di luar AWS Microsoft Active Directory. Dukungan Active DirectorySAML 2.0(Keamanan Penegasan Markup Language 2.0).

Untuk mengotentikasi pengguna dalam skenario ini, gunakan driver JDBC atau ODBC dengan SAML.2.0 dukungan untuk mengakses Active Directory Federation Services (ADFS) 3.0 dan mengaktifkan aplikasi klien untuk memanggil Athena API operasi.

Untuk informasi selengkapnya tentang dukungan SAMP 2.0 AWS, lihat Tentang federasi SAMP 2.0 di Panduan Pengguna IAM.

> **Note**
>
> Akses gabungan ke API Athena didukung untuk jenis tertentu penyedia identitas (IdP), Active Directory Federation Service (ADFS 3.0), yang merupakan bagian dari Windows Server. Akses federasi tidak kompatibel dengan fitur propagasi identitas tepercaya IAM Identity Center. Akses dibuat melalui versi driver JDBC atau ODBC yang mendukung SAML 2.0.

Untuk informasi selengkapnya, lihat Menghubungkan ke Amazon Athena dengan JDBC dan Menghubungkan ke Amazon Athena dengan ODBC.

Topik

- Sebelum Anda mulai
- Diagram arsitektur
- Prosedur: Akses federasi berbasis SAML ke Athena API

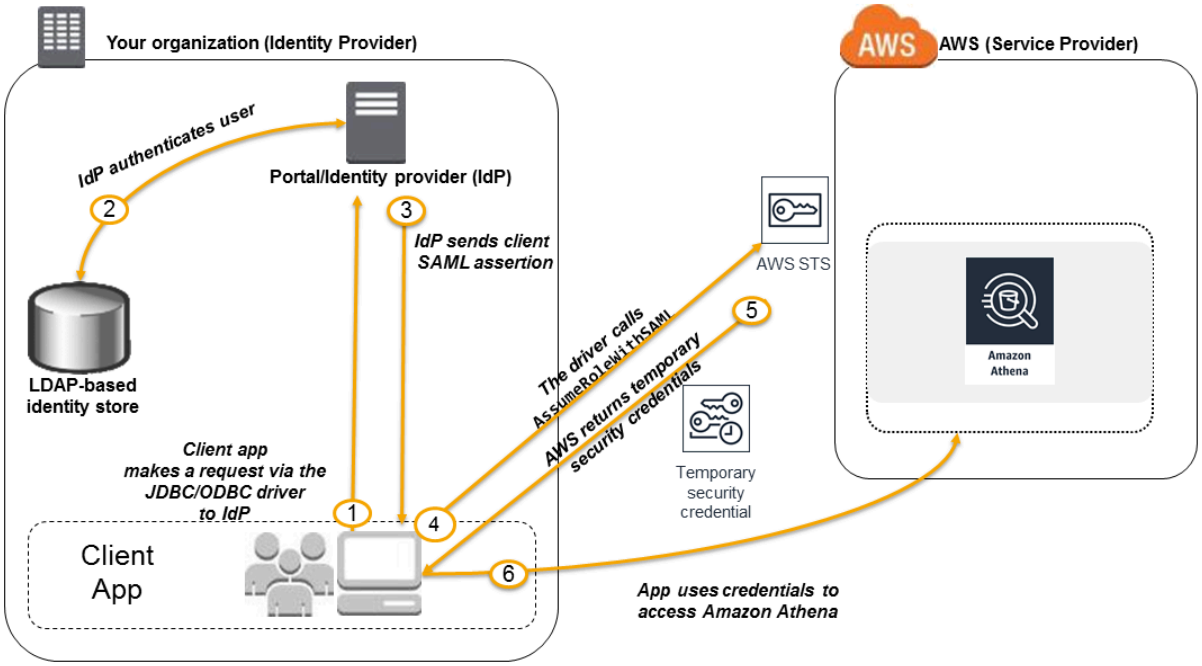
Sebelum Anda mulai

Sebelum menggunakan fungsi , pastikan untuk melengkapi prasyarat berikut:

- Di dalam organisasi Anda, instal dan konfigurasikan ADFS 3.0 sebagai IdP Anda.
- Menginstal dan mengonfigurasi versi terbaru yang tersedia dari driver JDBC atau ODBC pada klien yang digunakan untuk mengakses Athena. Driver harus menyertakan dukungan untuk akses gabungan kompatibel dengan SAML 2.0. Untuk informasi selengkapnya, lihat Menghubungkan ke Amazon Athena dengan JDBC dan Menghubungkan ke Amazon Athena dengan ODBC.

Diagram arsitektur

Diagram berikut menggambarkan proses.



1. Pengguna dalam organisasi Anda menggunakan aplikasi klien untuk meminta autentikasi dari organisasi Anda. IdP adalah ADFS 3.0.
2. mengautentikasi pengguna terhadap toko identitas organisasi Anda.
3. menyusun pernyataan SAML dengan informasi tentang pengguna dan mengirimkan pertanyaan ke aplikasi klien.
4. Driver JDBC atau ODBC memanggil operasi AWS Security Token Service AssumeRoleWithSAMP API, meneruskannya parameter berikut:
   - ARN penyedia SAML
   - ARN peran untuk mengasumsikan
   - Pernyataan SAML dari IdP

   Untuk informasi selengkapnya, lihat AssumeRoleWithSAMP, di Referensi AWS Security Token Service API.
5. Tanggapan API ke aplikasi klien meliputi kredensial keamanan sementara.
6. Aplikasi klien menggunakan kredensial keamanan sementara untuk menghubungi operasi API Athena, memungkinkan pengguna mengakses operasi API Athena.

Prosedur: Akses federasi berbasis SAML ke Athena API

Prosedur ini menetapkan kepercayaan antara IDP organisasi Anda dan akun AWS Anda untuk mengaktifkan akses federasi berbasis SAML ke operasi Amazon Athena API.

Untuk mengaktifkan akses gabungan ke API Athena:

1. Di organisasi Anda, daftar AWS sebagai penyedia layanan (SP) di IDP Anda. Proses ini dikenal sebagaiMengonfigurasi kepercayaan pihak. Untuk informasi selengkapnya, lihat Mengonfigurasi IDP SAMP 2.0 Anda dengan mengandalkan kepercayaan pihak pada Panduan Pengguna IAM. Sebagai bagian dari tugas ini, lakukan langkah-langkah berikut:

   a. Dapatkan dokumen metadata SAML sampel dari URL ini: https://signin.aws.amazon.com/static/saml-metadata.xml.

   b. Di iDP (ADFS) organisasi Anda, buat file XMLmetadata setara yang menjelaskan IDP Anda sebagai penyedia identitas. AWS File metadata Anda harus menyertakan nama penerbit, tanggal pembuatan, tanggal kedaluwarsa, dan kunci yang AWS digunakan untuk memvalidasi tanggapan otentikasi (pernyataan) dari organisasi Anda.

2. Di konsol IAM, buat entitas penyedia identitas SAML. Untuk informasi selengkapnya, lihat Membuat penyedia identitas SAMP di Panduan Pengguna IAM. Sebagai bagian dari langkah ini, lakukan hal berikut:

   a. Buka konsol IAM di https://console.aws.amazon.com/iam/.

   b. Upload dokumen metadata SAML yang dihasilkan oleh IdP (ADFS) di langkah 1 dalam prosedur ini.

3. Di konsol IAM, buat satu atau lebih IAM role untuk IdP Anda. Untuk informasi selengkapnya, lihat Membuat peran untuk Penyedia Identitas pihak ketiga (federasi) di Panduan Pengguna IAM. Sebagai bagian dari langkah ini, lakukan hal berikut:

   - Dalam kebijakan izin peran, cantumkan tindakan yang diizinkan untuk dilakukan oleh pengguna dari organisasi Anda di AWS.
   - Dalam kebijakan kepercayaan peran, tetapkan entitas penyedia SAML yang Anda buat di Langkah 2 prosedur ini sebagai utama.

   Ini membangun hubungan kepercayaan antara organisasi Anda dan AWS.

4. Dalam organisasi , Anda mendefinisikan pernyataan yang memetakan pengguna atau grup dalam organisasi Anda ke IAM role. Pemetaan pengguna dan grup untuk IAM role juga dikenal

sebagaiAturan klaim. Perhatikan bahwa pengguna dan grup yang berbeda pada organisasi Anda mungkin memetakan IAM role yang berbeda.

Untuk informasi tentang mengonfigurasi pemetaan di ADFS, lihat posting blog: Mengaktifkan federasi untuk AWS menggunakan Windows Active Directory, ADFS, dan SAMP 2.0.

5. Menginstal dan mengonfigurasi driver JDBC atau ODBC dengan dukungan SAML 2.0. Untuk informasi selengkapnya, lihat Menghubungkan ke Amazon Athena dengan JDBC dan Menghubungkan ke Amazon Athena dengan ODBC.
6. Tentukan string koneksi dari aplikasi Anda ke driver JDBC atau ODBC. Untuk informasi tentang rangkaian koneksi yang harus digunakan aplikasi Anda, lihat topik“Menggunakan penyedia kredensials Active Directory Federation Services (ADFS)”diJDBC Driver Instalasi dan Panduan Konfigurasi, atau topik serupa diODBC Driver instalasi dan konfigurasi panduantersedia sebagai unduhan PDF dariMenghubungkan ke Amazon Athena dengan JDBCdanMenghubungkan ke Amazon Athena dengan ODBCtopik.

   Berikut ini adalah ringkasan level tinggi mengonfigurasi string koneksi ke driver:

   1. Di`AwsCredentialsProviderClass configuration`, mengatur`com.simba.athena.iamsupport.plugin.AdfsCredentialsProvider`untuk menunjukkan bahwa Anda ingin menggunakan otentikasi berbasis SAML 2.0 melalui ADFS IdP.
   2. Untuk`idp_host`, memberikan nama host ADFS IdP server.
   3. Untuk`idp_port`, menyediakan nomor port yang ADFS IdP mendengarkan untuk permintaan pernyataan SAML.
   4. Untuk`UID`dan`PWD`, berikan kredensial pengguna domain AD. Saat menggunakan driver di Windows, jika`UID`dan`PWD`tidak disediakan, driver mencoba untuk mendapatkan kredensial pengguna yang masuk ke mesin Windows.
   5. Opsional, set`ssl_insecure`ke`true`. Dalam kasus ini, driver tidak memeriksa keaslian sertifikat SSL untuk ADFS IdP server. Pengaturan ke `true` diperlukan jika sertifikat SSL ADFS iDP belum dikonfigurasi untuk dipercaya oleh driver.
   6. Untuk mengaktifkan pemetaan pengguna domain direktori aktif atau grup untuk satu atau lebih IAM role (seperti yang disebutkan dalam langkah 4 prosedur ini), dalam`preferred_role`untuk koneksi JDBC atau ODBC, menentukan IAM role (ARN) untuk mengasumsikan untuk koneksi driver. Menentukan`preferred_role`bersifat opsional, dan berguna jika peran bukan peran pertama yang tercantum dalam aturan klaim.

Sebagai hasil dari prosedur ini, tindakan berikut terjadi:

1. Driver JDBC atau ODBC memanggil AWS STSAssumeRoleWithSAMP API, dan meneruskannya pernyataan, seperti yang ditunjukkan pada langkah 4 diagram arsitektur.
2. AWS memastikan bahwa permintaan untuk mengambil peran berasal dari idP yang direferensikan di entitas penyedia SAMP.
3. Jika permintaan berhasil, operasi AWS STS AssumeRoleWithSAMP API mengembalikan satu set kredensil keamanan sementara, yang digunakan aplikasi klien Anda untuk membuat permintaan yang ditandatangani ke Athena.

   Aplikasi Anda sekarang memiliki informasi tentang pengguna saat ini dan dapat mengakses Athena secara pemrograman.

Penebangan dan pemantauan di Athena

Untuk mendeteksi insiden, menerima peringatan saat insiden terjadi, dan menanggapinya, gunakan opsi ini dengan Amazon Athena:

- Memantau Athena dengan AWS CloudTrail — AWS CloudTrailmenyediakan catatan tindakan yang diambil oleh pengguna, peran, atau di Layanan AWS Athena. Ini menangkap panggilan dari konsol Athena dan panggilan kode ke operasi API Athena sebagai peristiwa. Anda dapat menentukan permintaan yang dibuat ke Athena, alamat IP tempat permintaan dibuat, siapa yang membuat permintaan, kapan permintaan dibuat, dan detail tambahan. Untuk informasi selengkapnya, lihat Mencatat panggilan API Amazon Athena dengan AWS CloudTrail.

  Anda juga dapat menggunakan Athena untuk menanyakan file CloudTrail log tidak hanya untuk Athena, tetapi untuk yang lain. Layanan AWS Untuk informasi selengkapnya, lihat Meminta log AWS CloudTrail.
- Pantau penggunaan Athena dengan dan Amazon — CloudTrail QuickSight Amazon QuickSight adalah layanan intelijen bisnis bertenaga cloud yang dikelola sepenuhnya yang memungkinkan Anda membuat dasbor interaktif yang dapat diakses organisasi Anda dari perangkat apa pun. Untuk contoh solusi yang menggunakan CloudTrail dan Amazon QuickSight untuk memantau penggunaan Athena, lihat posting blog AWS Big Data Bagaimana Realtor.com memantau penggunaan Amazon Athena dengan dan Amazon. AWS CloudTrail QuickSight

- Gunakan EventBridge dengan Athena - Amazon EventBridge memberikan aliran peristiwa sistem yang mendekati waktu nyata yang menggambarkan perubahan sumber daya. AWS EventBridge menyadari perubahan operasional saat terjadi, meresponsnya, dan mengambil tindakan korektif seperlunya, dengan mengirim pesan untuk merespons lingkungan, mengaktifkan fungsi, membuat perubahan, dan menangkap informasi negara. Peristiwa dipancarkan atas dasar upaya terbaik. Untuk informasi selengkapnya, lihat Memulai Amazon EventBridge di Panduan EventBridge Pengguna Amazon.
- Gunakan grup kerja untuk memisahkan pengguna, tim, aplikasi, atau beban kerja, dan untuk menetapkan batas kueri dan mengontrol biaya kueri — Anda dapat melihat metrik terkait kueri di Amazon CloudWatch, mengontrol biaya kueri dengan mengonfigurasi batas jumlah data yang dipindai, membuat ambang batas, dan memicu tindakan, seperti alarm Amazon SNS, saat ambang batas ini dilanggar. Untuk prosedur tingkat tinggi, lihatMenyiapkan kelompok kerja. Gunakan izin IAM level sumber daya untuk mengontrol akses ke grup kerja tertentu. Lihat informasi yang lebih lengkap di Menggunakan workgroup untuk menjalankan kueri dan Mengontrol biaya dan memantau kueri dengan CloudWatch metrik dan peristiwa.

Topik

- Mencatat panggilan API Amazon Athena dengan AWS CloudTrail

Mencatat panggilan API Amazon Athena dengan AWS CloudTrail

Athena terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau di Layanan AWS Athena.

CloudTrail menangkap semua panggilan API untuk Athena sebagai acara. Panggilan yang direkam meliputi panggilan dari konsol Athena dan panggilan kode ke operasi API Athena. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara berkelanjutan ke bucket Amazon S3, termasuk acara untuk Athena. Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara.

Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke Athena, alamat IP dari mana permintaan itu dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, lihat Panduan AWS CloudTrail Pengguna.

Anda dapat menggunakan Athena untuk menanyakan file CloudTrail log dari Athena itu sendiri dan dari yang lain. Layanan AWS Untuk informasi lebih lanjut, lihatMeminta log AWS CloudTrail,Sarang

JSON SerDe, dan posting Blog AWS Big Data Gunakan pernyataan CTAS dengan Amazon Athena untuk mengurangi biaya dan meningkatkan kinerja, yang CloudTrail digunakan untuk memberikan wawasan tentang penggunaan Athena.

Informasi Athena di CloudTrail

CloudTrail diaktifkan di akun Amazon Web Services Anda saat Anda membuat akun. Ketika aktivitas terjadi di Athena, aktivitas tersebut dicatat dalam suatu CloudTrail peristiwa bersama dengan acara AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh kejadian terbaru di akun Amazon Web Services Anda. Untuk informasi selengkapnya, lihat Melihat peristiwa dengan riwayat CloudTrail acara.

Untuk catatan peristiwa yang sedang berlangsung di akun Amazon Web Services Anda, termasuk peristiwa untuk Athena, buatlah jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, saat Anda membuat jejak di konsol, jejak tersebut berlaku untuk semua Wilayah AWS. Jejak mencatat peristiwa dari semua Wilayah di AWS partisi dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi lainnya Layanan AWS untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat berikut:

- Gambaran umum untuk membuat jejak
- CloudTrail layanan dan integrasi yang didukung
- Mengonfigurasi notifikasi Amazon SNS untuk CloudTrail
- Menerima file CloudTrail log dari beberapa wilayah dan Menerima file CloudTrail log dari beberapa akun

Semua tindakan Athena dicatat oleh CloudTrail dan didokumentasikan dalam Referensi API Amazon Athena. Misalnya, panggilan ke StartQueryExecutiondan GetQueryResultstindakan menghasilkan entri dalam file CloudTrail log.

Setiap entri peristiwa atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut ini:

- Apakah permintaan itu dibuat dengan kredenal pengguna root atau AWS Identity and Access Management (IAM).
- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna terfederasi.
- Apakah permintaan tersebut dibuat oleh Layanan AWS lain.

Untuk informasi selengkapnya, lihat elemen CloudTrail UserIdentity.

Memahami entri file log Athena

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber manapun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, jadi file tersebut tidak muncul dalam urutan tertentu.

> **Note**
>
> Untuk mencegah pengungkapan informasi sensitif yang tidak diinginkan, `queryString` entri dalam `CreateNamedQuery` log `StartQueryExecution` dan log memiliki nilai. `***OMITTED***` Ini dengan desain. Untuk mengakses string kueri yang sebenarnya, Anda dapat menggunakan Athena GetQueryExecutionAPI dan meneruskan nilai `responseElements.queryExecutionId` dari log. CloudTrail

Contoh berikut menunjukkan entri CloudTrail log untuk:

- StartQueryExecution(Berhasil)
- StartQueryExecution (Gagal)
- CreateNamedQuery

StartQueryExecution (berhasil)

```
{
 "eventVersion":"1.05",
 "userIdentity":{
    "type":"IAMUser",
    "principalId":"EXAMPLE_PRINCIPAL_ID",
    "arn":"arn:aws:iam::123456789012:user/johndoe",
    "accountId":"123456789012",
    "accessKeyId":"EXAMPLE_KEY_ID",
    "userName":"johndoe"
 },
 "eventTime":"2017-05-04T00:23:55Z",
```

```
 "eventSource":"athena.amazonaws.com",
 "eventName":"StartQueryExecution",
 "awsRegion":"us-east-1",
 "sourceIPAddress":"77.88.999.69",
 "userAgent":"aws-internal/3",
 "requestParameters":{
    "clientRequestToken":"16bc6e70-f972-4260-b18a-db1b623cb35c",
    "resultConfiguration":{
       "outputLocation":"s3://DOC-EXAMPLE-BUCKET/test/"
    },
    "queryString":"***OMITTED***"
 },
 "responseElements":{
    "queryExecutionId":"b621c254-74e0-48e3-9630-78ed857782f9"
 },
 "requestID":"f5039b01-305f-11e7-b146-c3fc56a7dc7a",
 "eventID":"c97cf8c8-6112-467a-8777-53bb38f83fd5",
 "eventType":"AwsApiCall",
 "recipientAccountId":"123456789012"
}
```

StartQueryExecution (gagal)

```
{
 "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"
},
```

Validasi kepatuhan untuk Amazon Athena

Auditor pihak ketiga menilai keamanan dan kepatuhan Amazon Athena sebagai bagian dari beberapa program kepatuhan AWS. Ini mencakup SOC, PCI, FedRAMP, dan lainnya.

Untuk daftarLayanan AWSdalam lingkup program kepatuhan spesifik, lihatLayanan AWSdalam lingkup oleh program kepatuhan. Untuk informasi umum, lihat Program kepatuhan AWS.

Anda bisa mengunduh laporan audit pihak ke tiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact.

Tanggung jawab kepatuhan Anda saat menggunakan Athena ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, serta hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu dengan kepatuhan:

- Panduan mulai cepat keamanan dan kepatuhan – Panduan deployment ini membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk men-deploy lingkungan dasar yang berfokus pada keamanan dan kepatuhan di AWS.
- Merancang untuk keamanan dan kepatuhan HIPAA di Amazon Web Services- Whitepaper ini menjelaskan bagaimana perusahaan dapat menggunakanAWSuntuk membuat aplikasi yang sesuai dengan HIPAA.
- Sumber daya kepatuhan AWS – Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- AWS Config— IniLayanan AWSmenilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub – Layanan AWS ini menyediakan pandangan yang komprehensif tentang status keamanan Anda dalam AWS yang membantu Anda memeriksa kepatuhan Anda terhadap standar industri dan praktik terbaik untuk keamanan.

Ketahanan di Athena

Infrastruktur AWS global dibangun di sekitar Wilayah AWS dan Availability Zones. Wilayah AWS menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan failover di antara Zona Ketersediaan tanpa gangguan. Zona Ketersediaan

memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur biasa yang terdiri dari satu atau beberapa pusat data.

Untuk informasi selengkapnya tentang Wilayah AWS dan Availability Zone, lihat infrastruktur AWS global.

Selain infrastruktur AWS global, Athena menawarkan beberapa fitur untuk membantu mendukung ketahanan data dan kebutuhan cadangan Anda.

Athena tidak memiliki server, sehingga tidak ada infrastruktur untuk disiapkan atau dikelola. Athena sangat tersedia dan menjalankan kueri menggunakan sumber daya komputasi di beberapa Availability Zone, secara otomatis merutekan kueri dengan tepat jika Availability Zone tertentu tidak terjangkau. Athena menggunakan Amazon S3 sebagai penyimpanan data yang mendasarinya, membuat data Anda sangat tersedia dan tahan lama. Amazon S3 menyediakan infrastruktur tahan lama untuk menyimpan data penting dan dirancang untuk daya tahan 99,999999999% objek. Data Anda disimpan secara berlebihan di berbagai fasilitas dan beberapa perangkat di setiap fasilitas.

Keamanan infrastruktur di Athena

Sebagai layanan terkelola, Amazon Athena dilindungi oleh keamanan jaringan AWS global. Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat Keamanan AWS Cloud. Untuk mendesain AWS lingkungan Anda menggunakan praktik terbaik untuk keamanan infrastruktur, lihat Perlindungan Infrastruktur dalam Kerangka Kerja yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses Athena melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS). Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Sandi cocok dengan sistem kerahasiaan maju sempurna (perfect forward secrecy, PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Selain itu, permintaan harus ditandatangani menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan prinsipal IAM. Atau Anda dapat menggunakan AWS Security Token Service (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk menandatangani permintaan.

Gunakan kebijakan IAM untuk membatasi akses ke operasi Athena. Setiap kali Anda menggunakan kebijakan IAM, pastikan bahwa Anda mengikuti praktik terbaik IAM. Untuk informasi selengkapnya tentang administrator, lihat Praktik terbaik keamanan dalam IAM di Panduan Pengguna IAM.

Kebijakan terkelola Athena mudah digunakan, dan diperbarui secara otomatis dengan tindakan yang diperlukan seiring berkembangnya layanan. Kebijakan yang dikelola pelanggan dan inline memungkinkan Anda untuk menyempurnakan kebijakan dengan menentukan tindakan Athena yang lebih terperinci dalam kebijakan. Memberikan akses yang sesuai ke lokasi data Amazon S3. Untuk informasi dan skenario mendetail tentang cara memberikan akses Amazon S3, lihat Contoh penelusuran: Mengelola akses di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon. Untuk informasi selengkapnya dan contoh tindakan Amazon S3 yang diizinkan, lihat kebijakan bucket contoh di Akses Lintas Akun.

Topik

- Connect ke Amazon Athena menggunakan antarmuka VPC endpoint

Connect ke Amazon Athena menggunakan antarmuka VPC endpoint

Anda dapat meningkatkan postur keamanan VPC Anda dengan menggunakan antarmuka VPC endpoint ()AWS PrivateLink dan titik akhir VPC di Virtual Private Cloud (AWS Glue VPC) Anda. Endpoint VPC antarmuka meningkatkan keamanan dengan memberi Anda kontrol atas tujuan apa yang dapat dicapai dari dalam VPC Anda. Masing-masing VPC endpoint diwakili oleh satu Antarmuka jaringan elastis (ENI) dengan alamat IP privat di subnet VPC Anda.

Titik akhir VPC antarmuka menghubungkan VPC Anda langsung ke Athena tanpa gateway internet, perangkat NAT, koneksi VPN, atau koneksi. AWS Direct Connect Instans dalam VPC Anda tidak memerlukan alamat IP publik untuk berkomunikasi dengan API Athena.

Untuk menggunakan Athena melalui VPC Anda, Anda harus terhubung dari sebuah instans yang ada di dalam VPC atau menghubungkan jaringan privat Anda ke VPC Anda dengan menggunakan Amazon Virtual Private Network (VPN) atau AWS Direct Connect. Untuk informasi tentang Amazon VPN, lihat Koneksi VPN di Panduan Pengguna Amazon Virtual Private Cloud. Untuk selengkapnya AWS Direct Connect, lihat Membuat sambungan di Panduan AWS Direct Connect Pengguna.

Athena mendukung titik akhir VPC di semua tempat Wilayah AWS Amazon VPC dan Athena tersedia.

Anda dapat membuat titik akhir VPC antarmuka untuk terhubung ke Athena menggunakan perintah or (). AWS Management Console AWS Command Line Interface AWS CLI Untuk informasi selengkapnya, lihat Membuat titik akhir antarmuka.

Setelah Anda membuat VPC endpoint antarmuka, jika Anda mengaktifkan nama host DNS privat untuk titik akhir, titik akhir Athena default (https://athena.*Region*.amazonaws.com) diselesaikan ke VPB endpoint Anda.

Jika Anda tidak mengaktifkan nama host DNS privat, Amazon VPC menyediakan nama titik akhir DNS yang dapat Anda gunakan dalam format berikut:

```
VPC_Endpoint_ID.athena.Region.vpce.amazonaws.com
```

Untuk informasi selengkapnya, lihat Titik akhir VPC Antarmuka (AWS PrivateLink) di Panduan Pengguna Amazon VPC.

Athena mendukung panggilan ke semua tindakan API-nya di dalam VPC Anda.

Membuat kebijakan titik akhir VPC untuk Athena

Anda dapat membuat kebijakan untuk titik akhir VPC Amazon untuk Athena untuk menentukan batasan seperti berikut:

- Prinsipal — Kepala sekolah yang dapat melakukan tindakan.
- Tindakan — Tindakan yang dapat dilakukan.
- Sumber daya — Sumber daya di mana tindakan dapat dilakukan.
- Hanya identitas tepercaya — Gunakan `aws:PrincipalOrgId` kondisi untuk membatasi akses hanya ke kredenal yang merupakan bagian dari organisasi Anda. AWS Ini dapat membantu mencegah akses oleh kepala sekolah yang tidak diinginkan.
- Hanya sumber daya tepercaya — Gunakan `aws:ResourceOrgId` kondisi untuk mencegah akses ke sumber daya yang tidak diinginkan.
- Hanya identitas dan sumber daya tepercaya — Buat kebijakan gabungan untuk titik akhir VPC yang membantu mencegah akses ke prinsip dan sumber daya yang tidak diinginkan.

Untuk informasi selengkapnya, lihat Mengontrol akses ke layanan dengan titik akhir VPC di Panduan Pengguna Amazon VPC dan Lampiran 2 — Contoh kebijakan titik akhir VPC di Whitepaper Membangun perimeter data. AWS AWS

Example — Kebijakan titik akhir VPC

Contoh berikut memungkinkan permintaan oleh identitas organisasi ke sumber daya organisasi dan memungkinkan permintaan oleh kepala AWS layanan.

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

Setiap kali Anda menggunakan kebijakan IAM, pastikan bahwa Anda mengikuti praktik terbaik IAM. Untuk informasi selengkapnya tentang administrator, lihat Praktik terbaik keamanan di IAM dalam Panduan Pengguna IAM.

Subnet bersama

Anda tidak dapat membuat, mendeskripsikan, memodifikasi, atau menghapus titik akhir VPC di subnet yang dibagikan dengan Anda. Namun, Anda dapat menggunakan titik akhir VPC di subnet yang dibagikan dengan Anda. Untuk informasi tentang berbagi VPC, lihat Membagikan VPC Anda dengan akun lain di Panduan Pengguna Amazon VPC.

Analisis konfigurasi dan kerentanan di Athena

Athena tidak memiliki server, jadi tidak ada infrastruktur untuk diatur atau dikelola. AWS menangani tugas-tugas keamanan dasar, seperti sistem operasi tamu (OS) dan patch database, konfigurasi firewall, dan pemulihan bencana. Prosedur ini telah ditinjau dan disertifikasi oleh pihak ketiga yang sesuai. Untuk detail selengkapnya, lihat AWS sumber daya berikut:

- Model tanggung jawab bersama
- Praktik terbaik untuk keamanan, identitas, & kepatuhan

Menggunakan Athena untuk menanyakan data yang terdaftar AWS Lake Formation

AWS Lake Formationmemungkinkan Anda untuk menentukan dan menegakkan basis data, tabel, dan kolom-level akses kebijakan saat menggunakan kueri Athena untuk membaca data yang disimpan di Amazon S3. Lake Formation menyediakan otorisasi dan tata kelola pada data yang disimpan di Amazon S3. Anda dapat menggunakan hierarki izin di Lake Formation untuk memberikan atau mencabut izin untuk membaca objek katalog data seperti basis data, tabel, dan kolom. Lake Formation menyederhanakan pengelolaan izin dan memungkinkan Anda menerapkan kontrol akses ketat (FGAC) untuk data Anda.

Anda dapat menggunakan Athena untuk mengkueri kedua data yang terdaftar dengan Lake Formation dan data yang tidak terdaftar dengan Lake Formation.

Izin Lake Formation berlaku saat menggunakan Athena untuk meminta data sumber dari lokasi Amazon S3 yang terdaftar dengan Lake Formation. Lake Formation izin juga berlaku saat Anda membuat basis data dan tabel yang mengarah ke terdaftar Amazon S3 lokasi data. Untuk menggunakan Athena dengan data yang terdaftar menggunakan Lake Formation, Athena harus dikonfigurasi untuk menggunakan AWS Glue Data Catalog.

Izin Lake Formation tidak berlaku saat menulis objek ke Amazon S3, juga tidak berlaku saat kueri data yang disimpan di Amazon S3 atau metadata yang tidak terdaftar dengan Lake Formation. Untuk

data sumber di Amazon S3 dan metadata yang tidak terdaftar di Lake Formation, akses ditentukan oleh kebijakan izin IAM untuk Amazon S3 dan tindakan. AWS Glue Lokasi hasil kueri Athena di Amazon S3 tidak dapat didaftarkan dengan Lake Formation, dan kebijakan izin IAM untuk akses kontrol Amazon S3. Selain itu, Lake Formation izin tidak berlaku untuk riwayat permintaan Athena. Anda dapat menggunakan grup kerja Athena untuk mengontrol akses ke riwayat kueri.

Untuk informasi selengkapnya tentang Lake Formation, lihatTanya Jawab FormationdanAWS Lake Formation Panduan Developer.

Topik

- Bagaimana Athena mengakses data yang terdaftar di Lake Formation
- Pertimbangan dan batasan saat menggunakan Athena untuk menanyakan data yang terdaftar di Lake Formation
- Mengelola izin pengguna Lake Formation dan Athena
- Menerapkan izin Lake Formation ke database dan tabel yang ada
- Menggunakan Lake Formation dan Athena JDBC dan ODBC driver untuk akses federasi ke Athena

Bagaimana Athena mengakses data yang terdaftar di Lake Formation

Alur kerja akses yang dijelaskan dalam bagian ini hanya berlaku saat menjalankan kueri Athena di lokasi Amazon S3 dan objek metadata yang terdaftar dengan Lake Formation. Untuk informasi selengkapnya, lihat Mendaftarkan data lake di Panduan AWS Lake Formation Pengembang. Selain mendaftarkan data, administrator Lake Formation menerapkan izin Lake Formation yang memberikan atau mencabut akses ke metadata di Katalog Data dan lokasi data di Amazon S3. Untuk informasi selengkapnya, lihat Keamanan dan kontrol akses ke metadata dan data di Panduan AWS Lake Formation Pengembang.

Setiap kali seorang utama Athena (pengguna, grup, atau peran) menjalankan kueri data yang terdaftar menggunakan Lake Formation, Lake Formation memverifikasi bahwa utama memiliki izin Pembentukan Danau yang sesuai untuk basis data, tabel, dan lokasi Amazon S3 yang sesuai untuk mengkueri. Jika utama memiliki akses, Lake Formationvendssementara mandat untuk Athena, dan permintaan berjalan.

Diagram berikut menggambarkan alur yang dijelaskan di atas.

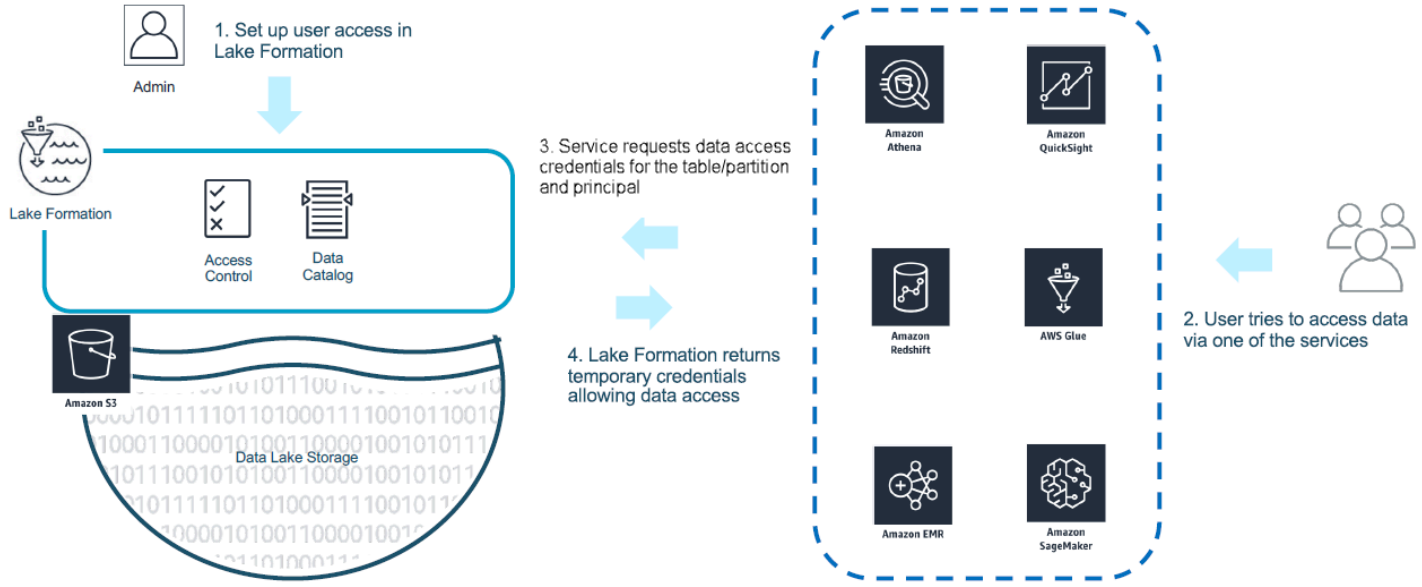
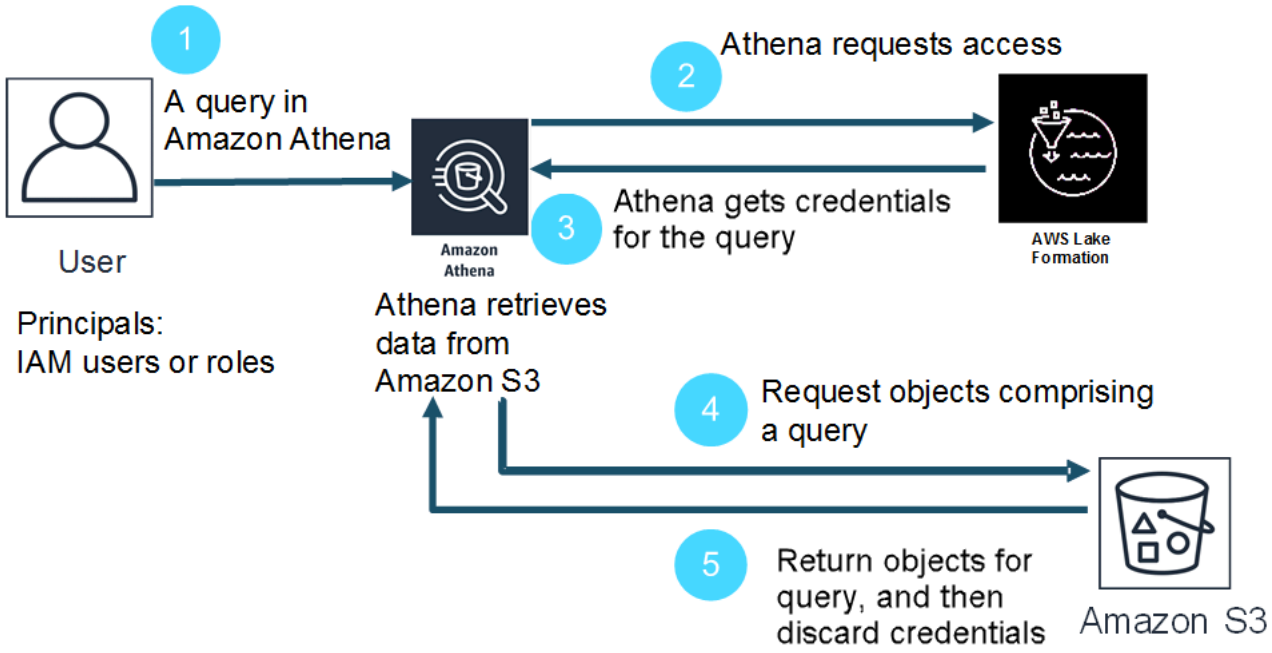


Diagram berikut menunjukkan bagaimana penjual kredenal bekerja di Athena query-by-query berdasarkan SELECT kueri hipotetis di atas meja dengan lokasi Amazon S3 terdaftar di Lake Formation:



1. utama menjalankan kueri SELECT di Athena.
2. Athena menganalisis kueri dan memeriksa izin Lake Formation untuk melihat apakah utama telah diberikan akses ke kolom tabel dan tabel.

3. Jika utama memiliki akses, Athena meminta mandat dari Lake Formation. Jika prinsipaltidakmemiliki akses, Athena masalah akses ditolak kesalahan.
4. Lake Formation mengeluarkan mandat untuk Athena untuk digunakan saat membaca data dari Amazon S3, bersama dengan daftar kolom yang diizinkan.
5. Athena menggunakan mandat sementara Lake Formation untuk kueri data dari Amazon S3. Setelah kueri selesai, Athena membuang mandat.

Pertimbangan dan batasan saat menggunakan Athena untuk menanyakan data yang terdaftar di Lake Formation

Pertimbangkan hal berikut saat menggunakan Athena untuk kueri data yang terdaftar di Lake Formation. Untuk informasi tambahan, lihat Masalah yang diketahui AWS Lake Formation di Panduan AWS Lake Formation Pengembang.

Pertimbangan dan batasan

- Metadata kolom terlihat oleh pengguna yang tidak sah dalam beberapa keadaan dengan Avro dan kustom SerDe
- Bekerja dengan izin Lake Formation untuk tampilan
- Kontrol akses halus Formasi Danau dan kelompok kerja Athena
- Lokasi hasil kueri Athena di Amazon S3 tidak terdaftar di Lake Formation
- Gunakan workgroup Athena untuk membatasi akses ke riwayat kueri
- Akses Katalog Data lintas akun
- Lokasi Amazon S3 terenkripsi CSE-KMS terdaftar di Lake Formation
- Lokasi data yang dipartisi yang terdaftar dengan Lake Formation harus dalam subdirektori tabel
- Buat tabel sebagai kueri pilih (CTAS) memerlukan izin tulis Amazon S3
- Izin DESCRIBE diperlukan pada database default

Metadata kolom terlihat oleh pengguna yang tidak sah dalam beberapa keadaan dengan Avro dan kustom SerDe

Otorisasi level kolom Lake Formation mencegah pengguna mengakses data dalam kolom yang pengguna tidak memiliki izin Lake Formation. Namun, dalam situasi tertentu, pengguna dapat mengakses metadata yang menjelaskan semua kolom dalam tabel, termasuk kolom yang mereka tidak memiliki izin untuk data.

Hal ini terjadi ketika metadata kolom disimpan dalam properti tabel untuk tabel menggunakan format penyimpanan Apache Avro atau menggunakan kustom Serializer/Deserializer (SerDe) di mana skema tabel didefinisikan dalam properti tabel bersama dengan definisi. SerDe Saat menggunakan Athena dengan Lake Formation, sebaiknya Anda meninjau isi properti tabel yang Anda daftarkan dengan Lake Formation dan, jika memungkinkan, batasi informasi yang tersimpan di properti tabel untuk mencegah metadata sensitif terlihat oleh pengguna.

Bekerja dengan izin Lake Formation untuk tampilan

Untuk data yang terdaftar di Lake Formation, pengguna Athena dapat membuat`VIEW`hanya jika mereka memiliki izin Lake Formation ke tabel, kolom, dan sumber lokasi data Amazon S3 tempat`VIEW`didasarkan. Setelah`VIEW`dibuat di Athena, izin Lake Formation dapat diterapkan ke`VIEW`. Izin level kolom tidak tersedia untuk`VIEW`. Pengguna yang memiliki izin Lake Formation ke`VIEW`tetapi tidak memiliki izin ke tabel dan kolom tempat tampilan didasarkan tidak dapat menggunakan`VIEW`untuk kueri data. Namun, pengguna dengan campuran izin ini dapat menggunakan pernyataan seperti`DESCRIBE VIEW`,`SHOW CREATE VIEW`, dan`SHOW COLUMNS`Untuk melihat`VIEW`Metadata. Untuk alasan ini, pastikan untuk menyelaraskan izin Lake Formation untuk setiap`VIEW`dengan izin tabel yang mendasari. Filter sel yang didefinisikan pada tabel tidak berlaku `VIEW` untuk tabel itu. Nama tautan sumber daya harus memiliki nama yang sama dengan sumber daya di akun asal. Ada batasan tambahan saat bekerja dengan tampilan dalam pengaturan lintas akun. Untuk informasi selengkapnya tentang menyiapkan izin untuk tampilan bersama di seluruh akun, lihatAkses Katalog Data lintas akun.

Kontrol akses halus Formasi Danau dan kelompok kerja Athena

Pengguna di workgroup Athena yang sama dapat melihat data yang telah dikonfigurasi oleh kontrol akses berbutir halus Lake Formation agar dapat diakses oleh workgroup. Untuk informasi selengkapnya tentang penggunaan kontrol akses berbutir halus di Lake Formation, lihat Mengelola kontrol akses berbutir halus menggunakan AWS Lake Formation di Blog Big Data.AWS

Lokasi hasil kueri Athena di Amazon S3 tidak terdaftar di Lake Formation

Hasil kueri lokasi di Amazon S3 untuk Athena tidak dapat didaftarkan dengan Lake Formation. Izin Lake Formation tidak membatasi akses ke lokasi ini. Kecuali Anda membatasi akses, pengguna Athena dapat mengakses file hasil kueri dan metadata saat mereka tidak memiliki izin Lake Formation untuk data tersebut. Untuk menghindari hal ini, kami sarankan Anda menggunakan grup kerja untuk menentukan lokasi untuk hasil kueri dan menyelaraskan keanggotaan grup kerja dengan Lake Formation izin. Anda kemudian dapat menggunakan kebijakan izin IAM untuk membatasi akses

ke lokasi hasil permintaan. Untuk informasi selengkapnya tentang string kueri, lihat Bekerja dengan hasil kueri, kueri terbaru, dan file keluaran.

Gunakan workgroup Athena untuk membatasi akses ke riwayat kueri

Sejarah kueri Athena mengekspos daftar kueri disimpan dan string kueri lengkap. Kecuali Anda menggunakan grup kerja untuk memisahkan akses ke riwayat kueri, pengguna Athena yang tidak berwenang untuk meminta data di Lake Formation dapat melihat string kueri yang dijalankan pada data tersebut, termasuk nama kolom, kriteria pemilihan, dan sebagainya. Kami menyarankan Anda menggunakan grup kerja untuk memisahkan riwayat permintaan, dan menyelaraskan keanggotaan Athena grup kerja dengan izin Lake Formation untuk membatasi akses. Untuk informasi selengkapnya, lihat Menggunakan kelompok kerja untuk mengontrol akses kueri dan biaya.

Akses Katalog Data lintas akun

Untuk mengakses katalog data di akun lain, Anda dapat menggunakan AWS Glue fitur lintas akun Athena atau mengatur akses lintas akun di Lake Formation.

Akses Katalog Data lintas akun Athena

Anda dapat menggunakan fitur AWS Glue katalog lintas akun Athena untuk mendaftarkan katalog di akun Anda. Kemampuan ini hanya tersedia di mesin Athena versi 2 dan versi yang lebih baru dan terbatas pada penggunaan wilayah yang sama antar akun. Untuk informasi selengkapnya, lihat Mendaftarkan akun AWS Glue Data Catalog dari akun lain.

Jika Katalog Data yang akan dibagikan memiliki kebijakan sumber daya yang dikonfigurasi AWS Glue, katalog tersebut harus diperbarui untuk mengizinkan akses ke AWS Resource Access Manager dan memberikan izin ke Akun B untuk menggunakan Katalog Data Akun A, seperti pada contoh berikut.

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

Untuk informasi selengkapnya, lihat Akses lintas akun ke katalog AWS Glue data.

Menyiapkan akses lintas akun di Lake Formation

AWS Lake Formation memungkinkan Anda menggunakan satu akun untuk mengelola Katalog Data pusat. Anda dapat menggunakan fitur ini untuk menerapkanAkses lintas akununtuk metadata Katalog Data dan data yang mendasari. Misalnya, akun pemilik dapat memberikan akun (penerima) lain izin SELECT terhadap tabel.

Untuk basis data bersama atau tabel untuk muncul di Athena Kueri Editor, Andabuat tautan sumber dayadi Lake Formation ke basis data bersama atau tabel. Saat akun penerima di Lake Formation menanyakan tabel pemilik, CloudTrailmenambahkan peristiwa akses data ke log untuk akun penerima dan akun pemilik.

Untuk tampilan bersama, ingatlah hal-hal berikut:

- Kueri dijalankan pada link sumber daya target, bukan pada tabel sumber atau tampilan, dan kemudian output dibagikan ke akun target.
- Tidak cukup hanya berbagi pandangan. Semua tabel yang terlibat dalam membuat tampilan harus menjadi bagian dari pangsa lintas akun.
- Nama tautan sumber daya yang dibuat pada sumber daya bersama harus cocok dengan nama sumber daya di akun pemilik. Jika nama tidak cocok, pesan kesalahan seperti Gagal menganalisis

tampilan tersimpan 'awsdatacatalog. *my-lf-resource-link*. *my-lf-view*': baris 3:3: Skema *schema_name* tidak ada terjadi.

Untuk informasi selengkapnya tentang akses lintas akun di Lake Formation, lihat sumber daya berikut di Panduan AWS Lake Formation Pengembang:

Akses lintas akun

Cara kerja tautan sumber daya di Lake Formation

Pencatatan lintas akun CloudTrail

Lokasi Amazon S3 terenkripsi CSE-KMS terdaftar di Lake Formation

Tabel Open Table Format (OTF) seperti Apache Iceberg yang memiliki karakteristik berikut tidak dapat ditanyakan dengan Athena:

- Tabel didasarkan pada lokasi data Amazon S3 yang terdaftar di Lake Formation.
- Objek di Amazon S3 dienkripsi menggunakan enkripsi sisi klien (CSE).
- Enkripsi menggunakan kunci yang AWS KMS dikelola pelanggan ()`CSE_KMS`.

Untuk menanyakan tabel non-OTF yang dienkripsi dengan `CSE_KMS` kunci), tambahkan blok berikut ke kebijakan AWS KMS kunci yang Anda gunakan untuk enkripsi CSE. <KMS_KEY_ARN>adalah ARN dari AWS KMS kunci yang mengenkripsi data. <IAM-ROLE-ARN>adalah ARN dari peran IAM yang mendaftarkan lokasi Amazon S3 di Lake Formation.

```
{
    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal": {
        "AWS": "*"
    },
    "Action": "kms:Decrypt",
    "Resource": "<KMS-KEY-ARN>",
    "Condition": {
        "ArnLike": {
            "aws:PrincipalArn": "<IAM-ROLE-ARN>"
        }
    }
}
```

Lokasi data yang dipartisi yang terdaftar dengan Lake Formation harus dalam subdirektori tabel

tabel dipartisi terdaftar dengan Lake Formation harus memiliki data dipartisi dalam direktori yang subdirektori dari tabel di Amazon S3. Misalnya, tabel dengan lokasi`s3://DOC-EXAMPLE-BUCKET/mytable`dan partisi`s3://DOC-EXAMPLE-BUCKET/mytable/dt=2019-07-11`,`s3://DOC-EXAMPLE-BUCKET/mytable/dt=2019-07-12`, dan sebagainya dapat didaftarkan dengan Lake Formation dan bertanya menggunakan Athena. Di sisi lain, tabel dengan lokasi`s3://DOC-EXAMPLE-BUCKET/mytable`dan partisi yang terletak di`s3://DOC-EXAMPLE-BUCKET/dt=2019-07-11`,`s3://DOC-EXAMPLE-BUCKET/dt=2019-07-12`, dan sebagainya, tidak dapat didaftarkan di Lake Formation. Karena partisi tersebut bukan subdirektori dari`s3://DOC-EXAMPLE-BUCKET/mytable`, mereka juga tidak dapat dibaca dari Athena.

Buat tabel sebagai kueri pilih (CTAS) memerlukan izin tulis Amazon S3

Buat Tabel Sebagai Laporan (CTAS) memerlukan akses menulis ke lokasi Amazon S3 tabel. Untuk menjalankan kueri CTAS pada data yang terdaftar dengan Lake Formation, pengguna Athena harus memiliki izin IAM untuk menulis ke tabel lokasi Amazon S3 selain izin Pembentukan Danau yang sesuai untuk membaca lokasi data. Untuk informasi selengkapnya, lihat Membuat tabel dari hasil query (CTAS).

Izin DESCRIBE diperlukan pada database default

Formation Lake FormationMENGGAMBARKANdiperlukan pada`default`basis data. Contoh AWS CLI perintah berikut memberikan `DESCRIBE` izin pada `default` database untuk pengguna `datalake_user1` dalam AWS akun`111122223333`.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --
permissions "DESCRIBE" --resource '{ "Database": {"Name":"default"}}
```

Untuk informasi selengkapnya, lihat referensi izin Lake Formation di Panduan AWS Lake Formation Pengembang.

Mengelola izin pengguna Lake Formation dan Athena

Lake Formation memberikan mandat untuk kueri toko data Amazon S3 yang terdaftar dengan Lake Formation. Jika sebelumnya Anda menggunakan kebijakan IAM untuk mengizinkan atau menolak izin untuk membaca lokasi data di Amazon S3, Anda dapat menggunakan izin Lake Formation sebagai gantinya. Namun, izin IAM lainnya masih diperlukan.

Setiap kali Anda menggunakan kebijakan IAM, pastikan bahwa Anda mengikuti praktik terbaik IAM. Untuk informasi selengkapnya tentang administrator, lihat Praktik Terbaik IAM dalam Panduan Pengguna IAM.

Bagian berikut meringkas izin yang diperlukan untuk menggunakan Athena untuk kueri data yang terdaftar di Lake Formation. Untuk informasi selengkapnya, lihat Keamanan di AWS Lake Formation dalam AWS Lake Formation Panduan Developer.

Ringkasan Izin

- Izin berbasis identitas untuk Lake Formation dan Athena
- Izin Amazon S3 untuk lokasi hasil kueri Athena
- Keanggotaan workgroup Athena ke riwayat kueri
- Izin Lake Formation untuk data
- Izin IAM untuk menulis ke lokasi Amazon S3
- Izin untuk data terenkripsi, metadata, dan hasil kueri Athena
- Izin berbasis sumber daya untuk bucket Amazon S3 di akun eksternal (opsional)

Izin berbasis identitas untuk Lake Formation dan Athena

Siapa pun yang menggunakan Athena untuk meminta data yang terdaftar dengan Lake Formation harus memiliki kebijakan izin IAM yang mengizinkan tindakan `lakeformation:GetDataAccess`. AWS kebijakan terkelola: AmazonAthenaFullAccess mengizinkan tindakan ini. Jika Anda menggunakan kebijakan inline, pastikan untuk memperbarui kebijakan izin untuk mengizinkan tindakan ini.

Dalam Lake Formation,Administrator data lakememiliki izin untuk membuat objek metadata seperti basis data dan tabel, memberikan izin Lake Formation untuk pengguna lain, dan mendaftarkan lokasi Amazon S3 baru. Untuk mendaftarkan lokasi baru, izin untuk peran terkait layanan untuk Lake Formation diperlukan. Untuk informasi selengkapnya, lihat Membuat administrator data lake dan izin peran terkait Layanan untuk Lake Formation di Panduan Pengembang AWS Lake Formation .

Pengguna Lake Formation dapat menggunakan Athena untuk menanyakan database, tabel, kolom tabel, dan penyimpanan data Amazon S3 yang mendasari berdasarkan izin Lake Formation yang diberikan kepada mereka oleh administrator data lake. Pengguna tidak dapat membuat basis data atau tabel, atau mendaftarkan lokasi Amazon S3 baru dengan Lake Formation. Untuk informasi selengkapnya, lihat Membuat pengguna data lake di Panduan AWS Lake Formation Pengembang.

Di Athena, kebijakan izin berbasis identitas, termasuk untuk grup kerja Athena, masih mengontrol akses ke tindakan Athena untuk Amazon Web Services pengguna akun. Selain itu, akses gabungan mungkin disediakan melalui otentikasi berbasis SAML yang tersedia dengan driver Athena. Lihat informasi selengkapnya di Menggunakan kelompok kerja untuk mengontrol akses kueri dan biaya, Kebijakan IAM untuk mengakses workgroup, dan Mengaktifkan akses federasi ke Athena API.

Untuk informasi selengkapnya, lihat Memberikan izin Lake Formation di Panduan AWS Lake Formation Pengembang.

Izin Amazon S3 untuk lokasi hasil kueri Athena

Hasil kueri lokasi di Amazon S3 untuk Athena tidak dapat didaftarkan dengan Lake Formation. Izin Lake Formation tidak membatasi akses ke lokasi ini. Kecuali Anda membatasi akses, pengguna Athena dapat mengakses file hasil kueri dan metadata saat mereka tidak memiliki izin Lake Formation untuk data tersebut. Untuk menghindari hal ini, kami sarankan Anda menggunakan grup kerja untuk menentukan lokasi untuk hasil kueri dan menyelaraskan keanggotaan grup kerja dengan Lake Formation izin. Anda kemudian dapat menggunakan kebijakan izin IAM untuk membatasi akses ke lokasi hasil permintaan. Untuk informasi selengkapnya tentang string kueri, lihat Bekerja dengan hasil kueri, kueri terbaru, dan file keluaran.

Keanggotaan workgroup Athena ke riwayat kueri

Sejarah kueri Athena mengekspos daftar kueri disimpan dan string kueri lengkap. Kecuali Anda menggunakan grup kerja untuk memisahkan akses ke riwayat kueri, pengguna Athena yang tidak berwenang untuk meminta data di Lake Formation dapat melihat string kueri yang dijalankan pada data tersebut, termasuk nama kolom, kriteria pemilihan, dan sebagainya. Kami menyarankan Anda menggunakan grup kerja untuk memisahkan riwayat permintaan, dan menyelaraskan keanggotaan Athena grup kerja dengan izin Lake Formation untuk membatasi akses. Untuk informasi selengkapnya, lihat Menggunakan kelompok kerja untuk mengontrol akses kueri dan biaya.

Izin Lake Formation untuk data

Selain izin dasar untuk menggunakan Lake Formation, pengguna Athena harus memiliki izin Lake Formation untuk mengakses sumber daya yang mereka kueri. Izin ini diberikan dan dikelola oleh administrator Lake Formation. Untuk informasi selengkapnya, lihat Keamanan dan kontrol akses ke metadata dan data di Panduan AWS Lake Formation Pengembang.

Izin IAM untuk menulis ke lokasi Amazon S3

Lake Formation izin untuk Amazon S3 tidak termasuk kemampuan untuk menulis ke Amazon S3. Buat Tabel Sebagai Laporan (CTAS) memerlukan akses menulis ke lokasi Amazon S3 tabel. Untuk

menjalankan kueri CTAS pada data yang terdaftar dengan Lake Formation, pengguna Athena harus memiliki izin IAM untuk menulis ke tabel lokasi Amazon S3 selain izin Pembentukan Danau yang sesuai untuk membaca lokasi data. Untuk informasi selengkapnya, lihat Membuat tabel dari hasil query (CTAS).

Izin untuk data terenkripsi, metadata, dan hasil kueri Athena

Data sumber yang mendasari di Amazon S3 dan metadata dalam Katalog Data yang terdaftar dengan Lake Formation dapat dienkripsi. Tidak ada perubahan pada cara Athena menangani enkripsi hasil kueri saat menggunakan Athena untuk kueri data terdaftar dengan Lake Formation. Untuk informasi selengkapnya, lihat Mengenkripsi hasil kueri Athena yang disimpan di Amazon S3.

- Mengenkripsi data sumber — Enkripsi data sumber lokasi data Amazon S3 didukung. Pengguna Athena yang mencari lokasi Amazon S3 terenkripsi yang terdaftar dengan Lake Formation memerlukan izin untuk mengenkripsi dan mendekripsi data. Untuk informasi selengkapnya tentang persyaratan, lihat Opsi enkripsi Amazon S3 yang didukung danIzin untuk data terenkripsi di Amazon S3.
- Enkripsi metadata— Enkripsi metadata dalam Katalog Data didukung.
  Bagi utama yang menggunakan Athena, kebijakan berbasis identitas harus mengizinkan`"kms:GenerateDataKey"`,`"kms:Decrypt"`, dan`"kms:Encrypt"`Tindakan untuk kunci yang digunakan untuk mengenkripsi metadata. Untuk informasi selengkapnya, lihat Mengenkripsi Katalog Data Anda di Panduan AWS Glue Pengembang dan. Akses dari Athena ke metadata terenkripsi di AWS Glue Data Catalog

Izin berbasis sumber daya untuk bucket Amazon S3 di akun eksternal (opsional)

Untuk kueri lokasi data Amazon S3 di akun yang berbeda, berbasis sumber daya IAM kebijakan (bucket kebijakan) harus memungkinkan akses ke lokasi. Untuk informasi selengkapnya, lihat Akses lintas akun di Athena ke ember Amazon S3.

Untuk informasi tentang mengakses Katalog Data di akun lain, lihatAkses Katalog Data lintas akun Athena.

Menerapkan izin Lake Formation ke database dan tabel yang ada

Jika Anda baru mengenal Athena dan menggunakan Lake Formation untuk mengonfigurasi akses ke data kueri, Anda tidak perlu mengonfigurasi kebijakan IAM sehingga pengguna dapat membaca data Amazon S3 dan membuat metadata. Anda dapat menggunakan Lake Formation untuk mengelola izin.

Mendaftarkan data dengan Lake Formation dan memperbarui kebijakan izin IAM bukan merupakan persyaratan. Jika data tidak terdaftar di Lake Formation, pengguna Athena yang memiliki izin yang sesuai di Amazon S3—dan AWS Glue, jika berlaku—dapat melanjutkan kueri data yang tidak terdaftar di Lake Formation.

Jika Anda memiliki pengguna Athena yang menanyakan data yang tidak terdaftar di Lake Formation, Anda dapat memperbarui izin IAM untuk Amazon S3—dan AWS Glue Data Catalog, jika berlaku—sehingga Anda dapat menggunakan izin Lake Formation untuk mengelola akses pengguna secara terpusat. Untuk izin untuk membaca lokasi data Amazon S3, Anda dapat memperbarui berbasis sumber daya dan kebijakan berbasis identitas untuk memodifikasi izin Amazon S3. Untuk akses ke metadata, jika Anda mengonfigurasi kebijakan tingkat sumber daya untuk kontrol akses berbutir halus, Anda AWS Glue dapat menggunakan izin Lake Formation untuk mengelola akses.

Untuk informasi selengkapnya, lihat Akses berbutir halus ke database dan tabel di AWS Glue Data Catalog dan Memutakhirkan izin AWS Glue data ke AWS Lake Formation model di Panduan AWS Lake Formation Pengembang.

Menggunakan Lake Formation dan Athena JDBC dan ODBC driver untuk akses federasi ke Athena

Athena JDBC dan ODBC driver mendukung Gabungan berbasis SAML 2.0 dengan Athena menggunakan Okta dan Microsoft Active Directory Federation Services (AD FS) penyedia identitas. Dengan mengintegrasikan Amazon Athena AWS Lake Formation dengan, Anda mengaktifkan otentikasi berbasis SAMP ke Athena dengan kredensi perusahaan. Dengan Lake Formation dan AWS Identity and Access Management (IAM), Anda dapat mempertahankan kontrol akses tingkat kolom berbutir halus atas data yang tersedia untuk pengguna SAMP. Dengan driver Athena JDBC dan ODBC, akses gabungan tersedia untuk alat atau program akses.

Untuk menggunakan Athena untuk mengakses sumber data yang dikendalikan oleh Lake Formation, Anda perlu mengaktifkan federasi berbasis SAMP 2.0 dengan mengonfigurasi peran penyedia identitas (iDP) dan (IAM) Anda. AWS Identity and Access Management Untuk langkah mendetail, lihat Tutorial: Mengkonfigurasi akses federasi untuk pengguna Okta ke Athena menggunakan Lake Formation dan JDBC.

Prasyarat

Untuk menggunakan Amazon Athena dan Lake Formation untuk akses gabungan, Anda harus memenuhi persyaratan berikut:

- Anda mengelola identitas perusahaan menggunakan penyedia identitas berbasis SAML yang ada, seperti Okta atau Microsoft Active Directory Federation Services (AD FS).
- Anda menggunakan AWS Glue Data Catalog sebagai toko metadata.
- Anda menentukan dan mengelola izin di Lake Formation untuk mengakses basis data, tabel, dan kolom di AWS Glue Data Catalog. Lihat informasi selengkapnya di Panduan Developer AWS Lake Formation.
- Anda menggunakan versi 2.0.14 atau yang lebih baru dari driver Athena JDBC atau versi 1.1.3 atau yang lebih baru dari driver Athena ODBC.

Pertimbangan dan batasan

Saat menggunakan driver Athena JDBC atau ODBC dan Lake Formation untuk mengonfigurasi akses gabungan ke Athena, ingatlah hal-hal berikut:

- Saat ini, driver Athena JDBC dan driver ODBC mendukung Okta, Microsoft Active Directory Federation Services (AD FS), dan penyedia identitas Azure AD. Meskipun driver Athena JDBC memiliki kelas SAMP generik yang dapat diperluas untuk menggunakan penyedia identitas lain, dukungan untuk ekstensi khusus yang memungkinkan penyedia identitas lain () IdPs untuk digunakan dengan Athena mungkin terbatas.
- Akses federasi menggunakan driver JDBC dan ODBC tidak kompatibel dengan fitur propagasi identitas terpercaya IAM Identity Center.
- Saat ini, Anda tidak dapat menggunakan konsol Athena untuk mengonfigurasi dukungan untuk penggunaan IdP dan SAML dengan Athena. Untuk mengonfigurasi dukungan ini, Anda menggunakan penyedia identitas pihak ketiga, Lake Formation dan IAM konsol manajemen, dan JDBC atau ODBC driver klien.
- Anda harus memahamiSpesifikasi SAML 2.0dan cara kerjanya dengan penyedia identitas Anda sebelum mengonfigurasi penyedia identitas dan SAML untuk digunakan dengan Lake Formation dan Athena.
- Penyedia SAMP dan driver Athena JDBC dan ODBC disediakan oleh pihak ketiga, sehingga dukungan AWS melalui masalah yang terkait dengan penggunaannya mungkin terbatas.

Topik

- Tutorial: Mengkonfigurasi akses federasi untuk pengguna Okta ke Athena menggunakan Lake Formation dan JDBC

Tutorial: Mengkonfigurasi akses federasi untuk pengguna Okta ke Athena menggunakan Lake Formation dan JDBC

Tutorial ini menunjukkan kepada Anda cara mengkonfigurasi Okta, AWS Lake Formation, AWS Identity and Access Management izin, dan driver Athena JDBC untuk mengaktifkan penggunaan federasi Athena berbasis SAMP. Lake Formation menyediakan kontrol akses berbutir halus atas data yang tersedia di Athena untuk pengguna berbasis SAML. Untuk mengatur konfigurasi ini, tutorial menggunakan konsol pengembang Okta, konsol AWS IAM dan Lake Formation, dan alat SQL Workbench/J.

Prasyarat

Tutorial ini mengasumsikan bahwa Anda telah melakukan hal berikut:

- Membuat akun Amazon Web Services. Untuk membuat akun, kunjungiAmazon Web Services.
- Menyiapkan lokasi hasil kueripara Athena en Amazon S3.
- Mendaftarkan lokasi bucket data Amazon S3dengan Lake Formation.
- Ditetapkan sebuahbasis datadantabelpadaAWS Glue Katalog dataYang menunjuk ke data Anda di Amazon S3.
  - Jika Anda belum mendefinisikan tabel, jalankan AWS Glue crawler atau gunakan Athena untuk menentukan database dan satu atau beberapa tabel untuk data yang ingin Anda akses.
  - Tutorial ini menggunakan tabel berdasarkan dataset perjalanan taksi NYC yang tersedia di Registry data terbuka pada. AWS Tutorial menggunakan nama database`tripdb`dan nama tabel`nyctaxi`.

Langkah tutorial

Langkah 1: Buat akun Okta

Tutorial ini menggunakan Okta sebagai penyedia identitas berbasis SAML. Jika Anda belum memiliki akun Okta, Anda dapat membuat akun gratis. Akun Okta diperlukan agar Anda dapat membuat aplikasi Okta untuk otentikasi SAML.

Untuk membuat akun

1. Untuk menggunakan Okta, navigasikan keHalaman pendaftaran developer Oktadan membuat akun percobaan Okta gratis. Layanan Edisi Developer tidak dipungut biaya hingga batas yang ditentukan oleh Okta dideveloper.okta.com/pricing.
2. Saat Anda menerima email aktivasi, aktifkan akun Anda.

   Nama domain Okta akan diberikan kepada Anda. Simpan nama domain untuk referensi. Kemudian, Anda menggunakan nama domain (`< okta-idp-domain >`) dalam string JDBC yang terhubung ke Athena.

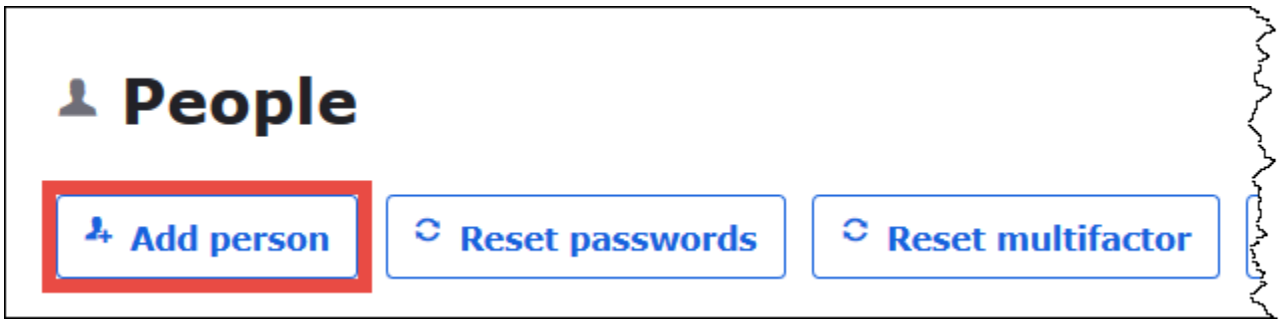
Langkah 2: Tambahkan pengguna dan grup ke Okta

Pada langkah ini, Anda menggunakan konsol Okta untuk melakukan tugas berikut:

- Buat dua pengguna Okta.
- Buat dua grup Okta.
- Tambahkan satu pengguna Okta ke setiap grup Okta.

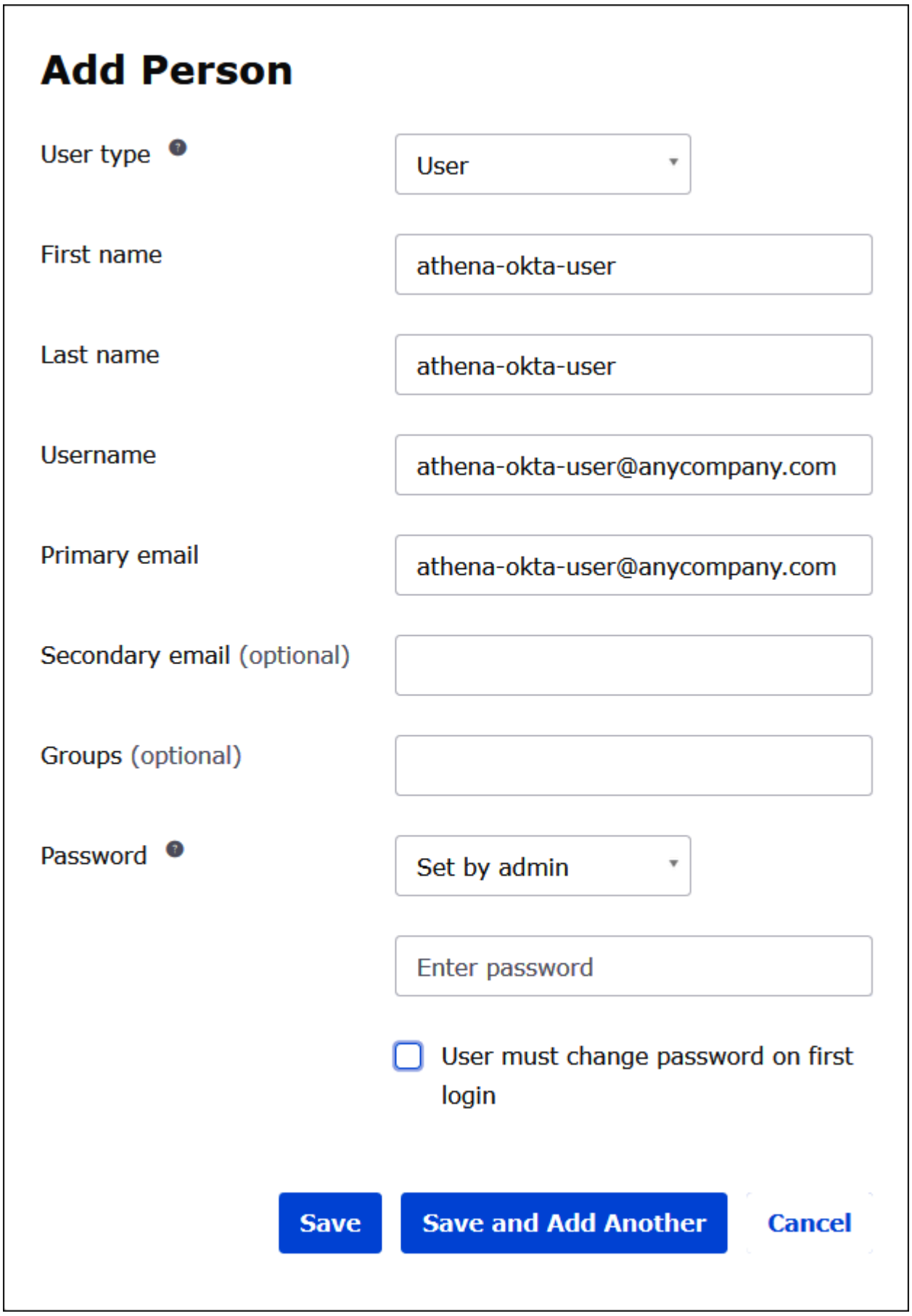
Untuk menambahkan pengguna ke Okta

1. Setelah Anda mengaktifkan akun Okta Anda, masuk sebagai pengguna administratif ke domain Okta yang ditetapkan.
2. Di panel navigasi kiri, pilihDirektori, lalu pilihOrang.
3. PilihTambahkan Oranguntuk menambahkan pengguna baru yang akan mengakses Athena melalui driver JDBC.

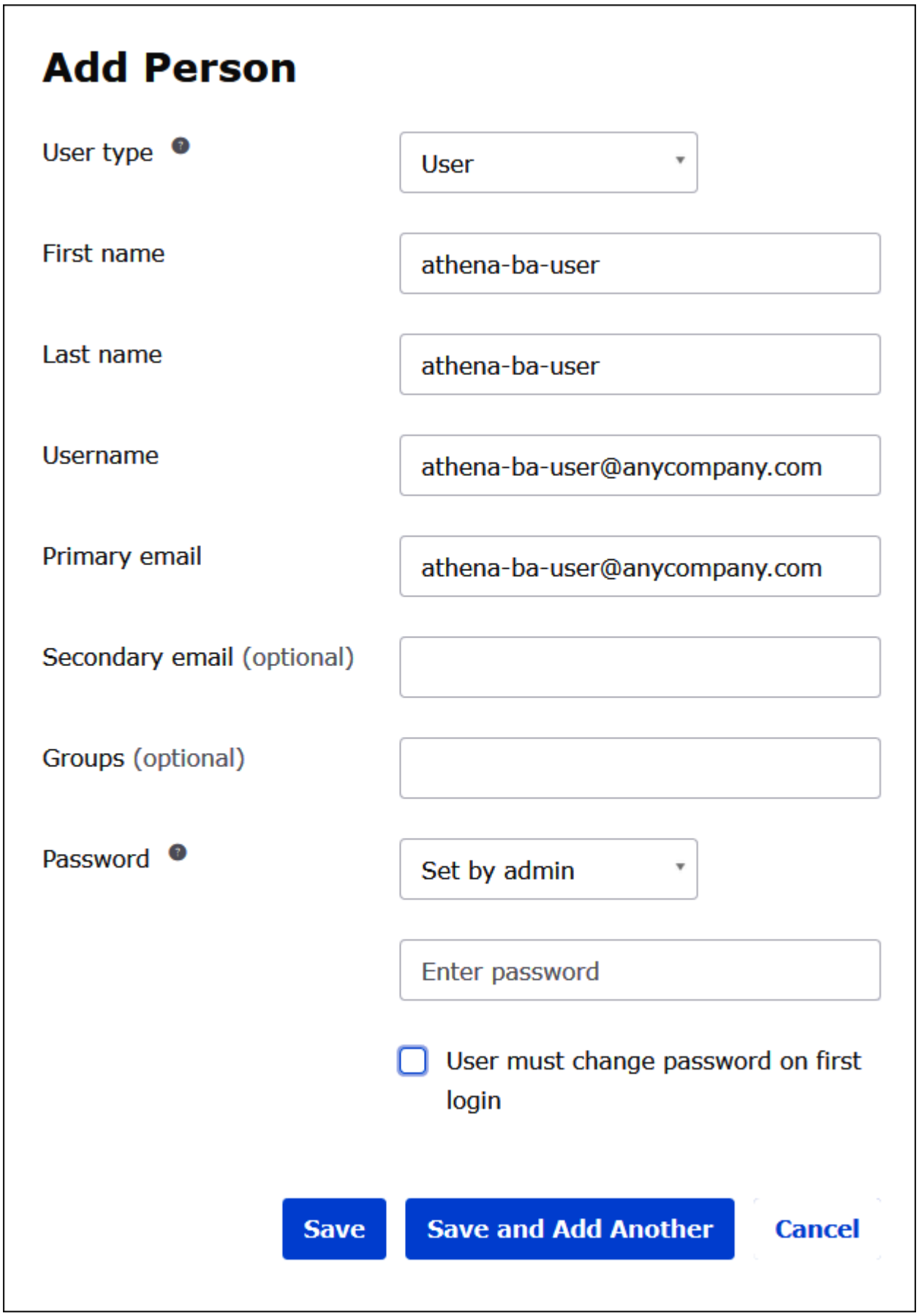


4. DiTambahkan OrangDi kotak dialog, masukkan informasi yang diperlukan.

   - Masukkan nilai untukNama depandanNama terakhir. Tutorial ini menggunakan *athena-okta-user*.
   - MasukkanNama penggunadanEmail utama. Tutorial ini menggunakan *athena-okta-user@anycompany .com*.
   - UntukKata SandiPilihDitetapkan oleh admin, dan kemudian memberikan sandi. Tutorial ini membersihkan pilihan untukPengguna harus mengubah kata sandi pada login pertama; persyaratan keamanan Anda mungkin berbeda.


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. PilihSimpan dan Tambah Lainnya.
6. Masukkan informasi untuk pengguna lain. Contoh ini menambahkan pengguna analis bisnis *athena-ba-user@anycompany .com*.

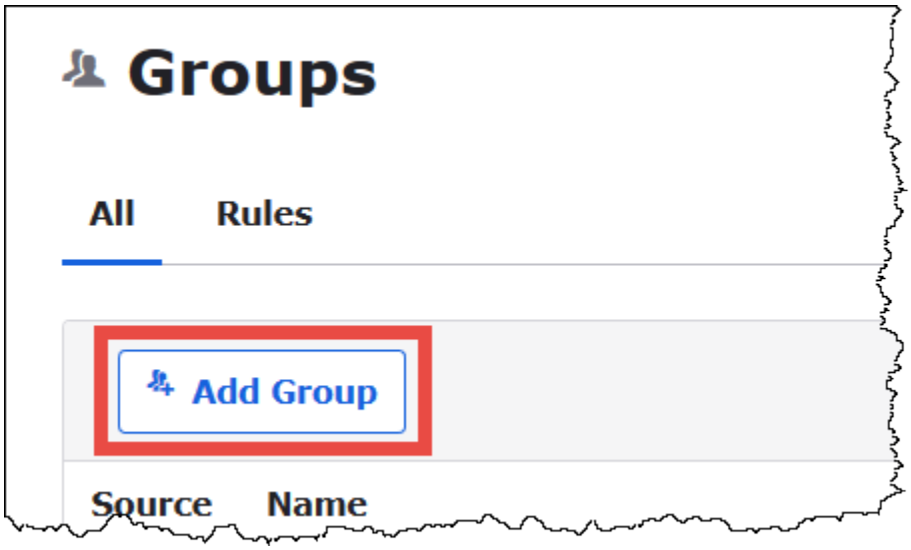

Add Person
User type
User
First name
athena-ba-user
Last name
athena-ba-user
Username
athena-ba-user@anycompany.com
Primary email
athena-ba-user@anycompany.com
Secondary email (optional)
Groups (optional)
Password
Set by admin
Enter password
User must change password on first login
Save
Save and Add Another
Cancel

7. Pilih Simpan.

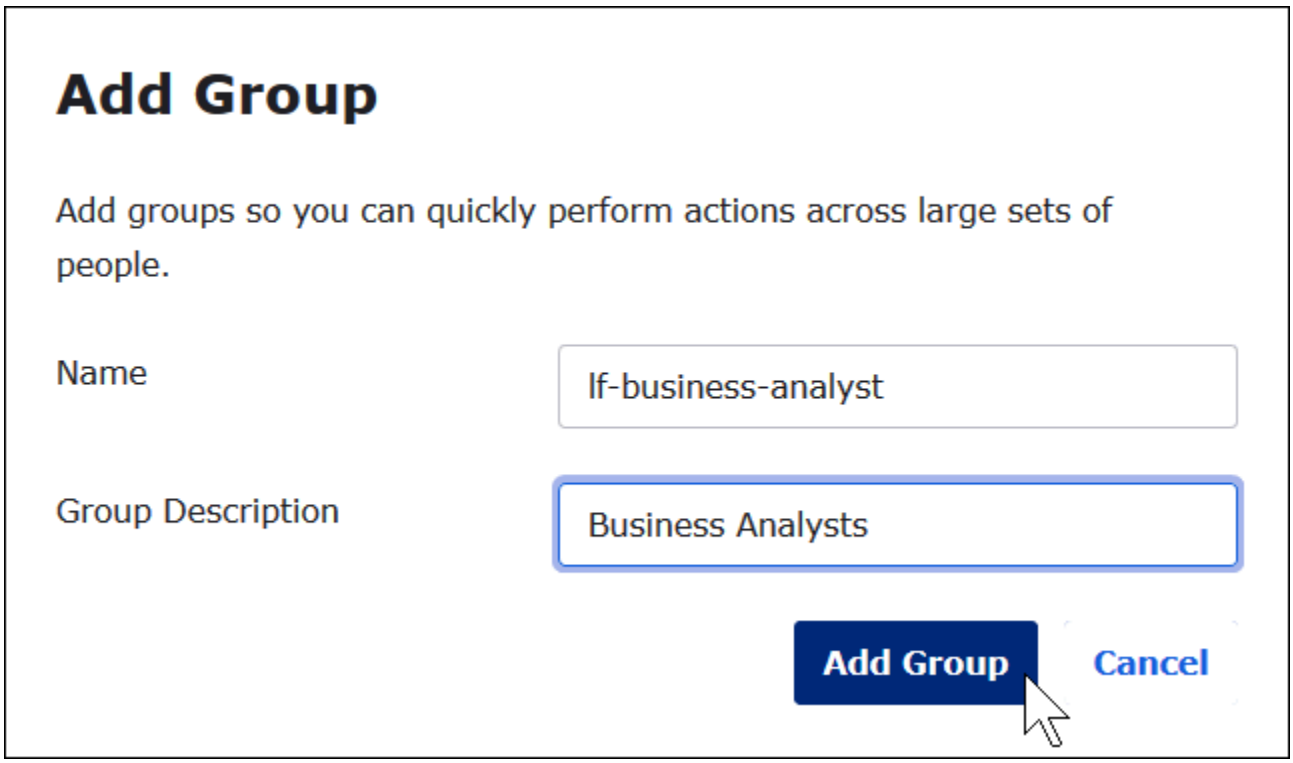
Dalam prosedur berikut, Anda menyediakan akses untuk dua grup Okta melalui driver Athena JDBC dengan menambahkan grup “Analis Bisnis” dan grup “Developer”.

Untuk menambahkan grup Okta

1. Dalam panel navigasi, pilih Groups lalu pilih Create New Group.
2. PadaGrup, pilihTambah Grup.



3. DiTambah GrupDi kotak dialog, masukkan informasi yang diperlukan.
   - Untuk Nama, masukkan *`lf-business-analyst`*.
   - UntukDeskripsi grup, masukkan*`Analis Bisnis`*.

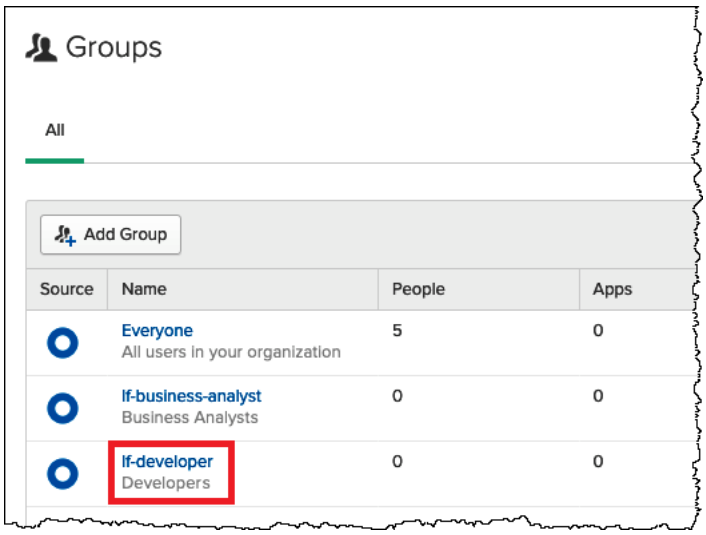


4. Pilih Tambahkan grup kunci.
5. PadaGrup, pilihTambah GrupSekali lagi. Kali ini Anda akan memasukkan informasi untuk grup Developer.
6. Masukkan informasi yang diperlukan.
   - UntukNama, masukkan*Pengembang lf*.
   - UntukDeskripsi grup, masukkan*Pengembang*.
7. Pilih Tambahkan grup kunci.

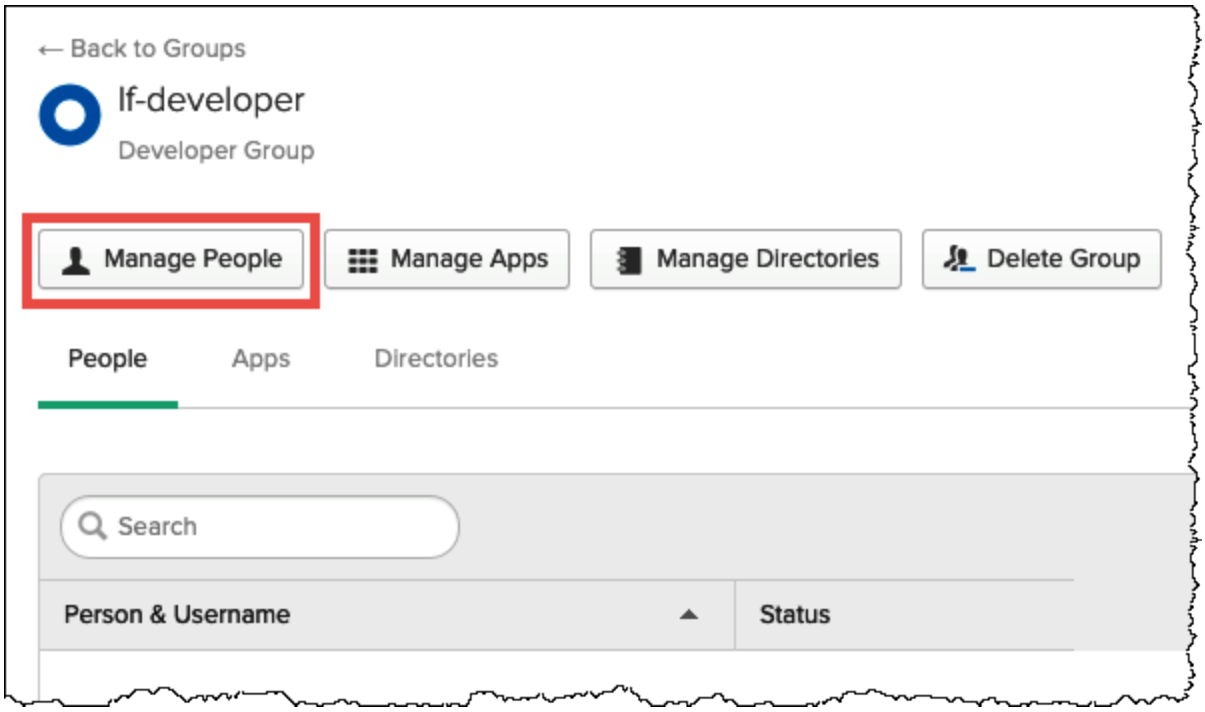
Setelah Anda memiliki dua pengguna dan dua grup, Anda siap untuk menambahkan pengguna ke setiap grup.

Untuk menambahkan pengguna ke grup

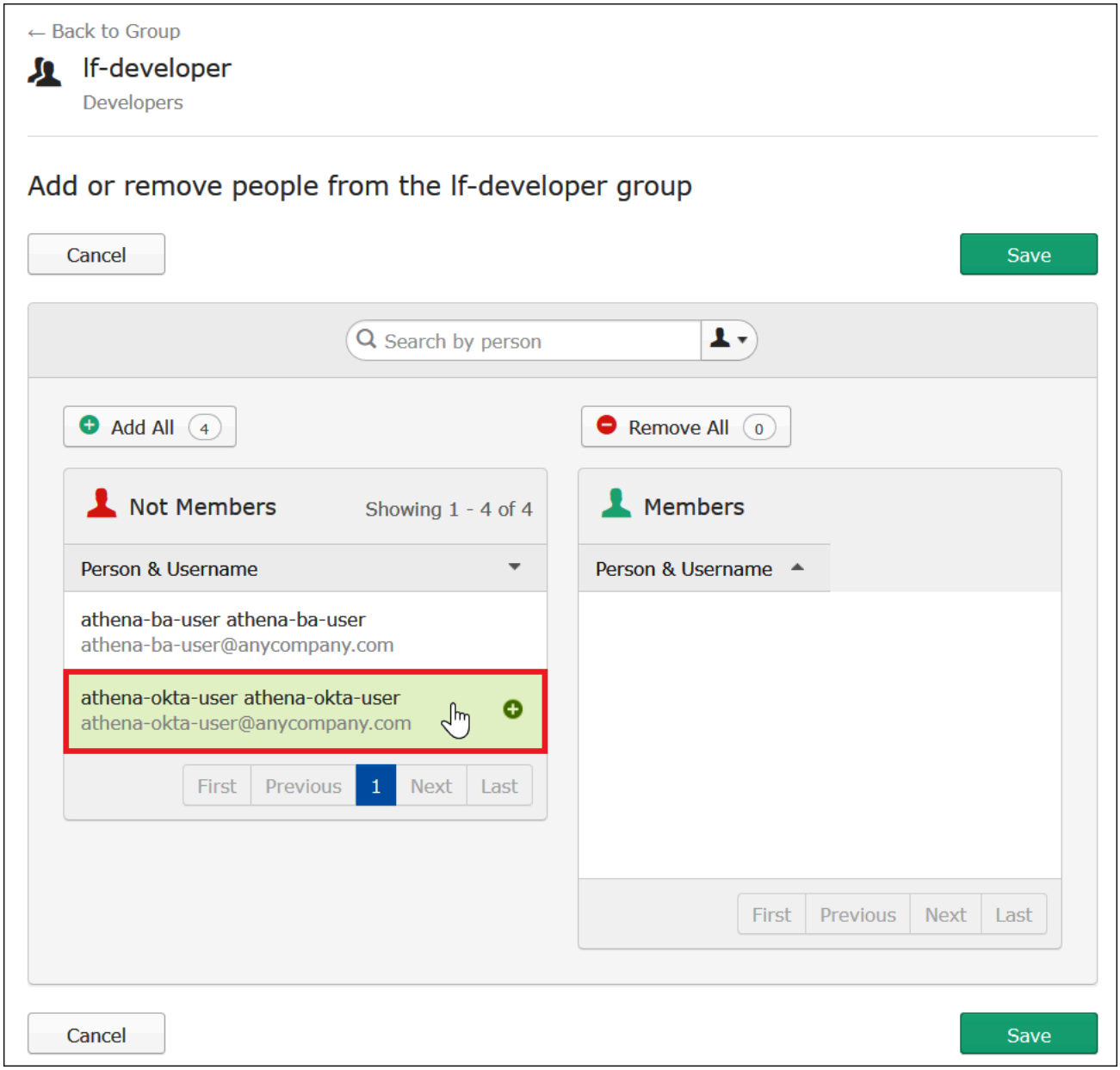
1. PadaGrup, pilihPengembang lfGrup yang baru saja Anda buat. Anda akan menambahkan salah satu pengguna Okta yang Anda buat sebagai developer ke grup ini.



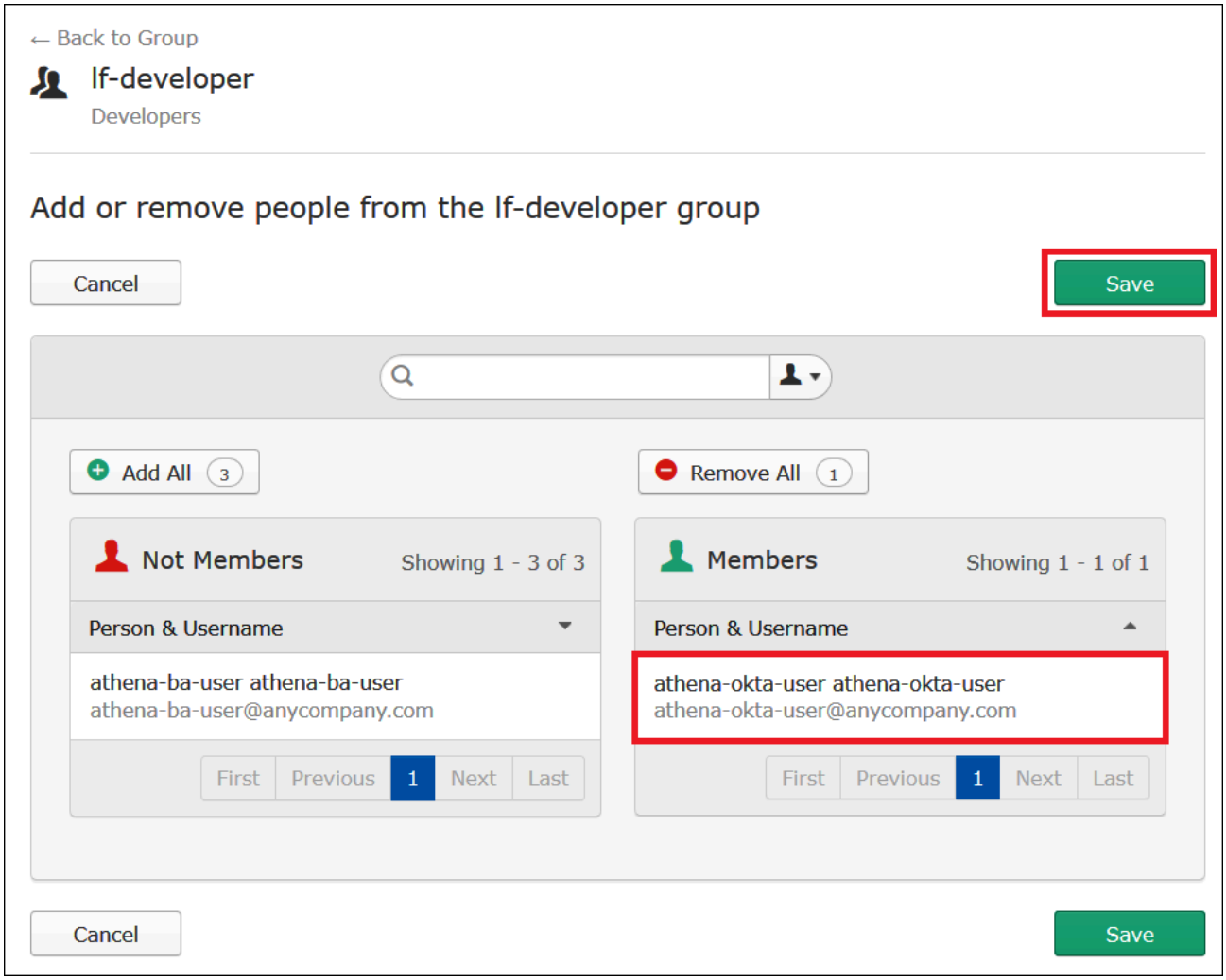
2. PilihMengelola orang.



3. Dari daftar Bukan Anggota, pilih athena-okta-user.

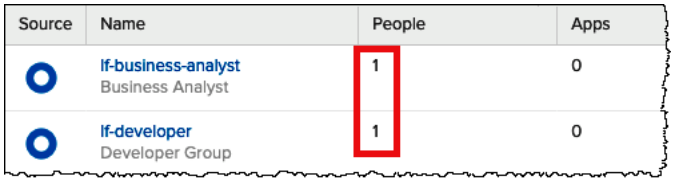


Entri untuk pengguna bergerak dariBukan anggotadaftar di sebelah kiri keAnggotaDaftar di sebelah kanan.



4. Pilih Simpan.
5. PilihKembali ke Grup, atau pilihDirektori, lalu pilihGrup.
6. Pilih lf-business-analystgrup.
7. PilihMengelola orang.
8. Tambahkan athena-ba-userke daftar Anggota lf-business-analystgrup, lalu pilih Simpan.
9. PilihKembali ke Grup, atau pilihDirektori,Grup.

ParameterGrupsekarang menunjukkan bahwa setiap grup memiliki satu pengguna Okta.



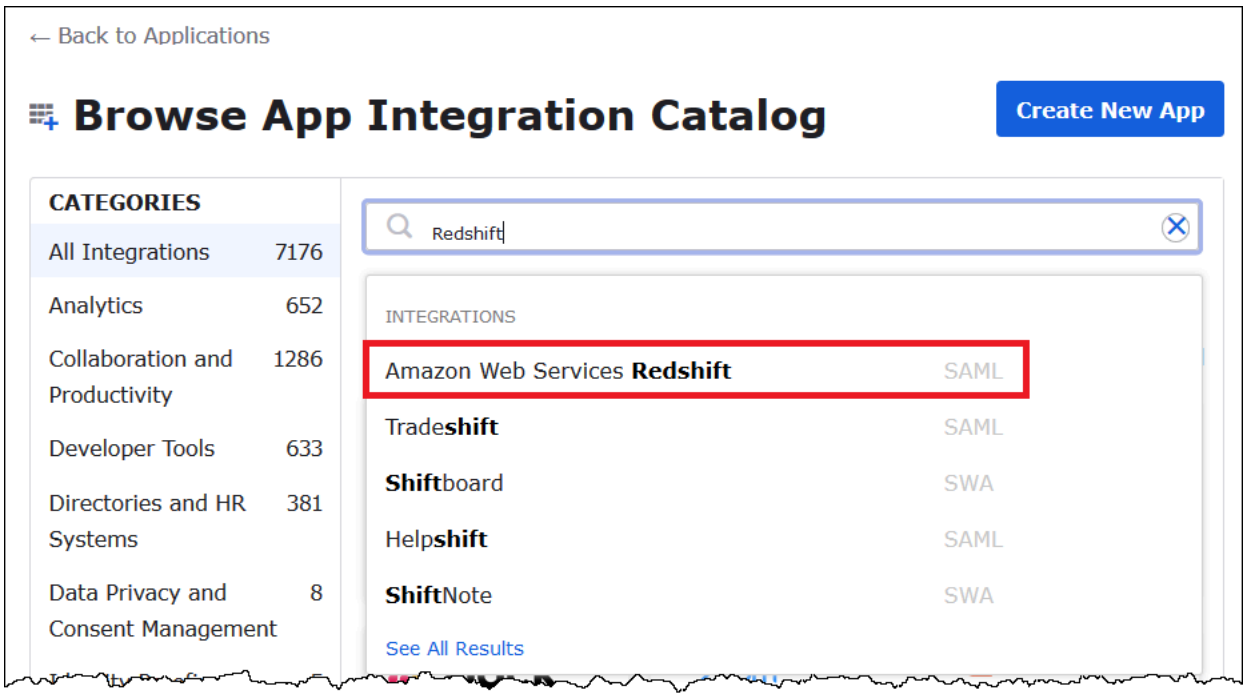
Langkah 3: Siapkan aplikasi Okta untuk otentikasi SAFL

Pada langkah ini, Anda menggunakan konsol developer Okta untuk melakukan tugas berikut:

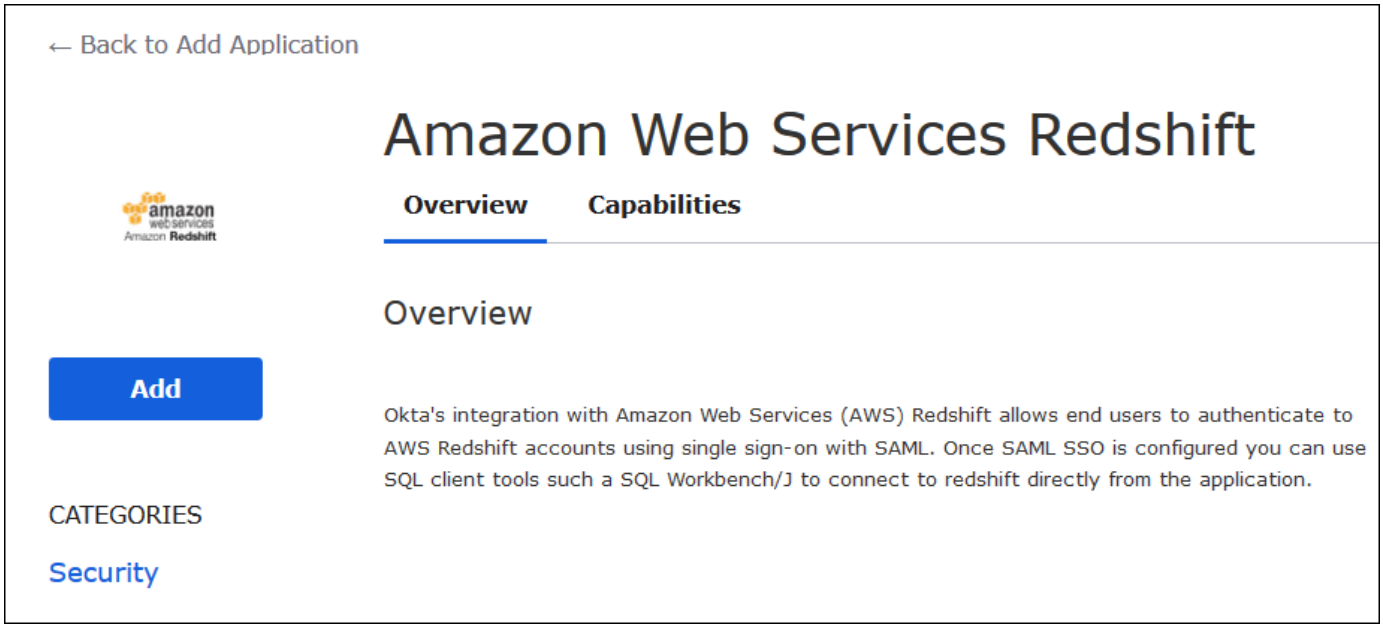
- Tambahkan aplikasi SAFL untuk digunakan dengan AWS.
- Tetapkan aplikasi ke pengguna Okta.
- Menetapkan aplikasi ke grup Okta.
- Unduh metadata penyedia identitas yang dihasilkan untuk digunakan nanti dengan AWS.

Cara menambahkan aplikasi untuk otentikasi SAML

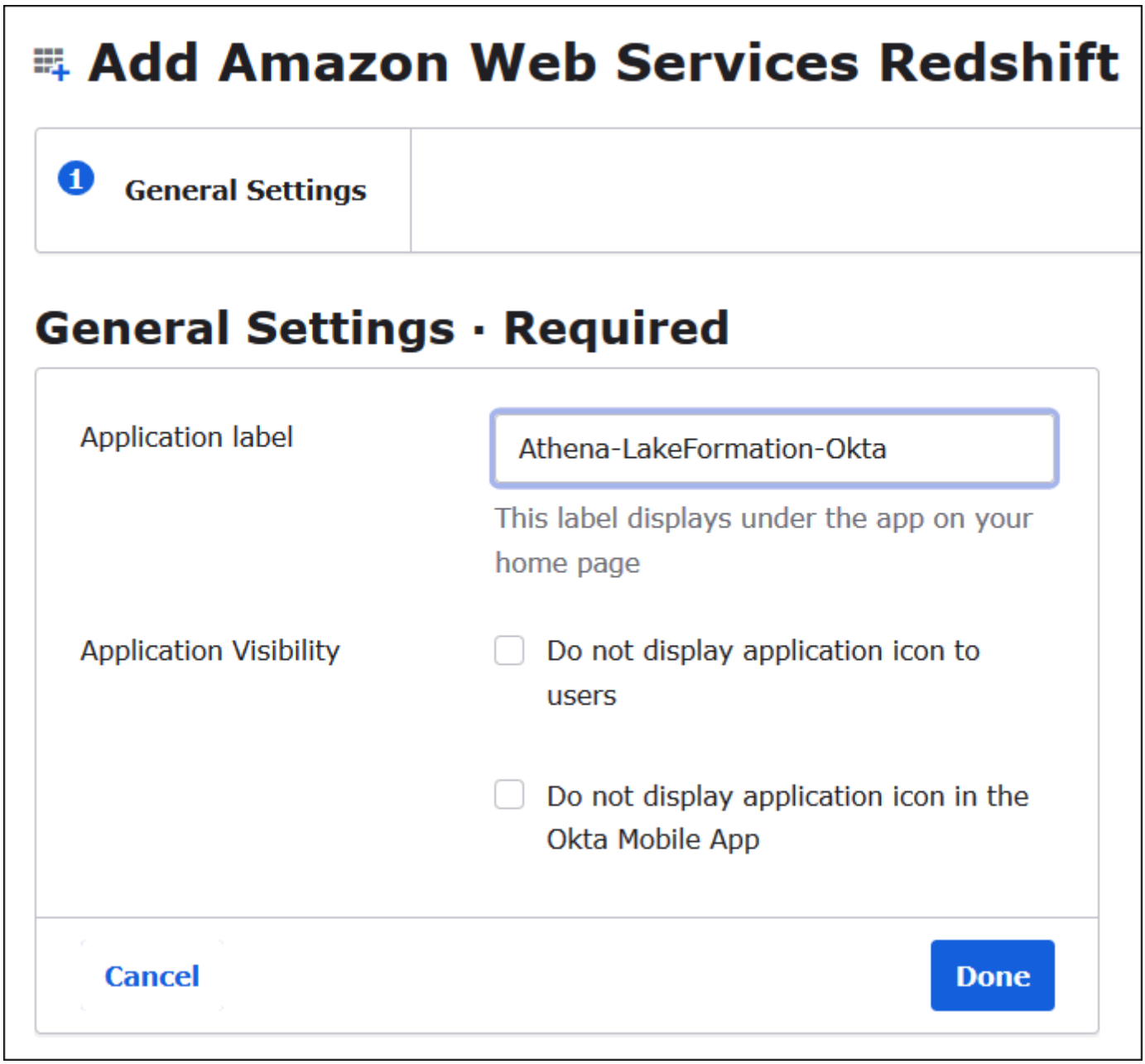
1. Di panel navigasi Okta, pilihAplikasi,Aplikasisehingga Anda dapat mengonfigurasi aplikasi Okta untuk otentikasi SAML ke Athena.
2. KlikJelajahi Katalog Aplikasi.
3. Dalam kotak pencarian, masukkan **`Redshift`**.
4. PilihRedshift Amazon Web Services. Aplikasi Okta dalam tutorial ini menggunakan integrasi SAML yang ada untuk Amazon Redshift.



5. PadaRedshift Amazon Web Services, pilihTambahkanUntuk membuat aplikasi berbasis SAML untuk Amazon Redshift.



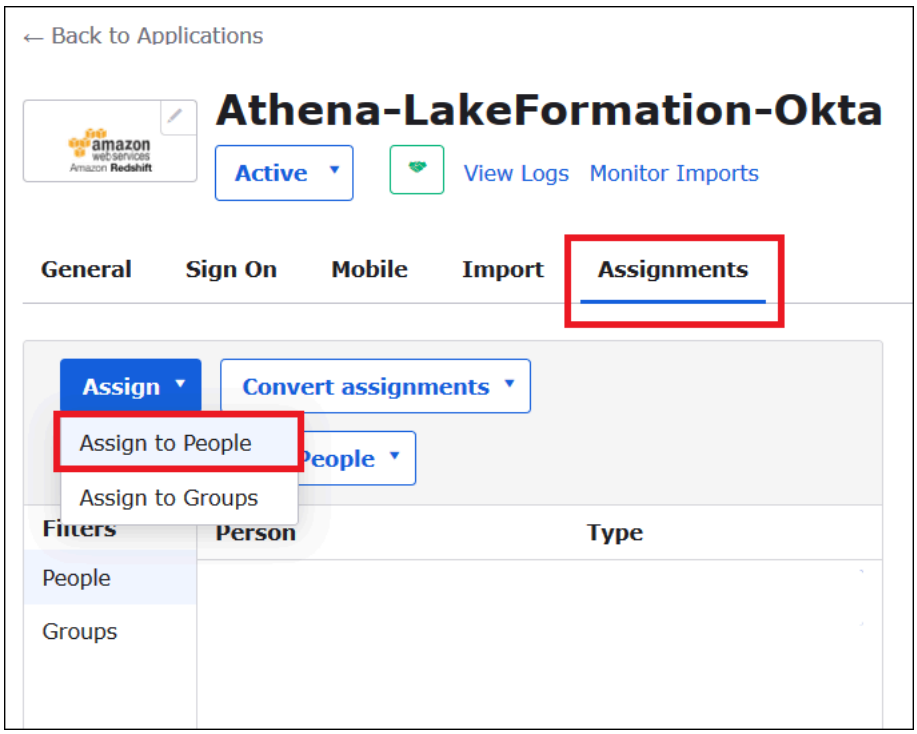
6. UntukAplikasi label, masukkan`Athena-LakeFormation-Okta`, lalu pilihSelesai.



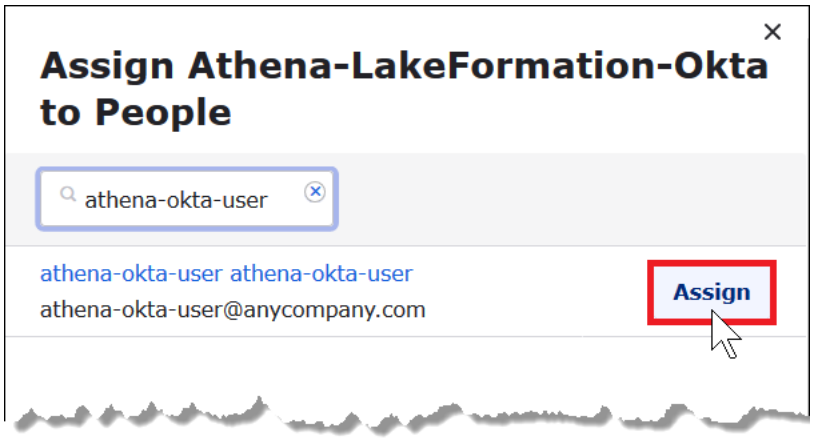
Sekarang setelah Anda membuat aplikasi Okta, Anda dapat menetapkannya ke pengguna dan grup yang Anda buat.

Menetapkan aplikasi ke pengguna dan grup

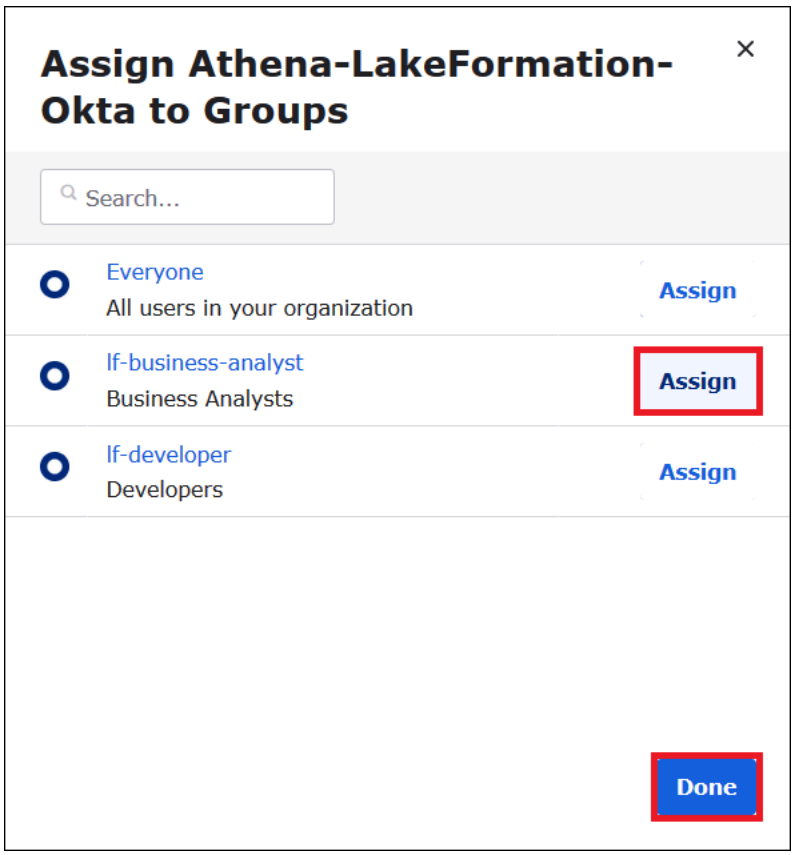
1. Pada halaman Aplikasi, pilih aplikasi Athena- LakeFormation -Okta.
2. PadaPenugasantab, pilihTetapkan,Tugaskan ke Orang.



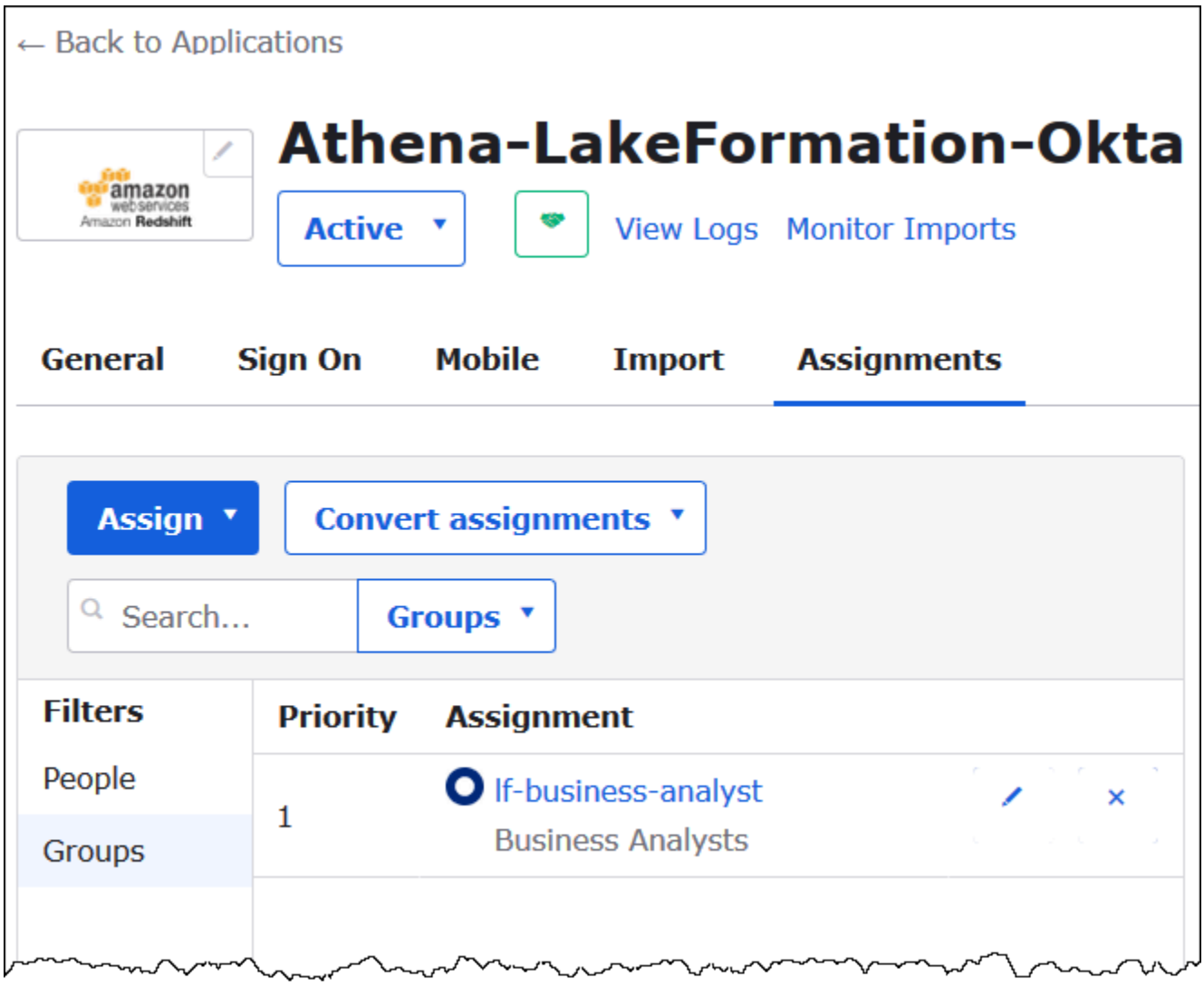
3. Di kotak dialog Tetapkan Athena LakeFormation - -Okta ke Orang, temukan pengguna yang Anda buat athena-okta-usersebelumnya.
4. PilihTetapkanuntuk menetapkan pengguna ke aplikasi.



5. PilihSimpan dan Kembali.
6. PilihSelesai.
7. Pada tab Tugas untuk aplikasi Athena LakeFormation - -Okta, pilih Tetapkan, Tetapkan ke Grup.
8. Untuk lf-business-analyst, pilih Tetapkan untuk menetapkan aplikasi Athena- LakeFormation - Okta ke lf-business-analystgrup, lalu pilih Selesai.



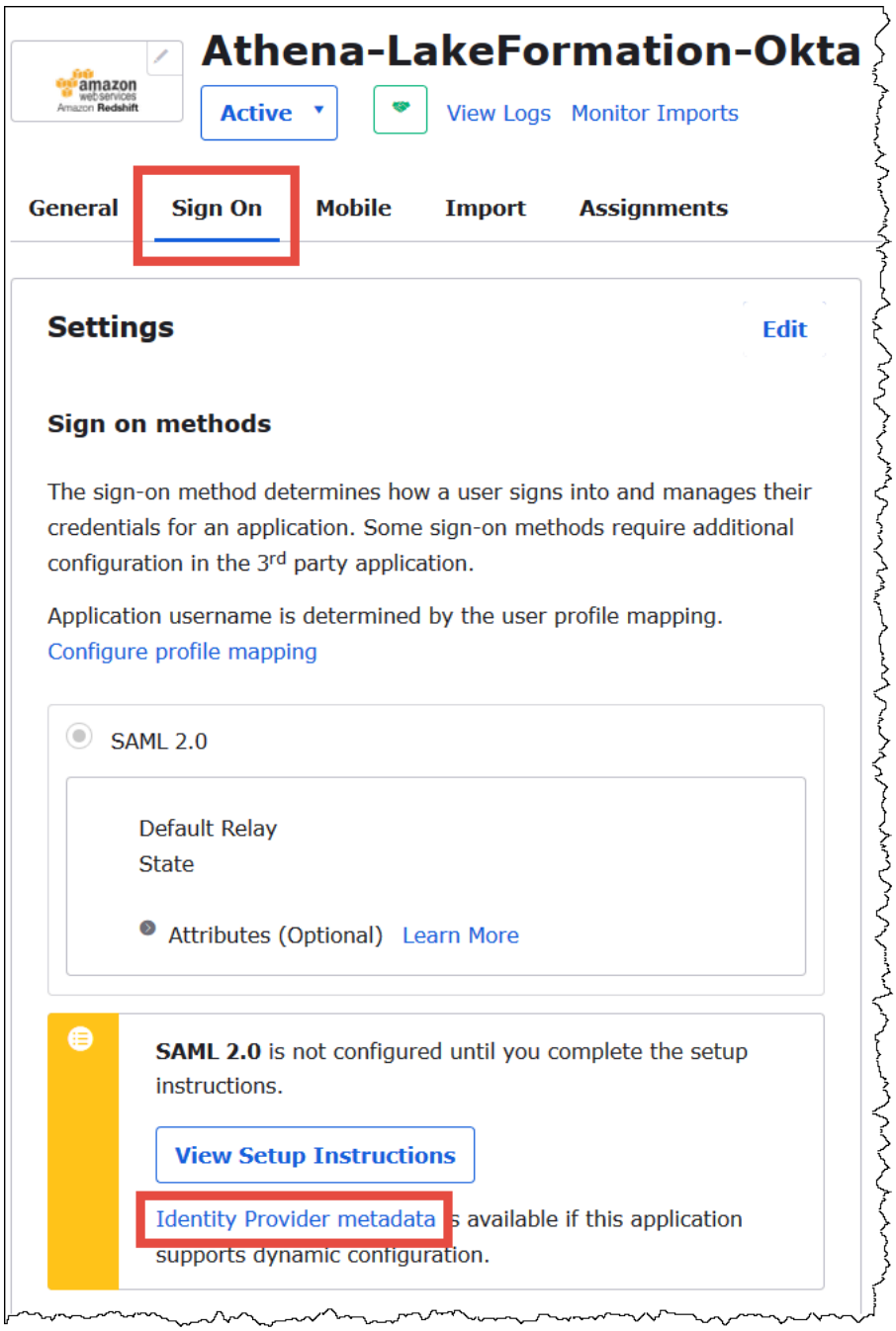
Grup muncul dalam daftar grup untuk aplikasi.



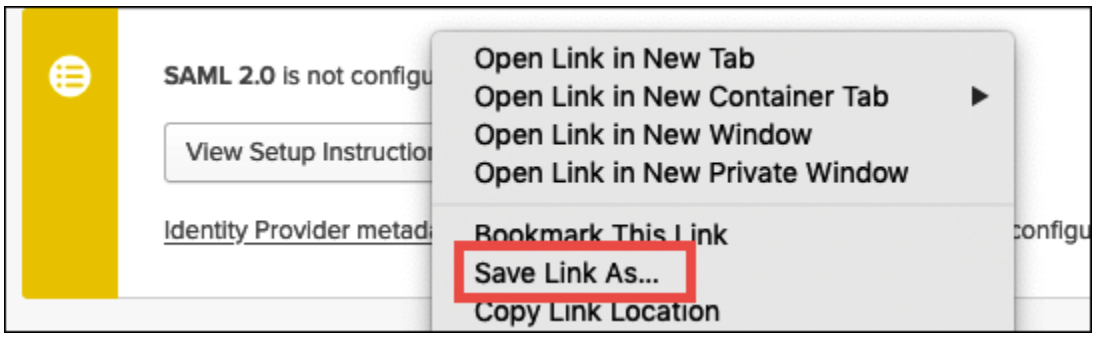
Sekarang Anda siap mengunduh metadata aplikasi penyedia identitas untuk digunakan dengan AWS.

Untuk mengunduh metadata aplikasi

1. Pilih aplikasi OktaTanda Padatab, dan kemudian klik kananMetadata penyedia identitas.



2. PilihSimpan Tautan Sebagaiuntuk menyimpan metadata penyedia identitas, yang dalam format XML, ke file. Berikan nama yang Anda kenali (misalnya,`Athena-LakeFormation-idp-metadata.xml`).



Langkah 4: Buat Penyedia Identitas AWS SALL dan Lake Formation mengakses peran IAM

Pada langkah ini, Anda menggunakan konsol AWS Identity and Access Management (IAM) untuk melakukan tugas-tugas berikut:

- Buat penyedia identitas untuk AWS.
- Buat IAM role untuk akses Lake Formation.
- Tambahkan kebijakan AmazonAthenaFullAccess terkelola ke peran.
- Tambahkan kebijakan untuk Lake Formation dan AWS Glue peran.
- Menambahkan kebijakan untuk Athena hasil kueri untuk peran.

Untuk membuat penyedia identitas AWS SAFL

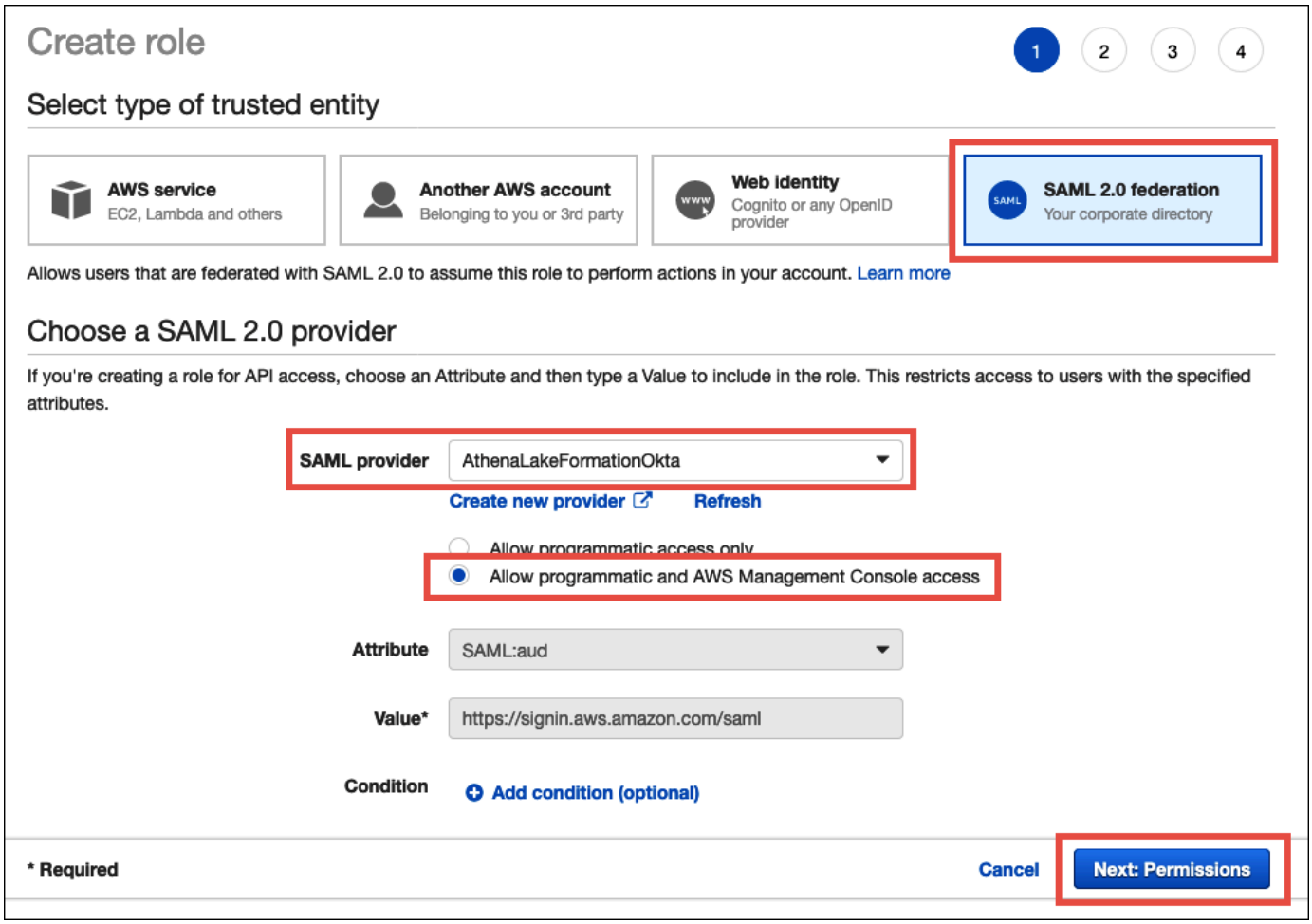
1. Masuklah ke konsol akun Amazon Web Services sebagai administrator akun Amazon Web Services dan arahkan ke konsol IAM (https://console.aws.amazon.com/iam/).
2. Di panel navigasi, pilih Penyedia identitas, lalu klik Tambah penyedia.
3. Pada layar Konfigurasi penyedia, masukkan informasi berikut:
   - Untuk tipe Provider, pilih SAFL.
   - Untuk nama Penyedia, masukkan`AthenaLakeFormationOkta`.
   - Untuk dokumen Metadata, gunakan opsi Pilih file untuk mengunggah file XMLmetadata penyedia identitas (iDP) yang Anda unduh.
4. Pilih Tambah penyedia.

Selanjutnya, Anda membuat peran IAM untuk AWS Lake Formation akses. Anda menambahkan dua inline kebijakan untuk peran. Satu kebijakan memberikan izin untuk mengakses Lake Formation dan

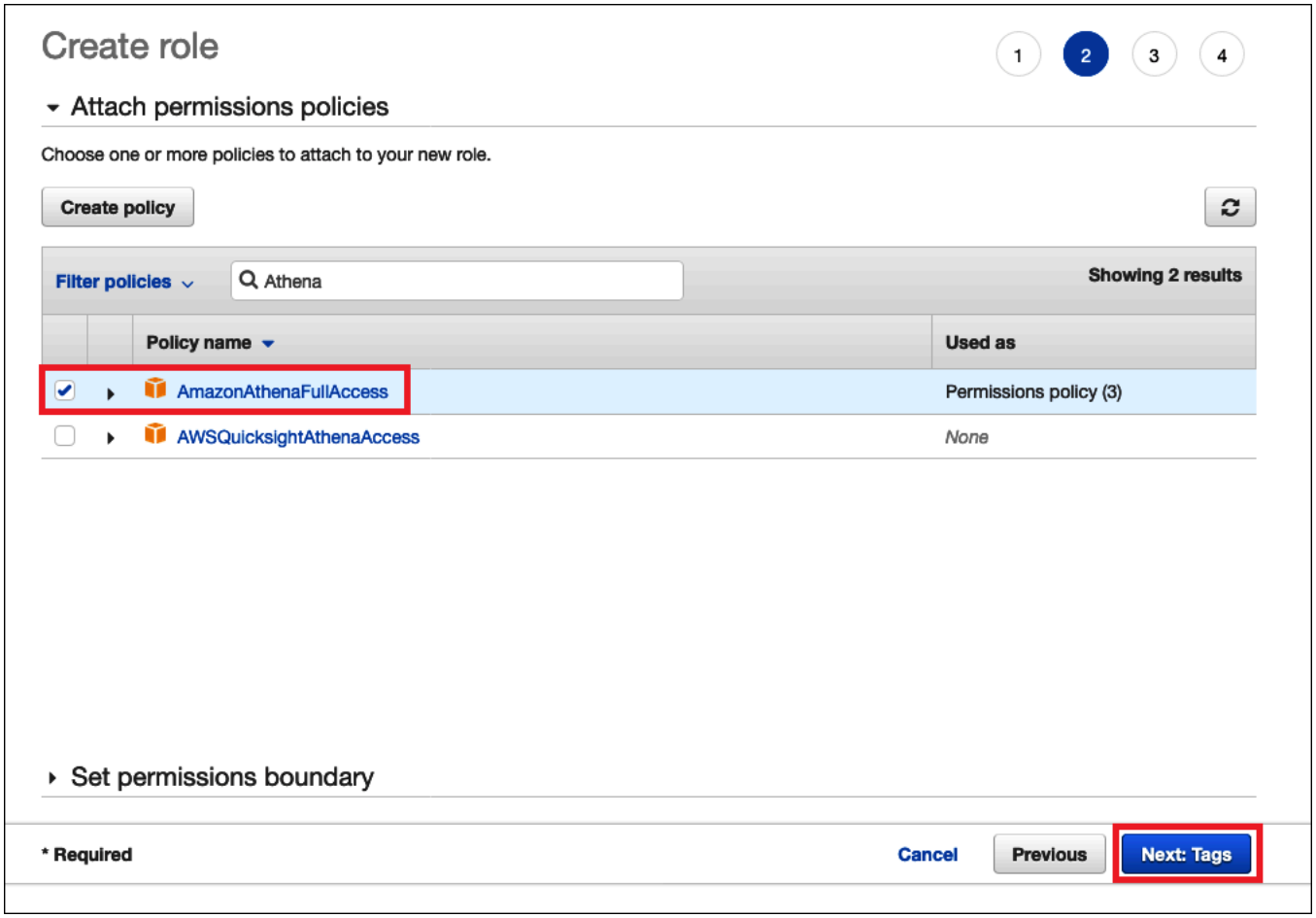
AWS Glue API. Kebijakan lainnya menyediakan akses ke Athena dan lokasi hasil kueri Athena di Amazon S3.

Untuk membuat peran IAM untuk akses AWS Lake Formation

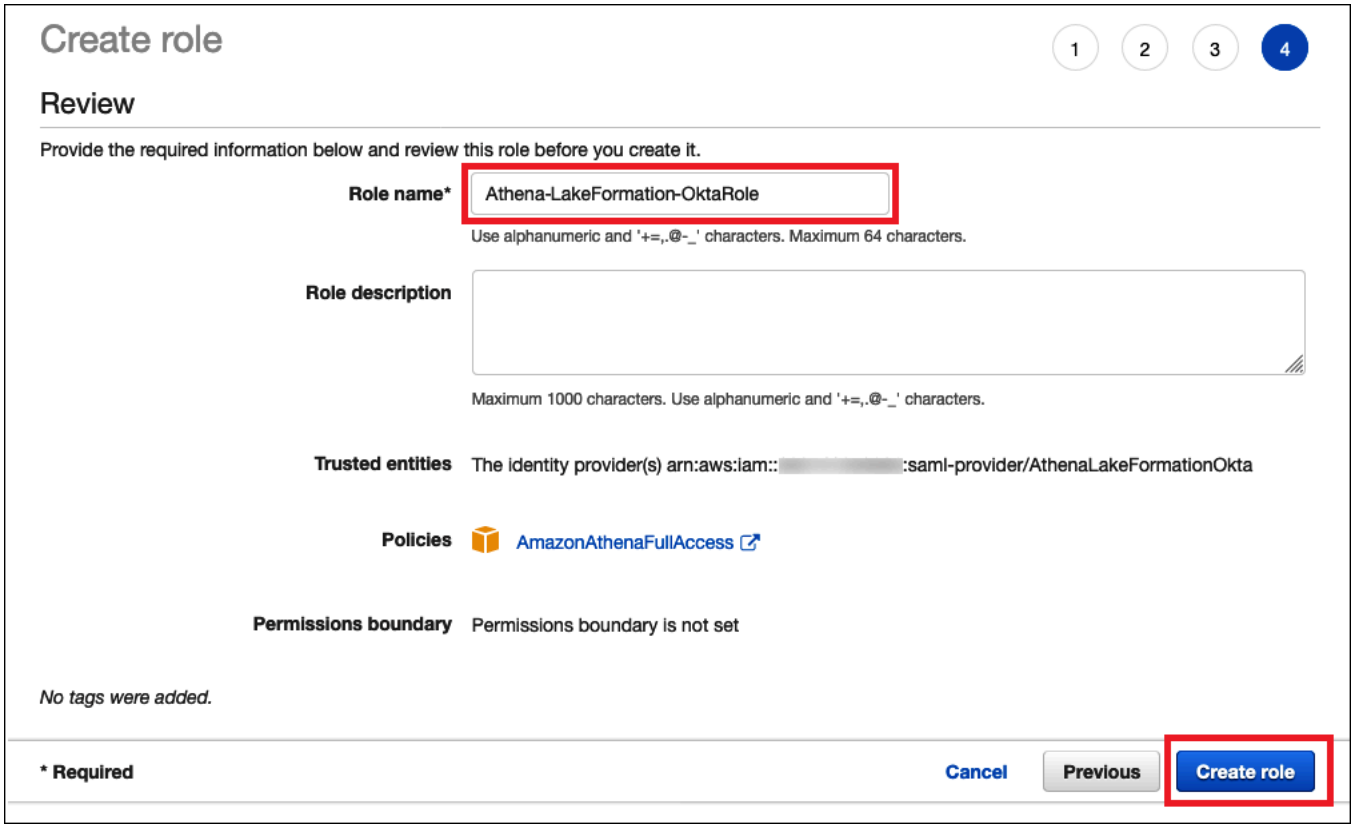
1. Di panel navigasi konsol IAM, pilih Peran, dan lalu pilih Buat peran.
2. Pada halaman Buat peran, lakukan langkah-langkah berikut:



   a. Untuk Pilih jenis entitas terpercaya, pilih Federasi SAML 2.0.
   b. Untuk penyedia SAFL, pilih AthenaLakeFormationOkta.
   c. Untuk penyedia SAFL, pilih opsi Izinkan programatik dan AWS Management Console akses.
   d. Pilih Berikutnya: Izin.
3. PadaMelampirkan kebijakan Izinhalaman, untukKebijakan filter, masukkan**`Athena`**.
4. Pilih kebijakan AmazonAthenaFullAccessterkelola, lalu pilih Berikutnya: Tag.



5. Pada halaman Tambahkan tanda, pilih Berikutnya:
6. Pada halaman Ulasan, untuk nama Peran, masukkan nama untuk peran (misalnya, *`Athena-LakeFormation - OktaRole`*), lalu pilih Buat peran.



Selanjutnya, Anda menambahkan kebijakan sebaris yang memungkinkan akses ke Lake Formation, AWS Glue API, dan hasil kueri Athena di Amazon S3.

Setiap kali Anda menggunakan kebijakan IAM, pastikan bahwa Anda mengikuti praktik terbaik IAM. Untuk informasi selengkapnya tentang administrator, lihat Praktik terbaik keamanan di IAM dalam Panduan Pengguna IAM.

Untuk menambahkan kebijakan inline ke peran Lake Formation dan AWS Glue

1. Dari daftar peran di konsol IAM, pilih opsi baru dibuat`Athena-LakeFormation-OktaRole`.
2. PadaRingkasanhalaman untuk peran, padaIzintab, pilihTambahkan kebijakan inline.
3. Di halaman Buat kebijakan, pilih JSON.
4. Tambahkan kebijakan sebaris seperti berikut yang menyediakan akses ke Lake Formation dan API AWS Glue .

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
```

```
        "Action": [
            "lakeformation:GetDataAccess",
            "glue:GetTable",
            "glue:GetTables",
            "glue:GetDatabase",
            "glue:GetDatabases",
            "glue:CreateDatabase",
            "glue:GetUserDefinedFunction",
            "glue:GetUserDefinedFunctions"
        ],
        "Resource": "*"
    }
}
```

5. Pilih Tinjau kebijakan.
6. Untuk Nama, masukkan nama untuk kebijakan (misalnya, **`LakeFormationGlueInlinePolicy`**).
7. Pilih Buat kebijakan.

Untuk menambahkan kebijakan inline untuk peran untuk lokasi hasil permintaan Athena

1. PadaRingkasanhalaman untuk`Athena-LakeFormation-OktaRole`peran, padaIzintab, pilihTambahkan kebijakan inline.
2. Di halaman Buat kebijakan, pilih tab JSON.
3. Tambahkan kebijakan inline seperti berikut yang memungkinkan akses peran ke lokasi hasil permintaan Athena. Ganti placeholder *`< athena-query-results-bucket >`* pada contoh dengan nama bucket Amazon S3 Anda.

```
{
    "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. Pilih Tinjau kebijakan.
5. Untuk Nama, masukkan nama untuk kebijakan (misalnya, **`AthenaQueryResultsInlinePolicy`**).
6. Pilih Buat kebijakan.

Selanjutnya, Anda menyalin ARN peran akses Lake Formation dan ARN penyedia SAML yang Anda buat. Ini diperlukan saat Anda mengonfigurasi aplikasi Okta SAML di bagian berikutnya dari tutorial.

Untuk menyalin peran ARN dan SAML identitas penyedia ARN

1. Di konsol IAM, padaRingkasanhalaman untuk`Athena-LakeFormation-OktaRole`peran, pilihSalin ke clipboardikon di sampingARN ARN. ARN memiliki format berikut:

   ```
   arn:aws:iam::<account-id>:role/Athena-LakeFormation-OktaRole
   ```

2. Simpan ARN penuh aman untuk referensi nanti.
3. Di panel navigasi konsol IAM, pilihPenyedia identitas.
4. Pilih AthenaLakeFormationOktapenyedia.
5. PadaRingkasan, pilihSalin ke clipboardikon di sampingPenyedia ARN. Outputnya akan terlihat seperti berikut:

   ```
   arn:aws:iam::<account-id>:saml-provider/AthenaLakeFormationOkta
   ```

6. Simpan ARN penuh aman untuk referensi nanti.

Langkah 5: Tambahkan peran IAM dan Penyedia Identitas SALL ke aplikasi Okta

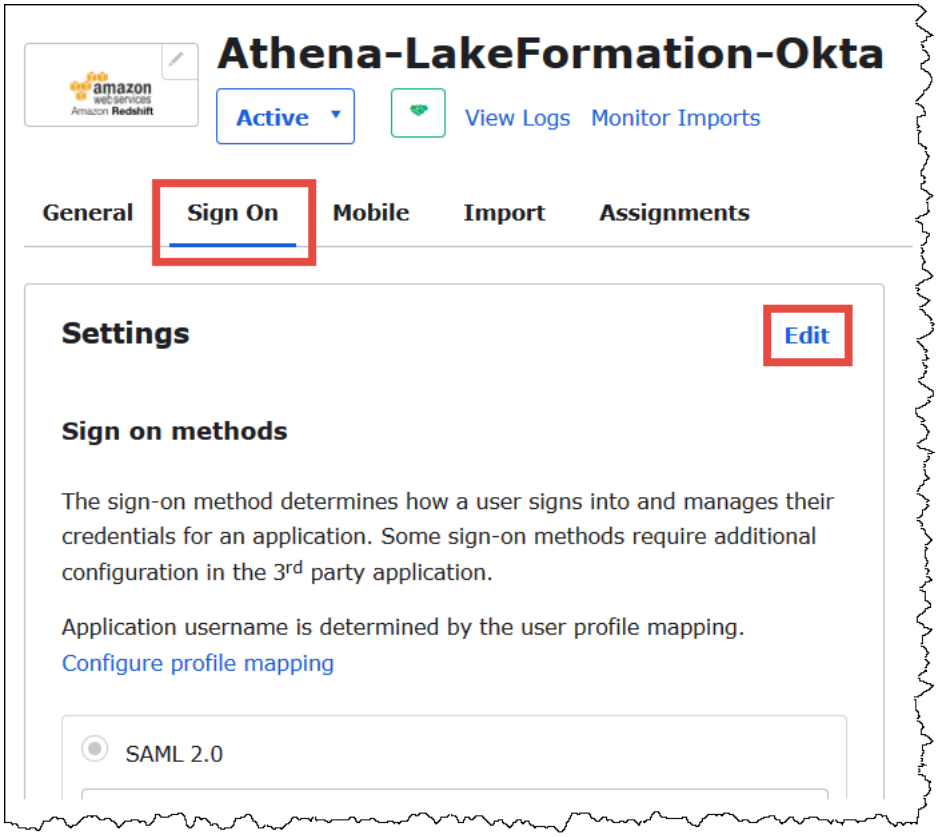
Pada langkah ini, Anda kembali ke konsol developer Okta dan melakukan tugas berikut:

- Tambahkan pengguna dan grup atribut URL Lake Formation ke aplikasi Okta.
- Menambahkan ARN untuk penyedia identitas dan ARN untuk IAM role untuk aplikasi Okta.

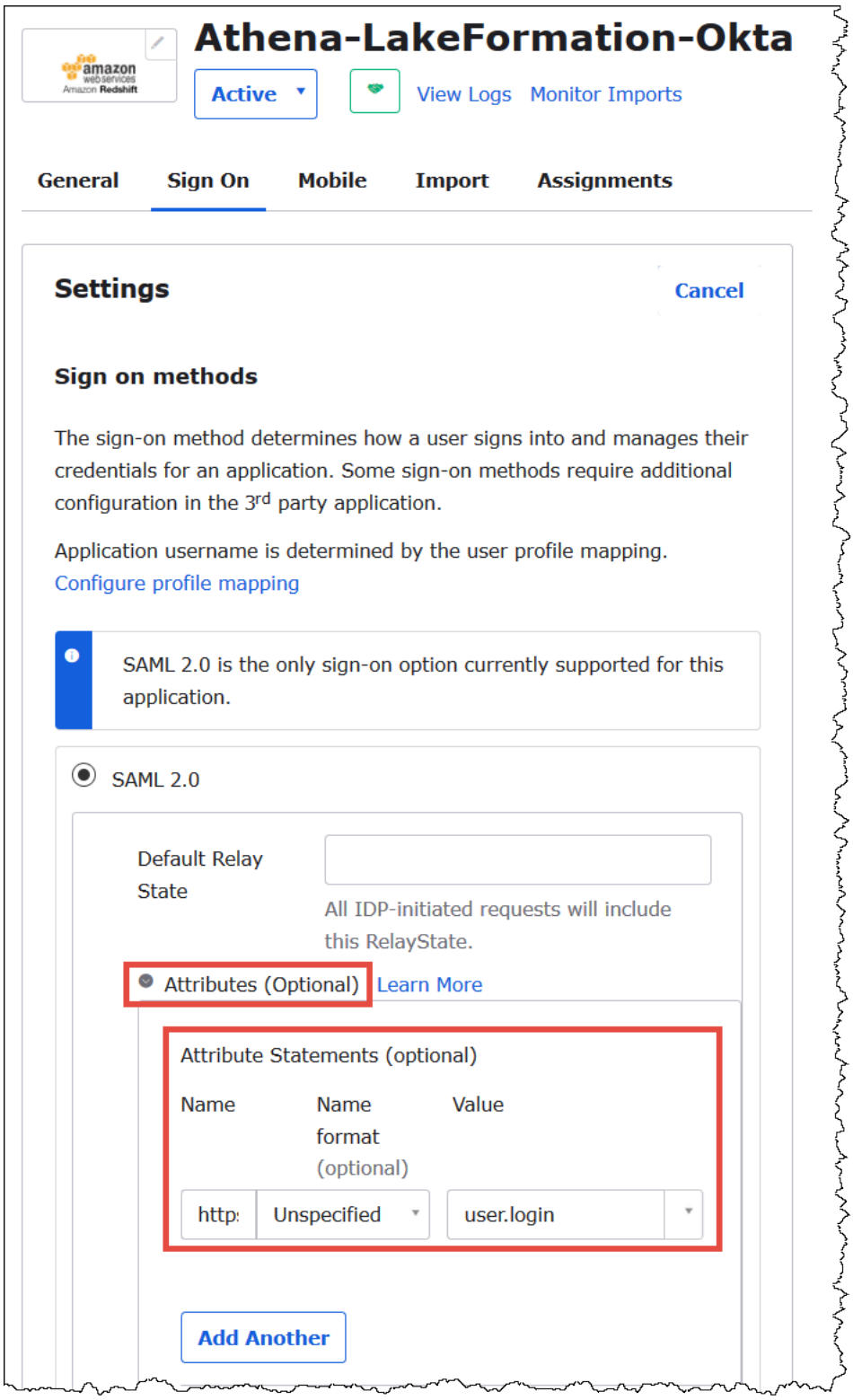
- Salin ID aplikasi Okta. ID aplikasi Okta diperlukan dalam profil JDBC yang terhubung ke Athena.

Untuk menambahkan pengguna dan grup atribut URL Lake Formation ke aplikasi Okta

1. Masuk ke konsol developer Okta.
2. PilihAplikasitab, dan kemudian pilih`Athena-LakeFormation-Okta`aplikasi.
3. Pilih padaTanda Padatab untuk aplikasi, dan kemudian pilihMengedit.



4. PilihAtribut (opsional)untuk memperluasnya.



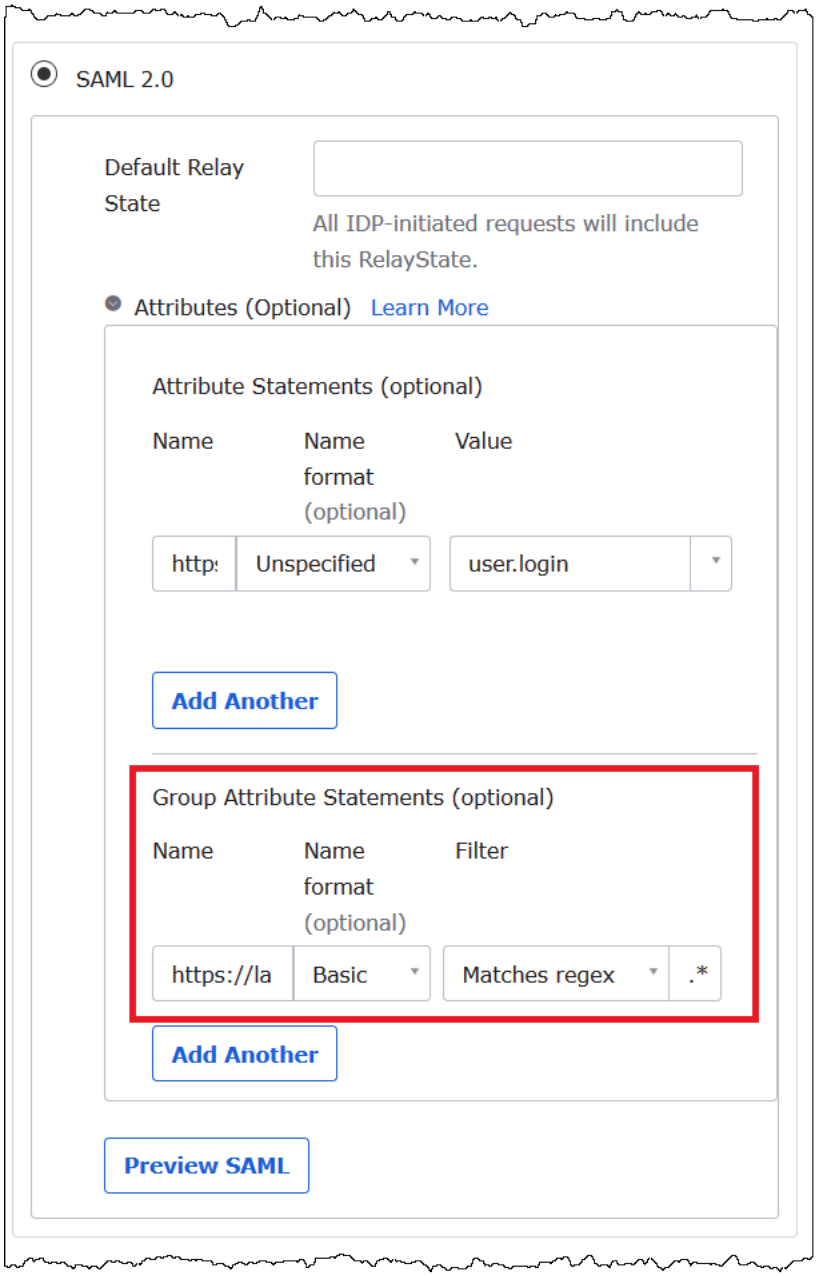
5. UntukPernyataan atribut (opsional), tambahkan atribut berikut:

   - Untuk Nama, masukkan **`https://lakeformation.amazon.com/SAML/Attributes/Username`**.

- Untuk Nilai, masukkan **`user.login`**.

6. Di bawahKelompok Pernyataan Atribut (opsional), tambahkan atribut berikut:

   - Untuk Nama, masukkan **`https://lakeformation.amazon.com/SAML/Attributes/Groups`**.
   - UntukFormat nama, masukkan**`Basic`**
   - UntukFilterPilihCocok regex, dan kemudian masukkan**`.*`**di kotak filter.

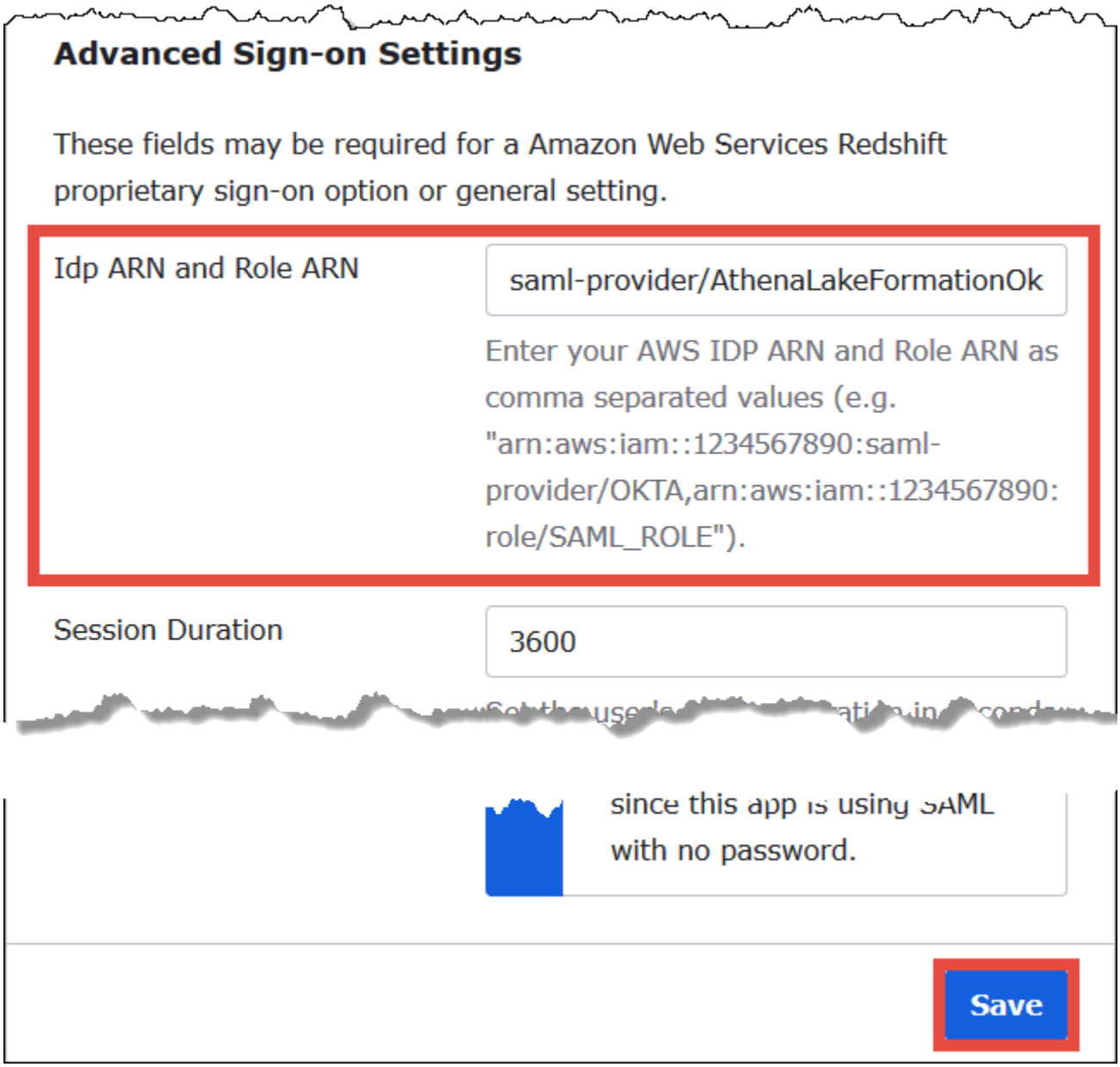


7. Gulir ke bawah kePengaturan Masuk Lanjutanbagian, tempat Anda akan menambahkan penyedia identitas dan IAM Peran ARN untuk aplikasi Okta.

Untuk menambahkan ARN untuk penyedia identitas dan IAM role untuk aplikasi Okta

1. <saml-arn><role-arn>Untuk Idp ARN dan ARN Peran, masukkan penyedia AWS identitas ARN dan peran ARN sebagai nilai yang dipisahkan koma dalam format,. String gabungan akan terlihat seperti berikut:

```
arn:aws:iam::<account-id>:saml-provider/
AthenaLakeFormationOkta,arn:aws:iam::<account-id>:role/Athena-LakeFormation-
OktaRole
```

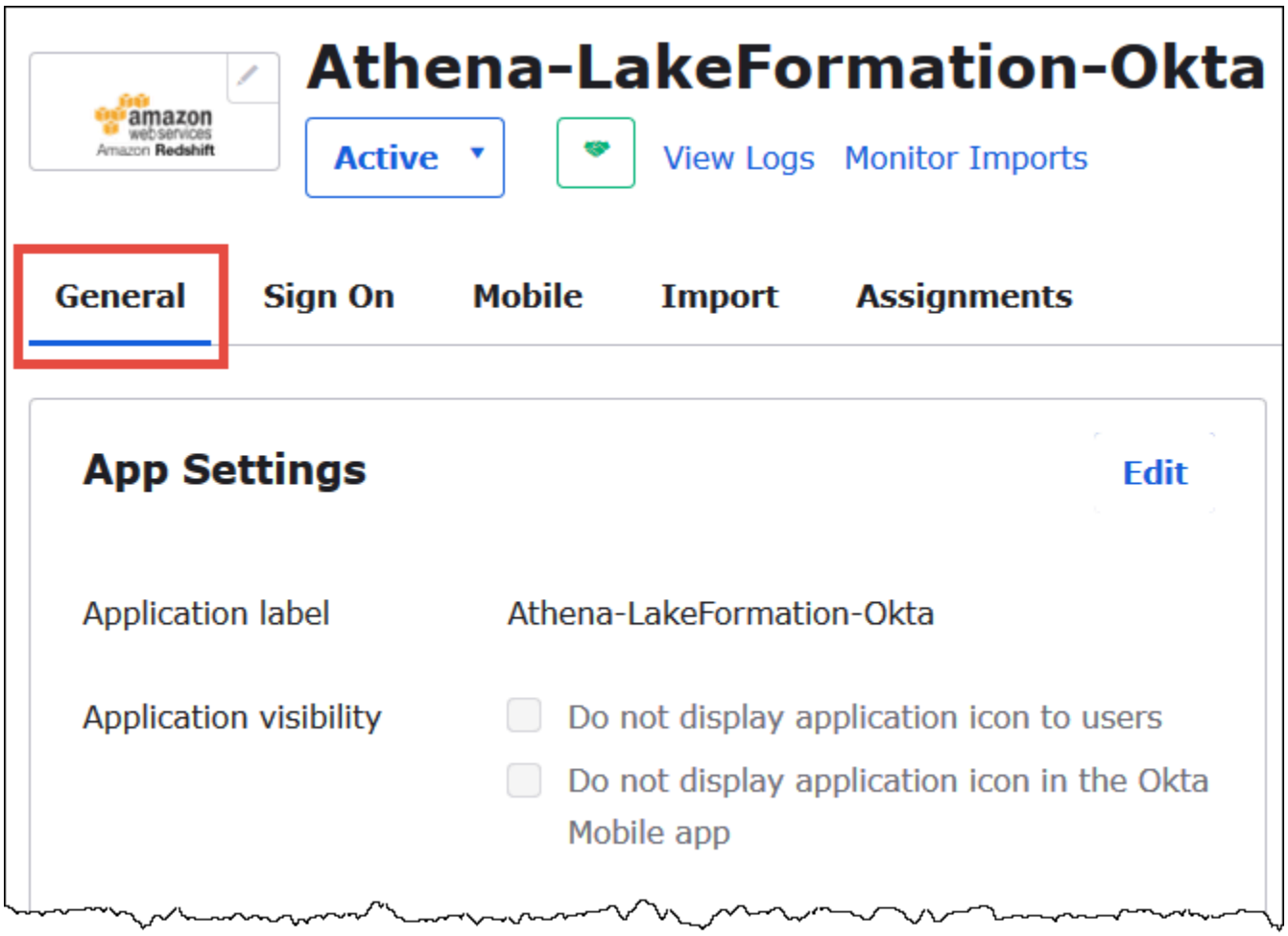


2. Pilih Simpan.

Selanjutnya, Anda menyalin ID aplikasi Okta. Anda akan membutuhkan ini nanti untuk string JDBC yang menghubungkan ke Athena.

Untuk mencari dan menyalin ID aplikasi Okta

1. PilihUmumtab aplikasi Okta.



2. Gulir ke bawah keTautan Sematkan AplikasiBagian.
3. DariTautan Sematkan, salin dan simpan dengan aman bagian ID aplikasi Okta dari URL. ID aplikasi Okta adalah bagian dari URL setelah`amazon_aws_redshift/`tapi sebelum garis miring berikutnya. Misalnya, jika URL berisi`amazon_aws_redshift/aaa/bbb`, ID aplikasi adalah`aaa`.

> **Note**
>
> Tautan sematan tidak dapat digunakan untuk masuk langsung ke konsol Athena untuk melihat database. Izin Lake Formation untuk pengguna dan grup SALL hanya dikenali jika Anda menggunakan driver JDBC atau ODBC untuk mengirimkan kueri ke Athena. Untuk melihat database, Anda dapat menggunakan alat SQL Workbench/J, yang menggunakan driver JDBC untuk terhubung ke Athena. Alat SQL Workbench/J tercakup dalam. Langkah 7: Verifikasi akses melalui klien Athena JDBC

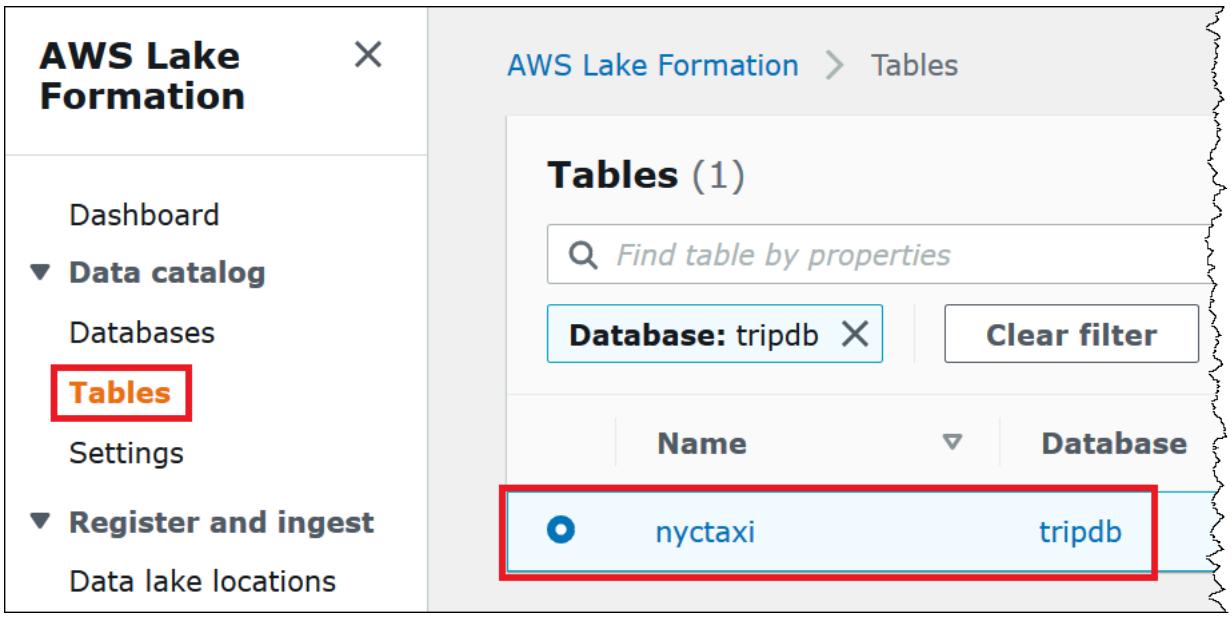
Langkah 6: Berikan izin pengguna dan grup melalui AWS Lake Formation

Pada langkah ini, Anda menggunakan konsol Lake Formation untuk memberikan izin pada tabel untuk pengguna dan grup SAML. Anda harus melakukan langkah-langkah berikut:

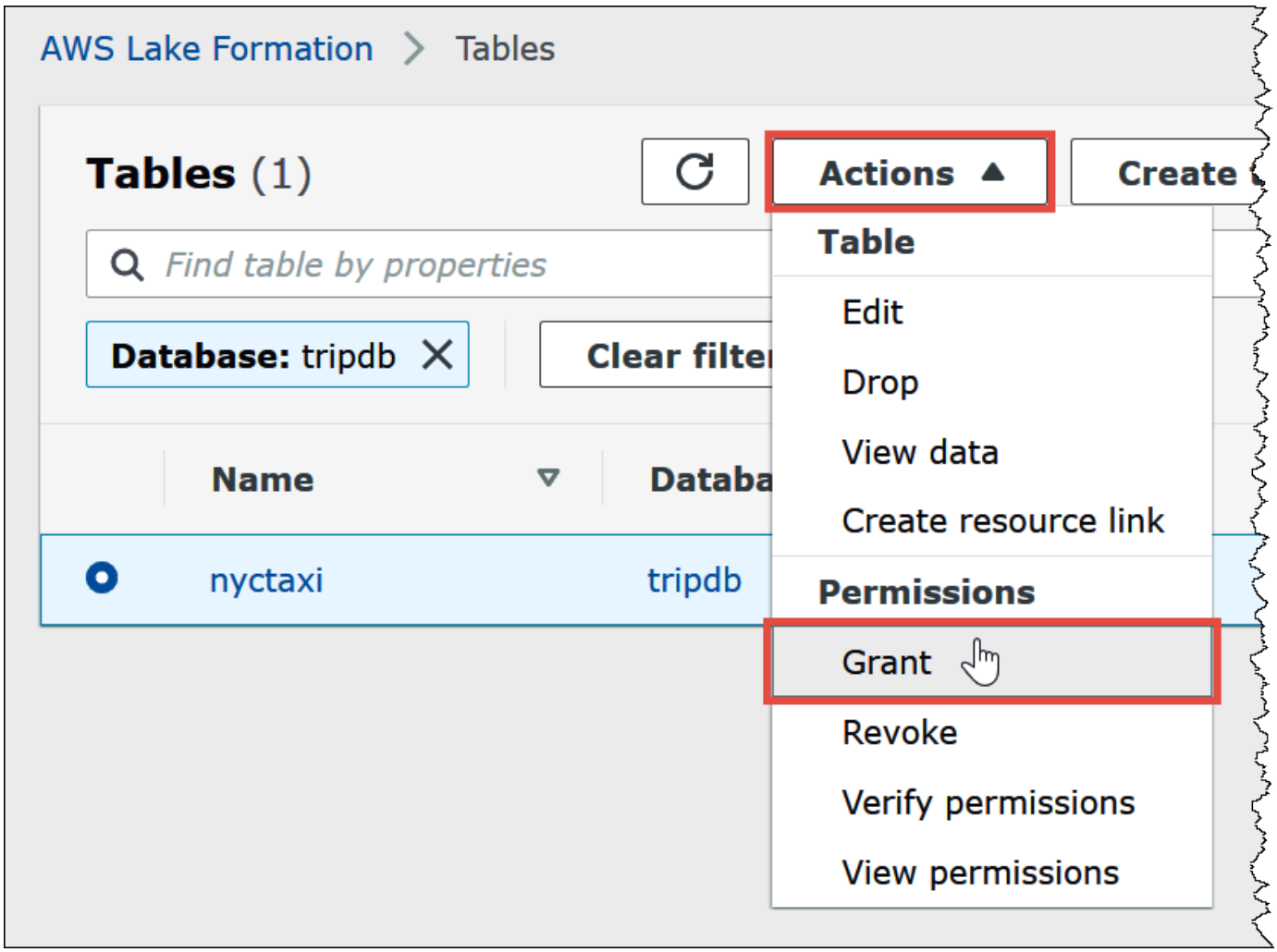
- Tentukan ARN dari pengguna Okta SAML dan izin pengguna terkait di atas tabel.
- Tentukan ARN dari grup Okta SAML dan izin grup terkait di atas tabel.
- Verifikasi izin yang Anda berikan.

Untuk memberikan izin di Lake Formation untuk pengguna Okta

1. Masuk sebagai administrator danau data ke AWS Management Console.
2. Buka konsol Lake Formation dihttps://console.aws.amazon.com/lakeformation/.
3. Dari panel navigasi, pilihTabel, kemudian pilih tabel yang ingin Anda berikan izin untuk. Tutorial ini menggunakan`nyctaxi`tabel dari`tripdb`basis data.



4. DariTindakanPilihIzin.



5. DiBerikan izin, masukkan informasi berikut:

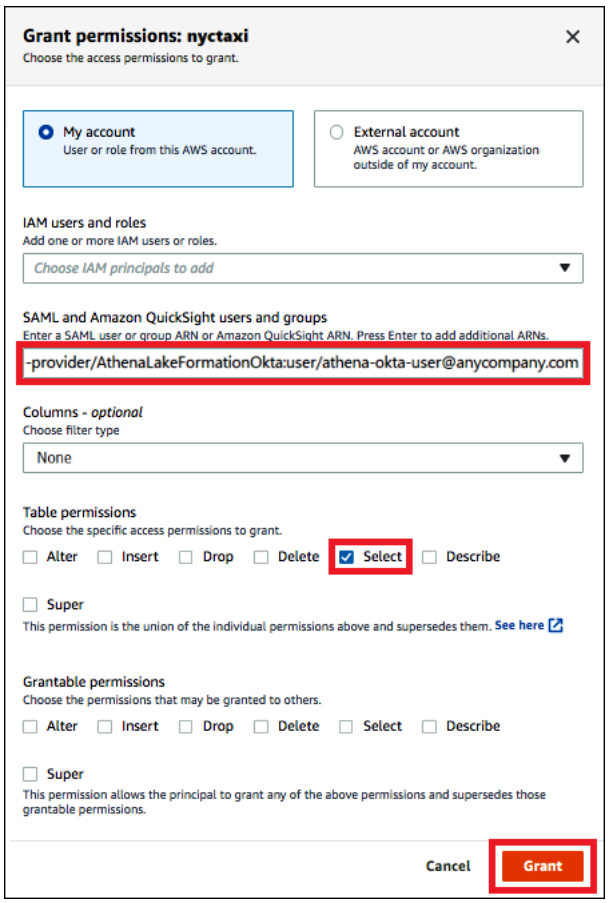
   a. Di bawah QuickSight pengguna dan grup SAFL dan Amazon, masukkan ARN pengguna OKTA SALL dalam format berikut:

   ```
   arn:aws:iam::<account-id>:saml-provider/AthenaLakeFormationOkta:user/<athena-okta-user>@<anycompany.com>
   ```

   b. UntukKolom, untukPilih jenis filter, dan opsional memilihSertakan kolomatauMengecualikan kolom.

   c. GunakanPilih satu kolom atau lebihdi bawah filter untuk menentukan kolom yang ingin Anda sertakan atau kecualikan untuk atau dari pengguna.

   d. UntukIzin tabelPilihPilih. Tutorial ini hanya memberikan`SELECT`; kebutuhan Anda mungkin berbeda.



6. PilihIzin.

Sekarang Anda melakukan langkah serupa untuk grup Okta.

Untuk memberikan izin di Lake Formation untuk grup Okta

1. PadaTabelkonsol Lake Formation, pastikan bahwanyctaxitabel masih dipilih.
2. DariTindakanPilihIzin.
3. DiBerikan izin, masukkan informasi berikut:

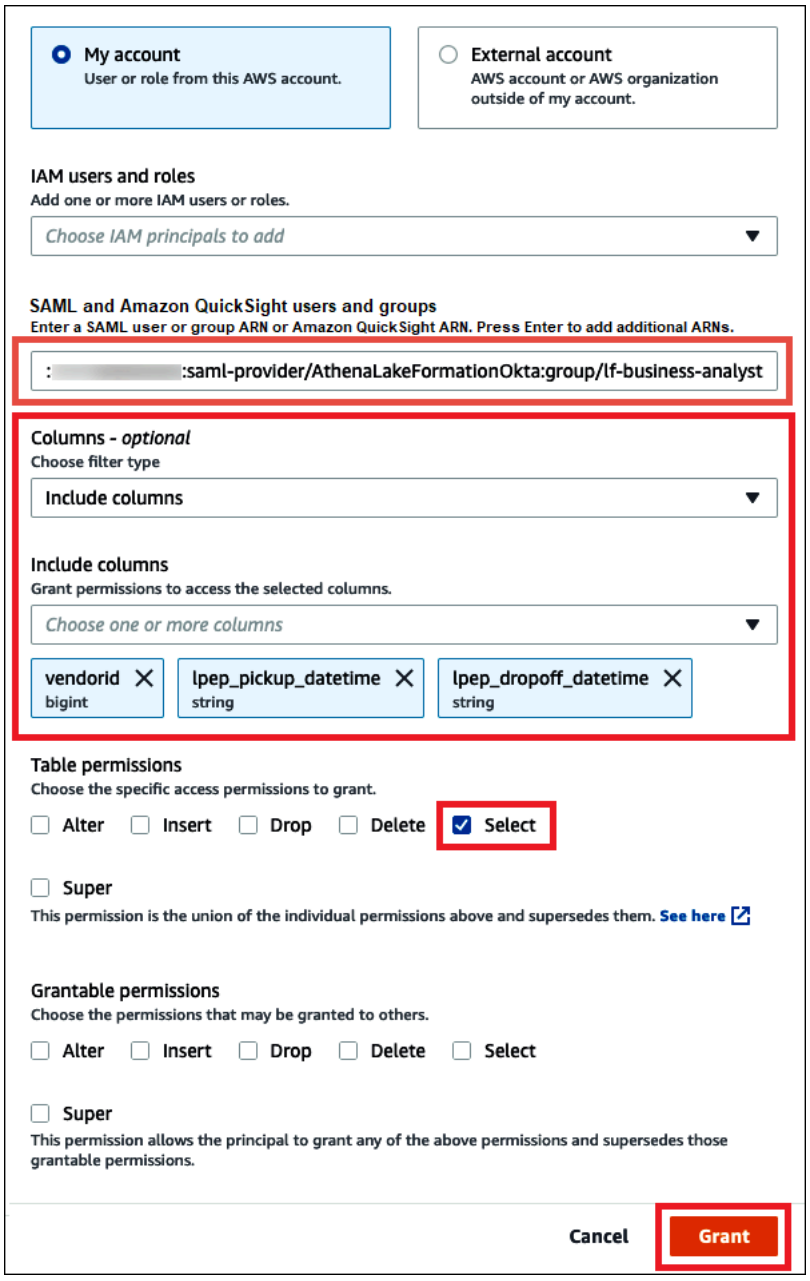
   a. Di bawah QuickSight pengguna dan grup SAFL dan Amazon, masukkan ARN grup OKTA SALL dalam format berikut:

   ```
   arn:aws:iam::<account-id>:saml-provider/AthenaLakeFormationOkta:group/lf-business-analyst
   ```

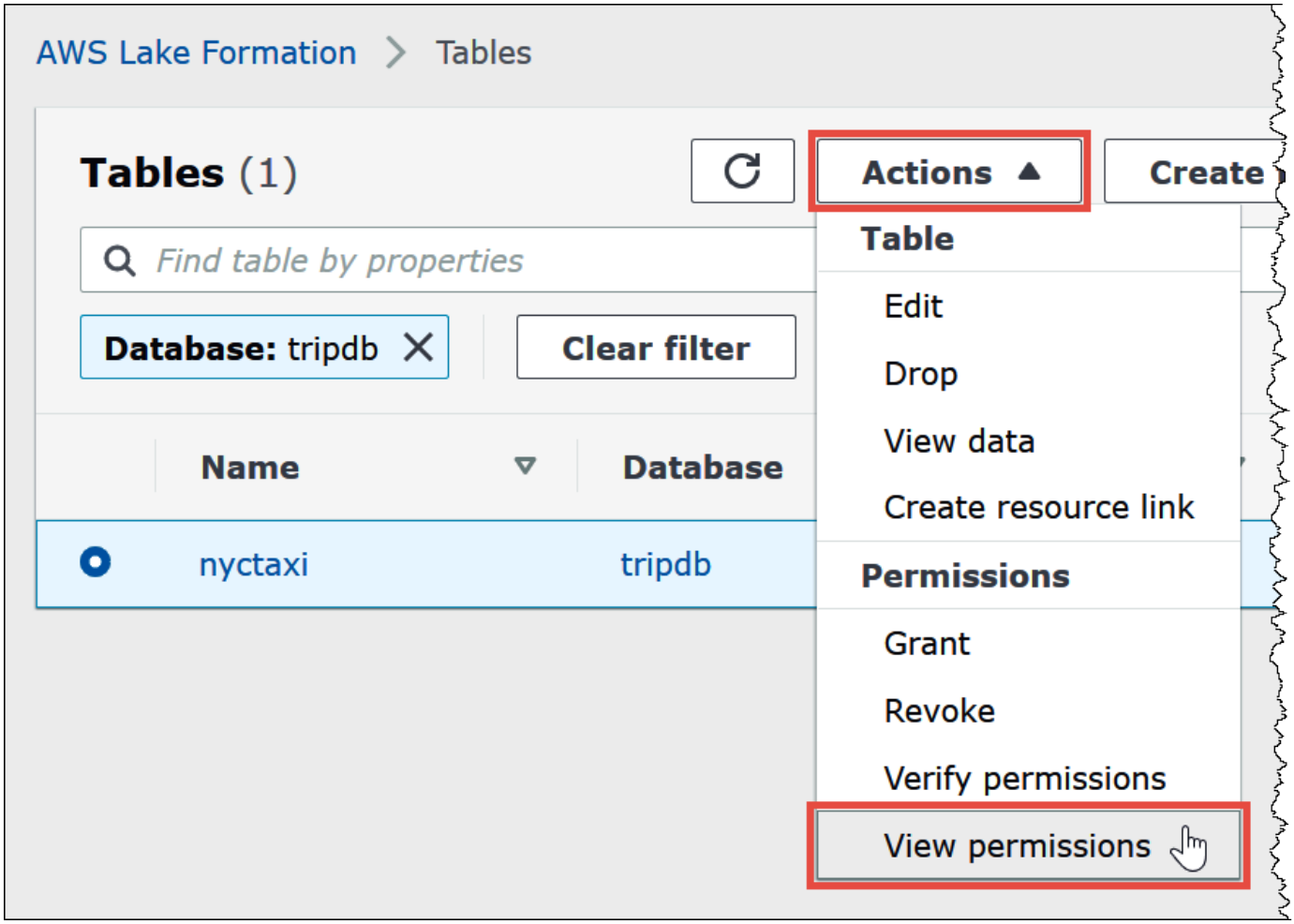
   b. UntukKolom,Pilih jenis filterPilihSertakan kolom.

   c. UntukPilih satu kolom atau lebih, pilih tiga kolom pertama dari tabel.

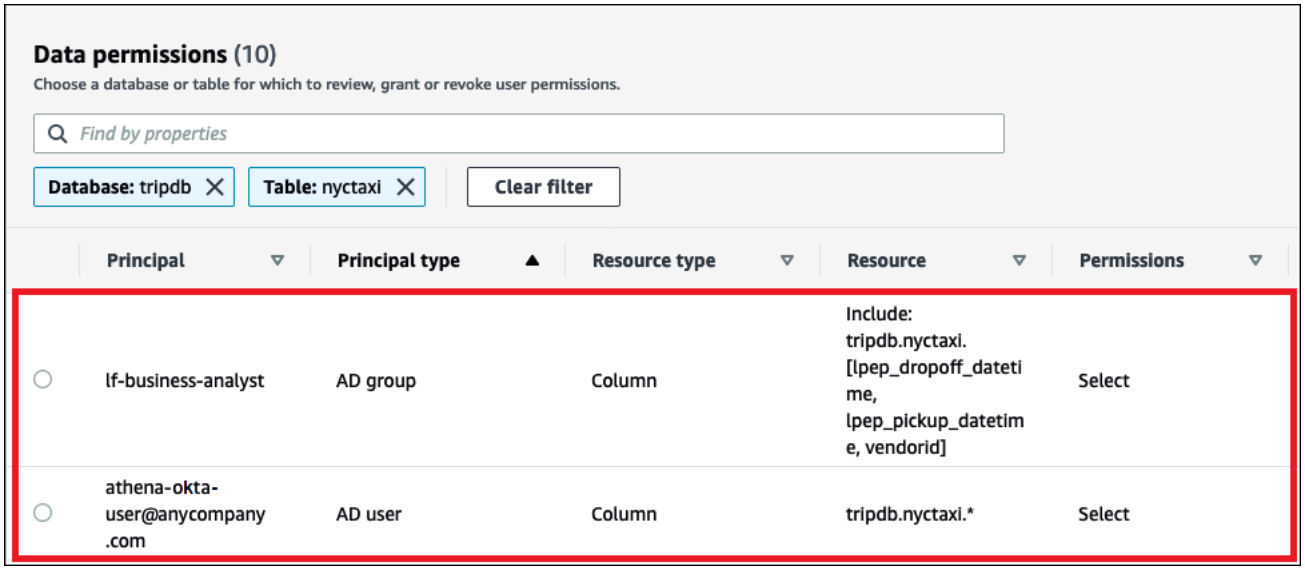
d. UntukIzin tabel, pilih izin akses khusus untuk diberikan. Tutorial ini hanya memberikanSELECT; kebutuhan Anda mungkin berbeda.



4. PilihIzin.
5. Untuk memverifikasi izin yang Anda berikan, pilihTindakan,Lihat izin.



Halaman izin data untuk `nyctaxi` tabel menunjukkan izin untuk athena-okta-userdan grup. lf-business-analyst



Langkah 7: Verifikasi akses melalui klien Athena JDBC

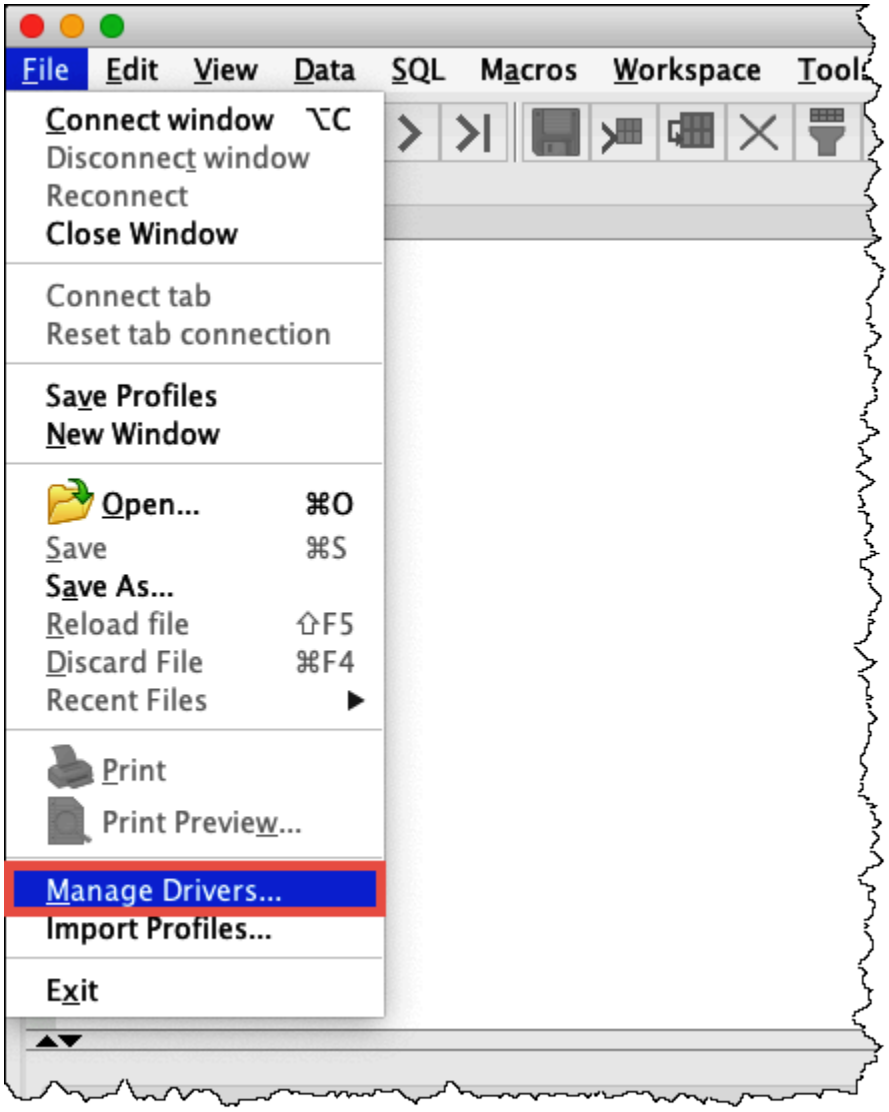
Sekarang Anda siap untuk menggunakan klien JDBC untuk melakukan koneksi tes ke Athena sebagai pengguna Okta SAML.

Di bagian ini, Anda harus melakukan tugas berikut:

- Mempersiapkan klien uji — Download driver Athena JDBC, menginstal SQL Workbench, dan menambahkan driver untuk Workbench. Tutorial ini menggunakan SQL Workbench untuk mengakses Athena melalui otentikasi Okta dan untuk memverifikasi izin Lake Formation.
- NoSQL Workbench
  - Buat koneksi untuk pengguna Athena Okta.
  - Jalankan kueri tes sebagai pengguna Athena Okta.
  - Membuat dan menguji koneksi untuk pengguna analis bisnis.
- Di konsol Okta, tambahkan pengguna analis bisnis ke grup developer.
- Di konsol Lake Formation, konfigurasikan izin tabel untuk grup developer.
- Dalam SQL Workbench, jalankan kueri tes sebagai pengguna analis bisnis dan memverifikasi bagaimana perubahan izin mempengaruhi hasil.

Untuk menyiapkan klien uji

1. Unduh dan ekstrak driver Athena JDBC yang kompatibel dengan Lake Formation (2.0.14 atau versi yang lebih baru) dariMenghubungkan ke Amazon Athena dengan JDBC.
2. Unduh dan instalSQL Workbench /JAlat kueri SQL, tersedia di bawah lisensi Apache 2.0 yang dimodifikasi.
3. Dalam SQL Workbench, pilihBerkas, lalu pilihMengelola driver.



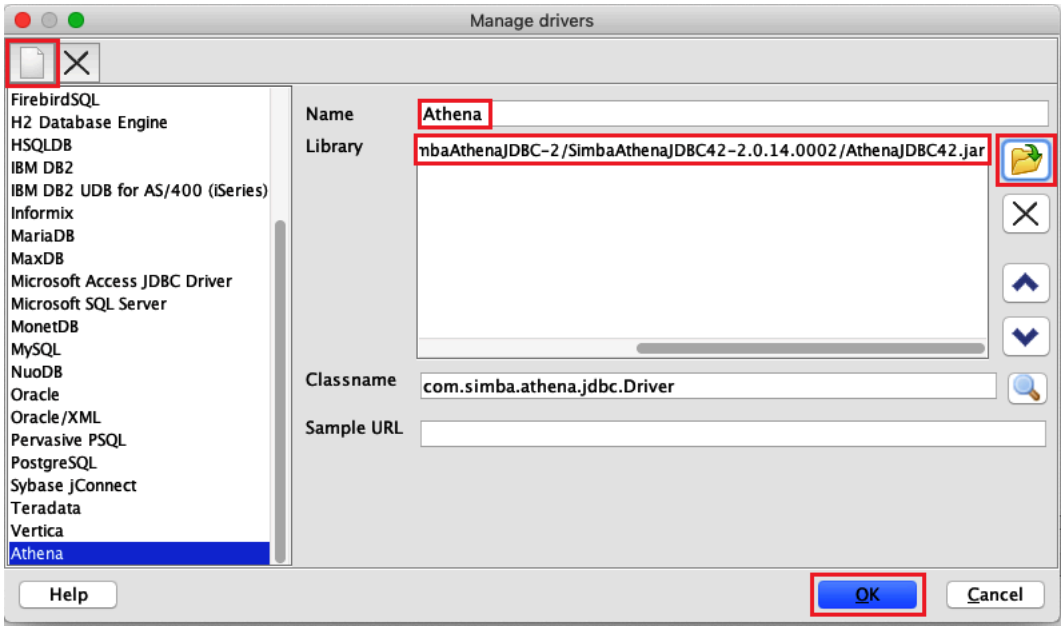
4. DiMengelola driver, lakukan langkah-langkah berikut:

   a. Pilih ikon pemandu baru.

   b. Untuk Nama, masukkan **`Athena`**.

   c. UntukPerpustakaan, browse ke dan pilih Simba Athena JDBC`.jar`yang baru saja diunduh.

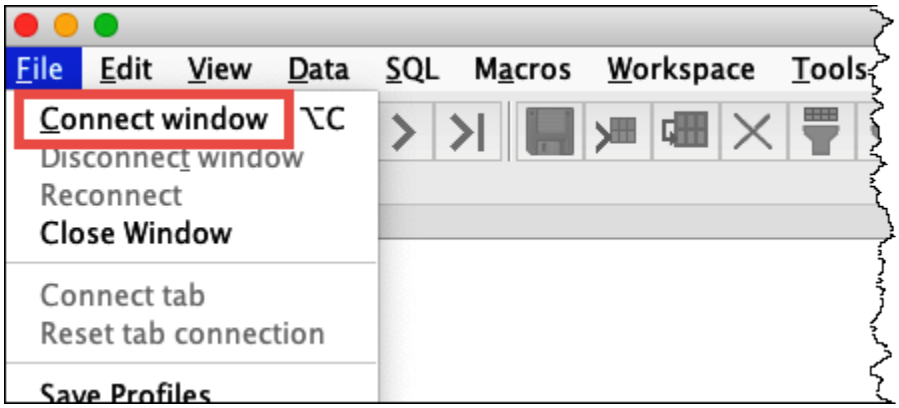
   d. Pilih OK.



Anda sekarang siap untuk membuat dan menguji koneksi untuk pengguna Athena Okta.

Untuk membuat koneksi untuk pengguna Okta

1. PilihBerkas,Connect jendela.



2. DiProfil koneksikotak dialog, membuat koneksi dengan memasukkan informasi berikut:

   - Di Nama , masukkan**`Athena_Okta_User_Connection`**
   - UntukDriver, pilih Sopir Simba Athena JDBC.
   - Untuk Source, lakukan salah satu hal berikut:
     - Untuk menggunakan URL koneksi, masukkan string koneksi satu baris. Contoh berikut menambahkan jeda baris untuk dibaca.

```
jdbc:awsathena://AwsRegion=region-id;
S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/athena_results;
AwsCredentialsProviderClass=com.simba.athena.iamsupport.plugin.OktaCredentialsProvider;
user=athena-okta-user@anycompany.com;
password=password;
idp_host=okta-idp-domain;
App_ID=okta-app-id;
SSL_Insecure=true;
LakeFormationEnabled=true;
```

- Untuk menggunakan URL AWS berbasis profil, lakukan langkah-langkah berikut:

  1. Konfigurasikan AWS profil yang memiliki file AWS kredensial seperti contoh berikut.

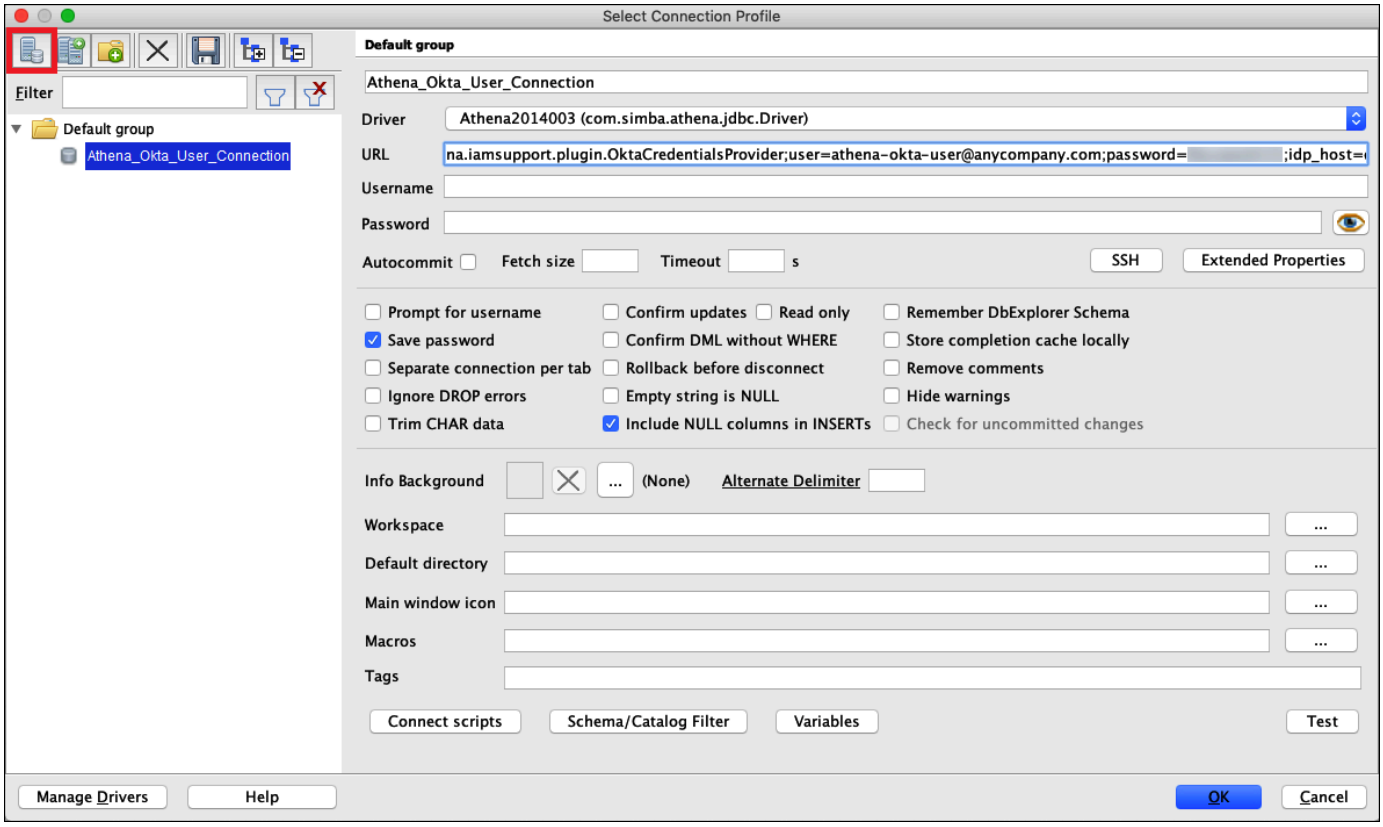
     ```
     [athena_lf_dev]
     plugin_name=com.simba.athena.iamsupport.plugin.OktaCredentialsProvider
     idp_host=okta-idp-domain
     app_id=okta-app-id
     uid=athena-okta-user@anycompany.com
     pwd=password
     ```

  2. UntukURL, masukkan string koneksi single-line seperti contoh berikut. Contoh menambahkan jeda baris untuk dibaca.

     ```
     jdbc:awsathena://AwsRegion=region-id;
     S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/athena_results;
     profile=athena_lf_dev;
     SSL_Insecure=true;
     LakeFormationEnabled=true;
     ```

  Perhatikan bahwa contoh-contoh ini adalah representasi dasar dari URL yang diperlukan untuk terhubung ke Athena. Untuk daftar lengkap parameter yang didukung di URL, lihat dokumentasi JDBC.

Gambar berikut menunjukkan profil koneksi SQL Workbench yang menggunakan URL koneksi.

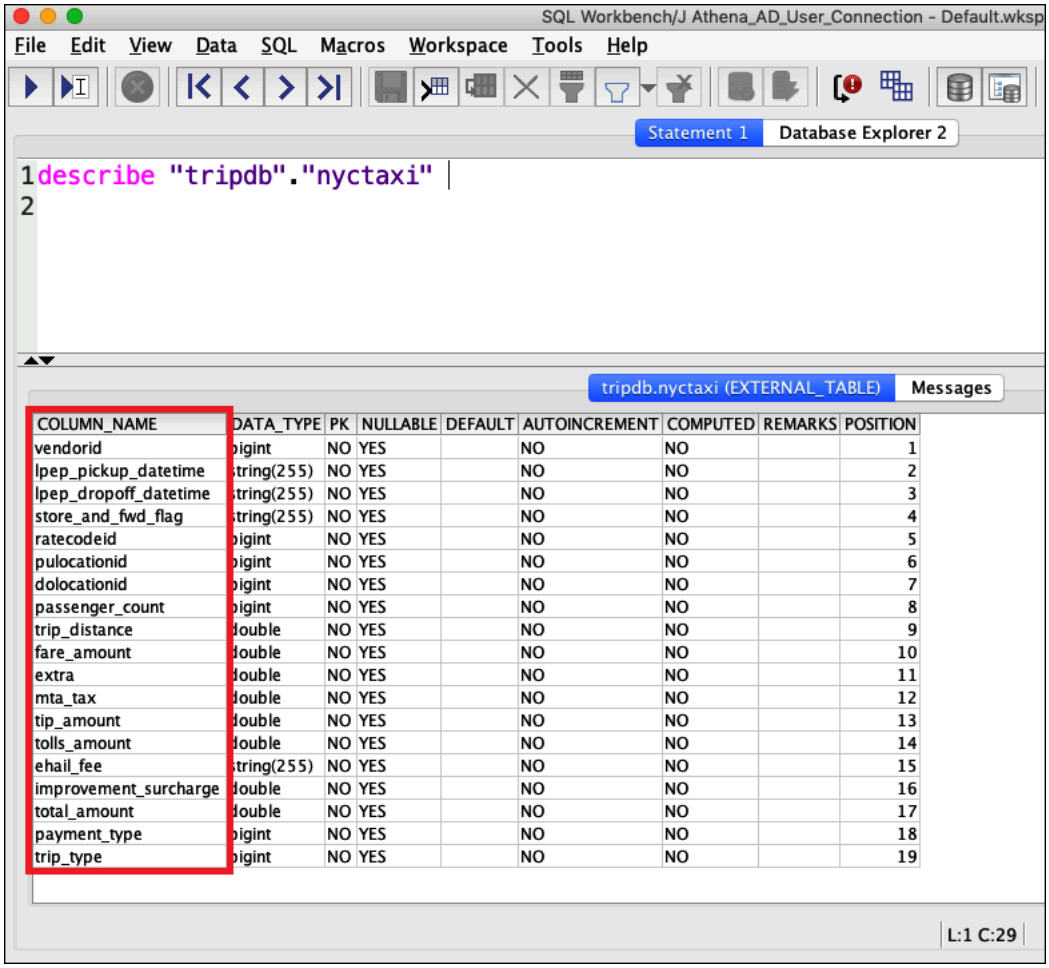


Sekarang bahwa Anda telah membuat koneksi untuk pengguna Okta, Anda dapat mengujinya dengan mengambil beberapa data.

Untuk menguji koneksi untuk pengguna Okta

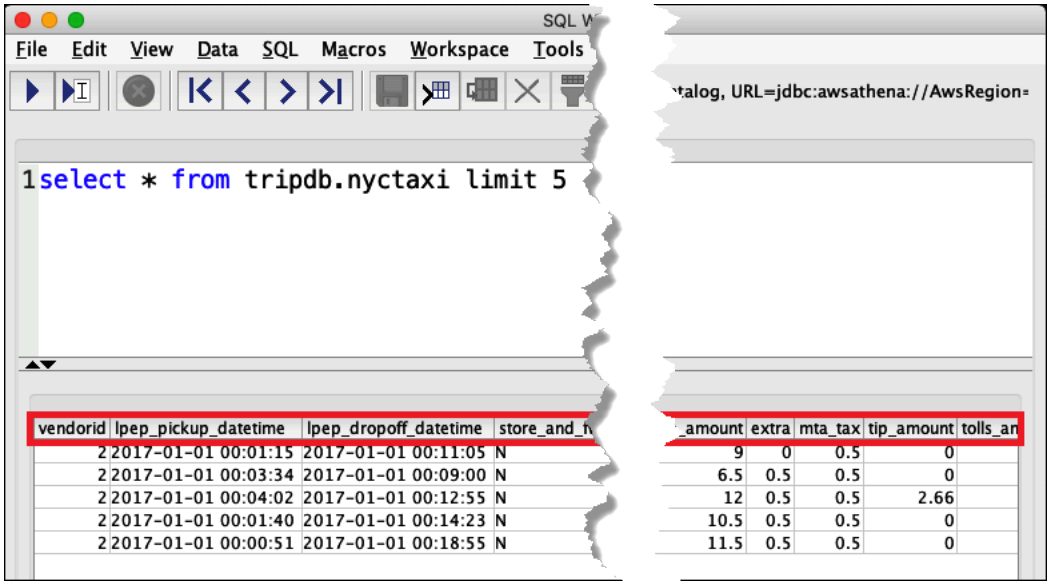
1. PilihUji, dan kemudian verifikasi bahwa koneksi berhasil.
2. Dari SQL WorkbenchPernyataan, jalankan SQL berikut`DESCRIBE`Perintah. Verifikasi bahwa semua kolom ditampilkan.

```
DESCRIBE "tripdb"."nyctaxi"
```



3. Dari SQL WorkbenchPernyataan, jalankan SQL berikutSELECTPerintah. Verifikasi bahwa semua kolom ditampilkan.

```
SELECT * FROM tripdb.nyctaxi LIMIT 5
```



Selanjutnya, Anda memverifikasi bahwa athena-ba-user, sebagai anggota lf-business-analystgrup, hanya memiliki akses ke tiga kolom pertama dari tabel yang Anda tentukan sebelumnya di Lake Formation.

Untuk memverifikasi akses untuk athena-ba-user

1. Dalam SQL Workbench, dalamProfil koneksikotak dialog, membuat profil koneksi lain.
   - Untuk nama profil koneksi, masukkan **`Athena_Okta_Group_Connection`**.
   - UntukDriver, pilih driver Simba Athena JDBC.
   - Untuk Source, lakukan salah satu hal berikut:
     - Untuk menggunakan URL koneksi, masukkan string koneksi satu baris. Contoh berikut menambahkan jeda baris untuk dibaca.

```
jdbc:awsathena://AwsRegion=region-id;
S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/athena_results;
AwsCredentialsProviderClass=com.simba.athena.iamsupport.plugin.OktaCredentialsProvider;
user=athena-ba-user@anycompany.com;
password=password;
idp_host=okta-idp-domain;
App_ID=okta-application-id;
SSL_Insecure=true;
LakeFormationEnabled=true;
```

- Untuk menggunakan URL AWS berbasis profil, lakukan langkah-langkah berikut:

  1. Konfigurasikan AWS profil yang memiliki file kredensial seperti contoh berikut.

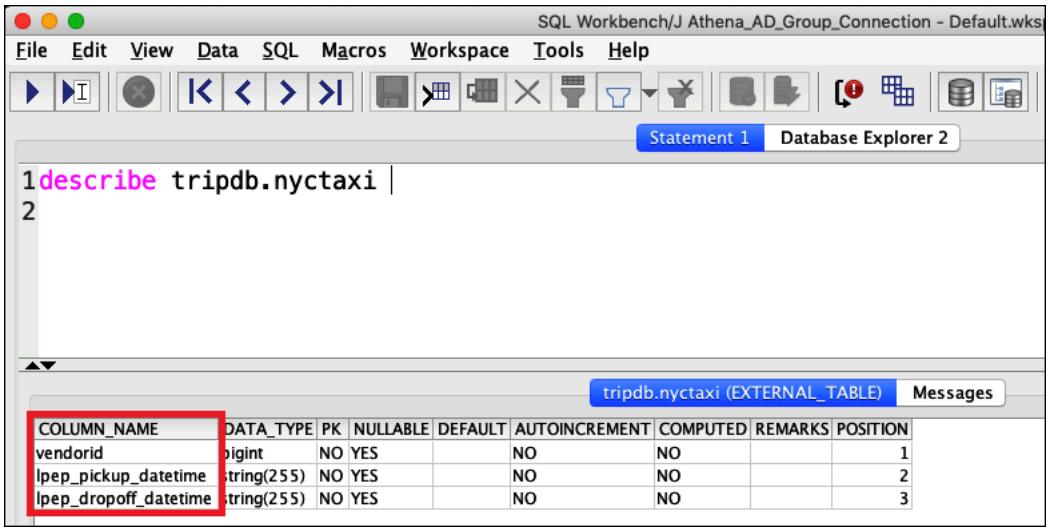
```
[athena_lf_ba]
plugin_name=com.simba.athena.iamsupport.plugin.OktaCredentialsProvider
idp_host=okta-idp-domain
app_id=okta-application-id
uid=athena-ba-user@anycompany.com
pwd=password
```

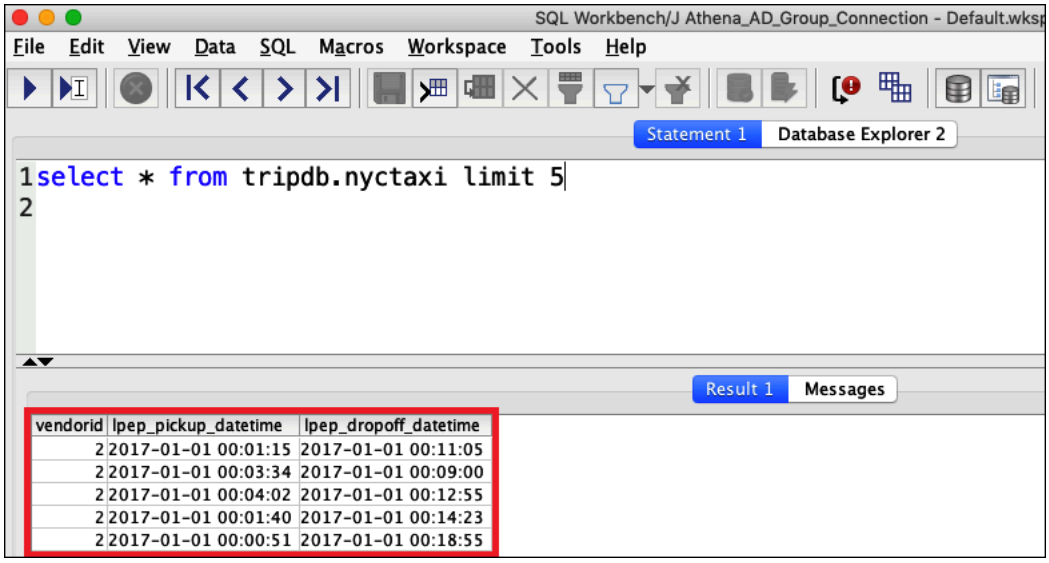
  2. UntukURL, masukkan string koneksi single-line seperti berikut. Contoh menambahkan jeda baris untuk dibaca.

```
jdbc:awsathena://AwsRegion=region-id;
S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/athena_results;
profile=athena_lf_ba;
SSL_Insecure=true;
LakeFormationEnabled=true;
```

2. PilihUjiuntuk mengkonfirmasi bahwa koneksi berhasil.
3. DariPernyataan SQLjendela, jalankan yang samaDESCRIBEdanSELECTPerintah SQL yang Anda lakukan sebelum dan memeriksa hasil.

   Karena athena-ba-useradalah anggota lf-business-analystgrup, hanya tiga kolom pertama yang Anda tentukan di konsol Lake Formation yang dikembalikan.

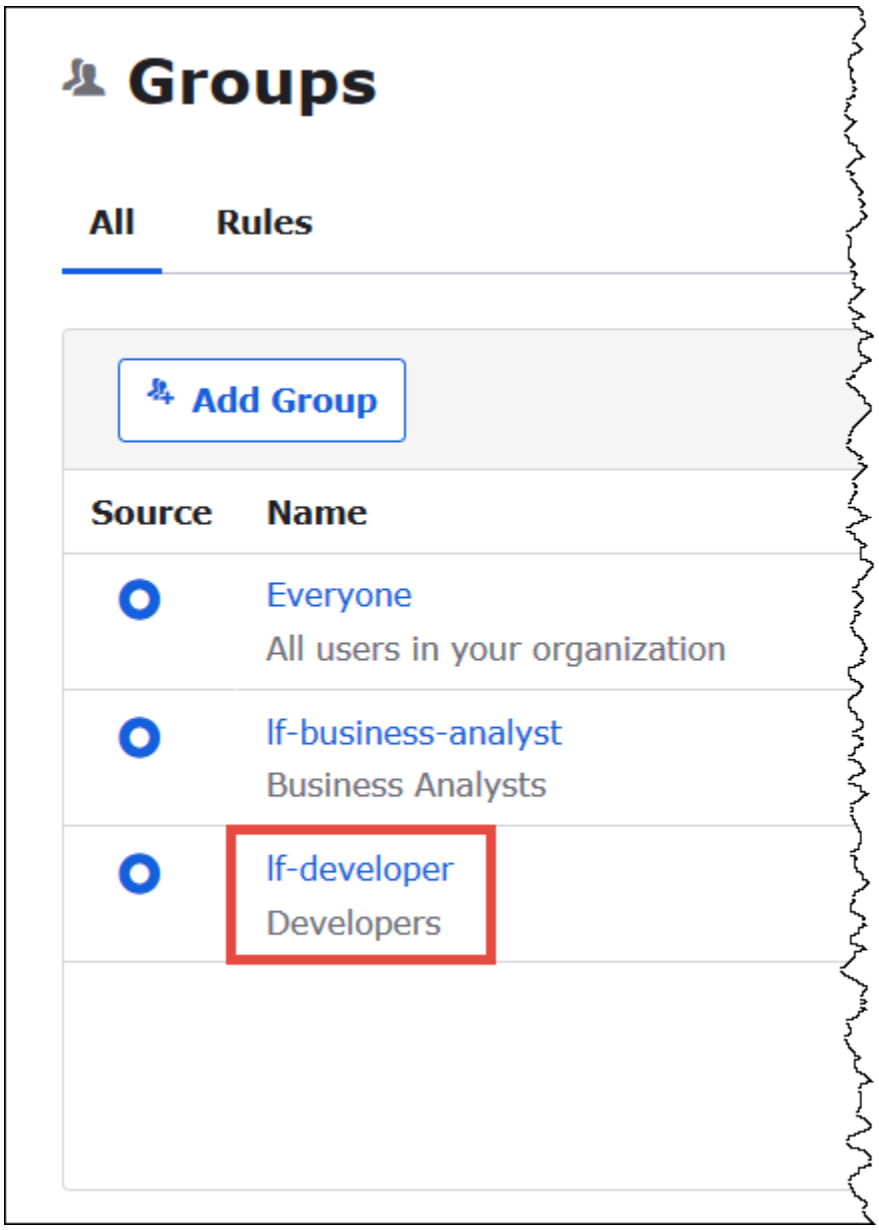




Selanjutnya, Anda kembali ke konsol Okta untuk menambahkan`athena-ba-user`kelf-`developer`Grup Okta.

Untuk menambahkan athena-ba-user ke grup pengembang lf

1. Masuk ke konsol Okta sebagai pengguna administratif dari domain Okta yang ditugaskan.
2. Pilih Grup, lalu pilih Tester.
3. Pada halaman Grup, pilih opsiPengembang lfkelompok.



4. PilihMengelola orang.
5. Dari daftar Bukan Anggota, pilih athena-ba-useruntuk menambahkannya ke grup pengembang lf.
6. Pilih Simpan.

Sekarang Anda kembali ke konsol Lake Formation untuk mengonfigurasi izin tabel untukPengembang lfkelompok.

Untuk mengkonfigurasi izin tabel untuk lf-developer-group

1. Masuk ke konsol Lake Formation sebagai administrator Data Lake.

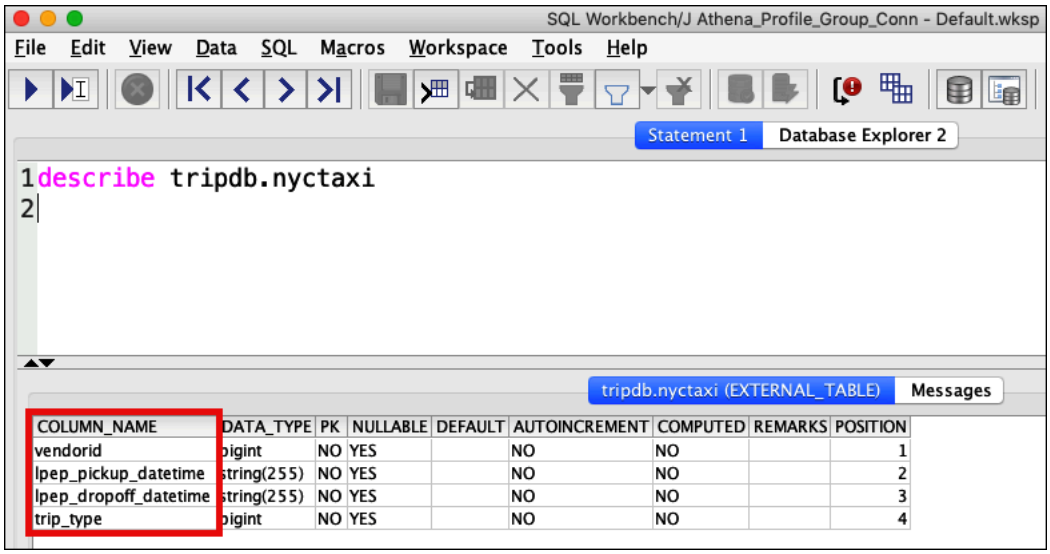
2. Di panel navigasi, pilih Tabel.
3. Pilihnyctaxitabel.
4. PilihTindakan,Izin.
5. DiBerikan Izin, masukkan informasi berikut:
   - Untuk QuickSight pengguna dan grup SAFL dan Amazon, masukkan ARN grup pengembang lf Okta SAM dalam format berikut:
   - Untuk Kolom, Pilih jenis filter, pilih Sertakan kolom.
   - Pilih kolom trip_type.
   - UntukIzin tabel, pilih SELECT.
6. Pilih Izin.

Sekarang Anda dapat menggunakan SQL Workbench untuk memverifikasi perubahan izin untuk grup lf-developer. Perubahan harus tercermin dalam data yang tersedia untuk athena-ba-user, yang sekarang menjadi anggota grup pengembang lf.

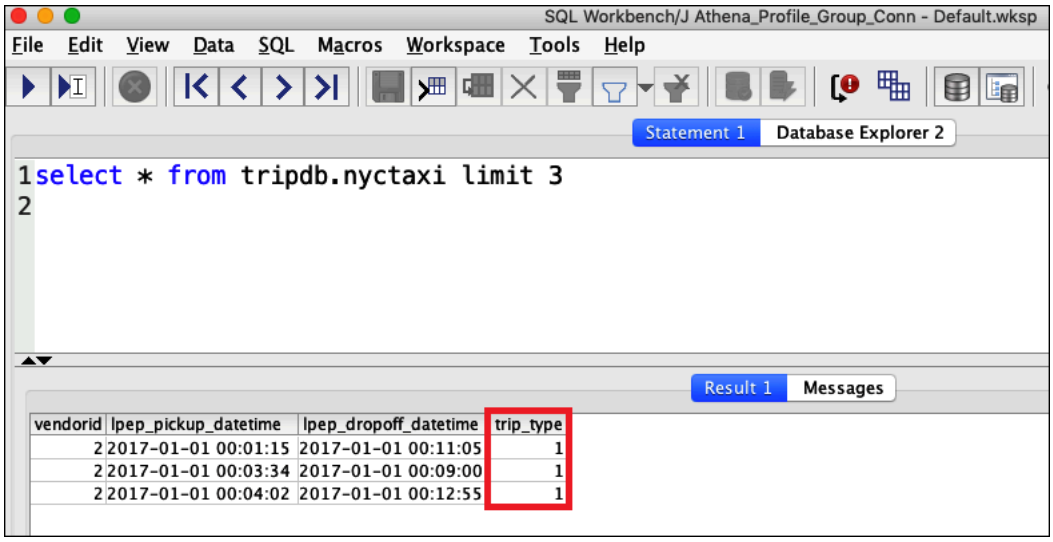
Untuk memverifikasi perubahan izin untuk athena-ba-user

1. Tutup program SQL Workbench, dan kemudian buka kembali.
2. Connect ke profil untuk athena-ba-user.
3. DariPernyataanjendela, masalah pernyataan SQL yang sama yang Anda jalankan sebelumnya:

   Kali ini,trip_typekolom ditampilkan.



Karena athena-ba-usersekarang menjadi anggota pengembang lf dan lf-business-analystgrup, kombinasi izin Lake Formation untuk grup tersebut menentukan kolom yang dikembalikan.



Kesimpulan

Dalam tutorial ini Anda mengonfigurasi integrasi Athena dengan AWS Lake Formation menggunakan Okta sebagai penyedia SAFL. Anda menggunakan Lake Formation dan IAM untuk mengontrol sumber daya yang tersedia untuk pengguna SAFL di Katalog Data lake AWS Glue data Anda.

Sumber daya terkait

Untuk informasi terkait, lihat sumber daya berikut:

- Menghubungkan ke Amazon Athena dengan JDBC
- Mengaktifkan akses federasi ke Athena API
- AWS Lake Formation Panduan Pengembang
- Memberikan dan mencabut izin Katalog Data di Panduan Pengembang.AWS Lake Formation
- Penyedia identitas dan federasi dalam Panduan Pengguna IAM.
- Membuat penyedia identitas SALL IAM di Panduan Pengguna IAM.
- Mengaktifkan federasi untuk AWS menggunakan Windows Active Directory, ADFS, dan SAFL 2.0 di Blog Keamanan.AWS

Manajemen beban kerja

Anda dapat menggunakan grup kerja Athena, manajemen kapasitas, penyetelan kinerja, dukungan kompresi, tag, dan fitur kuota layanan untuk mengelola beban kerja Anda.

Topik

Menggunakan kelompok kerja untuk mengontrol akses kueri dan biaya

Gunakan grup kerja untuk memisahkan pengguna, tim, aplikasi, atau beban kerja, untuk menetapkan batasan jumlah data yang dapat diproses oleh setiap kueri atau seluruh grup kerja, dan untuk melacak biaya. Karena grup kerja bertindak sebagai sumber daya, Anda dapat menggunakan kebijakan berbasis identitas tingkat sumber daya untuk mengontrol akses ke grup kerja tertentu. Anda juga dapat melihat metrik terkait kueri di Amazon CloudWatch, mengontrol biaya dengan mengonfigurasi batas jumlah data yang dipindai, membuat ambang batas, dan memicu tindakan, seperti Amazon SNS, saat ambang batas ini dilanggar.

Untuk mengontrol biaya lebih lanjut, Anda dapat membuat reservasi kapasitas dengan jumlah unit pemrosesan data yang Anda tentukan dan menambahkan satu atau beberapa kelompok kerja ke reservasi. Untuk informasi selengkapnya, lihat Mengelola kapasitas pemrosesan kueri.

Workgroup terintegrasi dengan IAM, CloudWatch Amazon Simple Notification Service, dan Laporan AWS Biaya dan Penggunaan sebagai berikut:

- Kebijakan berbasis identitas IAM dengan izin tingkat sumber daya mengontrol siapa yang dapat menjalankan kueri di grup kerja.
- Athena menerbitkan metrik kueri grup kerja ke, jika Anda mengaktifkan metrik CloudWatch kueri.
- Di Amazon SNS, Anda dapat membuat topik Amazon SNS yang mengeluarkan alarm ke pengguna grup kerja tertentu saat kontrol penggunaan data untuk kueri dalam grup kerja melebihi ambang batas yang ditetapkan.

- Saat Anda menandai grup kerja dengan tag yang dikonfigurasi sebagai tag alokasi biaya di konsol Billing and Cost Management, biaya yang terkait dengan menjalankan kueri di grup kerja tersebut akan muncul di Laporan Biaya dan Penggunaan dengan tag alokasi biaya tersebut.

Topik

- Menggunakan workgroup untuk menjalankan kueri
- Mengontrol biaya dan memantau kueri dengan CloudWatch metrik dan peristiwa

Lihat juga posting Blog AWS Big Data Memisahkan kueri dan mengelola biaya menggunakan grup kerja Amazon Athena, yang menunjukkan cara menggunakan grup kerja untuk memisahkan beban kerja, mengontrol akses pengguna, dan mengelola penggunaan dan biaya kueri.

Menggunakan workgroup untuk menjalankan kueri

Sebaiknya gunakan kelompok kerja untuk mengisolasi kueri untuk tim, aplikasi, atau beban kerja yang berbeda. Misalnya, Anda dapat membuat grup kerja terpisah untuk dua tim berbeda di organisasi Anda. Anda juga dapat memisahkan beban kerja. Misalnya, Anda dapat membuat dua kelompok kerja independen, satu untuk aplikasi terjadwal otomatis, seperti pembuatan laporan, dan satu lagi untuk penggunaan ad-hoc oleh analis. Anda dapat beralih di antara kelompok kerja.

Topik

- Manfaat menggunakan kelompok kerja
- Bagaimana kelompok kerja bekerja
- Menyiapkan kelompok kerja
- Kebijakan IAM untuk mengakses workgroup
- Pengaturan Workgroup
- Mengelola kelompok kerja
- Menggunakan IAM Identity Center mengaktifkan kelompok kerja Athena
- API grup kerja Athena
- Pemecahan masalah kelompok kerja

Manfaat menggunakan kelompok kerja

Workgroup memungkinkan Anda untuk:

Mengisolasi pengguna, tim, aplikasi, atau beban kerja ke dalam grup.	Setiap kelompok kerja memiliki riwayat kueri yang berbeda dan daftar kueri yang disimpan. Untuk informasi selengkapnya, lihat Bagaimana kelompok kerja bekerja. Untuk semua kueri di workgroup, Anda dapat memilih untuk mengonfigurasi pengaturan workgroup. Mereka menyertakan lokasi Amazon S3 untuk menyimpan hasil kueri, pemilik bucket yang diharapkan, enkripsi, dan kontrol objek yang ditulis ke bucket hasil kueri. Anda juga dapat menerapkan pengaturan workgroup. Untuk informasi selengkapnya, lihat Pengaturan Workgroup.
Menegakkan kendala biaya.	Anda dapat mengatur dua jenis batasan biaya untuk kueri dalam grup kerja: • Batas per kueri adalah ambang batas untuk jumlah data yang dipindai untuk setiap kueri. Athena membatalkan kueri saat melebihi ambang batas yang ditentukan. Batas berlaku untuk setiap kueri yang berjalan dalam workgroup. Anda hanya dapat menetapkan satu batas per kueri dan memperbaruinya jika diperlukan. • Batas per grup kerja adalah ambang batas yang dapat Anda tetapkan untuk setiap grup kerja untuk jumlah data yang dipindai oleh kueri di grup kerja. Melanggar ambang batas mengaktifkan alarm Amazon SNS yang memicu tindakan pilihan Anda, seperti mengirim email ke pengguna tertentu. Anda dapat menetapkan beberapa batas per workgroup untuk setiap workgroup. Untuk langkah mendetail, lihat Mengatur batas kontrol penggunaan data.
Lacak metrik terkait kueri untuk semua kueri grup kerja di. CloudWatch	Untuk setiap kueri yang berjalan di workgroup, jika Anda mengonfig urasi workgroup untuk menerbitkan metrik, Athena menerbitkannya. CloudWatch Anda dapat melihat metrik kueri untuk setiap grup kerja Anda dalam konsol Athena. Di CloudWatch, Anda dapat membuat dasbor khusus, dan mengatur ambang batas dan alarm pada metrik ini.

Bagaimana kelompok kerja bekerja

Kelompok kerja di Athena memiliki karakteristik sebagai berikut:

- Secara default, setiap akun memiliki workgroup utama dan izin default memungkinkan semua pengguna yang diautentikasi mengakses workgroup ini. Workgroup utama tidak dapat dihapus.
- Setiap grup kerja yang Anda buat menampilkan kueri tersimpan dan riwayat kueri hanya untuk kueri yang berjalan di dalamnya, dan tidak untuk semua kueri di akun. Ini memisahkan kueri Anda dari kueri lain dalam akun dan membuatnya lebih efisien bagi Anda untuk menemukan kueri dan kueri tersimpan Anda sendiri dalam riwayat.
- Menonaktifkan workgroup mencegah kueri berjalan di dalamnya, hingga Anda mengaktifkannya. Kueri yang dikirim ke grup kerja yang dinonaktifkan gagal, hingga Anda mengaktifkannya lagi.
- Jika memiliki izin, Anda dapat menghapus grup kerja kosong, dan grup kerja yang berisi kueri tersimpan. Dalam hal ini, sebelum menghapus workgroup, Athena memperingatkan Anda bahwa kueri yang disimpan akan dihapus. Sebelum menghapus grup kerja yang dapat diakses pengguna lain, pastikan penggunanya memiliki akses ke grup kerja lain di mana mereka dapat terus menjalankan kueri.
- Anda dapat mengatur setelan seluruh grup kerja dan menerapkan penggunaannya dengan semua kueri yang berjalan di grup kerja. Setelan tersebut mencakup lokasi hasil kueri di Amazon S3, pemilik bucket yang diharapkan, enkripsi, dan kontrol objek yang ditulis ke bucket hasil kueri.

> **Important**
>
> Saat Anda menerapkan setelan seluruh grup kerja, semua kueri yang berjalan di grup kerja ini menggunakan setelan workgroup. Ini terjadi bahkan jika pengaturan sisi klien mereka mungkin berbeda dari pengaturan grup kerja. Untuk informasi, lihat Pengaturan Workgroup mengesampingkan setelan sisi klien.

Keterbatasan untuk kelompok kerja

- Anda dapat membuat hingga 1000 grup kerja per Wilayah di akun Anda.
- Workgroup utama tidak dapat dihapus.
- Anda dapat membuka hingga sepuluh tab kueri dalam setiap workgroup. Saat Anda beralih di antara grup kerja, tab kueri Anda tetap terbuka hingga tiga grup kerja.

Menyiapkan kelompok kerja

Menyiapkan grup kerja melibatkan pembuatan dan menetapkan izin untuk penggunaannya. Pertama, tentukan kelompok kerja mana yang dibutuhkan organisasi Anda, dan buat mereka. Selanjutnya, siapkan kebijakan grup kerja IAM yang mengontrol akses dan tindakan pengguna pada sumber daya. `workgroup` Pengguna dengan akses ke workgroup ini sekarang dapat menjalankan kueri di dalamnya.

> **Note**
>
> Gunakan tugas-tugas ini untuk menyiapkan kelompok kerja saat Anda mulai menggunakannya untuk pertama kalinya. Jika akun Athena Anda sudah menggunakan grup kerja, setiap pengguna akun memerlukan izin untuk menjalankan kueri di satu atau beberapa grup kerja di akun tersebut. Sebelum menjalankan kueri, periksa kebijakan IAM Anda untuk melihat grup kerja mana yang dapat Anda akses, sesuaikan kebijakan Anda jika diperlukan, dan beralih ke grup kerja yang ingin Anda gunakan.

Secara default, jika Anda belum membuat grup kerja apa pun, semua kueri di akun Anda berjalan di workgroup utama.

Athena menampilkan workgroup saat ini di opsi Workgroup di kanan atas konsol. Anda dapat menggunakan opsi ini untuk beralih kelompok kerja. Saat Anda menjalankan kueri, kueri tersebut berjalan di workgroup saat ini. Anda dapat menjalankan kueri dalam konteks workgroup di konsol, melalui operasi API, melalui antarmuka baris perintah, atau melalui aplikasi klien dengan menggunakan driver JDBC atau ODBC. Jika Anda memiliki akses ke grup kerja, Anda dapat melihat setelan, metrik, dan batas kontrol penggunaan data grup kerja. Dengan izin tambahan, Anda dapat mengedit pengaturan dan batas kontrol penggunaan data.

Untuk mengatur kelompok kerja

1. Tentukan kelompok kerja mana yang akan dibuat. Misalnya, Anda dapat memutuskan yang berikut:

   - Siapa yang dapat menjalankan kueri di setiap workgroup, dan siapa yang memiliki konfigurasi workgroup. Ini menentukan kebijakan IAM yang Anda buat. Untuk informasi selengkapnya, lihat Kebijakan IAM untuk mengakses workgroup.
   - Lokasi mana di Amazon S3 yang akan digunakan untuk hasil kueri untuk kueri yang berjalan di setiap grup kerja. Lokasi harus ada di Amazon S3 sebelum Anda dapat menentukannya untuk

hasil kueri grup kerja. Semua pengguna yang menggunakan workgroup harus memiliki akses ke lokasi ini. Untuk informasi selengkapnya, lihat Pengaturan Workgroup.

- Apakah pemilik bucket hasil kueri Amazon S3 memiliki kontrol penuh atas objek baru yang ditulis ke bucket. Misalnya, jika lokasi hasil kueri Anda dimiliki oleh akun lain, Anda dapat memberikan kepemilikan dan kontrol penuh atas hasil kueri Anda ke akun lain. Untuk informasi lebih lanjut, lihat AclConfiguration.
- Tentukan ID Akun AWS yang Anda harapkan sebagai pemilik bucket lokasi keluaran. Ini adalah tindakan keamanan tambahan opsional. Jika ID akun pemilik bucket tidak cocok dengan ID yang Anda tentukan di sini, upaya untuk menampilkan ke bucket akan gagal. Untuk informasi selengkapnya, lihat Memverifikasi kepemilikan bucket dengan kondisi pemilik bucket di Panduan Pengguna Amazon S3. Pengaturan ini tidak berlaku untuk pernyataan CTAS, INSERT INTO, atau UNLOAD.
- Pengaturan enkripsi mana yang diperlukan, dan kelompok kerja mana yang memiliki kueri yang harus dienkripsi. Kami menyarankan Anda membuat grup kerja terpisah untuk kueri terenkripsi dan tidak terenkripsi. Dengan begitu, Anda dapat menerapkan enkripsi untuk grup kerja yang berlaku untuk semua kueri yang berjalan di dalamnya. Untuk informasi selengkapnya, lihat Mengenkripsi hasil kueri Athena yang disimpan di Amazon S3.

2. Buat workgroup sesuai kebutuhan, dan tambahkan tag ke dalamnya. Untuk langkah, lihat Buat grup kerja.
3. Buat kebijakan IAM untuk pengguna, grup, atau peran Anda untuk mengaktifkan akses mereka ke grup kerja. Kebijakan menetapkan keanggotaan kelompok kerja dan akses ke tindakan pada `workgroup` sumber daya. Untuk langkah mendetail, lihat Kebijakan IAM untuk mengakses workgroup. Misalnya kebijakan JSON, lihatAkses ke grup kerja dan tag.
4. Mengatur pengaturan workgroup. Tentukan lokasi di Amazon S3 untuk hasil kueri dan secara opsional tentukan pemilik bucket yang diharapkan, setelan enkripsi, dan kontrol objek yang ditulis ke bucket hasil kueri. Anda dapat menerapkan pengaturan workgroup. Untuk informasi selengkapnya, lihat pengaturan workgroup.

> **Important**
>
> Jika Anda mengganti pengaturan sisi klien, Athena akan menggunakan pengaturan grup kerja. Ini memengaruhi kueri yang Anda jalankan di konsol, dengan menggunakan driver, antarmuka baris perintah, atau operasi API.
> Sementara kueri terus berjalan, otomatisasi yang dibuat berdasarkan ketersediaan hasil di bucket Amazon S3 tertentu dapat rusak. Kami menyarankan Anda memberi tahu

pengguna Anda sebelum mengganti. Setelah pengaturan workgroup disetel ke override, Anda dapat menghilangkan penetapan setelan sisi klien di driver atau API.

5. Beri tahu pengguna workgroup mana yang akan digunakan untuk menjalankan kueri. Kirim email untuk memberi tahu pengguna akun Anda tentang nama grup kerja yang dapat mereka gunakan, kebijakan IAM yang diperlukan, dan pengaturan grup kerja.
6. Konfigurasikan batas kontrol biaya, juga dikenal sebagai batas kontrol penggunaan data, untuk kueri dan grup kerja. Untuk memberi tahu Anda saat ambang batas dilanggar, buat topik Amazon SNS dan konfigurasikan langganan. Untuk langkah-langkah mendetail, lihat Mengatur batas kontrol penggunaan data dan Memulai Amazon SNS di Panduan Pengembang Layanan Pemberitahuan Sederhana Amazon.
7. Beralih ke workgroup sehingga Anda dapat menjalankan kueri.Untuk menjalankan kueri, beralih ke workgroup yang sesuai. Untuk langkah mendetail, lihat the section called “Tentukan workgroup untuk menjalankan kueri”.

Kebijakan IAM untuk mengakses workgroup

Untuk mengontrol akses ke grup kerja, gunakan izin IAM tingkat sumber daya atau kebijakan IAM berbasis identitas. Setiap kali Anda menggunakan kebijakan IAM, pastikan bahwa Anda mengikuti praktik terbaik IAM. Untuk informasi selengkapnya tentang administrator, lihat Praktik terbaik keamanan di IAM dalam Panduan Pengguna IAM.

> **Note**
>
> Untuk mengakses grup kerja yang diaktifkan propagasi identitas tepercaya, pengguna IAM Identity Center harus ditetapkan ke `IdentityCenterApplicationArn` yang dikembalikan oleh respons tindakan API Athena. GetWorkGroup

Prosedur berikut khusus untuk Athena.

Untuk informasi khusus IAM, lihat tautan yang tercantum di akhir bagian ini. Untuk informasi tentang contoh kebijakan grup kerja JSON, lihat. Kebijakan contoh kelompok kerja

Untuk menggunakan editor visual di konsol IAM untuk membuat kebijakan workgroup

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.

2. Pada panel navigasi yang ada di sebelah kiri, pilih Kebijakan, lalu pilih Buat kebijakan.
3. Pada tab Editor visual, pilih Pilih layanan. Pilih Athena untuk ditambahkan ke kebijakan.
4. Pilih Pilih tindakan, kemudian pilih tindakan untuk ditambahkan ke kebijakan. Editor visual menunjukkan tindakan yang tersedia di Athena. Untuk informasi selengkapnya, lihat Kunci tindakan, sumber daya, dan kondisi untuk Amazon Athena di Referensi Otorisasi Layanan.
5. Pilih tambahkan tindakanuntuk mengetik tindakan tertentu atau gunakan wildcard (*) untuk menentukan beberapa tindakan.

   Secara default, kebijakan yang Anda buat mengizinkan tindakan yang Anda pilih. Jika Anda memilih satu atau lebih tindakan yang mendukung izin level sumber daya ke sumber daya `workgroup` di Athena, editor akan mencantumkan sumber daya `workgroup` tersebut.
6. Pilih Sumber daya untuk menentukan grup kerja tertentu untuk kebijakan Anda. Misalnya kebijakan grup kerja JSON, lihat. Kebijakan contoh kelompok kerja
7. Tentukan sumber daya `workgroup` sebagai berikut:

   ```
   arn:aws:athena:<region>:<user-account>:workgroup/<workgroup-name>
   ```
8. Pada halaman Tinjau kebijakan, ketik Nama dan Deskripsi (opsional) untuk kebijakan yang Anda buat. Tinjau ringkasan kebijakan untuk memastikan bahwa Anda memberikan izin yang dimaksud.
9. Pilih Buat kebijakan untuk menyimpan kebijakan baru Anda.
10. Lampirkan kebijakan berbasis identitas ini ke pengguna, grup, atau peran.

Untuk informasi selengkapnya, lihat topik berikut di Referensi Otorisasi Layanan dan Panduan Pengguna IAM:

- Tindakan, sumber daya, dan kunci kondisi untuk Amazon Athena
- Membuat kebijakan dengan editor visual
- Menambahkan dan menghapus kebijakan IAM
- Mengontrol akses ke sumber daya

Misalnya kebijakan grup kerja JSON, lihat. Kebijakan contoh kelompok kerja

Untuk daftar lengkap tindakan Amazon Athena, lihat nama tindakan API diReferensi API Amazon Athena.

Kebijakan contoh kelompok kerja

Bagian ini mencakup contoh kebijakan yang dapat Anda gunakan untuk mengaktifkan berbagai tindakan pada kelompok kerja. Setiap kali Anda menggunakan kebijakan IAM, pastikan bahwa Anda mengikuti praktik terbaik IAM. Untuk informasi selengkapnya tentang administrator, lihat Praktik terbaik keamanan di IAM dalam Panduan Pengguna IAM.

Workgroup adalah sumber daya IAM yang dikelola oleh Athena. Oleh karena itu, jika kebijakan workgroup Anda menggunakan tindakan yang diambil `workgroup` sebagai input, Anda harus menentukan ARN workgroup sebagai berikut:

```
"Resource": [arn:aws:athena:<region>:<user-account>:workgroup/<workgroup-name>]
```

*<workgroup-name>*Dimana nama workgroup Anda. Misalnya, untuk kelompok kerja bernama`test_workgroup`, tentukan sebagai sumber daya sebagai berikut:

```
"Resource": ["arn:aws:athena:us-east-1:123456789012:workgroup/test_workgroup"]
```

Untuk daftar lengkap tindakan Amazon Athena, lihat nama tindakan API diReferensi API Amazon Athena. Untuk informasi selengkapnya tentang kebijakan IAM, lihat Membuat kebijakan dengan editor visual di Panduan Pengguna IAM. Untuk informasi selengkapnya tentang cara membuat kebijakan IAM untuk grup kerja, lihat Kebijakan IAM untuk mengakses workgroup.

- 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 Contoh kebijakan untuk akses penuh ke semua kelompok kerja

Kebijakan berikut memungkinkan akses penuh ke semua sumber daya grup kerja yang mungkin ada di akun. Kami menyarankan Anda menggunakan kebijakan ini untuk pengguna di akun Anda yang harus mengelola dan mengelola grup kerja untuk semua pengguna lain.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Example Contoh kebijakan untuk akses penuh ke workgroup tertentu

Kebijakan berikut memungkinkan akses penuh ke satu sumber daya grup kerja tertentu, yang diberi nama`workgroupA`. Anda dapat menggunakan kebijakan ini untuk pengguna dengan kontrol penuh atas grup kerja tertentu.

```
{
    "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 Contoh kebijakan untuk menjalankan kueri di workgroup tertentu

Dalam kebijakan berikut, pengguna diizinkan untuk menjalankan kueri dalam yang ditentukan`workgroupA`, dan melihatnya. Pengguna tidak diperbolehkan melakukan tugas manajemen untuk workgroup itu sendiri, seperti memperbarui atau menghapusnya.

```
{
    "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 Contoh kebijakan untuk menjalankan kueri di workgroup utama

Anda dapat memodifikasi contoh sebelumnya untuk memungkinkan pengguna tertentu juga menjalankan kueri di workgroup utama.

> **Note**
>
> Sebaiknya Anda menambahkan sumber daya workgroup utama untuk semua pengguna yang dikonfigurasi untuk menjalankan kueri di grup kerja yang ditunjuk. Menambahkan sumber daya ini ke kebijakan pengguna grup kerja mereka berguna jika grup kerja yang ditunjuk dihapus atau dinonaktifkan. Dalam hal ini, mereka dapat terus menjalankan kueri di workgroup utama.

Untuk mengizinkan pengguna di akun Anda menjalankan kueri di grup kerja utama, tambahkan baris yang berisi ARN grup kerja utama ke bagian sumber dayaExample policy for running queries in a specified workgroup, seperti pada contoh berikut.

```
arn:aws:athena:us-east-1:123456789012:workgroup/primary"
```

Example Contoh kebijakan untuk operasi manajemen pada workgroup tertentu

Dalam kebijakan berikut, pengguna diizinkan untuk membuat, menghapus, memperoleh detail, dan memperbarui grup kerja`test_workgroup`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListEngineVersions"
            ],
```

```
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "athena:CreateWorkGroup",
                "athena:GetWorkGroup",
                "athena:DeleteWorkGroup",
                "athena:UpdateWorkGroup"
            ],
            "Resource": [
                "arn:aws:athena:us-east-1:123456789012:workgroup/test_workgroup"
            ]
        }
    ]
}
```

Example Contoh kebijakan untuk daftar kelompok kerja

Kebijakan berikut memungkinkan semua pengguna untuk mencantumkan semua grup kerja:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListWorkGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

Example Contoh kebijakan untuk menjalankan dan menghentikan kueri di grup kerja tertentu

Dalam kebijakan ini, pengguna diizinkan untuk menjalankan kueri di grup kerja:

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

```
            "Action": [
                "athena:StartQueryExecution",
                "athena:StopQueryExecution"
            ],
            "Resource": [
                "arn:aws:athena:us-east-1:123456789012:workgroup/test_workgroup"
            ]
        }
    ]
}
```

Example Contoh kebijakan untuk bekerja dengan kueri bernama dalam kelompok kerja tertentu

Dalam kebijakan berikut, pengguna memiliki izin untuk membuat, menghapus, dan memperoleh informasi tentang kueri bernama di grup kerja yang ditentukan:

```
{
    "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 Contoh kebijakan untuk bekerja dengan notebook Spark di Athena

Gunakan kebijakan seperti berikut ini untuk bekerja dengan notebook Spark di Athena.

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

```
                "athena:CreateWorkGroup",
                "s3:CreateBucket",
                "iam:CreateRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy",
                "s3:GetBucketLocation",
                "athena:ImportNotebook"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:athena:us-east-1:123456789012:workgroup/Demo*",
                "arn:aws:s3:::123456789012-us-east-1-athena-results-bucket-*",
                "arn:aws:iam::123456789012:role/service-role/
AWSAthenaSparkExecutionRole-*",
                "arn:aws:iam::123456789012:policy/service-role/
AWSAthenaSparkRolePolicy-*"
            ]
        },
        {
            "Sid": "AllowRunningCalculations",
            "Action": [
                "athena:ListWorkGroups",
                "athena:GetWorkGroup",
                "athena:StartSession",
                "athena:CreateNotebook",
                "athena:ListNotebookMetadata",
                "athena:ListNotebookSessions",
                "athena:GetSessionStatus",
                "athena:GetSession",
                "athena:GetNotebookMetadata",
                "athena:CreatePresignedNotebookUrl"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:athena:us-east-1:123456789012:workgroup/Demo*"
        },
        {
            "Sid": "AllowListWorkGroupAndEngineVersions",
            "Action": [
                "athena:ListWorkGroups",
                "athena:ListEngineVersions"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
```

```
    ]
}
```

Pengaturan Workgroup

Setiap workgroup memiliki pengaturan berikut:

- Nama yang unik. Ini dapat berisi dari 1 hingga 128 karakter, termasuk karakter alfanumerik, tanda hubung, dan garis bawah. Setelah Anda membuat workgroup, Anda tidak dapat mengubah namanya. Namun, Anda dapat membuat workgroup baru dengan pengaturan yang sama dan nama yang berbeda.
- Pengaturan yang berlaku untuk semua kueri yang berjalan di workgroup. Mereka termasuk:
  - Lokasi di Amazon S3 untuk menyimpan hasil kueri untuk semua kueri yang berjalan di workgroup ini. Lokasi ini harus ada sebelum Anda menentukannya untuk workgroup saat Anda membuatnya. Untuk informasi tentang membuat bucket Amazon S3, lihat Membuat bucket.
  - Kontrol pemilik bucket atas hasil kueri — Apakah pemilik bucket hasil kueri Amazon S3 memiliki kontrol penuh atas objek baru yang ditulis ke bucket. Misalnya, jika lokasi hasil kueri Anda dimiliki oleh akun lain, Anda dapat memberikan kepemilikan dan kontrol penuh atas hasil kueri Anda ke akun lain.
  - Pemilik bucket yang diharapkan — ID Akun AWS yang Anda harapkan menjadi pemilik bucket hasil kueri. Ini adalah langkah keamanan tambahan. Jika ID akun pemilik bucket tidak cocok dengan ID yang Anda tentukan di sini, upaya untuk menampilkan ke bucket akan gagal. Untuk informasi lebih lanjut, lihat Memverifikasi kepemilikan bucket dengan kondisi pemilik bucket di Panduan Pengguna Amazon S3..

    > **Note**
    >
    > Setelan pemilik bucket yang diharapkan hanya berlaku untuk lokasi keluaran Amazon S3 yang Anda tentukan untuk hasil kueri Athena. Ini tidak berlaku untuk lokasi Amazon S3 lainnya seperti lokasi sumber data di bucket Amazon S3 eksternal`CTAS`, `INSERT INTO` dan lokasi tabel tujuan, lokasi keluaran pernyataan`UNLOAD`, operasi untuk menumpahkan bucket untuk kueri federasi, `SELECT` atau kueri yang dijalankan terhadap tabel di akun lain.
  - Pengaturan enkripsi, jika Anda menggunakan enkripsi untuk semua kueri workgroup. Anda hanya dapat mengenkripsi semua kueri dalam grup kerja, bukan hanya beberapa di antaranya.

Cara terbaik adalah membuat workgroup terpisah untuk berisi kueri yang dienkripsi atau tidak dienkripsi.

Selain itu, grup kerja Anda dapat mengganti setelan sisi klien. Sebelum rilis grup kerja, Anda dapat menentukan lokasi hasil dan opsi enkripsi sebagai parameter di driver JDBC atau ODBC, atau di tab Properti di konsol Athena. Pengaturan ini juga dapat ditentukan secara langsung melalui operasi API. Pengaturan ini dikenal sebagai "pengaturan sisi klien". Dengan workgroup, Anda dapat mengonfigurasi pengaturan ini di tingkat workgroup untuk mengontrol opsi yang tersedia di tingkat klien. Menegakkan pengaturan tingkat workgroup juga menyelamatkan pengguna dari keharusan mengonfigurasi setelan sisi klien mereka satu per satu. Jika Anda memilih opsi Override Client-Side Settings untuk workgroup, kueri menggunakan pengaturan workgroup dan mengabaikan pengaturan sisi klien.

Jika Ganti Pengaturan Sisi Klien dipilih, pengguna akan diberi tahu di konsol bahwa pengaturannya telah berubah. Jika pengaturan workgroup diberlakukan dengan cara ini, pengguna dapat menghilangkan pengaturan sisi klien yang sesuai. Kemudian, kueri yang berjalan di konsol menggunakan pengaturan workgroup meskipun ada pengaturan sisi klien. Selain itu, ketika kueri di workgroup dijalankan melalui, operasi API AWS CLI, atau driver JDBC atau ODBC, pengaturan sisi klien seperti lokasi hasil kueri dan enkripsi diganti oleh pengaturan workgroup. Untuk melihat setelan workgroup, lihat detail workgroup.

Anda juga dapat menetapkan batas kueri untuk kueri dalam kelompok kerja.

Pengaturan Workgroup mengesampingkan setelan sisi klien

Dialog Create workgroup dan Edit workgroup memiliki field berjudul Override client-side settings. Bidang ini tidak dipilih secara default. Tergantung pada apakah Anda memilihnya, Athena melakukan hal berikut:

- Jika Override setelan sisi klien tidak dipilih, pengaturan workgroup tidak diberlakukan di tingkat klien. Jika opsi ganti setelan sisi klien tidak dipilih untuk grup kerja, Athena menggunakan setelan klien untuk semua kueri yang berjalan di grup kerja, termasuk pengaturan untuk lokasi hasil kueri, pemilik bucket yang diharapkan, enkripsi, dan kontrol objek yang ditulis ke bucket hasil kueri. Setiap pengguna dapat menentukan pengaturan mereka sendiri di menu Pengaturan di konsol. Jika pengaturan sisi klien tidak disetel, pengaturan seluruh grup kerja berlaku. Jika Anda menggunakan, tindakan API AWS CLI, atau driver JDBC dan ODBC untuk menjalankan kueri dalam grup kerja yang tidak mengganti setelan sisi klien, kueri Anda menggunakan setelan yang Anda tentukan dalam kueri.

- Jika Override setelan sisi klien dipilih, pengaturan workgroup diberlakukan di tingkat workgroup untuk semua klien di workgroup. Saat opsi ganti setelan sisi klien dipilih untuk grup kerja, Athena menggunakan setelan grup kerja untuk semua kueri yang berjalan di grup kerja, termasuk pengaturan untuk lokasi hasil kueri, pemilik bucket yang diharapkan, enkripsi, dan kontrol objek yang ditulis ke bucket hasil kueri. Pengaturan Workgroup mengganti setelan sisi klien yang Anda tentukan untuk kueri saat Anda menggunakan konsol, tindakan API, atau driver JDBC atau ODBC.

  Jika Anda mengganti setelan sisi klien, saat berikutnya Anda atau pengguna grup kerja mana pun membuka konsol Athena, Athena memberi tahu Anda bahwa kueri di grup kerja menggunakan pengaturan grup kerja, dan meminta Anda untuk mengakui perubahan ini.

  > **Important**
  >
  > Jika Anda menggunakan tindakan API, driver AWS CLI, atau JDBC dan ODBC untuk menjalankan kueri dalam grup kerja yang mengganti setelan sisi klien, pastikan Anda menghilangkan setelan sisi klien dalam kueri atau memperbaruinya agar sesuai dengan pengaturan grup kerja. Jika Anda menentukan setelan sisi klien dalam kueri tetapi menjalankannya di grup kerja yang mengesampingkan pengaturan, kueri akan berjalan, tetapi pengaturan grup kerja akan digunakan. Untuk informasi tentang melihat setelan untuk grup kerja, lihatLihat detail workgroup.

Mengelola kelompok kerja

Di https://console.aws.amazon.com/athena/, Anda dapat melakukan tugas-tugas berikut:

Pernyataan	Deskripsi
Buat grup kerja	Buat workgroup baru.
Mengedit grup kerja	Edit workgroup dan ubah pengaturannya. Anda tidak dapat mengubah nama grup kerja, tetapi Anda dapat membuat grup kerja baru dengan pengaturan yang sama dan nama yang berbeda.
Lihat detail workgroup	Lihat detail grup kerja, seperti nama, deskripsi, batas penggunaan data, lokasi hasil kueri, pemilik bucket hasil kueri yang diharapkan, enkripsi, dan kontrol objek yang ditulis ke bucket hasil kueri. Anda juga

Pernyataan	Deskripsi
	dapat memverifikasi apakah workgroup ini memberlakukan pengaturannya, jika Override setelan sisi klien dicentang.
Menghapus grup kerja	Hapus grup kerja. Jika Anda menghapus grup kerja, riwayat kueri, kueri yang disimpan, pengaturan grup kerja, dan kontrol batas data per kueri akan dihapus. Kontrol batas data seluruh grup kerja tetap ada CloudWatch, dan Anda dapat menghapusnya satu per satu. Workgroup utama tidak dapat dihapus.
Beralih kelompok kerja	Beralih di antara kelompok kerja yang dapat Anda akses.
Salin kueri yang disimpan di antara kelompok kerja	Salin kueri yang disimpan di antara kelompok kerja. Anda mungkin ingin melakukan ini jika, misalnya, Anda membuat kueri di workgroup pratinjau dan Anda ingin membuatnya tersedia di workgroup nonpreview.
Mengaktifkan dan menonaktifkan workgroup	Mengaktifkan atau menonaktifkan workgroup. Saat workgroup dinonaktifkan, penggunanya tidak dapat menjalankan kueri, atau membuat kueri bernama baru. Jika Anda memiliki akses ke sana, Anda masih dapat melihat metrik, kontrol batas penggunaan data, pengaturan grup kerja, riwayat kueri, dan kueri yang disimpan.
Tentukan workgroup untuk menjalankan kueri	Sebelum Anda dapat menjalankan kueri, Anda harus menentukan ke Athena workgroup mana yang akan digunakan. Anda harus memiliki izin untuk workgroup.
Buat workgroup Athena yang menggunakan autentikasi IAM Identity Center	Untuk menggunakan identitas Pusat Identitas IAM dengan Athena, Anda harus membuat grup kerja yang diaktifkan Pusat Identitas IAM. Setelah membuat grup kerja, Anda dapat menggunakan konsol Pusat Identitas IAM atau API untuk menetapkan pengguna atau grup Pusat Identitas IAM ke grup kerja.

Buat grup kerja

Membuat workgroup memerlukan izin untuk tindakan `CreateWorkgroup` API. Lihat Akses ke grup kerja dan tag dan Kebijakan IAM untuk mengakses workgroup. Jika Anda menambahkan tag, Anda juga perlu menambahkan izin ke`TagResource`. Lihat Contoh kebijakan tag untuk kelompok kerja.

Untuk membuat workgroup di konsol

1. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.

2. Di panel navigasi konsol Athena, pilih Workgroups.
3. Pada Grup Kerja, pilih Buat grup kerja.
4. Pada halaman Create workgroup, isi kolom sebagai berikut:

Bidang	Deskripsi
Nama Workgroup	Wajib. Masukkan nama unik untuk workgroup Anda. Gunakan 1 - 128 karakter. (A-Z, a-z, 0-9, _, -,.). Nama ini tidak dapat diubah.
Deskripsi	Tidak wajib. Masukkan deskripsi untuk grup kerja Anda. Ini dapat berisi hingga 1024 karakter.
Pilih jenis mesinnya	Pilih Athena SQL jika Anda ingin menjalankan kueri SQL ad-hoc pada data di Amazon S3 atau gunakan konektor sumber data bawaan untuk menjalankan kueri gabungan pada berbagai sumber

Bidang	Deskripsi
	data di luar Amazon S3. Anda dapat menjalankan kueri menggunak an editor kueri Athena AWS CLI, atau Athena API. Pilih Apache Spark jika Anda ingin membuat, mengedit, dan menjalankan aplikasi notebook Jupyter menggunakan Python dan Apache Spark. Notebook Jupyter berisi daftar sel yang dapat mencakup kode, teks, Markdown, matematika, plot, dan media kaya. Sel dijalankan secara berurutan sebagai perhitungan dalam sesi notebook interaktif di Athena. Untuk informasi tentang membuat dan mengonfigurasi workgroup berkemampuan SPARK, lihat. Membuat workgroup berkemampuan Spark di Athena Setelah Anda membuat grup kerja, mesin analitiknya dapat ditingkatkan (misalnya, dari mesin Athena versi 2 ke mesin Athena versi 3), tetapi jenis mesinnya tidak dapat diubah. Misalnya, workgroup mesin Athena versi 3 tidak dapat diubah menjadi workgroup PySpark engine versi 3.
Perbarui mesin kueri	Pilih bagaimana Anda ingin memperbarui workgroup Anda ketika versi mesin Athena baru dirilis. Anda dapat membiarkan Athena memutuskan kapan harus memperbarui workgroup Anda atau secara manual memilih versi mesin. Untuk informasi selengkapnya, lihat Pembuatan versi mesin Athena.
Modus otentikasi	Pilih AWS Identity and Access Management (IAM) untuk menggunakan autentikasi IAM atau federasi untuk workgroup. Pilih Pusat Identitas IAM jika Anda ingin mendukung identitas tenaga kerja seperti pengguna dan grup dari penyedia identitas SAMB 2.0 seperti Microsoft Active Directory. Untuk informasi selengkapnya, lihat Propagasi identitas tepercaya di seluruh aplikasi di Panduan AWS IAM Identity Center Pengguna.
Peran layanan untuk akses Pusat Identitas IAM	Athena memerlukan izin IAM untuk mengakses Pusat Identitas IAM atas nama Anda. Untuk informasi selengkapnya tentang peran layanan IAM, lihat Membuat peran untuk mendelegasikan izin ke AWS layanan di Panduan Pengguna IAM.

Bidang	Deskripsi
Lokasi hasil kueri	Tidak wajib. Masukkan jalur ke bucket atau awalan Amazon S3. Bucket dan awalan ini harus ada sebelum Anda dapat menentuka nnya. **Note** Jika Anda menjalankan kueri di konsol, menentukan lokasi hasil kueri adalah opsional. Jika Anda tidak menentukannya untuk grup kerja atau di Pengaturan, Athena menggunak an lokasi hasil kueri default. Jika Anda menjalankan kueri dengan API atau driver, Anda harus menentukan lokasi hasil kueri di setidaknya satu dari dua tempat: untuk kueri individual dengan OutputLocation, atau untuk grup kerja, dengan. WorkGroupConfiguration
Pemilik ember yang diharapkan	Tidak wajib. Masukkan ID Akun AWS yang Anda harapkan sebagai pemilik bucket lokasi keluaran. Ini adalah langkah keamanan tambahan. Jika ID akun pemilik bucket tidak cocok dengan ID yang Anda tentukan di sini, upaya untuk menampilkan ke bucket akan gagal. Untuk informasi lebih lanjut, lihat Memverifikasi kepemilik an bucket dengan kondisi pemilik bucket di Panduan Pengguna Amazon S3. **Note** Setelan pemilik bucket yang diharapkan hanya berlaku untuk lokasi keluaran Amazon S3 yang Anda tentukan untuk hasil kueri Athena. Ini tidak berlaku untuk lokasi Amazon S3 lainnya seperti lokasi sumber data di bucket Amazon S3 eksternal`CTAS`, `INSERT INTO` dan lokasi tabel tujuan, lokasi keluaran pernyataan`UNLOAD`, operasi untuk menumpahkan bucket untuk kueri federasi, `SELECT` atau kueri yang dijalankan terhadap tabel di akun lain.

Bidang	Deskripsi
Tetapkan kontrol penuh pemilik bucket atas hasil kueri	Bidang ini tidak dipilih secara default. Jika Anda memilihnya dan ACL diaktifkan untuk bucket lokasi hasil kueri, Anda memberikan akses kontrol penuh atas hasil kueri kepada pemilik bucket. Misalnya, jika lokasi hasil kueri Anda dimiliki oleh akun lain, Anda dapat menggunakan opsi ini untuk memberikan kepemilikan dan kontrol penuh atas hasil kueri Anda ke akun lain. Jika setelan Kepemilikan Objek S3 bucket lebih disukai pemilik Bucket, pemilik bucket juga memiliki semua objek hasil kueri yang ditulis dari workgroup ini. Misalnya, jika grup kerja akun eksternal mengaktifkan opsi ini dan menetapkan lokasi hasil kueri ke bucket Amazon S3 akun Anda yang memiliki pengaturan Kepemilikan Objek S3 dari pemilik Bucket yang lebih disukai, Anda memiliki dan memiliki akses kontrol penuh atas hasil kueri grup kerja eksternal. Memilih opsi ini ketika setelan Kepemilikan Objek S3 bucket hasil kueri diberlakukan oleh pemilik Bucket tidak berpengaruh. Untuk informasi selengkapnya, lihat Setelan kepemilikan objek di Panduan Pengguna Amazon S3.
Enkripsi hasil kueri	Tidak wajib. Enkripsi hasil yang disimpan di Amazon S3. Jika dipilih, semua kueri di workgroup dienkripsi. Jika dipilih, Anda dapat memilih jenis Enkripsi, kunci Enkripsi dan masukkan ARN Kunci KMS. Jika Anda tidak memiliki kunci, buka AWS KMS konsol untuk membuatnya. Untuk informasi selengkapnya, lihat Membuat kunci di Panduan AWS Key Management Service Pengembang.

Bidang	Deskripsi
Setel ***encryption_type*** sebagai enkripsi minimum	Tidak wajib. Pilih opsi ini untuk menerapkan jenis enkripsi minimum untuk hasil kueri bagi semua pengguna workgroup. Memilih opsi ini menunjukkan tabel dengan hierarki jenis enkripsi. Tabel ini juga menunjukkan kepada Anda jenis enkripsi mana yang akan diizinkan untuk digunakan oleh pengguna workgroup saat Anda menentukan jenis enkripsi tertentu sebagai minimum. Untuk menggunakan opsi ini, pengaturan Override sisi klien tidak boleh dipilih. Untuk informasi selengkapnya, lihat Mengkonfigurasi enkripsi minimum untuk grup kerja.
Aktifkan Hibah Akses S3	Bidang ini dipilih secara default ketika Anda memilih Pusat Identitas IAM sebagai mode otentikasi. Saat dipilih, opsi ini menerapkan izin berbasis pengguna atau grup IAM Identity Center ke lokasi Amazon S3.
Buat awalan S3 berbasis identitas pengguna	Saat opsi ini dipilih, Athena membuat awalan Amazon S3 saat menyimpan hasil kueri. Awalan didasarkan pada identitas pengguna IAM Identity Center pengguna.
Publikasikan metrik kueri ke CloudWatch	Bidang ini dipilih secara default. Publikasikan metrik kueri ke CloudWatch. Lihat Memantau kueri Athena dengan metrik CloudWatch .
Ganti pengaturan sisi klien	Bidang ini tidak dipilih secara default. Jika Anda memilihnya, pengaturan workgroup berlaku untuk semua kueri di workgroup dan mengganti setelan sisi klien. Untuk informasi selengkapnya, lihat Pengaturan Workgroup mengesampingkan setelan sisi klien.

Bidang	Deskripsi
Pemohon Membayar ember S3	Tidak wajib. Pilih Aktifkan kueri pada bucket pembayaran pemohon di Amazon S3 jika pengguna workgroup akan menjalankan kueri pada data yang disimpan di bucket Amazon S3 yang dikonfigurasi sebagai Requester Pays. Akun pengguna yang menjalankan kueri dikenakan biaya untuk akses data yang berlaku dan biaya transfer data yang terkait dengan kueri. Untuk informasi selengkapnya, lihat Bucket Peminta Membayar di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.
Mengelola kontrol penggunaan data per kueri	Tidak wajib. Menetapkan batas untuk jumlah maksimum data yang diizinkan untuk dipindai oleh kueri. Anda hanya dapat menetapkan satu per batas kueri untuk grup kerja. Batas berlaku untuk semua kueri di workgroup dan jika kueri melebihi batas, itu akan dibatalkan. Untuk informasi selengkapnya, lihat Mengatur batas kontrol penggunaan data.
Peringatan penggunaan data kelompok kerja	Tidak wajib. Tetapkan beberapa ambang peringatan saat kueri yang berjalan di grup kerja ini memindai jumlah data tertentu dalam periode tertentu. Peringatan diimplementasikan menggunakan CloudWatch alarm Amazon dan berlaku untuk semua kueri di workgroup. Untuk informasi selengkapnya, lihat Menggunakan CloudWatch alarm Amazon di Panduan CloudWatch Pengguna Amazon.
Tanda	Tidak wajib. Tambahkan satu atau beberapa tag ke workgroup. Tag adalah label yang Anda tetapkan ke sumber daya workgroup Athena. Ini terdiri dari kunci dan nilai. Gunakan praktik terbaik AWS penandaan untuk membuat kumpulan tag yang konsisten dan mengkategorikan kelompok kerja berdasarkan tujuan, pemilik, atau lingkungan. Anda juga dapat menggunakan tag dalam kebijakan IAM, dan untuk mengontrol biaya penagihan. Jangan gunakan kunci tag duplikat workgroup yang sama. Untuk informasi selengkapnya, lihat the section called “Penandaan sumber daya ”.

5. Pilih Buat grup kerja. Workgroup muncul dalam daftar di halaman Workgroups.

Anda juga dapat menggunakan operasi CreateWorkGroupAPI untuk membuat workgroup.

> **Important**
>
> Setelah Anda membuat grup kerja, buat Kebijakan IAM untuk mengakses workgroup IAM yang memungkinkan Anda menjalankan tindakan terkait workgroup.

Mengedit grup kerja

Mengedit workgroup memerlukan izin untuk operasi `UpdateWorkgroup` API. Lihat Akses ke grup kerja dan tag dan Kebijakan IAM untuk mengakses workgroup. Jika Anda menambahkan atau mengedit tag, Anda juga harus memiliki izin untuk`TagResource`. Lihat Contoh kebijakan tag untuk kelompok kerja.

Untuk mengedit workgroup di konsol

1. Di panel navigasi konsol Athena, pilih Workgroups.
2. Pada halaman Workgroups, pilih tombol untuk workgroup yang ingin Anda edit.
3. Pilih Tindakan, Edit.
4. Ubah bidang sesuai kebutuhan. Untuk daftar bidang, lihat Membuat workgroup. Anda dapat mengubah semua bidang kecuali nama workgroup. Jika Anda perlu mengubah nama, buat workgroup lain dengan nama baru dan pengaturan yang sama.
5. Pilih Simpan perubahan. Workgroup yang diperbarui muncul dalam daftar di halaman Workgroups.

Lihat detail workgroup

Untuk setiap workgroup, Anda dapat melihat detailnya. Detailnya mencakup nama grup kerja, deskripsi, apakah itu diaktifkan atau dinonaktifkan, dan setelan yang digunakan untuk kueri yang berjalan di grup kerja, yang mencakup lokasi hasil kueri, pemilik bucket yang diharapkan, enkripsi, dan kontrol objek yang ditulis ke bucket hasil kueri. Jika workgroup memiliki batas penggunaan data, mereka juga ditampilkan.

Untuk melihat detail workgroup

1. Di panel navigasi konsol Athena, pilih Workgroups.

2. Pada halaman Workgroups, pilih link workgroup yang ingin Anda lihat. Halaman Rincian Ikhtisar untuk tampilan workgroup.

Menghapus grup kerja

Anda dapat menghapus workgroup jika Anda memiliki izin untuk melakukannya. Workgroup utama tidak dapat dihapus.

Jika Anda memiliki izin, Anda dapat menghapus workgroup kosong kapan saja. Anda juga dapat menghapus workgroup yang berisi kueri tersimpan. Dalam hal ini, sebelum melanjutkan untuk menghapus workgroup, Athena memperingatkan Anda bahwa kueri yang disimpan akan dihapus.

Jika Anda menghapus workgroup saat Anda berada di dalamnya, konsol akan mengalihkan fokus ke workgroup utama. Jika Anda memiliki akses ke sana, Anda dapat menjalankan kueri dan melihat pengaturannya.

Jika Anda menghapus grup kerja, pengaturan dan kontrol batas data per kueri akan dihapus. Kontrol batas data seluruh grup kerja tetap ada CloudWatch, dan Anda dapat menghapusnya di sana jika diperlukan.

> **Important**
>
> Sebelum menghapus workgroup, pastikan bahwa penggunanya juga termasuk dalam kelompok kerja lain di mana mereka dapat terus menjalankan kueri. Jika kebijakan IAM pengguna mengizinkan mereka menjalankan kueri hanya di grup kerja ini, dan Anda menghapusnya, mereka tidak lagi memiliki izin untuk menjalankan kueri. Untuk informasi selengkapnya, lihat Example policy for running queries in the primary workgroup.

Untuk menghapus workgroup di konsol

1. Di panel navigasi konsol Athena, pilih Workgroups.
2. Pada halaman Workgroups, pilih tombol untuk workgroup yang ingin Anda hapus.
3. Pilih Tindakan, Hapus.
4. Pada prompt konfirmasi Delete workgroup, masukkan nama workgroup, lalu pilih Hapus.

Untuk menghapus workgroup dengan operasi API, gunakan `DeleteWorkGroup` tindakan.

Beralih kelompok kerja

Anda dapat beralih dari satu grup kerja ke grup lain jika Anda memiliki izin untuk keduanya.

Anda dapat membuka hingga sepuluh tab kueri dalam setiap workgroup. Saat Anda beralih di antara grup kerja, tab kueri Anda tetap terbuka hingga tiga grup kerja.

Untuk beralih kelompok kerja

1. Di konsol Athena, gunakan opsi Workgroup di kanan atas untuk memilih workgroup.
2. Jika kotak dialog pengaturan ***nama grup kerja Workgroup*** muncul, pilih Akui.

Opsi Workgroup menunjukkan nama workgroup yang Anda alihkan. Anda sekarang dapat menjalankan kueri di workgroup ini.

Salin kueri yang disimpan di antara kelompok kerja

Saat ini, konsol Athena tidak memiliki opsi untuk menyalin kueri yang disimpan dari satu grup kerja ke grup kerja lainnya secara langsung, tetapi Anda dapat melakukan tugas yang sama secara manual dengan menggunakan prosedur berikut.

Untuk menyalin kueri yang disimpan di antara kelompok kerja

1. Di konsol Athena, dari workgroup tempat Anda ingin menyalin kueri, pilih tab Kueri tersimpan.
2. Pilih tautan kueri tersimpan yang ingin Anda salin. Athena membuka kueri di editor kueri.
3. Di editor kueri, pilih teks kueri, lalu tekan **`Ctrl+C`** untuk menyalinnya.
4. Beralih ke workgroup tujuan, atau buat workgroup, lalu beralih ke workgroup tujuan.
5. Buka tab baru di editor kueri, lalu tekan **`Ctrl+V`** untuk menempelkan teks ke tab baru.
6. Di editor kueri, pilih Simpan sebagai untuk menyimpan kueri di workgroup tujuan.
7. Dalam kotak dialog Pilih nama, masukkan nama untuk kueri dan deskripsi opsional.
8. Pilih Simpan.

Mengaktifkan dan menonaktifkan workgroup

Jika Anda memiliki izin untuk melakukannya, Anda dapat mengaktifkan atau menonaktifkan grup kerja di konsol, dengan menggunakan operasi API, atau dengan driver JDBC dan ODBC.

Untuk mengaktifkan atau menonaktifkan workgroup

1. Di panel navigasi konsol Athena, pilih Workgroups.
2. Pada halaman Workgroups, pilih link untuk workgroup.
3. Di kanan atas, pilih Aktifkan grup kerja atau Nonaktifkan grup kerja.
4. Pada prompt konfirmasi, pilih Aktifkan atau Nonaktifkan. Jika Anda menonaktifkan workgroup, penggunanya tidak dapat menjalankan kueri di dalamnya, atau membuat kueri bernama baru. Jika Anda mengaktifkan workgroup, pengguna dapat menggunakannya untuk menjalankan kueri.

Tentukan workgroup untuk menjalankan kueri

Untuk menentukan workgroup yang akan digunakan, Anda harus memiliki izin ke workgroup.

Untuk menentukan workgroup yang akan digunakan

1. Pastikan izin Anda memungkinkan Anda menjalankan kueri di workgroup yang ingin Anda gunakan. Untuk informasi selengkapnya, lihat the section called “ Kebijakan IAM untuk mengakses workgroup”.
2. Untuk menentukan workgroup, gunakan salah satu opsi ini:
   - Jika Anda menggunakan konsol Athena, atur workgroup dengan beralih workgroup.
   - Jika Anda menggunakan operasi Athena API, tentukan nama workgroup dalam tindakan API. Misalnya, Anda dapat mengatur nama workgroup StartQueryExecution, sebagai berikut:

```
StartQueryExecutionRequest startQueryExecutionRequest = new
 StartQueryExecutionRequest()
              .withQueryString(ExampleConstants.ATHENA_SAMPLE_QUERY)
              .withQueryExecutionContext(queryExecutionContext)
              .withWorkGroup(WorkgroupName)
```

   - Jika Anda menggunakan driver JDBC atau ODBC, atur nama workgroup dalam string koneksi menggunakan parameter konfigurasi. `Workgroup` Sopir melewati nama grup kerja untuk Athena. Tentukan parameter workgroup dalam string koneksi seperti pada contoh berikut:

```
jdbc:awsathena://AwsRegion=<AWSREGION>;UID=<ACCESSKEY>;
PWD=<SECRETKEY>;S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/<athena-
output>-<AWSREGION>/;
Workgroup=<WORKGROUPNAME>;
```

Mengkonfigurasi enkripsi minimum untuk grup kerja

Sebagai administrator workgroup Athena SQL, Anda dapat menerapkan tingkat enkripsi minimal di Amazon S3 untuk semua hasil kueri dari workgroup. Anda dapat menggunakan fitur ini untuk memastikan bahwa hasil kueri tidak pernah disimpan dalam bucket Amazon S3 dalam status tidak terenkripsi.

Ketika pengguna dalam grup kerja dengan enkripsi minimum diaktifkan mengirimkan kueri, mereka hanya dapat mengatur enkripsi ke tingkat minimum yang Anda konfigurasikan, atau ke tingkat yang lebih tinggi jika tersedia. Athena mengenkripsi hasil kueri pada tingkat yang ditentukan saat pengguna menjalankan kueri atau pada tingkat yang ditetapkan dalam workgroup.

Level berikut tersedia:

- Dasar - Enkripsi sisi server Amazon S3 dengan kunci terkelola Amazon S3 (SSE_S3).
- Menengah - Enkripsi Sisi Server dengan kunci terkelola KMS (SSE_KMS).
- Advanced - Enkripsi sisi klien dengan kunci terkelola KMS (CSE_KMS).

Pertimbangan dan batasan

- Fitur enkripsi minimum tidak tersedia untuk grup kerja yang diaktifkan Apache Spark.
- Fitur enkripsi minimum hanya berfungsi jika workgroup tidak mengaktifkan opsi Override setelan sisi klien.
- Jika workgroup mengaktifkan opsi Override setelan sisi klien, setelan enkripsi workgroup akan berlaku, dan setelan enkripsi minimum tidak berpengaruh.
- Tidak ada biaya untuk mengaktifkan fitur ini.

Mengaktifkan enkripsi minimum untuk grup kerja

Anda dapat mengaktifkan tingkat enkripsi minimum untuk hasil kueri dari workgroup Athena SQL saat Anda membuat atau memperbarui workgroup. Untuk melakukan ini, Anda dapat menggunakan konsol Athena, Athena API, atau. AWS CLI

Menggunakan konsol Athena untuk mengaktifkan enkripsi minimum

Untuk mulai membuat atau mengedit grup kerja menggunakan konsol Athena, lihat Membuat grup kerja atau Mengedit grup kerja. Saat mengonfigurasi workgroup Anda, gunakan langkah-langkah berikut untuk mengaktifkan enkripsi minimum.

Untuk mengonfigurasi tingkat enkripsi minimum untuk hasil kueri grup kerja

1. Di bagian Konfigurasi tambahan, perluas Pengaturan.
2. Hapus opsi Ganti pengaturan sisi klien, atau verifikasi bahwa itu tidak dipilih.
3. Di bagian Konfigurasi tambahan, perluas konfigurasi hasil Kueri.
4. Pilih opsi Enkripsi hasil kueri.
5. Untuk jenis Enkripsi, pilih metode enkripsi yang ingin Athena gunakan untuk hasil kueri grup kerja Anda (SSE_S3, SSE_KMS, atau CSE_KMS). Jenis enkripsi ini sesuai dengan tingkat keamanan dasar, menengah, dan lanjutan.
6. Untuk menerapkan metode enkripsi yang Anda pilih sebagai tingkat enkripsi minimum untuk semua pengguna, pilih Setel ***encryption_method sebagai enkripsi*** minimum.

   Saat Anda memilih opsi ini, tabel menunjukkan hierarki enkripsi dan tingkat enkripsi yang akan diizinkan pengguna saat jenis enkripsi yang Anda pilih menjadi minimum.
7. Setelah Anda membuat workgroup atau memperbarui konfigurasi workgroup Anda, pilih Buat workgroup atau Simpan perubahan.

Menggunakan Athena API atau AWS CLI untuk mengaktifkan enkripsi minimum

Bila Anda menggunakan UpdateWorkGroupAPI CreateWorkGroupatau untuk membuat atau memperbarui workgroup Athena SQL, atur EnforceWorkGroupConfigurationke, EnableMinimumEncryptionConfigurationke `falsetrue`, dan gunakan EncryptionOptionuntuk menentukan jenis enkripsi.

Di AWS CLI, gunakan `update-work-group`perintah `create-work-group`or dengan `--configuration-updates` parameter `--configuration` atau dan tentukan opsi yang sesuai dengan yang ada untuk API.

Menggunakan IAM Identity Center mengaktifkan kelompok kerja Athena

Fitur propagasi identitas tepercaya AWS IAM Identity Center memungkinkan identitas tenaga kerja Anda digunakan di seluruh layanan analitik. AWS Propagasi identitas tepercaya menyelamatkan Anda dari keharusan melakukan konfigurasi penyedia identitas khusus layanan atau pengaturan peran IAM.

Dengan IAM Identity Center, Anda dapat mengelola keamanan masuk untuk identitas tenaga kerja Anda, juga dikenal sebagai pengguna tenaga kerja. IAM Identity Center menyediakan satu tempat

di mana Anda dapat membuat atau menghubungkan pengguna tenaga kerja dan mengelola akses mereka secara terpusat di semua AWS akun dan aplikasi mereka. Anda dapat menggunakan izin multi-akun untuk menetapkan akses pengguna ini. Akun AWS Anda dapat menggunakan penugasan aplikasi untuk menetapkan akses pengguna ke aplikasi yang diaktifkan IAM Identity Center, aplikasi cloud, dan aplikasi Customer Security Assertion Markup Language (SAMB 2.0). Untuk informasi selengkapnya, lihat Propagasi identitas tepercaya di seluruh aplikasi di Panduan AWS IAM Identity Center Pengguna.

Saat ini, dukungan Athena SQL untuk propagasi identitas tepercaya memungkinkan Anda menggunakan identitas yang sama untuk Amazon EMR Studio dan antarmuka Athena SQL di EMR Studio. Untuk menggunakan identitas Pusat Identitas IAM dengan Athena SQL di EMR Studio, Anda harus membuat grup kerja yang diaktifkan Pusat Identitas IAM di Athena. Anda kemudian dapat menggunakan konsol Pusat Identitas IAM atau API untuk menetapkan pengguna atau grup Pusat Identitas IAM ke grup kerja Athena yang diaktifkan Pusat Identitas IAM. Kueri dari workgroup Athena yang menggunakan propagasi identitas tepercaya harus dijalankan dari antarmuka Athena SQL di EMR Studio yang mengaktifkan IAM Identity Center.

Pertimbangan dan batasan

Saat Anda menggunakan propagasi identitas tepercaya dengan Amazon Athena, pertimbangkan hal-hal berikut:

- Anda tidak dapat mengubah metode otentikasi untuk workgroup setelah workgroup dibuat.
  - Workgroup Athena SQL yang ada tidak dapat dimodifikasi untuk mendukung grup kerja yang diaktifkan IAM Identity Center.
  - Grup kerja yang diaktifkan IAM Identity Center tidak dapat dimodifikasi untuk mendukung izin IAM tingkat sumber daya atau kebijakan IAM berbasis identitas.
- Untuk mengakses grup kerja yang diaktifkan propagasi identitas tepercaya, pengguna IAM Identity Center harus ditetapkan ke `IdentityCenterApplicationArn` yang dikembalikan oleh respons tindakan API Athena. GetWorkGroup
- Hibah Akses Amazon S3 harus dikonfigurasi untuk menggunakan identitas propagasi identitas tepercaya. Untuk informasi selengkapnya, lihat Hibah Akses S3 dan identitas direktori perusahaan di Panduan Pengguna Amazon S3.
- Kelompok kerja Athena yang mengaktifkan Pusat Identitas IAM mengharuskan Lake Formation dikonfigurasi untuk menggunakan identitas Pusat Identitas IAM. Untuk informasi konfigurasi, lihat Mengintegrasikan Pusat Identitas IAM di Panduan AWS Lake Formation Pengembang.

- Secara default, waktu kueri habis setelah 30 menit di grup kerja yang menggunakan propagasi identitas tepercaya. Anda dapat meminta peningkatan batas waktu kueri, tetapi maksimum kueri yang dapat dijalankan di grup kerja propagasi identitas tepercaya adalah satu jam.
- Perubahan hak pengguna atau grup dalam kelompok kerja propagasi identitas tepercaya dapat memerlukan waktu hingga satu jam untuk diterapkan.
- Kueri di workgroup Athena yang menggunakan propagasi identitas tepercaya tidak dapat dijalankan langsung dari konsol Athena. Mereka harus dijalankan dari antarmuka Athena di EMR Studio yang mengaktifkan IAM Identity Center. Untuk informasi selengkapnya tentang penggunaan Athena di EMR Studio, lihat Menggunakan editor SQL Amazon Athena di EMR Studio di Panduan Manajemen EMR Amazon.
- Perbanyakan identitas tepercaya tidak kompatibel dengan fitur Athena berikut.
  - `aws:CalledVia`kunci konteks.
  - Athena untuk kelompok kerja Spark.
  - Akses federasi ke Athena API.
  - Akses federasi ke Athena menggunakan Lake Formation dan Athena JDBC dan ODBC driver.
- Anda dapat menggunakan propagasi identitas tepercaya dengan Athena hanya dalam hal berikut: Wilayah AWS
  - `us-east-2`— AS Timur (Ohio)
  - `us-east-1`- AS Timur (Virginia N.)
  - `us-west-1`— AS Barat (California N.)
  - `us-west-2`— AS Barat (Oregon)
  - `af-south-1`— Afrika (Cape Town)
  - `ap-east-1`— Asia Pasifik (Hong Kong)
  - `ap-southeast-3`— Asia Pasifik (Jakarta)
  - `ap-south-1`— Asia Pasifik (Mumbai)
  - `ap-northeast-3`— Asia Pasifik (Osaka)
  - `ap-northeast-2`- Asia Pasifik (Seoul)
  - `ap-southeast-1`— Asia Pasifik (Singapura)
  - `ap-southeast-2`— Asia Pasifik (Sydney)
  - `ap-northeast-1`— Asia Pasifik (Tokyo)
  - `ca-central-1`— Kanada (Tengah)
  - `eu-central-1`— Eropa (Frankfurt)

- `eu-west-1`— Eropa (Irlandia)
- `eu-west-2`— Eropa (London)
- `eu-south-1`— Eropa (Milan)
- `eu-west-3`- Eropa (Paris)
- `eu-north-1`— Eropa (Stockholm)
- `me-south-1`- Timur Tengah (Bahrain)
- `sa-east-1`— Amerika Selatan (São Paulo)

Izin yang diperlukan

Pengguna IAM dari admin yang membuat grup kerja yang diaktifkan Pusat Identitas IAM di konsol Athena harus memiliki kebijakan berikut yang dilampirkan.

- Kebijakan yang `AmazonAthenaFullAccess` dikelola. Lihat perinciannya di AWS kebijakan terkelola: AmazonAthenaFullAccess.
- Kebijakan inline berikut yang memungkinkan tindakan IAM dan IAM Identity Center:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:createRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy",
                "iam:ListRoles",
                "iam:PassRole",
                "identitystore:ListUsers",
                "identitystore:ListGroups",
                "identitystore:CreateUser",
                "identitystore:CreateGroup",
                "sso:ListInstances",
                "sso:CreateInstance",
                "sso:DeleteInstance",
                "sso:DescribeUser",
                "sso:DescribeGroup",
                "sso:ListTrustedTokenIssuers",
                "sso:DescribeTrustedTokenIssuer",
                "sso:ListApplicationAssignments",
```

```
                "sso:DescribeRegisteredRegions",
                "sso:GetManagedApplicationInstance",
                "sso:GetSharedSsoConfiguration",
                "sso:PutApplicationAssignmentConfiguration",
                "sso:CreateApplication",
                "sso:DeleteApplication",
                "sso:PutApplicationGrant",
                "sso:PutApplicationAuthenticationMethod",
                "sso:PutApplicationAccessScope",
                "sso:ListDirectoryAssociations",
                "sso:CreateApplicationAssignment",
                "sso:DeleteApplicationAssignment",
                "organizations:ListDelegatedAdministrators",
                "organizations:DescribeAccount",
                "organizations:DescribeOrganization",
                "organizations:CreateOrganization",
                "sso-directory:SearchUsers",
                "sso-directory:SearchGroups",
                "sso-directory:CreateUser"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Membuat Pusat Identitas IAM mengaktifkan workgroup Athena

Prosedur berikut menunjukkan langkah-langkah dan opsi yang terkait dengan membuat grup kerja Athena yang diaktifkan Pusat Identitas IAM. Untuk deskripsi opsi konfigurasi lain yang tersedia untuk kelompok kerja Athena, lihat. Buat grup kerja

Untuk membuat workgroup yang diaktifkan SSO di konsol Athena

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Di panel navigasi konsol Athena, pilih Workgroups.
3. Pada Grup Kerja, pilih Buat grup kerja.
4. Pada halaman Buat workgroup, untuk nama Workgroup, masukkan nama untuk workgroup.
5. Untuk mesin Analytics, gunakan default Athena SQL.

6. Untuk Otentikasi, pilih Pusat Identitas IAM.
7. Untuk peran Layanan untuk akses Pusat Identitas IAM, pilih peran layanan yang ada, atau buat yang baru.

   Athena memerlukan izin untuk mengakses Pusat Identitas IAM untuk Anda. Peran layanan diperlukan bagi Athena untuk melakukan ini. Peran layanan adalah peran IAM yang Anda kelola yang mengizinkan AWS layanan untuk mengakses AWS layanan lain atas nama Anda. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke AWS layanan di Panduan Pengguna IAM.
8. Perluas konfigurasi hasil Kueri, lalu masukkan atau pilih jalur Amazon S3 untuk Lokasi hasil kueri.
9. (Opsional) Pilih Enkripsi hasil kueri.
10. (Opsional) Pilih Buat awalan S3 berbasis identitas pengguna.

    Saat Anda membuat grup kerja yang diaktifkan Pusat Identitas IAM, opsi Aktifkan Hibah Akses S3 dipilih secara default. Anda dapat menggunakan Amazon S3 Access Grants untuk mengontrol akses ke lokasi hasil kueri Athena (awalan) di Amazon S3. Untuk informasi selengkapnya tentang Hibah Akses Amazon S3, lihat Mengelola akses dengan Hibah Akses Amazon S3.

    Di grup kerja Athena yang menggunakan autentikasi Pusat Identitas IAM, Anda dapat mengaktifkan pembuatan lokasi hasil kueri berbasis identitas yang diatur oleh Hibah Akses Amazon S3. Awalan Amazon S3 berbasis identitas pengguna ini memungkinkan pengguna di workgroup Athena menjaga hasil kueri mereka terisolasi dari pengguna lain di workgroup yang sama.

    Saat Anda mengaktifkan opsi awalan pengguna, Athena menambahkan ID pengguna sebagai awalan jalur Amazon S3 ke lokasi keluaran hasil kueri untuk grup kerja (misalnya,). `s3://DOC-EXAMPLE-BUCKET/${user_id}` Untuk menggunakan fitur ini, Anda harus mengkonfigurasi Hibah Akses untuk mengizinkan hanya izin pengguna ke lokasi yang memiliki `user_id` awalan. Untuk contoh kebijakan peran lokasi Amazon S3 Access Grants yang membatasi akses ke hasil kueri Athena, lihat. Contoh kebijakan peran

    > **Note**
    >
    > Memilih opsi awalan identitas pengguna S3 secara otomatis memungkinkan opsi penggantian pengaturan sisi klien untuk grup kerja, seperti yang dijelaskan pada langkah

berikutnya. Opsi override setelan sisi klien adalah persyaratan untuk fitur awalan identitas pengguna.

11. Perluas Pengaturan, lalu konfirmasikan bahwa Override pengaturan sisi klien dipilih.

    Saat Anda memilih Ganti setelan sisi klien, pengaturan workgroup diberlakukan di tingkat workgroup untuk semua klien di workgroup. Untuk informasi selengkapnya, lihat Pengaturan Workgroup mengesampingkan setelan sisi klien.
12. (Opsional) Buat pengaturan konfigurasi lain yang Anda perlukan seperti yang dijelaskan dalamBuat grup kerja.
13. Pilih Buat grup kerja.
14. Gunakan bagian Workgroups pada konsol Athena untuk menetapkan pengguna atau grup dari direktori Pusat Identitas IAM Anda ke grup kerja Athena yang diaktifkan Pusat Identitas IAM Anda.

Contoh kebijakan peran

Contoh berikut menunjukkan kebijakan untuk peran yang akan dilampirkan ke lokasi Amazon S3 Access Grant yang membatasi akses ke hasil kueri Athena.

```
{
    "Statement": [{
        "Action": ["s3:*"],
        "Condition": {
            "ArnNotEquals": {
                "s3:AccessGrantsInstanceArn": "arn:aws:s3:${region}:${account}:access-
grants/default"
            },
            "StringNotEquals": {
                "aws:ResourceAccount": "${account}"
            }
        },
        "Effect": "Deny",
        "Resource": "*",
        "Sid": "ExplicitDenyS3"
    }, {
        "Action": ["kms:*"],
        "Effect": "Deny",
        "NotResource": "arn:aws:kms:${region}:${account}:key/${keyid}",
        "Sid": "ExplictDenyKMS"
```

```
    }, {
        "Action": ["s3:ListMultipartUploadParts", "s3:GetObject"],
        "Condition": {
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": "arn:aws:s3:${region}:${account}:access-
grants/default"
            },
            "StringEquals": {
                "aws:ResourceAccount": "${account}"
            }
        },
        "Effect": "Allow",
        "Resource": "arn:aws:s3:::ATHENA-QUERY-RESULT-LOCATION/${identitystore:UserId}/
*",
        "Sid": "ObjectLevelReadPermissions"
    }, {
        "Action": ["s3:PutObject", "s3:AbortMultipartUpload"],
        "Condition": {
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": "arn:aws:s3:${region}:${account}:access-
grants/default"
            },
            "StringEquals": {
                "aws:ResourceAccount": "${account}"
            }
        },
        "Effect": "Allow",
        "Resource": "arn:aws:s3:::ATHENA-QUERY-RESULT-LOCATION/${identitystore:UserId}/
*",
        "Sid": "ObjectLevelWritePermissions"
    }, {
        "Action": "s3:ListBucket",
        "Condition": {
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": "arn:aws:s3:${region}:${account}:access-
grants/default"
            },
            "StringEquals": {
                "aws:ResourceAccount": "${account}"
            },
            "StringLikeIfExists": {
                "s3:prefix": ["${identitystore:UserId}", "${identitystore:UserId}/*"]
            }
        },
```

```
        "Effect": "Allow",
        "Resource": "arn:aws:s3:::ATHENA-QUERY-RESULT-LOCATION",
        "Sid": "BucketLevelReadPermissions"
    }, {
        "Action": ["kms:GenerateDataKey", "kms:Decrypt"],
        "Effect": "Allow",
        "Resource": "arn:aws:kms:${region}:${account}:key/${keyid}",
        "Sid": "KMSPermissions"
    }],
    "Version": "2012-10-17"
}
```

API grup kerja Athena

Berikut ini adalah beberapa operasi REST API yang digunakan untuk workgroup Athena. Dalam semua operasi berikut kecuali untuk`ListWorkGroups`, Anda harus menentukan workgroup. Dalam operasi lain, seperti`StartQueryExecution`, parameter workgroup adalah opsional dan operasi tidak tercantum di sini. Untuk daftar lengkap operasi, lihat Referensi API Amazon Athena.

- CreateWorkGroup
- DeleteWorkGroup
- GetWorkGroup
- ListWorkGroups
- UpdateWorkGroup

Pemecahan masalah kelompok kerja

Gunakan tips berikut untuk memecahkan masalah kelompok kerja.

- Periksa izin untuk pengguna individu di akun Anda. Mereka harus memiliki akses ke lokasi untuk hasil kueri, dan ke workgroup tempat mereka ingin menjalankan kueri. Jika mereka ingin beralih kelompok kerja, mereka juga memerlukan izin untuk kedua kelompok kerja. Untuk informasi, lihat Kebijakan IAM untuk mengakses workgroup.
- Perhatikan konteks di konsol Athena, untuk melihat di workgroup mana Anda akan menjalankan kueri. Jika Anda menggunakan driver, pastikan untuk mengatur workgroup ke yang Anda butuhkan. Untuk informasi, lihat the section called “Tentukan workgroup untuk menjalankan kueri”.
- Jika Anda menggunakan API atau driver untuk menjalankan kueri, Anda harus menentukan lokasi hasil kueri menggunakan salah satu cara berikut: untuk kueri individual, gunakan

OutputLocation(sisi klien). Di workgroup, gunakan WorkGroupConfiguration. Jika lokasi tidak ditentukan dengan cara apa pun, Athena mengeluarkan kesalahan saat runtime kueri.

- Jika Anda mengganti setelan sisi klien dengan pengaturan grup kerja, Anda mungkin mengalami kesalahan dengan lokasi hasil kueri. Misalnya, pengguna workgroup mungkin tidak memiliki izin ke lokasi workgroup di Amazon S3 untuk menyimpan hasil kueri. Dalam hal ini, tambahkan izin yang diperlukan.
- Workgroup memperkenalkan perubahan dalam perilaku operasi API. Panggilan ke operasi API yang ada berikut mengharuskan pengguna di akun Anda memiliki izin berbasis sumber daya di IAM ke grup kerja tempat mereka membuatnya. Jika tidak ada izin untuk kelompok kerja dan tindakan kelompok kerja, tindakan API berikut akan menampilkan`AccessDeniedException`: CreateNamedQuery,,,, DeleteNamedQuery, GetNamedQueryListNamedQueries, StartQueryExecution, StopQueryExecutionListQueryExecutionsGetQueryExecutionGetQueryResults, dan GetQueryResultsStream(tindakan API ini hanya tersedia untuk digunakan dengan driver dan tidak diekspos sebaliknya untuk penggunaan publik). Untuk informasi selengkapnya, lihat Kunci tindakan, sumber daya, dan kondisi untuk Amazon Athena di Referensi Otorisasi Layanan.

  Panggilan ke operasi BatchGetQueryExecutiondan BatchGetNamedQueryAPI hanya mengembalikan informasi tentang kueri yang berjalan di grup kerja yang dapat diakses pengguna. Jika pengguna tidak memiliki akses ke workgroup, operasi API ini mengembalikan ID kueri yang tidak sah sebagai bagian dari daftar ID yang belum diproses. Untuk informasi selengkapnya, lihat the section called “ API grup kerja Athena”.
- Jika workgroup tempat kueri akan dijalankan dikonfigurasi dengan lokasi hasil kueri yang dipaksakan, jangan tentukan `external_location` untuk kueri CTAS. Athena mengeluarkan kesalahan dan gagal kueri yang menentukan `external_location` dalam kasus ini. Misalnya, kueri ini gagal, jika Anda mengganti setelan sisi klien untuk lokasi hasil kueri, memaksa grup kerja untuk menggunakan lokasinya sendiri: `CREATE TABLE <DB>.<TABLE1> WITH (format='Parquet', external_location='s3://DOC-EXAMPLE-BUCKET/test/') AS SELECT * FROM <DB>.<TABLE2> LIMIT 10;`

Anda mungkin melihat kesalahan berikut. Tabel ini menyediakan daftar beberapa kesalahan yang terkait dengan kelompok kerja dan menyarankan solusi.

Kesalahan kelompok kerja

Kesalahan	Terjadi ketika...
status kueri DIBATALKAN. Batas pemindaian byte terlampaui.	Kueri mencapai batas data per kueri dan dibatalkan. Pertimbangkan untuk menulis ulang kueri sehingga membaca lebih sedikit data, atau hubungi administrator akun Anda.
Pengguna: arn:aws:iam: :12345678 9012:user/abc tidak berwenang untuk melakukan: athena: on resource: arn:aws:athena:us-east- 1:123456789012:workgroup/wo rkgroupname StartQueryExecution	Seorang pengguna menjalankan kueri dalam workgroup, tetapi tidak memiliki akses ke sana. Perbarui kebijakan Anda agar memiliki akses ke grup kerja.
INVALID_INPUT. WorkGroup <name>din onaktifkan.	Pengguna menjalankan kueri di workgroup , tetapi workgroup dinonaktifkan. Workgroup Anda dapat dinonaktifkan oleh administr ator Anda. Mungkin juga Anda tidak memiliki akses ke sana. Dalam kedua kasus tersebut, hubungi administrator yang memiliki akses untuk memodifikasi grup kerja.
INVALID_INPUT. WorkGroup <name>tidak ditemukan.	Pengguna menjalankan kueri di workgroup, tetapi workgroup tidak ada. Ini bisa terjadi jika workgroup dihapus. Beralih ke workgroup lain untuk menjalankan kueri Anda.
InvalidRequestException: saat memanggil StartQueryExecution operasi: Tidak ada lokasi output yang disediakan. Lokasi keluaran diperlukan baik melalui pengaturan konfigurasi hasil Workgroup atau sebagai input API.	Pengguna menjalankan kueri dengan API tanpa menentukan lokasi untuk hasil kueri. Anda harus mengatur lokasi keluaran untuk hasil kueri menggunakan salah satu dari dua cara: baik untuk kueri individual, menggunak an OutputLocation(sisi klien), atau dalam kelompok kerja, menggunakan. WorkGroup Configuration

Kesalahan	Terjadi ketika...
Kueri Create Table As Select gagal karena dikirimkan dengan properti 'external_location ' ke Athena Workgroup yang memberlakukan lokasi keluaran terpusat untuk semua kueri. Harap hapus properti 'external_location' dan kirimkan kembali kueri.	Jika workgroup tempat kueri berjalan dikonfigu rasi dengan lokasi hasil kueri yang diberlaku kan, dan Anda menentukan `external_ location` untuk kueri CTAS. Dalam hal ini, hapus `external_location` dan jalankan kembali kueri.
Tidak dapat membuat pernyataan siap *`prepared_statement_name`* . Jumlah pernyataan yang disiapkan dalam kelompok kerja ini melebihi batas 1000.	Workgroup berisi lebih dari batas 1000 pernyataan yang disiapkan. Untuk mengatasi masalah ini, gunakan DEALLOCATE PREPARE untuk menghapus satu atau beberapa pernyataan yang disiapkan dari workgroup. Atau, buat workgroup baru.

Mengontrol biaya dan memantau kueri dengan CloudWatch metrik dan peristiwa

Workgroup memungkinkan Anda menyetel batas kontrol penggunaan data per kueri atau per grup kerja, mengatur alarm saat batas tersebut terlampaui, dan memublikasikan metrik kueri ke. CloudWatch

Di setiap kelompok kerja, Anda dapat:

- Konfigurasikan kontrol penggunaan data per kueri dan per grup kerja, dan buat tindakan yang akan diambil jika kueri melanggar ambang batas.
- Melihat dan menganalisis metrik kueri, dan mempublikasikannya ke CloudWatch. Jika Anda membuat grup kerja di konsol, pengaturan untuk memublikasikan metrik CloudWatch dipilih untuk Anda. Jika Anda menggunakan operasi API, Anda harus mengaktifkan penerbitan metrik. Saat metrik diterbitkan, metrik ditampilkan di bawah tab Metrik di panel Workgroups. Metrik dinonaktifkan secara default untuk workgroup utama.

Video

Video berikut menunjukkan cara membuat dasbor khusus dan menyetel alarm dan pemicu pada metrik. CloudWatch Anda dapat menggunakan dasbor yang telah diisi sebelumnya langsung dari konsol Athena untuk menggunakan metrik kueri ini.

Memantau kueri Amazon Athena menggunakan Amazon CloudWatch

Topik

Mengaktifkan metrik CloudWatch kueri

Saat Anda membuat grup kerja di konsol, pengaturan untuk memublikasikan metrik kueri ke dipilih CloudWatch secara default.

Untuk mengaktifkan atau menonaktifkan metrik kueri di konsol Athena untuk grup kerja

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.

3. Di panel navigasi, pilih Workgroups.
4. Pilih tautan grup kerja yang ingin Anda ubah.
5. Pada halaman detail untuk workgroup, pilih Edit.
6. Di bagian Pengaturan, pilih atau hapus Publikasikan metrik kueri ke AWS CloudWatch.

Jika Anda menggunakan operasi API, antarmuka baris perintah, atau aplikasi klien dengan driver JDBC untuk membuat grup kerja, untuk mengaktifkan penerbitan metrik kueri, disetel ke dalam. `PublishCloudWatchMetricsEnabled` `true` WorkGroupConfiguration Contoh berikut hanya menampilkan konfigurasi metrik dan menghilangkan konfigurasi lainnya:

```
"WorkGroupConfiguration": {
      "PublishCloudWatchMetricsEnabled": "true"
     ....
     }
```

Memantau kueri Athena dengan metrik CloudWatch

Athena menerbitkan metrik terkait kueri ke CloudWatch Amazon, saat metrik kueri publikasi ke opsi dipilih. CloudWatch Anda dapat membuat dasbor khusus, menyetel alarm, dan pemicu pada metrik CloudWatch, atau menggunakan dasbor yang telah diisi sebelumnya langsung dari konsol Athena.

Saat Anda mengaktifkan metrik kueri untuk kueri di grup kerja, metrik akan ditampilkan dalam tab Metrik di panel Workgroups, untuk setiap grup kerja di konsol Athena.

Athena menerbitkan metrik berikut ke konsol: CloudWatch

- `DPUAllocated`— Jumlah total DPU (unit pemrosesan data) yang disediakan dalam reservasi kapasitas untuk menjalankan kueri.
- `DPUConsumed`— Jumlah DPU yang secara aktif dikonsumsi oleh kueri dalam suatu `RUNNING` negara pada waktu tertentu dalam reservasi. Metrik yang dipancarkan hanya jika kelompok kerja dikaitkan dengan reservasi kapasitas dan mencakup semua kelompok kerja yang terkait dengan reservasi.
- `DPUCount`— Jumlah maksimum DPU yang dikonsumsi oleh kueri Anda, diterbitkan tepat sekali saat kueri selesai.
- `EngineExecutionTime`— Jumlah milidetik yang dibutuhkan kueri untuk dijalankan.
- `ProcessedBytes`— Jumlah byte yang dipindai Athena per kueri DML.
- `QueryPlanningTime`— Jumlah milidetik yang Athena ambil untuk merencanakan alur pemrosesan kueri.
- `QueryQueueTime`— Jumlah milidetik bahwa kueri berada dalam antrian kueri menunggu sumber daya.
- `ServicePreProcessingTime`— Jumlah milidetik yang Athena ambil untuk memproses kueri sebelum mengirimkan kueri ke mesin kueri.

- `ServiceProcessingTime`— Jumlah milidetik yang Athena ambil untuk memproses hasil kueri setelah mesin kueri selesai menjalankan kueri.
- `TotalExecutionTime`— Jumlah milidetik yang Athena ambil untuk menjalankan kueri DDL atau DHTML.

Untuk deskripsi yang lebih lengkap, lihat Daftar CloudWatch metrik dan dimensi untuk Athena nanti dalam dokumen ini.

Metrik ini memiliki dimensi sebagai berikut:

- `CapacityReservation`— Nama reservasi kapasitas yang digunakan untuk menjalankan kueri, jika berlaku.
- `QueryState` – `SUCCEEDED`, `FAILED`, atau `CANCELED`
- `QueryType` – `DML`, `DDL`, atau `UTILITY`
- `WorkGroup`— nama kelompok kerja

Athena menerbitkan metrik berikut ke CloudWatch konsol di bawah namespace: `AmazonAthenaForApacheSpark`

- `DPUCount`— jumlah DPU yang dikonsumsi selama sesi untuk mengeksekusi perhitungan.

Metrik ini memiliki dimensi sebagai berikut:

- `SessionId`— ID sesi di mana perhitungan diajukan.
- `WorkGroup`— Nama kelompok kerja.

Untuk informasi lebih lanjut, lihat Daftar CloudWatch metrik dan dimensi untuk Athena nanti dalam topik ini. Untuk informasi tentang metrik penggunaan Athena, lihat. Memantau metrik penggunaan Athena

Untuk melihat metrik kueri untuk grup kerja di konsol

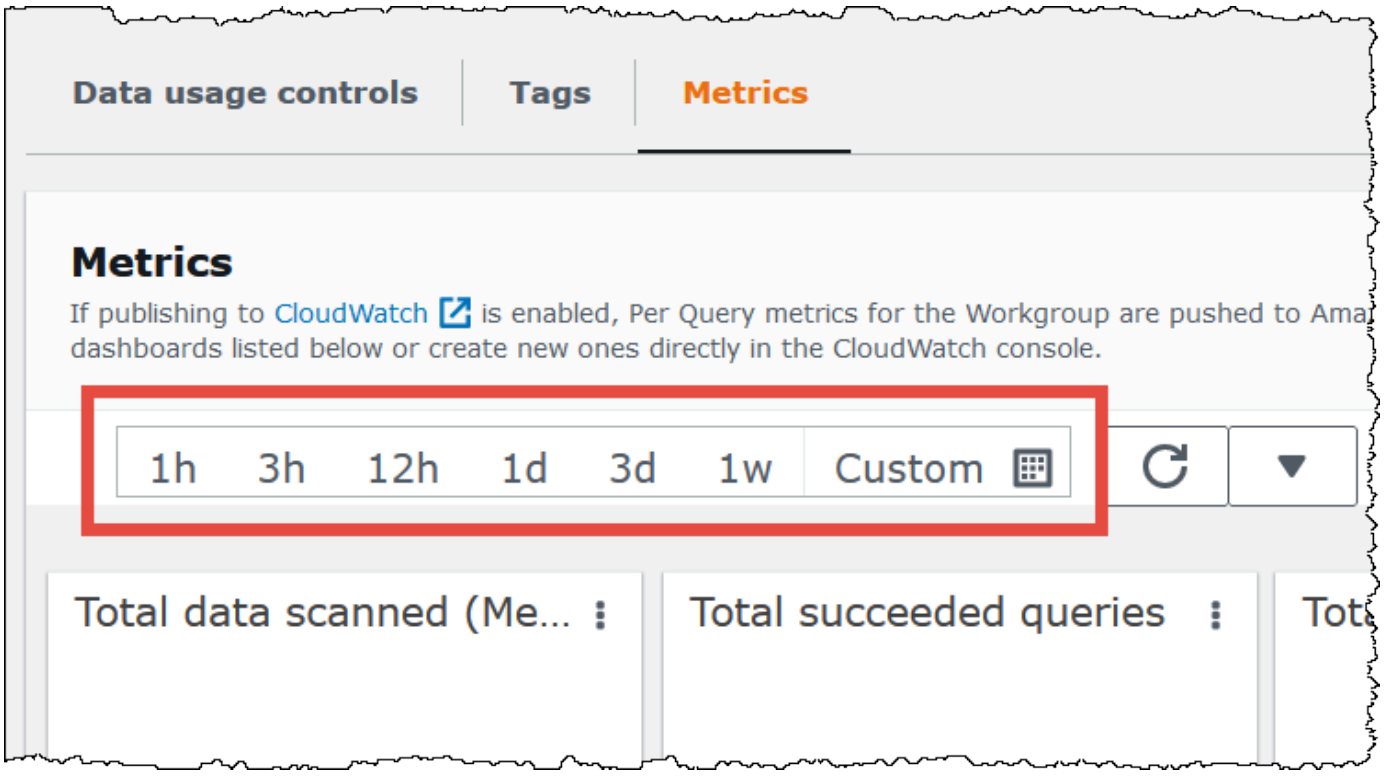
1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.

3. Di panel navigasi, pilih Workgroups.
4. Pilih workgroup yang Anda inginkan dari daftar, lalu pilih tab Metrik.

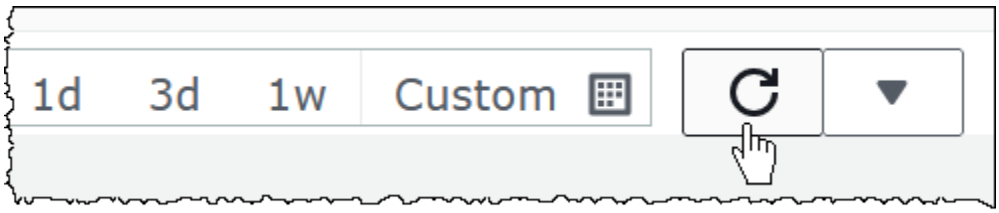
   Dasbor metrik ditampilkan.

   > **Note**
   >
   > Jika Anda baru saja mengaktifkan metrik untuk grup kerja dan/atau belum ada aktivitas kueri terbaru, grafik di dasbor mungkin kosong. Aktivitas kueri diambil dari CloudWatch tergantung pada interval yang Anda tentukan pada langkah berikutnya.

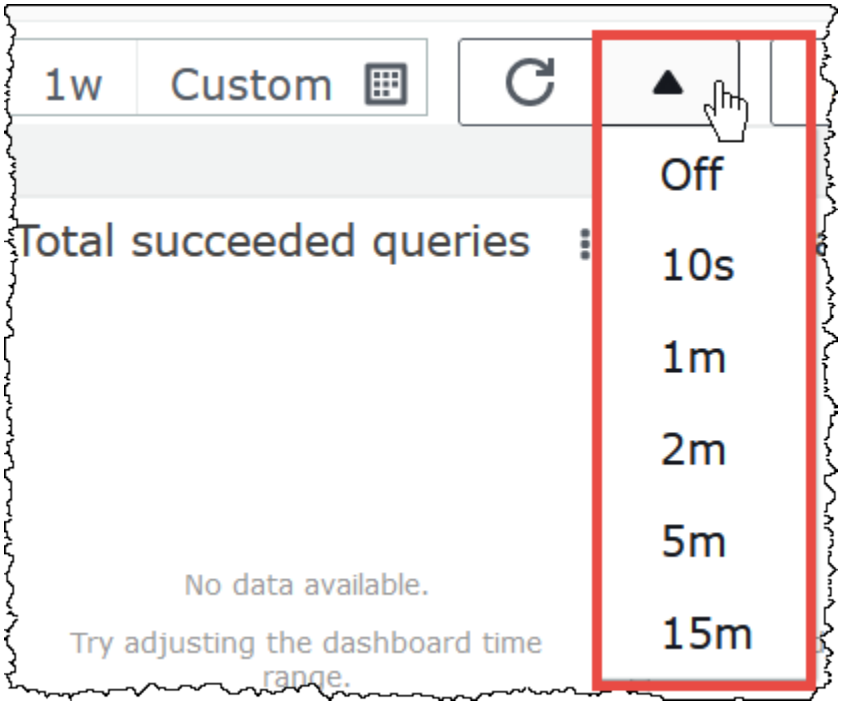
5. Di bagian Metrik, pilih interval metrik yang harus digunakan Athena untuk mengambil metrik kueri CloudWatch, atau tentukan interval khusus.



6. Untuk menyegarkan metrik yang ditampilkan, pilih ikon penyegaran.



7. Klik panah di sebelah ikon penyegaran untuk memilih seberapa sering Anda ingin tampilan metrik diperbarui.



Untuk melihat metrik di konsol Amazon CloudWatch

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, Semua metrik.
3. Pilih namespace AWS/Athena.

Untuk melihat metrik dengan CLI

- Lakukan salah satu hal berikut ini:
  - Untuk membuat daftar metrik Athena, buka prompt perintah, dan gunakan perintah berikut:

```
aws cloudwatch list-metrics --namespace "AWS/Athena"
```

  - Untuk mencantumkan semua metrik yang tersedia, gunakan perintah berikut:

```
aws cloudwatch list-metrics"
```

Daftar CloudWatch metrik dan dimensi untuk Athena

Jika Anda telah mengaktifkan CloudWatch metrik di Athena, metrik berikut akan dikirim ke per grup kerja CloudWatch . Metrik berikut menggunakan `AWS/Athena` namespace.

Nama metrik	Deskripsi
DPUalLokasi	Jumlah total DPU (unit pemrosesan data) yang disediakan dalam reservasi kapasitas untuk menjalankan kueri.
DPUDikonsumsi	Jumlah DPU yang secara aktif dikonsumsi oleh kueri dalam suatu `RUNNING` keadaan pada waktu tertentu dalam reservasi . Metrik ini dipancarkan hanya jika kelompok kerja dikaitkan dengan reservasi kapasitas dan mencakup semua kelompok kerja yang terkait dengan reservasi. Jika Anda memindahk an grup kerja dari satu reservasi ke reservasi lainnya, metrik tersebut menyertakan data dari saat grup kerja tersebut termasuk dalam reservasi pertama. Untuk informasi lebih lanjut tentang reservasi kapasitas, lihatMengelola kapasitas pemrosesa n kueri.
DPUCount	Jumlah maksimum DPU yang dikonsumsi oleh kueri Anda, diterbitkan tepat sekali saat kueri selesai. Metrik ini dipancarkan hanya untuk kelompok kerja yang dilampirkan pada reservasi kapasitas.
EngineExecutionTime	Jumlah milidetik yang dibutuhkan kueri untuk dijalankan.
ProcessedBytes	Jumlah byte yang dipindai Athena per kueri DML. Untuk kueri yang dibatalkan (baik oleh pengguna, atau secara otomatis, jika mereka mencapai batas), ini termasuk jumlah data yang dipindai sebelum waktu pembatalan. Metrik ini tidak dilaporkan untuk kueri DDL.
QueryPlanningTime	Jumlah milidetik yang Athena ambil untuk merencanakan alur pemrosesan kueri. Ini termasuk waktu yang dihabiskan untuk mengambil partisi tabel dari sumber data. Perhatikan bahwa

Nama metrik	Deskripsi
	karena mesin kueri melakukan perencanaan kueri, waktu perencanaan kueri adalah bagian dari. EngineExecutionTime
QueryQueueTime	Jumlah milidetik kueri berada dalam antrian kueri menunggu sumber daya. Perhatikan bahwa jika terjadi kesalahan sementara , kueri dapat secara otomatis ditambahkan kembali ke antrian.
ServicePreProcessingTime	Jumlah milidetik yang Athena ambil untuk memproses kueri sebelum mengirimkan kueri ke mesin kueri.
ServiceProcessingTime	Jumlah milidetik yang Athena ambil untuk memproses hasil kueri setelah mesin kueri selesai menjalankan kueri.
TotalExecutionTime	Jumlah milidetik yang Athena ambil untuk menjalankan kueri DDL atau DHTML. TotalExecutionTime termasuk QueryQueu eTime, QueryPlanningTime, EngineExecutionTime, dan ServiceProcessingTime.

Metrik untuk Athena ini memiliki dimensi berikut.

Dimensi	Deskripsi
CapacityReservation	Nama reservasi kapasitas yang digunakan untuk mengeksekusi query, jika berlaku. Ketika reservasi kapasitas tidak digunakan, dimensi ini tidak mengembalikan data.
QueryState	Status kueri. Statistik yang valid: BERHASIL, GAGAL, atau DIBATALKAN.
QueryType	Jenis kueri. Statistik yang valid:`DDL`,`DML`, atau`UTILITY`. Jenis pernyataa n query yang dijalankan. `DDL`menunjukkan pernyataan kueri DDL (Data Definition Language). `DML`menunjukkan pernyataa n query DHTML (Data Manipulation Language), sepertiCREATE TABLE AS SELECT. `UTILITY`menunjukkan pernyataan query

Dimensi	Deskripsi
	selain DDL dan DHTML, sepertiSHOW CREATE TABLE, atau. DESCRIBE TABLE
WorkGroup	Nama workgroup.

Memantau kueri Athena dengan acara Amazon EventBridge

Anda dapat menggunakan Amazon Athena dengan Amazon EventBridge untuk menerima pemberitahuan waktu nyata mengenai status pertanyaan Anda. Ketika kueri yang Anda kirimkan menyatakan transisi, Athena menerbitkan peristiwa EventBridge yang berisi informasi tentang transisi status kueri tersebut. Anda dapat menulis aturan sederhana untuk acara yang menarik bagi Anda dan mengambil tindakan otomatis saat acara cocok dengan aturan. Misalnya, Anda dapat membuat aturan yang memanggil AWS Lambda fungsi saat kueri mencapai status terminal. Peristiwa dipancarkan atas dasar upaya terbaik.

Sebelum Anda membuat aturan acara untuk Athena, Anda harus melakukan hal berikut:

- Biasakan diri Anda dengan acara, aturan, dan target di EventBridge. Untuk informasi selengkapnya, lihat Apa itu Amazon EventBridge? Untuk informasi selengkapnya tentang cara mengatur aturan, lihat Memulai Amazon EventBridge.
- Buat target tersebut atau target untuk digunakan dalam aturan peristiwa Anda.

> **Note**
>
> Athena saat ini menawarkan satu jenis acara, Athena Query State Change, tetapi dapat menambahkan jenis dan detail acara lainnya. Jika Anda secara terprogram deserialisasi data peristiwa JSON, pastikan aplikasi Anda siap untuk menangani properti yang tidak dikenal jika properti tambahan ditambahkan.

Format acara Athena

Berikut ini adalah pola dasar untuk acara Amazon Athena.

```
{
    "source":[
```

```
        "aws.athena"
    ],
    "detail-type":[
        "Athena Query State Change"
    ],
    "detail":{
        "currentState":[
            "SUCCEEDED"
        ]
    }
}
```

Acara perubahan status kueri Athena

Contoh berikut menunjukkan Athena Query State Change event dengan `currentState` nilai. `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"
    }
}
```

Contoh berikut menunjukkan Athena Query State Change event dengan `currentState` nilai. `FAILED` `athenaError`Blok hanya muncul ketika `currentState` ada`FAILED`. Untuk informasi tentang nilai untuk `errorCategory` dan`errorType`, lihat[Katalog kesalahan Athena](#).

```
{
    "version":"0",
    "id":"abcdef00-1234-5678-9abc-def012345678",
    "detail-type":"Athena Query State Change",
    "source":"aws.athena",
    "account":"123456789012",
    "time":"2019-10-06T09:30:10Z",
    "region":"us-east-1",
    "resources":[
    ],
    "detail":{
        "athenaError": {
            "errorCategory": 2.0, //Value depends on nature of exception
            "errorType": 1306.0, //Type depends on nature of exception
            "errorMessage": "Amazon S3 bucket not found", //Message depends on nature
 of exception
            "retryable":false //Retryable value depends on nature of exception
        },
        "versionId":"0",
        "currentState": "FAILED",
        "previousState": "RUNNING",
        "statementType":"DML",
        "queryExecutionId":"01234567-0123-0123-0123-012345678901",
        "workgroupName":"primary",
        "sequenceNumber":"3"
    }
}
```

Properti keluaran

Output JSON mencakup properti berikut.

Properti	Deskripsi
`athenaError`	Muncul hanya ketika `currentState` ituFAILED. Berisi informasi tentang kesalahan yang terjadi, termasuk kategori kesalahan, jenis kesalahan, pesan kesalahan, dan apakah tindakan yang menyebabk an kesalahan dapat dicoba ulang. Nilai untuk masing-masing bidang ini tergantung pada sifat kesalahan. Untuk informasi tentang nilai untuk `errorCategory` dan`errorType` , lihatKatalog kesalahan Athena.

Properti	Deskripsi
`versionId`	Nomor versi untuk skema objek detail.
`currentState`	Status bahwa query dialihkan ke pada saat acara.
`previousState`	Keadaan bahwa kueri dialihkan dari pada saat acara.
`statementType`	Jenis pernyataan query yang dijalankan.
`queryExecutionId`	Pengidentifikasi unik untuk kueri yang dijalankan.
`workgroupName`	Nama workgroup tempat kueri dijalankan.
`sequenceNumber`	Jumlah yang meningkat secara monoton yang memungkinkan deduplikasi dan pengurutan peristiwa masuk yang melibatkan satu kueri yang dijalankan. Ketika peristiwa duplikat diterbitkan untuk transisi status yang sama, `sequenceNumber` nilainya sama. Saat kueri mengalami transisi status lebih dari sekali, seperti kueri yang mengalami requeuing langka, Anda dapat menggunakannya `sequenceNumber` untuk mengurutkan peristiwa dengan nilai dan identik`currentState` . `previousState`

Contoh

Contoh berikut menerbitkan acara ke topik Amazon SNS yang telah Anda langgani. Ketika Athena ditanyai, Anda menerima email. Contohnya mengasumsikan bahwa topik Amazon SNS ada dan Anda telah berlangganan.

Untuk mempublikasikan acara Athena ke topik Amazon SNS

1. Buat target untuk topik Amazon SNS Anda. Berikan `events.amazonaws.com` izin Principal Layanan EventBridge acara untuk mempublikasikan ke topik Amazon SNS Anda, seperti pada contoh berikut.

```
{
    "Effect":"Allow",
    "Principal":{
        "Service":"events.amazonaws.com"
```

```
    },
    "Action":"sns:Publish",
    "Resource":"arn:aws:sns:us-east-1:111111111111:your-sns-topic"
}
```

2. Gunakan AWS CLI `events put-rule` perintah untuk membuat aturan untuk acara Athena, seperti pada contoh berikut.

```
aws events put-rule --name {ruleName} --event-pattern '{"source": ["aws.athena"]}'
```

3. Gunakan AWS CLI `events put-targets` perintah untuk melampirkan target topik Amazon SNS ke aturan, seperti pada contoh berikut.

```
aws events put-targets --rule {ruleName} --targets Id=1,Arn=arn:aws:sns:us-
east-1:111111111111:your-sns-topic
```

4. Tanyakan Athena dan amati target yang dipanggil. Anda harus menerima email yang sesuai dari topik Amazon SNS.

Menggunakan Notifikasi Pengguna AWS dengan Amazon Athena

Anda dapat menggunakan Notifikasi Pengguna AWSuntuk mengatur saluran pengiriman untuk mendapatkan pemberitahuan tentang acara Amazon Athena. Anda akan menerima notifikasi saat ada sebuah peristiwa yang cocok dengan sebuah aturan yang Anda tentukan. Anda dapat menerima notifikasi untuk peristiwa melalui beberapa saluran, termasuk email, notifikasi obrolan AWS Chatbot, atau notifikasi push AWS Console Mobile Application. Anda juga dapat melihat notifikasi di Pusat Pemberitahuan Konsol. Notifikasi Pengguna mendukung agregasi, yang dapat mengurangi jumlah notifikasi yang Anda terima selama acara tertentu.

Untuk informasi selengkapnya, silakan lihat Panduan Pengguna Notifikasi Pengguna AWS .

Memantau metrik penggunaan Athena

Anda dapat menggunakan metrik CloudWatch penggunaan untuk memberikan visibilitas tentang cara akun Anda menggunakan sumber daya dengan menampilkan penggunaan layanan saat ini pada CloudWatch grafik dan dasbor.

Untuk Athena, metrik ketersediaan penggunaan sesuai dengan kuota untuk Layanan AWS Athena. Anda dapat mengonfigurasi alarm yang memberi tahu Anda saat penggunaan mendekati kuota layanan. Untuk informasi lebih lanjut tentang kuota layanan Athena, lihat. Service Quotas Untuk

informasi selengkapnya tentang metrik AWS penggunaan, lihat metrik AWS penggunaan di CloudWatch Panduan Pengguna Amazon.

Athena menerbitkan metrik berikut di namespace. `AWS/Usage`

Nama metrik	Deskripsi
`ResourceCount`	Jumlah semua kueri antrian dan eksekusi Wilayah AWS per akun, dipisahkan oleh jenis kueri (DML/DDL). Maksimum adalah satu-satunya statistik yang berguna untuk metrik ini. Metrik ini diterbitkan secara berkala setiap menit. Jika Anda tidak menjalankan kueri apa pun, metrik tidak melaporkan apa pun (bahkan 0). Metrik hanya diterbitkan jika kueri aktif berjalan pada saat metrik diambil.

Dimensi berikut digunakan untuk menyempurnakan metrik penggunaan yang diterbitkan oleh Athena.

Dimensi	Deskripsi
`Service`	Nama yang Layanan AWS berisi sumber daya. Bagi Athena, nilai untuk dimensi ini adalah. `Athena`
`Resource`	Tipe sumber daya yang sedang berjalan. Nilai sumber daya untuk penggunaan kueri Athena adalah. `ActiveQueryCount`
`Type`	Jenis entitas yang dilaporkan. Saat ini, satu-satunya nilai yang valid untuk metrik penggunaan Athena adalah. `Resource`
`Class`	Kelas sumber daya yang akan dilacak. Untuk Athena, `Class` bisa `DML` atau. `DDL`

Melihat metrik penggunaan sumber daya Athena di konsol CloudWatch

Anda dapat menggunakan CloudWatch konsol untuk melihat grafik metrik penggunaan Athena dan mengonfigurasi alarm yang mengingatkan Anda saat penggunaan mendekati kuota layanan.

Untuk melihat metrik penggunaan sumber daya Athena

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, Semua metrik.
3. Pilih Usage, lalu pilih By AWS Resource.

   Daftar metrik penggunaan kuota layanan muncul.
4. Pilih kotak centang yang ada di sebelah Athena dan. ActiveQueryCount
5. Pilih tab Metrik bergrafik.

   Grafik di atas menampilkan penggunaan AWS sumber daya Anda saat ini.

Untuk informasi tentang menambahkan kuota layanan ke grafik dan menyetel alarm yang memberi tahu Anda jika Anda mendekati kuota layanan, lihat Memvisualisasikan kuota layanan Anda dan menyetel alarm di Panduan Pengguna Amazon. CloudWatch Untuk informasi tentang menyetel batas penggunaan per grup kerja, lihatMengatur batas kontrol penggunaan data.

Mengatur batas kontrol penggunaan data

Athena memungkinkan Anda mengatur dua jenis kontrol biaya: batas per kueri dan batas per kelompok kerja. Untuk setiap kelompok kerja, Anda hanya dapat menetapkan satu batas per kueri dan beberapa batas per grup kerja.

- Batas kontrol per kueri menentukan jumlah total data yang dipindai per kueri. Jika kueri apa pun yang berjalan di workgroup melebihi batas, kueri tersebut dibatalkan. Anda hanya dapat membuat satu batas kontrol per kueri dalam grup kerja dan itu berlaku untuk setiap kueri yang berjalan di dalamnya. Edit batas jika Anda perlu mengubahnya. Untuk langkah mendetail, lihat Untuk membuat kontrol penggunaan data per kueri.
- Batas kontrol penggunaan data seluruh kelompok kerja menentukan jumlah total data yang dipindai untuk semua kueri yang berjalan di grup kerja ini selama periode waktu yang ditentukan. Anda dapat membuat beberapa batasan per workgroup. Batas kueri seluruh grup kerja memungkinkan Anda menetapkan beberapa ambang batas pada agregat per jam atau harian pada data yang dipindai oleh kueri yang berjalan di grup kerja.

  Jika jumlah agregat data yang dipindai melebihi ambang batas, Anda dapat mendorong pemberitahuan ke topik Amazon SNS. Untuk melakukan ini, Anda mengonfigurasi alarm Amazon SNS dan tindakan di konsol Athena untuk memberi tahu administrator saat batas dilanggar. Untuk langkah mendetail, lihat Untuk membuat kontrol penggunaan data per grup kerja. Anda

juga dapat membuat alarm dan tindakan pada metrik apa pun yang diterbitkan Athena dari konsol. CloudWatch Misalnya, Anda dapat mengatur peringatan pada sejumlah kueri yang gagal. Peringatan ini dapat memicu email ke administrator jika nomor tersebut melewati ambang batas tertentu. Jika batas terlampaui, tindakan akan mengirimkan pemberitahuan alarm Amazon SNS ke pengguna yang ditentukan.

Tindakan lain yang dapat Anda ambil:

- Menginvokasi sebuah fungsi Lambda. Untuk informasi lebih lanjut, lihat Memanggil fungsi Lambda menggunakan pemberitahuan Amazon SNS dalam Panduan Pengembang Amazon Simple Notification Service.
- Nonaktifkan workgroup untuk menghentikan kueri lebih lanjut agar tidak berjalan. Untuk langkah, lihat Mengaktifkan dan menonaktifkan workgroup.

Batas per-query dan per-workgroup tidak tergantung satu sama lain. Tindakan tertentu diambil setiap kali salah satu batas terlampaui. Jika dua atau lebih pengguna menjalankan kueri pada saat yang sama dalam kelompok kerja yang sama, ada kemungkinan bahwa setiap kueri tidak melebihi batas yang ditentukan, tetapi jumlah total data yang dipindai melebihi batas penggunaan data per grup kerja. Dalam hal ini, alarm Amazon SNS dikirim ke pengguna.

Untuk membuat kontrol penggunaan data per kueri

Batas kontrol per kueri menentukan jumlah total data yang dipindai per kueri. Jika kueri apa pun yang berjalan di workgroup melebihi batas, kueri tersebut dibatalkan. Kueri yang dibatalkan dibebankan sesuai dengan harga Amazon Athena.

> **Note**
>
> Dalam kasus kueri yang dibatalkan atau gagal, Athena mungkin telah menulis sebagian hasil ke Amazon S3. Dalam kasus seperti itu, Athena tidak menghapus sebagian hasil dari awalan Amazon S3 tempat hasil disimpan. Anda harus menghapus awalan Amazon S3 dengan hasil sebagian. Athena menggunakan unggahan multipart Amazon S3 untuk menulis data Amazon S3. Kami menyarankan Anda menyetel kebijakan siklus hidup bucket untuk mengakhiri unggahan multibagian jika kueri gagal. Untuk informasi selengkapnya, lihat Membatalkan unggahan multibagian yang tidak lengkap menggunakan kebijakan siklus hidup bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Anda hanya dapat membuat satu batas kontrol per kueri dalam grup kerja dan itu berlaku untuk setiap kueri yang berjalan di dalamnya. Edit batas jika Anda perlu mengubahnya.

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.

3. Di panel navigasi, pilih Workgroups.
4. Pilih nama workgroup dari daftar.
5. Pada tab Kontrol penggunaan data, di bagian Kontrol penggunaan data per kueri, pilih Kelola.
6. Pada halaman kontrol penggunaan data Kelola per kueri, tentukan nilai berikut:

   - Untuk batas Data, tentukan nilai antara 10 MB (minimum) dan 7 EB (maksimum).

   > **Note**
   >
   > Ini adalah batasan yang diberlakukan oleh konsol untuk kontrol penggunaan data dalam kelompok kerja. Mereka tidak mewakili batas kueri apa pun di Athena.

   - Untuk unit, pilih nilai unit dari daftar drop-down (misalnya, Kilobytes KB atau Exabytes EB).

   Tindakan default adalah membatalkan kueri jika melebihi batas. Pengaturan ini tidak dapat diubah.
7. Pilih Simpan.

Untuk membuat atau mengedit peringatan penggunaan data per grup kerja

Anda dapat mengatur beberapa ambang peringatan saat kueri yang berjalan di grup kerja memindai jumlah data tertentu dalam periode tertentu. Peringatan diimplementasikan menggunakan CloudWatch alarm Amazon dan berlaku untuk semua kueri di workgroup. Ketika ambang batas tercapai, Anda dapat meminta Amazon SNS mengirim email ke pengguna yang Anda tentukan. Kueri tidak dibatalkan secara otomatis ketika ambang batas tercapai.

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.
3. Di panel navigasi, pilih Workgroups.
4. Pilih nama workgroup dari daftar.
5. Pilih Edit untuk mengedit pengaturan grup kerja.
6. Gulir ke bawah dan perluas peringatan penggunaan data Workgroup - opsional.
7. Pilih Tambahkan peringatan.
8. Untuk konfigurasi ambang penggunaan Data, tentukan nilai sebagai berikut:
   - Untuk Ambang data, tentukan angka, lalu pilih nilai satuan dari daftar drop-down.
   - Untuk jangka waktu, pilih periode waktu dari daftar drop-down.
   - Untuk pemilihan topik SNS, pilih topik Amazon SNS dari daftar drop-down. Atau, pilih Buat topik SNS untuk langsung masuk ke konsol Amazon SNS, buat topik Amazon SNS, dan siapkan langganan untuk salah satu pengguna di akun Athena Anda. Untuk informasi lebih lanjut, lihat Memulai dengan Amazon SNS di Panduan Developer Amazon Simple Notification Service.
9. Pilih Tambahkan peringatan jika Anda membuat peringatan baru, atau Simpan untuk menyimpan peringatan yang ada.

Mengelola kapasitas pemrosesan kueri

Anda dapat menggunakan reservasi kapasitas untuk mendapatkan kapasitas pemrosesan khusus untuk kueri yang Anda jalankan di Athena. Dengan reservasi kapasitas, Anda dapat memanfaatkan kemampuan manajemen beban kerja yang membantu Anda memprioritaskan, mengontrol, dan menskalakan beban kerja interaktif Anda yang paling penting. Misalnya, Anda dapat menambahkan kapasitas kapan saja untuk meningkatkan jumlah kueri yang dapat dijalankan secara bersamaan, mengontrol beban kerja mana yang dapat menggunakan kapasitas, dan berbagi kapasitas antar

beban kerja. Kapasitas dikelola sepenuhnya oleh Athena dan ditahan untuk Anda selama Anda membutuhkannya. Penyiapannya mudah dan tidak ada perubahan pada pernyataan SQL Anda yang diperlukan.

Untuk mendapatkan kapasitas pemrosesan untuk kueri Anda, Anda membuat reservasi kapasitas, menentukan jumlah Unit Pemrosesan Data (DPU) yang Anda butuhkan, dan menetapkan satu atau beberapa kelompok kerja ke reservasi.

Kelompok kerja memainkan peran penting ketika Anda menggunakan reservasi kapasitas. Workgroup memungkinkan Anda untuk mengatur kueri ke dalam pengelompokan logis. Dengan reservasi kapasitas, Anda secara selektif menetapkan kapasitas ke grup kerja sehingga Anda mengontrol perilaku kueri untuk setiap kelompok kerja dan bagaimana mereka ditagih. Untuk informasi selengkapnya tentang kelompok kerja, lihat[Menggunakan kelompok kerja untuk mengontrol akses kueri dan biaya](#).

Menetapkan grup kerja ke reservasi memungkinkan Anda memprioritaskan kueri yang Anda kirimkan ke grup kerja yang ditetapkan. Misalnya, Anda dapat mengalokasikan kapasitas ke grup kerja yang digunakan untuk kueri pelaporan keuangan yang sensitif terhadap waktu untuk mengisolasi kueri tersebut dari kueri yang kurang kritis di grup kerja lain. Ini memungkinkan eksekusi kueri yang konsisten untuk beban kerja kritis sambil memungkinkan beban kerja lain berjalan secara independen.

Anda dapat menggunakan reservasi kapasitas dan kelompok kerja bersama-sama untuk memenuhi persyaratan yang berbeda. Berikut ini adalah beberapa contoh skenario:

- Isolasi — Untuk mengisolasi beban kerja penting, Anda menetapkan satu kelompok kerja ke satu reservasi. Hanya kueri dari grup kerja yang ditetapkan yang menggunakan kapasitas pemrosesan dari reservasi yang dipilih.
- Berbagi — Beberapa beban kerja dapat berbagi kapasitas dari satu reservasi. Misalnya, jika Anda menginginkan biaya bulanan yang dapat diprediksi untuk serangkaian beban kerja tertentu, Anda dapat menetapkan beberapa grup kerja ke satu reservasi. Kelompok kerja yang ditugaskan berbagi kapasitas reservasi.
- Model campuran — Anda dapat menggunakan reservasi kapasitas dan penagihan per kueri secara bersamaan di akun yang sama. Misalnya, untuk memastikan eksekusi kueri yang andal yang mendukung aplikasi produksi, Anda menetapkan workgroup untuk kueri tersebut ke reservasi kapasitas. Saat mengembangkan kueri sebelum memindahkannya ke grup kerja produksi, Anda menggunakan workgroup terpisah yang tidak terkait dengan reservasi dan karenanya menggunakan penagihan per kueri.

Memahami DPU

Kapasitas diukur dalam Data Processing Unit (DPU). DPU mewakili sumber daya komputasi dan memori yang digunakan oleh Athena untuk mengakses dan memproses data atas nama Anda. Satu DPU menyediakan 4 vCPU dan 16 GB memori. Jumlah DPU yang Anda tentukan memengaruhi jumlah kueri yang dapat Anda jalankan secara bersamaan. Misalnya, reservasi dengan 256 DPU dapat mendukung kira-kira dua kali jumlah kueri bersamaan daripada reservasi dengan 128 DPU.

Anda dapat membuat hingga 100 kapasitas reservasi dengan hingga 1.000 total DPU per akun dan wilayah. Jumlah minimum DPU yang dapat Anda minta adalah 24. Jika Anda memerlukan lebih dari 1.000 DPU untuk kasus penggunaan Anda, silakan hubungi athena-feedback@amazon.com.

Untuk informasi tentang memperkirakan kebutuhan kapasitas Anda, lihatMenentukan persyaratan kapasitas. Untuk informasi harga, lihat harga Amazon Athena.

Pertimbangan dan batasan

- Fitur ini membutuhkan mesin Athena versi 3.
- Satu kelompok kerja dapat ditetapkan untuk paling banyak satu reservasi pada satu waktu, dan Anda dapat menambahkan maksimal 20 grup kerja ke reservasi.
- Anda tidak dapat menambahkan grup kerja berkemampuan Spark ke reservasi kapasitas.
- Untuk menghapus grup kerja yang telah ditetapkan ke reservasi, hapus grup kerja dari reservasi terlebih dahulu.
- Jumlah minimum DPU yang dapat Anda berikan adalah 24.
- Anda dapat membuat hingga 100 kapasitas reservasi dengan hingga 1.000 total DPU per akun dan wilayah.
- Permintaan kapasitas tidak dijamin dan dapat memakan waktu hingga 30 menit untuk diselesaikan.
- Ada periode penagihan minimal 1 jam per reservasi. Setelah 1 jam, kapasitas ditagih per menit. Untuk informasi harga, lihat harga Amazon Athena.
- Kapasitas cadangan tidak dapat dipindahtangankan ke reservasi kapasitas lain, Akun AWS, atau Wilayah AWS.
- Kueri DDL tentang reservasi kapasitas menggunakan DPU.
- Kueri yang berjalan pada kapasitas yang disediakan tidak dihitung terhadap batas kueri aktif Anda untuk DDL dan DHTML.
- Jika semua DPU sedang digunakan, kueri yang dikirimkan akan diantrian. Pertanyaan semacam itu tidak ditolak dan tidak masuk ke kapasitas sesuai permintaan.

- `DPUConsumed` CloudWatch Metriknya adalah per-workgroup, bukan per-reservasi. Jadi, jika Anda memindahkan grup kerja dari satu reservasi ke reservasi lainnya, `DPUConsumed` metrik tersebut menyertakan data dari saat grup kerja tersebut termasuk dalam reservasi pertama. Untuk informasi selengkapnya tentang penggunaan CloudWatch metrik di Athena, lihat. Memantau kueri Athena dengan metrik CloudWatch
- Saat ini, fitur tersebut tersedia sebagai berikut Wilayah AWS:
  - AS Timur (N. Virginia)
  - AS Timur (Ohio)
  - AS Barat (Oregon)
  - Asia Pasifik (Singapura)
  - Asia Pasifik (Sydney)
  - Asia Pasifik (Tokyo)
  - Eropa (Irlandia)
  - Eropa (Spanyol)
  - Eropa (Stockholm)
  - Amerika Selatan (Sao Paulo)

Topik

Menentukan persyaratan kapasitas

Sebelum Anda membuat reservasi kapasitas, Anda dapat memperkirakan kapasitas yang diperlukan sehingga Anda dapat menetapkan jumlah DPU yang benar. Dan, setelah reservasi digunakan, Anda mungkin ingin memeriksa reservasi untuk kapasitas yang tidak mencukupi atau kelebihan. Topik ini menjelaskan teknik yang dapat Anda gunakan untuk membuat perkiraan ini dan juga menjelaskan beberapa AWS alat untuk menilai penggunaan dan biaya.

Topik

Memperkirakan kapasitas yang dibutuhkan

Saat memperkirakan persyaratan kapasitas, penting untuk mempertimbangkan dua perspektif: berapa banyak kapasitas yang mungkin dibutuhkan kueri tertentu, dan berapa banyak kapasitas yang mungkin Anda butuhkan secara umum.

Memperkirakan persyaratan kapasitas per kueri

Untuk menentukan jumlah DPU yang mungkin diperlukan kueri, Anda dapat menggunakan panduan berikut:

- Kueri DDL mengkonsumsi 4 DPU.
- Kueri DHTML mengkonsumsi antara 4 dan 124 DPU.

Athena menentukan jumlah DPU yang dibutuhkan oleh kueri DHTML saat kueri dikirimkan. Jumlahnya bervariasi berdasarkan ukuran data, format penyimpanan, konstruksi kueri, dan faktor lainnya. Umumnya, Athena mencoba memilih nomor DPU terendah dan paling efisien. Jika Athena menentukan bahwa lebih banyak daya komputasi diperlukan agar kueri berhasil diselesaikan, itu meningkatkan jumlah DPU yang ditetapkan ke kueri.

Memperkirakan kebutuhan kapasitas spesifik beban kerja

Untuk menentukan berapa banyak kapasitas yang mungkin Anda perlukan untuk menjalankan beberapa kueri secara bersamaan, pertimbangkan pedoman umum dalam tabel berikut:

Kueri bersamaan	Diperlukan DPU
10	40 atau lebih
20	96 atau lebih
30 atau lebih	240 atau lebih

Perhatikan bahwa jumlah aktual DPU yang Anda butuhkan tergantung pada sasaran dan pola analisis Anda. Misalnya, jika Anda ingin kueri segera dimulai tanpa antrian, tentukan permintaan kueri bersamaan puncak Anda, lalu berikan jumlah DPU yang sesuai.

Anda dapat menyediakan DPU lebih sedikit daripada permintaan puncak Anda, tetapi antrian dapat terjadi ketika permintaan puncak terjadi. Saat antrian terjadi, Athena menyimpan kueri Anda dalam antrian dan menjalankannya saat kapasitas tersedia.

Jika tujuan Anda adalah menjalankan kueri dalam anggaran tetap, Anda dapat menggunakan Kalkulator AWS Harga untuk menentukan jumlah DPU yang sesuai dengan anggaran Anda.

Terakhir, ingat bahwa ukuran data, format penyimpanan, dan bagaimana kueri ditulis memengaruhi DPU yang dibutuhkan kueri. Untuk meningkatkan kinerja kueri, Anda dapat mengompres atau mempartisi data Anda atau mengubahnya menjadi format kolom. Untuk informasi selengkapnya, lihat Tuning kinerja di Athena.

Tanda-tanda bahwa lebih banyak kapasitas diperlukan

Pesan kesalahan kapasitas yang tidak mencukupi dan antrian kueri adalah dua indikasi bahwa kapasitas yang Anda tetapkan tidak memadai.

Jika kueri Anda gagal dengan pesan kesalahan kapasitas yang tidak mencukupi, jumlah DPU reservasi kapasitas Anda terlalu rendah untuk kueri Anda. Misalnya, jika Anda memiliki reservasi dengan 24 DPU dan menjalankan kueri yang membutuhkan lebih dari 24 DPU, kueri akan gagal. Untuk memantau kesalahan kueri ini, Anda dapat menggunakan EventBridgeacara Athena. Coba tambahkan lebih banyak DPU dan jalankan kembali kueri Anda.

Jika banyak pertanyaan antri, itu berarti kapasitas Anda sepenuhnya dimanfaatkan oleh pertanyaan lain. Untuk mengurangi antrian, lakukan salah satu hal berikut:

- Tambahkan DPU ke reservasi Anda untuk meningkatkan konkurensi kueri.
- Hapus grup kerja dari reservasi Anda untuk membebaskan kapasitas kueri lainnya.

Untuk memeriksa antrian kueri yang berlebihan, gunakan CloudWatchmetrik waktu antrian kueri Athena untuk grup kerja dalam reservasi kapasitas Anda. Jika nilainya di atas ambang batas pilihan Anda, Anda dapat menambahkan DPU ke reservasi kapasitas.

Memeriksa kapasitas idle

Untuk memeriksa kapasitas idle, Anda dapat mengurangi jumlah DPU dalam reservasi atau menambah beban kerjanya, dan kemudian mengamati hasilnya.

Untuk memeriksa kapasitas idle

1. Lakukan salah satu tindakan berikut:
   - Kurangi jumlah DPU dalam reservasi Anda (kurangi sumber daya yang tersedia)
   - Tambahkan grup kerja ke reservasi Anda (tambah beban kerja)
2. Gunakan CloudWatchuntuk mengukur waktu antrian kueri.
3. Jika waktu antrian meningkat melampaui level yang diinginkan, lakukan salah satu hal berikut
   - Hapus kelompok kerja
   - Tambahkan DPU ke reservasi kapasitas Anda
4. Setelah setiap perubahan, periksa kinerja dan waktu antrian kueri.
5. Terus sesuaikan beban kerja dan/atau jumlah DPU untuk mencapai keseimbangan yang diinginkan.

Jika Anda tidak ingin mempertahankan kapasitas di luar jangka waktu yang diinginkan, Anda dapat membatalkan reservasi dan membuat reservasi lain nanti. Namun, meskipun Anda baru saja membatalkan kapasitas dari reservasi lain, permintaan untuk kapasitas baru tidak dijamin, dan reservasi baru membutuhkan waktu untuk dibuat.

Alat untuk menilai kebutuhan kapasitas dan biaya

Anda dapat menggunakan layanan dan fitur berikut AWS untuk mengukur penggunaan dan biaya Athena Anda.

Metrik-metrik CloudWatch

Anda dapat mengonfigurasi Athena untuk memublikasikan metrik terkait kueri ke Amazon CloudWatch di tingkat workgroup. Setelah Anda mengaktifkan metrik untuk grup kerja, metrik untuk kueri grup kerja akan ditampilkan di konsol Athena di halaman detail grup kerja.

Untuk informasi tentang metrik Athena yang dipublikasikan CloudWatch dan dimensinya, lihat. Memantau kueri Athena dengan metrik CloudWatch

CloudWatch metrik penggunaan

Anda dapat menggunakan metrik CloudWatch penggunaan untuk memberikan visibilitas tentang cara akun Anda menggunakan sumber daya dengan menampilkan penggunaan layanan saat ini pada CloudWatch grafik dan dasbor. Untuk Athena, metrik ketersediaan penggunaan sesuai dengan kuota AWS layanan untuk Athena. Anda dapat mengonfigurasi alarm yang memberi tahu Anda saat penggunaan mendekati kuota layanan.

Untuk informasi selengkapnya, lihat Memantau metrik penggunaan Athena.

EventBridgeAcara Amazon

Anda dapat menggunakan Amazon Athena dengan Amazon EventBridge untuk menerima pemberitahuan waktu nyata mengenai status pertanyaan Anda. Ketika kueri yang Anda kirimkan berubah menyatakan, Athena menerbitkan peristiwa EventBridge yang berisi informasi tentang transisi status kueri. Anda dapat menulis aturan sederhana untuk acara yang menarik bagi Anda dan mengambil tindakan otomatis saat acara cocok dengan aturan.

Untuk informasi selengkapnya, lihat sumber daya berikut ini.

- Memantau kueri Athena dengan acara Amazon EventBridge
- Apa itu Amazon EventBridge?
- EventBridge Acara Amazon

Tanda

Di Athena, reservasi kapasitas mendukung tag. Tag terdiri dari kunci dan nilai. Untuk melacak biaya Anda di Athena, Anda dapat menggunakan tag alokasi biaya AWS yang dihasilkan. AWS menggunakan tag alokasi biaya untuk mengatur biaya sumber daya Anda pada Laporan Biaya dan Penggunaan Anda. Ini memudahkan Anda untuk mengkategorikan dan melacak biaya Anda AWS . Untuk mengaktifkan tag alokasi biaya untuk Athena, Anda menggunakan AWS Billing and Cost Management konsol.

Untuk informasi selengkapnya, lihat sumber daya berikut ini.

- Menandai sumber daya Athena
- Mengaktifkan tag AWS alokasi biaya yang dihasilkan
- Menggunakan tag alokasi biaya AWS

Membuat reservasi kapasitas

Untuk memulai, Anda membuat reservasi kapasitas yang memiliki jumlah DPU yang Anda butuhkan, dan kemudian menetapkan satu atau lebih kelompok kerja yang akan menggunakan kapasitas itu untuk kueri mereka. Anda dapat menyesuaikan kapasitas Anda nanti sesuai kebutuhan untuk memberikan kinerja yang lebih konsisten atau mengelola biaya dengan lebih baik. Untuk informasi tentang memperkirakan kebutuhan kapasitas Anda, lihatMenentukan persyaratan kapasitas.

> **Important**
>
> Permintaan kapasitas tidak dijamin dan dapat memakan waktu hingga 30 menit untuk diselesaikan.

Untuk membuat reservasi kapasitas

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.
3. Pilih Administrasi, Pemesanan kapasitas.
4. Pilih Buat reservasi kapasitas.
5. Pada halaman Buat reservasi kapasitas, untuk nama reservasi Kapasitas, masukkan nama. Nama harus unik, dari 1 hingga 128 karakter, dan hanya menggunakan karakter a-z, A-Z, 0-9, _ (garis bawah),. (periode) dan - (tanda hubung) .Anda tidak dapat mengubah nama setelah Anda membuat reservasi.
6. Untuk DPU, pilih atau masukkan jumlah unit pemrosesan data (DPU) yang Anda inginkan dengan penambahan 4. Untuk informasi selengkapnya, lihat Memahami DPU.
7. (Opsional) Perluas opsi Tag, lalu pilih Tambahkan tag baru untuk menambahkan satu atau beberapa pasangan kunci/nilai khusus untuk dikaitkan dengan sumber daya reservasi kapasitas. Untuk informasi selengkapnya, lihat Menandai sumber daya Athena.
8. Pilih Tinjau.
9. Pada prompt Konfirmasi buat reservasi kapasitas, konfirmasikan jumlah DPU, Wilayah AWS, dan informasi lainnya. Jika Anda menerima, pilih Kirim.

   Pada halaman detail, Status reservasi kapasitas Anda ditampilkan sebagai Tertunda. Ketika kapasitas reservasi Anda tersedia untuk menjalankan kueri, statusnya ditampilkan sebagai Aktif.

Pada titik ini, Anda siap untuk menambahkan satu atau lebih kelompok kerja ke reservasi Anda. Untuk langkah, lihat Menambahkan workgroup ke reservasi.

Mengelola reservasi

Anda dapat melihat dan mengelola reservasi kapasitas Anda di halaman reservasi Kapasitas. Anda dapat melakukan tugas manajemen seperti menambahkan atau mengurangi DPU, memodifikasi tugas kelompok kerja, dan menandai atau membatalkan reservasi.

Untuk melihat dan mengelola reservasi kapasitas

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.
3. Pilih Administrasi, Pemesanan kapasitas.
4. Pada halaman reservasi kapasitas, Anda dapat melakukan tugas-tugas berikut:
   - Untuk membuat reservasi kapasitas, pilih Buat reservasi kapasitas.
   - Gunakan kotak pencarian untuk memfilter reservasi berdasarkan nama atau jumlah DPU.
   - Pilih menu tarik-turun status untuk memfilter berdasarkan status reservasi kapasitas (misalnya, Aktif atau Dibatalkan). Untuk informasi tentang status reservasi, lihatMemahami status reservasi.
   - Untuk melihat detail reservasi kapasitas, pilih tautan untuk reservasi. Halaman detail untuk reservasi mencakup opsi untuk mengedit kapasitas, menambahkan grup kerja, menghapus grup kerja, dan untuk membatalkan reservasi.
   - Untuk mengedit reservasi (misalnya, dengan menambahkan atau menghapus DPU), pilih tombol untuk reservasi, lalu pilih Edit.
   - Untuk membatalkan reservasi, pilih tombol untuk reservasi, lalu pilih Batalkan.

Memahami status reservasi

Tabel berikut menjelaskan kemungkinan nilai status untuk reservasi kapasitas.

Status	Deskripsi
Tertunda	Athena sedang memproses permintaan kapasitas Anda. Kapasitas belum siap untuk menjalankan kueri.

Status	Deskripsi
Aktif	Kapasitas tersedia untuk menjalankan kueri.
Failed	Permintaan kapasitas Anda tidak berhasil diselesaikan. Perhatikan bahwa pemenuhan permintaan kapasitas tidak dijamin. Reservasi yang gagal dihitung terhadap batas DPU akun Anda. Untuk melepaskan penggunaan, Anda harus membatalkan reservasi.
Pembaruan tertunda	Athena sedang memproses perubahan pada reservasi. Misalnya, status ini terjadi setelah Anda mengedit reservasi untuk menambah atau menghapus DPU.
Membatalkan	Athena sedang memproses permintaan untuk membatalkan reservasi . Kueri yang masih berjalan di grup kerja yang menggunakan reservasi diizinkan untuk diselesaikan, tetapi kueri lain di grup kerja akan menggunakan kapasitas sesuai permintaan (tidak disediakan).
Dibatalkan	Pembatalan reservasi kapasitas selesai. Pemesanan yang dibatalka n tetap berada di konsol selama 45 hari. Setelah 45 hari, Athena akan menghapus reservasi. Selama 45 hari, Anda tidak dapat melakukan tujuan ulang atau menggunakan kembali reservasi, tetapi Anda dapat merujuk pada tag dan melihat detailnya untuk referensi historis. Kapasitas yang dibatalkan tidak dijamin dapat dipesan ulang di masa mendatang. Kapasitas tidak dapat ditransfer ke reservasi lain, Akun AWS atau Wilayah AWS.

Memahami DPU Aktif dan DPU Target

Dalam daftar reservasi kapasitas di konsol Athena, reservasi Anda menampilkan dua nilai DPU: DPU Aktif dan DPU Target.

- DPU Aktif — Jumlah DPU yang tersedia di reservasi Anda untuk menjalankan kueri. Misalnya, jika Anda meminta 100 DPU, dan permintaan Anda terpenuhi, DPU Aktif menampilkan 100.
- Target DPU — Jumlah DPU yang Anda reservasi sedang dalam proses pindah. Target DPU menampilkan nilai yang berbeda dari DPU Aktif ketika reservasi sedang dibuat, atau peningkatan atau penurunan jumlah DPU tertunda.

Misalnya, setelah Anda mengajukan permintaan untuk membuat reservasi dengan 24 DPU, Status reservasi akan Tertunda, DPU Aktif akan menjadi 0, dan DPU Target akan menjadi 24.

Jika Anda memiliki reservasi dengan 100 DPU, dan mengedit reservasi Anda untuk meminta peningkatan 20 DPU, Status akan Update tertunda, DPU Aktif akan menjadi 100, dan Target DPU akan menjadi 120.

Jika Anda memiliki reservasi dengan 100 DPU, dan mengedit reservasi Anda untuk meminta penurunan 20 DPU, Status akan Update tertunda, DPU Aktif akan menjadi 100, dan Target DPU akan menjadi 80.

Selama transisi ini, Athena secara aktif bekerja untuk memperoleh atau mengurangi jumlah DPU berdasarkan permintaan Anda. Ketika DPU Aktif menjadi sama dengan DPU Target, jumlah target telah tercapai dan tidak ada perubahan yang tertunda.

Untuk mengambil nilai-nilai ini secara terprogram, Anda dapat memanggil tindakan API. GetCapacityReservation API mengacu pada DPU Aktif dan DPU Target sebagai `AllocatedDpus` dan. `TargetDpus`

Topik

- Reservasi kapasitas pengeditan
- Menambahkan workgroup ke reservasi
- Menghapus workgroup dari reservasi
- Membatalkan reservasi kapasitas
- Menghapus reservasi kapasitas

Reservasi kapasitas pengeditan

Setelah Anda membuat reservasi kapasitas, Anda dapat menyesuaikan jumlah DPU-nya dan menambah atau menghapus tag kustomnya.

Untuk mengedit reservasi kapasitas

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.
3. Pilih Administrasi, Pemesanan kapasitas.
4. Dalam daftar reservasi kapasitas, lakukan salah satu hal berikut:

- Pilih tombol di sebelah reservasi, lalu pilih Edit.
- Pilih tautan reservasi, lalu pilih Edit.

5. Untuk DPU, pilih atau masukkan jumlah unit pemrosesan data yang Anda inginkan dengan penambahan 4. Jumlah minimum DPU yang dapat Anda miliki adalah 24. Untuk informasi selengkapnya, lihat Memahami DPU.

> **Note**
>
> Anda dapat menambahkan DPU ke reservasi kapasitas yang ada kapan saja. Namun, Anda tidak dapat mengurangi jumlah DPU hingga 1 jam setelah Anda membuat reservasi atau menambahkan DPU ke dalamnya.

6. (Opsional) Untuk Tag, pilih Hapus untuk menghapus tag, atau pilih Tambahkan tag baru untuk menambahkan tag baru.
7. Pilih Kirim. Halaman detail untuk reservasi menunjukkan konfigurasi yang diperbarui.

Menambahkan workgroup ke reservasi

Setelah membuat reservasi kapasitas, Anda dapat menambahkan hingga 20 grup kerja ke reservasi. Menambahkan grup kerja ke reservasi memberi tahu Athena kueri mana yang harus dijalankan pada kapasitas cadangan Anda. Kueri dari grup kerja yang tidak terkait dengan reservasi terus berjalan menggunakan model harga pindaian per terabyte (TB) default.

Jika reservasi memiliki dua grup kerja atau lebih, kueri dari kelompok kerja tersebut dapat menggunakan kapasitas reservasi. Anda dapat menambah dan menghapus workgroup kapan saja. Saat Anda menambah atau menghapus grup kerja, kueri yang sedang berjalan tidak akan terganggu.

Ketika reservasi Anda dalam status tertunda, kueri dari grup kerja yang telah Anda tambahkan terus berjalan menggunakan model harga yang dipindai per terabyte (TB) default hingga reservasi menjadi aktif.

Untuk menambahkan satu atau beberapa grup kerja ke reservasi kapasitas Anda

1. Pada halaman detail untuk reservasi kapasitas, pilih Tambahkan grup kerja.
2. Pada halaman Tambahkan grup kerja, pilih grup kerja yang ingin Anda tambahkan, lalu pilih Tambahkan grup kerja. Anda tidak dapat menetapkan workgroup ke lebih dari satu reservasi.

Halaman detail untuk reservasi kapasitas Anda mencantumkan grup kerja yang Anda tambahkan. Kueri yang berjalan dalam kelompok kerja tersebut akan menggunakan kapasitas yang Anda pesan saat reservasi aktif.

Menghapus workgroup dari reservasi

Jika Anda tidak lagi memerlukan kapasitas khusus untuk grup kerja atau ingin memindahkan grup kerja ke reservasi sendiri, Anda dapat menghapusnya kapan saja. Menghapus workgroup dari reservasi adalah proses yang mudah. Setelah Anda menghapus grup kerja dari reservasi, kueri dari grup kerja yang dihapus secara default menggunakan kapasitas sesuai permintaan (tidak disediakan) dan ditagih berdasarkan terabyte (TB) yang dipindai.

Untuk menghapus satu atau beberapa grup kerja dari reservasi

1. Pada halaman detail untuk reservasi kapasitas, pilih grup kerja yang ingin Anda hapus.
2. Pilih Hapus grup kerja. Hapus kelompok kerja? prompt memberi tahu Anda bahwa semua kueri yang saat ini aktif akan selesai sebelum workgroup dihapus dari reservasi..
3. Pilih Hapus. Halaman detail untuk reservasi kapasitas Anda menunjukkan bahwa grup kerja yang dihapus tidak lagi ada.

Membatalkan reservasi kapasitas

Jika Anda tidak lagi ingin menggunakan reservasi kapasitas, Anda dapat membatalkannya. Kueri yang masih berjalan di grup kerja yang menggunakan reservasi akan diizinkan untuk diselesaikan, tetapi kueri lain di grup kerja tidak akan lagi menggunakan reservasi.

> **Note**
>
> Kapasitas yang dibatalkan tidak dijamin dapat dipesan ulang di masa mendatang. Kapasitas tidak dapat ditransfer ke reservasi lain, Akun AWS atau Wilayah AWS.

Untuk membatalkan reservasi kapasitas

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.
3. Pilih Administrasi, Pemesanan kapasitas.

4. Dalam daftar reservasi kapasitas, lakukan salah satu hal berikut:
   - Pilih tombol di sebelah reservasi, lalu pilih Batalkan.
   - Pilih tautan reservasi, lalu pilih Batalkan reservasi kapasitas.
5. Pada reservasi kapasitas Batalkan? prompt, masukkan batal, lalu pilih Batalkan reservasi kapasitas.

   Status reservasi berubah menjadi Pembatalan, dan spanduk kemajuan memberi tahu Anda bahwa pembatalan sedang berlangsung.

   Ketika pembatalan selesai, reservasi kapasitas tetap ada, tetapi statusnya ditampilkan sebagai Dibatalkan. Reservasi akan dihapus 45 hari setelah pembatalan. Selama 45 hari, Anda tidak dapat mengubah tujuan atau menggunakan kembali reservasi yang telah dibatalkan, tetapi Anda dapat merujuk ke tag dan melihatnya untuk referensi historis.

Menghapus reservasi kapasitas

Jika Anda ingin menghapus semua referensi ke reservasi kapasitas yang dibatalkan, Anda dapat menghapus reservasi. Reservasi harus dibatalkan sebelum dapat dihapus. Reservasi yang dihapus segera dihapus dari akun Anda dan tidak dapat lagi direferensikan, termasuk oleh ARN-nya.

Untuk menghapus reservasi kapasitas

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.
3. Pilih Administrasi, Pemesanan kapasitas.
4. Dalam daftar reservasi kapasitas, lakukan salah satu hal berikut:
   - Pilih tombol di samping reservasi yang dibatalkan, lalu pilih Tindakan, Hapus.
   - Pilih tautan reservasi, lalu pilih Hapus.
5. Pada reservasi kapasitas Hapus? prompt, pilih Hapus.

   Spanduk memberi tahu Anda bahwa reservasi kapasitas telah berhasil dihapus. Reservasi yang dihapus tidak lagi muncul dalam daftar reservasi kapasitas.

Kebijakan IAM untuk pemesanan kapasitas

Untuk mengontrol akses ke reservasi kapasitas, gunakan izin IAM tingkat sumber daya atau kebijakan IAM berbasis identitas. Setiap kali Anda menggunakan kebijakan IAM, pastikan bahwa Anda mengikuti praktik terbaik IAM. Untuk informasi selengkapnya tentang administrator, lihat Praktik Terbaik IAM dalam Panduan Pengguna IAM.

Prosedur berikut khusus untuk Athena.

Untuk informasi khusus IAM, lihat tautan yang tercantum di akhir bagian ini. Untuk informasi tentang contoh kebijakan reservasi kapasitas JSON, lihatKebijakan contoh reservasi kapasitas.

Untuk menggunakan editor visual di konsol IAM untuk membuat kebijakan reservasi kapasitas

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi yang ada di sebelah kiri, pilih Kebijakan, lalu pilih Buat kebijakan.
3. Pada tab Editor visual, pilih Pilih layanan. Pilih Athena untuk ditambahkan ke kebijakan.
4. Pilih Pilih tindakan, kemudian pilih tindakan untuk ditambahkan ke kebijakan. Editor visual menunjukkan tindakan yang tersedia di Athena. Untuk informasi lebih lanjut, lihatTindakan, sumber daya, dan kunci kondisi untuk Amazon Athenadi dalamReferensi Otorisasi Layanan.
5. Pilihmenambahkan tindakanuntuk mengetikkan tindakan tertentu atau menggunakan karakter wild card (*) untuk menentukan beberapa tindakan.

   Secara default, kebijakan yang Anda buat mengizinkan tindakan yang Anda pilih. Jika Anda memilih satu atau lebih tindakan yang mendukung izin level sumber daya ke sumber daya `capacity-reservation` di Athena, editor akan mencantumkan sumber daya `capacity-reservation` tersebut.
6. PilihSumber Dayauntuk menentukan reservasi kapasitas khusus untuk polis Anda. Misalnya kebijakan reservasi kapasitas JSON, lihatKebijakan contoh reservasi kapasitas.
7. Tentukan sumber daya `capacity-reservation` sebagai berikut:

   ```
   arn:aws:athena:<region>:<user-account>:capacity-reservation/<capacity-reservation-name>
   ```
8. Pada halaman Tinjau kebijakan, ketik Nama dan Deskripsi (opsional) untuk kebijakan yang Anda buat. Tinjau ringkasan kebijakan untuk memastikan bahwa Anda memberikan izin yang dimaksud.

9. Pilih Buat kebijakan untuk menyimpan kebijakan baru Anda.
10. Lampirkan kebijakan berbasis identitas ini ke pengguna, grup, atau peran.

Untuk informasi selengkapnya, lihat topik berikut diReferensi Otorisasi LayanandanPanduan Pengguna IAM:

- Tindakan, sumber daya, dan kunci kondisi untuk Amazon Athena
- Membuat kebijakan dengan editor visual
- Menambahkan dan menghapus kebijakan IAM
- Mengontrol akses ke sumber daya

Misalnya kebijakan reservasi kapasitas JSON, lihatKebijakan contoh reservasi kapasitas.

Untuk daftar lengkap tindakan Amazon Athena, lihat nama tindakan API diReferensi API Amazon Athena.

Kebijakan contoh reservasi kapasitas

Bagian ini mencakup contoh kebijakan yang dapat Anda gunakan untuk mengaktifkan berbagai tindakan pada reservasi kapasitas. Setiap kali Anda menggunakan kebijakan IAM, pastikan bahwa Anda mengikuti praktik terbaik IAM. Untuk informasi selengkapnya tentang administrator, lihat Praktik Terbaik IAM dalam Panduan Pengguna IAM.

Reservasi kapasitas adalah sumber daya IAM yang dikelola oleh Athena. Oleh karena itu, jika kebijakan reservasi kapasitas Anda menggunakan tindakan yang diambil`capacity-reservation`sebagai masukan, Anda harus menentukan ARN reservasi kapasitas sebagai berikut:

```
"Resource": [arn:aws:athena:<region>:<user-account>:capacity-reservation/<capacity-reservation-name>]
```

Dimana*<capacity-reservation-name>*adalah nama reservasi kapasitas Anda. Misalnya, untuk reservasi kapasitas bernama`test_capacity_reservation`, tentukan sebagai sumber daya sebagai berikut:

```
"Resource": ["arn:aws:athena:us-east-1:123456789012:capacity-reservation/test_capacity_reservation"]
```

Untuk daftar lengkap tindakan Amazon Athena, lihat nama tindakan API diReferensi API Amazon Athena. Untuk informasi selengkapnya tentang kebijakan IAM, lihatMembuat kebijakan dengan editor visualdi dalamPanduan Pengguna IAM.

- Example policy to list capacity reservations
- Example policy for management operations

Example Contoh kebijakan untuk mencantumkan reservasi kapasitas

Kebijakan berikut memungkinkan semua pengguna untuk mencantumkan semua reservasi kapasitas.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListCapacityReservations"
            ],
            "Resource": "*"
        }
    ]
}
```

Example Contoh kebijakan untuk operasi manajemen

Kebijakan berikut memungkinkan pengguna untuk membuat, membatalkan, mendapatkan detail, dan memperbarui reservasi kapasitas`test_capacity_reservation`. Kebijakan ini juga memungkinkan pengguna untuk menetapkan`workgroupA`dan`workgroupB`kepada`test_capacity_reservation`.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect": "Allow",
         "Action": [
             "athena:CreateCapacityReservation",
             "athena:GetCapacityReservation",
             "athena:CancelCapacityReservation",
             "athena:UpdateCapacityReservation",
```

```
            "athena:GetCapacityAssignmentConfiguration",
            "athena:PutCapacityAssignmentConfiguration"
        ],
        "Resource": [
            "arn:aws:athena:us-east-1:123456789012:capacity-
reservation/test_capacity_reservation",
            "arn:aws:athena:us-east-1:123456789012:workgroup/workgroupA",
            "arn:aws:athena:us-east-1:123456789012:workgroup/workgroupB"
        ]
      }
   ]
}
```

API reservasi kapasitas Athena

Daftar berikut berisi tautan referensi ke tindakan API reservasi kapasitas Athena. Untuk struktur data dan tindakan API Athena lainnya, lihatReferensi API Amazon Athena.

- CancelCapacityReservation
- CreateCapacityReservation
- GetCapacityAssignmentConfiguration
- GetCapacityReservation
- ListCapacityReservations
- PutCapacityAssignmentConfiguration
- UpdateCapacityReservation

Tuning kinerja di Athena

Topik ini memberikan informasi umum dan saran khusus untuk meningkatkan kinerja kueri Athena Anda, dan cara mengatasi kesalahan yang terkait dengan batasan dan penggunaan sumber daya.

Kuota layanan

Athena memberlakukan kuota untuk metrik seperti waktu berjalan kueri, jumlah kueri bersamaan dalam akun, dan tingkat permintaan API. Untuk informasi lebih lanjut tentang kuota ini, lihatService Quotas. Melebihi kuota ini menyebabkan kueri gagal — baik saat dikirimkan, atau selama eksekusi kueri.

Banyak tips optimasi kinerja di halaman ini dapat membantu mengurangi waktu berjalan kueri. Optimalisasi membebaskan kapasitas sehingga Anda dapat menjalankan lebih banyak kueri dalam kuota konkurensi dan menjaga agar kueri tidak dibatalkan karena berjalan terlalu lama.

Kuota pada jumlah kueri bersamaan dan permintaan API adalah per dan. Akun AWS Wilayah AWS Kami merekomendasikan menjalankan satu beban kerja per Akun AWS (atau menggunakan reservasi kapasitas yang disediakan terpisah) agar beban kerja tidak bersaing untuk kuota yang sama.

Jika Anda menjalankan dua beban kerja di akun yang sama, salah satu beban kerja dapat menjalankan banyak kueri. Hal ini dapat menyebabkan beban kerja yang tersisa terhambat atau diblokir agar tidak menjalankan kueri. Untuk menghindari hal ini, Anda dapat memindahkan beban kerja ke akun terpisah untuk memberikan setiap beban kerja kuota konkurensi sendiri. Membuat reservasi kapasitas yang disediakan untuk salah satu atau kedua beban kerja mencapai tujuan yang sama.

Kuota di layanan lain

Ketika Athena menjalankan kueri, ia dapat memanggil layanan lain yang memberlakukan kuota. Selama eksekusi kueri, Athena dapat melakukan panggilan API ke, Amazon S3 AWS Glue Data Catalog, dan layanan AWS lain seperti IAM dan. AWS KMS Jika Anda menggunakan kueri federasi, Athena juga menelepon. AWS Lambda Semua layanan ini memiliki batas dan kuota sendiri yang dapat dilampaui. Ketika eksekusi kueri menemukan kesalahan dari layanan ini, gagal dan menyertakan kesalahan dari layanan sumber. Kesalahan yang dapat dipulihkan dicoba ulang, tetapi kueri masih bisa gagal jika masalah tidak teratasi dengan sendirinya tepat waktu. Pastikan untuk membaca pesan kesalahan secara menyeluruh untuk menentukan apakah pesan tersebut berasal dari Athena atau dari layanan lain. Beberapa kesalahan yang relevan tercakup dalam dokumen ini.

Untuk informasi selengkapnya tentang mengatasi kesalahan yang disebabkan oleh kuota layanan Amazon S3, lihat Hindari memiliki terlalu banyak file nanti di dokumen ini. Untuk informasi selengkapnya tentang pengoptimalan kinerja Amazon S3, lihat Pola desain praktik terbaik: mengoptimalkan kinerja Amazon S3 di Panduan Pengguna Amazon S3.

Batas sumber daya

Athena menjalankan kueri di mesin kueri terdistribusi. Saat Anda mengirimkan kueri, perencana kueri mesin Athena memperkirakan kapasitas komputasi yang diperlukan untuk menjalankan kueri dan menyiapkan sekelompok node komputasi yang sesuai. Beberapa query seperti query DDL berjalan hanya pada satu node. Kueri kompleks pada kumpulan data besar berjalan pada cluster yang

jauh lebih besar. Node seragam, dengan konfigurasi memori, CPU, dan disk yang sama. Athena meningkatkan, bukan meningkatkan, untuk memproses pertanyaan yang lebih menuntut.

Terkadang permintaan kueri melebihi sumber daya yang tersedia untuk cluster yang menjalankan kueri. Ketika ini terjadi, kueri gagal dengan kesalahan Sumber daya yang habis Kueri pada faktor skala ini.

Sumber daya yang paling sering habis adalah memori, tetapi dalam kasus yang jarang terjadi juga bisa berupa ruang disk. Kesalahan memori biasanya terjadi ketika mesin melakukan fungsi gabungan atau jendela, tetapi mereka juga dapat terjadi dalam jumlah dan agregasi yang berbeda.

Bahkan jika kueri gagal dengan kesalahan 'kehabisan sumber daya' sekali, itu mungkin berhasil ketika Anda menjalankannya lagi. Eksekusi kueri tidak deterministik. Faktor-faktor seperti berapa lama waktu yang dibutuhkan untuk memuat data dan bagaimana dataset menengah didistribusikan melalui node dapat menghasilkan penggunaan sumber daya yang berbeda. Misalnya, bayangkan kueri yang menggabungkan dua tabel dan memiliki kemiringan berat dalam distribusi nilai untuk kondisi gabungan. Kueri semacam itu dapat berhasil sebagian besar waktu tetapi kadang-kadang gagal ketika nilai yang paling umum akhirnya diproses oleh node yang sama.

Untuk mencegah kueri Anda melebihi sumber daya yang tersedia, gunakan kiat penyetelan kinerja yang disebutkan dalam dokumen ini. Khususnya, untuk tips tentang cara mengoptimalkan kueri yang menghabiskan sumber daya yang tersedia, lihat Mengoptimalkan bergabungMengoptimalkan fungsi jendela, danMengoptimalkan kueri dengan menggunakan perkiraan.

Teknik pengoptimalan kueri

Gunakan teknik pengoptimalan kueri yang dijelaskan di bagian ini untuk membuat kueri berjalan lebih cepat atau sebagai solusi untuk kueri yang melebihi batas sumber daya di Athena.

Mengoptimalkan bergabung

Ada banyak strategi berbeda untuk mengeksekusi gabungan dalam mesin kueri terdistribusi. Dua yang paling umum adalah gabungan hash terdistribusi dan kueri dengan kondisi gabungan yang kompleks.

Gabung hash terdistribusi

Jenis gabungan yang paling umum menggunakan perbandingan kesetaraan sebagai kondisi gabungan. Athena menjalankan jenis join ini sebagai gabungan hash terdistribusi.

Dalam gabungan hash terdistribusi, mesin membangun tabel pencarian (tabel hash) dari salah satu sisi gabungan. Sisi ini disebut sisi build. Catatan sisi build didistribusikan di seluruh node. Setiap node membangun tabel pencarian untuk subsetnya. Sisi lain dari gabungan, yang disebut sisi probe, kemudian dialirkan melalui node. Catatan dari sisi probe didistribusikan di atas node dengan cara yang sama seperti sisi build. Ini memungkinkan setiap node untuk melakukan gabungan dengan mencari catatan yang cocok di tabel pencariannya sendiri.

Saat tabel pencarian yang dibuat dari sisi build join tidak sesuai dengan memori, kueri bisa gagal. Bahkan jika ukuran total sisi build kurang dari memori yang tersedia, kueri dapat gagal jika distribusi catatan memiliki kemiringan yang signifikan. Dalam kasus ekstrim, semua catatan dapat memiliki nilai yang sama untuk kondisi gabungan dan harus masuk ke dalam memori pada satu node. Bahkan kueri dengan sedikit kemiringan dapat gagal jika satu set nilai dikirim ke node yang sama dan nilainya bertambah hingga lebih dari memori yang tersedia. Node memang memiliki kemampuan untuk menumpahkan catatan ke disk, tetapi tumpahan memperlambat eksekusi kueri dan tidak cukup untuk mencegah kueri gagal.

Athena mencoba menyusun ulang gabungan untuk menggunakan relasi yang lebih besar sebagai sisi probe, dan relasi yang lebih kecil sebagai sisi build. Namun, karena Athena tidak mengelola data dalam tabel, ia memiliki informasi yang terbatas dan sering harus mengasumsikan bahwa tabel pertama lebih besar dan tabel kedua lebih kecil.

Saat menulis gabungan dengan kondisi gabungan berbasis ekualitas, asumsikan bahwa tabel di sebelah kiri `JOIN` kata kunci adalah sisi probe dan tabel di sebelah kanan adalah sisi build. Pastikan bahwa tabel yang tepat, sisi build, adalah tabel yang lebih kecil. Jika tidak memungkinkan untuk membuat sisi build dari join cukup kecil agar sesuai dengan memori, pertimbangkan untuk menjalankan beberapa kueri yang menggabungkan subset tabel build.

Jenis bergabung lainnya

Kueri dengan kondisi gabungan yang kompleks (misalnya, kueri yang menggunakan`LIKE`,`>`, atau operator lain), seringkali menuntut komputasi. Dalam kasus terburuk, setiap catatan dari satu sisi bergabung harus dibandingkan dengan setiap catatan di sisi lain dari bergabung. Karena waktu eksekusi tumbuh dengan kuadrat jumlah catatan, kueri tersebut berisiko melebihi waktu eksekusi maksimum.

Untuk mengetahui bagaimana Athena akan menjalankan kueri Anda sebelumnya, Anda dapat menggunakan pernyataan tersebut. `EXPLAIN` Untuk informasi selengkapnya, lihat Menggunakan JELASKAN dan JELASKAN ANALISIS di Athena dan Memahami Athena MENJELASKAN hasil pernyataan.

Mengoptimalkan fungsi jendela

Karena fungsi jendela adalah operasi intensif sumber daya, mereka dapat membuat kueri berjalan lambat atau bahkan gagal dengan pesan Sumber daya yang habis Kueri pada faktor skala ini. Fungsi jendela menyimpan semua catatan yang mereka operasikan dalam memori untuk menghitung hasilnya. Ketika jendela sangat besar, fungsi jendela bisa kehabisan memori.

Untuk memastikan kueri Anda berjalan dalam batas memori yang tersedia, kurangi ukuran jendela tempat fungsi jendela Anda beroperasi. Untuk melakukannya, Anda dapat menambahkan `PARTITIONED BY` klausa atau mempersempit cakupan klausa partisi yang ada.

Gunakan fungsi non-jendela sebagai gantinya

Terkadang kueri dengan fungsi jendela dapat ditulis ulang tanpa fungsi jendela. Misalnya, alih-alih menggunakan `row_number` untuk menemukan `N` catatan teratas, Anda dapat menggunakan `ORDER BY` dan`LIMIT`. Alih-alih menggunakan `row_number` atau menghapus `rank` duplikat catatan, Anda dapat menggunakan fungsi agregat seperti max_by, min_by, dan arbitrer.

Misalnya, Anda memiliki kumpulan data dengan pembaruan dari sensor. Sensor secara berkala melaporkan status baterainya dan menyertakan beberapa metadata seperti lokasi. Jika Anda ingin mengetahui status baterai terakhir untuk setiap sensor dan lokasinya, Anda dapat menggunakan kueri ini:

```
SELECT sensor_id,
       arbitrary(location) AS location,
       max_by(battery_status, updated_at) AS battery_status
FROM sensor_readings
GROUP BY sensor_id
```

Karena metadata seperti lokasi sama untuk setiap catatan, Anda dapat menggunakan `arbitrary` fungsi untuk memilih nilai apa pun dari grup.

Untuk mendapatkan status baterai terakhir, Anda dapat menggunakan `max_by` fungsi ini. `max_by`Fungsi memilih nilai untuk kolom dari catatan di mana nilai maksimum kolom lain ditemukan. Dalam hal ini, ia mengembalikan status baterai untuk catatan dengan waktu pembaruan terakhir dalam grup. Kueri ini berjalan lebih cepat dan menggunakan lebih sedikit memori daripada kueri setara dengan fungsi jendela.

Mengoptimalkan agregasi

Ketika Athena melakukan agregasi, Athena mendistribusikan catatan di seluruh node pekerja menggunakan kolom dalam klausa. `GROUP BY` Untuk membuat tugas mencocokkan catatan ke grup seefisien mungkin, node berusaha menyimpan catatan dalam memori tetapi menumpahkannya ke disk jika perlu.

Ini juga merupakan ide yang baik untuk menghindari memasukkan kolom berlebihan dalam `GROUP BY` klausa. Karena lebih sedikit kolom membutuhkan lebih sedikit memori, kueri yang menggambarkan grup menggunakan lebih sedikit kolom lebih efisien. Kolom numerik juga menggunakan lebih sedikit memori daripada string. Misalnya, saat Anda menggabungkan kumpulan data yang memiliki ID kategori numerik dan nama kategori, gunakan hanya kolom ID kategori dalam klausa. `GROUP BY`

Terkadang kueri menyertakan kolom dalam `GROUP BY` klausa untuk mengatasi fakta bahwa kolom harus menjadi bagian dari `GROUP BY` klausa atau ekspresi agregat. Jika aturan ini tidak diikuti, Anda dapat menerima pesan galat seperti berikut:

EXPRESSION_NOT_AGGREGATE: baris 1:8: 'kategori' harus berupa ekspresi agregat atau muncul di klausa GROUP BY

Untuk menghindari keharusan menambahkan kolom redundan ke `GROUP BY` klausa, Anda dapat menggunakan fungsi arbitrer, seperti pada contoh berikut.

```
SELECT country_id,
       arbitrary(country_name) AS country_name,
       COUNT(*) AS city_count
FROM world_cities
GROUP BY country_id
```

`ARBITRARY`Fungsi mengembalikan nilai arbitrer dari grup. Fungsi ini berguna ketika Anda tahu semua catatan dalam grup memiliki nilai yang sama untuk kolom, tetapi nilainya tidak mengidentifikasi grup.

Mengoptimalkan kueri N teratas

`ORDER BY`Klausa mengembalikan hasil query dalam urutan diurutkan. Athena menggunakan pengurutan terdistribusi untuk menjalankan operasi pengurutan secara paralel pada beberapa node.

Jika Anda tidak benar-benar membutuhkan hasil Anda untuk diurutkan, hindari menambahkan `ORDER BY` klausa. Selain itu, hindari `ORDER BY` menambahkan kueri batin jika tidak benar-benar diperlukan.

Dalam banyak kasus, perencana kueri dapat menghapus penyortiran yang berlebihan, tetapi ini tidak dijamin. Pengecualian untuk aturan ini adalah jika kueri dalam melakukan `N` operasi teratas, seperti menemukan nilai terbaru, atau `N` paling umum. `N`

Ketika Athena melihat `ORDER BY` bersama`LIMIT`, ia memahami bahwa Anda menjalankan `N` kueri teratas dan menggunakan operasi khusus yang sesuai.

> **Note**
>
> Meskipun Athena juga sering dapat mendeteksi fungsi jendela seperti `row_number` itu menggunakan top`N`, kami merekomendasikan versi yang lebih sederhana yang menggunakan `ORDER BY` dan. `LIMIT` Untuk informasi selengkapnya, lihat Mengoptimalkan fungsi jendela.

Sertakan hanya kolom yang diperlukan

Jika Anda tidak benar-benar membutuhkan kolom, jangan sertakan dalam kueri Anda. Semakin sedikit data yang harus diproses oleh kueri, semakin cepat ia akan berjalan. Ini mengurangi jumlah memori yang dibutuhkan dan jumlah data yang harus dikirim antar node. Jika Anda menggunakan format file kolumnar, mengurangi kolom angka juga mengurangi jumlah data yang dibaca dari Amazon S3.

Athena tidak memiliki batasan spesifik pada jumlah kolom dalam hasil, tetapi bagaimana kueri dijalankan membatasi kemungkinan ukuran gabungan kolom. Ukuran gabungan kolom mencakup nama dan jenisnya.

Misalnya, kesalahan berikut disebabkan oleh relasi yang melebihi batas ukuran untuk deskriptor relasi:

GENERIC_INTERNAL_ERROR: io.airlift.bytecode. CompilationException

Untuk mengatasi masalah ini, kurangi jumlah kolom dalam kueri, atau buat subkueri dan gunakan `JOIN` yang mengambil sejumlah kecil data. Jika Anda memiliki kueri yang dilakukan `SELECT *` di kueri terluar, Anda harus mengubah `*` ke daftar hanya kolom yang Anda butuhkan.

Mengoptimalkan kueri dengan menggunakan perkiraan

Athena memiliki dukungan untuk fungsi agregat aproksimasi untuk menghitung nilai yang berbeda, nilai yang paling sering, persentil (termasuk perkiraan median), dan membuat histogram. Gunakan fungsi-fungsi ini setiap kali nilai yang tepat tidak diperlukan.

Tidak seperti `COUNT(DISTINCT col)` operasi, approx_distinct menggunakan lebih sedikit memori dan berjalan lebih cepat. Demikian pula, menggunakan numeric_histogram alih-alih histogram menggunakan metode perkiraan dan karenanya lebih sedikit memori.

Mengoptimalkan LIKE

Anda dapat menggunakan `LIKE` untuk menemukan string yang cocok, tetapi dengan string panjang, ini adalah komputasi intensif. Fungsi regexp_like dalam banyak kasus merupakan alternatif yang lebih cepat, dan juga memberikan lebih banyak fleksibilitas.

Seringkali Anda dapat mengoptimalkan pencarian dengan menambatkan substring yang Anda cari. *Misalnya, jika Anda mencari awalan, jauh lebih baik menggunakan 'substr %' daripada '% substr %'.* Atau, jika Anda menggunakan`regexp_like`, '^ *substr* '.

Gunakan UNION ALL alih-alih UNION

`UNION ALL`dan `UNION` dua cara untuk menggabungkan hasil dari dua query menjadi satu hasil. `UNION ALL`menggabungkan catatan dari kueri pertama dengan yang kedua, dan `UNION` melakukan hal yang sama, tetapi juga menghapus duplikat. `UNION`perlu memproses semua catatan dan menemukan duplikat, yang merupakan memori dan komputasi intensif, tetapi `UNION ALL` merupakan operasi yang relatif cepat. Kecuali Anda perlu menghapus duplikat catatan, gunakan `UNION ALL` untuk kinerja terbaik.

Gunakan UNLOAD untuk set hasil besar

Ketika hasil kueri diharapkan besar (misalnya, puluhan ribu baris atau lebih), gunakan UNLOAD untuk mengekspor hasilnya. Dalam kebanyakan kasus, ini lebih cepat daripada menjalankan kueri biasa, dan menggunakan `UNLOAD` juga memberi Anda lebih banyak kontrol atas output.

Saat kueri selesai dijalankan, Athena menyimpan hasilnya sebagai satu file CSV yang tidak terkompresi di Amazon S3. Ini membutuhkan waktu lebih lama dari`UNLOAD`, bukan hanya karena hasilnya tidak terkompresi, tetapi juga karena operasi tidak dapat diparalelkan. Sebaliknya, `UNLOAD` menulis hasil langsung dari node pekerja dan memanfaatkan sepenuhnya paralelisme cluster komputasi. Selain itu, Anda dapat mengonfigurasi `UNLOAD` untuk menulis hasil dalam format terkompresi dan dalam format file lain seperti JSON dan Parket.

Untuk informasi selengkapnya, lihat MEMBONGKAR.

Gunakan CTAS atau Glue ETL untuk mewujudkan agregasi yang sering digunakan

'Mewujudkan' kueri adalah cara mempercepat kinerja kueri dengan menyimpan hasil kueri kompleks yang telah dihitung sebelumnya (misalnya, agregasi dan gabungan) untuk digunakan kembali dalam kueri berikutnya.

Jika banyak kueri Anda menyertakan gabungan dan agregasi yang sama, Anda dapat mewujudkan subquery umum sebagai tabel baru dan kemudian menjalankan kueri terhadap tabel tersebut. Anda dapat membuat tabel baru dengan Membuat tabel dari hasil query (CTAS), atau alat ETL khusus seperti Glue ETL.

Misalnya, Anda memiliki dasbor dengan widget yang menunjukkan aspek berbeda dari kumpulan data pesanan. Setiap widget memiliki kueri sendiri, tetapi semua kueri berbagi gabungan dan filter yang sama. Tabel pesanan digabungkan dengan tabel item baris, dan ada filter untuk ditampilkan hanya tiga bulan terakhir. Jika Anda mengidentifikasi fitur umum dari kueri ini, Anda dapat membuat tabel baru yang dapat digunakan widget. Ini mengurangi duplikasi dan meningkatkan kinerja. Kerugiannya adalah Anda harus memperbarui tabel baru.

Gunakan kembali hasil kueri

Biasanya kueri yang sama berjalan beberapa kali dalam durasi singkat. Misalnya, ini dapat terjadi ketika beberapa orang membuka dasbor data yang sama. Saat menjalankan kueri, Anda dapat memberi tahu Athena untuk menggunakan kembali hasil yang dihitung sebelumnya. Anda menentukan usia maksimum hasil yang akan digunakan kembali. Jika kueri yang sama sebelumnya dijalankan dalam jangka waktu tersebut, Athena mengembalikan hasil tersebut alih-alih menjalankan kueri lagi. Untuk informasi selengkapnya, lihat di Menggunakan kembali hasil kueri sini di Panduan Pengguna Amazon Athena dan Kurangi biaya serta tingkatkan kinerja kueri dengan Penggunaan Kembali Hasil Kueri Amazon Athena di AWS Blog Big Data.

Teknik pengoptimalan data

Kinerja tidak hanya bergantung pada kueri, tetapi juga penting pada bagaimana dataset Anda diatur dan pada format file dan kompresi yang digunakannya.

Partisi data Anda

Partisi membagi tabel Anda menjadi beberapa bagian dan menyimpan data terkait bersama-sama berdasarkan properti seperti tanggal, negara, atau wilayah. Tombol partisi bertindak sebagai kolom virtual. Anda menentukan kunci partisi pada pembuatan tabel dan menggunakannya untuk memfilter

kueri Anda. Saat Anda memfilter kolom kunci partisi, hanya data dari partisi yang cocok yang dibaca. Misalnya, jika kumpulan data Anda dipartisi berdasarkan tanggal dan kueri Anda memiliki filter yang hanya cocok minggu lalu, hanya data untuk minggu terakhir yang dibaca. Untuk informasi selengkapnya tentang partisi, lihat. Partisi data di Athena

Pilih kunci partisi yang akan mendukung kueri Anda

Karena partisi memiliki dampak signifikan pada kinerja kueri, pastikan untuk mempertimbangkan bagaimana Anda mempartisi dengan hati-hati ketika Anda mendesain dataset dan tabel Anda. Memiliki terlalu banyak kunci partisi dapat mengakibatkan dataset terfragmentasi dengan terlalu banyak file dan file yang terlalu kecil. Sebaliknya, memiliki terlalu sedikit kunci partisi, atau tidak ada partisi sama sekali, mengarah ke kueri yang memindai lebih banyak data daripada yang diperlukan.

Hindari mengoptimalkan kueri langka

Strategi yang baik adalah mengoptimalkan kueri yang paling umum dan menghindari pengoptimalan untuk kueri langka. Misalnya, jika kueri Anda melihat rentang waktu hari, jangan partisi berdasarkan jam, bahkan jika beberapa kueri memfilter ke tingkat itu. Jika data Anda memiliki kolom timestamp granular, kueri langka yang memfilter berdasarkan jam dapat menggunakan kolom stempel waktu. Bahkan jika kasus yang jarang memindai sedikit lebih banyak data daripada yang diperlukan, mengurangi kinerja keseluruhan demi kasus yang jarang terjadi biasanya bukan tradeoff yang baik.

Untuk mengurangi jumlah data yang harus dipindai oleh kueri, dan dengan demikian meningkatkan kinerja, gunakan format file kolumnar dan simpan catatan yang diurutkan. Alih-alih mempartisi berdasarkan jam, simpan catatan yang diurutkan berdasarkan stempel waktu. Untuk kueri pada jendela waktu yang lebih pendek, pengurutan berdasarkan stempel waktu hampir sama efisiennya dengan mempartisi berdasarkan jam. Selain itu, penyortiran berdasarkan stempel waktu biasanya tidak merusak kinerja kueri pada jendela waktu yang dihitung dalam beberapa hari. Untuk informasi selengkapnya, lihat Gunakan format file kolumnar.

Perhatikan bahwa kueri pada tabel dengan puluhan ribu partisi berkinerja lebih baik jika ada predikat pada semua kunci partisi. Ini adalah alasan lain untuk merancang skema partisi Anda untuk kueri yang paling umum. Untuk informasi selengkapnya, lihat Partisi kueri berdasarkan kesetaraan.

Gunakan proyeksi partisi

Proyeksi partisi adalah fitur Athena yang menyimpan informasi partisi tidak AWS Glue Data Catalog di, tetapi sebagai aturan dalam properti tabel di. AWS Glue Ketika Athena merencanakan kueri pada tabel yang dikonfigurasi dengan proyeksi partisi, Athena membaca aturan proyeksi partisi

tabel. Athena menghitung partisi untuk dibaca dalam memori berdasarkan kueri dan aturan alih-alih mencari partisi di. AWS Glue Data Catalog

Selain menyederhanakan manajemen partisi, proyeksi partisi dapat meningkatkan kinerja untuk dataset yang memiliki sejumlah besar partisi. Ketika kueri menyertakan rentang alih-alih nilai spesifik untuk kunci partisi, mencari partisi yang cocok dalam katalog membutuhkan waktu lebih lama semakin banyak partisi yang ada. Dengan proyeksi partisi, filter dapat dihitung dalam memori tanpa masuk ke katalog, dan bisa jauh lebih cepat.

Dalam keadaan tertentu, proyeksi partisi dapat menghasilkan kinerja yang lebih buruk. Salah satu contoh terjadi ketika sebuah tabel “jarang.” Tabel sparse tidak memiliki data untuk setiap permutasi nilai kunci partisi yang dijelaskan oleh konfigurasi proyeksi partisi. Dengan tabel jarang, kumpulan partisi yang dihitung dari kueri dan konfigurasi proyeksi partisi semuanya terdaftar di Amazon S3 bahkan ketika mereka tidak memiliki data.

Bila Anda menggunakan proyeksi partisi, pastikan untuk menyertakan predikat pada semua kunci partisi. Persempit cakupan nilai yang mungkin untuk menghindari daftar Amazon S3 yang tidak perlu. Bayangkan kunci partisi yang memiliki kisaran satu juta nilai dan kueri yang tidak memiliki filter pada kunci partisi itu. Untuk menjalankan kueri, Athena harus melakukan setidaknya satu juta operasi daftar Amazon S3. Kueri tercepat ketika Anda menanyakan nilai tertentu, terlepas dari apakah Anda menggunakan proyeksi partisi atau menyimpan informasi partisi dalam katalog. Untuk informasi selengkapnya, lihat Partisi kueri berdasarkan kesetaraan.

Saat Anda mengonfigurasi tabel untuk proyeksi partisi, pastikan rentang yang Anda tentukan masuk akal. Jika kueri tidak menyertakan predikat pada kunci partisi, semua nilai dalam rentang untuk kunci tersebut digunakan. Jika kumpulan data Anda dibuat pada tanggal tertentu, gunakan tanggal tersebut sebagai titik awal untuk rentang tanggal apa pun. Gunakan `NOW` sebagai rentang akhir tanggal. Hindari rentang numerik yang memiliki sejumlah besar nilai, dan pertimbangkan untuk menggunakan tipe yang disuntikkan sebagai gantinya.

Untuk informasi selengkapnya tentang proyek partisi, lihat Proyeksi partisi dengan Amazon Athena.

Gunakan indeks partisi

Indeks partisi adalah fitur dalam AWS Glue Data Catalog yang meningkatkan kinerja pencarian partisi untuk tabel yang memiliki sejumlah besar partisi.

Daftar partisi dalam katalog seperti tabel dalam database relasional. Tabel memiliki kolom untuk kunci partisi dan kolom tambahan untuk lokasi partisi. Saat Anda menanyakan tabel yang dipartisi, lokasi partisi dicari dengan memindai tabel ini.

Sama seperti database relasional, Anda dapat meningkatkan kinerja kueri dengan menambahkan indeks. Anda dapat menambahkan beberapa indeks untuk mendukung pola kueri yang berbeda. Indeks AWS Glue Data Catalog partisi mendukung operator kesetaraan dan perbandingan seperti>,>=, dan < dikombinasikan dengan `AND` operator. Untuk informasi selengkapnya, lihat Bekerja dengan indeks partisi AWS Glue di Panduan AWS Glue Pengembang dan Meningkatkan kinerja kueri Amazon Athena AWS Glue Data Catalog menggunakan indeks partisi di AWS Blog Big Data.

Selalu gunakan STRING sebagai tipe untuk kunci partisi

Saat Anda menanyakan kunci partisi, ingatlah bahwa Athena memerlukan kunci partisi untuk menjadi tipe untuk menekan pemfilteran partisi ke `STRING` dalam. AWS Glue Jika jumlah partisi tidak sedikit, menggunakan jenis lain dapat menyebabkan kinerja yang lebih buruk. Jika nilai kunci partisi Anda seperti tanggal atau seperti angka, lemparkan ke jenis yang sesuai dalam kueri Anda.

Hapus partisi lama dan kosong

Jika Anda menghapus data dari partisi di Amazon S3 (misalnya, dengan menggunakan siklus hidup Amazon S3), Anda juga harus menghapus entri partisi dari. AWS Glue Data Catalog Selama perencanaan kueri, partisi apa pun yang cocok dengan kueri terdaftar di Amazon S3. Jika Anda memiliki banyak partisi kosong, overhead daftar partisi ini dapat merugikan.

Juga, jika Anda memiliki ribuan partisi, pertimbangkan untuk menghapus metadata partisi untuk data lama yang tidak lagi relevan. Misalnya, jika kueri tidak pernah melihat data yang lebih tua dari setahun, Anda dapat menghapus metadata partisi secara berkala untuk partisi yang lebih lama. Jika jumlah partisi bertambah menjadi puluhan ribu, menghapus partisi yang tidak terpakai dapat mempercepat kueri yang tidak menyertakan predikat pada semua kunci partisi. Untuk informasi tentang menyertakan predikat pada semua kunci partisi dalam kueri Anda, lihat. Partisi kueri berdasarkan kesetaraan

Partisi kueri berdasarkan kesetaraan

Kueri yang menyertakan predikat kesetaraan pada semua kunci partisi berjalan lebih cepat karena metadata partisi dapat dimuat secara langsung. Hindari kueri di mana satu atau lebih kunci partisi tidak memiliki predikat, atau predikat memilih rentang nilai. Untuk kueri seperti itu, daftar semua partisi harus disaring untuk menemukan nilai yang cocok. Untuk sebagian besar tabel, overhead minimal, tetapi untuk tabel dengan puluhan ribu atau lebih partisi, overhead bisa menjadi signifikan.

Jika tidak mungkin untuk menulis ulang kueri Anda untuk memfilter partisi berdasarkan kesetaraan, Anda dapat mencoba proyeksi partisi. Untuk informasi selengkapnya, lihat Gunakan proyeksi partisi.

Hindari menggunakan MSCK REPAIR TABLE untuk pemeliharaan partisi

Karena `MSCK REPAIR TABLE` bisa memakan waktu lama untuk dijalankan, hanya menambahkan partisi baru, dan tidak menghapus partisi lama, itu bukan cara yang efisien untuk mengelola partisi (lihat). Pertimbangan dan batasan

Partisi lebih baik dikelola secara manual menggunakan AWS Glue Data Catalog API,ALTER TABLE ADD PARTITION, atau AWS Glue crawler. Sebagai alternatif, Anda dapat menggunakan proyeksi partisi, yang menghilangkan kebutuhan untuk mengelola partisi sama sekali. Untuk informasi selengkapnya, lihat Proyeksi partisi dengan Amazon Athena.

Validasi bahwa kueri Anda kompatibel dengan skema partisi

Anda dapat memeriksa terlebih dahulu partisi mana yang akan dipindai kueri dengan menggunakan `EXPLAIN`pernyataan tersebut. Awali kueri Anda dengan `EXPLAIN` kata kunci, lalu cari fragmen sumber (misalnya,`Fragment 2 [SOURCE]`) untuk setiap tabel di dekat bagian bawah output. `EXPLAIN` Cari tugas di mana sisi kanan didefinisikan sebagai kunci partisi. Baris di bawahnya menyertakan daftar semua nilai untuk kunci partisi yang akan dipindai saat kueri dijalankan.

Misalnya, Anda memiliki kueri di atas meja dengan kunci `dt` partisi dan awalan kueri dengan`EXPLAIN`. Jika nilai dalam kueri adalah tanggal, dan filter memilih rentang tiga hari, `EXPLAIN` output mungkin terlihat seperti ini:

```
dt := dt:string:PARTITION_KEY
    :: [[2023-06-11], [2023-06-12], [2023-06-13]]
```

`EXPLAIN`Output menunjukkan bahwa perencana menemukan tiga nilai untuk kunci partisi ini yang cocok dengan kueri. Ini juga menunjukkan kepada Anda apa nilai-nilai itu. Untuk informasi selengkapnya tentang penggunaan`EXPLAIN`, lihat Menggunakan JELASKAN dan JELASKAN ANALISIS di Athena dan Memahami Athena MENJELASKAN hasil pernyataan.

Gunakan format file kolumnar

Format file kolumnar seperti Parket dan ORC dirancang untuk beban kerja analitik terdistribusi. Mereka mengatur data berdasarkan kolom, bukan berdasarkan baris. Mengatur data dalam format kolumnar menawarkan keuntungan sebagai berikut:

- Hanya kolom yang diperlukan untuk kueri yang dimuat
- Jumlah keseluruhan data yang perlu dimuat berkurang

- Nilai kolom disimpan bersama, sehingga data dapat dikompresi secara efisien
- File dapat berisi metadata yang memungkinkan mesin melewati pemuatan data yang tidak dibutuhkan

Sebagai contoh bagaimana metadata file dapat digunakan, metadata file dapat berisi informasi tentang nilai minimum dan maksimum dalam halaman data. Jika nilai yang ditanyakan tidak berada dalam rentang yang dicatat dalam metadata, halaman dapat dilewati.

Salah satu cara untuk menggunakan metadata ini untuk meningkatkan kinerja adalah memastikan bahwa data dalam file diurutkan. Misalnya, Anda memiliki kueri yang mencari catatan di mana `created_at` entri berada dalam rentang waktu yang singkat. Jika data Anda diurutkan berdasarkan `created_at` kolom, Athena dapat menggunakan nilai minimum dan maksimum dalam metadata file untuk melewati bagian file data yang tidak diperlukan.

Saat menggunakan format file kolumnar, pastikan file Anda tidak terlalu kecil. Seperti disebutkan dalamHindari memiliki terlalu banyak file, kumpulan data dengan banyak file kecil menyebabkan masalah kinerja. Hal ini terutama berlaku dengan format file kolumnar. Untuk file kecil, overhead format file kolumnar melebihi manfaatnya.

Perhatikan bahwa Parket dan ORC diatur secara internal oleh kelompok baris (Parket) dan garis (ORC). Ukuran default untuk grup baris adalah 128 MB, dan untuk garis, 64 MB. Jika Anda memiliki banyak kolom, Anda dapat meningkatkan grup baris dan ukuran garis untuk kinerja yang lebih baik. Mengurangi grup baris atau ukuran garis menjadi kurang dari nilai defaultnya tidak disarankan.

Untuk mengonversi format data lain ke Parket atau ORC, Anda dapat menggunakan AWS Glue ETL atau Athena. Untuk informasi lebih lanjut tentang menggunakan Athena untuk ETL, lihat. Menggunakan CTAS dan INSERT INTO untuk ETL dan analisis data

Kompres data

Athena mendukung berbagai format kompresi. Menanyakan data terkompresi lebih cepat dan juga lebih murah karena Anda membayar jumlah byte yang dipindai sebelum dekompresi.

Format gzip memberikan rasio kompresi yang baik dan memiliki berbagai dukungan di seluruh alat dan layanan lainnya. Format zstd (Zstandard) adalah format kompresi yang lebih baru dengan keseimbangan yang baik antara kinerja dan rasio kompresi.

Saat mengompresi file teks seperti data JSON dan CSV, cobalah untuk mencapai keseimbangan antara jumlah file dan ukuran file. Sebagian besar format kompresi mengharuskan pembaca untuk

membaca file dari awal. Ini berarti bahwa file teks terkompresi tidak dapat, secara umum, diproses secara paralel. File besar yang tidak terkompresi sering dibagi antara pekerja untuk mencapai paralelisme yang lebih tinggi selama pemrosesan kueri, tetapi ini tidak mungkin dilakukan dengan sebagian besar format kompresi.

Seperti yang dibahas dalamHindari memiliki terlalu banyak file, lebih baik tidak memiliki terlalu banyak file atau terlalu sedikit. Karena jumlah file adalah batas berapa banyak pekerja yang dapat memproses kueri, aturan ini terutama berlaku untuk file terkompresi.

Untuk informasi lebih lanjut tentang penggunaan kompresi di Athena, lihat. Dukungan kompresi Athena

Gunakan bucketing untuk pencarian pada tombol dengan kardinalitas tinggi

Bucketing adalah teknik untuk mendistribusikan catatan ke dalam file terpisah berdasarkan nilai salah satu kolom. Ini memastikan bahwa semua catatan dengan nilai yang sama akan berada dalam file yang sama. Bucketing berguna ketika Anda memiliki kunci dengan kardinalitas tinggi dan banyak kueri Anda mencari nilai spesifik dari kunci tersebut.

Misalnya, Anda menanyakan satu set catatan untuk pengguna tertentu. Jika data diselimuti oleh ID pengguna, Athena tahu sebelumnya file mana yang berisi catatan untuk ID tertentu dan file mana yang tidak. Hal ini memungkinkan Athena untuk membaca hanya file yang dapat berisi ID, sangat mengurangi jumlah data yang dibaca. Ini juga mengurangi waktu komputasi yang jika tidak akan diperlukan untuk mencari melalui data untuk ID tertentu.

Kerugian dari bucketing

Bucketing kurang berharga ketika kueri sering mencari beberapa nilai di kolom yang dikandung oleh data. Semakin banyak nilai yang ditanyakan, semakin tinggi kemungkinan bahwa semua atau sebagian besar file harus dibaca. Misalnya, jika Anda memiliki tiga bucket, dan kueri mencari tiga nilai yang berbeda, semua file mungkin harus dibaca. Bucketing berfungsi paling baik saat kueri mencari nilai tunggal.

Untuk informasi selengkapnya, lihat Partisi dan bucketing di Athena.

Hindari memiliki terlalu banyak file

Kumpulan data yang terdiri dari banyak file kecil menghasilkan kinerja kueri keseluruhan yang buruk. Ketika Athena merencanakan kueri, itu mencantumkan semua lokasi partisi, yang membutuhkan waktu. Menangani dan meminta setiap file juga memiliki overhead komputasi. Oleh karena itu,

memuat satu file yang lebih besar dari Amazon S3 lebih cepat daripada memuat catatan yang sama dari banyak file yang lebih kecil.

Dalam kasus ekstrim, Anda mungkin mengalami batas layanan Amazon S3. Amazon S3 mendukung hingga 5.500 permintaan per detik untuk satu partisi indeks. Awalnya, bucket diperlakukan sebagai partisi indeks tunggal, tetapi ketika beban permintaan meningkat, itu dapat dibagi menjadi beberapa partisi indeks.

Amazon S3 melihat pola permintaan dan pemisahan berdasarkan awalan kunci. Jika dataset Anda terdiri dari ribuan file, permintaan yang berasal dari Athena dapat melebihi kuota permintaan. Bahkan dengan file yang lebih sedikit, kuota dapat dilampaui jika beberapa kueri bersamaan dibuat terhadap kumpulan data yang sama. Aplikasi lain yang mengakses file yang sama dapat berkontribusi pada jumlah total permintaan.

Ketika tingkat permintaan `limit` terlampaui, Amazon S3 mengembalikan kesalahan berikut. Kesalahan ini termasuk dalam informasi status untuk kueri di Athena.

SlowDown: Harap kurangi tingkat permintaan Anda

Untuk memecahkan masalah, mulailah dengan menentukan apakah kesalahan disebabkan oleh satu kueri atau oleh beberapa kueri yang membaca file yang sama. Jika yang terakhir, koordinasikan menjalankan kueri sehingga tidak berjalan pada saat yang sama. Untuk mencapai ini, tambahkan mekanisme antrian atau bahkan coba lagi di aplikasi Anda.

Jika menjalankan satu kueri memicu kesalahan, coba gabungkan file data atau modifikasi kueri untuk membaca lebih sedikit file. Waktu terbaik untuk menggabungkan file kecil adalah sebelum ditulis. Untuk melakukannya, pertimbangkan teknik-teknik berikut:

- Ubah proses yang menulis file untuk menulis file yang lebih besar. Misalnya, Anda dapat menyangga catatan untuk waktu yang lebih lama sebelum ditulis.
- Letakkan file di lokasi di Amazon S3 dan gunakan alat seperti Glue ETL untuk menggabungkannya menjadi file yang lebih besar. Kemudian, pindahkan file yang lebih besar ke lokasi yang ditunjukkan tabel. Untuk informasi selengkapnya, lihat Membaca file input dalam grup yang lebih besar di Panduan AWS Glue Pengembang dan Bagaimana cara mengonfigurasi pekerjaan AWS Glue ETL untuk menampilkan file yang lebih besar? di Pusat Pengetahuan AWS RE: Post.
- Kurangi jumlah tombol partisi. Ketika Anda memiliki terlalu banyak kunci partisi, setiap partisi mungkin hanya memiliki beberapa catatan, menghasilkan jumlah file kecil yang berlebihan. Untuk informasi tentang memutuskan partisi mana yang akan dibuat, lihatPilih kunci partisi yang akan mendukung kueri Anda.

Hindari hierarki penyimpanan tambahan di luar partisi

Untuk menghindari overhead perencanaan kueri, simpan file dalam struktur datar di setiap lokasi partisi. Jangan gunakan hierarki direktori tambahan apa pun.

Ketika Athena merencanakan kueri, Athena mencantumkan semua file di semua partisi yang cocok dengan kueri. Meskipun Amazon S3 tidak memiliki direktori sendiri, konvensi ini adalah untuk menafsirkan garis miring ke / depan sebagai pemisah direktori. Ketika Athena mencantumkan lokasi partisi, secara rekursif mencantumkan direktori apa pun yang ditemukannya. Ketika file dalam partisi diatur ke dalam hierarki, beberapa putaran daftar terjadi.

Ketika semua file berada langsung di lokasi partisi, sebagian besar waktu hanya satu operasi daftar yang harus dilakukan. Namun, beberapa operasi daftar sekuensial diperlukan jika Anda memiliki lebih dari 1000 file dalam partisi karena Amazon S3 hanya mengembalikan 1000 objek per operasi daftar. Memiliki lebih dari 1000 file dalam partisi juga dapat membuat masalah kinerja lain yang lebih serius. Untuk informasi selengkapnya, lihat Hindari memiliki terlalu banyak file.

Gunakan SymlinkTextInputFormat hanya jika diperlukan

Menggunakan SymlinkTextInputFormatteknik ini bisa menjadi cara untuk mengatasi situasi ketika file untuk tabel tidak tertata rapi ke dalam partisi. Misalnya, symlink dapat berguna ketika semua file berada dalam awalan yang sama atau file dengan skema berbeda berada di lokasi yang sama.

Namun, menggunakan symlink menambahkan tingkat indirection ke eksekusi kueri. Tingkat indirection ini berdampak pada kinerja keseluruhan. File symlink harus dibaca, dan lokasi yang mereka tentukan harus terdaftar. Ini menambahkan beberapa perjalanan pulang pergi ke Amazon S3 yang tidak diperlukan oleh tabel Hive biasa. Kesimpulannya, Anda harus menggunakan `SymlinkTextInputFormat` hanya ketika opsi yang lebih baik seperti mengatur ulang file tidak tersedia.

Sumber daya tambahan

Untuk informasi tambahan tentang penyetelan kinerja di Athena, pertimbangkan sumber daya berikut:

- Baca posting blog AWS Big Data 10 kiat penyetelan kinerja terbaik untuk Amazon Athena
- Untuk artikel tentang penggunaan pushdown predikat untuk meningkatkan kinerja dalam kueri federasi, lihat Meningkatkan kueri federasi dengan pushdown predikat di Amazon Athena di Blog Big Data.AWS

- Untuk artikel tentang pengoptimalan kinerja di mesin kueri Athena, lihat Menjalankan kueri 3x lebih cepat dengan penghematan biaya hingga 70% pada mesin Amazon Athena terbaru di Blog Big Data.AWS
- Baca posting Athena lainnya di blog data AWS besar
- Ajukan pertanyaan di AWS re:Post menggunakan tag Amazon Athena
- Konsultasikan topik Athena di pusat pengetahuan AWS
- Kontak AWS Support (di AWS Management Console, klik Support, Support Center)

Mencegah pelambatan Amazon S3

Throttling adalah proses membatasi tingkat di mana Anda menggunakan layanan, aplikasi, atau sistem. Di AWS, Anda dapat menggunakan pelambatan untuk mencegah penggunaan layanan Amazon S3 yang berlebihan dan meningkatkan ketersediaan dan daya tanggap Amazon S3 untuk semua pengguna. Namun, karena pembatasan membatasi kecepatan transfer data ke atau dari Amazon S3, penting untuk mempertimbangkan untuk mencegah interaksi Anda terhambat.

Kurangi pelambatan di tingkat layanan

Untuk menghindari pelambatan Amazon S3 pada tingkat layanan, Anda dapat memantau penggunaan dan menyesuaikan kuota layanan Anda, atau Anda menggunakan teknik tertentu seperti partisi. Berikut ini adalah beberapa kondisi yang dapat menyebabkan throttling:

- Melebihi batas permintaan API akun Anda - Amazon S3 memiliki batas permintaan API default yang didasarkan pada jenis dan penggunaan akun. Jika Anda melebihi jumlah maksimum permintaan per detik untuk satu objek, permintaan Anda mungkin dibatasi untuk mencegah kelebihan layanan Amazon S3.
- Partisi data tidak memadai - Jika Anda tidak mempartisi data dengan benar dan mentransfer sejumlah besar data, Amazon S3 dapat membatasi permintaan Anda. Untuk informasi selengkapnya tentang partisi, lihat Gunakan partisi bagian dalam dokumen ini.
- Sejumlah besar benda kecil — Jika memungkinkan, hindari memiliki sejumlah besar file kecil. Amazon S3 memiliki batas 5500 permintaan GET per detik per awalan yang dipartisi, dan kueri Athena Anda memiliki batas yang sama. Jika Anda memindai jutaan objek kecil dalam satu kueri, kueri Anda kemungkinan akan dibatasi oleh Amazon S3.

Untuk menghindari pemindaian berlebih, Anda dapat menggunakan AWS Glue ETL untuk memadatkan file Anda secara berkala, atau Anda mempartisi tabel dan menambahkan filter kunci partisi. Untuk informasi selengkapnya, lihat sumber daya berikut ini.

- Bagaimana saya bisa mengonfigurasi pekerjaan AWS Glue ETL untuk mengeluarkan file yang lebih besar? (Pusat AWS Pengetahuan)
- Membaca file input dalam kelompok yang lebih besar (Panduan AWS Glue Pengembang)

Mengoptimalkan tabel Anda

Penataan data Anda penting jika Anda mengalami masalah pelambatan. Meskipun Amazon S3 dapat menangani data dalam jumlah besar, pembatasan terkadang terjadi karena cara data terstruktur.

Bagian berikut menawarkan beberapa saran tentang cara menyusun data Anda di Amazon S3 untuk menghindari masalah pembatasan.

Gunakan partisi

Anda dapat menggunakan partisi untuk mengurangi pembatasan dengan membatasi jumlah data yang harus diakses pada waktu tertentu. Dengan mempartisi data pada kolom tertentu, Anda dapat mendistribusikan permintaan secara merata di beberapa objek dan mengurangi jumlah permintaan untuk satu objek. Mengurangi jumlah data yang harus dipindai meningkatkan kinerja kueri dan menurunkan biaya.

Anda dapat menentukan partisi, yang bertindak sebagai kolom virtual, saat Anda membuat tabel. Untuk membuat tabel dengan partisi dalam `CREATE TABLE` pernyataan, Anda menggunakan `PARTITIONED BY (`*`column_name data_type`*`)` klausa untuk menentukan kunci untuk mempartisi data Anda.

Untuk membatasi partisi yang dipindai oleh kueri, Anda dapat menentukannya sebagai predikat dalam `WHERE` klausa kueri. Dengan demikian, kolom yang sering digunakan sebagai filter adalah kandidat yang baik untuk partisi. Praktik umum adalah mempartisi data berdasarkan interval waktu, yang dapat menyebabkan skema partisi multi-level.

Perhatikan bahwa partisi juga memiliki biaya. Ketika Anda meningkatkan jumlah partisi dalam tabel Anda, waktu yang diperlukan untuk mengambil dan memproses metadata partisi juga meningkat. Dengan demikian, partisi berlebihan dapat menghilangkan manfaat yang Anda peroleh dengan mempartisi dengan lebih bijaksana. Jika data Anda sangat miring ke satu nilai partisi, dan sebagian besar kueri menggunakan nilai itu, maka Anda mungkin dikenakan biaya tambahan.

Untuk informasi lebih lanjut tentang partisi di Athena, lihat Apa itu partisi?

Bucket data Anda

Cara lain untuk mempartisi data Anda adalah dengan memasukkan data dalam satu partisi. Dengan bucketing, Anda menentukan satu atau beberapa kolom yang berisi baris yang ingin Anda kelompokkan bersama. Kemudian, Anda menempatkan baris itu ke dalam beberapa ember. Dengan cara ini, Anda hanya menanyakan bucket yang harus dibaca, yang mengurangi jumlah baris data yang harus dipindai.

Saat Anda memilih kolom yang akan digunakan untuk bucketing, pilih kolom yang memiliki kardinalitas tinggi (yaitu, yang memiliki banyak nilai berbeda), didistribusikan secara seragam, dan sering digunakan untuk memfilter data. Contoh kolom yang baik untuk digunakan untuk bucketing adalah kunci utama, seperti kolom ID.

Untuk informasi lebih lanjut tentang bucketing di Athena, lihat Apa itu bucketing?

Gunakan indeks AWS Glue partisi

Anda dapat menggunakan indeks AWS Glue partisi untuk mengatur data dalam tabel berdasarkan nilai satu atau lebih partisi. AWS Glue Indeks partisi dapat mengurangi jumlah transfer data, jumlah pemrosesan data, dan waktu untuk query untuk memproses.

Indeks AWS Glue partisi adalah file metadata yang berisi informasi tentang partisi dalam tabel, termasuk kunci partisi dan nilainya. Indeks partisi disimpan dalam bucket Amazon S3 dan diperbarui secara otomatis oleh AWS Glue saat partisi baru ditambahkan ke tabel.

Ketika indeks AWS Glue partisi hadir, query mencoba untuk mengambil subset dari partisi alih-alih memuat semua partisi dalam tabel. Query hanya berjalan pada subset data yang relevan dengan query.

Saat Anda membuat tabel di AWS Glue, Anda dapat membuat indeks partisi pada kombinasi tombol partisi yang ditentukan di atas tabel. Setelah Anda membuat satu atau lebih indeks partisi di atas meja, Anda harus menambahkan properti ke tabel yang memungkinkan pemfilteran partisi. Kemudian, Anda dapat menanyakan tabel dari Athena.

Untuk informasi tentang membuat indeks partisi AWS Glue, lihat Bekerja dengan indeks partisi AWS Glue di Panduan AWS Glue Pengembang. Untuk informasi tentang menambahkan properti tabel untuk mengaktifkan pemfilteran partisi, lihatAWS Glue pengindeksan partisi dan penyaringan.

Gunakan kompresi data dan pemisahan file

Kompresi data dapat mempercepat kueri secara signifikan jika file berada pada ukuran optimal atau jika mereka dapat dibagi menjadi kelompok-kelompok logis. Umumnya, rasio kompresi yang lebih tinggi membutuhkan lebih banyak siklus CPU untuk mengompres dan mendekompresi data. Untuk Athena, kami menyarankan Anda menggunakan Apache Parquet atau Apache ORC, yang mengompres data secara default. Untuk informasi tentang kompresi data di Athena, lihat. Dukungan kompresi Athena

Memisahkan file meningkatkan paralelisme dengan memungkinkan Athena untuk mendistribusikan tugas membaca satu file di antara banyak pembaca. Jika satu file tidak dapat dibagi, hanya satu pembaca yang dapat membaca file sementara pembaca lain menganggur. Apache Parquet dan Apache ORC juga mendukung file splittable.

Gunakan penyimpanan data kolumnar yang dioptimalkan

Kinerja kueri Athena meningkat secara signifikan jika Anda mengonversi data Anda menjadi format kolumnar. Saat Anda menghasilkan file kolumnar, salah satu teknik pengoptimalan yang perlu dipertimbangkan adalah memesan data berdasarkan kunci partisi.

Apache Parquet dan Apache ORC biasanya digunakan penyimpanan data kolumnar open source. Untuk informasi tentang mengonversi sumber data Amazon S3 yang ada ke salah satu format ini, lihat. Mengonversi ke format kolumnar

Gunakan ukuran blok Parket yang lebih besar atau ukuran garis ORC

Parket dan ORC memiliki parameter penyimpanan data yang dapat Anda atur untuk pengoptimalan. Di Parket, Anda dapat mengoptimalkan ukuran blok. Di ORC, Anda dapat mengoptimalkan ukuran garis. Semakin besar blok atau garis, semakin banyak baris yang dapat Anda simpan di masing-masing. Secara default, ukuran blok Parket adalah 128 MB, dan ukuran garis ORC adalah 64 MB.

Jika garis ORC kurang dari 8 MB (nilai default`hive.orc.max_buffer_size`), Athena membaca seluruh garis ORC. Ini adalah tradeoff yang dilakukan Athena antara selektivitas kolom dan operasi input/output per detik untuk garis-garis yang lebih kecil.

Jika Anda memiliki tabel dengan jumlah kolom yang sangat besar, ukuran blok atau garis kecil dapat menyebabkan lebih banyak data dipindai daripada yang diperlukan. Dalam kasus ini, ukuran blok yang lebih besar bisa lebih efisien.

Gunakan ORC untuk tipe kompleks

Saat ini, ketika Anda menanyakan kolom yang disimpan di Parket yang memiliki tipe data kompleks (misalnya,`array`,`map`, atau`struct`), Athena membaca seluruh baris data alih-alih membaca secara selektif hanya kolom yang ditentukan. Ini adalah masalah yang diketahui di Athena. Sebagai solusinya, pertimbangkan untuk menggunakan ORC.

Pilih algoritma kompresi

Parameter lain yang dapat Anda konfigurasikan adalah algoritma kompresi pada blok data. Untuk informasi tentang algoritma kompresi yang didukung untuk Parket dan ORC di Athena, lihat Dukungan kompresi Athena.

Untuk informasi lebih lanjut tentang optimalisasi format penyimpanan kolumnar di Athena, lihat bagian “Optimalkan pembuatan penyimpanan data kolumnar” di posting Blog AWS Big Data 10 Tips Penyetelan Kinerja Teratas untuk Amazon Athena.

Gunakan tabel Iceberg

Apache Iceberg adalah format tabel terbuka untuk kumpulan data analitik yang sangat besar yang dirancang untuk penggunaan yang dioptimalkan di Amazon S3. Anda dapat menggunakan tabel Iceberg untuk membantu mengurangi pelambatan di Amazon S3.

Tabel gunung es menawarkan keuntungan berikut:

- Anda dapat mempartisi tabel Iceberg pada satu atau lebih kolom. Ini mengoptimalkan akses data dan mengurangi jumlah data yang harus dipindai oleh kueri.
- Karena mode penyimpanan objek Iceberg mengoptimalkan tabel Iceberg agar berfungsi dengan Amazon S3, mode ini dapat memproses volume data yang besar dan beban kerja kueri yang berat.
- Tabel gunung es dalam mode penyimpanan objek dapat diskalakan, toleran terhadap kesalahan, dan tahan lama, yang dapat membantu mengurangi pelambatan.
- Dukungan transaksi ACID berarti bahwa beberapa pengguna dapat menambahkan dan menghapus objek Amazon S3 secara atom.

Untuk informasi lebih lanjut tentang Apache Iceberg, lihat Apache Iceberg. Untuk informasi selengkapnya tentang menggunakan tabel Apache Iceberg di Athena, lihat Menggunakan tabel Gunung Es.

Mengoptimalkan kueri

Gunakan saran di bagian ini untuk mengoptimalkan kueri SQL Anda di Athena.

Gunakan LIMIT dengan klausa ORDER BY

`ORDER BY`Klausa mengembalikan data dalam urutan yang diurutkan. Ini membutuhkan Athena untuk mengirim semua baris data ke node pekerja tunggal dan kemudian mengurutkan baris. Jenis kueri ini dapat berjalan untuk waktu yang lama atau bahkan gagal.

Untuk efisiensi yang lebih besar dalam kueri Anda, lihat nilai *N* atas atau bawah, dan kemudian gunakan juga `LIMIT` klausa. Ini secara signifikan mengurangi biaya pengurutan dengan mendorong penyortiran dan pembatasan ke node pekerja individu daripada ke satu pekerja.

Optimalkan klausa JOIN

Ketika Anda menggabungkan dua tabel, Athena mendistribusikan tabel di sebelah kanan ke node pekerja, dan kemudian mengalirkan tabel di sebelah kiri untuk melakukan gabungan.

Untuk alasan ini, tentukan tabel yang lebih besar di sisi kiri gabungan dan tabel yang lebih kecil di sisi kanan gabungan. Dengan cara ini, Athena menggunakan lebih sedikit memori dan menjalankan kueri dengan latensi yang lebih rendah.

Perhatikan juga poin-poin berikut:

- Saat Anda menggunakan beberapa `JOIN` perintah, tentukan tabel dari yang terbesar hingga terkecil.
- Hindari gabungan silang kecuali jika diperlukan oleh kueri.

Optimalkan klausa GROUP BY

`GROUP BY`Operator mendistribusikan baris berdasarkan `GROUP BY` kolom ke node pekerja. Kolom ini direferensikan dalam memori dan nilainya dibandingkan saat baris dicerna. Nilai-nilai dikumpulkan bersama ketika `GROUP BY` kolom cocok. Dengan mempertimbangkan cara kerja proses ini, disarankan untuk memesan kolom dari kardinalitas tertinggi ke yang terendah.

Gunakan angka, bukan string

Karena angka membutuhkan lebih sedikit memori dan lebih cepat diproses dibandingkan dengan string, gunakan angka alih-alih string jika memungkinkan.

Batasi jumlah kolom

Untuk mengurangi jumlah total memori yang diperlukan untuk menyimpan data Anda, batasi jumlah kolom yang ditentukan dalam `SELECT` pernyataan Anda.

Gunakan ekspresi reguler alih-alih LIKE

Kueri yang menyertakan klausa seperti `LIKE '%string%'` pada string besar bisa sangat intensif secara komputasi. Saat Anda memfilter beberapa nilai pada kolom string, gunakan fungsi regexp_like () dan ekspresi reguler sebagai gantinya. Ini sangat berguna ketika Anda membandingkan daftar panjang nilai.

Gunakan klausa LIMIT

Alih-alih memilih semua kolom saat Anda menjalankan kueri, gunakan `LIMIT` klausa untuk mengembalikan hanya kolom yang Anda butuhkan. Ini mengurangi ukuran kumpulan data yang diproses melalui pipeline eksekusi kueri. `LIMIT`klausa lebih membantu ketika Anda menanyakan tabel yang memiliki sejumlah besar kolom yang berbasis string. `LIMIT`klausa juga membantu ketika Anda melakukan beberapa gabungan atau agregasi pada kueri apa pun.

Sumber daya tambahan

Pola desain praktik terbaik: mengoptimalkan kinerja Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Tuning kinerja di Athena

Dukungan kompresi Athena

Topik

Athena mendukung berbagai format kompresi untuk membaca dan menulis data, termasuk membaca dari tabel yang menggunakan beberapa format kompresi. Misalnya, Athena berhasil membaca data

dalam tabel yang menggunakan format file Parket ketika beberapa file Parket dikompresi dengan Snappy dan file Parket lainnya dikompresi dengan GZIP. Prinsip yang sama berlaku untuk ORC, file teks, dan format penyimpanan JSON.

Athena mendukung format kompresi berikut:

- BZIP2 — Format yang menggunakan algoritma Burrows-Wheeler.
- DEFLATE — Algoritma kompresi berdasarkan pengkodean LZSS dan Huffman. Deflate hanya relevan untuk format file Avro.
- GZIP — Algoritma kompresi berdasarkan Deflate. Untuk tabel Hive di mesin Athena versi 2 dan 3, dan tabel Iceberg di mesin Athena versi 2, GZIP adalah format kompresi tulis default untuk file dalam format penyimpanan file Parket dan teks. File dalam `tar.gz` format tidak didukung.
- LZ4 — Anggota keluarga Lempel-Ziv 77 (LZ7) ini juga berfokus pada kecepatan kompresi dan dekompresi daripada kompresi data maksimum. LZ4 memiliki format pembingkaian berikut:
  - LZ4 Raw/Unframed — Implementasi standar format kompresi blok LZ4 yang tidak dibingkai. Untuk informasi selengkapnya, lihat deskripsi format blok LZ4 di. GitHub
  - LZ4 dibingkai - Implementasi pembingkaian LZ4 yang biasa. Untuk informasi selengkapnya, lihat deskripsi format bingkai LZ4 di. GitHub
  - LZ4 hadoop-kompatibel - Implementasi Apache Hadoop dari LZ4. Implementasi ini membungkus kompresi LZ4 dengan kelas.java. BlockCompressorStream
- LZO — Format yang menggunakan algoritma Lempel—Ziv-Oberhumer, yang berfokus pada kompresi tinggi dan kecepatan dekompresi daripada kompresi maksimum data. LZO memiliki dua implementasi:
  - LZO Standar — Untuk informasi lebih lanjut, lihat abstrak LZO di situs web Oberhumer.
  - LZO hadoop-compatible — Implementasi ini membungkus algoritma LZO dengan kelas.java. BlockCompressorStream
- SNAPPY — Algoritma kompresi yang merupakan bagian dari keluarga Lempel-Ziv 77 (LZ7). Snappy berfokus pada kecepatan kompresi dan dekompresi yang tinggi daripada kompresi maksimum data.
- ZLIB — Berdasarkan Defrate, ZLIB adalah format kompresi tulis default untuk file dalam format penyimpanan data ORC. Untuk informasi lebih lanjut, lihat halaman zlib di. GitHub
- ZSTD — Algoritma kompresi data real-time Zstandard adalah algoritma kompresi cepat yang memberikan rasio kompresi tinggi. Pustaka Zstandard (ZSTD) disediakan sebagai perangkat lunak sumber terbuka menggunakan lisensi BSD. ZSTD adalah kompresi default untuk tabel Iceberg. Saat menulis data terkompresi ZSTD, Athena menggunakan kompresi ZSTD level 3 secara

default. Untuk informasi lebih lanjut tentang menggunakan tingkat kompresi ZSTD di Athena, lihat. Menggunakan tingkat kompresi ZSTD di Athena

Menentukan format kompresi

Saat Anda menulis pernyataan CREATE TABLE atau CTAS, Anda dapat menentukan properti kompresi yang menentukan jenis kompresi yang akan digunakan saat Athena menulis ke tabel tersebut.

- Untuk CTAS, lihatProperti tabel CTAS. Sebagai contoh, lihat Contoh kueri CTAS.
- Untuk CREATE TABLE, lihat ALTER TABLE SET TBLPROPERTIES daftar properti tabel kompresi.

Menentukan tidak ada kompresi

Pernyataan CREATE TABLE mendukung penulisan file yang tidak terkompresi. Untuk menulis file yang tidak terkompresi, gunakan sintaks berikut:

- BUAT TABEL (file teks atau JSON) - Dalam`TBLPROPERTIES`, tentukan`write.compression = NONE`.
- BUAT TABEL (Parket) — Dalam`TBLPROPERTIES`, tentukan`parquet.compression = UNCOMPRESSED`.
- BUAT TABEL (ORC) - Dalam`TBLPROPERTIES`, tentukan`orc.compress = NONE`.

Catatan dan sumber daya

- Saat ini, ekstensi file huruf besar seperti `.GZ` atau tidak dikenali `.BZIP2` oleh Athena. Hindari menggunakan kumpulan data dengan ekstensi file huruf besar, atau ganti nama ekstensi file data menjadi huruf kecil.
- Untuk data dalam CSV, TSV, dan JSON, Athena menentukan jenis kompresi dari ekstensi file. Jika tidak ada ekstensi file, Athena memperlakukan data sebagai teks biasa yang tidak terkompresi. Jika data Anda dikompresi, pastikan nama file menyertakan ekstensi kompresi, sepertigz.
- Format file ZIP tidak didukung.
- Untuk menanyakan log Amazon Data Firehose dari Athena, format yang didukung mencakup kompresi GZIP atau file ORC dengan kompresi SNAPPY.

- Untuk informasi selengkapnya tentang penggunaan kompresi, lihat bagian 3 (“Kompres dan pisahkan file”) dari posting Blog AWS Big Data 10 kiat penyetelan kinerja teratas untuk Amazon Athena.

Dukungan kompresi tabel sarang dengan format file

Dukungan kompresi sarang di Athena tergantung pada versi mesin.

Dukungan kompresi sarang di mesin Athena versi 3

Tabel berikut merangkum dukungan format kompresi di Athena mesin versi 3 untuk format file penyimpanan di Apache Hive. Format file teks mencakup TSV, CSV, JSON, dan kustomSerDesuntuk teks. “Ya” atau “Tidak” dalam sel berlaku sama untuk membaca dan menulis operasi kecuali jika dicatat. Untuk keperluan tabel ini, CREATE TABLE, CTAS, dan INSERT INTO dianggap sebagai operasi tulis. Untuk informasi selengkapnya tentang penggunaan tingkat kompresi ZSTD di Athena, lihatMenggunakan tingkat kompresi ZSTD di Athena.

	Avro	Ion	ORC	Parquet:	File teks
BZIP2	Ya	Ya	Tidak	Tidak	Ya
DEFLATE	Ya	Tidak	Tidak	Tidak	Tidak
GZIP	Tidak	Ya	Tidak	Ya	Ya
LZ4	Tidak	Ya	Ya	Ya	Ya
LZO	Tidak	Tulis - Tidak Baca - Ya	Tidak	Ya	Tulis - Tidak Baca - Ya
SNAPPY	Ya	Ya	Ya	Ya	Ya
ZLIB	Tidak	Tidak	Ya	Tidak	Tidak
ZSTD	Ya	Ya	Ya	Ya	Ya
NONE	Ya	Ya	Ya	Ya	Ya

Dukungan kompresi sarang di mesin Athena versi 2

Tabel berikut merangkum dukungan format kompresi di Athena mesin versi 2 untuk Apache Hive. Format file teks mencakup TSV, CSV, JSON, dan kustomSerDesuntuk teks. “Ya” atau “Tidak” dalam sel berlaku sama untuk membaca dan menulis operasi kecuali jika dicatat. Untuk keperluan tabel ini, CREATE TABLE, CTAS, dan INSERT INTO dianggap sebagai operasi tulis.

	Avro	Ion	ORC	Parquet:	File teks
BZIP2	Ya	Ya	Tidak	Tidak	Ya
DEFLATE	Ya	Tidak	Tidak	Tidak	Tidak
GZIP	Tidak	Ya	Tidak	Ya	Ya
LZ4	Tidak	Tidak	Ya	Tulis - Ya Baca - Tidak	Tulis - Tidak Baca - Ya
LZO	Tidak	Tulis - Tidak Baca - Ya	Tidak	Ya	Tulis - Tidak Baca - Ya
SNAPPY	Ya	Ya	Ya	Ya	Ya
ZLIB	Tidak	Tidak	Ya	Tidak	Tidak
ZSTD	Tidak	Ya	Ya	Ya	Ya
NONE	Ya	Ya	Ya	Ya	Ya

Dukungan kompresi tabel gunung es dengan format file

Dukungan kompresi Apache Iceberg di Athena tergantung pada versi mesin.

Dukungan kompresi gunung es di mesin Athena versi 3

Tabel berikut merangkum dukungan format kompresi di mesin Athena versi 3 untuk format file penyimpanan di Apache Iceberg. “Ya” atau “Tidak” dalam sel berlaku sama untuk membaca dan

menulis operasi kecuali jika dicatat. Untuk keperluan tabel ini, CREATE TABLE, CTAS, dan INSERT INTO dianggap sebagai operasi tulis. Format penyimpanan default untuk Iceberg di mesin Athena versi 3 adalah Parket. Format kompresi default untuk Iceberg di mesin Athena versi 3 adalah ZSTD. Untuk informasi lebih lanjut tentang menggunakan tingkat kompresi ZSTD di Athena, lihat. Menggunakan tingkat kompresi ZSTD di Athena

	Avro	ORC	Parket (default)
BZIP2	Tidak	Tidak	Tidak
GZIP	Ya	Tidak	Ya
LZ4	Tidak	Ya	Tidak
SNAPPY	Ya	Ya	Ya
ZLIB	Tidak	Ya	Tidak
ZSTD	Ya	Ya	Ya (default)
NONE	Ya (tentukan `None` atau`Deflate`)	Ya	Ya (tentukan `None` atau`Uncompressed`)

Dukungan kompresi gunung es di mesin Athena versi 2

Tabel berikut merangkum dukungan format kompresi di Athena engine versi 2 untuk Apache Iceberg. “Ya” atau “Tidak” dalam sel berlaku sama untuk membaca dan menulis operasi kecuali jika dicatat. Untuk keperluan tabel ini, CREATE TABLE, CTAS, dan INSERT INTO dianggap sebagai operasi tulis. Format penyimpanan default untuk Iceberg di mesin Athena versi 2 adalah Parket. Format kompresi default untuk Iceberg di mesin Athena versi 2 adalah GZIP.

	Avro (Tidak didukung)	ORC (Tidak didukung)	Parket (default)
BZIP2	Tidak	Tidak	Tidak
GZIP	Tidak	Tidak	Ya (default)

	Avro (Tidak didukung)	ORC (Tidak didukung)	Parket (default)
LZ4	Tidak	Tidak	Tidak
SNAPPY	Tidak	Tidak	Ya
ZLIB	Tidak	Tidak	Tidak
ZSTD	Tidak	Tidak	Ya
NONE	Tidak	Tidak	Ya

Menggunakan tingkat kompresi ZSTD di Athena

Algoritma kompresi data real-time Zstandard adalah algoritma kompresi cepat yang memberikan rasio kompresi tinggi. Perpustakaan Zstandard (ZSTD) adalah perangkat lunak open source dan menggunakan lisensi BSD. Athena mendukung membaca dan menulis data file ORC, Parket, dan teks terkompresi ZSTD.

Anda dapat menggunakan tingkat kompresi ZSTD untuk menyesuaikan rasio kompresi dan kecepatan sesuai dengan kebutuhan Anda. Pustaka ZSTD mendukung tingkat kompresi dari 1 hingga 22. Athena menggunakan kompresi ZSTD level 3 secara default.

Tingkat kompresi memberikan trade-off granular antara kecepatan kompresi dan jumlah kompresi yang dicapai. Tingkat kompresi yang lebih rendah memberikan kecepatan yang lebih cepat tetapi ukuran file yang lebih besar. Misalnya, Anda dapat menggunakan level 1 jika kecepatan paling penting dan level 22 jika ukuran paling penting. Level 3 cocok untuk banyak kasus penggunaan dan merupakan default. Gunakan level yang lebih besar dari 19 dengan hati-hati karena membutuhkan lebih banyak memori. Pustaka ZSTD juga menawarkan tingkat kompresi negatif yang memperluas jangkauan kecepatan dan rasio kompresi. Untuk informasi lebih lanjut, lihat Zstandard Compression RFC.

Kelimpahan tingkat kompresi menawarkan peluang besar untuk fine tuning. Namun, pastikan Anda mengukur data Anda dan mempertimbangkan pengorbanan saat memutuskan tingkat kompresi. Sebaiknya gunakan level default 3 atau level dalam kisaran 6 hingga 9 untuk pertukaran yang wajar antara kecepatan kompresi dan ukuran data terkompresi. Tingkat cadangan 20 dan lebih besar untuk kasus di mana ukuran paling penting dan kecepatan kompresi tidak menjadi perhatian.

Pertimbangan dan batasan

Saat menggunakan tingkat kompresi ZSTD di Athena, pertimbangkan hal-hal berikut.

- `compression_level`Properti ZSTD hanya didukung di mesin Athena versi 3.
- `compression_level`Properti ZSTD didukung untuk`ALTER TABLE`,, `CREATE TABLE AS` (CTAS)`CREATE TABLE`, dan pernyataan. `UNLOAD`
- `compression_level`Properti ini opsional.
- `compression_level`Properti ini didukung hanya untuk kompresi ZSTD.
- Tingkat kompresi yang mungkin adalah 1 hingga 22.
- Tingkat kompresi default adalah 3.

Untuk informasi tentang dukungan kompresi Apache Hive ZSTD di Athena, lihat. Dukungan kompresi tabel sarang dengan format file Untuk informasi tentang dukungan kompresi Apache Iceberg ZSTD di Athena, lihat. Dukungan kompresi tabel gunung es dengan format file

Menentukan tingkat kompresi ZSTD

Untuk menentukan tingkat kompresi ZSTD untuk`ALTER TABLE`,, `CREATE TABLECREATE TABLE AS`, dan `UNLOAD` pernyataan, gunakan properti. `compression_level` Untuk menentukan kompresi ZSTD itu sendiri, Anda harus menggunakan properti kompresi individual yang digunakan sintaks untuk pernyataan tersebut.

MENGUBAH TABEL SET TBLPROPERTIES

Dalam `SET TBLPROPERTIES` klausa ALTER TABLE SET TBLPROPERTIES pernyataan, tentukan kompresi ZSTD menggunakan atau. `'write.compression' = ' ZSTD'` `'parquet.compression' = 'ZSTD'` Kemudian gunakan `compression_level` properti untuk menentukan nilai dari 1 hingga 22 (misalnya, `'compression_level' = 5`). Jika Anda tidak menentukan properti tingkat kompresi, tingkat kompresi default ke 3.

Contoh

Contoh berikut memodifikasi tabel `existing_table` untuk menggunakan format file Parket dengan kompresi ZSTD dan tingkat kompresi ZSTD 4. Perhatikan bahwa nilai tingkat kompresi harus dimasukkan sebagai string bukan bilangan bulat.

```
ALTER TABLE existing_table
SET TBLPROPERTIES ('parquet.compression' = 'ZSTD', 'compression_level' = 4)
```

CREATE TABLE

Dalam TBLPROPERTIES klausa CREATE TABLE pernyataan, tentukan 'write.compression' = 'ZSTD'atau'parquet.compression' = 'ZSTD', lalu gunakan compression_level = *compression_level* dan tentukan nilai dari 1 hingga 22. Jika compression_level properti tidak ditentukan, tingkat kompresi default adalah 3.

Contoh

Contoh berikut membuat tabel dalam format file Parket menggunakan kompresi ZSTD dan tingkat kompresi ZSTD 4.

```
CREATE EXTERNAL TABLE new_table (
  `col0` string COMMENT '',
  `col1` string COMMENT ''
)
STORED AS PARQUET
LOCATION 's3://DOC-EXAMPLE-BUCKET/'
TBLPROPERTIES ('write.compression' = 'ZSTD', 'compression_level' = 4)
```

BUAT TABEL SEBAGAI (CTAS)

Dalam WITH klausa CREATE TABLE AS pernyataan, tentukanwrite_compression = 'ZSTD', atauparquet_compression = 'ZSTD', lalu gunakan compression_level = *compression_level* dan tentukan nilai dari 1 hingga 22. Jika compression_level properti tidak ditentukan, tingkat kompresi default adalah 3.

Contoh

Contoh CTAS berikut menentukan Parket sebagai format file menggunakan kompresi ZSTD dengan tingkat kompresi 4.

```
CREATE TABLE new_table
WITH ( format = 'PARQUET', write_compression = 'ZSTD', compression_level = 4)
AS SELECT * FROM old_table
```

MEMBONGKAR

Dalam WITH klausa MEMBONGKAR pernyataan, tentukancompression = 'ZSTD', lalu gunakan compression_level = *compression_level* dan tentukan nilai dari 1 hingga 22. Jika compression_level properti tidak ditentukan, tingkat kompresi default adalah 3.

Contoh

Contoh berikut membongkar hasil query ke lokasi yang ditentukan menggunakan format file Parquet, kompresi ZSTD, dan tingkat kompresi ZSTD 4.

```
UNLOAD (SELECT * FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/'
WITH (format = 'PARQUET', compression = 'ZSTD', compression_level = 4)
```

Menandai sumber daya Athena

Setiap tanda terdiri atas sebuah kunci dan sebuah nilai, yang keduanya Anda tentukan. Ketika Anda menandai sumber daya Athena, Anda menetapkan metadata kustom untuk itu. Anda dapat menggunakan tag untuk mengkategorikanAWSsumber daya dengan cara yang berbeda; misalnya, dengan tujuan, pemilik, atau lingkungan. Di Athena, sumber daya seperti kelompok kerja, katalog data, dan pemesanan kapasitas adalah sumber daya yang dapat diberi tag. Misalnya, Anda dapat membuat sekumpulan tag untuk grup kerja di akun Anda yang membantu Anda melacak pemilik grup kerja, atau mengidentifikasi kelompok kerja berdasarkan tujuannya. Jika Anda juga mengaktifkan tag sebagai tag alokasi biaya di konsol Penagihan dan Manajemen Biaya, biaya yang terkait dengan menjalankan kueri akan muncul di Laporan Biaya dan Penggunaan dengan tag alokasi biaya tersebut. Kami menyarankan Anda yang Anda gunakanAWS menandai praktik terbaikuntuk membuat kumpulan tag yang konsisten untuk memenuhi persyaratan organisasi Anda.

Anda dapat bekerja dengan tag menggunakan konsol Athena atau operasi API.

Topik

- Dasar tanda
- Batasan tanda
- Bekerja dengan tag pada kelompok kerja di konsol
- Menggunakan operasi tag
- Kebijakan kontrol akses IAM berbasis tag

Dasar tanda

Tag adalah label yang Anda tetapkan ke sumber daya Athena. Setiap tanda terdiri atas sebuah kunci dan sebuah nilai opsional, yang keduanya Anda tentukan.

Tag memungkinkan Anda untuk mengkategorikanAWSsumber daya dengan cara yang berbeda. Misalnya, Anda dapat menentukan sekumpulan tag untuk grup kerja akun Anda yang membantu Anda melacak setiap pemilik atau tujuan grup kerja.

Anda dapat menambahkan tag saat membuat grup kerja Athena atau katalog data baru, atau Anda dapat menambahkan, mengedit, atau menghapus tag dari mereka. Anda dapat mengedit tag di konsol. Untuk menggunakan operasi API untuk mengedit tag, hapus tag lama dan tambahkan yang baru. Jika Anda menghapus sumber daya, semua tanda untuk sumber daya tersebut juga dihapus.

Athena tidak secara otomatis menetapkan tag ke sumber daya Anda. Anda dapat mengedit kunci dan nilai tanda, dan Anda dapat membuang tanda dari sumber daya kapan saja. Anda dapat mengatur nilai tanda ke string kosong, akan tetapi Anda tidak dapat mengatur nilai tanda ke nol. Jangan menambahkan kunci tag duplikat ke sumber daya yang sama. Jika Anda melakukannya, Athena mengeluarkan pesan kesalahan. Jika Anda menggunakanTagResourcetindakan untuk menandai sumber daya menggunakan kunci tag yang ada, nilai tag baru menimpa nilai lama.

Di IAM, Anda dapat mengontrol pengguna mana di akun Amazon Web Services Anda yang memiliki izin untuk membuat, mengedit, menghapus, atau mencantumkan tag. Untuk informasi selengkapnya, lihat Kebijakan kontrol akses IAM berbasis tag.

Untuk daftar lengkap tindakan tag Amazon Athena, lihat nama tindakan API diReferensi API Amazon Athena.

Anda dapat menggunakan tag untuk penagihan. Untuk informasi lebih lanjut, lihatMenggunakan tag untuk penagihandi dalamAWS Billing and Cost ManagementPanduan Pengguna.

Untuk informasi selengkapnya, lihat Batasan tanda.

Batasan tanda

Tag memiliki batasan sebagai berikut:

- Di Athena, Anda dapat menandai kelompok kerja dan katalog data. Anda tidak dapat menandai kueri.
- Jumlah maksimum tag per sumber daya adalah 50. Untuk tetap dalam batas, tinjau dan hapus tag yang tidak digunakan.
- Untuk setiap sumber daya, setiap kunci tanda harus unik, dan setiap kunci tanda hanya dapat memiliki satu nilai. Jangan menambahkan kunci tag duplikat pada saat yang sama ke sumber daya yang sama. Jika Anda melakukannya, Athena mengeluarkan pesan kesalahan. Jika Anda

menandai sumber daya menggunakan kunci tag yang ada di terpisah`TagResource`action, nilai tag baru menimpa nilai lama.

- Panjang kunci tag adalah 1-128 karakter Unicode di UTF-8.
- Panjang nilai tag adalah 0-256 karakter Unicode di UTF-8.

  Operasi penandaan, seperti menambahkan, mengedit, menghapus, atau mencantumkan tag, mengharuskan Anda menentukan ARN untuk sumber daya grup kerja.
- Athena memungkinkan Anda untuk menggunakan huruf, angka, spasi diwakili dalam UTF-8, dan karakter berikut: + - =. _:/@.
- Kunci dan nilai tanda peka huruf besar dan kecil.
- Yang`"aws:"`awalan dalam kunci tag dicadangkan untukAWSgunakan. Anda tidak dapat mengedit atau menghapus kunci tag dengan awalan ini. Tag dengan awalan ini tidak dihitung terhadap batas tag per sumber daya Anda.
- Tag yang Anda tetapkan hanya tersedia untuk akun Amazon Web Services Anda.

Bekerja dengan tag pada kelompok kerja di konsol

Dengan menggunakan konsol Athena, Anda dapat melihat tag mana yang digunakan oleh setiap grup kerja di akun Anda. Anda dapat melihat tag oleh workgroup saja. Anda juga dapat menggunakan konsol Athena untuk menerapkan, mengedit, atau menghapus tag dari satu grup kerja sekaligus.

Anda dapat mencari kelompok kerja menggunakan tag yang Anda buat.

Topik

- Menampilkan tag untuk kelompok kerja individu
- Menambahkan dan menghapus tag pada kelompok kerja individu

Menampilkan tag untuk kelompok kerja individu

Untuk menampilkan tag untuk kelompok kerja individu di konsol Athena

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.

3. Pada menu navigasi, pilihKelompok kerja, dan kemudian pilih workgroup yang Anda inginkan.
4. Lakukan salah satu dari berikut:
   - Pilih tab Tag (Tanda). Jika daftar tag panjang, gunakan kotak pencarian.
   - PilihMengedit, dan kemudian gulir ke bawah keTagbagian.

Menambahkan dan menghapus tag pada kelompok kerja individu

Anda dapat mengelola tag untuk kelompok kerja individu langsung dariKelompok kerjatab.

> **Note**
>
> Jika Anda ingin pengguna menambahkan tag saat mereka membuat grup kerja di konsol atau meneruskan tag saat mereka menggunakanCreateWorkGrouptindakan, pastikan bahwa Anda memberikan pengguna IAM izin keTagResourcedanCreateWorkGrouptindakan.

Untuk menambahkan tag saat Anda membuat grup kerja baru

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Pada menu navigasi, pilihKelompok kerja.
3. PilihBuat workgroupdan mengisi nilai-nilai yang diperlukan. Untuk langkah terperinci, lihat Buat grup kerja.

4. DalamTagbagian, tambahkan satu atau lebih tag dengan menentukan kunci dan nilai. Jangan menambahkan kunci tag duplikat pada saat yang sama ke workgroup yang sama. Jika Anda melakukannya, Athena mengeluarkan pesan kesalahan. Untuk informasi selengkapnya, lihat Batasan tanda.
5. Setelah selesai, pilihBuat workgroup.

Untuk menambahkan atau mengedit tag ke grup kerja yang ada

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Di panel navigasi, pilihKelompok kerja.
3. Pilih workgroup yang ingin Anda modifikasi.
4. Lakukan salah satu dari berikut:
   - PilihTagtab, dan kemudian pilihMengelola tag.
   - PilihMengedit, dan kemudian gulir ke bawah keTagbagian.
5. Tentukan kunci dan nilai untuk setiap tag. Untuk informasi lebih lanjut, lihat Batasan tanda.
6. Pilih Simpan.

Untuk menghapus tag dari grup kerja individual

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Di panel navigasi, pilihKelompok kerja.
3. Pilih workgroup yang ingin Anda modifikasi.
4. Lakukan salah satu dari berikut:
   - PilihTagtab, dan kemudian pilihMengelola tag.
   - PilihMengedit, dan kemudian gulir ke bawah keTagbagian.
5. Dalam daftar tag, pilihHapusuntuk tag yang ingin Anda hapus, lalu pilihSimpan.

Menggunakan operasi tag

Gunakan operasi tag berikut untuk menambah, menghapus, atau mencantumkan tag pada sumber daya.

API	CLI	Deskripsi tindakan
`TagResource`	`tag-resource`	Tambahkan atau timpa satu atau lebih tag pada sumber daya yang memiliki ARN yang ditentukan.
`UntagResource`	`untag-resource`	Hapus satu atau lebih tag dari sumber daya yang memiliki ARN yang ditentukan.
`ListTagsForResource`	`list-tags-for-resource`	Buat daftar satu atau lebih tag untuk sumber daya yang memiliki ARN yang ditentukan.

Menambahkan Tag Saat Membuat Sumber Daya

Untuk menambahkan tag saat Anda membuat grup kerja atau katalog data, gunakan`tags`parameter dengan`CreateWorkGroup`atau`CreateDataCatalog`Operasi API atau dengan`AWS CLI` `create-work-group`atau`create-data-catalog`perintah.

Mengelola tag menggunakan operasi API

Contoh di bagian ini menunjukkan cara menggunakan operasi API tag untuk mengelola tag pada kelompok kerja dan katalog data. Contohnya adalah dalam bahasa pemrograman Java.

Example TagResource

Contoh berikut menambahkan dua tag ke workgroup`workgroupA`:

```
List<Tag> tags = new ArrayList<>();
tags.add(new Tag().withKey("tagKey1").withValue("tagValue1"));
tags.add(new Tag().withKey("tagKey2").withValue("tagValue2"));

TagResourceRequest request = new TagResourceRequest()
    .withResourceARN("arn:aws:athena:us-east-1:123456789012:workgroup/workgroupA")
    .withTags(tags);

client.tagResource(request);
```

Contoh berikut menambahkan dua tag ke katalog data`datacatalogA`:

```
List<Tag> tags = new ArrayList<>();
```

```
tags.add(new Tag().withKey("tagKey1").withValue("tagValue1"));
tags.add(new Tag().withKey("tagKey2").withValue("tagValue2"));

TagResourceRequest request = new TagResourceRequest()
    .withResourceARN("arn:aws:athena:us-east-1:123456789012:datacatalog/datacatalogA")
    .withTags(tags);

client.tagResource(request);
```

> **Note**
>
> Jangan menambahkan kunci tag duplikat ke sumber daya yang sama. Jika Anda melakukannya, Athena mengeluarkan pesan kesalahan. Jika Anda menandai sumber daya menggunakan kunci tag yang ada di terpisah`TagResource`action, nilai tag baru menimpa nilai lama.

Example UntagResource

Contoh berikut menghapus`tagKey2`dari workgroup`workgroupA`:

```
List<String> tagKeys = new ArrayList<>();
tagKeys.add("tagKey2");

UntagResourceRequest request = new UntagResourceRequest()
    .withResourceARN("arn:aws:athena:us-east-1:123456789012:workgroup/workgroupA")
    .withTagKeys(tagKeys);

client.untagResource(request);
```

Contoh berikut menghapus`tagKey2`dari katalog data`datacatalogA`:

```
List<String> tagKeys = new ArrayList<>();
tagKeys.add("tagKey2");

UntagResourceRequest request = new UntagResourceRequest()
    .withResourceARN("arn:aws:athena:us-east-1:123456789012:datacatalog/datacatalogA")
    .withTagKeys(tagKeys);

client.untagResource(request);
```

Example ListTagsForResource

Contoh berikut mencantumkan tag untuk workgroup`workgroupA`:

```
ListTagsForResourceRequest request = new ListTagsForResourceRequest()
    .withResourceARN("arn:aws:athena:us-east-1:123456789012:workgroup/workgroupA");

ListTagsForResourceResult result = client.listTagsForResource(request);

List<Tag> resultTags = result.getTags();
```

Contoh berikut mencantumkan tag untuk katalog data`datacatalogA`:

```
ListTagsForResourceRequest request = new ListTagsForResourceRequest()
    .withResourceARN("arn:aws:athena:us-east-1:123456789012:datacatalog/datacatalogA");

ListTagsForResourceResult result = client.listTagsForResource(request);

List<Tag> resultTags = result.getTags();
```

Mengelola tanda menggunakan AWS CLI

Bagian berikut menunjukkan cara menggunakanAWS CLIuntuk membuat dan mengelola tag pada katalog data.

Menambahkan tag ke sumber daya: Tag-resource

Yang`tag-resource`perintah menambahkan satu atau lebih tag ke sumber daya tertentu.

Sintaksis

```
aws athena tag-resource --resource-arn
arn:aws:athena:region:account_id:datacatalog/catalog_name --tags
Key=string,Value=string Key=string,Value=string
```

Yang`--resource-arn`parameter menentukan sumber daya yang tag ditambahkan. Yang`--tags`parameter menentukan daftar pasangan kunci-nilai spasi dipisahkan untuk menambahkan sebagai tag ke sumber daya.

Example

Contoh berikut menambahkan tag kemy`datacatalog`katalog data.

```
aws athena tag-resource --resource-arn arn:aws:athena:us-
east-1:111122223333:datacatalog/mydatacatalog --tags Key=Color,Value=Orange
 Key=Time,Value=Now
```

Untuk menunjukkan hasilnya, gunakan`list-tags-for-resource`perintah.

Untuk informasi tentang menambahkan tag saat menggunakan`create-data-catalog`perintah, lihatMendaftarkan katalog: C reate-data-catalog.

Daftar tag untuk sumber daya: List-tags-for-resource

Yang`list-tags-for-resource`perintah mencantumkan tag untuk sumber daya yang ditentukan.

Sintaksis

`aws athena list-tags-for-resource --resource-arn arn:aws:athena:`*`region`*`:`*`account_id`*`:datacatalog/`*`catalog_name`*

Yang`--resource-arn`parameter menentukan sumber daya yang tag terdaftar.

Contoh berikut mencantumkan tag untuk`mydatacatalog`katalog data.

```
aws athena list-tags-for-resource --resource-arn arn:aws:athena:us-
east-1:111122223333:datacatalog/mydatacatalog
```

Hasil contoh berikut adalah dalam format JSON.

```
{
    "Tags": [
        {
            "Key": "Time",
            "Value": "Now"
        },
        {
            "Key": "Color",
            "Value": "Orange"
        }
    ]
}
```

Menghapus tag dari sumber daya: Untag-resource

Yanguntag-resourceperintah menghapus kunci tag tertentu dan nilai-nilai yang terkait dari sumber daya yang ditentukan.

Sintaksis

```
aws athena untag-resource --resource-arn
arn:aws:athena:region:account_id:datacatalog/catalog_name --tag-keys
key_name [key_name ...]
```

Yang--resource-arnparameter menentukan sumber daya dari mana tag dihapus. Yang--tag-keysparameter mengambil daftar spasi-dipisahkan dari nama-nama kunci. Untuk setiap nama kunci yang ditentukan,untag-resourceperintah menghapus kedua kunci dan nilainya.

Contoh berikut menghapusColordanTimekunci dan nilai-nilai mereka darimydatacatalogsumber daya katalog.

```
aws athena untag-resource --resource-arn arn:aws:athena:us-
east-1:111122223333:datacatalog/mydatacatalog --tag-keys Color Time
```

Kebijakan kontrol akses IAM berbasis tag

Memiliki tag memungkinkan Anda untuk menulis kebijakan IAM yang mencakupConditionblok untuk mengontrol akses ke sumber daya berdasarkan tag-nya.

Contoh kebijakan tag untuk kelompok kerja

Example 1. kebijakan penandaan dasar

Kebijakan IAM berikut memungkinkan Anda menjalankan kueri dan berinteraksi dengan tag untuk grup kerja bernamaworkgroupA:

```
{
    "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: Blok kebijakan yang menolak tindakan pada grup kerja berdasarkan kunci tag dan pasangan nilai tag

Tag yang terkait dengan sumber daya seperti workgroup disebut sebagai tag sumber daya. Tag sumber daya memungkinkan Anda menulis blok kebijakan seperti berikut ini yang menolak tindakan yang tercantum pada grup kerja apa pun yang ditandai dengan pasangan kunci-nilai seperti`stack`,`production`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "athena:GetWorkGroup",
                "athena:UpdateWorkGroup",
                "athena:DeleteWorkGroup",
                "athena:TagResource",
                "athena:UntagResource",
                "athena:ListTagsForResource",
                "athena:StartQueryExecution",
                "athena:GetQueryExecution",
                "athena:BatchGetQueryExecution",
                "athena:ListQueryExecutions",
                "athena:StopQueryExecution",
                "athena:GetQueryResults",
                "athena:GetQueryResultsStream",
                "athena:CreateNamedQuery",
                "athena:GetNamedQuery",
                "athena:BatchGetNamedQuery",
                "athena:ListNamedQueries",
                "athena:DeleteNamedQuery",
                "athena:CreatePreparedStatement",
                "athena:GetPreparedStatement",
                "athena:ListPreparedStatements",
                "athena:UpdatePreparedStatement",
                "athena:DeletePreparedStatement"
            ],
            "Resource": "arn:aws:athena:us-east-1:123456789012:workgroup/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stack": "production"
                }
            }
        }
    ]
}
```

Example 3. blok kebijakan yang membatasi permintaan aksi pengubahan tag ke tag tertentu

Tag yang diteruskan sebagai parameter ke operasi yang mengubah tag (misalnya,`TagResource`,`UntagResource`, atau`CreateWorkGroup`dengan tag) disebut sebagai tag permintaan. Berikut blok kebijakan contoh memungkinkan`CreateWorkGroup`operasi hanya jika salah satu tag berlalu memiliki kunci`costcenter`dan nilainya`1`,`2`, atau`3`.

> **Note**
>
> Jika Anda ingin mengizinkan peran IAM untuk meneruskan tag sebagai bagian dari`CreateWorkGroup`operasi, pastikan Anda memberikan izin peran ke`TagResource`dan`CreateWorkGroup`tindakan.

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

Contoh kebijakan tag untuk katalog data

Example 1. kebijakan penandaan dasar

Kebijakan IAM berikut ini memungkinkan Anda berinteraksi dengan tag untuk katalog data bernama`datacatalogA`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListWorkGroups",
                "athena:ListEngineVersions",
                "athena:ListDataCatalogs",
                "athena:ListDatabases",
                "athena:GetDatabase",
                "athena:ListTableMetadata",
                "athena:GetTableMetadata"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "athena:GetWorkGroup",
                "athena:TagResource",
                "athena:UntagResource",
                "athena:ListTagsForResource",
                "athena:StartQueryExecution",
                "athena:GetQueryExecution",
                "athena:BatchGetQueryExecution",
                "athena:ListQueryExecutions",
                "athena:StopQueryExecution",
                "athena:GetQueryResults",
                "athena:GetQueryResultsStream",
                "athena:CreateNamedQuery",
                "athena:GetNamedQuery",
                "athena:BatchGetNamedQuery",
                "athena:ListNamedQueries",
                "athena:DeleteNamedQuery"
            ],
            "Resource": [
```

```
                "arn:aws:athena:us-east-1:123456789012:workgroup/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "athena:CreateDataCatalog",
                "athena:GetDataCatalog",
                "athena:UpdateDataCatalog",
                "athena:DeleteDataCatalog",
                "athena:ListDatabases",
                "athena:GetDatabase",
                "athena:ListTableMetadata",
                "athena:GetTableMetadata",
                "athena:TagResource",
                "athena:UntagResource",
                "athena:ListTagsForResource"
            ],
            "Resource": "arn:aws:athena:us-
east-1:123456789012:datacatalog/datacatalogA"
        }
    ]
}
```

Example 2: Pemblokiran kebijakan yang menolak tindakan pada Katalog Data berdasarkan kunci tag dan pasangan nilai tag

Anda dapat menggunakan tag sumber daya untuk menulis blok kebijakan yang menolak tindakan tertentu pada katalog data yang ditandai dengan pasangan kunci-nilai tag tertentu. Kebijakan contoh berikut menolak tindakan pada katalog data yang memiliki pasangan nilai kunci tag`stack`,`production`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "athena:CreateDataCatalog",
                "athena:GetDataCatalog",
                "athena:UpdateDataCatalog",
                "athena:DeleteDataCatalog",
                "athena:GetDatabase",
```

```
                "athena:ListDatabases",
                "athena:GetTableMetadata",
                "athena:ListTableMetadata",
                "athena:StartQueryExecution",
                "athena:TagResource",
                "athena:UntagResource",
                "athena:ListTagsForResource"
            ],
            "Resource": "arn:aws:athena:us-east-1:123456789012:datacatalog/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stack": "production"
                }
            }
        }
    ]
}
```

Example 3. blok kebijakan yang membatasi permintaan aksi pengubahan tag ke tag tertentu

Tag yang diteruskan sebagai parameter ke operasi yang mengubah tag (misalnya,`TagResource`,`UntagResource`, atau`CreateDataCatalog`dengan tag) disebut sebagai tag permintaan. Berikut blok kebijakan contoh memungkinkan`CreateDataCatalog`operasi hanya jika salah satu tag berlalu memiliki kunci`costcenter`dan nilainya`1`,`2`, atau`3`.

> **Note**
>
> Jika Anda ingin mengizinkan peran IAM untuk meneruskan tag sebagai bagian dari`CreateDataCatalog`operasi, pastikan Anda memberikan izin peran ke`TagResource`dan`CreateDataCatalog`tindakan.

```
{
    "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**
>
> Konsol Service Quotas menyediakan informasi tentang kuota EventBridge. Anda juga dapat menggunakan konsol Service Quotas untuk meminta kenaikan kuota untuk kuota yang dapat disesuaikan. Untuk batasan skema AWS Glue terkait, lihat halaman AWS Glue titik akhir dan kuota. Untuk informasi umum tentang kuota AWS layanan, lihat kuota AWS layanan di. Referensi Umum AWS

Kueri

Akun Anda memiliki kuota terkait kueri berikut untuk Amazon Athena. Untuk detailnya, lihat halaman titik akhir dan kuota Amazon Athena di halaman. Referensi Umum AWS

- Kueri DDL aktif — Jumlah kueri DDL aktif. DDL kueri termasuk kueri `CREATE TABLE` dan `ALTER TABLE ADD PARTITION`.
- Batas waktu kueri DDL - Jumlah waktu maksimum dalam hitungan menit kueri DDL dapat dijalankan sebelum dibatalkan.
- Active DMLQuery — Jumlah query DML aktif. Kueri DML meliputi`SELECT`, `CREATE TABLE AS` (CTAS), dan kueri. `INSERT INTO` Kuota spesifik bervariasi menurut AWS Wilayah.
- Batas waktu kueri DML—Jumlah waktu maksimum dalam hitungan menit kueri DML dapat dijalankan sebelum dibatalkan. Anda dapat meminta peningkatan batas waktu ini hingga maksimal 240 menit.

Untuk meminta kenaikan kuota, Anda dapat menggunakan konsol Service Quotas Athena.

Athena memproses kueri dengan menetapkan sumber daya berdasarkan beban layanan secara keseluruhan dan jumlah permintaan yang masuk. Kueri Anda mungkin antrian sementara sebelum dijalankan. Proses asynchronous mengambil kueri dari antrian dan menjalankannya pada sumber daya fisik segera setelah sumber daya menjadi tersedia dan selama konfigurasi akun Anda memungkinkan.

Kuota permintaan DDLL atau DDL mencakup kueri berjalan dan antrian. Misalnya, jika kuota kueri DML Anda adalah 25 dan total kueri berjalan dan antrian Anda adalah 26, kueri 26 akan menghasilkan kesalahan. TooManyRequestsException

> **Note**
>
> Jika Anda ingin mengontrol konkurensi secara langsung untuk kueri yang Anda jalankan di Athena, Anda dapat menggunakan reservasi kapasitas. Untuk informasi selengkapnya, lihat Mengelola kapasitas pemrosesan kueri.

Panjang string kueri

Maksimum diperbolehkan panjang string kueri adalah 262144 byte, tempat string dikodekan dalam UTF-8. Ini bukan kuota yang dapat disesuaikan. Namun, Anda dapat bekerja di sekitar keterbatasan ini dengan membelah kueri panjang ke beberapa kueri yang lebih kecil. Untuk informasi selengkapnya, lihatBagaimana cara meningkatkan panjang string kueri maksimum di Athena?di AWS Pusat Pengetahuan.

Kelompok kerja

Saat Anda bekerja dengan grup kerja Athena, ingat hal-hal berikut:

- service quotas Athena dibagi di semua grup kerja dalam akun.
- Jumlah maksimum grup kerja yang dapat Anda buat per Wilayah di akun adalah 1000.
- Jumlah maksimum pernyataan yang disiapkan dalam kelompok kerja adalah 1000.
- Jumlah maksimum tanda per grup log adalah 50. Untuk informasi selengkapnya, lihat Batasan tanda.

Database, tabel, dan partisi

- Jika Anda menggunakan AWS Glue Data Catalog with Athena, lihat AWS Glue titik akhir dan kuota untuk kuota layanan pada tabel, database, dan partisi — misalnya, jumlah maksimum database atau tabel per akun.
  - Meskipun Athena mendukung AWS Glue tabel kueri yang memiliki 10 juta partisi, Athena tidak dapat membaca lebih dari 1 juta partisi dalam satu pemindaian.
- Jika Anda tidak menggunakan AWS Glue Data Catalog, jumlah partisi per tabel adalah 20.000. Anda dapat meminta penambahan kuota.

Bucket Amazon S3

Saat Anda bekerja dengan bucket Amazon S3, ingat hal-hal berikut:

- Amazon S3 memiliki service quotas default 100 bucket per akun.
- Athena membutuhkan bucket terpisah untuk mencatat hasil.
- Anda dapat meminta peningkatan kuota hingga 1.000 bucket Amazon S3 per AWS akun.

Kuota panggilan API per akun

Athena API memiliki kuota default berikut untuk jumlah panggilan ke API per akun (bukan per kueri):

Nama API	Jumlah panggilan default per detik	Kapasitas lonjakan
`BatchGetNamedQuery` , `ListNamedQueries` , `ListQueryExecutions`	5	hingga 10
`CreateNamedQuery` , `DeleteNamedQuery` , `GetNamedQuery`	5	hingga 20
`BatchGetQueryExecution`	20	hingga 40
`StartQueryExecution` , `StopQueryExecution`	20	hingga 80

Nama API	Jumlah panggilan default per detik	Kapasitas lonjakan
`GetQueryExecution` , `GetQueryResults`	100	hingga 200

Misalnya, Anda dapat melakukan hingga 20 panggilan per detik untuk`StartQueryExecution`. Selain itu, jika API ini tidak dipanggil selama 4 detik, akun Anda akan terakumulasiKapasitas Bursthingga 80 panggilan. Dalam kasus ini, aplikasi Anda dapat membuat hingga 80 panggilan ke API ini dalam mode burst.

Jika Anda menggunakan salah satu API ini dan melebihi kuota default untuk jumlah panggilan per detik, atau kapasitas burst di akun Anda, API Athena mengeluarkan kesalahan yang serupa dengan berikut: "ClientError": Terjadi kesalahan ThrottlingException () saat memanggil operasi: Nilai <API_name>terlampaui.” Kurangi jumlah panggilan per detik, atau kapasitas burst untuk API untuk akun ini.

Kuota Athena untuk panggilan API per akun tidak dapat diubah di konsol Service Quotas Athena. Untuk meminta peningkatan kuota panggilan API Athena, buka halaman peningkatan batas Layanan dan lengkapi dan kirimkan formulir. AWS Support

Pembuatan versi mesin Athena

Athena terkadang merilis versioning mesin baru untuk memberikan peningkatan performa, fungsionalitas, dan perbaikan kode. Saat versioning mesin baru tersedia, Athena memberi tahu Anda melalui konsol Athena dan AWS Health Dashboard. Anda AWS Health Dashboard memberi tahu Anda tentang peristiwa yang dapat memengaruhi AWS layanan atau akun Anda. Untuk informasi selengkapnya AWS Health Dashboard, lihat Memulai dengan AWS Health Dashboard.

Versioning mesin dikonfigurasi per grup kerja. Anda dapat menggunakan grup kerja untuk mengontrol mesin kueri mana yang digunakan kueri Anda dan apakah Athena akan meng-upgrade grup kerja Anda secara otomatis. Mesin kueri yang sedang digunakan ditampilkan di editor kueri, pada halaman detail workgroup, dan tersedia melalui Athena API.

- Secara default, workgroup dikonfigurasi untuk auto upgrade. Saat workgroup disetel ke auto upgrade, Athena mengupgrade workgroup untuk Anda kecuali jika menemukan ketidakcocokan.

- Jika Anda mengonfigurasi workgroup untuk menggunakan versi tertentu, Athena tidak akan mengubah versi workgroup.

Dalam kedua kasus tersebut, Athena meningkatkan grup kerja Anda saat versi tidak lagi tersedia. Athena memberi tahu Anda AWS Health Dashboardtentang kapan versi mesin tidak lagi ditawarkan. Anda AWS Health Dashboard memberi tahu Anda tentang peristiwa yang dapat memengaruhi AWS layanan atau akun Anda. Untuk informasi selengkapnya AWS Health Dashboard, lihat Memulai dengan AWS Health Dashboard.

Saat Anda mulai menggunakan versioning mesin baru, subset kecil kueri mungkin pecah karena tidak kompatibel. Perubahan yang melanggar diumumkan ketika versi Athena baru dirilis. Anda harus menggunakan workgroup untuk menguji kueri Anda sebelum upgrade dengan membuat workgroup pengujian yang menggunakan mesin baru atau dengan menguji upgrade workgroup yang ada. Untuk informasi selengkapnya, lihat Menguji kueri sebelum upgrade versi mesin.

Topik

- Mengubah versi mesin Athena
- Referensi versi mesin Athena

Mengubah versi mesin Athena

Athena terkadang merilis versioning mesin baru untuk memberikan peningkatan performa, fungsionalitas, dan perbaikan kode. Jika versioning mesin baru tersedia, Athena akan memberi tahu Anda di konsol. Anda dapat memilih untuk membiarkan Athena memutuskan kapan akan memutakhirkan, atau secara manual menentukan versioning mesin Athena per grup kerja.

Topik

- Menemukan versi mesin kueri untuk grup kerja
- Mengubah versi mesin di konsol Athena
- Mengubah versi mesin menggunakan AWS CLI
- Menentukan versi mesin saat Anda membuat workgroup
- Menguji kueri sebelum upgrade versi mesin
- Memecahkan masalah kueri yang gagal

Menemukan versi mesin kueri untuk grup kerja

Anda dapat menggunakan halaman Workgroups untuk menemukan versi mesin saat ini untuk workgroup apa pun.

Untuk menemukan versioning mesin saat ini untuk setiap grup kerja

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.

3. Di panel navigasi konsol Athena, pilih Workgroups.
4. Pada halaman Workgroups, temukan workgroup yang Anda inginkan. Kolom versi mesin Query untuk workgroup menampilkan versi mesin kueri.

Mengubah versi mesin di konsol Athena

Saat versioning mesin baru tersedia, Anda dapat memilih untuk membiarkan Athena memutuskan kapan akan memutakhirkan grup kerja, atau secara manual menentukan versioning mesin Athena yang digunakan grup kerja. Jika hanya satu versi yang tersedia saat ini, menentukan versi yang berbeda secara manual tidak dimungkinkan.

> **Note**
>
> Untuk mengubah versioning mesin untuk grup kerja, Anda harus memiliki izin untuk melakukan tindakan `athena:ListEngineVersions` pada grup kerja. Untuk contoh kebijakan IAM, lihat Kebijakan contoh kelompok kerja.

Untuk membiarkan Athena memutuskan kapan akan memutakhirkan grup kerja

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.
3. Di panel navigasi konsol, pilih Workgroups.
4. Dalam daftar workgroup, pilih link untuk workgroup yang ingin Anda konfigurasi.
5. Pilih Edit.
6. Di bagian Versi mesin kueri, untuk mesin kueri Perbarui, pilih Otomatis untuk membiarkan Athena memilih kapan harus meng-upgrade workgroup Anda. Ini adalah pengaturan default.
7. Pilih Simpan perubahan.

   Dalam daftar kelompok kerja, status pembaruan mesin kueri untuk grup kerja menunjukkan Otomatis.

Cara memilih versioning mesin secara manual

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.
3. Di panel navigasi konsol, pilih Workgroups.
4. Dalam daftar workgroup, pilih link untuk workgroup yang ingin Anda konfigurasi.
5. Pilih Edit.
6. Di bagian Versi mesin kueri, untuk Perbarui mesin kueri, pilih Manual untuk memilih versi mesin secara manual.
7. Gunakan opsi Query engine version untuk memilih versi mesin yang ingin digunakan oleh workgroup. Jika versi mesin yang berbeda tidak tersedia, versi mesin yang berbeda tidak dapat ditentukan.
8. Pilih Simpan perubahan.

Dalam daftar workgroup, status update mesin Query untuk workgroup menunjukkan Manual.

Mengubah versi mesin menggunakan AWS CLI

Untuk mengubah versi mesin menggunakan AWS CLI, gunakan sintaks dalam contoh berikut.

```
aws athena update-work-group --work-group workgroup-name --configuration-updates
 EngineVersion={SelectedEngineVersion='Athena engine version 3'}
```

Menentukan versi mesin saat Anda membuat workgroup

Saat Anda membuat grup kerja, Anda dapat menentukan versioning mesin yang menggunakan grup kerja atau membiarkan Athena memutuskan kapan akan memutakhirkan grup kerja. Jika versi mesin baru tersedia, praktik terbaik adalah membuat grup kerja untuk menguji mesin baru sebelum Anda meningkatkan grup kerja Anda yang lain. Untuk menentukan versioning mesin untuk grup kerja, Anda harus memiliki izin `athena:ListEngineVersions` pada grup kerja. Untuk contoh kebijakan IAM, lihat Kebijakan contoh kelompok kerja.

Untuk menentukan versioning mesin saat Anda membuat grup kerja

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.
3. Di panel navigasi konsol, pilih Workgroups.
4. Pada Grup Kerja, pilih Buat grup kerja.
5. Pada halaman Create workgroup, di bagian Query Engine Version, lakukan salah satu hal berikut:
   - Pilih Otomatis untuk membiarkan Athena memilih kapan harus meng-upgrade workgroup Anda. Ini adalah pengaturan default.
   - Pilih Manual untuk memilih versi mesin yang berbeda secara manual jika tersedia.
6. Masukkan informasi untuk bidang lain yang diperlukan. Untuk informasi tentang bidang lainnya, lihat Buat grup kerja.
7. Pilih Buat grup kerja.

Menguji kueri sebelum upgrade versi mesin

Saat workgroup ditingkatkan ke versi mesin baru, beberapa kueri Anda dapat rusak karena ketidakcocokan. Untuk memastikan pemutakhiran versioning mesin Anda berjalan lancar, Anda dapat menguji kueri Anda terlebih dahulu.

Untuk menguji kueri Anda sebelum pemutakhiran versioning mesin

1. Verifikasi versioning mesin grup kerja yang Anda gunakan. Versi mesin yang Anda gunakan ditampilkan pada halaman Workgroups di kolom Query engine version untuk workgroup. Untuk informasi selengkapnya, lihat Menemukan versi mesin kueri untuk grup kerja.
2. Buat grup kerja tes yang menggunakan versioning mesin baru. Untuk informasi selengkapnya, lihat Menentukan versi mesin saat Anda membuat workgroup.
3. Gunakan grup kerja baru untuk menjalankan kueri yang ingin Anda uji.
4. Jika permintaan gagal, gunakan Referensi versi mesin Athena untuk memeriksa perubahan pemecahan yang mungkin memengaruhi permintaan. Beberapa perubahan mungkin mengharuskan Anda untuk memperbarui sintaks kueri Anda.
5. Jika pertanyaan Anda masih gagal, hubungi AWS Support untuk bantuan. Di bagian AWS Management Console, pilih Support, Support Center, atau ajukan pertanyaan di AWS re:Post menggunakan tag Amazon Athena.

Memecahkan masalah kueri yang gagal

Jika permintaan gagal setelah pemutakhiran versioning mesin, gunakan Referensi versi mesin Athena untuk memeriksa perubahan pemecahan, termasuk perubahan yang dapat memengaruhi sintaks dalam kueri Anda.

Jika pertanyaan Anda masih gagal, hubungi AWS Support untuk bantuan. Di bagian AWS Management Console, pilih Support, Support Center, atau ajukan pertanyaan di AWS re:Post menggunakan tag Amazon Athena.

Referensi versi mesin Athena

Bagian ini mencantumkan perubahan pada mesin kueri Athena.

Topik

- Mesin Athena versi 3

- Versi mesin Athena 2

Mesin Athena versi 3

Untuk engine versi 3, Athena telah memperkenalkan pendekatan integrasi berkelanjutan untuk manajemen perangkat lunak open source yang meningkatkan konkurensi dengan proyek Trino dan Presto sehingga Anda mendapatkan akses lebih cepat ke peningkatan komunitas, terintegrasi dan disetel dalam mesin Athena.

Rilis mesin Athena versi 3 ini mendukung semua fitur mesin Athena versi 2. Dokumen ini menyoroti perbedaan utama antara mesin Athena versi 2 dan mesin Athena versi 3. Untuk informasi selengkapnya, lihat artikel AWS Big Data Blog Upgrade ke mesin Athena versi 3 untuk meningkatkan kinerja kueri dan mengakses lebih banyak fitur analitik.

- Memulai
- Perbaikan dan fitur baru
  - Fitur Ditambahkan
  - Fungsi Menambahkan
  - Peningkatan kinerja
  - Peningkatan keandalan
  - Penyempurnaan sintaks kueri
  - Format data dan peningkatan tipe data
- Melanggar perubahan
  - Perubahan sintaks kueri
  - Perubahan pemrosesan data
  - Perubahan stempel waktu
- Batasan

Memulai

Untuk memulai, buat workgroup Athena baru yang menggunakan mesin Athena versi 3 atau konfigurasikan workgroup yang ada untuk menggunakan versi 3. Setiap workgroup Athena dapat meng-upgrade dari engine versi 2 ke engine versi 3 tanpa gangguan dalam kemampuan Anda untuk mengirimkan pertanyaan.

Untuk informasi selengkapnya, lihat Mengubah versi mesin Athena.

Perbaikan dan fitur baru

Fitur dan pembaruan yang tercantum termasuk peningkatan dari Athena sendiri dan dari fungsionalitas yang tergabung dari Trino open source. Untuk daftar lengkap operator dan fungsi kueri SQL, lihat dokumentasi Trino.

Fitur Ditambahkan

Dukungan algoritma bucketing Apache Spark

Athena dapat membaca bucket yang dihasilkan oleh algoritma hash Spark. Untuk menentukan bahwa data awalnya ditulis oleh algoritma hash Spark, masukkan `('bucketing_format'='spark')` `TBLPROPERTIES` klausa pernyataan Anda. `CREATE TABLE` Jika properti ini tidak ditentukan, algoritma hash Hive digunakan.

```
CREATE EXTERNAL TABLE `spark_bucket_table`(
  `id` int,
  `name` string
  )
CLUSTERED BY (`name`)
INTO 8 BUCKETS
STORED AS PARQUET
LOCATION
  's3://DOC-EXAMPLE-BUCKET/to/bucketed/table/'
TBLPROPERTIES ('bucketing_format'='spark')
```

Fungsi Menambahkan

Fungsi di bagian ini baru untuk mesin Athena versi 3.

Fungsi agregat

listagg (x, separator) - Mengembalikan nilai input digabungkan, dipisahkan oleh string pemisah.

```
SELECT listagg(value, ',') WITHIN GROUP (ORDER BY value) csv_value
FROM (VALUES 'a', 'c', 'b') t(value);
```

Fungsi array

contains_sequence (x, seq) - Mengembalikan nilai true jika array x berisi semua array seq sebagai subset sekuensial (semua nilai dalam urutan berurutan yang sama).

```
SELECT contains_sequence(ARRAY [1,2,3,4,5,6], ARRAY[1,2]);
```

Fungsi biner

murmur3 (biner) — Menghitung hash biner 128-bit MurmurHash 3.

```
SELECT murmur3(from_base64('aaaaaa'));
```

Fungsi konversi

format_number (number) - Mengembalikan string diformat menggunakan simbol unit.

```
SELECT format_number(123456); -- '123K'
```

```
SELECT format_number(1000000); -- '1M'
```

Fungsi tanggal dan waktu

timezone_hour (timestamp) - Mengembalikan jam zona waktu offset dari stempel waktu.

```
SELECT EXTRACT(TIMEZONE_HOUR FROM TIMESTAMP '2020-05-10 12:34:56 +08:35');
```

timezone_minute (timestamp) - Mengembalikan menit zona waktu offset dari stempel waktu.

```
SELECT EXTRACT(TIMEZONE_MINUTE FROM TIMESTAMP '2020-05-10 12:34:56 +08:35');
```

Fungsi geospasial

to_encoded_polyline (Geometry) - Mengkodekan linestring atau multipoint ke polyline.

```
SELECT to_encoded_polyline(ST_GeometryFromText(
   'LINESTRING (-120.2 38.5, -120.95 40.7, -126.453 43.252)'));
```

from_encoded_polyline (varchar) - Mendekode polyline ke linestring.

```
SELECT ST_AsText(from_encoded_polyline('_p~iF~ps|U_ulLnnqC_mqNvxq`@'));
```

to_geojson_geometry (SphericalGeography) - Mengembalikan geografi bola tertentu dalam format GeoJSON.

```
SELECT to_geojson_geometry(to_spherical_geography(ST_GeometryFromText(
   'LINESTRING (0 0, 1 2, 3 4)')));
```

from_geojson_geometry (varchar) - Mengembalikan objek tipe geografi bola dari representasi GeoJSON, menghapus kunci/nilai non geometri. `Featuredan FeatureCollection` tidak didukung.

```
SELECT
 from_geojson_geometry(to_geojson_geometry(to_spherical_geography(ST_GeometryFromText(
   'LINESTRING (0 0, 1 2, 3 4)'))));
```

geometry_nearest_points (Geometry, Geometry) — Mengembalikan titik-titik pada setiap geometri yang terdekat satu sama lain. Jika salah satu geometri kosong, mengembalikan NULL. Jika tidak, mengembalikan deretan dua `Point` objek yang memiliki jarak minimum dari dua titik pada geometri. Poin pertama adalah dari argumen Geometri pertama, yang kedua dari argumen Geometri kedua. Jika ada beberapa pasangan dengan jarak minimum yang sama, satu pasangan dipilih secara sewenang-wenang.

```
SELECT geometry_nearest_points(ST_GeometryFromText(
   'LINESTRING (50 100, 50 200)'), ST_GeometryFromText(
   'LINESTRING (10 10, 20 20)'));
```

Atur fungsi Digest

make_set_digest (x) — Menyusun semua nilai masukan x ke dalam setdigest.

```
SELECT make_set_digest(value) FROM (VALUES 1, 2, 3) T(value);
```

Fungsi string

soundex (char) - Mengembalikan string karakter yang berisi representasi fonetik char.

```
SELECT name
FROM nation
WHERE SOUNDEX(name) = SOUNDEX('CHYNA'); -- CHINA
```

concat_ws (string0, string1,..., stringN) - Mengembalikan rangkaian menggunakan sebagai pemisah. `string1, string2, ..., stringN` `string0` Jika `string0` null, maka nilai kembalinya adalah null. Setiap nilai null yang disediakan dalam argumen setelah pemisah dilewati.

```
SELECT concat_ws(',', 'def', 'pqr', 'mno');
```

Fungsi jendela

GROUPS - Menambahkan dukungan untuk bingkai jendela berdasarkan kelompok.

```
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);
```

Peningkatan kinerja

Peningkatan kinerja di mesin Athena versi 3 meliputi yang berikut ini.

- Pengambilan metadata AWS Glue tabel yang lebih cepat — Kueri yang melibatkan beberapa tabel akan mengurangi waktu perencanaan kueri.
- Pemfilteran dinamis untuk RIGHT JOIN - Pemfilteran dinamis sekarang diaktifkan untuk penggabungan kanan yang memiliki kondisi gabungan kesetaraan, seperti pada contoh berikut.

  ```
  SELECT *
  FROM lineitem RIGHT JOIN tpch.tiny.supplier
  ON lineitem.suppkey = supplier.suppkey
  WHERE supplier.name = 'abc';
  ```
- Pernyataan siap besar - Meningkatkan ukuran header permintaan/respons HTTP default menjadi 2 MB untuk memungkinkan pernyataan disiapkan besar.
- approx_percentile () - `approx_percentile` Fungsi sekarang menggunakan `tdigest` alih-alih `qdigest` untuk mengambil nilai kuantil perkiraan dari distribusi. Ini menghasilkan kinerja yang lebih tinggi dan penggunaan memori yang lebih rendah. Perhatikan bahwa sebagai akibat dari perubahan ini, fungsi mengembalikan hasil yang berbeda dari yang terjadi di mesin Athena versi 2. Untuk informasi selengkapnya, lihat Fungsi approx_percentile mengembalikan hasil yang berbeda.

Peningkatan keandalan

Penggunaan dan pelacakan memori mesin umum di mesin Athena versi 3 telah ditingkatkan. Kueri besar kurang rentan terhadap kegagalan dari crash node.

Penyempurnaan sintaks kueri

INTERSECT ALL - Ditambahkan dukungan untuk. `INTERSECT ALL`

```
SELECT * FROM (VALUES 1, 2, 3, 4) INTERSECT ALL SELECT * FROM (VALUES 3, 4);
```

KECUALI SEMUA - Ditambahkan dukungan untuk`EXCEPT ALL`.

```
SELECT * FROM (VALUES 1, 2, 3, 4) EXCEPT ALL SELECT * FROM (VALUES 3, 4);
```

RANGE PRECEDING - Ditambahkan dukungan untuk `RANGE PRECEDING` dalam fungsi jendela.

```
SELECT sum(x) over (order by x range 1 preceding)
FROM (values (1), (1), (2), (2)) t(x);
```

MATCH_RECOGNITION - Ditambahkan dukungan untuk pencocokan pola baris, seperti pada contoh berikut.

```
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;
```

Format data dan peningkatan tipe data

Mesin Athena versi 3 memiliki format data dan peningkatan tipe data berikut.

- LZ4 dan ZSTD - Menambahkan dukungan untuk membaca data Parket terkompresi LZ4 dan ZSTD. Menambahkan dukungan untuk menulis data ORC terkompresi ZSTD.
- Tabel berbasis symlink - Menambahkan dukungan untuk membuat tabel berbasis symlink pada file Avro. Berikut contohnya.

  ```
  CREATE TABLE test_avro_symlink
  ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
  ...
  INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
  ```

- SphericalGeography SphericalGeography Jenis ini memberikan dukungan asli untuk fitur spasial yang diwakili pada koordinat geografis (kadang-kadang disebut koordinat geodetik, lat/lon, atau lon/lat). Koordinat geografis adalah koordinat bola yang dinyatakan dalam satuan sudut (derajat).

  `to_spherical_geography`Fungsi mengembalikan koordinat geografis (bola) dari koordinat geometris (planar), seperti pada contoh berikut.

```
SELECT to_spherical_geography(ST_GeometryFromText(
   'LINESTRING (-40.2 28.9, -40.2 31.9, -37.2 31.9)'));
```

Melanggar perubahan

Saat Anda bermigrasi dari mesin Athena versi 2 ke mesin Athena versi 3, perubahan tertentu dapat memengaruhi skema tabel, sintaks, atau penggunaan tipe data. Bagian ini mencantumkan pesan kesalahan terkait dan menyediakan solusi yang disarankan.

Perubahan sintaks kueri

IGNORE NULLS tidak dapat digunakan dengan fungsi jendela non-nilai

Pesan galat: Tidak dapat menentukan klausa perlakuan nol untuk `bool_or` fungsi.

Penyebab: sekarang `IGNORE NULLS` dapat digunakan hanya dengan fungsi nilai`first_value`,`last_value`,`nth_value`,`lead`, dan`lag`. Perubahan ini dibuat agar sesuai dengan spesifikasi ANSI SQL.

Solusi yang disarankan: Hapus `IGNORE NULLS` dari fungsi jendela non-nilai dalam string kueri.

Fungsi CONCAT harus memiliki dua argumen atau lebih

Pesan Kesalahan: INVALID_FUNCTION_ARGUMENT: Harus ada dua atau lebih argumen penggabungan

Penyebab: Sebelumnya, fungsi `CONCAT` string menerima argumen tunggal. Di mesin Athena versi 3, `CONCAT` fungsi ini membutuhkan minimal dua argumen.

Solusi yang disarankan: Ubah kejadian `CONCAT(str)` to`CONCAT(str, '')`.

Di mesin Athena versi 3, fungsi dapat memiliki tidak lebih dari 127 argumen. Untuk informasi selengkapnya, lihat Terlalu banyak argumen untuk panggilan fungsi.

Fungsi approx_percentile mengembalikan hasil yang berbeda

`approx_percentile`Fungsi ini mengembalikan hasil yang berbeda di mesin Athena versi 3 daripada yang dilakukan di mesin Athena versi 2.

Pesan kesalahan: Tidak ada.

Penyebab: `approx_percentile` Fungsi ini tunduk pada perubahan versi.

> **Important**
>
> Karena output `approx_percentile` fungsi adalah perkiraan, dan perkiraan dapat berubah dari satu versi ke versi berikutnya, Anda tidak harus bergantung pada `approx_percentile` fungsi untuk aplikasi kritis.

Solusi yang Disarankan: Untuk memperkirakan perilaku mesin Athena versi 2`approx_percentile`, Anda dapat menggunakan serangkaian fungsi yang berbeda di mesin Athena versi 3. Misalnya, Anda memiliki kueri berikut di mesin Athena versi 2:

```
SELECT approx_percentile(somecol, 2E-1)
```

Untuk memperkirakan output yang sama di mesin Athena versi 3, Anda dapat mencoba `qdigest_agg` dan `value_at_quantile` berfungsi, seperti pada contoh berikut. Perhatikan bahwa, bahkan dengan solusi ini, perilaku yang sama tidak dijamin.

```
SELECT value_at_quantile(qdigest_agg(somecol, 1), 2E-1)
```

Fungsi geospasial tidak mendukung input varbinary

Pesan kesalahan: FUNCTION_NOT_FOUND untuk ST_xxx

Penyebab: Beberapa fungsi geospasial tidak lagi mendukung jenis `VARBINARY` input lama atau tanda tangan fungsi terkait teks.

Solusi yang disarankan: Gunakan fungsi geospasial untuk mengonversi tipe input menjadi tipe yang didukung. Jenis input yang didukung ditunjukkan dalam pesan kesalahan.

Dalam klausa GROUP BY, kolom bersarang harus dikutip ganda

Pesan galat: "*column_name* “.” *nested_column* "harus berupa ekspresi agregat atau muncul di klausa GROUP BY

Penyebab: Mesin Athena versi 3 mengharuskan nama kolom bersarang dalam `GROUP BY` klausa dikutip ganda. Misalnya, kueri berikut menghasilkan kesalahan karena, dalam `GROUP BY` klausa, tidak `user.name` dikutip ganda.

```
SELECT "user"."name" FROM dataset
GROUP BY user.name
```

Solusi yang disarankan: Tempatkan tanda kutip ganda di sekitar nama kolom bersarang dalam `GROUP BY` klausa, seperti pada contoh berikut.

```
SELECT "user"."name" FROM dataset
GROUP BY "user"."name"
```

FilterNode Kesalahan tak terduga saat menggunakan OPTIMIZE pada tabel Iceberg

Pesan kesalahan: Tak terduga FilterNode ditemukan dalam rencana; mungkin konektor tidak dapat menangani ekspresi WHERE yang disediakan.

Penyebab: `OPTIMIZE` Pernyataan yang dijalankan pada tabel Iceberg menggunakan `WHERE` klausa yang menyertakan kolom non-partisi dalam ekspresi filternya.

Solusi yang Disarankan: `OPTIMIZE` Pernyataan ini mendukung penyaringan berdasarkan partisi saja. Saat Anda menjalankan `OPTIMIZE` tabel yang dipartisi, sertakan hanya kolom partisi dalam klausa. `WHERE` Jika Anda menjalankan `OPTIMIZE` tabel non-partisi, jangan tentukan klausa. `WHERE`

Log () urutan fungsi argumen

Di mesin Athena versi 2, urutan argumen untuk `log()` fungsi tersebut adalah. `log(`*value*`, `*base*`)` Di mesin Athena versi 3, ini telah berubah sesuai dengan `log(`*base*`, `*value*`)` standar SQL.

Fungsi menit () tidak mendukung interval tahun ke bulan

Pesan kesalahan: Parameter tak terduga (interval tahun ke bulan) untuk menit fungsi. Diharapkan: menit (stempel waktu dengan zona waktu), menit (waktu dengan zona waktu), menit (stempel waktu), menit (waktu), menit (interval hari ke detik).

Penyebab: Pada mesin Athena versi 3, pemeriksaan tipe telah dibuat lebih tepat sesuai `EXTRACT` dengan spesifikasi ANSI SQL.

Solusi yang disarankan: Perbarui kueri untuk memastikan jenis cocok dengan tanda tangan fungsi yang disarankan.

Ekspresi ORDER BY harus muncul di daftar SELECT

Pesan galat: Untuk SELECT DISTINCT, ekspresi ORDER BY harus muncul di daftar SELECT

Penyebab: Aliasing tabel yang salah digunakan dalam `SELECT` klausa.

Solusi yang disarankan: Periksa kembali apakah semua kolom dalam `ORDER BY` ekspresi memiliki referensi yang tepat dalam `SELECT DISTINCT` klausa.

Kegagalan kueri saat membandingkan beberapa kolom yang dikembalikan dari subquery

Contoh pesan kesalahan: Ekspresi nilai dan hasil subquery harus dari jenis yang sama: baris (varchar, varchar) vs baris (baris (varchar, varchar))

Penyebab: Karena pembaruan sintaks di mesin Athena versi 3, kesalahan ini terjadi ketika kueri mencoba membandingkan beberapa nilai yang dikembalikan dari subquery, dan pernyataan subquery melampirkan daftar kolomnya dalam tanda kurung, seperti pada contoh berikut. `SELECT`

```
SELECT *
FROM table1
WHERE (t1_col1, t1_col2)
IN (SELECT (t2_col1, t2_col2) FROM table2)
```

Solusi: Di mesin Athena versi 3, hapus tanda kurung di sekitar daftar kolom dalam `SELECT` pernyataan subquery, seperti pada contoh query diperbarui berikut.

```
SELECT *
FROM table1
WHERE (t1_col1, t1_col2)
IN (SELECT t2_col1, t2_col2 FROM table2)
```

SKIP adalah kata yang dicadangkan untuk kueri DHTML

Kata `SKIP` ini sekarang menjadi kata yang dicadangkan untuk kueri DML. seperti. `SELECT` Untuk digunakan `SKIP` sebagai pengenal dalam kueri DHTML, lampirkan dalam tanda kutip ganda.

Untuk informasi lebih lanjut tentang kata-kata yang dipesan di Athena, lihat. Kata kunci terpesan

Klausa SYSTEM_TIME dan SYSTEM_VERSION tidak digunakan lagi untuk perjalanan waktu

Pesan galat: masukan tidak cocok 'SYSTEM_TIME'. Mengharapkan: 'TIMESTAMP', 'VERSION'

Penyebab: Di mesin Athena versi 2, tabel Iceberg menggunakan `FOR SYSTEM_VERSION AS OF` klausa dan untuk stempel `FOR SYSTEM_TIME AS OF` waktu dan perjalanan waktu versi. Mesin Athena versi 3 menggunakan klausa `FOR TIMESTAMP AS OF` dan`FOR VERSION AS OF`.

Solusi yang disarankan: Perbarui kueri SQL untuk menggunakan `VERSION AS OF` klausa `TIMESTAMP AS OF` dan untuk operasi perjalanan waktu, seperti pada contoh berikut.

Perjalanan waktu dengan stempel waktu:

```
SELECT * FROM TABLE FOR TIMESTAMP AS OF (current_timestamp - interval '1' day)
```

Perjalanan waktu berdasarkan versi:

```
SELECT * FROM TABLE FOR VERSION AS OF 949530903748831860
```

Terlalu banyak argumen untuk konstruktor array

Pesan Kesalahan: TOO_MANY_ARGUMENTS: Terlalu banyak argumen untuk konstruktor array.

Penyebab: Jumlah maksimum elemen dalam konstruktor array sekarang diatur pada 254.

Solusi yang disarankan: Pecah elemen menjadi beberapa array yang masing-masing memiliki 254 atau lebih sedikit elemen, dan gunakan `CONCAT` fungsi untuk menggabungkan array, seperti pada contoh berikut.

```
CONCAT(
ARRAY[x1,x2,x3...x254],
ARRAY[y1,y2,y3...y254],
...
)
```

Pengenal terbatas panjang nol tidak diizinkan

Pesan galat: Pengenal terbatas tanpa panjang nol tidak diizinkan.

Penyebab: Sebuah query menggunakan string kosong sebagai alias kolom.

Solusi yang disarankan: Perbarui kueri untuk menggunakan alias yang tidak kosong untuk kolom.

Perubahan pemrosesan data

Validasi bucket

Pesan Kesalahan: HIVE_INVALID_BUCKET_FILES: Tabel sarang rusak.

Penyebab: Meja mungkin rusak. Untuk memastikan kebenaran kueri untuk tabel bucketed, Athena engine versi 3 memungkinkan validasi tambahan pada tabel berember untuk memastikan kebenaran kueri dan menghindari kegagalan yang tidak terduga saat runtime.

Solusi yang disarankan: Buat ulang tabel menggunakan mesin Athena versi 3.

Casting struct ke JSON sekarang mengembalikan nama bidang

Saat Anda mentransmisikan `struct` ke JSON dalam `SELECT` kueri di mesin Athena versi 3, pemeran sekarang mengembalikan nama bidang dan nilai (misalnya `useragent":null` "bukan hanya nilai (misalnya`null`,).

Perubahan penegakan keamanan tingkat kolom tabel gunung es

Pesan Kesalahan: Akses Ditolak: Tidak dapat memilih dari kolom

Penyebab: Tabel Iceberg dibuat di luar Athena dan menggunakan versi Apache Iceberg SDK lebih awal dari 0.13.0. Karena versi SDK sebelumnya tidak mengisi kolom AWS Glue, Lake Formation tidak dapat menentukan kolom yang diizinkan untuk diakses.

Solusi yang disarankan: Lakukan pembaruan menggunakan MENGUBAH PROPERTI SET TABEL pernyataan Athena atau gunakan Iceberg SDK terbaru untuk memperbaiki tabel dan memperbarui informasi kolom di. AWS Glue

Null dalam tipe data Daftar sekarang disebarkan ke UDF

Pesan kesalahan: Pengecualian Pointer Null

Penyebab: Masalah ini dapat memengaruhi Anda jika Anda menggunakan konektor UDF dan telah menerapkan fungsi Lambda yang ditentukan pengguna.

Athena engine versi 2 memfilter null dalam tipe data Daftar yang diteruskan ke fungsi yang ditentukan pengguna. Di mesin Athena versi 3, nol sekarang dipertahankan dan diteruskan ke UDF. Hal ini

dapat menyebabkan pengecualian pointer null jika UDF mencoba untuk dereferensi elemen null tanpa memeriksa.

Misalnya, jika Anda memiliki data `[null, 1, null, 2, 3, 4]` dalam sumber data asal seperti DynamoDB, berikut ini diteruskan ke fungsi Lambda yang ditentukan pengguna:

Mesin Athena versi 2: `[1,2,3,4]`

Mesin Athena versi 3: `[null, 1, null, 2, 3, 4]`

Solusi yang disarankan: Pastikan fungsi Lambda yang ditentukan pengguna Anda menangani elemen nol dalam tipe data daftar.

Substring dari array karakter tidak lagi berisi spasi empuk

Pesan kesalahan: Tidak ada kesalahan yang dilemparkan, tetapi string yang dikembalikan tidak lagi berisi spasi empuk. Misalnya, `substr(char[20],1,100)` sekarang mengembalikan string dengan panjang 20 bukan 100.

Solusi yang disarankan: Tidak diperlukan tindakan.

Pemaksaan tipe kolom desimal yang tidak didukung

Pesan kesalahan*: HIVE_CURSOR_ERROR: Gagal membaca file Parket: s3://DOC-EXAMPLE-BUCKET/ path/file_name .parquet atau Jenis kolom yang tidak didukung (varchar) untuk kolom Parket ([column_name]*

Penyebab: Athena engine versi 2 kadang-kadang berhasil (tetapi sering gagal) ketika mencoba pemaksaan tipe data dari desimal. `varchar` Karena Athena engine versi 3 memiliki validasi tipe yang memeriksa apakah tipe tersebut kompatibel sebelum mencoba membaca nilainya, upaya pemaksaan seperti itu sekarang selalu gagal.

Solusi yang Disarankan: Untuk mesin Athena versi 2 dan mesin Athena versi 3, ubah skema Anda AWS Glue untuk menggunakan tipe data numerik, bukan untuk kolom desimal dalam file Parket`varchar`. Baik rawl ulang data dan pastikan bahwa tipe data kolom baru adalah tipe desimal, atau secara manual membuat ulang tabel di Athena dan menggunakan `decimal(`*`precision`*`,` *`scale`*`)` sintaks untuk menentukan tipe data untuk kolom. decimal

Nilai NaN mengambang atau ganda tidak dapat lagi dilemparkan ke bigint

Pesan Kesalahan: INVALID_CAST_ARGUMENT: Tidak dapat mentransmisikan NaN nyata/ganda ke bigint

Penyebab: Di mesin Athena versi 3, tidak `NaN` bisa lagi dilemparkan ke 0 sebagai. `bigint`

Solusi yang disarankan: Pastikan `NaN` nilai tidak ada dalam `float` atau `double` kolom saat Anda mentransmisikan`bigint`.

uuid () fungsi mengembalikan perubahan tipe

Masalah berikut mempengaruhi tabel dan tampilan.

Pesan galat: Jenis sarang yang tidak didukung: uuid

Penyebab: Di mesin Athena versi 2, `uuid()` fungsi mengembalikan string, tetapi di mesin Athena versi 3, ia mengembalikan UUID semu yang dihasilkan secara acak (tipe 4). Karena tipe data kolom UUID tidak didukung di Athena, `uuid()` fungsi tidak dapat lagi digunakan secara langsung dalam kueri CTAS untuk menghasilkan kolom UUID di mesin Athena versi 3.

Misalnya, `CREATE TABLE` pernyataan berikut berhasil diselesaikan di mesin Athena versi 2 tetapi mengembalikan NOT_SUPPORTED: Unsupported Hive type: uuid di Athena engine versi 3:

```
CREATE TABLE uuid_table AS
   SELECT uuid() AS myuuid
```

Demikian pula, `CREATE VIEW` pernyataan berikut berhasil diselesaikan di mesin Athena versi 2 tetapi mengembalikan Jenis kolom tidak valid untuk kolom myuuid: Jenis sarang yang tidak didukung: uuid di mesin Athena versi 3:

```
CREATE VIEW uuid_view AS
   SELECT uuid() AS myuuid
```

Ketika tampilan yang dibuat di mesin Athena versi 2 ditanyakan di mesin Athena versi 3, kesalahan seperti berikut terjadi:

VIEW_IS_STALE: baris 1:15: Lihat 'awsdatacatalog.mydatabase.uuid_view' basi atau dalam keadaan tidak valid: kolom [myuuid] tipe uuid yang diproyeksikan dari tampilan kueri pada posisi 0 tidak dapat dipaksa ke kolom [myuuid] dari tipe varchar yang disimpan dalam definisi tampilan

Solusi yang Disarankan: Saat Anda membuat tabel atau tampilan, gunakan `cast()` fungsi untuk mengonversi `uuid()` output menjadi`varchar`, seperti pada contoh berikut:

```
CREATE TABLE uuid_table AS
```

```
   SELECT CAST(uuid() AS VARCHAR) AS myuuid
```

```
CREATE VIEW uuid_view AS
   SELECT CAST(uuid() AS VARCHAR) AS myuuid
```

Masalah paksaan CHAR dan VARCHAR

Gunakan solusi di bagian ini jika Anda mengalami masalah paksaan dengan dan `varchar` di `char` mesin Athena versi 3. Jika Anda tidak dapat menggunakan solusi ini, silakan hubungi. AWS Support

Kegagalan fungsi CONCAT dengan input CHAR dan VARCHAR campuran

Masalah: Kueri berikut berhasil pada mesin Athena versi 2.

```
SELECT concat(CAST('abc' AS VARCHAR(20)), '12', CAST('a' AS CHAR(1)))
```

Namun, pada mesin Athena versi 3, kueri yang sama gagal dengan yang berikut:

Pesan kesalahan: FUNCTION_NOT_FOUND: baris 1:8: Parameter tak terduga (varchar (20), varchar (2), char (1)) untuk fungsi concat. Diharapkan: concat (char (x), char (y)), concat (array (E), E) E, concat (E, array (E)) E, concat (array (E)) E, concat (varchar), concat (varbinary)

Solusi yang Disarankan: Saat menggunakan `concat` fungsi, lemparkan ke `char` atau`varchar`, tetapi tidak ke campuran keduanya.

SQL || kegagalan penggabungan dengan input CHAR dan VARCHAR

Pada mesin Athena versi 3, operator `||` penggabungan batang vertikal ganda memerlukan input. `varchar` Input tidak dapat berupa kombinasi `varchar` dan `char` jenis.

Pesan kesalahan: TYPE_NOT_FOUND: baris 1:26: Jenis tidak diketahui: char (65537)

Penyebab: Kueri yang digunakan `||` untuk menggabungkan a `char` dan a `varchar` dapat menghasilkan kesalahan, seperti pada contoh berikut.

```
SELECT CAST('a' AS CHAR) || CAST('b' AS VARCHAR)
```

Solusi yang Disarankan: Gabungkan `varchar` dengan`varchar`, seperti pada contoh berikut.

```
SELECT CAST('a' AS VARCHAR) || CAST('b' AS VARCHAR)
```

Kegagalan kueri CHAR dan VARCHAR UNION

Pesan galat: NOT_SUPPORTED: Jenis sarang yang tidak didukung: char (65536). Jenis CHAR yang didukung: CHAR (<= 255)

Penyebab: Sebuah query yang mencoba untuk menggabungkan `char` dan`varchar`, seperti dalam contoh berikut:

```
CREATE TABLE t1 (c1) AS SELECT CAST('a' as CHAR) as c1 UNION ALL SELECT CAST('b' AS
 VARCHAR) AS c1
```

Solusi yang Disarankan: Dalam kueri contoh, `'a'` gunakan sebagai `varchar` bukan`char`.

Ruang kosong yang tidak diinginkan setelah paksaan CHAR atau VARCHAR

Di Athena engine versi 3, kapan `char(X)` dan `varchar` data dipaksa ke satu jenis saat membentuk array atau kolom tunggal, `char(65535)` adalah tipe target, dan setiap bidang berisi banyak spasi trailing yang tidak diinginkan.

Penyebab: Mesin Athena versi 3 memaksa `varchar` dan `char(X)` ke `char(65535)` dan kemudian melapisi data dengan spasi.

Solusi yang Disarankan: Lemparkan setiap bidang secara eksplisit ke. `varchar`

Perubahan stempel waktu

Casting Timestamp dengan zona waktu ke perubahan perilaku varchar

Di mesin Athena versi 2, casting a `Timestamp` dengan zona waktu `varchar` menyebabkan beberapa literal zona waktu berubah (misalnya, `US/Eastern` diubah menjadi). `America/New_York` Perilaku ini tidak terjadi di mesin Athena versi 3.

Tanggal timestamp overflow melempar kesalahan

Pesan kesalahan: Millis overflow: XXX

Penyebab: Karena tanggal ISO 8601 tidak diperiksa untuk overflow di mesin Athena versi 2, beberapa tanggal menghasilkan stempel waktu negatif. Mesin Athena versi 3 memeriksa luapan ini dan memberikan pengecualian.

Solusi yang Disarankan: Pastikan stempel waktu berada dalam jangkauan.

Zona waktu politik dengan WAKTU tidak didukung

Pesan galat: LITERAL TIDAK VALID

Penyebab: Pertanyaan sepertiSELECT TIME '13:21:32.424 America/Los_Angeles'.

Solusi yang disarankan: Hindari menggunakan zona waktu politik denganTIME.

Ketidakcocokan presisi di kolom Timestamp menyebabkan kesalahan serialisasi

Pesan kesalahan *: SERIALIZATION_ERROR: Tidak dapat membuat serial kolom 'COLUMNZ' dari tipe 'stempel waktu (3) 'pada posisi X: Y*

COLUMNZ adalah nama output dari kolom yang menyebabkan masalah. Angka *X*: *Y* menunjukkan posisi kolom dalam output.

Penyebab: Mesin Athena versi 3 memeriksa untuk memastikan bahwa ketepatan stempel waktu dalam data sama dengan presisi yang ditentukan untuk tipe data kolom dalam spesifikasi tabel. Saat ini, presisi ini selalu 3. Jika data memiliki presisi yang lebih besar dari ini, kueri gagal dengan kesalahan yang dicatat.

Solusi yang disarankan: Periksa data Anda untuk memastikan bahwa stempel waktu Anda memiliki presisi milidetik.

Ketepatan stempel waktu yang salah dalam kueri UNLOAD dan CTAS untuk tabel Iceberg

Pesan kesalahan: Ketepatan stempel waktu yang salah untuk stempel waktu (6); presisi yang dikonfigurasi adalah MILLISECONDS

Penyebab: Mesin Athena versi 3 memeriksa untuk memastikan bahwa ketepatan stempel waktu dalam data sama dengan presisi yang ditentukan untuk tipe data kolom dalam spesifikasi tabel. Saat ini, presisi ini selalu 3. Jika data memiliki presisi yang lebih besar dari ini (misalnya, mikrodetik, bukan milidetik), kueri dapat gagal dengan kesalahan yang dicatat.

Solusi: Untuk mengatasi masalah ini, pertama-tama CAST presisi stempel waktu ke 6, seperti pada contoh CTAS berikut yang membuat tabel Iceberg. Perhatikan bahwa presisi harus ditentukan sebagai 6 bukan 3 untuk menghindari kesalahan presisi Timestamp (3) tidak didukung untuk Iceberg.

```
CREATE TABLE my_iceberg_ctas
WITH (table_type = 'ICEBERG', location = 's3://DOC-EXAMPLE-BUCKET/table_ctas/',
```

```
format = 'PARQUET')
AS SELECT id, CAST(dt AS timestamp(6)) AS "dt"
FROM my_iceberg
```

Kemudian, karena Athena tidak mendukung stempel waktu 6, berikan nilai lagi ke stempel waktu (misalnya, dalam tampilan). Contoh berikut menciptakan tampilan dari `my_iceberg_ctas` tabel.

```
CREATE OR REPLACE VIEW my_iceberg_ctas_view AS
SELECT cast(dt AS timestamp) AS dt
FROM my_iceberg_ctas
```

Membaca tipe Long sebagai Timestamp atau sebaliknya dalam file ORC sekarang menyebabkan kesalahan file ORC yang salah

Pesan kesalahan: Kesalahan saat membuka Hive split 'FILE (SPLIT POSITION) 'File ORC cacat. Tidak dapat membaca stempel waktu tipe SQL dari aliran ORC .long_type tipe LONG

Penyebab: Mesin Athena versi 3 sekarang menolak paksaan implisit dari tipe data ke atau dari ke`Long`. `Timestamp` `Timestamp` `Long` Sebelumnya, `Long` nilai secara implisit diubah menjadi stempel waktu seolah-olah mereka adalah milidetik epoch.

Solusi yang disarankan: Gunakan `from_unixtime` fungsi untuk mentransmisikan kolom secara eksplisit, atau gunakan `from_unixtime` fungsi untuk membuat kolom tambahan untuk kueri future.

Waktu dan interval tahun ke bulan tidak didukung

Pesan galat: TYPE MISMATCH

Penyebab: Mesin Athena versi 3 tidak mendukung waktu dan interval tahun ke bulan (misalnya,`SELECT TIME '01:00' + INTERVAL '3' MONTH`).

Timestamp overflow untuk format parket int96

Pesan galat: Nanos tidak valid timeOfDay

Penyebab: Stempel waktu meluap untuk format Parket. `int96`

Solusi yang disarankan: Identifikasi file tertentu yang memiliki masalah. Kemudian buat file data lagi dengan perpustakaan Parket terkenal up-to-date, atau gunakan Athena CTAS. Jika masalah berlanjut, hubungi dukungan Athena dan beri tahu kami bagaimana file data dihasilkan.

Ruang yang diperlukan antara nilai tanggal dan waktu saat casting dari string ke stempel waktu

Pesan galat: INVALID_CAST_ARGUMENT: Nilai tidak dapat dilemparkan ke stempel waktu.

Penyebab: Athena engine versi 3 tidak lagi menerima tanda hubung sebagai pemisah yang valid antara nilai tanggal dan waktu dalam string input ke. `cast` Misalnya, kueri berikut berfungsi di mesin Athena versi 2 tetapi tidak di mesin Athena versi 3:

```
SELECT CAST('2021-06-06-23:38:46' AS timestamp) AS this_time
```

Solusi yang disarankan: Di mesin Athena versi 3, ganti tanda hubung antara tanggal dan waktu dengan spasi, seperti pada contoh berikut.

```
SELECT CAST('2021-06-06 23:38:46' AS timestamp) AS this_time
```

to_iso8601 () stempel waktu mengembalikan perubahan nilai

Pesan galat: Tidak ada

Penyebab: Di mesin Athena versi 2, `to_iso8601` fungsi mengembalikan stempel waktu dengan zona waktu meskipun nilai yang diteruskan ke fungsi tidak termasuk zona waktu. Di mesin Athena versi 3, `to_iso8601` fungsi mengembalikan stempel waktu dengan zona waktu hanya ketika argumen yang dilewatkan termasuk zona waktu.

Misalnya, kueri berikut meneruskan tanggal saat ini ke `to_iso8601` fungsi dua kali: pertama sebagai stempel waktu dengan zona waktu, dan kemudian sebagai stempel waktu.

```
SELECT TO_ISO8601(CAST(CURRENT_DATE AS TIMESTAMP WITH TIME ZONE)),
 TO_ISO8601(CAST(CURRENT_DATE AS TIMESTAMP))
```

Output berikut menunjukkan hasil query di setiap mesin.

Mesin Athena versi 2:

#	_col0	_col1
1	2023-02-24T00:00:00.000Z	2023-02-24T00:00:00.000Z

Mesin Athena versi 3:

#	_col0	_col1
1	2023-02-24T00:00:00.000Z	2023-02-24T00:00:00.000

Solusi yang disarankan: Untuk mereplikasi perilaku sebelumnya, Anda dapat meneruskan nilai stempel waktu ke `with_timezone` fungsi sebelum meneruskannya`to_iso8601`, seperti pada contoh berikut:

```
SELECT to_iso8601(with_timezone(TIMESTAMP '2023-01-01 00:00:00.000', 'UTC'))
```

Hasil

#	_col0
1	2023-01-01T00:00:00.000Z

at_timezone () parameter pertama harus menentukan tanggal

Masalah: Di mesin Athena versi 3, `at_timezone` fungsi tidak dapat mengambil `time_with_timezone` nilai sebagai parameter pertama.

Penyebab: Tanpa informasi tanggal, tidak dapat ditentukan apakah nilai yang dilewati adalah siang hari atau waktu standar. Misalnya, `at_timezone('12:00:00 UTC', 'America/Los_Angeles')` ambigu karena tidak ada cara untuk menentukan apakah nilai yang dilewati adalah Pacific Daylight Time (PDT) atau Pacific Standard Time (PST).

Batasan

Mesin Athena versi 3 memiliki keterbatasan sebagai berikut.

- Kinerja kueri — Banyak kueri berjalan lebih cepat di mesin Athena versi 3, tetapi beberapa paket kueri dapat berbeda dari mesin Athena versi 2. Akibatnya, beberapa pertanyaan dapat berbeda dalam latensi atau biaya.
- Konektor Trino dan Presto - Baik konektor Trino maupun Presto tidak didukung. Gunakan Kueri Gabungan Amazon Athena untuk menghubungkan sumber data. Untuk informasi selengkapnya, lihat Menggunakan Amazon Athena.

- Eksekusi toleran kesalahan — Eksekusi toleran kesalahan Trino (Trino Tardigrade) tidak didukung.
- Batas parameter fungsi - Fungsi tidak dapat mengambil lebih dari 127 parameter. Untuk informasi selengkapnya, lihat Terlalu banyak argumen untuk panggilan fungsi.

Batas berikut diperkenalkan di Athena mesin versi 2 untuk memastikan bahwa permintaan tidak gagal karena keterbatasan sumber daya. Batas ini tidak dapat dikonfigurasi oleh pengguna.

- Jumlah elemen hasil— Jumlah elemen hasilndibatasi hingga 10.000 atau kurang untuk fungsi-fungsi berikut:`min(col, n)`,`max(col, n)`,`min_by(col1, col2, n)`, dan`max_by(col1, col2, n)`.
- SET PENGELOMPOKAN— Jumlah maksimum irisan dalam satu set pengelompokan adalah 2048.
- Panjang baris file teks maksimum - Panjang baris maksimum default untuk file teks adalah 200 MB.
- Fungsi urutan ukuran hasil maksimal— Ukuran hasil maksimum dari fungsi urutan adalah 50000 entri. Misalnya,`SELECT sequence(0,45000,1)`berhasil, tapi`SELECT sequence(0,55000,1)`gagal dengan pesan kesalahanHasil dari fungsi urutan tidak harus memiliki lebih dari 50000 entri. Batas ini berlaku untuk semua jenis input untuk fungsi urutan, termasuk cap waktu.

Versi mesin Athena 2

Mesin Athena versi 2 memperkenalkan perubahan berikut.

- Perbaikan dan fitur baru
  - Pengelompokan, bergabung, dan peningkatan subquery
  - Penyempurnaan tipe data
  - Fungsi yang ditambahkan
  - Peningkatan kinerja
  - Perbaikan terkait JSON
- Melanggar perubahan
  - Perbaikan bug
  - Perubahan fungsi geospasial
  - Kepatuhan ANSI SQL
  - Fungsi yang diganti
  - Batas

Perbaikan dan fitur baru

- JELASKAN dan JELASKAN ANALISIS - Anda dapat menggunakan `EXPLAIN` pernyataan di Athena untuk melihat rencana eksekusi untuk kueri SQL Anda. Gunakan `EXPLAIN ANALYZE` untuk melihat rencana eksekusi terdistribusi untuk kueri SQL Anda dan biaya setiap operasi. Untuk informasi selengkapnya, lihat Menggunakan JELASKAN dan JELASKAN ANALISIS di Athena.
- Kueri gabungan— permintaan Gabungan didukung di Athena mesin versi 2. Untuk informasi selengkapnya, lihat Menggunakan Amazon Athena.
- Fungsi geospasial— Lebih dari 25 fungsi geospasial telah ditambahkan. Untuk informasi selengkapnya, lihat Fungsi geospasial baru di mesin Athena versi 2.
- Skema Nest— Support telah ditambahkan untuk membaca skema Nest, yang mengurangi biaya.
- Pernyataan yang disiapkan — Gunakan pernyataan yang disiapkan untuk eksekusi berulang dari kueri yang sama dengan parameter kueri yang berbeda. Pernyataan yang disiapkan berisi parameter placeholder yang nilainya Anda lewati saat runtime. Pernyataan yang disiapkan membantu mencegah serangan injeksi SQL. Untuk informasi selengkapnya, lihat Menggunakan kueri berparameter.
- Dukungan evolusi skema— Dukungan evolusi skema telah ditambahkan untuk data dalam format Parquet.
  - Menambahkan dukungan untuk membaca kolom larik, peta, atau jenis baris dari partisi tempat skema partisi berbeda dari skema tabel. Ini dapat terjadi saat skema tabel diperbarui setelah partisi dibuat. Jenis kolom yang diubah harus kompatibel. Untuk jenis baris, bidang trailing dapat ditambahkan atau dijatuhkan, tetapi bidang yang sesuai (menurut ordinal) harus memiliki nama yang sama.
  - File ORC sekarang dapat memiliki kolom struct dengan bidang yang hilang. Ini memungkinkan skema tabel untuk diubah tanpa menulis ulang file ORC.
  - Kolom struct ORC sekarang dipetakan dengan nama ketimbang ordinal. Ini dengan benar menangani hilang atau tambahan struct bidang dalam ORC file.
- SQL OFFSET - `OFFSET` Klausul SQL sekarang didukung dalam pernyataan. `SELECT` Untuk informasi selengkapnya, lihat SELECT.
- Pernyataan UNLOAD — Anda dapat menggunakan `UNLOAD` pernyataan untuk menulis output `SELECT` kueri ke format PARQUET, ORC, AVRO, dan JSON. Untuk informasi selengkapnya, lihat MEMBONGKAR.

Pengelompokan, bergabung, dan peningkatan subquery

- Pengelompokan kompleks— Menambahkan dukungan untuk operasi pengelompokan kompleks.
- Subqueries berkorelasi— Menambahkan dukungan untuk subqueries berkorelasi di`IN`predikat dan untuk subqueries berkorelasi yang membutuhkan pemaksaan.
- GABUNG SILANG— Dukungan tambahan untuk`CROSS JOIN`terhadap`LATERAL`berasal tabel.
- SET PENGELOMPOKAN— Dukungan tambahan untuk`ORDER BY`klausul dalam agregasi untuk mengkueri yang menggunakan`GROUPING SETS`.
- Ekspresi Lambda— Menambahkan dukungan untuk dereferencing kolom baris dalam ekspresi Lambda.
- nilai null dalam semijoins— Menambahkan dukungan untuk nilai null di sisi kiri semijoin (yaitu,`IN`predikat dengan subqueries).
- Spasial Bergabung— Menambahkan dukungan untuk siaran spasial bergabung dan kiri spasial bergabung.
- Tumpahan ke disk— Untuk memori intensif`INNER JOIN`dan`LEFT JOIN`operasi, Athena offloads hasil operasi menengah ke disk. Ini memungkinkan eksekusi kueri yang memerlukan memori dalam jumlah besar.

Penyempurnaan tipe data

- INT untuk INTE— Dukungan tambahan untuk`INT`sebagai alias untuk`INTEGER`Jenis data.
- Jenis INTER— Dukungan tambahan untuk pengecoran`INTERVAL`jenis.
- IPADDRESS - Menambahkan `IPADDRESS` tipe baru untuk mewakili alamat IP dalam kueri DML. Dukungan tambahan untuk pengecoran antara`VARBINARY`Jenis dan`IPADDRESS`Jenis. `IPADDRESS`Jenis ini tidak dikenali dalam kueri DDL.
- BERBEDA DARI— Ditambahkan`IS DISTINCT FROM`Dukungan untuk`JSON`dan`IPADDRESS`jenis.
- Pemeriksaan kesetaraan Null— Pemeriksaan kesetaraan untuk nilai null di`ARRAY`,`MAP`, dan`ROW`struktur data sekarang didukung. Sebagai contoh, ungkapan`ARRAY ['1', '3', null] = ARRAY ['1', '2', null]`Pengembalian`false`. Sebelumnya, elemen null kembali pesan kesalahanperbandingan tidak didukung.
- Tipe baris pemaksaan— Pemaksaan antara jenis baris terlepas dari nama field sekarang diperbolehkan. Sebelumnya, jenis baris dipaksakan ke yang lain hanya jika nama bidang dalam jenis sumber cocok dengan jenis target, atau saat jenis target memiliki nama bidang anonim.
- Pengurangan waktu— Pengurangan yang dilaksanakan untuk semua`TIME`dan`TIMESTAMP`jenis.

- Unicode— Menambahkan dukungan untuk lolos Unicode urutan dalam string literal.
- Rangkaian VARBINARY— Dukungan tambahan untuk rangkaian`VARBINARY`nilai.

  Fungsi nilai jendela - Fungsi nilai jendela sekarang mendukung `IGNORE NULLS` dan`RESPECT NULLS`.

Jenis input tambahan untuk fungsi

Fungsi berikut sekarang menerima jenis input tambahan. Untuk informasi selengkapnya tentang setiap fungsi, kunjungi tautan yang sesuai dengan dokumentasi Presto.

- approx_distinct ()—approx_distinct ()Fungsi sekarang mendukung jenis berikut:`INTEGER`,`SMALLINT`,`TINYINT`,`DECIMAL`,`REAL`,`DATE`,`TIMESTAMP`,`TIMESTAMP WITH TIME ZONE`,`TIME`,`TIME WITH TIME ZONE`,`IPADDRESS`, dan`CHAR`.
- Avg (), sum () - Fungsi agregat avg () dan sum () sekarang mendukung tipe data. `INTERVAL`
- Lpad (), rpad () - Fungsi lpad dan rpad sekarang bekerja pada input. `VARBINARY`
- Min (), max () - Fungsi agregasi min () dan max () sekarang memungkinkan jenis input yang tidak diketahui pada waktu analisis kueri sehingga Anda dapat menggunakan fungsi dengan `NULL` literal.
- regexp_replace ()— Varian dariregexp_replace ()fungsi menambahkan yang dapat menjalankan fungsi Lambda untuk setiap penggantian.
- Sequence () — Menambahkan `DATE` varian untuk fungsi sequence (), termasuk varian dengan kenaikan langkah satu hari implisit.
- ST_Area ()—ST_Area ()fungsi geospasial sekarang mendukung semua jenis geometri.
- Substr () - Fungsi substr sekarang bekerja pada `VARBINARY` input.
- zip_dengan ()— Larik panjang tidak cocok sekarang dapat digunakan denganzip_dengan (). Posisi yang hilang diisi dengan nol. Sebelumnya, kesalahan dimunculkan saat larik panjang yang berbeda dilewatkan. Perubahan ini dapat membuat sulit untuk membedakan antara nilai-nilai yang awalnya nol dari nilai-nilai yang ditambahkan ke pad larik dengan panjang yang sama.

Fungsi yang ditambahkan

Daftar berikut berisi fungsi-fungsi yang baru dimulai di mesin Athena versi 2. Daftar ini tidak termasuk fungsi geospasial. Untuk daftar fungsi geospasial, lihat Fungsi geospasial baru di mesin Athena versi 2.

Untuk informasi selengkapnya tentang setiap fungsi, kunjungi tautan yang sesuai dengan dokumentasi Presto.

Fungsi agregat

reduce_agg ()

Fungsi dan operator array

array_sort ()- Varian dari fungsi ini menambahkan yang mengambil fungsi Lambda sebagai komparator.

ngram ()

Fungsi dan operator biner

dari_big_endian_32 ()

dari_ieee754_32 ()

dari_ieee754_64 ()

hmac_md5 ()

hmac_sha1 ()

hmac_sha256 ()

hmac_sha512 ()

seram_hash_v2_32 ()

seram_hash_v2_64 ()

to_big_endian_32 ()

to_ieee754_32 ()

to_ieee754_64 ()

Fungsi dan operator tanggal dan waktu

milidetik ()

parse_duration ()

to_milidetik ()

Fungsi peta dan operator

multimap_from_entries ()

Fungsi dan operator matematika

inverse_normal_cdf ()

wilson_interval_lower ()

wilson_interval_atas ()

Fungsi intisari kuantil

Fungsi kuantildan`qdigest`tipe quantile digest ditambahkan.

Fungsi dan operator string

hamming_distance ()

split_to_multimap ()

Peningkatan kinerja

Performa fitur berikut telah meningkat di Athena mesin versi 2.

Kinerja kueri

- Kinerja gabungan siaran — Peningkatan kinerja gabungan siaran dengan menerapkan pemangkasan partisi dinamis di node pekerja.
- Tabel Bucket— Peningkatan performa untuk menulis ke tabel bucketed saat data yang ditulis sudah dipartisi dengan tepat (misalnya, saat output dari gabungan bucketed).
- BERBEDA— Peningkatan performa untuk beberapa kueri yang menggunakan`DISTINCT`.

  Pemfilteran dinamis dan pemangkasan partisi — Peningkatan meningkatkan kinerja dan mengurangi jumlah data yang dipindai dalam kueri.
- Operasi filter dan proyeksi— Operasi filter dan proyeksi sekarang selalu diproses oleh kolom jika memungkinkan. Mesin secara otomatis mengambil keuntungan dari pengkodean kamus tempat efektif.
- Mengumpulkan pertukaran— Peningkatan performa untuk mengkueri dengan mengumpulkan pertukaran.

- Agregasi Global— Peningkatan performa untuk beberapa kueri yang melakukan agregasi global yang disaring.
- PENGELOMPOKAN SET, KUBUS, ROLLUP— Peningkatan performa untuk mengkueri yang melibatkan`GROUPING SETS`,`CUBE`atau`ROLLUP`, yang dapat Anda gunakan untuk menggabungkan beberapa set kolom dalam satu kueri.
- Filter yang sangat selektif— Meningkatkan performa kueri dengan filter yang sangat selektif.
- Gabung dan Agregat operasi — Kinerja `JOIN` dan `AGGREGATE` operasi telah ditingkatkan.
- KE— Meningkatkan performa kueri yang menggunakan`LIKE`predikat pada kolom`information_schema`tabel.
- ORDER OLEH— Peningkatan rencana, performa, dan penggunaan memori untuk pertanyaan yang melibatkan`ORDER BY`dan`LIMIT`untuk menghindari Data Exchange yang tidak perlu.
- PESANAN OLEH—`ORDER BY`sekarang didistribusikan secara default, memungkinkan lebih besar`ORDER BY`klausa yang akan digunakan.
- Konversi jenis ROW— Peningkatan performa saat mengkonversi antara`ROW`jenis.
- Jenis struktural— Peningkatan performa kueri yang memproses tipe struktural dan berisi pemindaian, bergabung, agregasi, atau penulisan tabel.
- Pemindaian tabel — Aturan optimasi telah diperkenalkan untuk menghindari pemindaian tabel duplikat dalam kasus tertentu.
- SERIKAT— Peningkatan performa untuk`UNION`kueri.

Kinerja perencanaan kueri

- Performa perencanaan— Peningkatan performa perencanaan untuk mengkueri yang bergabung dengan beberapa tabel dengan sejumlah besar kolom.
- Evaluasi predikat— Peningkatan performa evaluasi predikat selama predikat pushdown dalam perencanaan.
- Dukungan pushdown predikat untuk casting— Support predikat pushdown untuk*<column>* `IN` *<values list>*predikat tempat nilai-nilai dalam daftar nilai membutuhkan casting untuk mencocokkan jenis kolom.
- Predikat inferensi dan pushdown— Predikat inferensi dan pushdown diperpanjang untuk mengkueri yang menggunakan*<symbol>* `IN` *<subquery>*predikat.
- Timeout - Memperbaiki bug yang dalam kasus yang jarang terjadi menyebabkan batas waktu perencanaan kueri.

Bergabunglah dengan kinerja

- Bergabung dengan kolom peta— Meningkatkan performa bergabung dan agregasi yang mencakup kolom peta.
- Bergabung dengan syarat non-kesetaraan semata-mata— Peningkatan performa bergabung dengan hanya syarat non-kesetaraan dengan menggunakan bergabung loop Nest bukannya bergabung hash.
- Bergabung ke luar— Jenis distribusi bergabung sekarang secara otomatis dipilih untuk mengkueri yang melibatkan luar bergabung.
- Rentang atas fungsi bergabung— Peningkatan performa bergabung tempat syarat adalah rentang atas fungsi (misalnya,`a JOIN b ON b.x < f(a.x) AND b.x > g(a.x)`).
- S pill-to-disk - Memperbaiki bug spill-to-disk terkait dan masalah memori untuk meningkatkan kinerja dan mengurangi kesalahan memori dalam `JOIN` operasi.

Kinerja subquery

- Berkorelasi ADA subqueries— Peningkatan performa berkorelasi`EXISTS`subqueries.
- Subqueries berkorelasi dengan persamaan— Peningkatan dukungan untuk subqueries berkorelasi yang mengandung predikat kesetaraan.
- Subqueries berkorelasi dengan ketidaksetaraan— Peningkatan performa untuk subqueries berkorelasi yang mengandung ketidaksetaraan.
- Hitung (*) agregasi atas subkueri — Peningkatan kinerja **`count(*)`** agregasi atas subkueri dengan kardinalitas konstan yang diketahui.
- Propagasi filter permintaan luar— Peningkatan performa subqueries berkorelasi saat filter dari permintaan luar dapat di-deploy ke subquery.

Kinerja fungsi

- Fungsi Jendela Agregat— Peningkatan performa fungsi jendela agregat.
- element_at ()— Peningkatan performa`element_at()`untuk peta menjadi waktu yang konstan dan bukan sebanding dengan ukuran peta.
- Pengelompokan () - Peningkatan kinerja untuk kueri yang melibatkan. `grouping()`
- JSON pengecoran— Meningkatkan performa pengecoran dari`JSON`ke`ARRAY`atau`MAP`jenis.
- Fungsi Map-kembali— Peningkatan performa fungsi yang mengembalikan peta.

- M ap-to-map casting - Meningkatkan kinerja map-to-map pemeran.
- Min () dan max () — `max()` Fungsi `min()` dan telah dioptimalkan untuk menghindari pembuatan objek yang tidak perlu, sehingga mengurangi overhead pengumpulan sampah.
- row_number ()— Peningkatan performa dan penggunaan memori untuk mengkueri menggunakan`row_number()`diikuti oleh filter pada nomor baris yang dihasilkan.
- Fungsi Jendela— Peningkatan performa kueri yang berisi fungsi jendela dengan identik`PARTITION BY`dan`ORDER BY`Klausul.
- Fungsi Jendela— Peningkatan performa fungsi jendela tertentu (misalnya,`LAG`) yang mempunyai spesifikasi yang sama.

Kinerja geospasial

- Geometri serialisasi— Meningkatkan performa serialisasi nilai geometri.
- Fungsi geospasial— Meningkatkan performa`ST_Intersects()`,`ST_Contains()`,`ST_Touches()`,`ST_Within()`,`ST_Overlaps()`,`ST_Dis` dan`array_intersect()`.
- ST_jarak ()— Peningkatan performa bergabung kueri yang melibatkan`ST_Distance()`fungsi.
- ST_Persimpangan ()— Dioptimalkan`ST_Intersection()`fungsi untuk segi empat tepat sejajar dengan paksi koordinat (sebagai contoh, poligon yang dihasilkan oleh`ST_Envelope()`dan`bing_tile_polygon()`fungsi).

Perbaikan terkait JSON

Fungsi Peta

- Peningkatan performa subscript peta dari`O(n)`ke`O(1)`dalam semua kasus. Sebelumnya, hanya peta yang diproduksi oleh fungsi dan pembaca tertentu yang memanfaatkan perbaikan ini.
- Menambahkan`map_from_entries()`dan`map_entries()`fungsi.

Casting

- Menambahkan kemampuan untuk dilemparkan ke`JSON`dari`REAL`,`TINYINT`atau`SMALLINT`.
- Sekarang Anda dapat melemparkan`JSON`ke`ROW`bahkan jika`JSON`tidak berisi setiap bidang di`ROW`.
- Peningkatan performa`CAST(json_parse(...) AS ...)`.

- Meningkatkan performa pengecoran dari`JSONkeARRAYatauMAP`jenis.

Fungsi JSON Baru

- is_json_scalar ()

Melanggar perubahan

Melanggar perubahan termasuk perbaikan bug, perubahan fungsi geospasial, fungsi yang diganti, dan pengenalan batas. Perbaikan ANSI SQL kepatuhan mungkin melanggar permintaan yang bergantung pada perilaku non-standar.

Perbaikan bug

Perubahan berikut memperbaiki masalah perilaku yang menyebabkan permintaan untuk menjalankan berhasil, tetapi dengan hasil yang tidak akurat.

- kolom parket fixed_len_byte_array sekarang diterima sebagai DECIMAL - Kueri pada kolom Parket tipe `fixed_len_byte_array` berhasil dan mengembalikan nilai yang benar jika dijelaskan seperti dalam Skema Parket. `DECIMAL` Kueri pada `fixed_len_byte_array` kolom tanpa `DECIMAL` anotasi gagal dengan kesalahan. Sebelumnya, kueri pada `fixed_len_byte_array` kolom tanpa anotasi DECIMAL berhasil tetapi mengembalikan nilai yang tidak dapat dipahami.
- json_parse () tidak lagi mengabaikan karakter trailing— Sebelumnya, masukan seperti`[1,2]abc`akan berhasil mengurai sebagai`[1,2]`. Menggunakan karakter trailing sekarang menghasilkan pesan kesalahanTidak dapat mengkonversi '[1, 2] abc' ke JSON.
- Putaran () presisi desimal dikoreksi - `round(x, d)` sekarang membulatkan dengan benar `x` kapan `x` DESIMAL atau kapan DESIMAL dengan skala 0 dan `x` merupakan bilangan bulat negatif. `d` Sebelumnya, tidak ada pembulatan yang terjadi dalam kasus ini.
- bulat (x, d) dan memotong (x, d) — Parameterddalam tanda tangan fungsi `round(x, d)` dan `truncate(x, d)` sekarang tipe `INTEGER`. Sebelumnya, d bisa berupa tipe`BIGINT`.
- Map () dengan kunci duplikat — `map()` sekarang memunculkan kesalahan pada kunci duplikat daripada secara diam-diam menghasilkan peta yang rusak. Kueri yang saat ini membangun peta nilai menggunakan kunci duplikat sekarang gagal dengan kesalahan.
- map_from_entries () menimbulkan kesalahan dengan entri null—`map_from_entries()`sekarang menimbulkan kesalahan saat larik masukan berisi entri null. Pertanyaan yang membangun peta dengan melewati`NULL`sebagai nilai sekarang gagal.

- Tabel— Tabel yang memiliki tipe partisi yang tidak didukung tidak dapat lagi dibuat.
- Peningkatan stabilitas numerik dalam fungsi statistik— Stabilitas numerik untuk fungsi statistik`corr()`,`covar_samp()`,`regr_intercept()`, dan`regr_slope()`telah diperbaiki.
- Presisi TIMESTAMP yang didefinisikan dalam parket sekarang dihormati — Ketepatan `TIMESTAMP` nilai dan presisi yang ditentukan untuk `TIMESTAMP` kolom dalam skema Parket sekarang harus cocok. Presisi yang tidak cocok menghasilkan stempel waktu yang salah.
- Informasi zona waktu— Informasi zona waktu sekarang dihitung menggunakanjava.timepaket dari Java 1.8 SDK.
- JUMLAH dari INTERVAL_DAY_TO_SECOND dan INTERVAL_YEAR_TO_— Anda tidak dapat lagi menggunakan`SUM(NULL)`secara langsung. Untuk menggunakan`SUM(NULL)`, cor`NULL`ke tipe data seperti`BIGINT`,`DECIMAL`,`REAL`,`DOUBLE`,`INTERVAL_DAY_TO_SECOND`atau`INTERVAL_YEAR_TO_MONTH`.

Perubahan fungsi geospasial

Perubahan yang dilakukan pada fungsi geospasial adalah sebagai berikut.

- Perubahan nama fungsi— Beberapa nama fungsi telah berubah. Untuk informasi selengkapnya, lihat Perubahan nama fungsi geospasial di mesin Athena versi 2.
- Masukan VARBINA—`VARBINARY`tidak lagi didukung secara langsung untuk input ke fungsi geospasial. Misalnya, untuk menghitung luas geometri secara langsung, geometri sekarang harus menjadi masukan baik`VARCHAR`atau`GEOMETRY`format. Solusi adalah dengan menggunakan mengubah fungsi, seperti dalam contoh berikut.
  - Untuk menggunakan`ST_area()`untuk menghitung luas untuk`VARBINARY`masukan dalam format Biner Terkenal (WKB), berikan masukan ke`ST_GeomFromBinary()`pertama, misalnya:

    ```
    ST_area(ST_GeomFromBinary(<wkb_varbinary_value>))
    ```

  - Untuk menggunakan`ST_area()`untuk menghitung luas untuk`VARBINARY`masukan dalam format biner warisan, lulus masukan yang sama ke`ST_GeomFromLegacyBinary()`fungsi pertama, misalnya:

    ```
    ST_area(ST_GeomFromLegacyBinary(<legacy_varbinary_value>))
    ```

- ST_ ExteriorRing () dan ST_polygon () — `ST_ExteriorRing()`dan `ST_Polygon()`sekarang hanya menerima poligon sebagai input. Sebelumnya, fungsi-fungsi ini keliru menerima geometri lainnya.

- ST_jarak ()— Seperti yang dipersyaratkan olehSpesifikasi SQL/MM,`ST_Distance()`fungsi sekarang kembali`NULL`jika salah satu input adalah geometri kosong. Sebelumnya,`NaN`dikembalikan.

Kepatuhan ANSI SQL

Masalah sintaks dan perilaku berikut telah diperbaiki untuk mengikuti standar ANSI SQL.

- Operasi cast ()- Cast () operasi dari REAL atau DOUBLE untuk DECIMAL sekarang sesuai dengan standar SQL. Misalnya,`cast (double '10000000000000000000000000000000' as decimal(38))`sebelumnya dikembalikan`10000000000000000536616220439347 2`Tapi sekarang kembali`10000000000000000000000000000000`.
- BERGABUNG... MENGGUNAKAN—`JOIN ... USING`sekarang sesuai dengan semantik SQL standar. Sebelumnya,`JOIN ... USING`diperlukan kualifikasi nama tabel dalam kolom, dan kolom dari kedua tabel akan hadir dalam output. Kualifikasi tabel sekarang tidak valid dan kolom hadir hanya sekali dalam output.
- Jenis ROW literal dihapus— Format literal jenis ROW`ROW<int, int>(1, 2)`tidak lagi didukung. Gunakan sintaks`ROW(1 int, 2 int)`Sebagai gantinya
- Semantik agregasi yang dikelompokkan— Agregasi dikelompokkan menggunakan`IS NOT DISTINCT FROM`semantik bukan semantik kesetaraan. Agregasi yang dikelompokkan sekarang mengembalikan hasil yang benar dan menunjukkan performa yang lebih baik saat mengelompokkan`NaN`nilai floating point. Pengelompokan pada peta, daftar, dan jenis baris yang berisi nulls didukung.
- Jenis dengan tanda petik tidak lagi dibenarkan— Sesuai dengan standar ANSI SQL, tipe data tidak dapat lagi dilampirkan dalam tanda kutip. Misalnya,`SELECT "date" '2020-02-02'`tidak lagi kueri yang valid. Sebaliknya, gunakan sintaks`SELECT date '2020-02-02'`.
- Akses bidang baris anonim— Anonymous baris kolom tidak lagi dapat diakses dengan menggunakan sintaks `[.field0, .field1, ...]`.
- Operasi pengelompokan yang kompleks— Operasi pengelompokan kompleks`GROUPING SETS`,`CUBE`, dan`ROLLUP`tidak mendukung pengelompokan pada ekspresi yang terdiri dari kolom masukan. Hanya nama kolom yang diperbolehkan.

Fungsi yang diganti

Fungsi berikut tidak lagi didukung dan telah digantikan oleh sintaks yang menghasilkan hasil yang sama.

- information_schema. __internal_partitions__ - Penggunaan `__internal_partitions__` tidak lagi didukung di mesin Athena versi 2. Untuk sintaks yang setara, gunakan`SELECT * FROM "<table_name>$partitions"`atau`SHOW PARTITIONS`. Untuk informasi selengkapnya, lihat Daftar partisi untuk tabel tertentu.
- Fungsi geospasial yang diganti - Untuk daftar fungsi geospasial yang namanya telah berubah, lihat Perubahan nama fungsi geospasial di mesin Athena versi 2.

Batas

Batas berikut diperkenalkan di Athena mesin versi 2 untuk memastikan bahwa permintaan tidak gagal karena keterbatasan sumber daya. Batas ini tidak dapat dikonfigurasi oleh pengguna.

- Jumlah elemen hasil— Jumlah elemen hasilndibatasi hingga 10.000 atau kurang untuk fungsi-fungsi berikut:`min(col, n)`,`max(col, n)`,`min_by(col1, col2, n)`, dan`max_by(col1, col2, n)`.
- SET PENGELOMPOKAN— Jumlah maksimum irisan dalam satu set pengelompokan adalah 2048.
- Panjang baris file teks maksimum - Panjang baris maksimum default untuk file teks adalah 200 MB.
- Fungsi urutan ukuran hasil maksimal— Ukuran hasil maksimum dari fungsi urutan adalah 50000 entri. Misalnya,`SELECT sequence(0,45000,1)`berhasil, tapi`SELECT sequence(0,55000,1)`gagal dengan pesan kesalahanHasil dari fungsi urutan tidak harus memiliki lebih dari 50000 entri. Batas ini berlaku untuk semua jenis input untuk fungsi urutan, termasuk cap waktu.

Referensi SQL untuk Athena

Amazon Athena mendukung subset pernyataan, fungsi, operator, dan tipe data Data Definition Language (DDL) and Data Manipulation Language (DML). Dengan beberapa pengecualian, Athena DDL didasarkan pada HiveQL DDL dan Athena DMLdidasarkan pada Trino. Untuk informasi selengkapnya tentang versi mesin Aurora, lihat Pembuatan versi mesin Athena.

Topik

- Tipe data di Amazon Athena
- Kueri, fungsi, dan operator DML
- Pernyataan DDL
- Pertimbangan dan batasan untuk kueri SQL di Amazon Athena

Tipe data di Amazon Athena

Saat menjalankan`CREATE TABLE`, Anda menentukan nama kolom dan tipe data yang dapat berisi setiap kolom. Tabel yang Anda buat disimpan di AWS Glue Data Catalog.

Untuk memfasilitasi interoperabilitas dengan mesin query lainnya, Athena menggunakan nama tipe data Apache Hive untuk pernyataan DDL seperti. `CREATE TABLE` Untuk kueri DHTML seperti`SELECT`,, dan`CTAS`, `INSERT INTO` Athena menggunakan nama tipe data Trino. Tabel berikut menunjukkan tipe data yang didukung di Athena. Dimana tipe DDL dan DML berbeda dalam hal nama, ketersediaan, atau sintaks, mereka ditampilkan dalam kolom terpisah.

DDL	DML~	Deskripsi
BOOLEAN		Nilai adalah `true` dan`false`.
TINYINT		Bilangan bulat bertanda 8-bit dalam format komplemen dua, dengan nilai minimum -2 ^{7} dan nilai maksimum 2 ^{7} -1.
SMALLINT		Integer bertanda 16-bit dalam format komplemen dua, dengan nilai minimum -2 ^{15} dan nilai maksimum 2 ^{15} -1.
INT, BILANGAN BULAT		Nilai bertanda 32-bit dalam format komplemen dua, dengan nilai minimum -2 ^{31} dan nilai maksimum 2 ^{31} -1.
BIGINT		Integer bertanda 64-bit dalam format komplemen dua, dengan nilai minimum -2 ^{63} dan nilai maksimum 2 ^{63} -1.
FLOAT	REAL	Nomor floating point presisi tunggal bertanda 32-bit. Kisarannya adalah 1.40129846432481707e-45 hingga 3.40282346638528860e+38, positif atau negatif. Mengikuti Standar IEEE untuk Aritmatika Floating-Point (IEEE 754).
DOUBLE		Nomor floating point presisi ganda yang ditandata ngani 64-bit. Kisarannya adalah 4.9406564584124654 4e-324d hingga 1.79769313486231570e+308d, positif atau negatif. Mengikuti Standar IEEE untuk Aritmatika Floating-Point (IEEE 754).

DDL	DML~	Deskripsi
DESIMAL (presisi, skala)		*precision* adalah jumlah total digit. *scale*(opsional) adalah jumlah digit di bagian pecahan dengan default 0. Sebagai contoh, gunakan definisi jenis ini:`decimal(11,5)` ,`decimal(15)` . Nilai maksimum untuk *presisi* adalah 38, dan nilai maksimum untuk *skala* adalah 38.
CHAR, CHAR (panjang)		Data karakter panjang tetap, dengan panjang tertentu antara 1 dan 255, seperti char (10). Jika *panjang* ditentukan, string terpotong pada panjang yang ditentukan saat dibaca. Jika string data yang mendasarinya lebih panjang, string data yang mendasarinya tetap tidak berubah. Untuk informasi lebih lanjut, lihat tipe data CHAR Hive.
STRING	VARCHAR	Data karakter panjang variabel.
VARCHAR (panjang)		Data karakter panjang variabel dengan panjang baca maksimum. String terpotong pada panjang yang ditentukan saat dibaca. Jika string data yang mendasarinya lebih panjang, string data yang mendasarinya tetap tidak berubah.
BINARY	VARBINARY	Data biner panjang variabel.
TIME		Waktu dalam sehari dengan presisi milidetik.
Tidak tersedia	WAKTU (*presisi*)	Waktu dalam sehari dengan presisi tertentu. `TIME(3)`setara dengan`TIME`.
Tidak tersedia	TIME WITH TIME ZONE	Waktu dalam satu zona waktu. Zona waktu harus ditentukan sebagai offset dari UTC.
DATE		Tanggal kalender dengan tahun, bulan, dan hari.

DDL	DML~	Deskripsi
TIMESTAMP	STEMPEL WAKTU, STEMPEL WAKTU TANPA ZONA WAKTU	Tanggal kalender dan waktu hari dengan presisi milidetik.
Tidak tersedia	TIMESTAMP (*presisi*), TIMESTAMP (*presisi*) TANPA ZONA WAKTU	Tanggal kalender dan waktu hari dengan presisi tertentu. `TIMESTAMP(3)` setara dengan`TIMESTAMP` .
Tidak tersedia	TIMESTAMP WITH TIME ZONE	Tanggal kalender dan waktu dalam zona waktu. Zona waktu dapat ditentukan sebagai offset dari UTC, sebagai nama zona waktu IANA, atau menggunakan UTC, UT, Z, atau GMT.
Tidak tersedia	TIMESTAMP (*presisi*) DENGAN ZONA WAKTU	Tanggal kalender dan waktu hari dengan presisi tertentu, dalam zona waktu.
Tidak tersedia	INTERVAL TAHUN KE BULAN	Interval satu atau lebih bulan penuh
Tidak tersedia	INTERVAL HARI KE DETIK	Interval satu atau lebih detik, menit, jam, atau hari
ARRAY< element_type >	ARRAY [*element_type*]	Sebuah array nilai. Semua nilai harus memiliki tipe yang sama.

DDL	DML~	Deskripsi
PETA < key_type, value_type >	*PETA (key_type, value_type)*	Peta di mana nilai dapat dicari dengan kunci. Semua kunci harus memiliki nilai yang sama, dan semua nilai harus memiliki nilai yang sama.
STRUCT< field_name_1: field_type_1, field_name_2: field_type_2 ,... >	*BARIS (field_name_1 field_type_1, field_name_2 field_type_2 ,...)*	Struktur data dengan bidang bernama dan nilainya.
Tidak tersedia	JSON	Jenis nilai JSON, yang dapat berupa objek JSON, array JSON, nomor JSON, string JSON, atau. `true` `false` `null`
Tidak tersedia	UUID	Sebuah UUID (Universalally Unique Identifier).
Tidak tersedia	IPADDRESS	Alamat IPv4 atau IPv6.
Tidak tersedia	HyperLogLog	Tipe data ini mendukung perkiraan fungsi internal. Untuk informasi lebih lanjut tentang setiap jenis, kunjungi tautan ke entri yang sesuai dalam dokumentasi Trino.
	P4 HyperLogLog	
	SetDigest	
	QDigest	
	TDigest	

Contoh tipe data

Tabel berikut menunjukkan contoh literal untuk tipe data DHTML.

Tipe data	Contoh
BOOLEAN	`true` `false`
TINYINT	`TINYINT '123'`
SMALLINT	`SMALLINT '123'`
INT, BILANGAN BULAT	`123456790`
BIGINT	`BIGINT '1234567890'` `2147483648`
REAL	`'123456.78'`
DOUBLE	`1.234`
DESIMAL (presisi, skala)	`DECIMAL '123.456'`
CHAR, CHAR (panjang)	`CHAR 'hello world'`, `CHAR 'hello ''world''!'`
VARCHAR, VARCHAR (panjang)	`VARCHAR 'hello world'`, `VARCHAR 'hello ''world''!'`
VARBINARY	`X'00 01 02'`
WAKTU, WAKTU (*presisi*)	`TIME '10:11:12'` , `TIME '10:11:12.345'`
TIME WITH TIME ZONE	`TIME '10:11:12.345 -06:00'`
DATE	`DATE '2024-03-25'`
TIMESTAMP, TIMESTAMP TANPA ZONA WAKTU, TIMESTAMP (presisi),	`TIMESTAMP '2024-03-25 11:12:13'` , `TIMESTAMP '2024-03-25 11:12:13.456'`

Tipe data	Contoh
TIMESTAMP (presisi) TANPA ZONA WAKTU	
TIMESTAMP DENGAN ZONA WAKTU, TIMESTAMP (*presisi*) DENGAN ZONA WAKTU	`TIMESTAMP '2024-03-25 11:12:13.456 Europe/Berlin'`
INTERVAL TAHUN KE BULAN	`INTERVAL '3' MONTH`
INTERVAL HARI KE DETIK	`INTERVAL '2' DAY`
ARRAY [*element_type*]	`ARRAY['one', 'two', 'three']`
PETA (key_type, value_type)	`MAP(ARRAY['one', 'two', 'three'], ARRAY[1, 2, 3])` Perhatikan bahwa peta dibuat dari array kunci dan array nilai.

BARIS (field_name_1 field_type_1, field_name_2 field_type_2 ,...)	`ROW('one', 'two', 'three')`
	
	Perhatikan bahwa baris yang dibuat dengan cara ini tidak memiliki nama kolom. Untuk menambahkan nama kolom, Anda dapat menggunakanCAST, seperti pada contoh berikut:
	
	`CAST(ROW(1, 2, 3) AS ROW(one INT, two INT, three INT))`

JSON	`JSON '{"one":1, "two": 2, "three": 3}'`
UUID	`UUID '12345678-90ab-cdef-1234-567890abcdef'`

IPADDRESS	`IPADDRESS '10.0.0.1'`
	
	`IPADDRESS '2001:db8::1'`

Pertimbangan untuk tipe data

Batas ukuran

Untuk tipe data yang tidak menentukan batas ukuran, perlu diingat bahwa ada batas praktis 32MB untuk semua data dalam satu baris. Untuk informasi selengkapnya, lihat Row or column size limitation di Pertimbangan dan batasan untuk kueri SQL di Amazon Athena.

CHAR dan VARCHAR

`CHAR(`*n*`)`Nilai selalu memiliki hitungan *n* karakter. Misalnya, jika Anda mentransmisikan 'abc' ke`CHAR(7)`, 4 spasi tambahan ditambahkan.

Perbandingan `CHAR` nilai termasuk ruang depan dan belakang.

Jika panjang ditentukan untuk `CHAR` atau`VARCHAR`, string dipotong pada panjang yang ditentukan saat dibaca. Jika string data yang mendasarinya lebih panjang, string data yang mendasarinya tetap tidak berubah.

Untuk menghindari satu kutipan dalam `CHAR` atau`VARCHAR`, gunakan kutipan tunggal tambahan.

Untuk mentransmisikan tipe data non-string ke string dalam kueri DHTML, lemparkan ke tipe `VARCHAR` data.

Untuk menggunakan `substr` fungsi untuk mengembalikan substring dengan panjang tertentu dari tipe `CHAR` data, Anda harus terlebih dahulu melemparkan `CHAR` nilai sebagai`VARCHAR`. Dalam contoh berikut, `col1` menggunakan tipe `CHAR` data.

```
substr(CAST(col1 AS VARCHAR), 1, 4)
```

DECIMAL

Untuk menentukan nilai desimal sebagai literal dalam `SELECT` kueri, seperti saat memilih baris dengan nilai desimal tertentu, Anda dapat menentukan `DECIMAL` jenis dan mencantumkan nilai desimal sebagai literal dalam tanda kueri tunggal dalam kueri Anda, seperti pada contoh berikut.

```
SELECT * FROM my_table
WHERE decimal_value = DECIMAL '0.12'
```

```
SELECT DECIMAL '44.6' + DECIMAL '77.2'
```

Bekerja dengan data stempel waktu

Bagian ini menjelaskan beberapa pertimbangan untuk bekerja dengan data stempel waktu di Athena.

> **Note**
>
> Perawatan stempel waktu agak berubah antara mesin Athena versi 2 dan mesin Athena versi 3. Untuk informasi tentang kesalahan terkait stempel waktu yang dapat terjadi di mesin Athena versi 3 dan solusi yang disarankan, lihat di referensi. Perubahan stempel waktu Mesin Athena versi 3

Format untuk menulis data stempel waktu ke objek Amazon S3

Format di mana data stempel waktu harus ditulis ke objek Amazon S3 bergantung pada tipe data kolom dan pustaka SerDeyang Anda gunakan.

- Jika Anda memiliki kolom tabel tipe`DATE`, Athena mengharapkan kolom atau properti data yang sesuai menjadi string dalam format ISO`YYYY-MM-DD`, atau tipe tanggal bawaan seperti untuk Parket atau ORC.
- Jika Anda memiliki kolom tabel tipe`TIME`, Athena mengharapkan kolom atau properti data yang sesuai menjadi string dalam format ISO`HH:MM:SS`, atau tipe waktu bawaan seperti untuk Parket atau ORC.
- Jika Anda memiliki kolom tabel tipe`TIMESTAMP`, Athena mengharapkan kolom atau properti data yang sesuai menjadi string dalam format `YYYY-MM-DD HH:MM:SS.SSS` (perhatikan ruang antara tanggal dan waktu), atau tipe waktu bawaan seperti untuk Parket, ORC, atau Ion.

  > **Note**
  >
  > Stempel waktu SerDe OpenCSV adalah pengecualian dan harus dikodekan sebagai zaman UNIX resolusi milidetik.

Memastikan bahwa data yang dipartisi waktu cocok dengan bidang stempel waktu dalam catatan

Produsen data harus memastikan nilai partisi sejajar dengan data di dalam partisi. Misalnya, jika data Anda memiliki `timestamp` properti dan Anda menggunakan Firehose untuk memuat data ke Amazon S3, Anda harus menggunakan partisi dinamis karena partisi default Firehose adalah. wall-clock-based

Gunakan string sebagai tipe data untuk kunci partisi

Untuk alasan kinerja, lebih baik digunakan `STRING` sebagai tipe data untuk kunci partisi. Meskipun Athena mengenali nilai partisi dalam format `YYYY-MM-DD` sebagai tanggal saat Anda menggunakan `DATE` tipe, ini dapat menyebabkan kinerja yang buruk. Untuk alasan ini, kami menyarankan Anda menggunakan tipe `STRING` data untuk kunci partisi sebagai gantinya.

Cara menulis kueri untuk bidang stempel waktu yang juga dipartisi waktu

Cara Anda menulis kueri untuk bidang stempel waktu yang dipartisi waktu tergantung pada jenis tabel yang ingin Anda kueri.

Tabel sarang

Dengan tabel Hive yang paling umum digunakan di Athena, mesin kueri tidak memiliki pengetahuan tentang hubungan antara kolom dan kunci partisi. Untuk alasan ini, Anda harus selalu menambahkan predikat dalam kueri Anda untuk kolom dan kunci partisi.

Misalnya, Anda memiliki `event_time` kolom dan kunci `event_date` partisi dan ingin menanyakan peristiwa antara pukul 23:00 dan 03:00. Dalam hal ini, Anda harus menyertakan predikat dalam kueri Anda untuk kolom dan kunci partisi, seperti pada contoh berikut.

```
WHERE event_time BETWEEN start_time AND end_time
  AND event_date BETWEEN start_time_date AND end_time_date
```

Tabel gunung es

Dengan tabel Iceberg, Anda dapat menggunakan nilai partisi yang dihitung, yang menyederhanakan kueri Anda. Misalnya, tabel Iceberg Anda dibuat dengan `PARTITIONED BY` klausa seperti berikut:

```
PARTITIONED BY (event_date month(event_time))
```

Dalam hal ini, mesin kueri secara otomatis memangkas partisi berdasarkan nilai predikat. `event_time` Karena itu, kueri Anda hanya perlu menentukan predikat untuk`event_time`, seperti pada contoh berikut.

```
WHERE event_time BETWEEN start_time AND end_time
```

Untuk informasi selengkapnya, lihat Membuat tabel Iceberg.

Kueri, fungsi, dan operator DML

Mesin kueri Athena DMLumumnya mendukung sintaks Trino dan Presto dan menambahkan peningkatannya sendiri. Athena tidak mendukung semua fitur Trino atau Presto. Untuk informasi selengkapnya, lihat topik untuk pernyataan khusus dalam bagian ini danPertimbangan dan batasan. Untuk informasi tentang fungsi, lihatFungsi di Amazon Athena. Untuk informasi selengkapnya tentang versi mesin Aurora, lihat Pembuatan versi mesin Athena.

Untuk informasi tentang pernyataan DDL, lihatPernyataan DDL. Untuk daftar pernyataan DDL yang tidak didukung, lihat. DDL tidak didukung

SELECT

Mengambil baris data dari nol atau lebih tabel.

> **Note**
>
> Topik ini menyediakan informasi ringkasan untuk referensi. Untuk informasi lengkap tentang penggunaan`SELECT`dan bahasa SQL berada di luar cakupan dokumentasi ini. Untuk informasi tentang menggunakan SQL yang khusus untuk Athena, lihatPertimbangan dan batasan untuk kueri SQL di Amazon AthenadanMenjalankan kueri SQL menggunakan Amazon Athena. Untuk contoh membuat database, membuat tabel, dan menjalankan `SELECT` kueri pada tabel di Athena, lihat. Memulai

Sinopsis

```
[ 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**
>
> kata Reserved dalam pernyataan SQL SELECT harus tertutup dalam tanda kutip ganda. Untuk informasi selengkapnya, lihat Daftar kata kunci yang dicadangkan dalam pernyataan SQL SELECT.

Parameter

[DENGAN with_query [,...]]

Anda dapat menggunakan`WITH`untuk meratakan kueri Nest, atau untuk menyederhanakan subqueries.

Menggunakan `WITH` klausa untuk membuat kueri rekursif didukung mulai dari mesin Athena versi 3. Kedalaman rekursi maksimum adalah 10.

Parameter`WITH`mendahului`SELECT`dalam kueri dan mendefinisikan satu atau lebih subqueries untuk digunakan dalam`SELECT`kueri.

Setiap subquery mendefinisikan tabel sementara, mirip dengan definisi tampilan, yang dapat Anda referensi dalam`FROM`Klausul WHER Tabel yang digunakan hanya saat kueri berjalan.

`with_query`Sintaks adalah:

```
subquery_table_name [ ( column_name [, ...] ) ] AS (subquery)
```

Di mana:

- `subquery_table_name`adalah nama unik untuk tabel sementara yang mendefinisikan hasil`WITH`Klausul subkueri. Setiap`subquery`harus memiliki nama tabel yang dapat direferensikan dalam`FROM`Klausul WHER
- `column_name [, ...]`adalah daftar opsional nama kolom output. Jumlah nama kolom harus sama dengan atau kurang dari jumlah kolom yang ditentukan oleh`subquery`.
- `subquery`adalah pernyataan kueri.

[SEMUA | BERBEDA] select_expression

`select_expression`menentukan baris yang akan dipilih. A `select_expression` dapat menggunakan salah satu format berikut:

```
expression [ [ AS ] column_alias ] [, ...]
```

```
row_expression.* [ AS ( column_alias [, ...] ) ]
```

```
relation.*
```

```
*
```

- `expression [ [ AS ] column_alias ]`Sintaks menentukan kolom output. `[AS] column_alias`Sintaks opsional menentukan nama heading kustom yang akan digunakan untuk kolom dalam output.
- Untuk`row_expression.* [ AS ( column_alias [, ...] ) ]`, `row_expression` adalah ekspresi arbitrer dari tipe `ROW` data. Bidang baris menentukan kolom keluaran yang akan dimasukkan dalam hasil.
- Untuk`relation.*`, kolom termasuk dalam hasil. `relation` Sintaks ini tidak mengizinkan penggunaan alias kolom.
- Tanda bintang * menentukan bahwa semua kolom dimasukkan dalam set hasil.
- Dalam kumpulan hasil, urutan kolom sama dengan urutan spesifikasinya dengan ekspresi pilih. Jika ekspresi pilih mengembalikan beberapa kolom, urutan kolom mengikuti urutan yang digunakan dalam relasi sumber atau ekspresi tipe baris.
- Ketika alias kolom ditentukan, alias akan mengganti nama kolom atau kolom baris yang sudah ada sebelumnya. Jika ekspresi pilih tidak memiliki nama kolom, nama kolom anonim yang diindeks nol (`_col0`,`_col1`,`_col2, ...`) ditampilkan dalam output.
- `ALL` sebagai default. Menggunakan`ALL`diperlakukan sama seperti jika dihilangkan; semua baris untuk semua kolom yang dipilih dan duplikat disimpan.
- Gunakan`DISTINCT`untuk kembali nilai-nilai hanya berbeda saat kolom berisi nilai-nilai duplikat.

DARI from_item [,...]

Menunjukkan masukan untuk kueri, dimana`from_item`bisa menjadi tampilan, bergabung membangun, atau subquery seperti yang dijelaskan di bawah ini.

Parameter`from_item`dapat berupa:

- `table_name [ [ AS ] alias [ (column_alias [, ...]) ] ]`

Di manatable_nameadalah nama dari tabel target untuk memilih baris,aliasadalah nama untuk memberikan output dariSELECTpernyataan, dancolumn_aliasmendefinisikan kolom untukaliasditentukan.

-ATAU-

- `join_type from_item [ ON join_condition | USING ( join_column [, ...] ) ]`

  Di manajoin_typeadalah salah satu dari:

  - `[ INNER ] JOIN`
  - `LEFT [ OUTER ] JOIN`
  - `RIGHT [ OUTER ] JOIN`
  - `FULL [ OUTER ] JOIN`
  - `CROSS JOIN`
  - `ON join_condition | USING (join_column [, ...])`Di mana menggunakan`join_condition`memungkinkan Anda untuk menentukan nama kolom untuk bergabung kunci dalam beberapa tabel, dan menggunakan`join_column`membutuhkan`join_column`nada di kedua tabel.

[syarat WHERE]

Menyaring hasil sesuai dengan`condition`Anda tentukan, tempat`condition`umumnya memiliki sintaks berikut.

```
column_name operator value [[[AND | OR] column_name operator value] ...]
```

Parameter*`Operator`*dapat menjadi salah satu pembanding=,>,<,>=,<=,<>,!=.

Ekspresi subquery berikut juga dapat digunakan dalam`WHERE`Klausul WHER

- `[NOT] BETWEEN` *`integer_A`* `AND` *`integer_B`*- Menentukan rentang antara dua bilangan bulat, seperti dalam contoh berikut. Jika tipe data kolom adalah`varchar`, kolom harus dilemparkan ke integer terlebih dahulu.

  ```
  SELECT DISTINCT processid FROM "webdata"."impressions"
  WHERE cast(processid as int) BETWEEN 1500 and 1800
  ORDER BY processid
  ```

- `[NOT] LIKE` *`value`*— Pencarian untuk pola yang ditentukan. Gunakan tanda persen (`%`) sebagai karakter wildcard, seperti dalam contoh berikut.

```
SELECT * FROM "webdata"."impressions"
WHERE referrer LIKE '%.org'
```

- `[NOT] IN (`*`value`*`[, `*`value`*`[, ...])`— Menentukan daftar nilai yang mungkin untuk kolom, seperti dalam contoh berikut.

```
SELECT * FROM "webdata"."impressions"
WHERE referrer IN ('example.com','example.net','example.org')
```

[GROUP BY [SEMUA | berbeda] grouping_expressions [,...]]

Membagi output dari`SELECT`pernyataan ke dalam baris dengan nilai-nilai yang cocok.

`ALL`dan`DISTINCT`menentukan apakah duplikat pengelompokan set masing-masing menghasilkan baris output yang berbeda. Jika dihilangkan,`ALL`diasumsikan.

`grouping_expressions`memungkinkan Anda untuk melakukan operasi pengelompokan kompleks. Anda dapat menggunakan operasi pengelompokan kompleks untuk melakukan analisis yang memerlukan agregasi pada beberapa set kolom dalam satu kueri.

Parameter`grouping_expressions`elemen dapat fungsi apapun, seperti`SUM`,`AVG`, atau`COUNT`, dilakukan pada kolom input.

`GROUP BY`ekspresi dapat grup output dengan masukan nama kolom yang tidak muncul dalam output dari`SELECT`.

Semua ekspresi output harus baik fungsi agregat atau kolom hadir dalam`GROUP BY`Klausul WHER

Anda dapat menggunakan kueri tunggal untuk melakukan analisis yang membutuhkan menggabungkan beberapa set kolom.

Athena mendukung agregasi kompleks menggunakan`GROUPING SETS`,`CUBE`dan`ROLLUP`.`GROUP BY GROUPING SETS`menentukan beberapa daftar kolom untuk grup pada.`GROUP BY CUBE`menghasilkan semua set pengelompokan mungkin untuk satu set tertentu kolom.`GROUP BY ROLLUP`menghasilkan semua subtotal mungkin untuk satu set tertentu kolom. Operasi pengelompokan kompleks tidak mendukung pengelompokan ekspresi terdiri dari kolom input. Hanya nama kolom yang diperbolehkan.

Anda sering dapat menggunakanUNION ALLuntuk mencapai hasil yang sama seperti iniGROUP BY, tetapi kueri yang menggunakanGROUP BYmemiliki keuntungan membaca data satu kali, sedangkanUNION ALLmembaca data yang mendasari tiga kali dan dapat menghasilkan hasil yang tidak konsisten saat sumber data dapat berubah.

[Memiliki syarat]

Digunakan dengan fungsi agregat danGROUP BYKlausul WHER Mengontrol grup mana yang dipilih, menghilangkan grup yang tidak memuaskan`condition`. Penyaringan ini terjadi setelah grup dan agregat dihitung.

[{UNION | INTERSECT | KECUL} [SEMUA | berbeda] union_query]]

UNION,INTERSECT, danEXCEPTmenggabungkan hasil lebih dari satuSELECTpernyataan ke dalam kueri tunggal.ALLatauDISTINCTmengontrol keunikan baris termasuk dalam set hasil akhir.

UNIONmenggabungkan baris yang dihasilkan dari kueri pertama dengan baris yang dihasilkan dari kueri kedua. Untuk menghilangkan duplikat,UNIONmembangun sebuah tabel hash, yang mengkonsumsi memori. Untuk performa yang lebih baik, pertimbangkan menggunakanUNION ALLjika permintaan Anda tidak memerlukan penghapusan duplikat. BanyakUNIONdiproses kiri ke kanan kecuali Anda menggunakan tanda kurung untuk secara eksplisit menentukan urutan pemrosesan.

INTERSECTkembali hanya baris yang hadir dalam hasil kedua pertama dan kedua kueri.

EXCEPTmengembalikan baris dari hasil kueri pertama, tidak termasuk baris ditemukan oleh kueri kedua.

ALLmenyebabkan semua baris untuk dimasukkan, bahkan jika baris identik.

DISTINCTmenyebabkan baris hanya unik untuk dimasukkan dalam set hasil gabungan.

[ORDER BY ekspresi [ASC | DESC] [NULLS PERTAMA | NULLS TERAKHIR] [,...]]

Mengurutkan hasil yang ditetapkan oleh satu atau lebih output`expression`.

Saat klausa berisi beberapa ekspresi, hasil set diurutkan sesuai dengan yang pertama`expression`. Lalu yang kedua`expression`diterapkan ke baris yang memiliki nilai yang cocok dari ekspresi pertama, dan seterusnya.

Setiap`expression`dapat menentukan kolom keluaran dariSELECTatau nomor urut untuk kolom output dengan posisi, mulai dari satu.

ORDER BYdievaluasi sebagai langkah terakhir setelahGROUP BYatauHAVINGKlausul WHERASCdanDESCMenentukan apakah hasil diurutkan dalam urutan naik atau turun.

Default null pemesanan adalahNULLS LAST, terlepas dari urutan menaik atau menurun.

[Hitungan OFFSET [BARIS | BARIS]]

Gunakan OFFSET klausa untuk membuang sejumlah baris terdepan dari kumpulan hasil. Jika ORDER BY klausa ada, OFFSET klausa dievaluasi melalui kumpulan hasil yang diurutkan, dan himpunan tetap diurutkan setelah baris yang dilewati dibuang. Jika kueri tidak memiliki ORDER BY klausa, itu adalah arbitrer baris mana yang dibuang. Jika hitungan ditentukan OFFSET sama dengan atau melebihi ukuran set hasil, hasil akhirnya kosong.

LIMIT [menghitung | SEMUA]

Membatasi jumlah baris dalam hasil diatur kecount.LIMIT ALLadalah sama dengan menghilangkanLIMITKlausul WHER Jika kueri tidak memilikiORDER BYklausul, hasilnya sewenang-wenang.

Tablesample [BERNOULLI | SISTEM] (persentase

Operator opsional untuk memilih baris dari tabel berdasarkan metode sampling.

BERNOULLImemilih setiap baris untuk berada dalam sampel tabel dengan probabilitaspercentage. Semua blok fisik tabel dipindai, dan baris tertentu dilewati berdasarkan perbandingan antara sampelpercentagedan nilai acak dihitung pada saat runtime.

DenganSYSTEM, tabel dibagi menjadi segmen logis data, dan tabel sampel di granularity ini.

Entah semua baris dari segmen tertentu yang dipilih, atau segmen dilewati berdasarkan perbandingan antara sampelpercentagedan nilai acak dihitung pada saat runtime.SYSTEMsampling tergantung pada konektor. Metode ini tidak menjamin probabilitas sampling independen.

[UNNEST (array_or_map) [DENGAN ORDIALITAS]]

Memperluas larik atau peta ke dalam relasi. Larik diperluas menjadi satu kolom. Peta diperluas menjadi dua kolom (kunci,nilai).

Anda dapat menggunakanUNNESTdengan beberapa argumen, yang diperluas menjadi beberapa kolom dengan banyak baris sebagai argumen kardinalitas tertinggi.

kolom lainnya empuk dengan nulls.

ParameterWITH ORDINALITYklausa menambahkan kolom ordinalitas sampai akhir.

`UNNEST`biasanya digunakan dengan`JOIN`dan dapat referensi kolom dari relasi di sisi kiri`JOIN`.

Mendapatkan lokasi file untuk data sumber di Amazon S3

Untuk melihat lokasi file Amazon S3 untuk data dalam baris tabel, Anda dapat menggunakan`"$path"`dalam`SELECT`query, seperti dalam contoh berikut:

```
SELECT "$path" FROM "my_database"."my_table" WHERE year=2019;
```

Perintah ini akan menampilkan hasil berikut.

```
s3://DOC-EXAMPLE-BUCKET/datasets_mytable/year=2019/data_file1.json
```

Untuk kembali diurutkan, daftar unik dari jalur nama file S3 untuk data dalam tabel, Anda dapat menggunakan`SELECT DISTINCT`dan`ORDER BY`, seperti dalam contoh berikut.

```
SELECT DISTINCT "$path" AS data_source_file
FROM sampledb.elb_logs
ORDER By data_source_file ASC
```

Untuk mengembalikan hanya nama file tanpa path, Anda dapat melewati`"$path"`sebagai parameter untuk`regexp_extract`, seperti dalam contoh berikut.

```
SELECT DISTINCT regexp_extract("$path", '[^/]+$') AS data_source_file
FROM sampledb.elb_logs
ORDER By data_source_file ASC
```

Untuk mengembalikan data dari file tertentu, tentukan file di`WHERE`, seperti dalam contoh berikut.

```
SELECT *,"$path" FROM my_database.my_table WHERE "$path" = 's3://DOC-EXAMPLE-BUCKET/
my_table/my_partition/file-01.csv'
```

Untuk informasi selengkapnya dan untuk contoh, lihat artikel Pusat PengetahuanBagaimana saya bisa melihat file sumber Amazon S3 untuk baris dalam tabel Athena?.

> **Note**
>
> Di Athena, kolom metadata tersembunyi Hive atau Iceberg`$bucket`, `$file_modified_time`$`file_size`, dan tidak didukung untuk tampilan. `$partition`

Melarikan diri dari kutipan tunggal

Untuk melarikan diri kutipan tunggal, mendahului dengan kutipan tunggal lain, seperti dalam contoh berikut. Jangan mengelirukan ini dengan sebut harga berganda.

```
Select 'O''Reilly'
```

Hasil

```
O'Reilly
```

Sumber daya tambahan

Untuk informasi selengkapnya tentang penggunaan`SELECT`pernyataan di Athena, lihat sumber daya berikut.

Untuk informasi tentang ini	Lihat ini
Menjalankan kueri di Athena	Menjalankan kueri SQL menggunakan Amazon Athena
Untuk membuat tabel menggunakan `SELECT` API	Membuat tabel dari hasil query (CTAS)
Memasukkan data dari`SELECT`query ke dalam tabel lain	INSERT INTO
Menggunakan fungsi bawaan di`SELECT`pernyataan	Fungsi di Amazon Athena
Menggunakan fungsi yang ditetapkan pengguna di`SELECT`pernyataan	Query dengan fungsi yang ditentukan pengguna
Memkueri metadata Katalog Data	Mengkueri AWS Glue Data Catalog

INSERT INTO

Menyisipkan baris baru ke dalam tabel tujuan berdasarkan`SELECT`pernyataan permintaan yang berjalan pada tabel sumber, atau didasarkan pada satu set`VALUES`disediakan sebagai bagian dari pernyataan. Saat tabel sumber didasarkan pada data dasar dalam satu format, seperti CSV

atau JSON, dan tabel tujuan didasarkan pada format lain, seperti Parquet atau ORC, Anda dapat menggunakan`INSERT INTO`query untuk mengubah data yang dipilih ke dalam format tabel tujuan.

Pertimbangan dan batasan

Pertimbangkan hal berikut saat menggunakan`INSERT`Pertanyaan dengan Athena.

- Saat menjalankan`INSERT`permintaan pada tabel dengan data dasar yang dienkripsi di Amazon S3, output file yang`INSERT`query menulis tidak dienkripsi secara default. Kami merekomendasikan bahwa Anda mengenkripsi`INSERT`permintaan hasil jika Anda memasukkan ke dalam tabel dengan data terenkripsi.

  Untuk informasi selengkapnya tentang enkripsi hasil kueri menggunakan konsol, lihatMengenkripsi hasil kueri Athena yang disimpan di Amazon S3. Untuk mengaktifkan enkripsi menggunakan AWS CLI atau Athena API, gunakan `EncryptionConfiguration` properti StartQueryExecutiontindakan untuk menentukan opsi enkripsi Amazon S3 sesuai dengan kebutuhan Anda.
- Untuk `INSERT INTO` pernyataan, setelan pemilik bucket yang diharapkan tidak berlaku untuk lokasi tabel tujuan di Amazon S3. Setelan pemilik bucket yang diharapkan hanya berlaku untuk lokasi keluaran Amazon S3 yang Anda tentukan untuk hasil kueri Athena. Untuk informasi selengkapnya, lihat Menentukan lokasi hasil kueri menggunakan konsol Athena.
- Untuk `INSERT INTO` pernyataan yang sesuai dengan ACID, lihat `INSERT INTO` bagian. Memperbarui data tabel Gunung Es

Format yang didukung dan SerDes

Anda dapat menjalankan `INSERT` kueri pada tabel yang dibuat dari data dengan format berikut dan SerDes.

Format data	SerDe
Avro	org.apache.hadoop.hive.serde2.avro. AvroSerDe
Ion	com.amazon.ionhiveserde. IonHiveSerDe
JSON	org.apache.hive.hcatalog.data. JsonSerDe

Format data	SerDe
ORC	org.apache.hadoop.hive.ql.io.orc. OrcSerde
Parquet	org.apache.hadoop.hive.ql.io.parquet.serde. ParquetHiveSerDe
File teks	org.apache.hadoop.hive.serde2.lazy. LazySimpleSerDe **Note** CSV, TSV, dan file yang dibatasi khusus didukung.

Tabel bucketed tidak didukung

`INSERT INTO`tidak didukung pada tabel bucketed. Untuk informasi selengkapnya, lihat Partisi dan bucketing di Athena.

Kueri federasi tidak didukung

`INSERT INTO`tidak didukung untuk kueri federasi. Mencoba melakukannya dapat mengakibatkan pesan kesalahan Operasi ini saat ini tidak didukung untuk katalog eksternal. Untuk informasi tentang kueri federasi, lihat. Menggunakan Amazon Athena

Partisi

Pertimbangkan poin di bagian ini saat menggunakan partisi dengan `INSERT INTO` atau `CREATE TABLE AS SELECT` kueri.

Batas

Parameter`INSERT INTO`mendukung menulis maksimal 100 partisi ke tabel tujuan. Jika Anda menjalankan`SELECT`klausul di atas tabel dengan lebih dari 100 partisi, permintaan gagal kecuali`SELECT`permintaan terbatas untuk 100 partisi atau lebih sedikit.

Untuk informasi tentang cara mengatasi batasan ini, lihatMenggunakan CTAS dan INSERT INTO untuk bekerja di sekitar batas partisi 100.

Pemesanan kolom

`INSERT INTO`atau`CREATE TABLE AS SELECT`berharap kolom dipartisi menjadi kolom terakhir dalam daftar kolom diproyeksikan dalam`SELECT`.

Jika tabel sumber non-dipartisi, atau dipartisi pada kolom yang berbeda dibandingkan dengan tabel tujuan, kueri seperti`INSERT INTO` *`destination_table`* `SELECT * FROM` *`source_table`*mempertimbangkan nilai-nilai di kolom terakhir dari tabel sumber menjadi nilai untuk kolom partisi dalam tabel tujuan. Ingatlah hal ini saat mencoba membuat tabel dipartisi dari tabel non-dipartisi.

Sumber daya

Untuk informasi selengkapnya tentang penggunaan`INSERT INTO`dengan partisi, lihat sumber daya berikut.

- Untuk memasukkan data dipartisi ke dalam tabel dipartisi, lihatMenggunakan CTAS dan INSERT INTO untuk bekerja di sekitar batas partisi 100.
- Untuk memasukkan data unpartitioned ke dalam tabel dipartisi, lihatMenggunakan CTAS dan INSERT INTO untuk ETL dan analisis data.

File yang ditulis ke Amazon S3

Athena menulis file ke lokasi sumber data di Amazon S3 sebagai hasil dari`INSERT`Perintah. Setiap`INSERT`operasi menciptakan file baru, daripada menambahkan ke file yang ada. Lokasi file tergantung pada struktur tabel dan`SELECT`query, jika ada. Athena menghasilkan file manifest data untuk setiap`INSERT`kueri. Manifest melacak file yang kueri menulis. Ini disimpan ke lokasi hasil kueri Athena di Amazon S3. Untuk informasi selengkapnya, lihat Mengidentifikasi file keluaran kueri.

Hindari pembaruan yang sangat transaksional

Saat Anda menggunakan `INSERT INTO` untuk menambahkan baris ke tabel di Amazon S3, Athena tidak menulis ulang atau memodifikasi file yang ada. Sebaliknya, ia menulis baris sebagai satu atau lebih file baru. Karena tabel dengan banyak file kecil menghasilkan kinerja kueri yang lebih rendah, dan operasi tulis dan baca seperti `PutObject` dan `GetObject` menghasilkan biaya yang lebih tinggi dari Amazon S3, pertimbangkan opsi berikut saat menggunakan: `INSERT INTO`

- Jalankan `INSERT INTO` operasi lebih jarang pada batch baris yang lebih besar.
- Untuk volume konsumsi data yang besar, pertimbangkan untuk menggunakan layanan seperti Amazon Data Firehose.

- Hindari menggunakan `INSERT INTO` sama sekali. Sebagai gantinya, kumpulkan baris ke dalam file yang lebih besar dan unggah langsung ke Amazon S3 di mana mereka dapat ditanyakan oleh Athena.

Menemukan file yatim piatu

Jika `INSERT INTO` pernyataan `CTAS` atau gagal, data yatim piatu dapat ditinggalkan di lokasi data dan dapat dibaca dalam kueri berikutnya. Untuk menemukan file yatim piatu untuk pemeriksaan atau penghapusan, Anda dapat menggunakan file manifest data yang disediakan Athena untuk melacak daftar file yang akan ditulis. Untuk informasi selengkapnya, lihat Mengidentifikasi file keluaran kueri dan DataManifestLocation.

MASUKKAN KE... PILIH

Menentukan kueri untuk berjalan di satu tabel,`source_table`, yang menentukan baris untuk dimasukkan ke dalam tabel kedua,`destination_table`. Jika`SELECT`query menentukan kolom dalam`source_table`, kolom harus tepat sesuai dengan yang ada di`destination_table`.

Untuk informasi selengkapnya tentang kueri `SELECT` ini, lihat SELECT.

Sinopsis

```
INSERT INTO destination_table
SELECT select_query
FROM source_table_or_view
```

Contoh

Pilih semua baris di`vancouver_pageviews`tabel dan memasukkan mereka ke dalam`canada_pageviews`Tabel:

```
INSERT INTO canada_pageviews
SELECT *
FROM vancouver_pageviews;
```

Pilih hanya baris tersebut di`vancouver_pageviews`Tabel tempat`date`kolom memiliki nilai antara`2019-07-01`dan`2019-07-31`, dan kemudian masukkan ke dalam`canada_july_pageviews`:

```
INSERT INTO canada_july_pageviews
```

```
SELECT *
FROM vancouver_pageviews
WHERE date
    BETWEEN date '2019-07-01'
        AND '2019-07-31';
```

Pilih nilai dalamcitydanstatekolom dicities_worldtabel hanya dari baris dengan nilaiusadicountrykolom dan memasukkan mereka kecitydanstatekolom dicities_usaTabel:

```
INSERT INTO cities_usa (city,state)
SELECT city,state
FROM cities_world
    WHERE country='usa'
```

MASUKKAN KEDALAM... NILAI

Menyisipkan baris ke dalam tabel yang ada dengan menentukan kolom dan nilai-nilai. kolom yang ditentukan dan tipe data yang terkait harus tepat sesuai dengan kolom dan tipe data dalam tabel tujuan.

> **Important**
>
> Kami tidak menyarankan untuk memasukkan baris menggunakanVALUESkarena Athena menghasilkan file untuk setiapINSERToperasi. Ini dapat menyebabkan banyak file kecil yang akan dibuat dan menurunkan performa permintaan tabel. Untuk mengidentifikasi file yangINSERTquery menciptakan, memeriksa file manifest data. Untuk informasi selengkapnya, lihat Bekerja dengan hasil kueri, kueri terbaru, dan file keluaran.

Sinopsis

```
INSERT INTO destination_table [(col1,col2,...)]
VALUES (col1value,col2value,...)[,
       (col1value,col2value,...)][,
       ...]
```

Contoh

Dalam contoh berikut, tabel kota memiliki tiga kolom:id,city,state,state_motto. ParameteridKolom adalah tipeINTdan semua lajur lain adalah jenisVARCHAR.

Sisipkan satu baris ke dalam`cities`tabel, dengan semua nilai kolom yang ditentukan:

```
INSERT INTO cities
VALUES (1,'Lansing','MI','Si quaeris peninsulam amoenam circumspice')
```

Masukkan dua baris ke dalam`cities`Tabel:

```
INSERT INTO cities
VALUES (1,'Lansing','MI','Si quaeris peninsulam amoenam circumspice'),
       (3,'Boise','ID','Esto perpetua')
```

HAPUS

Menghapus baris dalam tabel Apache Iceberg.`DELETE`bersifat transaksional dan hanya didukung untuk tabel Apache Iceberg.

Sinopsis

Untuk menghapus baris dari tabel Iceberg, gunakan sintaks berikut.

```
DELETE FROM [db_name.]table_name [WHERE predicate]
```

Untuk informasi lebih lanjut dan contoh, lihat`DELETE`bagian dariMemperbarui data tabel Gunung Es.

PERBARUI

Update baris dalam tabel Apache Iceberg.`UPDATE`bersifat transaksional dan hanya didukung untuk tabel Apache Iceberg.

Sinopsis

Untuk memperbarui baris dalam tabel Iceberg, gunakan sintaks berikut.

```
UPDATE [db_name.]table_name SET xx=yy[,...] [WHERE predicate]
```

Untuk informasi lebih lanjut dan contoh, lihat`UPDATE`bagian dariMemperbarui data tabel Gunung Es.

BERGABUNG MENJADI

Secara kondisional memperbarui, menghapus, atau menyisipkan baris ke dalam tabel Apache Iceberg. Sebuah pernyataan tunggal dapat menggabungkan tindakan pembaruan, menghapus, dan menyisipkan.

> **Note**
>
> `MERGE INTO` bersifat transaksional dan hanya didukung untuk tabel Apache Iceberg di mesin Athena versi 3.

Sinopsis

Untuk memperbarui, menghapus, atau menyisipkan baris secara kondisional dari tabel Iceberg, gunakan sintaks berikut.

```
MERGE INTO target_table [ [ AS ]  target_alias ]
USING { source_table | query } [ [ AS ] source_alias ]
ON search_condition
when_clause [...]
```

When_clause adalah salah satu dari berikut ini:

```
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` mendukung sejumlah `WHEN` klausa yang sewenang-wenang dengan kondisi yang berbeda. `MATCHED` Klausa kondisi mengeksekusi `DELETE`, `UPDATE` atau `INSERT` operasi dalam `WHEN` klausa pertama yang dipilih oleh `MATCHED` status dan kondisi kecocokan.

Untuk setiap baris sumber, `WHEN` klausa diproses secara berurutan. Hanya `WHEN` klausa pencocokan pertama yang dieksekusi. Klausul selanjutnya diabaikan. Kesalahan pengguna muncul ketika satu baris tabel target cocok dengan lebih dari satu baris sumber.

Jika baris sumber tidak cocok dengan `WHEN` klausa apa pun dan tidak ada `WHEN NOT MATCHED` klausa, baris sumber diabaikan.

Dalam WHEN klausa yang memiliki UPDATE operasi, ekspresi nilai kolom dapat merujuk ke bidang apa pun dari target atau sumber. Dalam NOT MATCHED kasus ini, INSERT ekspresi dapat merujuk ke bidang sumber apa pun.

Contoh

Contoh berikut menggabungkan baris dari tabel kedua ke tabel pertama jika baris tidak ada di tabel pertama. Perhatikan bahwa kolom yang tercantum dalam VALUES klausa harus diawali dengan alias tabel sumber. Kolom target yang tercantum dalam INSERT klausa tidak boleh diawali.

```
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)
```

Untuk MERGE INTO contoh lainnya, lihatMemperbarui data tabel Gunung Es.

MENGOPTIMALKAN

Mengoptimalkan baris dalam tabel Apache Iceberg dengan menulis ulang file data ke dalam tata letak yang lebih dioptimalkan berdasarkan ukuran dan jumlah file penghapusan terkait.

> **Note**
>
> OPTIMIZEbersifat transaksional dan hanya didukung untuk tabel Apache Iceberg.

Sintaks

Ringkasan sintaks berikut menunjukkan cara mengoptimalkan tata letak data untuk tabel Iceberg.

```
OPTIMIZE [db_name.]table_name REWRITE DATA USING BIN_PACK
  [WHERE predicate]
```

> **Note**
>
> Hanya kolom partisi yang diizinkan dalam *predikat WHERE* klausa. Menentukan kolom non-partisi akan menyebabkan query gagal.

Tindakan pemadatan dibebankan oleh jumlah data yang dipindai selama proses penulisan ulang. `REWRITE DATA`Tindakan ini menggunakan predikat untuk memilih file yang berisi baris yang cocok. Jika ada baris dalam file yang cocok dengan predikat, file dipilih untuk pengoptimalan. Dengan demikian, untuk mengontrol jumlah file yang terpengaruh oleh operasi pemadatan, Anda dapat menentukan `WHERE` klausa.

Mengkonfigurasi properti pemadatan

Untuk mengontrol ukuran file yang akan dipilih untuk pemadatan dan ukuran file yang dihasilkan setelah pemadatan, Anda dapat menggunakan parameter properti tabel. Anda dapat menggunakan MENGUBAH PROPERTI SET TABEL perintah untuk mengkonfigurasi properti tabel terkait.

Sumber daya tambahan

Mengoptimalkan tabel Iceberg

VAKUM

`VACUUM`Pernyataan tersebut melakukan pemeliharaan tabel pada tabel Apache Iceberg dengan menghapus file data yang tidak lagi diperlukan.

> **Note**
>
> `VACUUM`bersifat transaksional dan hanya didukung untuk tabel Apache Iceberg di mesin Athena versi 3.

Menjalankan `VACUUM` pernyataan pada tabel Iceberg disarankan untuk menghapus file data yang tidak lagi relevan dan untuk mengurangi ukuran metadata dan konsumsi penyimpanan. Perhatikan bahwa, karena `VACUUM` pernyataan tersebut membuat panggilan API ke Amazon S3, biaya berlaku untuk permintaan terkait ke Amazon S3.

> **Warning**
>
> Jika Anda menjalankan operasi kedaluwarsa snapshot, Anda tidak dapat lagi melakukan perjalanan waktu ke snapshot yang kedaluwarsa.

Sinopsis

Untuk menghapus file data yang tidak lagi diperlukan untuk tabel Iceberg, gunakan sintaks berikut.

```
VACUUM [database_name.]target_table
```

Untuk berjalan VACUUM di atas meja dengan nama yang dimulai dengan garis bawah (misalnya,_mytable), lampirkan nama tabel di backticks, seperti pada contoh berikut. Jika Anda mengawali nama tabel dengan nama database, jangan lampirkan nama database di backticks. Perhatikan bahwa tanda kutip ganda tidak akan berfungsi sebagai pengganti backticks.

Perilaku ini khusus untukVACUUM. INSERT INTOPernyataan CREATE dan tidak memerlukan backticks untuk nama tabel yang dimulai dengan garis bawah.

```
VACUUM `_mytable`
VACUUM my_database.`_mytable`
```

Perhatikan juga bahwa VACUUM mengharapkan data Iceberg berada di folder Amazon S3 daripada bucket Amazon S3. Misalnya, jika data Iceberg Anda dis3://DOC-EXAMPLE-BUCKET/bukans3://DOC-EXAMPLE-BUCKET/myicebergfolder/, VACUUM pernyataan gagal dengan pesan kesalahan GENERIC_INTERNAL_ERROR: Jalur hilang di lokasi sistem file:. s3://DOC-EXAMPLE-BUCKET

Operasi dilakukan

VACUUMmelakukan operasi berikut:

- Menghapus snapshot yang lebih tua dari jumlah waktu yang ditentukan oleh properti vacuum_max_snapshot_age_seconds tabel. Secara default, properti ini diatur ke 432000 detik (5 hari).
- Menghapus snapshot yang tidak berada dalam periode yang akan dipertahankan yang melebihi jumlah yang ditentukan oleh properti vacuum_min_snapshots_to_keep tabel. Default-nya adalah 1.

  Anda dapat menentukan properti tabel ini dalam CREATE TABLE pernyataan Anda. Setelah tabel dibuat, Anda dapat menggunakan MENGUBAH PROPERTI SET TABEL pernyataan untuk memperbaruinya.
- Menghapus semua metadata dan file data yang tidak dapat dijangkau sebagai akibat dari penghapusan snapshot. Anda dapat mengonfigurasi jumlah file metadata lama yang akan dipertahankan dengan mengatur properti tabel. vacuum_max_metadata_files_to_keep Nilai default-nya adalah 100.

- Menghapus file yatim piatu yang lebih tua dari waktu yang ditentukan dalam properti `vacuum_max_snapshot_age_seconds` tabel. File yatim adalah file dalam direktori data tabel yang bukan bagian dari status tabel.

Untuk informasi lebih lanjut tentang membuat dan mengelola tabel Apache Iceberg di Athena, lihat dan. Membuat tabel Iceberg Mengelola tabel Iceberg

Menggunakan JELASKAN dan JELASKAN ANALISIS di Athena

ParameterEXPLAINpernyataan menunjukkan rencana eksekusi logis atau didistribusikan dari pernyataan SQL tertentu, atau memvalidasi pernyataan SQL. Anda dapat menampilkan hasil dalam format teks atau dalam format data untuk rendering ke grafik.

> **Note**
>
> Anda dapat melihat representasi grafis dari rencana logis dan terdistribusi untuk kueri Anda di konsol Athena tanpa menggunakan sintaks. `EXPLAIN` Untuk informasi selengkapnya, lihat Melihat rencana eksekusi untuk kueri SQL.

`EXPLAIN ANALYZE`Pernyataan tersebut menunjukkan rencana eksekusi terdistribusi dari pernyataan SQL tertentu dan biaya komputasi setiap operasi dalam kueri SQL. Anda dapat menampilkan hasil dalam format teks atau JSON.

Pertimbangan dan batasan

`EXPLAIN ANALYZE`Pernyataan `EXPLAIN` dan di Athena memiliki batasan sebagai berikut.

- KarenaEXPLAINpertanyaan tidak memindai data apapun, Athena tidak mengenakan biaya untuk mereka. Namun, karena `EXPLAIN` kueri melakukan panggilan AWS Glue untuk mengambil metadata tabel, Anda mungkin dikenakan biaya dari Glue jika panggilan melebihi batas tingkat gratis untuk lem.
- Karena `EXPLAIN ANALYZE` kueri dijalankan, mereka memindai data, dan Athena mengenakan biaya untuk jumlah data yang dipindai.
- Informasi pemfilteran baris atau sel yang ditentukan dalam Lake Formation dan informasi statistik kueri tidak ditampilkan dalam output dan. `EXPLAIN` `EXPLAIN ANALYZE`

JELASKAN sintaks

```
EXPLAIN [ ( option [, ...]) ] statement
```

opsi dapat berupa satu dari yang berikut:

```
FORMAT { TEXT | GRAPHVIZ | JSON }
TYPE { LOGICAL | DISTRIBUTED | VALIDATE | IO }
```

Jika `FORMAT` opsi tidak ditentukan, output default ke format. `TEXT` Tipe `IO` menyediakan informasi tentang tabel dan skema yang kueri baca. `IO` hanya didukung di mesin Athena versi 2 dan dapat dikembalikan hanya dalam format JSON.

JELASKAN ANALYSIS sintaks

Selain output yang disertakan`EXPLAIN`, `EXPLAIN ANALYZE` output juga mencakup statistik runtime untuk kueri yang ditentukan seperti penggunaan CPU, jumlah baris input, dan jumlah baris output.

```
EXPLAIN ANALYZE [ ( option [, ...]) ] statement
```

opsi dapat berupa satu dari yang berikut:

```
FORMAT { TEXT | JSON }
```

Jika `FORMAT` opsi tidak ditentukan, output default ke format. `TEXT` Karena semua kueri untuk `EXPLAIN ANALYZE` adalah`DISTRIBUTED`, `TYPE` opsi tidak tersedia untuk`EXPLAIN ANALYZE`.

pernyataan dapat menjadi salah satu dari berikut:

```
SELECT
CREATE TABLE AS SELECT
INSERT
UNLOAD
```

JELASKAN contoh

Contoh berikut untuk `EXPLAIN` kemajuan dari yang lebih mudah ke yang lebih kompleks.

JELASKAN contoh 1: Gunakan pernyataan EXPLOW untuk menampilkan rencana kueri dalam format teks

Dalam contoh berikut, `EXPLAIN` menunjukkan rencana eksekusi untuk `SELECT` kueri pada log Elastic Load Balancing. Format default ke output teks.

```
EXPLAIN
SELECT
   request_timestamp,
   elb_name,
   request_ip
FROM sampledb.elb_logs;
```

Hasil

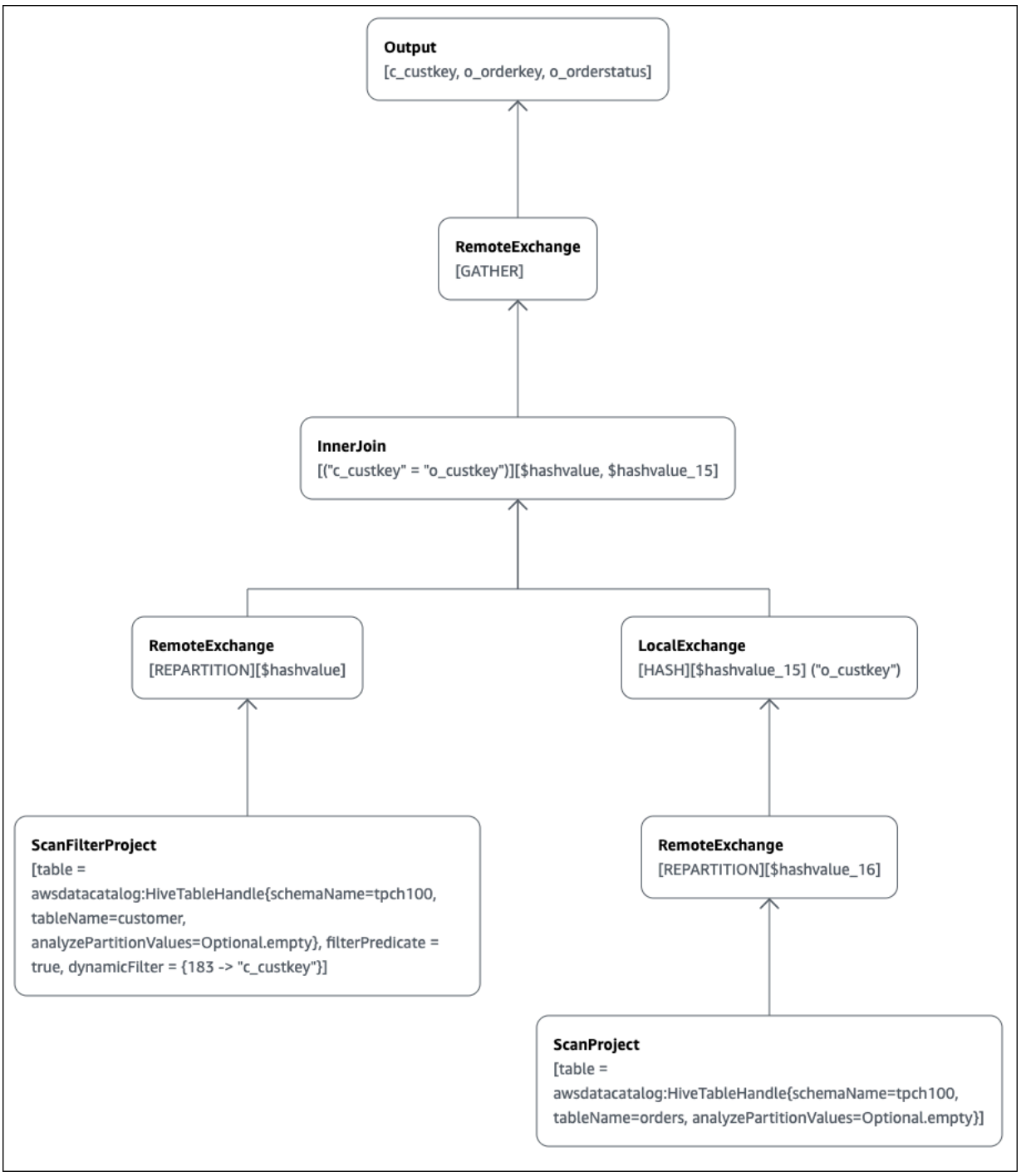
```
- Output[request_timestamp, elb_name, request_ip] => [[request_timestamp, elb_name,
 request_ip]]
    - RemoteExchange[GATHER] => [[request_timestamp, elb_name, request_ip]]
        - TableScan[awsdatacatalog:HiveTableHandle{schemaName=sampledb,
 tableName=elb_logs,
analyzePartitionValues=Optional.empty}] => [[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
```

JELASKAN contoh 2: Buat grafik rencana kueri

Anda dapat menggunakan konsol Athena untuk membuat grafik rencana kueri untuk Anda. Masukkan **`SELECT`** pernyataan seperti berikut ini ke dalam editor kueri Athena, lalu pilih JELASKAN.

```
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
```

Halaman Jelaskan editor kueri Athena terbuka dan menunjukkan kepada Anda rencana terdistribusi dan rencana logis untuk kueri tersebut. Grafik berikut menunjukkan rencana logis untuk contoh.



Important

Saat ini, beberapa filter partisi mungkin tidak terlihat di grafik pohon operator bersarang meskipun Athena menerapkannya ke kueri Anda. Untuk memverifikasi efek filter tersebut, jalankan `EXPLAIN` atau `EXPLAIN ANALYZE` pada kueri Anda dan lihat hasilnya.

Untuk informasi selengkapnya tentang penggunaan fitur grafik paket kueri di konsol Athena, lihat. Melihat rencana eksekusi untuk kueri SQL

JELASKAN contoh 3: Gunakan pernyataan EXPLOW untuk memverifikasi pemangkasan partisi

Saat Anda menggunakan predikat penyaringan pada bukti kunci dipartisi untuk kueri tabel dipartisi, mesin permintaan berlaku predikat untuk bukti kunci dipartisi untuk mengurangi jumlah data yang dibaca.

Contoh berikut menggunakanEXPLAINquery untuk memverifikasi partisi pemangkasan untukSELECTquery pada tabel dipartisi. Pertama,CREATE TABLEpernyataan menciptakantpch100.orders_partitionedtabel. Tabel dipartisi pada kolomo_orderdate.

```
CREATE TABLE `tpch100.orders_partitioned`(
  `o_orderkey` int,
  `o_custkey` int,
  `o_orderstatus` string,
  `o_totalprice` double,
  `o_orderpriority` string,
  `o_clerk` string,
  `o_shippriority` int,
  `o_comment` string)
PARTITIONED BY (
  `o_orderdate` string)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/<your_directory_path>/'
```

Parametertpch100.orders_partitionedtabel memiliki beberapa partisi padao_orderdate, seperti yang ditunjukkan olehSHOW PARTITIONSPerintah.

```
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
```

BerikutEXPLAINquery memverifikasi partisi pemangkasan pada ditentukanSELECT.

```
EXPLAIN
SELECT
   o_orderkey,
   o_custkey,
   o_orderdate
FROM tpch100.orders_partitioned
WHERE o_orderdate = '1995'
```

Hasil

```
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
```

Teks tebal dalam hasil menunjukkan bahwa predikato_orderdate = '1995'diterapkan padaPARTITION_KEY.

JELASKAN contoh 4: Gunakan kueri EXPLOW untuk memeriksa jenis join order dan join

BerikutEXPLAINmemeriksa kueriSELECTperintah bergabung pernyataan dan bergabung jenis. Gunakan kueri seperti ini untuk memeriksa penggunaan memori kueri sehingga Anda dapat mengurangi kemungkinan mendapatkanEXCEEDED_LOCAL_MEMORY_LIMITkesalahan.

```
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
```

Hasil

```
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
```

Contoh kueri dioptimalkan menjadi silang bergabung untuk performa yang lebih baik. Hasil menunjukkan bahwa`tpch100.orders`akan didistribusikan sebagai`BROADCAST`distribusi jenis. Ini berarti bahwa`tpch100.orders`tabel akan didistribusikan ke semua simpul yang melakukan operasi bergabung. Parameter`BROADCAST`jenis distribusi akan mengharuskan semua hasil disaring dari`tpch100.orders`tabel masuk ke dalam memori dari setiap simpul yang melakukan bergabung operasi.

Namun,`tpch100.customer`tabel lebih kecil dari`tpch100.orders`. Karena`tpch100.customer`membutuhkan lebih sedikit memori, Anda dapat menulis ulang kueri ke`BROADCAST tpch100.customer`Sebagai gantinya`tpch100.orders`. Ini mengurangi kemungkinan kueri menerima`EXCEEDED_LOCAL_MEMORY_LIMIT`kesalahan. Strategi ini mengasumsikan poin-poin berikut:

- Parameter`tpch100.customer.c_custkey`unik di`tpch100.customer`tabel.
- Ada hubungan one-to-many pemetaan antara `tpch100.customer` dan`tpch100.orders`.

Contoh berikut menunjukkan kueri ditulis ulang.

```
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
```

JELASKAN contoh 5: Gunakan kueri EXPLOW untuk menghapus predikat yang tidak berpengaruh

Anda dapat menggunakan`EXPLAIN`query untuk memeriksa efektivitas penyaringan predikat. Anda dapat menggunakan hasil untuk menghapus predikat yang tidak berpengaruh, seperti dalam contoh berikut.

```
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
```

Hasil

```
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
```

Parameter`filterPredicate`dalam hasil menunjukkan bahwa optimizer menggabungkan tiga predikat asli menjadi dua predikat dan mengubah urutan aplikasi mereka.

```
filterPredicate = (("c_custkey" = 1500) AND ("c_custkey" = CAST(("random"() * 1E3) AS
 int)))
```

Karena hasil menunjukkan bahwa predikat`AND c.c_custkey BETWEEN 1000 AND 2000`tidak berpengaruh, Anda dapat menghapus predikat ini tanpa mengubah hasil kueri.

Untuk informasi tentang istilah yang digunakan dalam hasil`EXPLAIN`kueri, lihat[Memahami Athena MENJELASKAN hasil pernyataan](#).

JELASKAN CONTOH ANALISIS

Contoh berikut menunjukkan contoh `EXPLAIN ANALYZE` query dan output.

JELASKAN ANALISIS contoh 1: Gunakan EXPLY ANALYZE untuk menampilkan rencana kueri dan biaya komputasi dalam format teks

Dalam contoh berikut, `EXPLAIN ANALYZE` menunjukkan rencana eksekusi dan biaya komputasi untuk `SELECT` kueri pada CloudFront log. Format default ke output teks.

```
EXPLAIN ANALYZE SELECT FROM cloudfront_logs LIMIT 10
```

Hasil

```
 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
```

JELASKAN ANALISIS contoh 2: Gunakan EXPLOW ANALYZE untuk menampilkan rencana kueri dalam format JSON

Contoh berikut menunjukkan rencana eksekusi dan biaya komputasi untuk SELECT kueri pada CloudFront log. Contoh menentukan JSON sebagai format output.

```
EXPLAIN ANALYZE (FORMAT JSON) SELECT * FROM cloudfront_logs LIMIT 10
```

Hasil

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

Sumber daya tambahan

Untuk informasi tambahan, lihat sumber daya berikut.

- Memahami Athena MENJELASKAN hasil pernyataan
- Melihat rencana eksekusi untuk kueri SQL
- Melihat statistik dan detail eksekusi untuk kueri yang diselesaikan
- Dokumentasi Trino `EXPLAIN`
- Dokumentasi Trino `EXPLAIN ANALYZE`

- Optimalkan Kinerja Kueri Federasi menggunakan JELASKAN dan JELASKAN ANALISIS di Amazon Athena di Blog AWS Big Data.

Memahami Athena MENJELASKAN hasil pernyataan

Topik ini memberikan panduan singkat untuk istilah operasional yang digunakan di Athena`EXPLAIN`.

JELASKAN jenis keluaran pernyataan

`EXPLAIN`output pernyataan dapat menjadi salah satu dari dua jenis:

- Rencana logis - Menunjukkan rencana logis yang digunakan mesin SQL untuk mengeksekusi pernyataan. Sintaks untuk opsi ini`EXPLAIN`atau`EXPLAIN (TYPE LOGICAL)`.
- Rencana terdistribusi - Menunjukkan rencana eksekusi di lingkungan terdistribusi. Output menunjukkan fragmen, yang memproses tahap. Setiap fragmen rencana diproses oleh satu atau lebih simpul. Data dapat dipertukarkan antara simpul yang memproses fragmen. Sintaks untuk opsi ini`EXPLAIN (TYPE DISTRIBUTED)`.

  Dalam output untuk rencana terdistribusi, fragmen (tahap pemrosesan) ditunjukkan oleh`Fragment` *`nomor`*[*`fragment_type`*], tempat*`nomor`*adalah bilangan bulat berbasis-nol dan*`fragment_type`*menentukan bagaimana fragmen dijalankan oleh simpul. Jenis fragmen, yang memberikan wawasan tata letak Data Exchange, dijelaskan dalam tabel berikut.

  Jenis fragmen rencana terdistribusi

Jenis fragmen	Deskripsi
`SINGLE`	Fragmen dieksekusi pada satu simpul.
`HASH`	Fragmen dieksekusi pada sejumlah simpul tetap. Input data didistribusikan menggunakan fungsi hash.
`ROUND_ROBIN`	Fragmen dieksekusi pada sejumlah simpul tetap. Input data didistribusikan secara round-robin.
`BROADCAST`	Fragmen dieksekusi pada sejumlah simpul tetap. Input data disiarkan ke semua simpul.
`SOURCE`	Fragmen dijalankan pada simpul tempat perpecahan input diakses.

.exchange

Istilah terkait pertukaran menggambarkan bagaimana data dipertukarkan antara simpul pekerja. Transfer dapat berupa lokal atau remote.

LocalExchange [***exchange_type***]

Transfer data secara lokal dalam simpul pekerja untuk berbagai tahap kueri. Nilai untuk*exchange_type*dapat menjadi salah satu jenis pertukaran logis atau terdistribusi seperti yang dijelaskan nanti di bagian ini.

RemoteExchange [***exchange_type***]

Transfer data antara simpul pekerja untuk berbagai tahap kueri. Nilai untuk*exchange_type*dapat menjadi salah satu jenis pertukaran logis atau terdistribusi seperti yang dijelaskan nanti di bagian ini.

Jenis Pertukaran Logis

Jenis pertukaran berikut menggambarkan tindakan yang diambil selama fase pertukaran dari rencana logis.

- **GATHER**— Sebuah simpul pekerja tunggal mengumpulkan output dari semua simpul pekerja lainnya. Misalnya, tahap terakhir dari kueri pilih mengumpulkan hasil dari semua simpul dan menulis hasilnya ke Amazon S3.
- **REPARTITION**— Mengirim data baris ke pekerja tertentu berdasarkan skema partisi yang diperlukan untuk diterapkan ke operator berikutnya.
- **REPLICATE**— Salinan data baris untuk semua pekerja.

Jenis Pertukaran Terdistribusi

Jenis pertukaran berikut menunjukkan tata letak data saat mereka dipertukarkan antara simpul dalam rencana terdistribusi.

- **HASH**— Pertukaran mendistribusikan data ke beberapa tujuan menggunakan fungsi hash.
- **SINGLE**— Pertukaran mendistribusikan data ke satu tujuan.

Pemindaian

Istilah berikut menjelaskan bagaimana data dipindai selama kueri.

TableScan

Memindai data sumber tabel dari Amazon S3 atau konektor Apache Hive dan berlaku pemangkasan partisi yang dihasilkan dari predikat filter.

ScanFilter

Memindai data sumber tabel dari Amazon S3 atau konektor Hive dan berlaku pemangkasan partisi yang dihasilkan dari predikat filter dan dari predikat filter tambahan tidak diterapkan melalui pemangkasan partisi.

ScanFilterProject

Pertama, memindai data sumber tabel dari Amazon S3 atau konektor Hive dan berlaku pemangkasan partisi yang dihasilkan dari predikat filter dan dari predikat filter tambahan tidak diterapkan melalui pemangkasan partisi. Kemudian, memodifikasi tata letak memori data output ke proyeksi baru untuk meningkatkan performa tahap selanjutnya.

Join

Bergabung data antara dua tabel. Bergabung dapat dikategorikan berdasarkan bergabung jenis dan jenis distribusi.

Bergabunglah dengan tipe

Bergabung jenis menentukan cara tempat bergabung operasi terjadi.

CrossJoin— Menghasilkan produk Cartesian dari dua tabel yang digabungkan.

InnerJoin— Memilih catatan yang memiliki nilai yang cocok di kedua tabel.

LeftJoin— Memilih semua catatan dari tabel kiri dan catatan yang cocok dari tabel kanan. Jika tidak ada pertandingan terjadi, hasil di sisi kanan adalah NULL.

RightJoin— Memilih semua catatan dari tabel kanan, dan catatan yang cocok dari tabel kiri. Jika tidak ada pertandingan terjadi, hasil di sisi kiri adalah NULL.

FullJoin— Memilih semua catatan di mana ada kecocokan di catatan tabel kiri atau kanan. Tabel bergabung berisi semua catatan dari kedua tabel dan mengisi NULLs untuk pertandingan hilang di kedua sisi.

> **Note**
>
> Untuk alasan performa, mesin permintaan dapat menulis ulang kueri bergabung ke jenis bergabung yang berbeda untuk menghasilkan hasil yang sama. Sebagai contoh, batin bergabung kueri dengan predikat pada satu tabel dapat ditulis ulang menjadi`CrossJoin`. Ini mendorong predikat ke fase pemindaian tabel sehingga data yang lebih sedikit dipindai.

Bergabunglah dengan jenis distribusi

jenis distribusi menentukan bagaimana data dipertukarkan antara simpul pekerja saat bergabung operasi dilakukan.

Dipartisi— Kedua tabel kiri dan kanan hash-dipartisi di semua simpul pekerja. distribusi dipartisi mengkonsumsi lebih sedikit memori di setiap simpul. distribusi dipartisi dapat jauh lebih lambat dari direplikasi bergabung. Dipartisi bergabung cocok saat Anda bergabung dua tabel besar.

Replikasi— Satu tabel hash-dipartisi di semua simpul pekerja dan tabel lainnya direplikasi ke semua simpul pekerja untuk melakukan operasi bergabung. Distribusi direplikasi dapat jauh lebih cepat daripada dipartisi bergabung, tetapi mengkonsumsi lebih banyak memori di setiap simpul pekerja. Jika tabel yang direplikasi terlalu besar, node pekerja dapat mengalami out-of-memory kesalahan. Replikasi bergabung cocok saat salah satu tabel bergabung kecil.

MEMPERSIAPKAN

Membuat pernyataan SQL dengan nama `statement_name` yang akan dijalankan di lain waktu. Pernyataan tersebut dapat mencakup parameter yang diwakili oleh tanda tanya. Untuk memberikan nilai untuk parameter dan menjalankan pernyataan yang disiapkan, gunakanEXECUTE.

Sinopsis

```
PREPARE statement_name FROM statement
```

Tabel berikut menjelaskan parameter.

Parameter	Deskripsi
`statement_name`	Nama pernyataan yang harus dipersiapkan. Nama dalam buket harus unik.

Parameter	Deskripsi
`statement`	Kueri `SELECT`, `CTAS`, atau `INSERT INTO`.

> **Note**
>
> Jumlah maksimum pernyataan yang disiapkan dalam kelompok kerja adalah 1000.

Contoh

Contoh berikut menyiapkan query pilih tanpa parameter.

```
PREPARE my_select1 FROM
SELECT * FROM nation
```

Contoh berikut menyiapkan kueri pilih yang mencakup parameter. Nilai untuk `productid` dan `quantity` akan diberikan oleh `USING` klausa `EXECUTE` pernyataan:

```
PREPARE my_select2 FROM
SELECT order FROM orders WHERE productid = ? and quantity < ?
```

Contoh berikut menyiapkan query insert.

```
PREPARE my_insert FROM
INSERT INTO cities_usa (city, state)
SELECT city, state
FROM cities_world
WHERE country = ?
```

Sumber daya tambahan

Menanyakan dengan pernyataan yang disiapkan

EXECUTE

DEALLOCATE PREPARE

INSERT INTO

EXECUTE

Menjalankan pernyataan yang disiapkan dengan nama`statement_name`. Nilai parameter untuk tanda tanya dalam pernyataan yang disiapkan didefinisikan dalam `USING` klausa dalam daftar dipisahkan koma. Untuk membuat pernyataan yang disiapkan, gunakanMEMPERSIAPKAN.

Sinopsis

```
EXECUTE statement_name [ USING parameter1[, parameter2, ... ] ]
```

Contoh

Contoh berikut mempersiapkan dan mengeksekusi query tanpa parameter.

```
PREPARE my_select1 FROM
SELECT name FROM nation
EXECUTE my_select1
```

Contoh berikut mempersiapkan dan mengeksekusi query dengan parameter tunggal.

```
PREPARE my_select2 FROM
SELECT * FROM "my_database"."my_table" WHERE year = ?
EXECUTE my_select2 USING 2012
```

Ini setara dengan:

```
SELECT * FROM "my_database"."my_table" WHERE year = 2012
```

Contoh berikut mempersiapkan dan mengeksekusi query dengan dua parameter.

```
PREPARE my_select3 FROM
SELECT order FROM orders WHERE productid = ? and quantity < ?
EXECUTE my_select3 USING 346078, 12
```

Sumber daya tambahan

Menanyakan dengan pernyataan yang disiapkan

MEMPERSIAPKAN

INSERT INTO

DEALLOCATE PREPARE

Menghapus pernyataan yang disiapkan dengan nama yang ditentukan dari pernyataan yang disiapkan dalam workgroup saat ini.

Sinopsis

```
DEALLOCATE PREPARE statement_name
```

Contoh

Contoh berikut menghapus pernyataan yang disiapkan `my_select1` dari grup kerja saat ini.

```
DEALLOCATE PREPARE my_select1
```

Sumber daya tambahan

Menanyakan dengan pernyataan yang disiapkan

MEMPERSIAPKAN

MEMBONGKAR

Menulis hasil kueri dari `SELECT` pernyataan ke format data yang ditentukan. Format yang didukung untuk `UNLOAD` termasuk Apache Parquet, ORC, Apache Avro, dan JSON. CSV adalah satu-satunya format output yang didukung oleh perintah `SELECT` Athena, tetapi Anda dapat menggunakan `UNLOAD` perintah, yang mendukung berbagai format output, untuk melampirkan kueri `SELECT` Anda dan menulis ulang outputnya ke salah satu format yang mendukung. `UNLOAD`

Meskipun Anda dapat menggunakan pernyataan CTAS untuk mengeluarkan data dalam format selain CSV, pernyataan tersebut juga memerlukan pembuatan tabel di Athena. `UNLOAD`Pernyataan ini berguna ketika Anda ingin menampilkan hasil `SELECT` kueri dalam format non-CSV tetapi tidak memerlukan tabel terkait. Misalnya, aplikasi hilir mungkin memerlukan hasil `SELECT` kueri dalam format JSON, dan Parquet atau ORC mungkin memberikan keunggulan kinerja dibandingkan CSV jika Anda bermaksud menggunakan hasil kueri untuk analisis tambahan. `SELECT`

Pertimbangan dan batasan

Saat Anda menggunakan `UNLOAD` pernyataan di Athena, ingatlah poin-poin berikut:

- Tidak ada urutan global file — UNLOAD hasilnya ditulis ke beberapa file secara paralel. Jika SELECT kueri dalam UNLOAD pernyataan menentukan urutan pengurutan, isi setiap file dalam urutan diurutkan, tetapi file tidak diurutkan relatif satu sama lain.
- Data yatim piatu tidak dihapus — Jika terjadi kegagalan, Athena tidak berusaha menghapus data yatim piatu. Perilaku ini sama dengan CTAS dan INSERT INTO pernyataan.
- Partisi maksimum — Jumlah maksimum partisi yang dapat digunakan UNLOAD adalah 100.
- File metadata dan manifes — Athena menghasilkan file metadata dan file manifes data untuk setiap kueri. UNLOAD Manifest melacak file yang kueri menulis. Kedua file disimpan ke lokasi hasil kueri Athena Anda di Amazon S3. Untuk informasi selengkapnya, lihat Mengidentifikasi file keluaran kueri.
- Enkripsi — file UNLOAD output dienkripsi sesuai dengan konfigurasi enkripsi yang digunakan untuk Amazon S3. Untuk mengatur konfigurasi enkripsi untuk mengenkripsi UNLOAD hasil Anda, Anda dapat menggunakan EncryptionConfiguration API.
- Pernyataan yang disiapkan - UNLOAD dapat digunakan dengan pernyataan yang disiapkan. Untuk informasi tentang pernyataan yang disiapkan di Athena, lihat. Menggunakan kueri berparameter
- Kuota layanan - UNLOAD menggunakan kuota kueri DML. Untuk informasi kuota, lihatService Quotas.
- Pemilik bucket yang diharapkan — Pengaturan pemilik bucket yang diharapkan tidak berlaku untuk lokasi Amazon S3 tujuan yang ditentukan dalam kueri. UNLOAD Setelan pemilik bucket yang diharapkan hanya berlaku untuk lokasi keluaran Amazon S3 yang Anda tentukan untuk hasil kueri Athena. Untuk informasi selengkapnya, lihat Menentukan lokasi hasil kueri menggunakan konsol Athena.

Sintaks

UNLOADPernyataan ini menggunakan sintaks berikut.

```
UNLOAD (SELECT col_name[, ...] FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/my_folder/'
WITH ( property_name = 'expression' [, ...] )
```

Kecuali saat menulis ke partisi, TO tujuan harus menentukan lokasi di Amazon S3 yang tidak memiliki data. Sebelum UNLOAD kueri menulis ke lokasi yang ditentukan, kueri memverifikasi bahwa lokasi bucket kosong. Karena UNLOAD tidak menulis data ke lokasi yang ditentukan jika lokasi sudah memiliki data di dalamnya, UNLOAD tidak menimpa data yang ada. Untuk menggunakan kembali lokasi bucket sebagai tujuanUNLOAD, hapus data di lokasi bucket, lalu jalankan kueri lagi.

Perhatikan bahwa ketika UNLOAD menulis ke partisi, perilaku ini berbeda. Jika Anda menjalankan UNLOAD kueri yang sama beberapa kali yang memiliki SELECT pernyataan yang sama, TO lokasi yang sama, dan partisi yang sama, setiap UNLOAD kueri akan membongkar data ke Amazon S3 di lokasi dan partisi yang ditentukan.

Parameter

Nilai yang mungkin untuk *property_name* adalah sebagai berikut.

format = '*file_format*'

Wajib. Menentukan format file output. Nilai yang mungkin untuk *file_format* adalahORC,,, PARQUETAVRO, JSON atau. TEXTFILE

kompresi = '*compression_format*'

Tidak wajib. Opsi ini khusus untuk format ORC dan Parket. Untuk ORC, defaultnya adalahzlib, dan untuk Parket, defaultnya adalah. gzip Untuk informasi tentang format kompresi yang didukung, lihat Dukungan kompresi Athena.

> **Note**
>
> Opsi ini tidak berlaku untuk AVRO format. Athena menggunakan gzip untuk JSON dan TEXTFILE format.

compression_level = compression_level

Tidak wajib. Tingkat kompresi yang digunakan untuk kompresi ZSTD. Properti ini hanya berlaku untuk kompresi ZSTD. Untuk informasi selengkapnya, lihat Menggunakan tingkat kompresi ZSTD di Athena.

field_delimiter = 'pembatas'

Tidak wajib. Menentukan pembatas bidang karakter tunggal untuk file dalam CSV, TSV, dan format teks lainnya. Contoh berikut menentukan pembatas koma.

```
WITH (field_delimiter = ',')
```

Saat ini, pembatas bidang multikarakter tidak didukung. Jika Anda tidak menentukan pembatas bidang, karakter oktal \001 (^A) digunakan.

partitioned_by = ARRAY [*col_name* [,...]]

Tidak wajib. Daftar array kolom dimana output dipartisi.

> **Note**
>
> Dalam SELECT pernyataan Anda, pastikan bahwa nama-nama kolom yang dipartisi terakhir dalam daftar kolom Anda.

Contoh

Contoh berikut menulis output dari SELECT query ke lokasi Amazon S3 `s3://DOC-EXAMPLE-BUCKET/unload_test_1/` menggunakan format JSON.

```
UNLOAD (SELECT * FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/unload_test_1/'
WITH (format = 'JSON')
```

Contoh berikut menulis output dari SELECT query dalam format Parquet menggunakan kompresi Snappy.

```
UNLOAD (SELECT * FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/'
WITH (format = 'PARQUET',compression = 'SNAPPY')
```

Contoh berikut menulis empat kolom dalam format teks, dengan output dipartisi oleh kolom terakhir.

```
UNLOAD (SELECT name1, address1, comment1, key1 FROM table1)
TO 's3://DOC-EXAMPLE-BUCKET/ partitioned/'
WITH (format = 'TEXTFILE', partitioned_by = ARRAY['key1'])
```

Contoh berikut membongkar hasil query ke lokasi yang ditentukan menggunakan format file Parquet, kompresi ZSTD, dan tingkat kompresi ZSTD 4.

```
UNLOAD (SELECT * FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/'
WITH (format = 'PARQUET', compression = 'ZSTD', compression_level = 4)
```

Sumber daya tambahan

- Sederhanakan pipeline ETL dan ML Anda menggunakan fitur Amazon Athena UNLOAD di Big Data Blog.AWS

Fungsi di Amazon Athena

Untuk perubahan fungsi antara versi mesin Athena, lihat. Referensi versi mesin Athena Untuk daftar zona waktu yang dapat digunakan dengan`AT TIME ZONE`operator, lihatZona waktu yang didukung.

Mesin Athena versi 3

Fungsi di mesin Athena versi 3 didasarkan pada Trino. Untuk informasi tentang fungsi, operator, dan ekspresi Trino, lihat Fungsi dan operator dan subbagian berikut dari dokumentasi Trino.

- Agregat
- Array
- Biner
- Bitwise
- Warna
- Perbandingan
- Bersyarat
- Konversi
- Tanggal dan waktu
- Desimal
- Geospasial
- HyperLogLog
- Alamat IP
- JSON
- Lambda
- Logis
- Pembelajaran mesin
- Peta
- Matematika

- Intisari kuantil
- Ekspresi reguler
- Sesi
- Set Intisari
- Tali
- Tabel
- Teradata
- T-Digest
- URL
- UUID
- Jendela

fungsi invoker_principal ()

`invoker_principal`Fungsinya unik untuk mesin Athena versi 3 dan tidak ditemukan di Trino.

Mengembalikan `VARCHAR` yang berisi ARN dari prinsipal (peran IAM atau identitas Pusat Identitas) yang menjalankan kueri memanggil fungsi. Misalnya, jika pemanggil kueri menggunakan izin peran IAM untuk menjalankan kueri, fungsi mengembalikan ARN dari peran IAM. Peran yang menjalankan kueri harus mengizinkan `LakeFormation:GetDataLakePrincipal` tindakan.

Penggunaan

```
SELECT invoker_principal()
```

Tabel berikut menunjukkan hasil contoh.

#	_col0
1	*arn:aws:iam:: 111122223333:peran/admin*

Versi mesin Athena 2

Fungsi di mesin Athena versi 2 didasarkan pada Presto 0.217. Untuk fungsi geospasial di Athena mesin versi 2, lihatFungsi geospasial di mesin Athena versi 2.

> **Note**
>
> Dokumentasi khusus versi untuk fungsi Presto 0.217 tidak lagi tersedia. Untuk informasi tentang fungsi, operator, dan ekspresi Presto saat ini, lihat fungsi dan operator Presto, atau kunjungi tautan subkategori di bagian ini.

- Operator logis
- Fungsi perbandingan dan operator
- Ekspresi bersyarat
- Fungsi konversi
- Fungsi dan operator matematika
- Fungsi bitwise
- Fungsi desimal dan operator
- Fungsi dan operator string
- Fungsi biner
- Fungsi dan operator tanggal dan waktu
- Fungsi ekspresi reguler
- Fungsi dan operator JSON
- Fungsi URL
- Fungsi agregat
- Fungsi jendela
- Fungsi warna
- Fungsi dan operator array
- Fungsi peta dan operator
- Ekspresi dan fungsi Lambda
- Fungsi Teradata

Zona waktu yang didukung

Anda dapat menggunakan `AT TIME ZONE` operator dalam `SELECT timestamp` pernyataan untuk menentukan zona waktu untuk stempel waktu yang dikembalikan, seperti pada contoh berikut:

```
SELECT timestamp '2012-10-31 01:00 UTC' AT TIME ZONE 'America/Los_Angeles' AS la_time;
```

Hasil

```
la_time

2012-10-30 18:00:00.000 America/Los_Angeles
```

Daftar berikut berisi zona waktu yang dapat digunakan dengan`AT TIME ZONE`Operator di Athena. Untuk fungsi dan contoh terkait zona waktu tambahan, lihat[Fungsi dan contoh zona waktu](#).

```
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
```

Fungsi dan contoh zona waktu

Berikut ini adalah beberapa fungsi dan contoh terkait zona waktu tambahan.

- at_timezone (***timestamp***, ***zone***) - *Mengembalikan nilai timestamp dalam waktu lokal yang sesuai untuk zona.*

  Contoh

  ```
  SELECT at_timezone(timestamp '2021-08-22 00:00 UTC', 'Canada/Newfoundland')
  ```

  Hasil

  ```
  2021-08-21 21:30:00.000 Canada/Newfoundland
  ```

- timezone_hour (***timestamp***) - Mengembalikan jam zona waktu offset dari stempel waktu sebagai. `bigint`

  Contoh

  ```
  SELECT timezone_hour(timestamp '2021-08-22 04:00 UTC' AT TIME ZONE 'Canada/
  Newfoundland')
  ```

  Hasil

  ```
  -2
  ```

- timezone_minute (***timestamp***) *- Mengembalikan menit zona waktu offset dari stempel waktu sebagai.* `bigint`

  Contoh

  ```
  SELECT timezone_minute(timestamp '2021-08-22 04:00 UTC' AT TIME ZONE 'Canada/
  Newfoundland')
  ```

  Hasil

  ```
  -30
  ```

- with_timezone (***timestamp***, ***zone***) *- Mengembalikan timestamp dengan zona waktu dari timestamp dan nilai zona yang ditentukan.*

  Contoh

  ```
  SELECT with_timezone(timestamp '2021-08-22 04:00', 'Canada/Newfoundland')
  ```

  Hasil

  ```
  2021-08-22 04:00:00.000 Canada/Newfoundland
  ```

Pernyataan DDL

Gunakan pernyataan DDL berikut langsung di Athena.

Mesin permintaan Athena didasarkan sebagian padaHiveQL DDDDL.

Athena tidak mendukung semua pernyataan DDL, dan ada beberapa perbedaan antara HiveQL DDL dan Athena DDL. Untuk informasi selengkapnya, lihat topik referensi dalam bagian ini DDL tidak didukung.

Topik

- DDL tidak didukung
- 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

DDL tidak didukung

Pernyataan DDL berikut tidak didukung oleh Athena:

- ALTER INDEX
- UBAH TABEL *TABLE_name* PARTISI ARSIP
- UBAH TABEL *table_name* DIKELOMPOKKAN OLEH
- UBAH TABEL *table_name* EXCHANGE PARTITION
- UBAH TABEL *table_name* TIDAK DIKELOMPOKKAN
- UBAH TABEL *table_name* TIDAK MIRING
- UBAH TABEL *table_name* TIDAK DIURUTKAN
- ALTER TABLE *table_name* NOT STORED AS DIRECTORIES
- UBAH TABEL *table_name* partisiSpec UBAH KOLOM
- ALTER TABLE *table_name* PartisionSpec COMPACT
- UBAH TABEL *table_name* partisiSpec CONCATENATE
- UBAH TABEL *table_name* partisiSpec SET FILEFORMAT
- UBAH TABEL *table_name* SET SERDEPROPERTIES
- UBAH TABEL *table_name* SET LOKASI MIRING
- UBAH TABEL *table_name* MIRING OLEH
- ALTER TABLE *table_name* TOUCH
- UBAH TABEL *table_name* UNARCHIVE PARTITION
- COMMIT
- CREATE INDEX
- CREATE ROLE
- *CREATE TABLE table_name LIKE existing_table_name*
- CREATE TEMPORARY MACRO
- DELETE FROM
- DESCRIBE DATABASE
- DFS

- DROP INDEX
- DROP ROLE
- DROP TEMPORARY MACRO
- EXPORT TABLE
- GRANT ROLE
- IMPORT TABLE
- LOCK DATABASE
- LOCK TABLE
- REVOKE ROLE
- ROLLBACK
- SHOW COMPACTIONS
- SHOW CURRENT ROLES
- SHOW GRANT
- SHOW INDEXES
- SHOW LOCKS
- SHOW PRINCIPALS
- SHOW ROLE GRANT
- SHOW ROLES
- SHOW STATS
- SHOW TRANSACTIONS
- START TRANSACTION
- UNLOCK DATABASE
- UNLOCK TABLE

ALTER DATABASE SET DBPROPERTIES

Menciptakan satu atau lebih properti untuk basis data. Penggunaan`DATABASE`dan`SCHEMA`dapat dipertukarkan; itu berarti hal yang sama.

Sinopsis

```
ALTER {DATABASE|SCHEMA} database_name
```

```
  SET DBPROPERTIES ('property_name'='property_value' [, ...] )
```

Parameter

SET DBPROPERTIES ('property_name'='property_value' [,...]

Menentukan properti atau properti untuk basis data bernama`property_name`dan menetapkan nilai untuk masing-masing properti masing-masing sebagai`property_value`. Jika`property_name`sudah ada, nilai lama ditimpa dengan`property_value`.

Contoh

```
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

Menambahkan satu atau lebih kolom ke tabel yang ada. Saat opsional`PARTITION`sintaks yang digunakan, update metadata partisi.

Sinopsis

```
ALTER TABLE table_name
  [PARTITION
   (partition_col1_name = partition_col1_value
   [,partition_col2_name = partition_col2_value][,...])]
  ADD COLUMNS (col_name data_type)
```

Parameter

PARTASI (partition_col_name=partition_col_value [,...])

Membuat partisi dengan kombinasi nama kolom/nilai yang Anda tentukan. Lampirkan`partition_col_value`dalam tanda kutip hanya jika tipe data kolom adalah string.

TAMBAHKAN KOLOM (col_name data_type [, col_name data_type,...])

Menambahkan kolom setelah kolom yang ada tapi sebelum kolom partisi.

Contoh

```
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)
```

Catatan

- Untuk melihat kolom tabel baru di panel navigasi Athena Kueri Editor setelah Anda menjalankan`ALTER TABLE ADD COLUMNS`, secara manual refresh daftar tabel di editor, dan kemudian memperluas tabel lagi.
- `ALTER TABLE ADD COLUMNS`Tidak bekerja untuk kolom dengan`date`Datatype. Untuk mengatasi masalah ini, gunakan`timestamp`datatype sebagai gantinya.

ALTER TABLE ADD PARTITION

Menciptakan satu atau lebih kolom partisi untuk tabel. Setiap partisi terdiri dari satu atau lebih kombinasi kolom nama/nilai yang berbeda. Sebuah direktori data terpisah dibuat untuk setiap kombinasi tertentu, yang dapat meningkatkan performa kueri dalam beberapa keadaan. Kolom dipartisi tidak ada dalam data tabel itu sendiri, jadi jika Anda menggunakan nama kolom yang memiliki nama yang sama dengan kolom dalam tabel itu sendiri, Anda mendapatkan kesalahan. Untuk informasi selengkapnya, lihat Partisi data di Athena.

Di Athena, tabel dan partisi yang harus menggunakan format data yang sama tetapi skema mereka mungkin berbeda. Untuk informasi selengkapnya, lihat Pembaruan dalam tabel dengan partisi.

Untuk informasi tentang izin tingkat sumber daya yang diperlukan dalam kebijakan IAM (termasuk`glue:CreatePartition`), lihat Izin AWS Glue API: Referensi tindakan dan sumber daya, dan. Akses berbutir halus ke database dan tabel di AWS Glue Data Catalog Untuk informasi pemecahan masalah tentang izin saat menggunakan Athena, lihatIzinBagian dariPemecahan Masalah di Athenatopik.

Sinopsis

```
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

Saat Anda menambahkan partisi, Anda menentukan satu atau lebih kolom nama/nilai pasangan untuk partisi dan jalur Amazon S3 tempat file data untuk partisi tersebut berada.

[JIKA TIDAK ADA]

Menyebabkan kesalahan yang akan ditekan jika partisi dengan definisi yang sama sudah ada.

PARTASI (partition_col_name=partition_col_value [,...])

Membuat partisi dengan kombinasi nama kolom/nilai yang Anda tentukan. Lampirkan`partition_col_value`dalam karakter string hanya jika tipe data kolom adalah string.

[LOKASI 'lokasi']

Menentukan direktori di mana untuk menyimpan partisi didefinisikan oleh pernyataan sebelumnya. `LOCATION`Klausa bersifat opsional ketika data menggunakan partisi gaya Hive (). `pk1=v1/pk2=v2/pk3=v3` Dengan partisi gaya HIVE, URI Amazon S3 lengkap dibangun secara otomatis dari lokasi tabel, nama kunci partisi, dan nilai kunci partisi. Untuk informasi selengkapnya, lihat Partisi data di Athena.

Pertimbangan

Amazon Athena tidak memaksakan batas tertentu pada jumlah partisi yang dapat Anda tambahkan dalam satu pernyataan DDL. `ALTER TABLE ADD PARTITION` Namun, jika Anda perlu menambahkan sejumlah besar partisi, pertimbangkan untuk memecah operasi menjadi batch yang lebih kecil untuk menghindari potensi masalah kinerja. Contoh berikut menggunakan perintah

berturut-turut untuk menambahkan partisi secara individual dan digunakan `IF NOT EXISTS` untuk menghindari penambahan duplikat.

```
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')
```

Ketika bekerja dengan partisi di Athena, juga perlu diingat hal-hal berikut:

- Meskipun Athena mendukung AWS Glue tabel kueri yang memiliki 10 juta partisi, Athena tidak dapat membaca lebih dari 1 juta partisi dalam satu pemindaian.
- Untuk mengoptimalkan kueri Anda dan mengurangi jumlah partisi yang dipindai, pertimbangkan strategi seperti pemangkasan partisi atau menggunakan indeks partisi.
- Jika Anda tidak menggunakan AWS Glue Data Catalog, jumlah maksimum partisi per tabel adalah 20.000. Anda dapat meminta penambahan kuota.

Untuk pertimbangan tambahan mengenai bekerja dengan partisi di Athena, lihat. Partisi data di Athena

Contoh

Contoh berikut menambahkan partisi tunggal ke tabel untuk HIVE-style data partisi.

```
ALTER TABLE orders ADD
  PARTITION (dt = '2016-05-14', country = 'IN');
```

Contoh berikut menambahkan beberapa partisi ke tabel untuk HIVE-style data partisi.

```
ALTER TABLE orders ADD
  PARTITION (dt = '2016-05-31', country = 'IN')
  PARTITION (dt = '2016-06-01', country = 'IN');
```

Jika tabel bukan untuk data partisi gaya HIVE, `LOCATION` klausa diperlukan dan harus menjadi URI Amazon S3 lengkap untuk awalan yang berisi data partisi.

```
ALTER TABLE orders ADD
  PARTITION (dt = '2016-05-31', country = 'IN') LOCATION 's3://DOC-EXAMPLE-BUCKET/path/
to/INDIA_31_May_2016/'
```

```
  PARTITION (dt = '2016-06-01', country = 'IN') LOCATION 's3://DOC-EXAMPLE-BUCKET/path/
to/INDIA_01_June_2016/';
```

Untuk mengabaikan kesalahan ketika partisi sudah ada, gunakan `IF NOT EXISTS` klausa, seperti pada contoh berikut.

```
ALTER TABLE orders ADD IF NOT EXISTS
  PARTITION (dt = '2016-05-14', country = 'IN');
```

`_$folder$`File byte nol

Jika Anda menjalankan `ALTER TABLE ADD PARTITION` pernyataan dan salah menentukan partisi yang sudah ada dan lokasi Amazon S3 yang salah, file placeholder nol byte dari *`partition_value`*`_$folder$` format dibuat di Amazon S3. Anda harus menghapus file-file ini secara manual.

Untuk mencegah hal ini terjadi, gunakan `ADD IF NOT EXISTS` sintaks dalam `ALTER TABLE ADD PARTITION` pernyataan Anda, seperti pada contoh berikut.

```
ALTER TABLE table_name ADD IF NOT EXISTS PARTITION […]
```

ALTER TABLE DROP PARTITION

Tetes satu atau lebih ditentukan partisi untuk tabel bernama.

Sinopsis

```
ALTER TABLE table_name DROP [IF EXISTS] PARTITION (partition_spec) [, PARTITION
 (partition_spec)]
```

Parameter

[JIKA ADA]

Menekan pesan kesalahan jika partisi yang ditentukan tidak ada.

PARTASI (partisi_spec)

Setiap`partition_spec`menentukan kombinasi nama kolom/nilai dalam bentuk`partition_col_name = partition_col_value [,...]`.

Contoh

```
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');
```

Catatan

`ALTER TABLE DROP PARTITION`Pernyataan tersebut tidak menyediakan sintaks tunggal untuk menjatuhkan semua partisi sekaligus atau mendukung kriteria penyaringan untuk menentukan rentang partisi yang akan dijatuhkan.

Sebagai solusinya, Anda dapat menggunakan AWS Glue API GetPartitionsdan BatchDeletePartitiontindakan dalam pembuatan skrip. `GetPartitions`Tindakan ini mendukung ekspresi filter yang kompleks seperti yang ada dalam `WHERE` ekspresi SQL. Setelah Anda gunakan `GetPartitions` untuk membuat daftar partisi yang difilter untuk dihapus, Anda dapat menggunakan `BatchDeletePartition` tindakan untuk menghapus partisi dalam batch 25.

> **Important**
>
> Karena masalah yang diketahui, ketika partisi yang tidak valid ditentukan untuk `ALTER TABLE DROP PARTITION` pernyataan, semua partisi untuk tabel dijatuhkan. AWS Glue Misalnya, pernyataan berikut akan menjatuhkan semua partisi untuk tabel *`my_table`* meskipun partisi yang ditentukan tidak ada. Sebagai solusinya, pastikan Anda memasukkan informasi partisi dengan benar sebelum menjalankan pernyataan. `ALTER TABLE DROP PARTITION`
>
> ```
> ALTER TABLE my_table DROP IF EXISTS PARTITION(zzz='');
> ```

ALTER TABLE RENAME PARTITION

Mengganti nama nilai partisi.

> **Note**
>
> ALTER TABLE RENAME PARTITIONtidak mengganti nama kolom partisi. Untuk mengubah nama kolom partisi, Anda dapat menggunakan AWS Glue konsol. Untuk informasi selengkapnya, lihat Mengganti nama kolom partisi di AWS Glue nanti dalam dokumen ini.

Sinopsis

Untuk tabel bernama`table_name`, ganti nama nilai partisi yang ditentukan oleh `partition_spec` ke nilai yang ditentukan oleh. `new_partition_spec`

```
ALTER TABLE table_name PARTITION (partition_spec) RENAME TO PARTITION
 (new_partition_spec)
```

Parameter

PARTASI (partisi_spec)

Setiap`partition_spec`menentukan kombinasi nama kolom/nilai dalam bentuk`partition_col_name = partition_col_value [,...]`.

Contoh

```
ALTER TABLE orders
PARTITION (dt = '2014-05-14', country = 'IN') RENAME TO PARTITION (dt = '2014-05-15',
 country = 'IN');
```

Mengganti nama kolom partisi di AWS Glue

Gunakan prosedur berikut untuk mengganti nama nama kolom partisi di AWS Glue konsol.

Untuk mengganti nama kolom partisi tabel di konsol AWS Glue

1. Masuk ke AWS Management Console dan buka AWS Glue konsol di https://console.aws.amazon.com/glue/.
2. Di panel navigasi, pilih Tabel.
3. Pada halaman Tabel, gunakan kotak pencarian Filter tabel untuk menemukan tabel yang ingin Anda ubah.

4. Di kolom Nama, pilih tautan tabel yang ingin Anda ubah.
5. Pada halaman detail untuk tabel, di bagian Skema, lakukan salah satu hal berikut:
   - Untuk membuat perubahan nama dalam format JSON, pilih Edit skema sebagai JSON.
   - Untuk mengubah nama secara langsung, pilih Edit skema. Prosedur ini memilih skema Edit.
6. Pilih kotak centang untuk kolom yang dipartisi yang ingin Anda ganti namanya, lalu pilih Edit.
7. Dalam kotak dialog Edit skema entri, untuk Nama, masukkan nama baru untuk kolom partisi.
8. Pilih Simpan sebagai versi tabel baru. Tindakan ini memperbarui nama kolom partisi dan mempertahankan riwayat evolusi skema tanpa membuat salinan fisik terpisah dari data Anda.
9. Untuk membandingkan versi tabel, pada halaman detail untuk tabel, pilih Tindakan, lalu pilih Bandingkan versi.

Sumber daya tambahan

Untuk informasi selengkapnya tentang partisi, lihat. Partisi data di Athena

ALTER TABLE REPLACE COLUMNS

Menghapus semua kolom yang ada dari tabel yang dibuat dengan LazySimpleSerDedan menggantinya dengan kumpulan kolom yang ditentukan. Saat opsionalPARTITIONsintaks yang digunakan, update metadata partisi. Anda juga dapat menggunakanALTER TABLE REPLACE COLUMNSuntuk menjatuhkan kolom dengan menentukan hanya kolom yang ingin Anda simpan.

Sinopsis

```
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

PARTASI (partition_col_name=partition_col_value [,...])

Menentukan partisi dengan kombinasi nama kolom/nilai yang Anda tentukan. Lampirkan`partition_col_value`dalam tanda kutip hanya jika tipe data kolom adalah string.

GANTI KOLOM (col_name data_type [, col_name data_type,...])

Menggantikan kolom yang ada dengan nama kolom dan tipe data yang ditentukan.

Catatan

- Untuk melihat perubahan kolom tabel di panel navigasi Editor Kueri Athena setelah menjalankan`ALTER TABLE REPLACE COLUMNS`, Anda mungkin harus menyegarkan daftar tabel di editor secara manual, lalu memperluas tabel lagi.
- `ALTER TABLE REPLACE COLUMNS`tidak bekerja untuk kolom dengan`date`Datatype. Untuk mengatasi masalah ini, gunakan`timestamp`datatype dalam tabel sebagai gantinya.
- Perhatikan bahwa bahkan jika Anda mengganti hanya satu kolom, sintaksnya harus`ALTER TABLE` *`table-name`* `REPLACE COLUMNS`, dengan kolom dalam bentuk jamak. Anda harus menentukan tidak hanya kolom yang ingin Anda ganti, tetapi kolom yang ingin Anda simpan - jika tidak, kolom yang tidak Anda tentukan akan dijatuhkan. Sintaks dan perilaku ini berasal dari Apache Hive DDL. Untuk referensi, lihat Tambahkan/Ganti kolom dalam dokumentasi Apache.

Contoh

Dalam contoh berikut, tabel`names_cities`, yang dibuat menggunakan LazySimpleSerDe, memiliki tiga kolom bernama`col1`,`col2`, dan`col3`. Semua kolom bertipe`string`. Untuk menampilkan kolom dalam tabel, perintah berikut menggunakan SHOW COLUMNS pernyataan.

```
SHOW COLUMNS IN names_cities
```

Hasil kueri:

```
col1
col2
col3
```

`ALTER TABLE REPLACE COLUMNS`Perintah berikut menggantikan nama kolom dengan`first_name`,`last_name`, dan`city`. Data sumber yang mendasarinya tidak terpengaruh.

```
ALTER TABLE names_cities
REPLACE COLUMNS (first_name string, last_name string, city string)
```

Untuk menguji hasilnya, `SHOW COLUMNS` dijalankan lagi.

```
SHOW COLUMNS IN names_cities
```

Hasil kueri:

```
first_name
last_name
city
```

Cara lain untuk menampilkan nama kolom baru adalah dengan melihat pratinjau tabel di Athena Query Editor atau menjalankan kueri Anda sendiri`SELECT`.

ALTER TABLE SET LOCATION

Mengubah lokasi untuk tabel bernama`table_name`, dan secara opsional partisi dengan`partition_spec`.

Sinopsis

```
ALTER TABLE table_name [ PARTITION (partition_spec) ] SET LOCATION 'new location'
```

Parameter

PARTASI (partisi_spec)

Menentukan partisi dengan parameter`partition_spec`Lokasi yang ingin Anda ubah. Parameter`partition_spec`menentukan kombinasi nama kolom/nilai dalam bentuk`partition_col_name = partition_col_value`.

SET LOKASI 'lokasi baru'

Menentukan lokasi baru, yang harus menjadi lokasi Amazon S3. Untuk informasi tentang sintaks, lihat,Lokasi Tabel di Amazon S3.

Contoh

```
ALTER TABLE customers PARTITION (zip='98040', state='WA') SET LOCATION 's3://DOC-
EXAMPLE-BUCKET/custdata/';
```

ALTER TABLE SET TBLPROPERTIES

Menambahkan properti metadata kustom atau yang telah ditetapkan ke tabel dan menetapkan nilai-nilai yang ditetapkan mereka. Untuk melihat properti dalam tabel, gunakanSHOW TBLPROPERTIESPerintah.

Apache HiveTabel terkelolatidak didukung, jadi pengaturan`'EXTERNAL'='FALSE'`tidak berpengaruh.

Sinopsis

```
ALTER TABLE table_name SET TBLPROPERTIES ('property_name' = 'property_value' [ , ... ])
```

Parameter

SET TBLPROPERTIES ('property_name' = 'property_value' [,...])

Menentukan properti metadata untuk menambahkan sebagai`property_name`edan nilai untuk masing-masing sebagai`property value`. Jika`property_name`sudah ada, nilainya diatur ke yang baru ditentukan`property_value`.

Properti tabel yang telah ditetapkan berikut memiliki kegunaan khusus.

Properti yang telah ditentukan	Deskripsi
`classification`	Menunjukkan tipe data untuk AWS Glue. Kemungkinan nilai adalah `csv`, `parquet`, `orc`, `avro`, atau `json`. Tabel yang dibuat untuk Athena di CloudTrail konsol ditambahkan `cloudtrail` sebagai nilai untuk properti. `classification` Untuk informasi selengkapnya, lihat CREATE TABLE di halaman ini.
`has_encrypted_data`	Menunjukkan apakah set data yang ditentukan oleh `LOCATION` dienkripsi. Untuk informasi selengkapnya, lihat bagian TBLPROPERTIES CREATE TABLE dan Membuat tabel berdasarkan kumpulan data terenkripsi di Amazon S3.
`orc.compress`	Menentukan format kompresi untuk data dalam format ORC. Untuk informasi selengkapnya, lihat ORC SerDe.

Properti yang telah ditentukan	Deskripsi
`parquet.compression`	Menentukan format kompresi untuk data dalam format Parquet. Untuk informasi selengkapnya, lihat Parket SerDe.
`write.compression`	Menentukan format kompresi untuk data dalam file teks atau format JSON. Untuk format Parket dan ORC, gunakan `orc.compress` properti `parquet.compression` dan masing-masing.
`compression_level`	Menentukan tingkat kompresi untuk digunakan. Properti ini hanya berlaku untuk kompresi ZSTD. Nilai yang mungkin adalah dari 1 hingga 22. Nilai default-nya adalah 3. Untuk informasi selengkapnya, lihat Menggunakan tingkat kompresi ZSTD di Athena.
`projection.*`	Properti kustom yang digunakan dalam proyeksi partisi yang memungkinkan Athena mengetahui pola partisi apa yang diharapkan saat menjalankan kueri di atas tabel. Untuk informasi selengkapnya, lihat Proyeksi partisi dengan Amazon Athena.
`skip.header.line.count`	Mengabaikan header dalam data saat Anda menentukan tabel. Untuk informasi selengkapnya, lihat Mengabaikan header.
`storage.location.template`	Menentukan templat jalur Amazon S3 kustom untuk partisi diproyeksikan. Untuk informasi selengkapnya, lihat Menyiapkan proyeksi partisi.

Contoh

Contoh berikut menambahkan catatan komentar untuk properti tabel.

```
ALTER TABLE orders
SET TBLPROPERTIES ('notes'="Please don't drop this table.");
```

Contoh berikut memodifikasi tabel `existing_table` untuk menggunakan format file Parket dengan kompresi ZSTD dan tingkat kompresi ZSTD 4.

```
ALTER TABLE existing_table
SET TBLPROPERTIES ('parquet.compression' = 'ZSTD', 'compression_level' = 4)
```

CREATE DATABASE

Menciptakan basis data. PenggunaanDATABASEdanSCHEMAdapat dipertukarkan. Mereka berarti hal yang sama.

> **Note**
>
> Untuk contoh membuat database, membuat tabel, dan menjalankan SELECT kueri pada tabel di Athena, lihat. Memulai

Sinopsis

```
CREATE {DATABASE|SCHEMA} [IF NOT EXISTS] database_name
  [COMMENT 'database_comment']
  [LOCATION 'S3_loc']
  [WITH DBPROPERTIES ('property_name' = 'property_value') [, ...]]
```

Parameter

[JIKA TIDAK ADA]

Menyebabkan kesalahan yang akan ditekan jika basis data bernamadatabase_namesudah ada.

[KOMENTAR database_comment]

Menetapkan nilai metadata untuk properti metadata built-in bernamacommentdan nilai yang Anda berikan untukdatabase_comment. Dalam AWS Glue, COMMENT konten ditulis ke Description bidang properti database.

[LOKASI S3_loc]

Menentukan lokasi tempat file basis data dan metastore akan ada sebagaiS3_loc. Lokasi harus lokasi Amazon S3.

[DENGAN DBPROPERTIES ('property_name' = 'property_value') [,...]]

Memungkinkan Anda untuk menentukan properti metadata kustom untuk definisi basis data.

Contoh

```
CREATE DATABASE clickstreams;
```

```
CREATE DATABASE IF NOT EXISTS clickstreams
  COMMENT 'Site Foo clickstream data aggregates'
  LOCATION 's3://DOC-EXAMPLE-BUCKET/clickstreams/'
  WITH DBPROPERTIES ('creator'='Jane D.', 'Dept.'='Marketing analytics');
```

Melihat properti database

Untuk melihat properti database untuk database yang Anda buat dalam AWSDataCatalog menggunakanCREATE DATABASE, Anda dapat menggunakan AWS CLI perintah `aws glue get-database`, seperti pada contoh berikut:

```
aws glue get-database --name <your-database-name>
```

Output akan terlihat seperti berikut.

```
{
    "Database": {
        "Name": "<your-database-name>",
        "Description": "<your-database-comment>",
        "LocationUri": "s3://DOC-EXAMPLE-BUCKET",
        "Parameters": {
            "<your-database-property-name>": "<your-database-property-value>"
        },
        "CreateTime": 1603383451.0,
        "CreateTableDefaultPermissions": [
            {
                "Principal": {
                    "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                },
                "Permissions": [
                    "ALL"
                ]
            }
        ]
    }
}
```

Untuk informasi selengkapnya tentang AWS CLI, lihat Panduan AWS Command Line Interface Pengguna.

CREATE TABLE

Membuat tabel dengan nama dan parameter yang Anda tentukan.

> **Note**
>
> Halaman ini berisi informasi referensi ringkasan. Untuk informasi selengkapnya tentang membuat tabel di Athena dan `CREATE TABLE` pernyataan contoh, lihat. Membuat tabel di Athena Untuk contoh membuat database, membuat tabel, dan menjalankan `SELECT` kueri pada tabel di Athena, lihat. Memulai

Sinopsis

```
CREATE EXTERNAL TABLE [IF NOT EXISTS]
 [db_name.]table_name [(col_name data_type [COMMENT col_comment] [, ...] )]
 [COMMENT table_comment]
 [PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]
 [CLUSTERED BY (col_name, col_name, ...) INTO num_buckets BUCKETS]
 [ROW FORMAT row_format]
 [STORED AS file_format]
 [WITH SERDEPROPERTIES (...)]
 [LOCATION 's3://DOC-EXAMPLE-BUCKET/[folder]/']
 [TBLPROPERTIES ( ['has_encrypted_data'='true | false',]
 ['classification'='aws_glue_classification',] property_name=property_value [, ...] ) ]
```

Parameter

EKSTERNAL

Menentukan bahwa tabel didasarkan pada file data yang mendasari yang ada di Amazon S3, di`LOCATION`yang Anda tentukan. Kecuali saat membuat tabel Iceberg, selalu gunakan kata kunci. `EXTERNAL` Jika Anda menggunakan `CREATE TABLE` tanpa `EXTERNAL` kata kunci untuk tabel non-Iceberg, Athena mengeluarkan kesalahan. Saat Anda membuat tabel eksternal, data yang direferensikan harus sesuai dengan format default atau format yang Anda tentukan dengan`ROW FORMAT`,`STORED AS`, dan`WITH SERDEPROPERTIES`Klausul.

[JIKA TIDAK ADA]

Parameter ini memeriksa apakah tabel dengan nama yang sama sudah ada. Jika ya, parameter kembaliTRUE, dan Amazon Athena membatalkan tindakan. CREATE TABLE Karena pembatalan terjadi sebelum Athena memanggil katalog data, itu tidak memancarkan AWS CloudTrail acara.

[db_name.] table_name

Menentukan nama untuk tabel yang akan dibuat. Parameter db_name opsional menentukan basis data tempat tabel ada. Jika dihilangkan, basis data saat ini diasumsikan. Jika nama tabel termasuk angka, lampirkan table_name dalam tanda petikan, sebagai contoh "table123". Jika table_name dimulai dengan garis bawah, gunakan backticks, misalnya, `_mytable`. Karakter khusus (selain garis bawah) tidak didukung.

Nama tabel Athena adalah sensitif hurup; tetapi, jika Anda bekerja dengan Apache Spark, Spark membutuhkan nama tabel huruf kecil.

[(col_name data_type [COMMENT col_comment] [, ...])]

Menentukan nama untuk setiap kolom yang akan dibuat, bersama dengan tipe data kolom ini. Nama kolom tidak mengizinkan karakter khusus selain garis bawah(_). Jikacol_namedimulai dengan garis bawah, lampirkan nama kolom di backticks, misalnya`_mycolumn`.

Nilai data_type dapat menjadi salah satu dari yang berikut:

- boolean – Nilai adalah true dan false.
- tinyint— Bilangan bulat bertanda 8-bit dalam format komplemen dua, dengan nilai minimum -2^7 dan nilai maksimum 2 ^ 7-1.
- smallint— Bilangan bulat bertanda 16-bit dalam format komplemen dua, dengan nilai minimum -2^15 dan nilai maksimum 2 ^ 15-1.
- int— Dalam kueri Data Definition Language (DDL) sepertiCREATE TABLE, gunakan int kata kunci untuk mewakili bilangan bulat. Dalam kueri lain, gunakan kata kunciinteger, di mana integer direpresentasikan sebagai nilai bertanda 32-bit dalam format komplemen dua, dengan nilai minimum 2^31 dan nilai maksimum 2^31-1. Dalam driver JDBC,integerdikembalikan, untuk memastikan kompatibilitas dengan aplikasi analisis bisnis.
- bigint— Integer bertanda 64-bit dalam format komplemen dua, dengan nilai minimum -2^63 dan nilai maksimum 2 ^ 63-1.
- double— Nomor floating point presisi ganda yang ditandatangani 64-bit. Kisarannya adalah 4.94065645841246544e-324d hingga 1.79769313486231570e+308d, positif atau negatif. doublemengikuti Standar IEEE untuk Floating-Point Arithmetic (IEEE 754).

- `float`— Nomor floating point presisi tunggal bertanda 32-bit. Kisarannya adalah 1.40129846432481707e-45 hingga 3.40282346638528860e+38, positif atau negatif. `float`mengikuti Standar IEEE untuk Floating-Point Arithmetic (IEEE 754). Setara dengan `real` di Presto. Di Athena, gunakan `float`dalam pernyataan DDL seperti `CREATE TABLE` dan`real` dalam fungsi SQL seperti `SELECT CAST`. AWS Glue Crawler mengembalikan nilai dalam`float`, dan Athena `real` menerjemahkan `float` dan mengetik secara internal (lihat 5 Juni 2018 catatan rilis).
- `decimal [ (`*`precision`*`, `*`scale`*`) ]`, tempat*`precision`*adalah jumlah digit, dan*`scale`*(opsional) adalah jumlah digit di bagian pecahan, default adalah 0. Sebagai contoh, gunakan definisi jenis ini:`decimal(11,5)`,`decimal(15)`. Nilai maksimum untuk *`presisi`* adalah 38, dan nilai maksimum untuk *`skala`* adalah 38.

  Untuk menentukan nilai desimal sebagai literal, seperti saat memilih baris dengan nilai desimal tertentu dalam ekspresi DDL kueri, tentukan tipe definisi `decimal`, dan cantumkan nilai desimal sebagai literal (dalam tanda kutip tunggal) dalam kueri Anda, seperti dalam contoh ini: `decimal_value = decimal '0.12'`.
- `char` – Data karakter dengan panjang tetap, dengan panjang yang ditentukan antara 1 dan 255, seperti `char(10)`. Untuk informasi lebih lanjut, lihat tipe data CHAR Hive.
- `varchar` – Berbagai data karakter panjang, dengan panjang yang ditentukan antara 1 dan 65535, seperti `varchar(10)`. Untuk informasi selengkapnya, lihat tipe data VARCHAR Hive.
- `string` – Sebuah literal string yang disertakan dalam tanda kutip tunggal atau ganda.

  > **Note**
  >
  > Tipe data non-string tidak dapat dilemparkan ke`string`di Athena; melemparkan mereka ke`varchar`Sebagai gantinya
- `binary`— (untuk data di Parket)
- `date` – Tanggal dalam format ISO, seperti *`YYYY`*`-`*`MM`*`-`*`DD`*. Misalnya, `date '2008-09-15'`. Pengecualian adalah SerDe OpenCSV, yang menggunakan jumlah hari yang telah berlalu sejak 1 Januari 1970. Untuk informasi selengkapnya, lihat OpenCSV untuk SerDe memproses CSV.
- `timestamp`— Tanggal dan waktu instan dalam format yang `java.sql.Timestamp`kompatibel hingga resolusi maksimum milidetik, seperti*`yyyy`*`-`*`MM`*`-`*`dd HH`*`:`*`mm`*`:`*`ss`*`[.f...]`. Misalnya, `timestamp '2008-09-15 03:04:05.324'`. Pengecualian adalah SerDe OpenCSV, yang `TIMESTAMP` menggunakan data dalam format numerik UNIX

(misalnya,). `1579059880000` Untuk informasi selengkapnya, lihat OpenCSV untuk SerDe memproses CSV.

- `array` < data_type >
- `map` < primitive_type, data_type >
- `struct`< col_name: data_type [komentar col_comment] [, ...] >

[Komentar table_comment]

Membuat`comment`tabel properti dan populates dengan`table_comment`Anda tentukan.

[Partisi OLEH (col_name data_type [COMMENT col_comment],...)]

Membuat tabel dipartisi dengan satu atau lebih kolom partisi yang memiliki`col_name`,`data_type`dan`col_comment`ditentukan. Sebuah tabel dapat memiliki satu atau lebih partisi, yang terdiri dari nama kolom yang berbeda dan kombinasi nilai. Sebuah direktori data terpisah dibuat untuk setiap kombinasi tertentu, yang dapat meningkatkan performa kueri dalam beberapa keadaan. Kolom yang dipartisi tidak ada dalam data tabel itu sendiri. Jika Anda menggunakan nilai untuk`col_name`Yang sama dengan kolom tabel, Anda akan mendapatkan pesan kesalahan. Untuk informasi selengkapnya, lihat,Data partisi.

> **Note**
>
> Setelah Anda membuat tabel dengan partisi, jalankan kueri berikutnya yang terdiri dariTABEL PERBAIKAN MSCKuntuk me-refresh metadata partisi, misalnya,`MSCK REPAIR TABLE cloudfront_logs;`. Untuk partisi yang tidak Hive kompatibel, gunakanALTER TABLE ADD PARTITIONuntuk memuat partisi sehingga Anda dapat mengkueri data.

[Berkelompokkan BY (col_name, col_name,...) Ke bucket num_ember]

Membagi, dengan atau tanpa partisi, data dalam ditentukan`col_name`ke dalam subset data yang disebutember. Parameter `num_buckets` menentukan jumlah bucket untuk dibuat. Bucketing dapat meningkatkan performa beberapa kueri pada set data yang besar.

[ROW FORMAT row_format]

Menentukan format baris tabel dan data sumber yang mendasari jika berlaku. Untuk`row_format`, Anda dapat menentukan satu atau lebih pembatas dengan`DELIMITED`atau, alternatif,

gunakanSERDEseperti yang dijelaskan di bawah ini. Jika ROW FORMAT dihilangkan atau ROW FORMAT DELIMITED ditentukan, asli SerDe digunakan.

- [BIDANG dibatasi diakhiri oleh char [melarikan diri oleh char]]
- [DILIMITED KOLEKSI ITEM diakhiri oleh arang]
- [KUNCI MAP diakhiri oleh char]
- [LINES diakhiri oleh char]
- [NULL didefinisikan sebagai char]

  Tersedia hanya dengan Hive 0.13 dan saat format file tersimpan AS adalahTEXTFILE.

--ATAU--

- SERDE 'serde_name' [DENGAN SERDEPROPERTIES (“property_name” = “property_value”, “property_name” = “property_value” [,...])]

  serde_nameIni menunjukkan SerDe penggunaan. WITH SERDEPROPERTIESKlausul ini memungkinkan Anda untuk menyediakan satu atau lebih properti kustom yang diizinkan oleh. SerDe

[STORED AS file_format]

Menentukan format file untuk data tabel. Jika dihilangkan,TEXTFILEadalah default. Pilihan untukfile_formatadalah:

- SEQUENCEFILE
- TEXTFILE
- RCFILE
- ORC
- PARQUET
- AVRO
- ION
- INputFORMAT input_format_classname output_format_format_classname

[LOKASI 's3://DOC-EXAMPLE-BUCKET/[folder]/']

Menentukan lokasi data yang mendasari di Amazon S3 dari mana tabel dibuat. Lokasi jalur harus nama bucket atau nama bucket dan satu atau lebih folder. Jika Anda menggunakan partisi,

tentukan akar data yang dipartisi. Untuk informasi selengkapnya tentang pengambilan tabel, lihat Lokasi tabel di Amazon S3. Untuk informasi tentang format data dan izin, lihatPersyaratan untuk tabel di Athena dan data di Amazon S3.

Gunakan garis miring trailing untuk folder atau bucket Anda. Jangan gunakan nama file atau karakter glob.

Gunakan:

`s3://DOC-EXAMPLE-BUCKET/`

`s3://DOC-EXAMPLE-BUCKET/folder/`

`s3://DOC-EXAMPLE-BUCKET/folder/anotherfolder/`

Jangan gunakan:

`s3://DOC-EXAMPLE-BUCKET`

`s3://DOC-EXAMPLE-BUCKET/*`

`s3://DOC-EXAMPLE-BUCKET/mydatafile.dat`

[TBLPROPERTIES (['has_encrypted_data'='true | false',] ['classification'='classification_value',] property_name=property_value [,...])]

Menentukan kustom metadata pasangan kunci-nilai untuk definisi tabel selain properti tabel yang telah ditetapkan, seperti`"comment"`.

has_encrypted_data — Athena memiliki properti bawaan,. `has_encrypted_data` Mengatur properti ini untuk`true`untuk menunjukkan bahwa set data yang mendasari ditentukan oleh`LOCATION`dienkripsi. Jika dihilangkan dan jika pengaturan grup kerja tidak menimpa pengaturan sisi klien,`false`diasumsikan. Jika dihilangkan atau diatur ke`false`ketika data dasar dienkripsi, hasil kueri dalam kesalahan. Untuk informasi selengkapnya, lihat Enkripsi diam.

klasifikasi — Tabel yang dibuat untuk Athena di CloudTrail konsol ditambahkan `cloudtrail` sebagai nilai untuk properti. `classification` Untuk menjalankan pekerjaan ETL AWS Glue , Anda harus membuat tabel dengan `classification` properti untuk menunjukkan tipe data untuk AWS Glue as`csv`,,`parquet`, `orc`a`vro`, ata`ujson`. Misalnya, `'classification'='csv'`. ETL tugas akan gagal jika Anda tidak menentukan properti ini. Anda kemudian dapat menentukan menggunakan AWS Glue konsol, API, atau CLI. Untuk informasi selengkapnya, lihat

Menggunakan AWS Glue pekerjaan untuk ETL dengan Athena dan Menulis Pekerjaan di AWS Glue di Panduan AWS Glue Pengembang.

compression_level - `compression_level` Properti menentukan tingkat kompresi untuk digunakan. Properti ini hanya berlaku untuk kompresi ZSTD. Nilai yang mungkin adalah dari 1 hingga 22. Nilai default-nya adalah 3. Untuk informasi selengkapnya, lihat Menggunakan tingkat kompresi ZSTD di Athena.

Untuk informasi selengkapnya tentang properti tabel lainnya, lihatALTER TABLE SET TBLPROPERTIES.

Contoh

`CREATE TABLE`Pernyataan contoh berikut membuat tabel berdasarkan data planet yang dipisahkan tab yang disimpan di Amazon S3.

```
CREATE EXTERNAL TABLE planet_data (
  planet_name string,
  order_from_sun int,
  au_to_sun float,
  mass float,
  gravity_earth float,
  orbit_years float,
  day_length float
  )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
LOCATION 's3://DOC-EXAMPLE-BUCKET/tsv/'
```

Perhatikan bidang berikut:

- `ROW FORMAT DELIMITED`Klausa menunjukkan bahwa data dibatasi oleh karakter tertentu.
- `FIELDS TERMINATED BY '\t'`Klausa menentukan bahwa bidang dalam data TSV dipisahkan oleh karakter tab ('\ t').
- `STORED AS TEXTFILE`Klausul menunjukkan bahwa data disimpan sebagai file teks biasa di Amazon S3.

Untuk menanyakan data, Anda dapat menggunakan `SELECT` pernyataan sederhana seperti berikut:

```
SELECT * FROM planet_data
```

Untuk menggunakan contoh untuk membuat tabel TSV Anda sendiri di Athena, ganti nama tabel dan kolom dengan nama dan tipe data tabel dan kolom Anda sendiri, dan perbarui `LOCATION` klausa untuk menunjuk ke jalur Amazon S3 tempat file TSV Anda disimpan.

Untuk informasi selengkapnya tentang membuat tabel, lihat Membuat tabel di Athena.

CREATE TABLE AS

Menciptakan tabel baru diisi dengan hasilSELECTkueri. Untuk membuat tabel kosong, gunakanCREATE TABLE. `CREATE TABLE AS`menggabungkan pernyataan `CREATE TABLE` DDL dengan pernyataan `SELECT` DHTML dan oleh karena itu secara teknis berisi DDL dan DML.2. Perhatikan bahwa meskipun `CREATE TABLE AS` dikelompokkan di sini dengan pernyataan DDL lainnya, kueri CTAS di Athena diperlakukan sebagai DMLuntuk tujuan Service Quotas. Untuk informasi tentang Service Quotas di Athena, lihat. Service Quotas

> **Note**
>
> Untuk pernyataan CTAS, setelan pemilik bucket yang diharapkan tidak berlaku untuk lokasi tabel tujuan di Amazon S3. Setelan pemilik bucket yang diharapkan hanya berlaku untuk lokasi keluaran Amazon S3 yang Anda tentukan untuk hasil kueri Athena. Untuk informasi selengkapnya, lihat Menentukan lokasi hasil kueri menggunakan konsol Athena.

Untuk informasi tambahan tentang hal `CREATE TABLE AS` itu di luar cakupan topik referensi ini, lihatMembuat tabel dari hasil query (CTAS).

Topik

- Sinopsis
- Properti tabel CTAS
- Contoh

Sinopsis

```
CREATE TABLE table_name
```

```
[ WITH ( property_name = expression [, ...] ) ]
AS query
[ WITH [ NO ] DATA ]
```

Di mana:

Untuk (property_name = ekspresi [,...])

Daftar properti tabel CTAS opsional, beberapa di antaranya khusus untuk format penyimpanan data. Lihat Properti tabel CTAS.

kueri

ASELECTquery yang digunakan untuk membuat tabel baru.

> **Important**
>
> Jika Anda berencana untuk membuat kueri dengan partisi, menentukan nama kolom dipartisi terakhir dalam daftar kolom di`SELECT`.

[DENGAN [TIDAK] DATA]

Jika`WITH NO DATA`digunakan, tabel kosong baru dengan skema yang sama seperti tabel asli dibuat.

> **Note**
>
> Untuk menyertakan header kolom dalam output hasil kueri, Anda dapat menggunakan `SELECT` kueri sederhana alih-alih kueri CTAS. Anda dapat mengambil hasil dari lokasi hasil kueri atau mengunduh hasilnya secara langsung menggunakan konsol Athena. Untuk informasi selengkapnya, lihat Bekerja dengan hasil kueri, kueri terbaru, dan file keluaran.

Properti tabel CTAS

Setiap tabel CTAS di Athena memiliki daftar opsional CTAS properti tabel yang Anda tentukan menggunakan`WITH (property_name = expression [, ...] )`. Untuk informasi tentang menggunakan prosedur ini, lihat Contoh kueri CTAS.

WITH (property_name = expression [, ...],)

table_type = ['HIVE', 'ICEBERG']

Tidak wajib. Nilai default-nya `HIVE`. Menentukan jenis tabel tabel yang dihasilkan

Contoh:

```
WITH (table_type ='ICEBERG')
```

external_location = [location]

> **Note**
>
> Karena tabel Iceberg tidak eksternal, properti ini tidak berlaku untuk tabel Iceberg. Untuk menentukan lokasi root dari tabel Iceberg dalam pernyataan CTAS, gunakan `location` properti yang dijelaskan nanti di bagian ini.

Tidak wajib. Lokasi tempat Athena menyimpan kueri CTAS Anda di Amazon S3.

Contoh:

```
WITH (external_location ='s3://DOC-EXAMPLE-BUCKET/tables/parquet_table/')
```

Athena tidak menggunakan jalur yang sama untuk hasil kueri dua kali. Jika Anda menentukan lokasi secara manual, pastikan bahwa lokasi Amazon S3 yang Anda tentukan tidak memiliki data. Athena tidak pernah mencoba untuk menghapus data Anda. Jika Anda ingin menggunakan lokasi yang sama lagi, hapus data secara manual, atau kueri CTAS Anda akan gagal.

Jika Anda menjalankan permintaan CTAS yang menentukan`external_location`dalam grup kerja yang memberlakukan lokasi hasil kueri, permintaan gagal dengan pesan kesalahan. Untuk melihat lokasi hasil kueri yang ditentukan untuk grup kerja, lihat detail grup kerja.

Jika grup kerja Anda menimpa pengaturan sisi klien untuk permintaan hasil lokasi, Athena membuat tabel Anda di lokasi berikut:

```
s3://DOC-EXAMPLE-BUCKET/tables/query-id/
```

Jika Anda tidak menggunakan`external_location`properti untuk menentukan lokasi dan grup kerja Anda tidak menimpa pengaturan sisi klien, Athena menggunakanPengaturan sisi klienuntuk lokasi hasil kueri untuk membuat tabel Anda di lokasi berikut:

```
s3://DOC-EXAMPLE-BUCKET/Unsaved-or-query-name/year/month/date/tables/query-id/
```

`is_external = [boolean]`

Tidak wajib. Menunjukkan apakah tabel adalah tabel eksternal. Bawaannya adalah benar. Untuk tabel Iceberg, ini harus disetel ke false.

Contoh:

```
WITH (is_external = false)
```

`location = [location]`

Diperlukan untuk tabel Iceberg. Menentukan lokasi root untuk tabel Iceberg yang akan dibuat dari hasil query.

Contoh:

```
WITH (location ='s3://DOC-EXAMPLE-BUCKET/tables/iceberg_table/')
```

`field_delimiter = [delimiter]`

Opsional dan spesifik untuk format penyimpanan data berbasis teks. Pembatas bidang karakter tunggal untuk file dalam CSV, TSV, dan file teks. Misalnya, `WITH (field_delimiter = ',')`. Saat ini, pembatas bidang multikarakter tidak didukung untuk permintaan CTAS. Jika Anda tidak menentukan pembatas bidang,`\001`digunakan secara default.

`format = [storage_format]`

Format penyimpanan untuk hasil query CTAS, seperti,`ORC`,`PARQUET`,`AVRO`, `JSONION`, atau`TEXTFILE`. Untuk tabel Iceberg, format yang diizinkan adalah`ORC`,`PARQUET`, dan. `AVRO` Jika dihilangkan,`PARQUET`digunakan secara default. Nama parameter ini,`format`, harus terdaftar dalam huruf kecil, atau permintaan CTAS Anda akan gagal.

Contoh:

```
WITH (format = 'PARQUET')
```

bucketed_by = ARRAY[column_name[,…], bucket_count = [int]]

> **Note**
>
> Properti ini tidak berlaku untuk tabel Iceberg. Untuk tabel Iceberg, gunakan partisi dengan transformasi bucket.

Daftar larik bucket untuk bucket data. Jika dihilangkan, Athena tidak bucket data Anda dalam kueri ini.

bucket_count = [int]

> **Note**
>
> Properti ini tidak berlaku untuk tabel Iceberg. Untuk tabel Iceberg, gunakan partisi dengan transformasi bucket.

Jumlah bucket untuk bucket data Anda. Jika dihilangkan, Athena tidak bucket data Anda. Contoh:

```
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**
>
> Properti ini tidak berlaku untuk tabel Iceberg. Untuk menggunakan transformasi partisi untuk tabel Iceberg, gunakan `partitioning` properti yang dijelaskan nanti di bagian ini.

Tidak wajib. Daftar larik kolom dimana tabel CTAS akan dipartisi. Verifikasi bahwa nama-nama kolom dipartisi terdaftar terakhir dalam daftar kolom di`SELECT`.

partitioning = ARRAY[partition_transform, ...]

Tidak wajib. Menentukan partisi dari tabel Iceberg yang akan dibuat. Iceberg mendukung berbagai macam transformasi partisi dan evolusi partisi. Transformasi partisi dirangkum dalam tabel berikut.

Transformasi	Deskripsi
`year(ts)`	Membuat partisi untuk setiap tahun. Nilai partisi adalah perbedaan bilangan bulat dalam tahun antara `ts` dan 1 Januari 1970.
`month(ts)`	Membuat partisi untuk setiap bulan setiap tahun. Nilai partisi adalah perbedaan bilangan bulat dalam bulan antara `ts` dan 1 Januari 1970.
`day(ts)`	Membuat partisi untuk setiap hari setiap tahun. Nilai partisi adalah perbedaan bilangan bulat dalam hari antara `ts` dan 1 Januari 1970.
`hour(ts)`	Membuat partisi untuk setiap jam setiap hari. Nilai partisi adalah stempel waktu dengan menit dan detik diatur ke nol.
`bucket(x, nbuckets)`	Hash data ke dalam jumlah ember yang ditentukan. Nilai partisi adalah hash integer darix, dengan nilai antara 0 dannbuckets - 1, inklusif.
`truncate(s, nchars)`	Membuat nilai partisi `nchars` karakter pertama daris.

Contoh:

```
WITH (partitioning = ARRAY['month(order_date)',
                           'bucket(account_number, 10)',
                           'country']))
```

optimize_rewrite_min_data_file_size_bytes = [long]

Tidak wajib. Konfigurasi spesifik pengoptimalan data. File yang lebih kecil dari nilai yang ditentukan disertakan untuk optimasi. Defaultnya adalah 0,75 kali nilai. `write_target_data_file_size_bytes` Properti ini hanya berlaku untuk tabel Iceberg. Untuk informasi selengkapnya, lihat Mengoptimalkan tabel Iceberg.

Contoh:

```
WITH (optimize_rewrite_min_data_file_size_bytes = 402653184)
```

optimize_rewrite_max_data_file_size_bytes = [long]

Tidak wajib. Konfigurasi spesifik pengoptimalan data. File yang lebih besar dari nilai yang ditentukan disertakan untuk optimasi. Defaultnya adalah 1,8 kali nilai`write_target_data_file_size_bytes`. Properti ini hanya berlaku untuk tabel Iceberg. Untuk informasi selengkapnya, lihat Mengoptimalkan tabel Iceberg.

Contoh:

```
WITH (optimize_rewrite_max_data_file_size_bytes = 966367641)
```

optimize_rewrite_data_file_threshold = [int]

Tidak wajib. Konfigurasi spesifik pengoptimalan data. Jika ada lebih sedikit file data yang memerlukan pengoptimalan daripada ambang batas yang diberikan, file tidak ditulis ulang. Hal ini memungkinkan akumulasi lebih banyak file data untuk menghasilkan file yang lebih dekat dengan ukuran target dan melewatkan perhitungan yang tidak perlu untuk penghematan biaya. Default-nya adalah 5. Properti ini hanya berlaku untuk tabel Iceberg. Untuk informasi selengkapnya, lihat Mengoptimalkan tabel Iceberg.

Contoh:

```
WITH (optimize_rewrite_data_file_threshold = 5)
```

`optimize_rewrite_delete_file_threshold = [int]`

Tidak wajib. Konfigurasi spesifik pengoptimalan data. Jika ada lebih sedikit file hapus yang terkait dengan file data daripada ambang batas, file data tidak ditulis ulang. Ini memungkinkan akumulasi lebih banyak file hapus untuk setiap file data untuk penghematan biaya. Defaultnya adalah 2. Properti ini hanya berlaku untuk tabel Iceberg. Untuk informasi selengkapnya, lihat Mengoptimalkan tabel Iceberg.

Contoh:

```
WITH (optimize_rewrite_delete_file_threshold = 2)
```

`vacuum_min_snapshots_to_keep = [int]`

Tidak wajib. Konfigurasi khusus vakum. Jumlah minimum snapshot terbaru yang harus dipertahankan. Default-nya adalah 1. Properti ini hanya berlaku untuk tabel Iceberg. Untuk informasi selengkapnya, lihat VAKUM.

> **Note**
>
> `vacuum_min_snapshots_to_keep`Properti ini membutuhkan mesin Athena versi 3.

Contoh:

```
WITH (vacuum_min_snapshots_to_keep = 1)
```

`vacuum_max_snapshot_age_seconds = [long]`

Tidak wajib. Konfigurasi khusus vakum. Periode dalam detik yang mewakili usia snapshot untuk dipertahankan. Defaultnya adalah 432000 (5 hari). Properti ini hanya berlaku untuk tabel Iceberg. Untuk informasi selengkapnya, lihat VAKUM.

> **Note**
>
> `vacuum_max_snapshot_age_seconds`Properti ini membutuhkan mesin Athena versi 3.

Contoh:

```
WITH (vacuum_max_snapshot_age_seconds = 432000)
```

`write_compression = [compression_format]`

Jenis kompresi yang digunakan untuk format penyimpanan apa pun yang memungkinkan kompresi ditentukan. `compression_format`Nilai menentukan kompresi yang akan digunakan ketika data ditulis ke tabel. Anda dapat menentukan kompresi untuk`TEXTFILE`,`JSON`,`PARQUET`, dan format `ORC` file.

Misalnya, jika `format` properti menentukan `PARQUET` sebagai format penyimpanan, nilai untuk `write_compression` menentukan format kompresi untuk Parket. Dalam hal ini, menentukan nilai untuk `write_compression` setara dengan menentukan nilai untuk. `parquet_compression`

Demikian pula, jika `format` properti menentukan `ORC` sebagai format penyimpanan, nilai untuk `write_compression` menentukan format kompresi untuk ORC. Dalam hal ini, menentukan nilai untuk `write_compression` setara dengan menentukan nilai untuk. `orc_compression`

Beberapa properti tabel format kompresi tidak dapat ditentukan dalam kueri CTAS yang sama. Misalnya, Anda tidak dapat menentukan keduanya `write_compression` dan `parquet_compression` dalam kueri yang sama. Hal yang sama berlaku untuk `write_compression` dan`orc_compression`. Untuk informasi tentang jenis kompresi yang didukung untuk setiap format file, lihatDukungan kompresi Athena.

`orc_compression = [compression_format]`

Jenis kompresi yang digunakan untuk format `ORC` file saat `ORC` data ditulis ke tabel. Misalnya, `WITH (orc_compression = 'ZLIB')`. Potongan dalam `ORC` file (kecuali `ORC` Postscript) dikompresi menggunakan kompresi yang Anda tentukan. Jika dihilangkan, kompresi ZLIB digunakan secara default untuk. `ORC`

> **Note**
>
> Untuk konsistensi, kami sarankan Anda menggunakan `write_compression` properti alih-alih`orc_compression`. Gunakan `format` properti untuk menentukan format penyimpanan sebagai`ORC`, dan kemudian gunakan `write_compression` properti untuk menentukan format kompresi yang `ORC` akan digunakan.

`parquet_compression = [compression_format]`

Jenis kompresi yang digunakan untuk format file Parket saat data Parket ditulis ke tabel. Misalnya, `WITH (parquet_compression = 'SNAPPY')`. Kompresi ini diterapkan pada potongan kolom dalam file Parket. Jika dihilangkan, kompresi GZIP digunakan secara default untuk Parket.

> **Note**
>
> Untuk konsistensi, kami sarankan Anda menggunakan `write_compression` properti alih-alih`parquet_compression`. Gunakan `format` properti untuk menentukan format penyimpanan sebagai`PARQUET`, dan kemudian gunakan `write_compression` properti untuk menentukan format kompresi yang `PARQUET` akan digunakan.

`compression_level = [compression_level]`

Tingkat kompresi yang digunakan. Properti ini hanya berlaku untuk kompresi ZSTD. Nilai yang mungkin adalah dari 1 hingga 22. Nilai default-nya adalah 3. Untuk informasi selengkapnya, lihat Menggunakan tingkat kompresi ZSTD di Athena.

Contoh

Untuk contoh pertanyaan CTAS, konsultasikan sumber daya berikut.

- Contoh kueri CTAS
- Menggunakan CTAS dan INSERT INTO untuk ETL dan analisis data
- Gunakan pernyataan CTAS dengan Amazon Athena untuk mengurangi biaya dan meningkatkan kinerja
- Menggunakan CTAS dan INSERT INTO untuk bekerja di sekitar batas partisi 100

CREATE VIEW

Menciptakan tampilan baru dari yang ditentukan`SELECT`kueri. Tampilan adalah tabel logis yang dapat direferensikan oleh kueri masa depan. Tampilan tidak mengandung data apapun dan tidak menulis data. Sebaliknya, kueri yang ditentukan oleh tampilan berjalan setiap kali Anda referensi tampilan oleh kueri lain.

> **Note**
>
> Topik ini menyediakan informasi ringkasan untuk referensi. Untuk informasi lebih rinci tentang menggunakan tampilan di Athena, lihatBekerja dengan pandangan. Untuk informasi tentang batasan tampilan, lihatBatasan untuk tampilan.

Sinopsis

```
CREATE [ OR REPLACE ] VIEW view_name AS query
```

Opsional`OR REPLACE`klausa memungkinkan Anda memperbarui tampilan yang ada dengan menggantinya. Untuk informasi selengkapnya, lihat Membuat tampilan.

Contoh

Untuk membuat tampilan`test`dari tabel`orders`Gunakan kueri yang serupa dengan yang berikut:

```
CREATE VIEW test AS
SELECT
orderkey,
orderstatus,
totalprice / 2 AS half
FROM orders;
```

Untuk membuat tampilan`orders_by_date`dari tabel`orders`Gunakan kueri berikut:

```
CREATE VIEW orders_by_date AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate;
```

Untuk memperbarui tampilan yang ada, gunakan contoh yang serupa dengan yang berikut:

```
CREATE OR REPLACE VIEW test AS
SELECT orderkey, orderstatus, totalprice / 4 AS quarter
FROM orders;
```

Lihat juga SHOW COLUMNS, SHOW CREATE VIEW, DESCRIBE VIEW, dan DROP VIEW.

DESCRIBE

Menampilkan satu atau lebih kolom, termasuk kolom partisi, untuk tabel yang ditentukan. Perintah ini berguna untuk memeriksa atribut kolom kompleks.

Sinopsis

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

> **Important**
>
> Sintaks untuk pernyataan ini adalah`DESCRIBE table_name`, tidak`DESCRIBE TABLE table_name`. Menggunakan sintaks yang terakhir menghasilkan pesan kesalahan GAGAL: SemanticException [Kesalahan 10001]: Tabel tidak ditemukan tabel.

Parameter

[DIPERPANJANG | DIFORMAT]

Menentukan format output. Menghilangkan parameter ini menunjukkan nama kolom dan tipe data yang sesuai, termasuk kolom partisi, dalam format tabel. Menentukan `FORMATTED` tidak hanya menampilkan nama kolom dan tipe data dalam format tabel, tetapi juga informasi tabel dan penyimpanan terperinci. `EXTENDED`menampilkan informasi kolom dan tipe data dalam format tabel, dan metadata rinci untuk tabel dalam bentuk serial Thrift. Format ini kurang mudah dibaca dan berguna terutama untuk debugging.

[Partisi partition_spec]

Jika disertakan, daftar metadata untuk partisi yang ditentukan oleh`partition_spec`, tempat`partition_spec`dalam format`(partition_column = partition_col_value, partition_column = partition_col_value, ...)`.

[col_name ([.field_name] | [. ' $ elem$ '] | [.' $ kunci$ '] | [.' $ nilai$ ']) *]

Menentukan kolom dan atribut untuk memeriksa. Anda dapat menentukan`.field_name`untuk elemen struct,`'$elem$'`untuk elemen larik,`'$key$'`untuk kunci peta, dan`'$value$'`untuk nilai peta. Anda dapat menentukan ini rekursif untuk lebih mengeksplorasi kolom kompleks.

Contoh

```
DESCRIBE orders
```

```
DESCRIBE FORMATTED mydatabase.mytable PARTITION (part_col = 100) columnA;
```

Kueri dan output berikut menunjukkan kolom dan informasi tipe data dari `impressions` tabel berdasarkan data sampel Amazon EMR.

```
DESCRIBE impressions
```

```
requestbegintime          string                                          from
 deserializer
adid                      string                                          from
 deserializer
impressionid              string                                          from
 deserializer
referrer                  string                                          from
 deserializer
useragent                 string                                          from
 deserializer
usercookie                string                                          from
 deserializer
ip                        string                                          from
 deserializer
number                    string                                          from
 deserializer
processid                 string                                          from
 deserializer
browsercokie              string                                          from
 deserializer
requestendtime            string                                          from
 deserializer
timers                    struct<modellookup:string,requesttime:string>  from
 deserializer
threadid                  string                                          from
 deserializer
hostname                  string                                          from
 deserializer
sessionid                 string                                          from
 deserializer
```

```
dt                      string

# Partition Information
# col_name              data_type               comment

dt                      string
```

Contoh query berikut dan output menunjukkan hasil untuk tabel yang sama ketika `FORMATTED` opsi digunakan.

```
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
```

Contoh query berikut dan output menunjukkan hasil untuk tabel yang sama ketika `EXTENDED` opsi digunakan. Informasi tabel rinci adalah output pada satu baris, tetapi diformat di sini untuk keterbacaan.

```
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:browsercookie, 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

Menunjukkan daftar kolom untuk tampilan bernama. Ini memungkinkan Anda untuk memeriksa atribut dari tampilan yang kompleks.

Sinopsis

```
DESCRIBE [db_name.]view_name
```

Contoh

```
DESCRIBE orders;
```

Lihat juga SHOW COLUMNS, SHOW CREATE VIEW, SHOW VIEWS, dan DROP VIEW.

DROP DATABASE

Menghapus basis data bernama dari katalog. Jika basis data berisi tabel, Anda harus baik drop tabel sebelum menjalankanDROP DATABASEatau menggunakanCASCADEKlausul. PenggunaanDATABASEdanSCHEMAdapat dipertukarkan. Mereka berarti hal yang sama.

Sinopsis

```
DROP {DATABASE | SCHEMA} [IF EXISTS] database_name [RESTRICT | CASCADE]
```

Parameter

[JIKA ADA]

Menyebabkan kesalahan yang akan ditekan jikadatabase_nametidak ada.

[MEMBATAS|KASKADE]

Menentukan bagaimana tabel dalamdatabase_namedianggap selamaDROPOperasi. Jika Anda menentukanRESTRICT, basis data tidak dijatuhkan jika berisi tabel. Ini adalah perilaku default. MenentukanCASCADEmenyebabkan basis data dan semua tabel yang akan dijatuhkan.

Contoh

```
DROP DATABASE clickstreams;
```

```
DROP SCHEMA IF EXISTS clickstreams CASCADE;
```

> **Note**
>
> Ketika Anda mencoba untuk menjatuhkan database yang namanya memiliki karakter khusus (misalnya,my-database), Anda mungkin menerima pesan kesalahan. Untuk mengatasi masalah ini, coba lampirkan nama database di karakter centang belakang (`). Untuk informasi tentang penamaan database di Athena, lihat. Nama untuk tabel, database, dan kolom

DROP TABLE

Menghapus definisi tabel metadata untuk tabel bernamatable_name. Saat Anda menjatuhkan tabel eksternal, data yang mendasarinya tetap utuh.

Sinopsis

```
DROP TABLE [IF EXISTS] table_name
```

Parameter

[JIKA ADA]

Menyebabkan kesalahan yang akan ditekan jika`table_name`tidak ada.

Contoh

```
DROP TABLE fulfilled_orders
```

```
DROP TABLE IF EXISTS fulfilled_orders
```

Jika menggunakan editor permintaan konsol Athena untuk menjatuhkan tabel yang memiliki karakter khusus selain garis bawah (_), menggunakan backticks, seperti dalam contoh berikut.

```
DROP TABLE `my-athena-database-01.my-athena-table`
```

Jika menggunakan konektor JDBC untuk menjatuhkan tabel yang memiliki karakter khusus, karakter backtick tidak diperlukan.

```
DROP TABLE my-athena-database-01.my-athena-table
```

DROP VIEW

Drops (menghapus) tampilan yang ada. Opsional`IF EXISTS`klausa menyebabkan kesalahan yang akan ditekan jika tampilan tidak ada.

Untuk informasi selengkapnya, lihat Bekerja dengan pandangan.

Sinopsis

```
DROP VIEW [ IF EXISTS ] view_name
```

Contoh

```
DROP VIEW orders_by_date
```

```
DROP VIEW IF EXISTS orders_by_date
```

Lihat juga SHOW COLUMNS, SHOW CREATE VIEW, CREATE VIEW, SHOW VIEWS, dan DESCRIBE VIEW.

MSCK REPAIR TABLE

Gunakan`MSCK REPAIR TABLE`perintah untuk memperbarui metadata dalam katalog setelah Anda menambahkan partisi kompatibel Hive.

Parameter`MSCK REPAIR TABLE`perintah memindai sistem file seperti Amazon S3 untuk partisi kompatibel Hive yang ditambahkan ke sistem file setelah tabel dibuat.`MSCK REPAIR TABLE`membandingkan partisi dalam metadata tabel dan partisi di S3. Jika partisi baru hadir di lokasi S3 yang Anda tentukan saat Anda membuat tabel, itu menambahkan partisi tersebut metadata dan tabel Athena.

Saat Anda menambahkan partisi fisik, metadata dalam katalog menjadi tidak konsisten dengan tata letak data dalam sistem file, dan informasi tentang partisi baru perlu ditambahkan ke katalog. Untuk memperbarui metadata, jalankan`MSCK REPAIR TABLE`sehingga Anda dapat mengkueri data di partisi baru dari Athena.

> **Note**
>
> `MSCK REPAIR TABLE`hanya menambahkan partisi ke metadata; itu tidak menghapusnya. Untuk menghapus partisi dari metadata setelah partisi dihapus secara manual di Amazon S3, jalankan perintah`ALTER TABLE` *`table-name`* `DROP PARTITION`. Untuk informasi selengkapnya, lihat ALTER TABLE DROP PARTITION.

Pertimbangan dan batasan

Saat menggunakan`MSCK REPAIR TABLE`, ingatlah poin-poin berikut:

- Ini dimungkinkan akan memakan waktu untuk menambahkan semua partisi. Jika operasi ini kali keluar, itu akan berada dalam keadaan tidak lengkap tempat hanya beberapa partisi ditambahkan

ke katalog. Kau harus lari`MSCK REPAIR TABLE`pada tabel yang sama sampai semua partisi ditambahkan. Untuk informasi selengkapnya, lihat Partisi data di Athena.

- Untuk partisi yang tidak kompatibel dengan Hive, gunakanALTER TABLE ADD PARTITIONuntuk memuat partisi sehingga Anda dapat mengkueri data mereka.
- Lokasi partisi yang akan digunakan dengan Athena harus menggunakan`s3`protokol (misalnya,`s3://DOC-EXAMPLE-BUCKET/folder/`). Di Athena, lokasi yang menggunakan protokol lain (contohnya,`s3a://bucket/folder/`) akan mengakibatkan kegagalan permintaan ketika`MSCK REPAIR TABLE`query dijalankan pada tabel yang mengandung.
- Karena `MSCK REPAIR TABLE` memindai folder dan subfoldernya untuk menemukan skema partisi yang cocok, pastikan untuk menyimpan data untuk tabel terpisah dalam hierarki folder terpisah. Misalnya, Anda memiliki data untuk tabel 1 in `s3://DOC-EXAMPLE-BUCKET1` dan data untuk tabel 2 in`s3://DOC-EXAMPLE-BUCKET1/table-2-data`. Jika kedua tabel dipartisi oleh string, `MSCK REPAIR TABLE` akan menambahkan partisi untuk tabel 2 ke tabel 1. Untuk menghindari hal ini, gunakan struktur folder terpisah seperti `s3://DOC-EXAMPLE-BUCKET1` dan `s3://DOC-EXAMPLE-BUCKET2` sebagai gantinya. Perhatikan bahwa perilaku ini konsisten dengan Amazon EMR dan Apache Hive.
- Karena masalah yang diketahui, `MSCK REPAIR TABLE` gagal secara diam-diam ketika nilai partisi berisi karakter titik dua (`:`) (misalnya, ketika nilai partisi adalah stempel waktu). Sebagai solusinya, gunakan. ALTER TABLE ADD PARTITION
- `MSCK REPAIR TABLE`tidak menambahkan nama kolom partisi yang dimulai dengan garis bawah (_). Untuk mengatasi batasan ini, gunakanALTER TABLE ADD PARTITION.

Sinopsis

```
MSCK REPAIR TABLE table_name
```

Contoh

```
MSCK REPAIR TABLE orders;
```

Pemecahan Masalah

Setelah Anda menjalankan`MSCK REPAIR TABLE`, jika Athena tidak menambahkan partisi ke tabel di AWS Glue Data Catalog, periksa yang berikut ini:

- AWS Glue akses — Pastikan bahwa peran AWS Identity and Access Management (IAM) memiliki kebijakan yang memungkinkan `glue:BatchCreatePartition` tindakan. Untuk informasi selengkapnya, lihat Izinkan lem: BatchCreatePartition dalam kebijakan IAM nanti dalam dokumen ini.
- Akses Amazon S3 — Pastikan peran tersebut memiliki kebijakan dengan izin yang memadai untuk mengakses Amazon S3, termasuk tindakannya. `s3:DescribeJob` Untuk contoh tindakan Amazon S3 yang memungkinkan, lihat kebijakan bucket contoh di Akses lintas akun di Athena ke ember Amazon S3.
- Casing kunci objek Amazon S3 — Pastikan jalur Amazon S3 dalam huruf kecil, bukan kotak unta (misalnya`userid`, bukan`userId`), atau `ALTER TABLE ADD PARTITION` gunakan untuk menentukan nama kunci objek. Untuk informasi selengkapnya, lihat Ubah atau definisikan ulang jalur Amazon S3 nanti dalam dokumen ini.
- Waktu habis kueri—`MSCK REPAIR TABLE`paling baik digunakan saat membuat tabel untuk pertama kalinya atau saat ada ketidakpastian tentang paritas antara data dan metadata partisi. Jika Anda menggunakan`MSCK REPAIR TABLE`untuk sering menambahkan partisi baru (misalnya, setiap hari) dan mengalami timeout kueri, pertimbangkan untuk menggunakanALTER TABLE ADD PARTITION.
- Partisi hilang dari sistem file - Jika Anda menghapus partisi secara manual di Amazon S3 dan kemudian `MSCK REPAIR TABLE` menjalankannya, Anda mungkin menerima pesan kesalahan Partisi hilang dari sistem file. Ini terjadi karena`MSCK REPAIR TABLE`tidak menghapus partisi basi dari metadata tabel. Untuk menghapus partisi yang dihapus dari tabel metadata, jalankanALTER TABLE DROP PARTITIONSebagai gantinya Perhatikan bahwaTAMPILKAN PARTISIjuga hanya mencantumkan partisi dalam metadata, bukan partisi dalam sistem file.
- Kesalahan "NullPointerException name is null”

  Jika Anda menggunakan operasi AWS Glue CreateTableAPI atau AWS CloudFormation `AWS::Glue::Table`template untuk membuat tabel untuk digunakan di Athena tanpa menentukan `TableType` properti dan kemudian menjalankan kueri DDL seperti `SHOW CREATE TABLE` atau`MSCK REPAIR TABLE`, Anda dapat menerima pesan kesalahan GAGAL: NullPointerException Nama adalah null.

  Untuk mengatasi kesalahan, tentukan nilai TableInput`TableType`eatribut sebagai bagian dari panggilan AWS Glue `CreateTable` API atau AWS CloudFormation templat. Nilai yang mungkin untuk `TableType` include `EXTERNAL_TABLE` atau`VIRTUAL_VIEW`.

Persyaratan ini hanya berlaku ketika Anda membuat tabel menggunakan operasi AWS Glue `CreateTable` API atau `AWS::Glue::Table` template. Jika Anda membuat tabel untuk Athena menggunakan pernyataan DDL atau AWS Glue crawler, `TableType` properti didefinisikan untuk Anda secara otomatis.

Bagian berikut memberikan informasi tambahan.

Izinkan lem: BatchCreatePartition dalam kebijakan IAM

Tinjau kebijakan IAM yang dilampirkan pada peran yang Anda gunakan untuk menjalankan`MSCK REPAIR TABLE`. Saat Anda menggunakan AWS Glue Data Catalog dengan Athena, kebijakan IAM harus mengizinkan tindakan tersebut. `glue:BatchCreatePartition` Untuk contoh kebijakan IAM yang memungkinkan`glue:BatchCreatePartition`tindakan, lihatAWS kebijakan terkelola: AmazonAthenaFullAccess.

Ubah atau definisikan ulang jalur Amazon S3

Jika satu atau beberapa kunci objek di jalur Amazon S3 berada dalam huruf unta, bukan huruf kecil, `MSCK REPAIR TABLE` mungkin tidak menambahkan partisi ke file. AWS Glue Data Catalog Misalnya, jika jalur Amazon S3 Anda menyertakan nama kunci objek`userId`, partisi berikut mungkin tidak ditambahkan ke: AWS Glue Data Catalog

```
s3://DOC-EXAMPLE-BUCKET/path/userId=1/

s3://DOC-EXAMPLE-BUCKET/path/userId=2/

s3://DOC-EXAMPLE-BUCKET/path/userId=3/
```

Untuk mengatasi masalah ini, lakukan salah satu hal berikut:

- Gunakan huruf kecil alih-alih casing unta saat Anda membuat kunci objek Amazon S3:

  ```
  s3://DOC-EXAMPLE-BUCKET/path/userid=1/

  s3://DOC-EXAMPLE-BUCKET/path/userid=2/

  s3://DOC-EXAMPLE-BUCKET/path/userid=3/
  ```

- Gunakan ALTER TABLE ADD PARTITION untuk mendefinisikan ulang lokasi, seperti pada contoh berikut:

```
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/'
```

Perhatikan bahwa meskipun nama kunci objek Amazon S3 dapat menggunakan huruf besar, nama bucket Amazon S3 itu sendiri harus selalu dalam huruf kecil. Untuk informasi selengkapnya, lihat Pedoman penamaan kunci objek dan aturan penamaan Bucket di Panduan Pengguna Amazon S3.

SHOW COLUMNS

Hanya menampilkan nama kolom untuk satu tabel atau tampilan tertentu. Untuk mendapatkan informasi yang lebih rinci, tanyakan AWS Glue Data Catalog sebagai gantinya. Untuk informasi dan contoh, lihat bagian Mengkueri AWS Glue Data Catalog topik berikut:

- Untuk melihat metadata kolom seperti tipe data, lihat. Daftar atau mencari kolom untuk tabel atau tampilan tertentu
- Untuk melihat semua kolom untuk semua tabel dalam database tertentu`AwsDataCatalog`, lihatDaftar atau mencari kolom untuk tabel atau tampilan tertentu.
- Untuk melihat semua kolom untuk semua tabel di semua database`AwsDataCatalog`, lihatDaftar semua kolom untuk semua tabel.
- Untuk melihat kolom yang memiliki kesamaan tabel tertentu dalam database, lihatDaftar kolom yang memiliki kesamaan tabel tertentu.

Sinopsis

```
SHOW COLUMNS {FROM|IN} database_name.table_name
```

```
SHOW COLUMNS {FROM|IN} table_name [{FROM|IN} database_name]
```

Parameter`FROM`dan`IN`kata kunci dapat digunakan secara bergantian. Jika *`table_name`* atau *`database_name`* memiliki karakter khusus seperti tanda hubung, kelilingi nama dengan tanda kutip belakang (misalnya,). `` `my-database`.`my-table` `` Jangan

mengelilingi*table_name*atau*database_name*dengan tanda kutip tunggal atau ganda. Saat ini, penggunaan`LIKE`dan ekspresi pencocokan pola tidak didukung.

Contoh

Contoh setara berikut menunjukkan kolom dari`orders`tabel di`customers`basis data. Dua contoh pertama berasumsi bahwa`customers`adalah basis data saat ini.

```
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

Menganalisis tabel yang ada bernama`table_name`untuk menghasilkan kueri yang menciptakannya.

Sinopsis

```
SHOW CREATE TABLE [db_name.]table_name
```

Parameter

TABLE [db_name.] table_name

Parameter db_name bersifat opsional. Jika dihilangkan, default konteks untuk basis data saat ini.

> **Note**
>
> Nama tabel diperlukan.

Contoh

```
SHOW CREATE TABLE orderclickstoday;
```

```
SHOW CREATE TABLE `salesdata.orderclickstoday`;
```

Pemecahan Masalah

Jika Anda menggunakan operasi AWS Glue CreateTableAPI atau AWS CloudFormation `AWS::Glue::Table`template untuk membuat tabel untuk digunakan di Athena tanpa menentukan `TableType` properti dan kemudian menjalankan kueri DDL seperti `SHOW CREATE TABLE` atau`MSCK REPAIR TABLE`, Anda dapat menerima pesan kesalahan GAGAL: NullPointerException Nama adalah null.

Untuk mengatasi kesalahan, tentukan nilai TableInput`TableType`eatribut sebagai bagian dari panggilan AWS Glue `CreateTable` API atau AWS CloudFormation templat. Nilai yang mungkin untuk `TableType` include `EXTERNAL_TABLE` atau`VIRTUAL_VIEW`.

Persyaratan ini hanya berlaku ketika Anda membuat tabel menggunakan operasi AWS Glue `CreateTable` API atau `AWS::Glue::Table` template. Jika Anda membuat tabel untuk Athena menggunakan pernyataan DDL atau AWS Glue crawler, `TableType` properti didefinisikan untuk Anda secara otomatis.

SHOW CREATE VIEW

Menunjukkan pernyataan SQL yang menciptakan tampilan tertentu.

Sinopsis

```
SHOW CREATE VIEW view_name
```

Contoh

```
SHOW CREATE VIEW orders_by_date
```

Lihat juga CREATE VIEW dan DROP VIEW.

SHOW DATABASES

Daftar semua basis data didefinisikan dalam metastore tersebut. Anda dapat menggunakan `DATABASES` atau `SCHEMAS`. Mereka berarti hal yang sama.

Setara program dari `SHOW DATABASES` adalah aksi API ListDatabasesAthena. Metode yang setara AWS SDK for Python (Boto3) adalah list_databases.

Sinopsis

```
SHOW {DATABASES | SCHEMAS} [LIKE 'regular_expression']
```

Parameter

[SEPERTI '***regular_expression***']

Menyaring daftar basis data untuk orang-orang yang cocok*regular_expression*yang Anda tentukan. Untuk pencocokan karakter wildcard, Anda dapat menggunakan kombinasi `.*`, yang cocok dengan karakter nol sampai waktu tak terbatas.

Contoh

```
SHOW SCHEMAS;
```

```
SHOW DATABASES LIKE '.*analytics';
```

SHOW PARTITIONS

Daftar semua partisi dalam tabel Athena dalam urutan unsorted.

Sinopsis

```
SHOW PARTITIONS table_name
```

- Untuk menampilkan partisi dalam tabel dan daftar mereka dalam urutan tertentu, lihatDaftar partisi untuk tabel tertentuBagian pada bagianMengkueri AWS Glue Data Cataloghalaman.
- Untuk melihat isi partisi, lihatKueri dataBagian pada bagianPartisi data di Athenahalaman.

- `SHOW PARTITIONS`tidak mencantumkan partisi yang diproyeksikan oleh Athena tetapi tidak terdaftar dalam katalog. AWS Glue Untuk informasi tentang proyeksi partisi, lihatProyeksi partisi dengan Amazon Athena.
- `SHOW PARTITIONS`daftar partisi dalam metadata, bukan partisi dalam sistem file yang sebenarnya. Untuk memperbarui metadata setelah Anda menghapus partisi secara manual di Amazon S3, jalankanALTER TABLE DROP PARTITION.

Contoh

Contoh kueri berikut menunjukkan partisi untuk`flight_delays_csv`tabel, yang menunjukkan tabel penerbangan data dari Departemen Perhubungan AS. Untuk informasi selengkapnya tentang tabel `flight_delays_csv` yang digunakan dalam contoh ini, lihat LazySimpleSerDe untuk CSV, TSV, dan file yang dibatasi khusus. Tabel dipartisi berdasarkan tahun.

```
SHOW PARTITIONS flight_delays_csv
```

Hasil

```
year=2007
year=2015
year=1999
year=1993
year=1991
year=2003
year=1996
year=2014
year=2004
year=2011
...
```

Contoh kueri berikut menunjukkan partisi untuk`impressions`tabel, yang berisi contoh web browsing data. Untuk informasi selengkapnya tentang tabel `impressions` yang digunakan dalam contoh ini, lihat Partisi data di Athena. Tabel dipartisi oleh`dt`(datetime) kolom.

```
SHOW PARTITIONS impressions
```

Hasil

```
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
...
```

Daftar partisi dalam urutan yang diurutkan

Untuk mengurutkan partisi dalam daftar hasil, gunakan `SELECT` sintaks berikut sebagai ganti. `SHOW PARTITIONS`

```
SELECT * FROM database_name."table_name$partitions" ORDER BY column_name
```

Kueri berikut menunjukkan daftar partisi untuk`flight_delays_csv`contoh, tetapi dalam urutan diurutkan.

```
SELECT * FROM "flight_delays_csv$partitions" ORDER BY year
```

Hasil

```
year
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
...
```

Untuk informasi selengkapnya, lihat bagian Daftar partisi untuk tabel tertentu di halaman Mengkueri AWS Glue Data Catalog.

SHOW TABLES

Daftar semua tabel dasar dan tampilan dalam basis data.

Sinopsis

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

Parameter

[DI database_name]

Menentukan`database_name`dari mana tabel akan terdaftar. Jika dihilangkan, basis data dari konteks saat ini diasumsikan.

> **Note**
>
> `SHOW TABLES`mungkin gagal jika `database_name` menggunakan karakter yang tidak didukung seperti tanda hubung. Sebagai solusinya, coba lampirkan nama database di backticks.

['regular_expression']

Menyaring daftar tabel untuk orang-orang yang cocok`regular_expression`Anda tentukan. Untuk menunjukkan karakter apa pun dalam `AWSDataCatalog` tabel, Anda dapat menggunakan ekspresi `.*` wildcard `*` atau. Untuk database Apache Hive, gunakan ekspresi wildcard. `.*` Untuk menunjukkan pilihan antar karakter, gunakan `|` karakter.

Contoh

Example — Tampilkan semua tabel dalam database **`sampledb`**

```
SHOW TABLES IN sampledb
```

`Results`

```
alb_logs
```

```
cloudfront_logs
elb_logs
flights_2016
flights_parquet
view_2016_flights_dfw
```

Example — Tampilkan nama semua tabel **`sampledb`** yang menyertakan kata “penerbangan”

```
SHOW TABLES IN sampledb '*flights*'
```

`Results`

```
flights_2016
flights_parquet
view_2016_flights_dfw
```

Example — Tampilkan nama semua tabel di ujung **`sampledb`** itu di kata “log”

```
SHOW TABLES IN sampledb '*logs'
```

`Results`

```
alb_logs
cloudfront_logs
elb_logs
```

SHOW TBLPROPERTIES

Daftar properti tabel untuk tabel bernama.

Sinopsis

```
SHOW TBLPROPERTIES table_name [('property_name')]
```

Parameter

[('property_name')]

Jika disertakan, hanya nilai properti bernama`property_name`terdaftar.

Contoh

```
SHOW TBLPROPERTIES orders;
```

```
SHOW TBLPROPERTIES orders('comment');
```

SHOW VIEWS

Daftar tampilan dalam basis data tertentu, atau dalam basis data saat ini jika Anda menghilangkan nama basis data. Gunakan pilihan`LIKE`klausul dengan ekspresi reguler untuk membatasi daftar nama tampilan.

Athena mengembalikan daftar`STRING`jenis nilai tempat setiap nilai adalah nama tampilan.

Sinopsis

```
SHOW VIEWS [IN database_name] [LIKE 'regular_expression']
```

Parameter

[DI database_name]

Menentukan`database_name`dari mana tampilan akan terdaftar. Jika dihilangkan, basis data dari konteks saat ini diasumsikan.

[SEPERTI 'regular_expression']

Menyaring daftar tampilan untuk orang-orang yang cocok`regular_expression`Anda tentukan. Hanya karakter wild card`*`, yang menunjukkan karakter apa pun, atau`|`, yang menunjukkan pilihan antar karakter, yang dapat digunakan.

Contoh

```
SHOW VIEWS;
```

```
SHOW VIEWS IN marketing_analytics LIKE 'orders*'
```

Lihat juga SHOW COLUMNS, SHOW CREATE VIEW, DESCRIBE VIEW, dan DROP VIEW.

Pertimbangan dan batasan untuk kueri SQL di Amazon Athena

Saat menjalankan kueri di Athena, ingatlah pertimbangan dan batasan berikut:

- Prosedur tersimpan - Prosedur tersimpan tidak didukung.
- Jumlah maksimum partisi - Jumlah maksimum partisi yang dapat Anda buat dengan pernyataan `CREATE TABLE AS SELECT` (CTAS) adalah 100. Untuk informasi selengkapnya, lihat BUAT TABEL SEBAGAI. Untuk solusi, lihat Menggunakan CTAS dan INSERT INTO untuk bekerja di sekitar batas partisi 100.
- Pernyataan tidak didukung - Pernyataan berikut tidak didukung:
  - `CREATE TABLE LIKE` tidak didukung.
  - `DESCRIBE INPUT` dan `DESCRIBE OUTPUT` tidak didukung.
  - `MERGE`Pernyataan ini didukung hanya untuk format tabel transaksional. Untuk informasi selengkapnya, lihat BERGABUNG MENJADI.
  - Pernyataan `UPDATE` tidak didukung.
- Konektor Trino dan Presto - Baik konektor Trino maupun Presto tidak didukung. Gunakan Kueri Gabungan Amazon Athena untuk menghubungkan sumber data. Untuk informasi selengkapnya, lihat Menggunakan Amazon Athena.
- Waktu habis pada tabel dengan banyak partisi - Athena mungkin kehabisan waktu saat mengkueri tabel yang memiliki ribuan partisi. Hal ini dapat terjadi saat tabel memiliki banyak partisi yang bukan tipe `string`. Saat Anda menggunakan tipe `string`, Athena akan memangkas partisi pada tingkat metastore. Namun, jika Anda menggunakan tipe data lainnya, Athena akan memangkas partisi di sisi server. Makin banyak partisi yang Anda miliki, makin lama proses ini berlangsung dan makin besar kemungkinan kueri Anda kehabisan waktu. Untuk mengatasi masalah ini, atur tipe partisi ke `string` sehingga Athena akan memangkas partisi pada tingkat metastore. Ini akan mengurangi overhead dan mencegah kueri kehabisan waktu.
- Dukungan S3 Glacier — Untuk informasi tentang kueri objek Amazon S3 Glacier yang dipulihkan, lihat. Menanyakan objek Amazon S3 Glacier yang dipulihkan
- fILE diperlakukan sebagai tersembunyi - Athena memperlakukan file sumber yang dimulai dengan garis bawah (_) atau titik (`.`) sebagai tersembunyi. Untuk mengatasi batasan ini, ganti nama file.
- Batasan ukuran baris atau kolom - Ukuran satu baris atau kolomnya tidak boleh melebihi 32 megabita. Batas ini dapat dilampaui ketika, misalnya, baris dalam file CSV atau JSON berisi satu kolom 300 megabita. Melebihi batas ini juga dapat menghasilkan pesan kesalahan Baris terlalu panjang dalam file teks. Untuk mengatasi batasan ini, pastikan bahwa jumlah data kolom di baris mana pun kurang dari 32 MB.

- BATAS klausa maksimum - Jumlah maksimum baris yang dapat ditentukan untuk `LIMIT` klausa adalah

  9223372036854775807. Saat menggunakan`ORDER BY`, jumlah maksimum baris yang didukung untuk klausa LIMIT adalah 2147483647. Melebihi batas ini menghasilkan pesan kesalahan NOT_SUPPORTED: ORDER BY LIMIT > 2147483647 tidak didukung.
- information_schema - Query paling berkinerja jika Anda memiliki jumlah metadata kecil hingga `information_schema` sedang. AWS Glue Jika Anda memiliki sejumlah besar metadata, kesalahan dapat terjadi. Untuk informasi tentang kueri `information_schema` database untuk AWS Glue metadata, lihat. Mengkueri AWS Glue Data Catalog
- Inisialisasi array — Karena keterbatasan di Java, tidak mungkin untuk menginisialisasi array di Athena yang memiliki lebih dari 254 argumen.
- Kolom metadata tersembunyi — Kolom metadata tersembunyi The Hive atau Iceberg`$bucket`, `$file_modified_time``$file_size`, dan tidak didukung untuk tampilan. `$partition` Untuk informasi tentang menggunakan kolom `$path` metadata di Athena, lihat. Mendapatkan lokasi file untuk data sumber di Amazon S3

Untuk informasi tentang panjang string kueri maksimum, kuota untuk batas waktu kueri, dan kuota untuk jumlah aktif kueri DHTML, lihat. Service Quotas

Pemecahan Masalah di Athena

Tim Athena telah mengumpulkan informasi pemecahan masalah berikut dari masalah pelanggan. Meskipun tidak komprehensif, ini mencakup saran mengenai beberapa masalah kinerja umum, batas waktu, dan kehabisan memori.

Topik

- BUAT TABEL SEBAGAI PILIH (CTAS)
- Masalah file data
- Tabel Delta Lake Yayasan Linux
- Pertanyaan federasi
- Kesalahan terkait JSON
- TABEL PERBAIKAN MSCK
- Masalah keluaran

BUAT TABEL SEBAGAI PILIH (CTAS)

Data duplikat terjadi dengan pernyataan CTAS bersamaan

Athena tidak mempertahankan validasi bersamaan untuk CTAS. Pastikan bahwa tidak ada pernyataan CTAS duplikat untuk lokasi yang sama pada saat yang sama. Bahkan jika pernyataan CTAS atau INSERT INTO gagal, data yatim piatu dapat ditinggalkan di lokasi data yang ditentukan dalam pernyataan.

HIVE_TOO_MANY_OPEN_PARTITIONS

Bila Anda menggunakan pernyataan CTAS untuk membuat tabel dengan lebih dari 100 partisi, Anda mungkin menerima kesalahan HIVE_TOO_MANY_OPEN_PARTITIONS: Melebihi batas 100 penulis terbuka untuk partisi/ember. Untuk mengatasi batasan ini, Anda dapat menggunakan pernyataan CTAS dan serangkaian `INSERT INTO` pernyataan yang membuat atau menyisipkan hingga 100 partisi masing-masing. Untuk informasi selengkapnya, lihat Menggunakan CTAS dan INSERT INTO untuk bekerja di sekitar batas partisi 100.

Masalah file data

Athena tidak dapat membaca file tersembunyi

Athena memperlakukan file sumber yang dimulai dengan garis bawah (_) atau titik (.) sebagai tersembunyi. Untuk mengatasi batasan ini, ganti nama file.

Athena membaca file yang saya kecualikan dari crawler AWS Glue

Athena tidak mengenali pola pengecualian yang Anda tentukan crawler. AWS Glue Misalnya, jika Anda memiliki bucket Amazon S3 yang berisi keduanya `.csv`dan `.json`file dan Anda mengecualikan `.json`file dari crawler, Athena mengkueri kedua grup file. Untuk menghindari hal ini, menempatkan file yang ingin Anda mengecualikan di lokasi yang berbeda.

HIVE_BAD_DATA: Kesalahan mengurai nilai bidang

Kesalahan ini dapat terjadi dalam skenario berikut:

- Tipe data yang ditentukan dalam tabel tidak cocok dengan data sumber, atau satu bidang berisi berbagai jenis data. Untuk resolusi yang disarankan, lihat Kueri Amazon Athena saya gagal dengan kesalahan “HIVE_BAD_DATA: Error parsing field value for field x: For input string: “12312845691"” di Pusat Pengetahuan. AWS
- Nilai nol hadir dalam bidang integer. Salah satu solusinya adalah membuat kolom dengan nilai nol sebagai `string` dan kemudian digunakan `CAST` untuk mengonversi bidang dalam kueri, memberikan nilai default untuk nol. `0` Untuk informasi lebih lanjut, lihat Ketika saya menanyakan data CSV di Athena, saya mendapatkan kesalahan “HIVE_BAD_DATA: Nilai bidang penguraian kesalahan" untuk bidang x: Untuk string input: “"” di Pusat Pengetahuan. AWS

HIVE_CANNOT_OPEN_SPLIT: Kesalahan saat membuka pemisahan Hive s3://DOC-EXAMPLE-BUCKET

Kesalahan ini dapat terjadi saat Anda menanyakan awalan bucket Amazon S3 yang memiliki banyak objek. Untuk informasi lebih lanjut, lihat Bagaimana cara mengatasi kesalahan “HIVE_CANNOT_OPEN_SPLIT: Error opening Hive split s3://DOC-EXAMPLE-BUCKET/: Slow down” di Athena? di pusat AWS pengetahuan.

HIVE_CURSOR_ERROR: com.amazonaws.services.s3.model.amazons3Exception: Kunci yang ditentukan tidak ada

Kesalahan ini biasanya terjadi ketika file dihapus ketika kueri sedang berjalan. Jalankan kembali kueri, atau periksa alur kerja Anda untuk melihat apakah pekerjaan atau proses lain memodifikasi file saat kueri sedang berjalan.

HIVE_CURSOR_ERROR: Akhir aliran input yang tidak terduga

Pesan ini menunjukkan file rusak atau kosong. Periksa integritas file dan jalankan kembali kueri.

HIVE_FILESYSTEM_ERROR: Ukuran file 1234567 salah untuk file

Pesan ini dapat terjadi ketika file telah berubah antara perencanaan kueri dan eksekusi kueri. Biasanya terjadi ketika file di Amazon S3 diganti di tempat (misalnya, a PUT dilakukan pada kunci di mana objek sudah ada). Athena tidak mendukung penghapusan atau penggantian isi file saat kueri sedang berjalan. Untuk menghindari kesalahan ini, jadwalkan pekerjaan yang menimpa atau menghapus file pada saat kueri tidak berjalan, atau hanya menulis data ke file atau partisi baru.

HIVE_UNKNOWN_ERROR: Tidak dapat membuat format masukan

Kesalahan ini dapat disebabkan oleh masalah seperti berikut:

- AWS Glue Crawler tidak dapat mengklasifikasikan format data
- Properti definisi AWS Glue tabel tertentu kosong
- Athena tidak mendukung format data file di Amazon S3

Untuk informasi selengkapnya, lihat Bagaimana cara mengatasi kesalahan “tidak dapat membuat format input” di Athena? di Pusat AWS Pengetahuan atau tonton video Pusat Pengetahuan.

Lokasi S3 yang disediakan untuk menyimpan hasil kueri Anda tidak valid.

Pastikan Anda telah menentukan lokasi S3 yang valid untuk hasil kueri Anda. Untuk informasi selengkapnya, lihat Menentukan lokasi hasil query dalam topik Bekerja dengan hasil kueri, kueri terbaru, dan file keluaran.

Tabel Delta Lake Yayasan Linux

Skema tabel Delta Lake tidak sinkron

Saat Anda menanyakan tabel Delta Lake yang memiliki skema AWS Glue yang sudah usang, Anda dapat menerima pesan galat berikut:

```
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.
```

Skema dapat menjadi usang jika dimodifikasi AWS Glue setelah ditambahkan ke Athena. Untuk memperbarui skema, lakukan salah satu langkah berikut:

- Masuk AWS Glue, jalankan AWS Glue crawler.
- Di Athena, jatuhkan meja dan buat lagi.
- Tambahkan kolom yang hilang secara manual, baik dengan menggunakan ALTER TABLE ADD COLUMNS pernyataan di Athena, atau dengan mengedit skema tabel di. AWS Glue

Pertanyaan federasi

Batas waktu saat menelepon ListTableMetadata

Panggilan ke ListTableMetadataAPI dapat batas waktu jika ada banyak tabel di sumber data, jika sumber data lambat, atau jika jaringan lambat. Untuk memecahkan masalah ini, coba langkah-langkah berikut.

- Periksa jumlah tabel — Jika Anda memiliki lebih dari 1000 tabel, coba kurangi jumlah tabel. Untuk `ListTableMetadata` respons tercepat, kami sarankan memiliki kurang dari 1000 tabel per katalog.
- Periksa konfigurasi Lambda - Memantau perilaku fungsi Lambda sangat penting. Saat Anda menggunakan katalog federasi, pastikan untuk memeriksa log eksekusi fungsi Lambda. Berdasarkan hasil, sesuaikan nilai memori dan batas waktu yang sesuai. Untuk mengidentifikasi potensi masalah dengan batas waktu, kunjungi kembali konfigurasi Lambda Anda. Untuk informasi selengkapnya, lihat Mengonfigurasi batas waktu fungsi (konsol) di Panduan AWS Lambda Pengembang.
- Periksa log sumber data federasi — Periksa log dan pesan kesalahan dari sumber data federasi untuk melihat apakah ada masalah atau kesalahan. Log dapat memberikan wawasan berharga tentang penyebab batas waktu.
- Gunakan **`StartQueryExecution`** untuk mengambil metadata — Jika Anda memiliki lebih dari 1000 tabel, dibutuhkan waktu lebih lama dari yang diharapkan untuk mengambil metadata menggunakan konektor federasi Anda. Karena sifat asinkron memastikan StartQueryExecutionbahwa Athena menjalankan kueri dengan cara yang paling optimal, pertimbangkan untuk menggunakan `StartQueryExecution` sebagai alternatif untuk. `ListTableMetadata` AWS CLI Contoh berikut menunjukkan bagaimana `StartQueryExecution` bisa digunakan bukan `ListTableMetadata` untuk mendapatkan semua metadata tabel dalam katalog data Anda.

  Pertama, jalankan query yang mendapatkan semua tabel, seperti pada contoh berikut.

```
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"
```

Selanjutnya, ambil metadata tabel individual, seperti pada contoh berikut.

```
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"
```

Waktu yang dibutuhkan untuk mendapatkan hasil tergantung pada jumlah tabel dalam katalog Anda.

Untuk informasi selengkapnya tentang pemecahan masalah kueri gabungan, lihat Common_Problems di aws-athena-query-federation bagian awslabs/, GitHub atau lihat dokumentasi untuk masing-masing konektor sumber data Athena.

Kesalahan terkait JSON

Kesalahan data NULL atau salah saat mencoba membaca data JSON

Kesalahan data NULL atau salah saat Anda mencoba membaca data JSON dapat disebabkan oleh sejumlah penyebab. Untuk mengidentifikasi garis yang menyebabkan kesalahan saat Anda menggunakan OpenX SerDe, atur `ignore.malformed.json` ke. `true` Catatan cacat akan kembali sebagai NULL. Untuk informasi lebih lanjut, lihat Saya mendapatkan kesalahan ketika saya mencoba membaca data JSON di Amazon Athena di AWS Pusat Pengetahuan atau menonton video Pusat Pengetahuan.

HIVE_BAD_DATA: Kesalahan mengurai nilai bidang untuk bidang 0: java.lang.String tidak dapat dilemparkan ke org.openx.data.jsonserde.json.jsonObject

Ini OpenX JSON SerDe melempar kesalahan ini ketika gagal mengurai kolom dalam kueri Athena. Ini bisa terjadi jika Anda mendefinisikan kolom sebagai `map` atau`struct`, tetapi data yang mendasarinya sebenarnya adalah`string`,`int`, atau tipe primitif lainnya.

HIVE_CURSOR_ERROR: Baris bukan objek JSON yang valid - JSONException: Kunci duplikat

Kesalahan ini terjadi ketika Anda menggunakan Athena untuk menanyakan AWS Config sumber daya yang memiliki beberapa tag dengan nama yang sama dalam kasus yang berbeda. Solusinya adalah menjalankan `CREATE TABLE` menggunakan `WITH SERDEPROPERTIES 'case.insensitive'='false'` dan memetakan nama. Untuk informasi tentang `case.insensitive` dan pemetaan, lihatPerpustakaan JSON SerDe . Untuk informasi lebih lanjut, lihat Bagaimana cara mengatasi “HIVE_CURSOR_ERROR: Baris bukan objek JSON yang valid - JSONException: Kunci duplikat” saat membaca file dari Athena? AWS Config di pusat AWS pengetahuan.

Pesan HIVE_CURSOR_ERROR dengan JSON yang dicetak cantik

OpenX JSON SerDePustaka Sarang JSON SerDe dan mengharapkan setiap dokumen JSON berada pada satu baris teks tanpa karakter penghentian baris yang memisahkan bidang dalam catatan. Jika teks JSON dalam format cetak cantik, Anda mungkin menerima pesan kesalahan seperti HIVE_CURSOR_ERROR: Row is not a valid JSON Object or HIVE_CURSOR_ERROR:: Unexpected JsonParseException end-of-input: expected close marker for OBJECT saat Anda mencoba menanyakan tabel setelah Anda membuatnya. Untuk informasi selengkapnya, lihat file data JSON dalam dokumentasi SerDe OpenX pada. GitHub

Beberapa catatan JSON mengembalikan HITUNGAN SELECT 1

Jika Anda menggunakanOpenX JSON SerDe, pastikan bahwa catatan dipisahkan oleh karakter baris baru. Untuk informasi selengkapnya, lihat Kueri SELECT COUNT di Amazon Athena hanya menampilkan satu catatan meskipun file JSON masukan memiliki beberapa catatan di Pusat Pengetahuan. AWS

Tidak dapat menanyakan tabel yang dibuat oleh AWS Glue crawler yang menggunakan pengklasifikasi JSON kustom

Mesin Athena tidak mendukung pengklasifikasi JSON khusus. Untuk mengatasi masalah ini, buat tabel baru tanpa pengklasifikasi khusus. Untuk mengubah JSON, Anda dapat menggunakan CTAS atau membuat tampilan. Misalnya, jika Anda bekerja dengan array, Anda dapat menggunakan opsi UNNEST untuk meratakan JSON. Pilihan lain adalah menggunakan pekerjaan AWS Glue ETL yang mendukung pengklasifikasi khusus, mengonversi data menjadi parket di Amazon S3, dan kemudian menanyakannya di Athena.

TABEL PERBAIKAN MSCK

Untuk informasi tentang masalah terkait MSCK REPAIR TABLE, lihat Pemecahan Masalah bagian Pertimbangan dan batasan dan MSCK REPAIR TABLE halaman.

Masalah keluaran

Tidak dapat memverifikasi/membuat bucket keluaran

Kesalahan ini dapat terjadi jika lokasi hasil kueri yang ditentukan tidak ada atau jika izin yang tepat tidak ada. Untuk informasi selengkapnya, lihat Bagaimana cara mengatasi kesalahan “tidak dapat memverifikasi/membuat bucket keluaran” di Amazon Athena? di pusat AWS pengetahuan.

Hasil TIMESTAMP kosong

Athena membutuhkan format Java TIMESTAMP. Untuk informasi selengkapnya, lihat Saat saya menanyakan tabel di Amazon Athena, hasil TIMESTAMP kosong di Pusat Pengetahuan. AWS

Simpan output kueri Athena dalam format selain CSV

Secara default, Athena hanya mengeluarkan file dalam format CSV. Untuk menampilkan hasil `SELECT` kueri dalam format yang berbeda, Anda dapat menggunakan `UNLOAD` pernyataan tersebut. Untuk informasi selengkapnya, lihat MEMBONGKAR. Anda juga dapat menggunakan kueri CTAS yang menggunakan properti `format` tabel untuk mengkonfigurasi format output. Tidak seperti`UNLOAD`, teknik CTAS membutuhkan pembuatan tabel. Untuk informasi selengkapnya, lihat Bagaimana cara menyimpan output kueri Athena dalam format selain CSV, seperti format terkompresi? di pusat AWS pengetahuan.

Lokasi S3 yang disediakan untuk menyimpan hasil kueri Anda tidak valid

Anda dapat menerima pesan galat ini jika lokasi bucket keluaran tidak berada di Region yang sama dengan Region tempat Anda menjalankan kueri. Untuk menghindari hal ini, tentukan lokasi hasil kueri di Wilayah tempat Anda menjalankan kueri. Untuk langkah, lihat Menentukan lokasi hasil query.

Masalah parket

org.apache.parquet.io. GroupColumnIO tidak dapat dilemparkan ke org.apache.parquet.io. PrimitiveColumnIO

Kesalahan ini disebabkan oleh ketidakcocokan skema parket. Kolom yang memiliki tipe non-primitif (misalnya,`array`) telah dideklarasikan sebagai tipe primitif (misalnya,`string`) di. AWS Glue Untuk

memecahkan masalah ini, periksa skema data dalam file dan bandingkan dengan skema yang dideklarasikan. AWS Glue

Masalah statistik parket

Ketika Anda membaca data Parket, Anda mungkin menerima pesan kesalahan seperti berikut:

```
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.
```

Untuk mengatasi masalah ini, gunakan ALTER TABLE SET TBLPROPERTIES pernyataan CREATE TABLE or untuk menyetel SerDe `parquet.ignore.statistics` properti Parket`true`, seperti pada contoh berikut.

CREATE TABLE contoh

```
...
ROW FORMAT SERDE
'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
WITH SERDEPROPERTIES ('parquet.ignore.statistics'='true')
STORED AS PARQUET
...
```

Contoh ALTER TABLE

```
ALTER TABLE ... SET TBLPROPERTIES ('parquet.ignore.statistics'='true')
```

Untuk informasi lebih lanjut tentang Parquet Hive SerDe, lihat. Parket SerDe

Masalah partisi

MSCK REPAIR TABLE tidak menghapus partisi basi

Jika Anda menghapus partisi secara manual di Amazon S3 dan kemudian menjalankan MSCK REPAIR TABLE, Anda mungkin menerima pesan kesalahan Partisi hilang dari sistem file. Hal ini terjadi karena MSCK REPAIR TABLE tidak menghapus partisi basi dari metadata tabel. Gunakan

ALTER TABLE DROP PARTITION untuk menghapus partisi basi secara manual. Untuk informasi selengkapnya, lihat bagian “Pemecahan Masalah” pada topik. MSCK REPAIR TABLE

Kegagalan TABEL PERBAIKAN MSCK

Ketika sejumlah besar partisi (misalnya, lebih dari 100.000) dikaitkan dengan tabel tertentu, `MSCK REPAIR TABLE` dapat gagal karena keterbatasan memori. Untuk mengatasi batas ini, gunakan ALTER TABLE ADD PARTITION sebagai gantinya.

MSCK REPAIR TABLE mendeteksi partisi tetapi tidak menambahkannya AWS Glue

Masalah ini dapat terjadi jika jalur Amazon S3 dalam kasus unta, bukan huruf kecil atau kebijakan IAM tidak mengizinkan tindakan tersebut. `glue:BatchCreatePartition` Untuk informasi lebih lanjut, lihat MSCK REPAIR TABLE mendeteksi partisi di Athena tetapi tidak menambahkannya ke dalam Pusat Pengetahuan. AWS Glue Data Catalog AWS

Rentang proyeksi partisi dengan format tanggal DD-MM-YYYY-HH-MM-SS atau YYYY-MM-DD tidak berfungsi

Agar berfungsi dengan benar, format tanggal harus diatur key`yyyy-MM-dd HH:00:00`. Untuk informasi selengkapnya, lihat Proyeksi partisi Stack Overflow post Athena tidak berfungsi seperti yang diharapkan.

PARTITION BY tidak mendukung tipe BIGINT

Ubah tipe data ke `string` dan coba lagi.

Tidak ada partisi yang berarti tersedia

Pesan kesalahan ini biasanya berarti pengaturan partisi telah rusak. Untuk mengatasi masalah ini, jatuhkan tabel dan buat tabel dengan partisi baru.

Proyeksi partisi tidak bekerja bersama dengan partisi rentang

Periksa apakah proyeksi unit rentang waktu. <columnName>.interval.unit cocok dengan pembatas untuk partisi. Misalnya, jika partisi dibatasi oleh hari, maka satuan rentang jam tidak akan berfungsi.

Kesalahan proyeksi partisi ketika rentang ditentukan oleh tanda hubung

Menentukan properti range tabel dengan tanda hubung alih-alih koma menghasilkan kesalahan seperti INVALID_TABLE_PROPERTY: Untuk string input:

"number - number". Pastikan nilai rentang dipisahkan dengan koma, bukan tanda hubung. Untuk informasi selengkapnya, lihat Jenis bilangan bulat.

HIVE_UNKNOWN_ERROR: Tidak dapat membuat format masukan

Satu atau lebih partisi lem dinyatakan dalam format yang berbeda karena setiap partisi lem memiliki format input spesifiknya sendiri secara independen. Silakan periksa bagaimana partisi Anda didefinisikan dalam AWS Glue.

HIVE_PARTITION_SCHEMA_MISMATCH

Jika skema partisi berbeda dari skema tabel, kueri dapat gagal dengan pesan kesalahan HIVE_PARTITION_SCHEMA_MISMATCH. Untuk informasi selengkapnya, lihat Menyinkronkan skema partisi untuk menghindari "HIVE_PARTITION_SCHEMA_MISMATCH".

SemanticException tabel tidak dipartisi tetapi spesifikasi partisi ada

Kesalahan ini dapat terjadi ketika tidak ada partisi yang didefinisikan dalam `CREATE TABLE` pernyataan. Untuk informasi lebih lanjut, lihat Bagaimana cara memecahkan masalah kesalahan "GAGAL: SemanticException tabel tidak dipartisi tetapi spesifikasi partisi ada" di Athena? di pusat AWS pengetahuan.

_$folder$File byte nol

Jika Anda menjalankan `ALTER TABLE ADD PARTITION` pernyataan dan salah menentukan partisi yang sudah ada dan lokasi Amazon S3 yang salah, file placeholder nol byte dari *`partition_value`*`_$folder$` format dibuat di Amazon S3. Anda harus menghapus file-file ini secara manual.

Untuk mencegah hal ini terjadi, gunakan `ADD IF NOT EXISTS` sintaks dalam `ALTER TABLE ADD PARTITION` pernyataan Anda, seperti ini:

```
ALTER TABLE table_name ADD IF NOT EXISTS PARTITIION […]
```

Nol catatan dikembalikan dari data yang dipartisi

Masalah ini dapat terjadi karena berbagai alasan. Untuk kemungkinan penyebab dan resolusi, lihat Saya membuat tabel di Amazon Athena dengan partisi yang ditentukan, tetapi ketika saya menanyakan tabel, nol catatan dikembalikan di AWS Pusat Pengetahuan.

Lihat juga HIVE_TOO_MANY_OPEN_PARTITIONS.

Izin

Kesalahan akses ditolak saat menanyakan Amazon S3

Hal ini dapat terjadi jika Anda tidak memiliki izin untuk membaca data di bucket, izin untuk menulis ke bucket hasil, atau jalur Amazon S3 berisi titik akhir Wilayah seperti. `us-east-1.amazonaws.com` Untuk informasi selengkapnya, lihat Ketika saya menjalankan kueri Athena, saya mendapatkan kesalahan “akses ditolak” di Pusat AWS Pengetahuan.

Akses ditolak dengan kode status: Kesalahan 403 saat menjalankan kueri DDL pada data terenkripsi di Amazon S3

Ketika Anda mungkin menerima pesan kesalahan Akses Ditolak (Layanan: Amazon S3; Kode Status: 403; Kode Kesalahan: AccessDenied; ID Permintaan:) <request_id>jika kondisi berikut benar:

1. Anda menjalankan query DDL seperti `ALTER TABLE ADD PARTITION` atau`MSCK REPAIR TABLE`.
2. Anda memiliki bucket yang memiliki enkripsi default yang dikonfigurasi untuk digunakan`SSE-S3`.
3. Bucket juga memiliki kebijakan bucket seperti berikut ini yang memaksa `PutObject` permintaan untuk menentukan PUT header `"s3:x-amz-server-side-encryption": "true"` dan`"s3:x-amz-server-side-encryption": "AES256"`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::<resource-name>/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption": "true"
                }
            }
        },
        {
            "Effect": "Deny",
```

```
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::<resource-name>/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "AES256"
                }
            }
        }
    ]
}
```

Dalam kasus seperti ini, solusi yang disarankan adalah menghapus kebijakan bucket seperti yang di atas mengingat enkripsi default bucket sudah ada.

Akses ditolak dengan kode status: 403 saat menanyakan bucket Amazon S3 di akun lain

Kesalahan ini dapat terjadi ketika Anda mencoba menanyakan log yang ditulis oleh orang lain Layanan AWS dan akun kedua adalah pemilik bucket tetapi tidak memiliki objek di ember. Untuk informasi lebih lanjut, lihat saya mendapatkan pengecualian Amazon S3 “akses ditolak dengan kode status: 403" di Amazon Athena ketika saya menanyakan ember di akun lain di Pusat Pengetahuan atau menonton video Pusat AWS Pengetahuan.

Gunakan kredensi peran IAM untuk terhubung ke driver Athena JDBC

Anda dapat mengambil kredensi sementara peran untuk mengautentikasi koneksi JDBC ke Athena. Kredensi sementara memiliki umur maksimum 12 jam. Untuk informasi selengkapnya, lihat Bagaimana cara menggunakan kredensi peran IAM saya atau beralih ke peran IAM lain saat menyambung ke Athena menggunakan driver JDBC? di pusat AWS pengetahuan.

Masalah sintaks kueri

GAGAL: NullPointerException nama adalah null

Jika Anda menggunakan operasi AWS Glue CreateTableAPI atau AWS CloudFormation `AWS::Glue::Table`template untuk membuat tabel untuk digunakan di Athena tanpa menentukan `TableType` properti dan kemudian menjalankan kueri DDL seperti `SHOW CREATE TABLE` atau`MSCK REPAIR TABLE`, Anda dapat menerima pesan kesalahan GAGAL: NullPointerException Nama adalah null.

Untuk mengatasi kesalahan, tentukan nilai TableInput`TableType`atribut sebagai bagian dari panggilan AWS Glue `CreateTable` API atau AWS CloudFormation templat. Nilai yang mungkin untuk `TableType` include `EXTERNAL_TABLE` atau`VIRTUAL_VIEW`.

Persyaratan ini hanya berlaku ketika Anda membuat tabel menggunakan operasi AWS Glue `CreateTable` API atau `AWS::Glue::Table` template. Jika Anda membuat tabel untuk Athena menggunakan pernyataan DDL atau AWS Glue crawler, `TableType` properti didefinisikan untuk Anda secara otomatis.

Fungsi tidak terdaftar

Kesalahan ini terjadi ketika Anda mencoba menggunakan fungsi yang tidak didukung Athena. Untuk daftar fungsi yang didukung Athena, lihat Fungsi di Amazon Athena atau jalankan `SHOW FUNCTIONS` pernyataan di Editor Kueri. Anda juga dapat menulis fungsi yang ditentukan pengguna (UDF) Anda sendiri. Untuk informasi selengkapnya, lihat Bagaimana cara mengatasi kesalahan sintaks "fungsi tidak terdaftar" di Athena? di pusat AWS pengetahuan.

PENGECUALIAN GENERIC_INTERNAL_ERROR

`GENERIC_INTERNAL_ERROR`pengecualian dapat memiliki berbagai penyebab, termasuk yang berikut:

- GENERIC_INTERNAL_ERROR: Null — Anda mungkin melihat pengecualian ini di bawah salah satu kondisi berikut:
  - Anda memiliki ketidakcocokan skema antara tipe data kolom dalam definisi tabel dan tipe data aktual dari kumpulan data.
  - Anda menjalankan kueri `CREATE TABLE AS SELECT` (CTAS) dengan sintaks yang tidak akurat.
- GENERIC_INTERNAL_ERROR: Pembuat induk adalah null — Anda mungkin melihat pengecualian ini saat Anda menanyakan tabel dengan kolom tipe data, `array` dan Anda menggunakan pustaka OpenCSV. SerDe Format OpenCSVserde tidak mendukung tipe data. `array`
- GENERIC_INTERNAL_ERROR: Nilai melebihi MAX_INT — Anda mungkin melihat pengecualian ini ketika kolom data sumber didefinisikan dengan tipe data `INT` dan memiliki nilai numerik lebih besar dari 2.147,483.647.
- GENERIC_INTERNAL_ERROR: Nilai melebihi MAX_BYTE — Anda mungkin melihat pengecualian ini ketika kolom data sumber memiliki nilai numerik melebihi ukuran yang diizinkan untuk tipe data. `BYTE` Tipe data `BYTE` setara dengan`TINYINT`. `TINYINT`adalah bilangan bulat bertanda 8-bit dalam format komplemen dua dengan nilai minimum -128 dan nilai maksimum 127.

- GENERIC_INTERNAL_ERROR: Jumlah nilai partisi tidak cocok dengan jumlah filter — Anda mungkin melihat pengecualian ini jika Anda memiliki partisi yang tidak konsisten pada data Amazon Simple Storage Service (Amazon S3). Anda mungkin memiliki partisi yang tidak konsisten di bawah salah satu dari kondisi berikut:
  - Partisi di Amazon S3 telah berubah (contoh: partisi baru ditambahkan).
  - Jumlah kolom partisi dalam tabel tidak cocok dengan yang ada di metadata partisi.

Untuk informasi lebih rinci tentang masing-masing kesalahan ini, lihat Bagaimana cara mengatasi kesalahan “GENERIC_INTERNAL_ERROR” saat saya menanyakan tabel di Amazon Athena? di pusat AWS pengetahuan.

Jumlah grup yang cocok tidak cocok dengan jumlah kolom

Kesalahan ini terjadi ketika Anda menggunakan Regex SerDe dalam pernyataan CREATE TABLE dan jumlah grup pencocokan regex tidak cocok dengan jumlah kolom yang Anda tentukan untuk tabel. Untuk informasi selengkapnya, lihat Bagaimana cara mengatasi RegexSerDe kesalahan “jumlah grup yang cocok tidak cocok dengan jumlah kolom” di Amazon Athena? di pusat AWS pengetahuan.

QueryString gagal memenuhi kendala: Anggota harus memiliki panjang kurang dari atau sama dengan 262144

Panjang string kueri maksimum di Athena (262.144 byte) bukanlah kuota yang dapat disesuaikan. AWS Support tidak dapat menambah kuota untuk Anda, tetapi Anda dapat mengatasi masalah dengan membagi kueri panjang menjadi yang lebih kecil. Untuk informasi selengkapnya, lihatBagaimana cara meningkatkan panjang string kueri maksimum di Athena?di AWS Pusat Pengetahuan.

SYNTAX_ERROR: Kolom tidak dapat diselesaikan

Kesalahan ini dapat terjadi ketika Anda menanyakan tabel yang dibuat oleh AWS Glue crawler dari file CSV yang dikodekan UTF-8 yang memiliki tanda urutan byte (BOM). AWS Glue tidak mengenali BOM dan mengubahnya menjadi tanda tanya, yang tidak dikenali Amazon Athena. Solusinya adalah menghapus tanda tanya di Athena atau di. AWS Glue

Terlalu banyak argumen untuk panggilan fungsi

Di mesin Athena versi 3, fungsi tidak dapat mengambil lebih dari 127 argumen. Keterbatasan ini berdasarkan desain. Jika Anda menggunakan fungsi dengan lebih dari 127 parameter, pesan kesalahan seperti berikut terjadi:

TOO_MANY_ARGUMENTS: baris *nnn: nn*: Terlalu banyak argumen *untuk fungsi panggilan function_name ().*

Untuk mengatasi masalah ini, gunakan lebih sedikit parameter per panggilan fungsi.

Masalah batas waktu kueri

Jika Anda mengalami kesalahan batas waktu dengan kueri Athena Anda, periksa log Anda. CloudTrail Kueri dapat habis waktu karena pembatasan atau Lake AWS Glue Formation API. Ketika kesalahan ini terjadi, pesan kesalahan terkait dapat menunjukkan masalah batas waktu kueri daripada masalah pelambatan. Untuk memecahkan masalah ini, Anda dapat memeriksa CloudTrail log Anda sebelum menghubungi. AWS Support Untuk informasi selengkapnya, lihat Meminta log AWS CloudTrail dan Mencatat panggilan API Amazon Athena dengan AWS CloudTrail.

Untuk informasi tentang masalah batas waktu kueri dengan kueri gabungan saat Anda memanggil `ListTableMetadata` API, lihat. Batas waktu saat menelepon ListTableMetadata

Masalah pelambatan

Jika kueri Anda melebihi batas layanan dependen seperti Amazon S3 AWS KMS,, AWS Glue, AWS Lambda atau, pesan berikut dapat diharapkan. Untuk mengatasi masalah ini, kurangi jumlah panggilan bersamaan yang berasal dari akun yang sama.

Layanan	Pesan kesalahan
AWS Glue	AWSGlueException: Tarif terlampaui.
AWS KMS	Anda telah melampaui tingkat di mana Anda dapat menghubungi KMS. Kurangi frekuensi panggilan Anda.
AWS Lambda	Tarif terlampaui TooManyRequestsException

Layanan	Pesan kesalahan
Amazon S3	Amazons3Exception: Harap kurangi tingkat permintaan Anda.

Untuk informasi tentang cara mencegah pelambatan Amazon S3 saat Anda menggunakan Athena, lihat. Mencegah pelambatan Amazon S3

Tampilan

Tampilan yang dibuat di shell Apache Hive tidak berfungsi di Athena

Karena implementasinya yang berbeda secara fundamental, tampilan yang dibuat di Apache Hive shell tidak kompatibel dengan Athena. Untuk mengatasi masalah ini, buat kembali tampilan di Athena.

Tampilan sudah basi; itu harus dibuat ulang

Anda dapat menerima kesalahan ini jika tabel yang mendasari tampilan telah diubah atau dijatuhkan. Resolusinya adalah untuk menciptakan kembali tampilan. Untuk informasi lebih lanjut, lihat Bagaimana cara mengatasi kesalahan “tampilan sudah basi; itu harus dibuat ulang” di Athena? di pusat AWS pengetahuan.

Kelompok kerja

Untuk informasi tentang pemecahan masalah grup kerja, lihat. Pemecahan masalah kelompok kerja

Sumber daya tambahan

Halaman-halaman berikut memberikan informasi tambahan untuk mengatasi masalah dengan Amazon Athena.

- Katalog kesalahan Athena
- Service Quotas
- Pertimbangan dan batasan untuk kueri SQL di Amazon Athena
- DDL tidak didukung
- Nama untuk tabel, database, dan kolom
- Tipe data di Amazon Athena

- Format yang didukung SerDes dan data
- Dukungan kompresi Athena
- Kata kunci terpesan
- Pemecahan masalah kelompok kerja

AWS Sumber daya berikut juga dapat membantu:

- Topik Athena di pusat pengetahuan AWS
- Pertanyaan Amazon Athena di re:posting AWS
- Posting Athena di blog data AWS besar

Pemecahan masalah sering membutuhkan permintaan berulang dan penemuan oleh seorang ahli atau dari komunitas pembantu. Jika Anda terus mengalami masalah setelah mencoba saran di halaman ini, hubungi AWS Support (di, klik Support AWS Management Console, Support Center) atau ajukan pertanyaan di AWS re:post menggunakan tag Amazon Athena.

Katalog kesalahan Athena

Athena menyediakan informasi kesalahan standar untuk membantu Anda memahami kueri yang gagal dan mengambil langkah-langkah setelah kegagalan kueri terjadi. `AthenaError`Fitur ini mencakup `ErrorCategory` bidang dan `ErrorType` bidang. `ErrorCategory`menentukan apakah penyebab kueri gagal adalah karena kesalahan sistem, kesalahan pengguna, atau kesalahan lainnya. `ErrorType`memberikan informasi yang lebih terperinci mengenai sumber kegagalan. Dengan menggabungkan dua bidang, Anda bisa mendapatkan pemahaman yang lebih baik tentang keadaan di sekitarnya dan penyebab kesalahan spesifik yang terjadi.

Kategori kesalahan

Tabel berikut mencantumkan nilai kategori kesalahan Athena dan artinya.

Kategori kesalahan	Sumber
1	SISTEM
2	USER
3	LAINNYA

Referensi tipe kesalahan

Tabel berikut mencantumkan nilai jenis kesalahan Athena dan artinya.

Jenis kesalahar	Deskripsi
0	Kueri sumber daya yang habis pada faktor skala ini
1	Kueri sumber daya yang habis pada faktor skala ini
2	Kueri sumber daya yang habis pada faktor skala ini
3	Kueri sumber daya yang habis pada faktor skala ini
4	Kueri sumber daya yang habis pada faktor skala ini
5	Kueri sumber daya yang habis pada faktor skala ini
6	Kueri sumber daya yang habis pada faktor skala ini
7	Kueri sumber daya yang habis pada faktor skala ini
8	Kueri sumber daya yang habis pada faktor skala ini
100	Kesalahan layanan internal
200	Mesin kueri memiliki kesalahan internal
201	Mesin kueri memiliki kesalahan internal
202	Mesin kueri memiliki kesalahan internal
203	Kesalahan driver
204	Metastore memiliki kesalahan
205	Mesin kueri memiliki kesalahan internal
206	Waktu kueri habis
207	Mesin kueri memiliki kesalahan internal

Jenis kesalahar	Deskripsi
208	Mesin kueri memiliki kesalahan internal
209	Gagal membatalkan kueri
210	Waktu kueri habis
211	Mesin kueri memiliki kesalahan internal
212	Mesin kueri memiliki kesalahan internal
213	Mesin kueri memiliki kesalahan internal
214	Mesin kueri memiliki kesalahan internal
215	Mesin kueri memiliki kesalahan internal
216	Mesin kueri memiliki kesalahan internal
217	Mesin kueri memiliki kesalahan internal
218	Mesin kueri memiliki kesalahan internal
219	Mesin kueri memiliki kesalahan internal
220	Mesin kueri memiliki kesalahan internal
221	Mesin kueri memiliki kesalahan internal
222	Mesin kueri memiliki kesalahan internal
223	Mesin kueri memiliki kesalahan internal
224	Mesin kueri memiliki kesalahan internal
225	Mesin kueri memiliki kesalahan internal
226	Mesin kueri memiliki kesalahan internal
227	Mesin kueri memiliki kesalahan internal

Jenis kesalahar	Deskripsi
228	Mesin kueri memiliki kesalahan internal
229	Mesin kueri memiliki kesalahan internal
230	Mesin kueri memiliki kesalahan internal
231	Mesin kueri memiliki kesalahan internal
232	Mesin kueri memiliki kesalahan internal
233	Kesalahan gunung es
234	Kesalahan Lake Formation
235	Mesin kueri memiliki kesalahan internal
236	Mesin kueri memiliki kesalahan internal
237	Kesalahan serialisasi
238	Gagal mengunggah metadata ke Amazon S3
239	Kesalahan persistensi umum
240	Gagal mengirimkan kueri
300	Kesalahan layanan internal
301	Kesalahan layanan internal
302	Kesalahan layanan internal
303	Kesalahan layanan internal
400	Kesalahan layanan internal
401	Gagal menulis hasil kueri ke Amazon S3
402	Gagal menulis hasil kueri ke Amazon S3

Jenis kesalahar	Deskripsi
1000	Kesalahan pengguna
1001	Kesalahan data
1002	Kesalahan data
1003	Tugas DDL gagal
1004	Kesalahan skema
1005	Kesalahan serialisasi
1006	Kesalahan sintaks
1007	Kesalahan data
1008	Kueri ditolak
1009	Kueri gagal
1010	Kesalahan layanan internal
1011	Kueri dibatalkan oleh pengguna
1012	Mesin kueri memiliki kesalahan internal
1013	Mesin kueri memiliki kesalahan internal
1014	Kueri dibatalkan oleh pengguna
1100	Argumen tidak valid disediakan
1101	Properti tidak valid yang disediakan
1102	Mesin kueri memiliki kesalahan internal
1103	Properti tabel tidak valid disediakan
1104	Mesin kueri memiliki kesalahan internal

Jenis kesalahan	Deskripsi
1105	Mesin kueri memiliki kesalahan internal
1106	Argumen fungsi tidak valid disediakan
1107	Tampilan tidak valid
1108	Gagal mendaftarkan fungsi
1109	Asalkan jalur Amazon S3 tidak ditemukan
1110	Tabel atau tampilan yang disediakan tidak ada
1200	Kueri tidak didukung
1201	Dekoder yang disediakan tidak didukung
1202	Jenis kueri tidak didukung
1300	Kesalahan umum tidak ditemukan
1301	Entitas umum tidak ditemukan
1302	File tidak ditemukan
1303	Fungsi yang disediakan atau implementasi fungsi tidak ditemukan
1304	Mesin kueri memiliki kesalahan internal
1305	Mesin kueri memiliki kesalahan internal
1306	Bucket Amazon S3 tidak ditemukan
1307	Mesin yang dipilih tidak ditemukan
1308	Mesin kueri memiliki kesalahan internal
1400	Kesalahan pelambatan
1401	Kueri gagal karena AWS Glue pembatasan

Jenis kesalahar	Deskripsi
1402	Kueri gagal karena terlalu banyak versi tabel di AWS Glue
1403	Kueri gagal karena pelambatan Amazon S3
1404	Kueri gagal karena Amazon Athena melambat
1405	Kueri gagal karena Amazon Athena melambat
1406	Kueri gagal karena Amazon Athena melambat
1500	Kesalahan izin
1501	Kesalahan izin Amazon S3
1602	Melebihi batas kapasitas cadangan. Kapasitas yang tidak memadai untuk mengeksek usi query ini.
1700	Kueri gagal karena pengecualian internal Lake Formation
1701	Kueri gagal karena pengecualian AWS Glue internal
9999	Kesalahan layanan internal

Sampel Kode

Contoh dalam topik ini menggunakan SDK for Java 2.x sebagai titik awal untuk menulis aplikasi Athena.

> **Note**
>
> Untuk informasi tentang pemrograman Athena menggunakan AWS SDK khusus bahasa lainnya, lihat sumber daya berikut:
>
> - 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`

Untuk informasi selengkapnya tentang menjalankan contoh kode Java di bagian ini, lihat readme Amazon Athena Java pada repositori contoh AWS kode. GitHub Untuk referensi pemrograman Java untuk Athena, lihat AthenaClientdi. AWS SDK for Java 2.x

- Contoh Kode Java
  - Konstanta
  - Buat klien untuk mengakses Athena
  - Bekerja dengan Eksekusi Kueri
    - Mulai eksekusi kueri
    - Hentikan eksekusi kueri
    - Daftar eksekusi kueri
  - Bekerja dengan Kueri
    - Buat kueri bernama
    - Hapus kueri bernama
    - Daftar eksekusi kueri

> **Note**
>
> Sampel ini menggunakan konstanta (misalnya,`ATHENA_SAMPLE_QUERY`) untuk string, yang didefinisikan dalam`ExampleConstants.java`deklarasi kelas. Ganti konstanta ini dengan string Anda sendiri atau konstanta didefinisikan.

Konstanta

`ExampleConstants.java`Kelas menunjukkan bagaimana untuk query tabel yang dibuat oleh Memulai tutorial di Athena.

```
package aws.example.athena;

public class ExampleConstants {

    public static final int CLIENT_EXECUTION_TIMEOUT = 100000;
    public static final String ATHENA_OUTPUT_BUCKET = "s3://bucketscott2"; // change
 the Amazon S3 bucket name to match
                                                                          // your
 environment
    // Demonstrates how to query a table with a comma-separated value (CSV) table.
    // For information, see
    // https://docs.aws.amazon.com/athena/latest/ug/work-with-data.html
    public static final String ATHENA_SAMPLE_QUERY = "SELECT * FROM scott2;"; // change
 the Query statement to match
                                                                            // your
 environment
    public static final long SLEEP_AMOUNT_IN_MS = 1000;
    public static final String ATHENA_DEFAULT_DATABASE = "mydatabase"; // change the
 database to match your database

}
```

Buat klien untuk mengakses Athena

Kelas `AthenaClientFactory.java` menunjukkan cara membuat dan mengonfigurasi klien Amazon Athena.

```
package aws.example.athena;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.athena.AthenaClient;
import software.amazon.awssdk.services.athena.AthenaClientBuilder;

public class AthenaClientFactory {
    private final AthenaClientBuilder builder = AthenaClient.builder()
            .region(Region.US_WEST_2)
            .credentialsProvider(ProfileCredentialsProvider.create());

    public AthenaClient createClient() {
        return builder.build();
    }
```

```
}
```

Mulai eksekusi kueri

ParameterStartQueryExamplemenunjukkan cara mengirimkan kueri ke Athena, tunggu sampai hasilnya tersedia, lalu proses hasilnya.

```
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());
            }
        }
    }
}
```

Hentikan eksekusi kueri

ParameterStopQueryExecutionExamplemenjalankan kueri contoh, segera berhenti kueri, dan memeriksa status kueri untuk memastikan bahwa itu dibatalkan.

```
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;

    }
}
```

Daftar eksekusi kueri

ParameterListQueryExecutionsExamplemenunjukkan bagaimana untuk mendapatkan daftar ID eksekusi kueri.

```
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);
        }
    }
```

```
}
```

Buat kueri bernama

ParameterCreateNamedQueryExampleMenampilkan cara membuat kueri bernama.

```
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);
        }
    }
}
```

Hapus kueri bernama

Parameter`DeleteNamedQueryExample`menunjukkan bagaimana untuk menghapus kueri bernama dengan menggunakan ID kueri bernama.

```
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;
    }
}
```

Daftar kueri bernama

ParameterListNamedQueryExamplemenunjukkan cara mendapatkan daftar ID kueri bernama.

```
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);
        }
    }
}
```

Menggunakan Apache Spark di Amazon Athena

Amazon Athena memudahkan untuk menjalankan analitik dan eksplorasi data secara interaktif menggunakan Apache Spark tanpa perlu merencanakan, mengonfigurasi, atau mengelola sumber daya. Menjalankan aplikasi Apache Spark di Athena berarti mengirimkan kode Spark untuk diproses dan menerima hasil secara langsung tanpa perlu konfigurasi tambahan. Anda dapat menggunakan pengalaman notebook yang disederhanakan di konsol Amazon Athena untuk mengembangkan aplikasi Apache Spark menggunakan API notebook Python atau Athena. Apache Spark di Amazon Athena tanpa server dan menyediakan penskalaan otomatis sesuai permintaan yang memberikan komputasi instan untuk memenuhi perubahan volume data dan persyaratan pemrosesan.

Amazon Athena menawarkan fitur-fitur berikut:

- Penggunaan konsol — Kirim aplikasi Spark Anda dari konsol Amazon Athena.
- Scripting — Membangun dan men-debug aplikasi Apache Spark dengan cepat dan interaktif dengan Python.
- Penskalaan dinamis — Amazon Athena secara otomatis menentukan sumber daya komputasi dan memori yang diperlukan untuk menjalankan pekerjaan dan terus menskalakan sumber daya tersebut hingga maksimum yang Anda tentukan. Penskalaan dinamis ini mengurangi biaya tanpa mempengaruhi kecepatan.
- Pengalaman buku catatan — Gunakan editor notebook Athena untuk membuat, mengedit, dan menjalankan perhitungan menggunakan antarmuka yang sudah dikenal. Notebook Athena kompatibel dengan notebook Jupyter dan berisi daftar sel yang dieksekusi secara berurutan sebagai perhitungan. Konten sel dapat mencakup kode, teks, penurunan harga, matematika, plot, dan media kaya.

Untuk informasi tambahan, lihat Menjalankan Spark SQL di Amazon Athena Spark dan Jelajahi data lake Anda menggunakan Amazon Athena untuk Apache Spark di Big Data Blog.AWS

Pertimbangan dan batasan

- Saat ini, Amazon Athena untuk Apache Spark tersedia sebagai berikut: Wilayah AWS
  - Asia Pasifik (Mumbai)
  - Asia Pasifik (Singapura)
  - Asia Pasifik (Sydney)

  - Asia Pasifik (Tokyo)
  - Eropa (Frankfurt)
  - Eropa (Irlandia)
  - AS Timur (N. Virginia)
  - AS Timur (Ohio)
  - AS Barat (Oregon)
- AWS Lake Formation tidak didukung.
- Tabel yang menggunakan proyeksi partisi tidak didukung.
- Workgroup yang diaktifkan Apache Spark dapat menggunakan editor notebook Athena, tetapi bukan editor kueri Athena. Hanya workgroup Athena SQL yang dapat menggunakan editor kueri Athena.
- Kueri tampilan lintas-mesin tidak didukung. Tampilan yang dibuat oleh Athena SQL tidak dapat ditanyakan oleh Athena untuk Spark. Karena tampilan untuk kedua mesin diimplementasikan secara berbeda, mereka tidak kompatibel untuk penggunaan cross-engine.
- MLLib (pustaka pembelajaran mesin Apache Spark) dan `pyspark.ml` paket tidak didukung. Untuk daftar pustaka Python yang didukung, lihat. Daftar pustaka Python yang sudah diinstal sebelumnya
- Saat `pip install` ini, tidak didukung di Athena untuk sesi Spark.
- Hanya satu sesi aktif per notebook yang diizinkan.
- Saat beberapa pengguna menggunakan konsol untuk membuka sesi yang ada di grup kerja, mereka mengakses buku catatan yang sama. Untuk menghindari kebingungan, hanya buka sesi yang Anda buat sendiri.
- Domain hosting untuk aplikasi Apache Spark yang mungkin Anda gunakan dengan Amazon Athena (misalnya,`analytics-gateway.us-east-1.amazonaws.com`) terdaftar di Daftar Akhiran Publik (PSL) internet. Jika Anda perlu mengatur cookie sensitif di domain Anda, kami sarankan Anda menggunakan cookie dengan `__Host-` awalan untuk membantu mempertahankan domain Anda dari upaya pemalsuan permintaan lintas situs (CSRF). Untuk informasi selengkapnya, lihat halaman Set-Cookie di dokumentasi developer Mozilla.org.
- Untuk informasi tentang pemecahan masalah buku catatan, sesi, dan kelompok kerja Spark di Athena, lihat. Pemecahan Masalah Athena untuk Spark

Memulai dengan Apache Spark di Amazon Athena

Untuk memulai dengan Apache Spark di Amazon Athena, Anda harus terlebih dahulu membuat workgroup yang diaktifkan Spark. Setelah beralih ke workgroup, Anda dapat membuat buku catatan atau membuka buku catatan yang ada. Ketika Anda membuka buku catatan di Athena, sesi baru dimulai untuk itu secara otomatis dan Anda dapat bekerja dengannya langsung di editor notebook Athena.

> **Note**
>
> Pastikan Anda membuat workgroup yang diaktifkan Spark sebelum mencoba membuat buku catatan.

Membuat workgroup berkemampuan Spark di Athena

Anda dapat menggunakan kelompok kerja di Athena untuk mengelompokkan pengguna, tim, aplikasi, atau beban kerja, dan untuk melacak biaya. Untuk menggunakan Apache Spark di Amazon Athena, Anda membuat workgroup Amazon Athena yang menggunakan mesin Spark.

> **Note**
>
> Workgroup yang diaktifkan Apache Spark dapat menggunakan editor notebook Athena, tetapi bukan editor kueri Athena. Hanya workgroup Athena SQL yang dapat menggunakan editor kueri Athena.

Untuk membuat workgroup berkemampuan Spark di Athena

1. Buka konsol Athena di https://console.aws.amazon.com/athena/
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.

3. Di panel navigasi, pilih Workgroups.
4. Pada Grup Kerja, pilih Buat grup kerja.
5. Untuk nama Workgroup, masukkan nama untuk workgroup Apache Spark Anda.
6. (Opsional) Untuk Deskripsi, masukkan deskripsi untuk grup kerja Anda.
7. Untuk mesin Analytics, pilih Apache Spark.

> **Note**
>
> Setelah Anda membuat grup kerja, jenis mesin analitik workgroup tidak dapat diubah. Misalnya, workgroup mesin Athena versi 3 tidak dapat diubah menjadi workgroup PySpark engine versi 3.

8. Untuk keperluan tutorial ini, pilih Aktifkan contoh notebook. Fitur opsional ini menambahkan contoh buku catatan dengan nama `example-notebook-`*`random_string`* ke grup kerja Anda dan menambahkan izin AWS Glue terkait -yang digunakan notebook untuk membuat, menampilkan, dan menghapus database dan tabel tertentu di akun Anda, dan membaca izin di Amazon S3 untuk kumpulan data sampel. Untuk melihat izin yang ditambahkan, pilih Lihat detail izin tambahan.

> **Note**
>
> Menjalankan contoh notebook mungkin dikenakan biaya tambahan.

9. Untuk konfigurasi tambahan, lakukan salah satu hal berikut:

- Gunakan pengaturan Gunakan default. Opsi ini adalah default dan membantu Anda memulai dengan workgroup berkemampuan SPARK Anda. Dengan opsi ini, Athena membuat peran IAM dan lokasi hasil perhitungan di Amazon S3 untuk Anda. Nama peran IAM dan lokasi bucket S3 yang akan dibuat ditampilkan di kotak di bawah judul Konfigurasi tambahan.
- Nonaktifkan pengaturan Gunakan default, lalu lanjutkan dengan langkah-langkah di Menentukan konfigurasi workgroup Anda sendiri bagian untuk mengonfigurasi grup kerja Anda secara manual.

10. (Opsional) Tag - Gunakan opsi ini untuk menambahkan tag ke workgroup Anda. Untuk informasi selengkapnya, lihat Menandai sumber daya Athena.
11. Pilih Buat grup kerja. Pesan memberi tahu Anda bahwa workgroup berhasil dibuat, dan workgroup ditampilkan dalam daftar workgroup.

Menentukan konfigurasi workgroup Anda sendiri

Jika Anda ingin menentukan peran IAM Anda sendiri dan lokasi hasil perhitungan untuk buku catatan Anda, ikuti langkah-langkah di bagian ini. Jika Anda memilih Gunakan default untuk opsi Konfigurasi tambahan, lewati bagian ini dan langsung ke. Membuka penjelajah notebook dan beralih kelompok kerja

Prosedur berikut mengasumsikan Anda telah menyelesaikan langkah 1 hingga 9 dari Untuk membuat grup kerja yang diaktifkan Spark di Athena prosedur di bagian sebelumnya.

Untuk menentukan konfigurasi workgroup Anda sendiri

1. Jika Anda ingin membuat atau menggunakan peran IAM Anda sendiri atau mengonfigurasi enkripsi notebook, perluas konfigurasi peran IAM.

   - Untuk Peran Layanan, pilih salah satu dari berikut ini:
     - Buat peran layanan — Pilih opsi ini agar Athena membuat peran layanan untuk Anda. Untuk melihat izin yang diberikan peran, pilih Lihat detail izin.
     - Pilih peran layanan yang ada — Dari menu tarik-turun, pilih peran yang ada. Peran yang Anda pilih harus menyertakan izin di opsi pertama. Untuk informasi selengkapnya tentang izin untuk grup kerja berkemampuan notebook, lihat. Memecahkan masalah grup kerja berkemampuan SPARK
   - Untuk manajemen kunci enkripsi Notebook dan kode perhitungan, pilih salah satu opsi berikut:

- Dimiliki oleh Amazon Athena — AWS KMS Kuncinya dimiliki dan dikelola oleh Amazon Athena. Anda tidak dikenakan biaya tambahan untuk menggunakan kunci ini.
- Kunci simetris yang disimpan di akun Anda, dimiliki dan dikelola oleh Anda — Untuk opsi ini, lakukan salah satu hal berikut:
  - Untuk menggunakan kunci yang ada, gunakan kotak pencarian untuk memilih AWS KMS atau memasukkan kunci ARN.
  - Untuk membuat kunci di AWS KMS konsol, pilih Buat AWS KMS kunci. Peran eksekusi Anda harus memiliki izin untuk menggunakan kunci yang Anda buat.

> **Important**
>
> Saat Anda mengubah AWS KMS keyuntuk grup kerja, buku catatan yang dikelola sebelum pembaruan masih mereferensikan kunci KMS lama. Notebook dikelola setelah pembaruan menggunakan kunci KMS baru. Untuk memperbarui notebook lama untuk referensi kunci KMS baru, ekspor dan kemudian impor masing-masing notebook lama. Jika Anda menghapus kunci KMS lama sebelum memperbarui referensi notebook lama ke kunci KMS baru, notebook lama tidak lagi dapat didekripsi dan tidak dapat dipulihkan.
>
> Perilaku ini juga berlaku untuk pembaruan alias, yang merupakan nama ramah untuk kunci KMS. Saat Anda memperbarui alias kunci KMS untuk menunjuk ke kunci KMS baru, notebook yang dikelola sebelum pembaruan alias masih mereferensikan kunci KMS lama, dan notebook yang dikelola setelah pembaruan alias menggunakan kunci KMS baru. Pertimbangkan poin-poin ini sebelum memperbarui kunci atau alias KMS Anda.

2. Jika Anda ingin menentukan pengaturan hasil perhitungan Anda sendiri, perluas Pengaturan hasil perhitungan, lalu pilih dari opsi berikut.

- Buat bucket S3 baru — Opsi ini membuat bucket Amazon S3 di akun Anda untuk hasil perhitungan Anda. Nama bucket memiliki format `account_id-region-athena-results-bucket-alphanumeric_id` dan menggunakan pengaturan ACL dinonaktifkan, akses publik diblokir, versi dinonaktifkan, dan pemilik bucket diberlakukan.
- Pilih lokasi S3 yang ada — Untuk opsi ini, lakukan hal berikut:
  - Masukkan jalur S3 ke lokasi yang sudah ada di kotak pencarian, atau pilih Browse S3 untuk memilih bucket dari daftar.

> **Note**
>
> Saat Anda memilih lokasi yang ada di Amazon S3, jangan tambahkan garis miring (/) ke lokasi. Melakukan hal itu menyebabkan tautan ke lokasi hasil perhitungan pada halaman detail perhitungan mengarah ke direktori yang salah. Jika ini terjadi, edit lokasi hasil grup kerja untuk menghapus garis miring ke depan.

- (Opsional) Pilih Lihat untuk membuka halaman Bucket di konsol Amazon S3 tempat Anda dapat melihat informasi selengkapnya tentang bucket yang sudah ada yang Anda pilih.
- (Opsional) Untuk pemilik bucket yang diharapkan, masukkan ID AWS akun yang Anda harapkan sebagai pemilik bucket lokasi keluaran hasil kueri Anda. Kami menyarankan Anda memilih opsi ini sebagai tindakan keamanan tambahan bila memungkinkan. Jika ID akun pemilik bucket tidak cocok dengan ID yang Anda tentukan, upaya untuk menampilkan ke bucket akan gagal. Untuk informasi lebih lanjut, lihat Memverifikasi kepemilikan bucket dengan kondisi pemilik bucket di Panduan Pengguna Amazon S3.
- (Opsional) Pilih Tetapkan kontrol penuh pemilik bucket atas hasil kueri jika lokasi hasil perhitungan Anda dimiliki oleh akun lain dan Anda ingin memberikan kontrol penuh atas hasil kueri Anda ke akun lain.

3. (Opsional) Pilih Enkripsi hasil perhitungan, lalu pilih salah satu dari berikut ini:

   - SSE_S3 - Ini adalah kunci enkripsi sisi server yang dikelola S3.
   - SSE_KMS — Kunci yang Anda berikan. Untuk Pilih AWS KMS kunci, Anda dapat memilih salah satu dari berikut ini:
     - Gunakan kunci yang AWS dimiliki — Gunakan kunci yang AWS memiliki dan mengelola untuk Anda.
     - Pilih AWS KMS kunci yang berbeda (lanjutan) - Pilih atau buat kunci.
       - Untuk menggunakan kunci yang ada, gunakan kotak pencarian untuk memilih AWS KMS atau memasukkan kunci ARN.
       - Untuk membuat kunci di konsol KMS, pilih Buat AWS KMS kunci. Setelah Anda selesai membuat kunci di konsol KMS, kembali ke halaman Buat grup kerja di konsol Athena, lalu gunakan tombol Pilih tombol atau masukkan kotak pencarian ARN untuk memilih AWS KMS kunci yang baru saja Anda buat.

4.

(Opsional) Pengaturan lainnya - Perluas opsi ini untuk mengaktifkan atau menonaktifkan opsi Publikasikan CloudWatch metrik untuk grup kerja. Bidang ini dipilih secara default. Untuk informasi selengkapnya, lihat Memantau perhitungan Apache Spark denganCloudWatchmetrik.

5. (Opsional) Tag - Gunakan opsi ini untuk menambahkan tag ke workgroup Anda. Untuk informasi selengkapnya, lihat Menandai sumber daya Athena.
6. Pilih Buat grup kerja. Pesan memberi tahu Anda bahwa workgroup berhasil dibuat, dan workgroup ditampilkan dalam daftar workgroup.

Membuka penjelajah notebook dan beralih kelompok kerja

Sebelum Anda dapat menggunakan workgroup yang diaktifkan Spark yang baru saja Anda buat, Anda harus beralih ke workgroup. Untuk mengganti workgroup yang diaktifkan Spark, Anda dapat menggunakan opsi Workgroup di Notebook explorer atau editor Notebook.

> **Note**
>
> Sebelum Anda mulai, periksa apakah browser Anda tidak memblokir cookie pihak ketiga. Browser apa pun yang memblokir cookie pihak ketiga baik secara default atau sebagai pengaturan yang diaktifkan pengguna akan mencegah notebook diluncurkan. Untuk informasi selengkapnya tentang mengelola cookie, lihat:
>
> - Chrome
> - Firefox
> - Safari

Untuk membuka penjelajah notebook dan beralih kelompok kerja

1. Di panel navigasi, pilih Notebook explorer.
2. Gunakan opsi Workgroup di kanan atas konsol untuk memilih workgroup yang diaktifkan Spark yang Anda buat. Contoh notebook ditampilkan dalam daftar notebook.

   Anda dapat menggunakan penjelajah notebook dengan cara berikut:

   - Pilih nama buku catatan yang ditautkan untuk membuka buku catatan di sesi baru.

- Untuk mengganti nama, menghapus, atau mengekspor buku catatan Anda, gunakan menu Tindakan.
- Untuk mengimpor file notebook, pilih Impor file.
- Untuk membuat buku catatan, pilih Buat buku catatan.

Menjalankan contoh notebook

Notebook sampel menanyakan data dari dataset perjalanan taksi New York City yang tersedia untuk umum. Notebook ini memiliki contoh yang menunjukkan cara bekerja dengan Spark DataFrames, Spark SQL, dan. AWS Glue Data Catalog

Untuk menjalankan contoh notebook

1. Di Notebook explorer, pilih nama tertaut dari contoh notebook.

   Ini memulai sesi notebook dengan parameter default dan membuka notebook di editor notebook. Sebuah pesan memberi tahu Anda bahwa sesi Apache Spark baru telah dimulai menggunakan parameter default (20 DPU maksimum).
2. Untuk menjalankan sel secara berurutan dan mengamati hasilnya, pilih tombol Run sekali untuk setiap sel di notebook.

   - Gulir ke bawah untuk melihat hasilnya dan tampilkan sel baru.
   - Untuk sel yang memiliki perhitungan, bilah kemajuan menunjukkan persentase selesai, waktu berlalu, dan waktu yang tersisa.
   - Notebook contoh membuat database sampel dan tabel di akun Anda. Sel terakhir menghilangkan ini sebagai langkah pembersihan.

> **Note**
>
> Jika Anda mengubah nama folder, tabel, atau database di buku catatan contoh, pastikan perubahan tersebut tercermin dalam peran IAM yang Anda gunakan. Jika tidak, notebook dapat gagal berjalan karena izin yang tidak mencukupi.

Mengedit detail sesi

Setelah memulai sesi buku catatan, Anda dapat mengedit detail sesi seperti format tabel, enkripsi, batas waktu idle sesi, dan jumlah maksimum unit pemrosesan data (DPU) bersamaan yang ingin Anda gunakan. DPU adalah ukuran relatif daya pemrosesan yang terdiri dari 4 vCPU kapasitas komputasi dan 16 GB memori.

Untuk mengedit detail sesi

1. Di editor buku catatan, dari menu Sesi di kanan atas, pilih Edit sesi.
2. Dalam kotak dialog Edit detail sesi, di bagian Properti percikan, pilih atau masukkan nilai untuk opsi berikut:

   - Format tabel tambahan — Pilih Linux Foundation Delta Lake, Apache Hudi, Apache Iceberg, atau Custom.
     - Untuk opsi tabel Delta, Hudi, atau Iceberg, properti tabel yang diperlukan untuk format tabel yang sesuai secara otomatis disediakan untuk Anda dalam opsi Edit dalam tabel dan Edit di JSON. Untuk informasi selengkapnya tentang menggunakan format tabel ini, lihatMenggunakan format tabel non-Hive di Amazon Athena untuk Apache Spark.
     - Untuk menambah atau menghapus properti tabel untuk Kustom atau jenis tabel lainnya, gunakan opsi Edit dalam tabel dan Edit di JSON.
     - Untuk opsi Edit dalam tabel, pilih Tambahkan properti untuk menambahkan properti, atau Hapus untuk menghapus properti. Untuk memasukkan nama properti dan nilainya, gunakan kotak Kunci dan Nilai.
     - Untuk opsi Edit di JSON, gunakan editor teks JSON untuk mengedit konfigurasi secara langsung.
       - Untuk menyalin teks JSON ke clipboard, pilih Salin.
       - Untuk menghapus semua teks dari editor JSON, pilih Hapus.
       - Untuk mengonfigurasi pembungkus garis atau memilih tema warna untuk editor JSON, pilih ikon pengaturan (roda gigi).
   - Aktifkan enkripsi Spark - — Pilih opsi ini untuk mengenkripsi data yang ditulis ke disk dan dikirim melalui node jaringan Spark. Untuk informasi selengkapnya, lihat Mengaktifkan enkripsi Apache Spark.
3. Di bagian Parameter sesi, pilih atau masukkan nilai untuk opsi berikut:

- Waktu tunggu siaga sesi - Pilih atau masukkan nilai antara 1 dan 480 menit. Defaultnya adalah 20.
- Ukuran koordinator - Koordinator adalah eksekutor khusus yang mengatur pekerjaan pemrosesan dan mengelola pelaksana lain dalam sesi notebook. Saat ini, 1 DPU adalah nilai default dan hanya mungkin.
- Ukuran pelaksana - Eksekutor adalah unit komputasi terkecil yang dapat diminta oleh sesi notebook dari Athena. Saat ini, 1 DPU adalah nilai default dan hanya mungkin.
- Nilai konkuren maksimum - Jumlah maksimum DPU yang dapat berjalan secara bersamaan. Defaultnya adalah 20, minimum adalah 3, dan maksimum adalah 60. Meningkatkan nilai ini tidak secara otomatis mengalokasikan sumber daya tambahan, tetapi Athena akan berusaha mengalokasikan hingga maksimum yang ditentukan ketika beban komputasi memerlukannya dan ketika sumber daya tersedia.

4. Pilih Simpan.
5. Pada prompt Konfirmasi edit, pilih Konfirmasi.

   Athena menyimpan notebook Anda dan memulai sesi baru dengan parameter yang Anda tentukan. Spanduk di editor notebook memberi tahu Anda bahwa sesi baru telah dimulai dengan parameter yang dimodifikasi.

> **Note**
>
> Athena mengingat pengaturan sesi Anda untuk notebook. Jika Anda mengedit parameter sesi dan kemudian mengakhiri sesi, Athena menggunakan parameter sesi yang Anda konfigurasikan saat berikutnya Anda memulai sesi untuk buku catatan.

Melihat sesi dan detail perhitungan

Setelah Anda menjalankan buku catatan, Anda dapat melihat sesi dan detail perhitungan Anda.

Untuk melihat detail sesi dan perhitungan

1. Dari menu Sesi di kanan atas, pilih Lihat detail.

   - Tab Sesi saat ini menampilkan informasi tentang sesi saat ini, termasuk ID sesi, waktu pembuatan, status, dan grup kerja.

- Tab Riwayat mencantumkan ID sesi untuk sesi sebelumnya. Untuk melihat detail sesi sebelumnya, pilih tab Riwayat, lalu pilih ID sesi dalam daftar.
- Bagian Perhitungan menunjukkan daftar perhitungan yang berjalan di sesi.

2. Untuk melihat detail perhitungan, pilih ID perhitungan.
3. Pada halaman Detail perhitungan, Anda dapat melakukan hal berikut:

- Untuk melihat kode perhitungan, lihat bagian Kode.
- Untuk melihat hasil perhitungan, pilih tab Hasil.
- Untuk mengunduh hasil yang Anda lihat dalam format teks, pilih Unduh hasil.
- Untuk melihat informasi tentang hasil perhitungan di Amazon S3, pilih Lihat di S3.

Mengakhiri sesi

Untuk mengakhiri sesi notebook

1. Di editor buku catatan, dari menu Sesi di kanan atas, pilih Terminate.
2. Pada prompt Konfirmasi penghentian sesi, pilih Konfirmasi. Notebook Anda disimpan dan Anda dikembalikan ke editor notebook.

> **Note**
>
> Menutup tab notebook di editor notebook tidak dengan sendirinya mengakhiri sesi untuk notebook aktif. Jika Anda ingin memastikan bahwa sesi dihentikan, gunakan opsi Session, Terminate.

Membuat buku catatan Anda sendiri

Setelah Anda membuat workgroup Athena yang diaktifkan Spark, Anda dapat membuat buku catatan Anda sendiri.

Untuk membuat buku catatan

1. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.
2. Di panel navigasi konsol Athena, pilih Notebook explorer atau Notebook editor.

3. Lakukan salah satu tindakan berikut:

   - Di penjelajah Notebook, pilih Buat buku catatan.
   - Di editor Notebook, pilih Buat buku catatan, atau pilih ikon plus (+) untuk menambahkan buku catatan.

4. Di kotak dialog Buat buku catatan, untuk nama Notebook, masukkan nama.

5. (Opsional) Perluas properti Spark, lalu pilih atau masukkan nilai untuk opsi berikut:

   - Format tabel tambahan — Pilih Linux Foundation Delta Lake, Apache Hudi, Apache Iceberg, atau Custom.
     - Untuk opsi tabel Delta, Hudi, atau Iceberg, properti tabel yang diperlukan untuk format tabel yang sesuai secara otomatis disediakan untuk Anda dalam opsi Edit dalam tabel dan Edit di JSON. Untuk informasi selengkapnya tentang menggunakan format tabel ini, lihatMenggunakan format tabel non-Hive di Amazon Athena untuk Apache Spark.
     - Untuk menambah atau menghapus properti tabel untuk Kustom atau jenis tabel lainnya, gunakan opsi Edit dalam tabel dan Edit di JSON.
     - Untuk opsi Edit dalam tabel, pilih Tambahkan properti untuk menambahkan properti, atau Hapus untuk menghapus properti. Untuk memasukkan nama properti dan nilainya, gunakan kotak Kunci dan Nilai.
     - Untuk opsi Edit di JSON, gunakan editor teks JSON untuk mengedit konfigurasi secara langsung.
       - Untuk menyalin teks JSON ke clipboard, pilih Salin.
       - Untuk menghapus semua teks dari editor JSON, pilih Hapus.
       - Untuk mengonfigurasi pembungkus garis atau memilih tema warna untuk editor JSON, pilih ikon pengaturan (roda gigi).
   - Aktifkan enkripsi Spark - — Pilih opsi ini untuk mengenkripsi data yang ditulis ke disk dan dikirim melalui node jaringan Spark. Untuk informasi selengkapnya, lihat Mengaktifkan enkripsi Apache Spark.

6. (Opsional) Perluas parameter Sesi, lalu pilih atau masukkan nilai untuk opsi berikut:

   - Waktu tunggu siaga sesi - pilih atau masukkan nilai antara 1 dan 480 menit. Defaultnya adalah 20.
   - Ukuran koordinator - Koordinator adalah eksekutor khusus yang mengatur pekerjaan pemrosesan dan mengelola pelaksana lain dalam sesi notebook. Saat ini, 1 DPU adalah

nilai default dan hanya mungkin. DPU (data processing unit) adalah ukuran relatif daya pemrosesan yang terdiri dari 4 vCPU kapasitas komputasi dan 16 GB memori.

- Ukuran pelaksana - Eksekutor adalah unit komputasi terkecil yang dapat diminta oleh sesi notebook dari Athena. Saat ini, 1 DPU adalah nilai default dan hanya mungkin.
- Nilai konkuren maksimum - Jumlah maksimum DPU yang dapat berjalan secara bersamaan. Defaultnya adalah 20 dan maksimum adalah 60. Meningkatkan nilai ini tidak secara otomatis mengalokasikan sumber daya tambahan, tetapi Athena akan berusaha mengalokasikan hingga maksimum yang ditentukan ketika beban komputasi memerlukannya dan ketika sumber daya tersedia.

7. Pilih Buat. Notebook Anda terbuka di sesi baru di editor notebook.

Membuka buku catatan yang dibuat sebelumnya

Untuk membuka notebook yang dibuat sebelumnya

1. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.
2. Di panel navigasi konsol Athena, pilih Editor buku catatan atau penjelajah Notebook.
3. Lakukan salah satu tindakan berikut:

   - Di Editor buku catatan, pilih buku catatan di buku catatan terbaru atau daftar buku catatan tersimpan. Notebook dibuka di sesi baru.
   - Di Notebook explorer, pilih nama buku catatan dalam daftar. Notebook dibuka di sesi baru.

Untuk informasi selengkapnya tentang mengelola file buku catatan Anda, lihatMengelola file notebook.

Bekerja dengan notebook

Anda mengelola buku catatan Anda di penjelajah notebook Athena dan mengedit dan menjalankannya dalam sesi menggunakan editor notebook Athena. Anda dapat mengonfigurasi penggunaan DPU untuk sesi notebook Anda sesuai dengan kebutuhan Anda.

Saat Anda menghentikan buku catatan, Anda mengakhiri sesi terkait. Semua file disimpan, tetapi perubahan sedang berlangsung dalam variabel, fungsi, dan kelas yang dideklarasikan hilang. Saat Anda me-restart notebook, Athena memuat ulang file notebook dan Anda dapat menjalankan kode Anda lagi.

Sesi dan perhitungan

Setiap notebook dikaitkan dengan satu kernel Python dan menjalankan kode Python. Notebook dapat memiliki satu atau lebih sel yang berisi perintah. Untuk menjalankan sel di buku catatan, pertama-tama Anda membuat sesi untuk buku catatan. Sesi melacak variabel dan status notebook.

Menjalankan sel di buku catatan berarti menjalankan perhitungan di sesi saat ini. Perhitungan memajukan status notebook dan dapat melakukan tugas-tugas seperti membaca dari Amazon S3 atau menulis ke penyimpanan data lainnya. Selama sesi berjalan, perhitungan menggunakan dan memodifikasi status yang dipertahankan untuk notebook.

Ketika Anda tidak lagi membutuhkan negara, Anda dapat mengakhiri sesi. Ketika Anda mengakhiri sesi, buku catatan tetap ada, tetapi variabel dan informasi status lainnya dihancurkan. Jika Anda perlu mengerjakan beberapa proyek secara bersamaan, Anda dapat membuat sesi untuk setiap proyek, dan sesi akan independen satu sama lain.

Sesi memiliki kapasitas komputasi khusus, diukur dalam DPU. Saat Anda membuat sesi, Anda dapat menetapkan sesi sejumlah DPU. Sesi yang berbeda dapat memiliki kapasitas yang berbeda tergantung pada persyaratan tugas.

Menggunakan editor notebook Athena

Editor notebook Athena adalah lingkungan interaktif untuk menulis dan menjalankan kode. Bagian berikut menjelaskan fitur lingkungan.

Mode perintah vs. mode edit

Editor notebook memiliki antarmuka pengguna modal: mode edit untuk memasukkan teks ke dalam sel, dan mode perintah untuk mengeluarkan perintah ke editor itu sendiri seperti salin, tempel, atau jalankan.

Untuk menggunakan mode edit dan mode perintah, Anda dapat melakukan tugas-tugas berikut:

- Untuk masuk ke mode edit, tekan**`ENTER`**, atau pilih sel. Ketika sel dalam mode edit, sel memiliki margin kiri hijau.
- Untuk masuk ke mode perintah, tekan**`ESC`**, atau klik di luar sel. Perhatikan bahwa perintah biasanya hanya berlaku untuk sel yang dipilih saat ini, bukan untuk semua sel. Ketika editor dalam mode perintah, sel memiliki margin kiri biru.

- Dalam mode perintah, Anda dapat menggunakan pintasan keyboard dan menu di atas editor, tetapi tidak memasukkan teks ke dalam sel individual.
- Untuk memilih sel, pilih sel.
- Untuk memilih semua sel, tekan **Ctrl+A** (Windows) atau **Cmd+A** (Mac).

Menu editor buku catatan

Ikon di menu di bagian atas editor notebook menawarkan opsi berikut:

- Simpan - Menyimpan status notebook saat ini.
- Masukkan sel di bawah ini - Menambahkan sel baru (kosong) di bawah yang saat ini dipilih.
- Potong sel yang dipilih - Menghapus sel yang dipilih dari lokasi saat ini dan menyalin sel ke memori.
- Salin sel yang dipilih - Salin sel yang dipilih ke memori.
- Tempel sel di bawah ini — Tempelkan sel yang disalin di bawah sel saat ini.
- Pindahkan sel yang dipilih ke atas - Memindahkan sel saat ini di atas sel di atas.
- Pindahkan sel yang dipilih ke bawah - Memindahkan sel saat ini di bawah sel di bawah ini.
- Jalankan - Menjalankan sel saat ini (dipilih). Output ditampilkan tepat di bawah sel saat ini.
- Jalankan semua — Menjalankan semua sel di notebook. Output untuk setiap sel ditampilkan tepat di bawah sel.
- Stop (Interrupt the kernel) — Menghentikan notebook saat ini dengan menginterupsi kernel.
- Opsi format - Memilih format sel, yang dapat menjadi salah satu dari berikut ini:
  - Kode - Gunakan untuk kode Python (default).
  - Markdown — Gunakan untuk memasukkan teks dalam format penurunan harga GitHub -style. Untuk membuat penurunan harga, jalankan sel.
  - Raw NbConvert - Gunakan untuk memasukkan teks dalam bentuk yang tidak dimodifikasi. Sel yang ditandai sebagai Raw NbConvert dapat diubah menjadi format yang berbeda seperti HTML oleh alat baris perintah Jupyter nbconvert.
- Heading — Gunakan untuk mengubah level heading sel.
- Palet perintah - Berisi perintah notebook Jupyter dan pintasan keyboard mereka. Untuk informasi selengkapnya tentang pintasan keyboard, lihat bagian nanti dalam dokumen ini.
- Sesi — Gunakan opsi di menu ini untuk melihat detail sesi, mengedit parameter sesi, atau mengakhiri sesi.

Pintasan keyboard mode perintah

Berikut ini adalah beberapa pintasan keyboard mode perintah editor notebook umum. Pintasan ini tersedia setelah menekan **ESC** untuk masuk ke mode perintah. Untuk melihat daftar lengkap perintah yang tersedia di editor, tekan**ESC + H**.

Kunci	Tindakan
1 - 6	Ubah jenis sel menjadi penurunan harga dan atur level heading ke nomor yang diketik
a	Buat sel di atas sel saat ini
b	Buat sel di bawah sel saat ini
c	Salin sel saat ini ke memori
d d	Hapus sel saat ini
h	Menampilkan layar bantuan pintasan keyboard
j	Turun satu sel
k	Pergi satu sel
m	Ubah format sel saat ini menjadi penurunan harga
r	Ubah format sel saat ini menjadi mentah
s	Simpan buku catatan
v	Tempel konten memori di bawah sel saat ini
x	Potong sel atau sel yang dipilih
y	Ubah format sel menjadi kode
z	Urungkan
Ctrl+Enter	Jalankan sel saat ini dan masuk ke mode perintah

Kunci	Tindakan
Shift+Enter atau **Alt+Enter**	Jalankan sel saat ini dan buat sel baru di bawah output, dan masukkan sel baru dalam mode edit
Space	Halaman bawah
Shift+Space	Halaman ke atas
Shift + L	Alihkan visibilitas nomor baris dalam sel

Mengedit pintasan mode perintah

Editor notebook memiliki opsi untuk menyesuaikan pintasan keyboard mode perintah.

Untuk mengedit pintasan mode perintah

1. Dari menu editor notebook, pilih palet Command.
2. Dari palet perintah, pilih perintah pintasan keyboard mode perintah Edit.
3. Gunakan antarmuka pintasan mode perintah Edit untuk memetakan atau memetakan ulang perintah yang Anda inginkan ke keyboard.

   Untuk melihat instruksi untuk mengedit pintasan mode perintah, gulir ke bagian bawah layar pintasan mode perintah Edit.

Untuk informasi tentang menggunakan perintah sihir di Athena untuk Apache Spark, lihat. Menggunakan perintah ajaib

Menggunakan perintah ajaib

Perintah ajaib, atausihir, adalah perintah khusus yang dapat Anda jalankan di sel notebook. Sebagai contoh,`%env`menunjukkan variabel lingkungan dalam sesi notebook. Athena mendukung fungsi ajaib di iPython 6.0.3.

Bagian ini menunjukkan beberapa perintah sihir kunci di Athena untuk Apache Spark.

- Untuk melihat daftar perintah ajaib di Athena, jalankan perintah**`%lsmagic`**di sel notebook.

- Untuk informasi tentang menggunakan sihir untuk membuat grafik di notebook Athena, lihatMagics untuk membuat grafik data.
- Untuk informasi tentang perintah ajaib tambahan, lihatPerintah ajaib bawaandalam dokumentasi IPython.

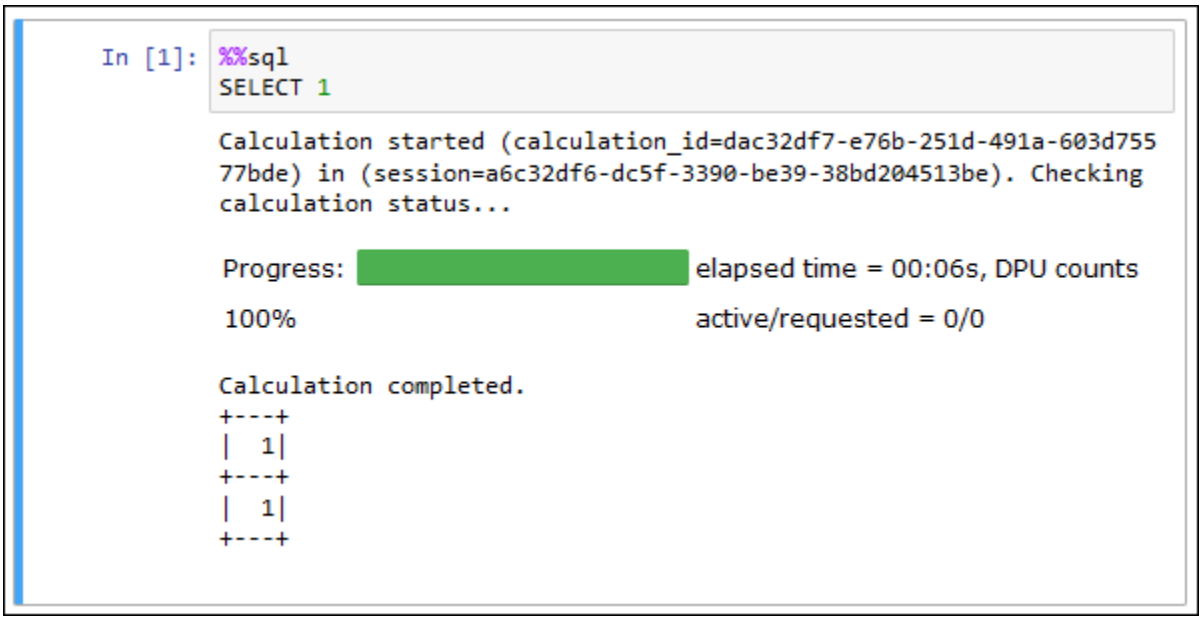
> **Note**
>
> Saat ini,`%pip`perintah gagal saat dieksekusi. Ini adalah masalah yang diketahui.

Sihir sel

Sihir yang ditulis pada beberapa baris didahului oleh tanda persen ganda (`%%`) dan disebutfungsi sihir selatausihir sel.

%%sql

Sihir sel ini memungkinkan untuk menjalankan pernyataan SQL secara langsung tanpa harus menghiasnya dengan pernyataan Spark SQL. Perintah ini juga menampilkan output dengan memanggil secara implisit`.show()`pada dataframe yang dikembalikan.



Yang`%%sql`perintah auto memotong output kolom dengan lebar 20 karakter. Saat ini, pengaturan ini tidak dapat dikonfigurasi. Untuk mengatasi batasan ini, gunakan sintaks lengkap berikut dan ubah parameter`show`metode yang sesuai.

```
spark.sql("""YOUR_SQL""").show(n=number, truncate=number, vertical=bool)
```

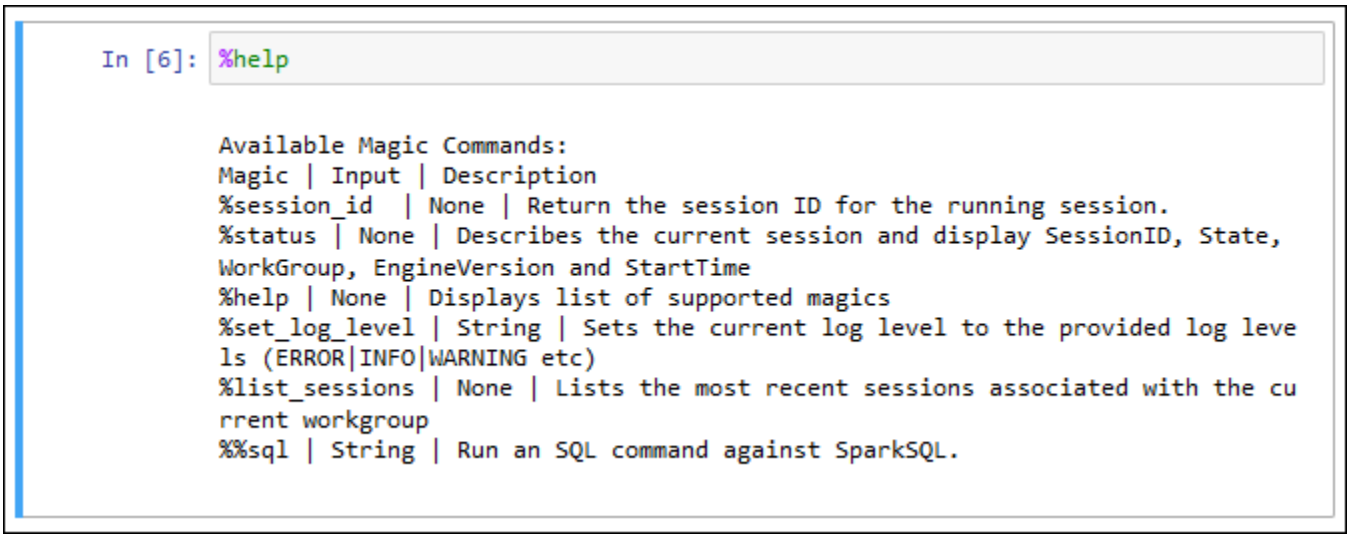
- n `int`, opsional. Jumlah baris untuk ditampilkan.
- memotong—`boolatauint`, opsional - Jika`true`, memotong string lebih dari 20 karakter. Ketika diatur ke angka yang lebih besar dari 1, memotong string panjang dengan panjang yang ditentukan dan sel sejajar kanan.
- vertikal—`bool`, opsional. Jika`true`, mencetak baris keluaran secara vertikal (satu baris per nilai kolom).

Garis Magics

Sihir yang berada pada satu baris didahului oleh tanda persen (%) dan disebutfungsi sihir garisatausihir garis.

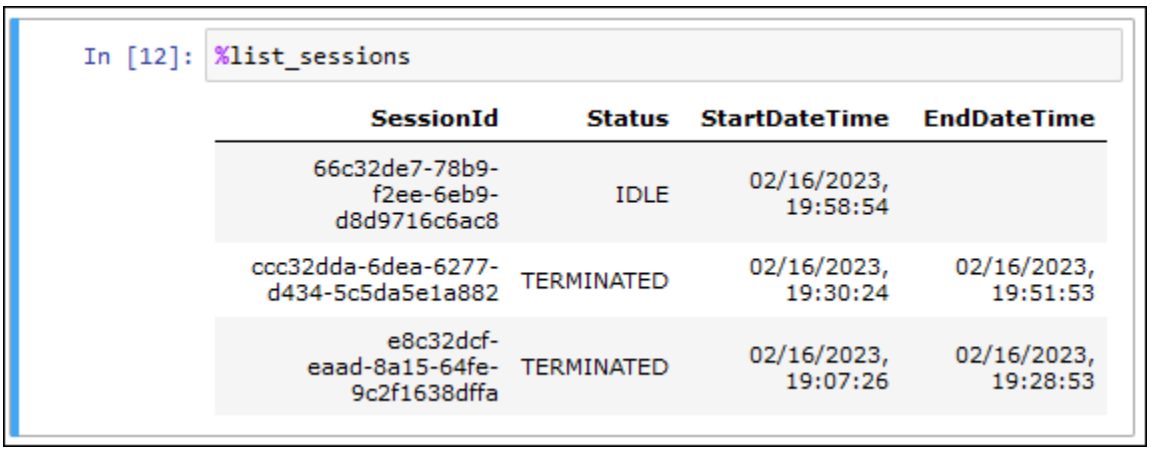
%bantuan

Menampilkan deskripsi dari perintah ajaib yang tersedia.



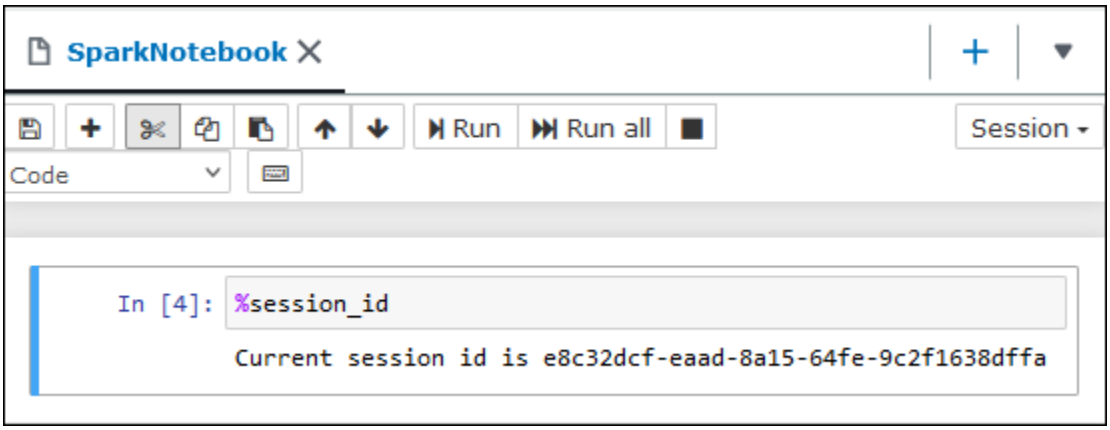
%list_session

Daftar sesi yang terkait dengan notebook. Informasi untuk setiap sesi mencakup ID sesi, status sesi, dan tanggal dan waktu sesi dimulai dan berakhir.



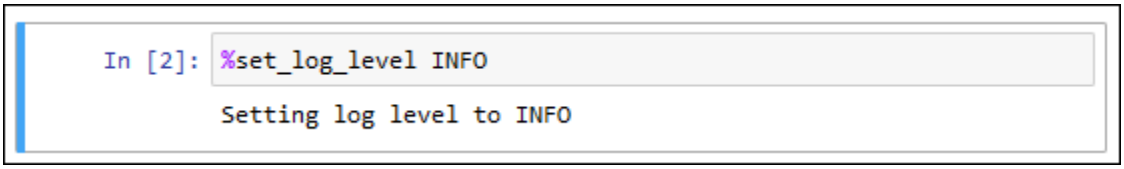
%session_id

Mengambil ID sesi saat ini.



%set_log_level

Set atau me-reset logger untuk menggunakan tingkat log yang ditentukan. Nilai yang mungkin adalah`DEBUG`,`ERROR`,`FATAL`,`INFO`, dan`WARN`atau`WARNING`. Nilai harus huruf besar dan tidak boleh dilampirkan dalam tanda kutip tunggal atau ganda.



%status

Menjelaskan sesi saat ini. Outputnya meliputi ID sesi, status sesi, nama workgroup,PySparkversi mesin, dan waktu mulai sesi. Perintah ajaib ini membutuhkan sesi aktif untuk mengambil rincian sesi.

Berikut ini adalah nilai yang mungkin untuk status:

MENCIPTAKAN— Sesi sedang dimulai, termasuk memperoleh sumber daya.

DIBUAT— Sesi telah dimulai.

MENGANGGUR— Sesi ini dapat menerima perhitungan.

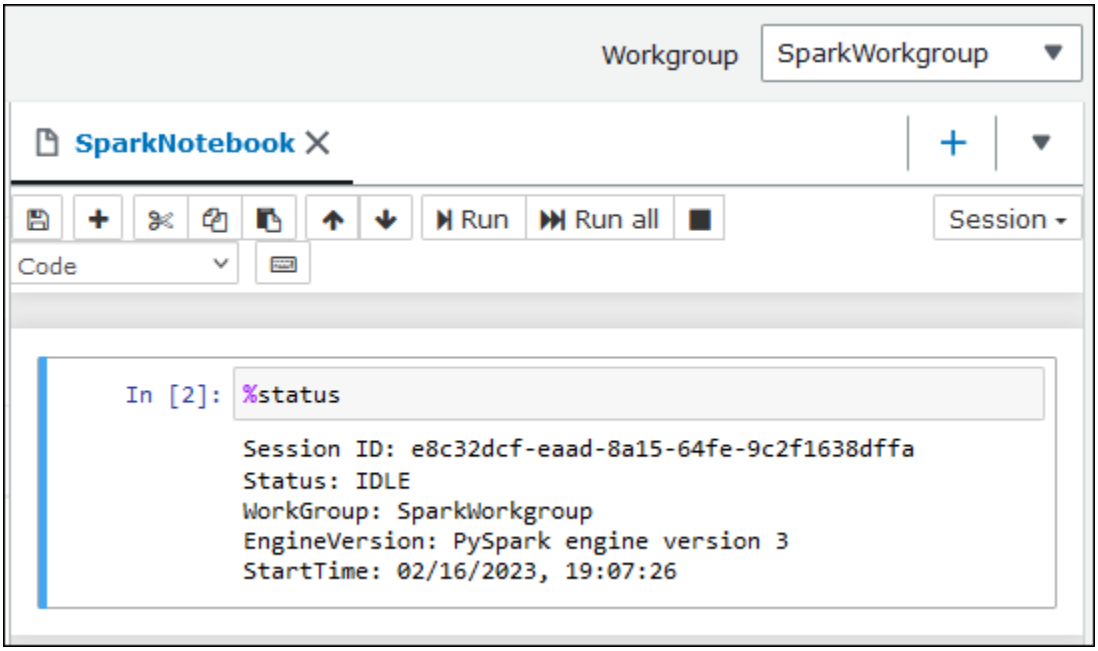
SIBUK- Sesi ini memproses tugas lain dan tidak dapat menerima perhitungan.

MENGAKHIRI— Sesi sedang dalam proses mematikan.

MENGAKHIRI- Sesi dan sumber dayanya tidak lagi berjalan.

TERDEGRADASI— Sesi ini tidak memiliki koordinator yang sehat.

GAGAL- Karena kegagalan, sesi dan sumber dayanya tidak lagi berjalan.



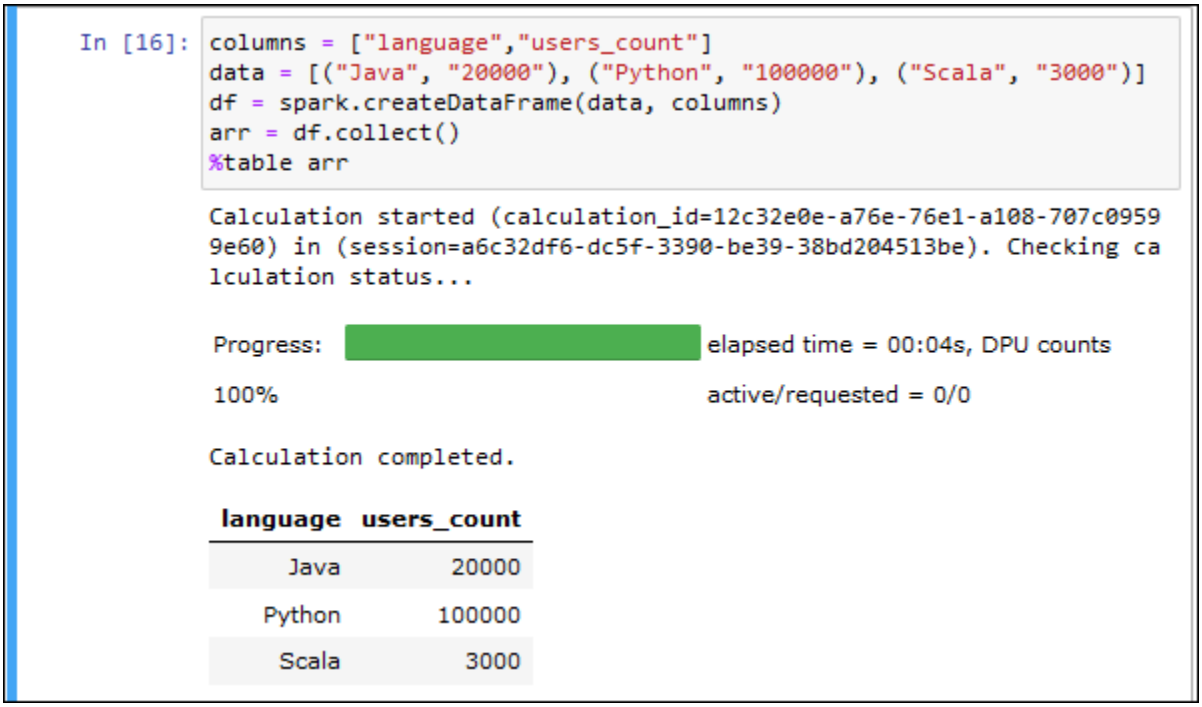
Magics untuk membuat grafik data

Sihir garis di bagian ini mengkhususkan diri dalam rendering data untuk jenis data tertentu atau dalam hubungannya dengan perpustakaan grafik.

%tabel

Anda dapat menggunakan`%table`perintah ajaib untuk menampilkan data dataframe dalam format tabel.

Contoh berikut membuat dataframe dengan dua kolom dan tiga baris data, kemudian menampilkan data dalam format tabel.



%matplot

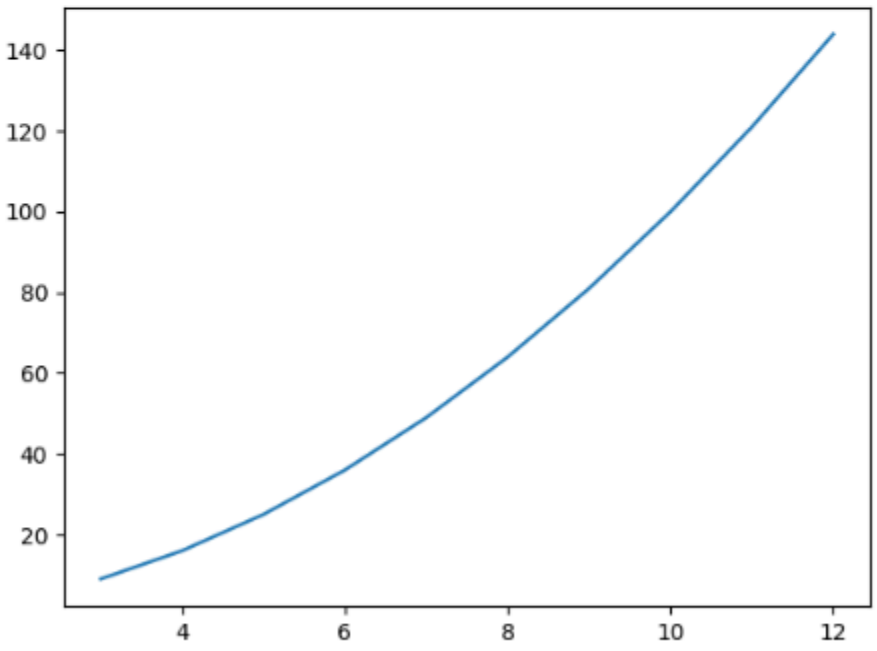
Matplotlibadalah perpustakaan yang komprehensif untuk membuat visualisasi statis, animasi, dan interaktif dengan Python. Anda dapat menggunakan`%matplot`perintah ajaib untuk membuat grafik setelah Anda mengimpor perpustakaan matplotlib ke dalam sel notebook.

Contoh berikut mengimpor perpustakaan matplotlib, menciptakan satu set x dan y koordinat, dan kemudian menggunakan penggunaan`%matplot`perintah sihir untuk membuat grafik poin.

```
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>]
```

Menggunakan perpustakaan matplotlib dan seaborn bersama-sama

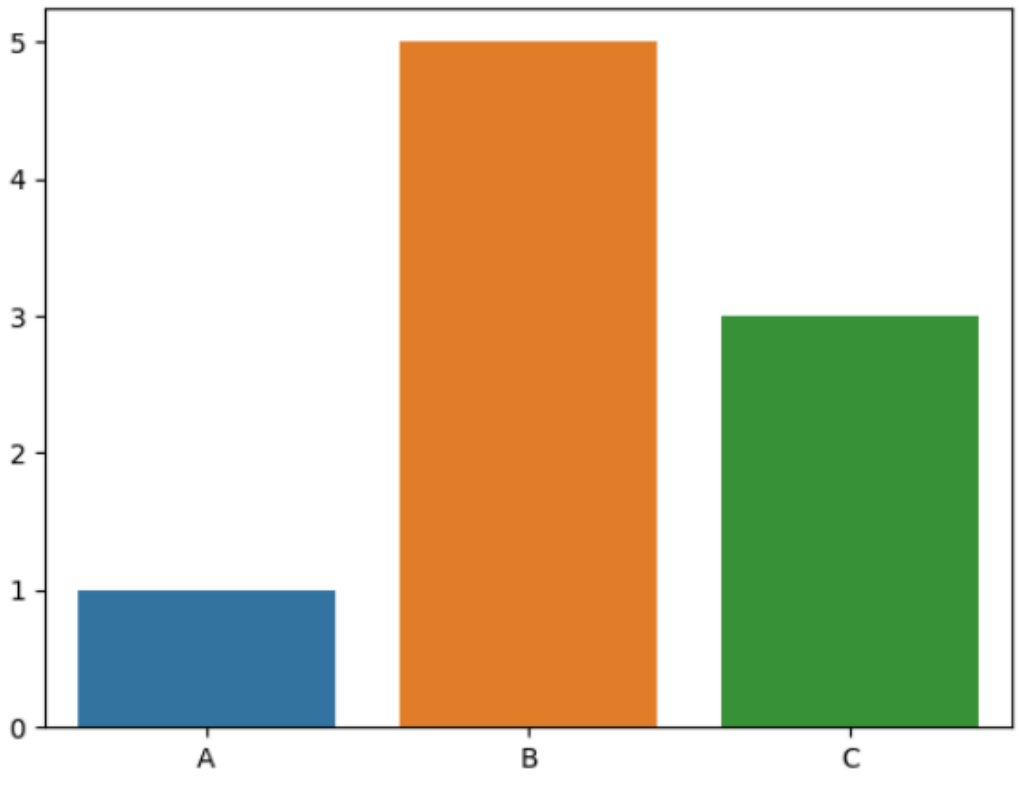
Seabornadalah perpustakaan untuk membuat grafik statistik dengan Python. Ini dibangun di atas matplotlib dan terintegrasi erat denganpanda(Analisis data Python) struktur data. Anda juga dapat menggunakan`%matplot`perintah ajaib untuk membuat data seaborn.

Contoh berikut menggunakan kedua pustaka matplotlib dan seaborn untuk membuat grafik batang sederhana.

```
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.
```

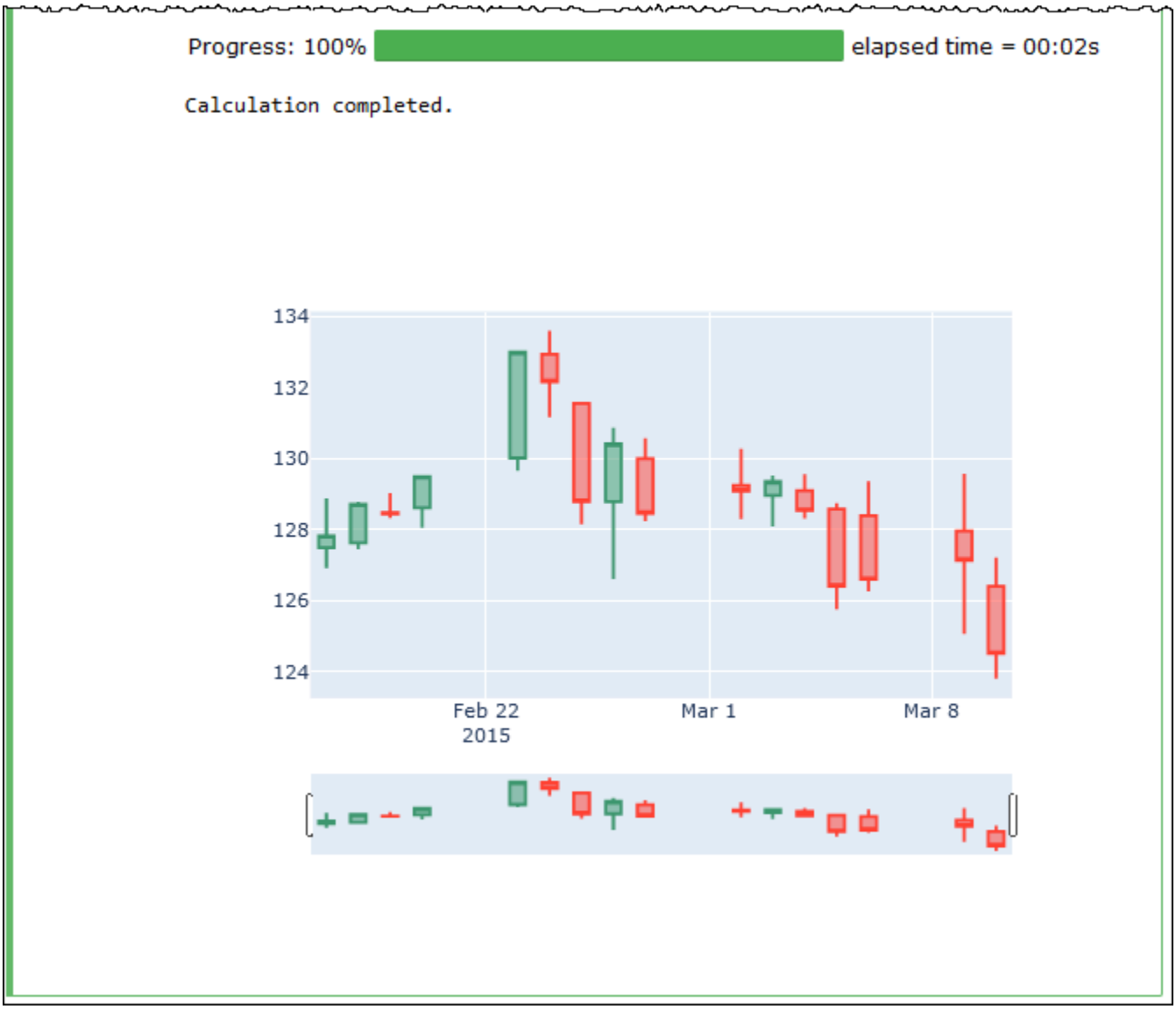


%plotal

Plotlyadalah perpustakaan grafik open source untuk Python yang dapat Anda gunakan untuk membuat grafik interaktif. Anda menggunakan`%plotly`perintah sihir untuk membuat data ploty.

Contoh berikut menggunakanStringIo, plotly, dan panda perpustakaan pada data harga saham untuk membuat grafik aktivitas saham dari Februari dan Maret 2015.

```
from io import StringIO
csvString = """
Date,AAPL.Open,AAPL.High,AAPL.Low,AAPL.Close,AAPL.Volume,AAPL.Adjusted,dn,mavg,up,direction
2015-02-17,127.489998,128.880005,126.919998,127.830002,63152400,122.905254,106.7410523,117.9276
2015-02-18,127.629997,128.779999,127.449997,128.720001,44891700,123.760965,107.842423,118.94033
2015-02-19,128.479996,129.029999,128.330002,128.449997,37362400,123.501363,108.8942449,119.8891
2015-02-20,128.619995,129.5,128.050003,129.5,48948400,124.510914,109.7854494,120.7635001,131.74
2015-02-23,130.020004,133,129.660004,133,70974100,127.876074,110.3725162,121.7201668,133.067817
2015-02-24,132.940002,133.600006,131.169998,132.169998,69228100,127.078049,111.0948689,122.6648
2015-02-25,131.559998,131.600006,128.149994,128.789993,74711700,123.828261,113.2119183,123.6296
2015-02-26,128.789993,130.869995,126.610001,130.419998,91287500,125.395469,114.1652991,124.2823
2015-02-27,130,130.570007,128.240005,128.460007,62014800,123.510987,114.9668484,124.8426669,134
2015-03-02,129.25,130.279999,128.300003,129.089996,48096700,124.116706,115.8770904,125.4036668,
2015-03-03,128.960007,129.520004,128.089996,129.360001,37816300,124.376308,116.9535132,125.9551
2015-03-04,129.100006,129.559998,128.320007,128.539993,31666300,123.587892,118.0874253,126.4730
2015-03-05,128.580002,128.75,125.760002,126.410004,56517100,121.539962,119.1048311,126.848667,1
2015-03-06,128.399994,129.369995,126.260002,126.599998,72842100,121.722637,120.190797,127.22883
2015-03-09,127.959999,129.570007,125.059998,127.139999,88528500,122.241834,121.6289771,127.6311
2015-03-10,126.410004,127.220001,123.800003,124.510002,68856600,119.71316,123.1164763,127.92350
"""
csvStringIO = StringIO(csvString)

from io import StringIO
import plotly.graph_objects as go
import pandas as pd
from datetime import datetime
df = pd.read_csv(csvStringIO)
fig = go.Figure(data=[go.Candlestick(x=df['Date'],
open=df['AAPL.Open'],
high=df['AAPL.High'],
low=df['AAPL.Low'],
close=df['AAPL.Close'])])
%plotly fig
```



Mengelola file notebook

Selain menggunakan penjelajah notebook untukberkaryadanterbukanotebook, Anda juga dapat menggunakannya untuk mengganti nama, menghapus, mengekspor, atau mengimpor notebook, atau melihat riwayat sesi untuk notebook.

Mengganti nama buku catatan

1. Mengakhirisetiap sesi aktif untuk notebook yang ingin Anda ganti nama. Sesi aktif notebook harus dihentikan sebelum Anda dapat mengganti nama notebook.
2. TerbukaPenjelajah notebook.
3. DalamNotebookdaftar, pilih tombol opsi untuk notebook yang ingin Anda ganti nama.

4. DariAksimenu, pilihGanti nama.
5. DiGanti nama buku catatanprompt, masukkan nama baru, dan kemudian pilihSimpan. Nama notebook baru muncul di daftar notebook.

Menghapus buku catatan

1. Mengakhirisetiap sesi aktif untuk notebook yang ingin Anda hapus. Sesi aktif notebook harus dihentikan sebelum Anda dapat menghapus notebook.
2. TerbukaPenjelajah notebook.
3. DalamNotebookdaftar, pilih tombol opsi untuk notebook yang ingin Anda hapus.
4. Dari menu Tindakan, pilih Hapus.
5. DiHapus notebook?prompt, masukkan nama notebook, dan kemudian pilihMenghapusuntuk mengkonfirmasi penghapusan. Nama notebook dihapus dari daftar notebook.

Untuk mengekspor buku catatan

1. TerbukaPenjelajah notebook.
2. DalamNotebookdaftar, pilih tombol opsi untuk notebook yang ingin Anda ekspor.
3. DariAksimenu, pilihEkspor file.

Untuk mengimpor notebook

1. TerbukaPenjelajah notebook.
2. PilihImpor berkas.
3. Jelajahi lokasi di komputer lokal Anda dari file yang ingin Anda impor, lalu pilihTerbuka. Notebook yang diimpor muncul dalam daftar notebook.

Melihat riwayat sesi untuk buku catatan

1. TerbukaPenjelajah notebook.
2. DalamNotebookdaftar, pilih tombol opsi untuk notebook yang sejarah sesi Anda ingin melihat.
3. DariAksimenu, pilihRiwayat sesi.
4. PadaSejarahtab, pilihID Sesiuntuk melihat informasi tentang sesi dan perhitungannya.

Menggunakan format tabel non-Hive di Amazon Athena untuk Apache Spark

Saat Anda bekerja dengan sesi dan notebook di Athena untuk Spark, Anda dapat menggunakan tabel Linux Foundation Delta Lake, Apache Hudi, dan Apache Iceberg, selain tabel Apache Hive.

Pertimbangan dan batasan

Bila Anda menggunakan format tabel selain Apache Hive dengan Athena untuk Spark, pertimbangkan hal-hal berikut:

- Selain Apache Hive, hanya satu format tabel yang didukung per notebook. Untuk menggunakan beberapa format tabel di Athena untuk Spark, buat buku catatan terpisah untuk setiap format tabel. Untuk informasi tentang membuat notebook di Athena untuk Spark, lihat. Membuat buku catatan Anda sendiri
- Format tabel Delta Lake, Hudi, dan Iceberg telah diuji di Athena untuk Spark dengan menggunakan sebagai metastore. AWS Glue Anda mungkin dapat menggunakan metastores lain, tetapi penggunaan tersebut saat ini tidak didukung.
- Untuk menggunakan format tabel tambahan, ganti `spark_catalog` properti default, seperti yang ditunjukkan di konsol Athena dan dalam dokumentasi ini. Katalog non-sarang ini dapat membaca tabel Hive, selain format tabelnya sendiri.

Versi tabel

Tabel berikut menunjukkan didukung versi tabel non-HIVE di Amazon Athena untuk Apache Spark.

Format tabel	Versi yang didukung
Gunung Es Apache	1.2.1
Apache Hudi	0,13
Yayasan Linux Delta Lake	2.0.2

Di Athena for Spark, `.jar` file format tabel ini dan dependensinya dimuat ke classpath untuk driver dan pelaksana Spark.

Untuk posting Blog AWS Big Data yang menunjukkan cara bekerja dengan format tabel Iceberg, Hudi, dan Delta Lake menggunakan Spark SQL di notebook Amazon Athena, lihat Menggunakan Amazon Athena dengan Spark SQL untuk format tabel transaksional open-source Anda.

Topik

- Gunung Es Apache
- Apache Hudi
- Yayasan Linux Delta Lake

Gunung Es Apache

Apache Iceberg adalah format tabel terbuka untuk kumpulan data besar di Amazon Simple Storage Service (Amazon S3). Ini memberi Anda kinerja kueri cepat di atas tabel besar, komit atom, penulisan bersamaan, dan evolusi tabel yang kompatibel dengan SQL.

Untuk menggunakan tabel Apache Iceberg di Athena untuk Spark, konfigurasikan properti Spark berikut. Properti ini dikonfigurasi untuk Anda secara default di konsol Athena untuk Spark ketika Anda memilih Apache Iceberg sebagai format tabel. Untuk langkah-langkah, lihat Mengedit detail sesi atauMembuat buku catatan Anda sendiri.

```
"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"
```

Prosedur berikut menunjukkan cara menggunakan tabel Apache Iceberg di Athena untuk notebook Spark. Jalankan setiap langkah di sel baru di notebook.

Untuk menggunakan tabel Apache Iceberg di Athena untuk Spark

1. Tentukan konstanta yang akan digunakan di notebook.

   ```
   DB_NAME = "NEW_DB_NAME"
   TABLE_NAME = "NEW_TABLE_NAME"
   TABLE_S3_LOCATION = "s3://DOC-EXAMPLE-BUCKET"
   ```

2. Buat Apache Spark DataFrame.

```
columns = ["language","users_count"]
data = [("Golang", 3000)]
df = spark.createDataFrame(data, columns)
```

3. Buat database.

```
spark.sql("CREATE DATABASE {} LOCATION '{}'".format(DB_NAME, TABLE_S3_LOCATION))
```

4. Buat tabel Apache Iceberg kosong.

```
spark.sql("""
CREATE TABLE {}.{} (
language string,
users_count int
) USING ICEBERG
""".format(DB_NAME, TABLE_NAME))
```

5. Masukkan deretan data ke dalam tabel.

```
spark.sql("""INSERT INTO {}.{} VALUES ('Golang',
 3000)""".format(DB_NAME, TABLE_NAME))
```

6. Konfirmasikan bahwa Anda dapat menanyakan tabel baru.

```
spark.sql("SELECT * FROM {}.{}".format(DB_NAME, TABLE_NAME)).show()
```

Untuk informasi selengkapnya dan contoh tentang bekerja dengan tabel Spark DataFrames dan Iceberg, lihat Kueri Spark di dokumentasi Apache Iceberg.

Apache Hudi

Apache Hudi adalah kerangka kerja manajemen data sumber terbuka yang menyederhanakan pemrosesan data tambahan. Tindakan penyisipan, pembaruan, peningkatan, dan penghapusan tingkat rekaman diproses dengan lebih presisi, yang mengurangi overhead.

Untuk menggunakan tabel Apache Hudi di Athena untuk Spark, konfigurasikan properti Spark berikut. Properti ini dikonfigurasi untuk Anda secara default di konsol Athena untuk Spark ketika Anda memilih Apache Hudi sebagai format tabel. Untuk langkah-langkah, lihat Mengedit detail sesi atauMembuat buku catatan Anda sendiri.

```
"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"
```

Prosedur berikut menunjukkan cara menggunakan tabel Apache Hudi di Athena untuk notebook Spark. Jalankan setiap langkah di sel baru di notebook.

Untuk menggunakan tabel Apache Hudi di Athena untuk Spark

1. Tentukan konstanta yang akan digunakan di notebook.

```
DB_NAME = "NEW_DB_NAME"
TABLE_NAME = "NEW_TABLE_NAME"
TABLE_S3_LOCATION = "s3://DOC-EXAMPLE-BUCKET"
```

2. Buat Apache Spark DataFrame.

```
columns = ["language","users_count"]
data = [("Golang", 3000)]
df = spark.createDataFrame(data, columns)
```

3. Buat database.

```
spark.sql("CREATE DATABASE {} LOCATION '{}'".format(DB_NAME, TABLE_S3_LOCATION))
```

4. Buat tabel Apache Hudi kosong.

```
spark.sql("""
CREATE TABLE {}.{} (
language string,
users_count int
) USING HUDI
TBLPROPERTIES (
primaryKey = 'language',
type = 'mor'
);
""".format(DB_NAME, TABLE_NAME))
```

5. Masukkan deretan data ke dalam tabel.

```
spark.sql("""INSERT INTO {}.{} VALUES ('Golang',
 3000)""".format(DB_NAME,TABLE_NAME))
```

6. Konfirmasikan bahwa Anda dapat menanyakan tabel baru.

```
spark.sql("SELECT * FROM {}.{}".format(DB_NAME, TABLE_NAME)).show()
```

Yayasan Linux Delta Lake

Linux Foundation Delta Lake adalah format tabel yang dapat Anda gunakan untuk analitik data besar. Anda dapat menggunakan Athena for Spark untuk membaca tabel Delta Lake yang disimpan di Amazon S3 secara langsung.

Untuk menggunakan tabel Delta Lake di Athena untuk Spark, konfigurasikan properti Spark berikut. Properti ini dikonfigurasi untuk Anda secara default di konsol Athena untuk Spark ketika Anda memilih Delta Lake sebagai format tabel. Untuk langkah-langkah, lihat Mengedit detail sesi atauMembuat buku catatan Anda sendiri.

```
"spark.sql.catalog.spark_catalog" : "org.apache.spark.sql.delta.catalog.DeltaCatalog",
"spark.sql.extensions" : "io.delta.sql.DeltaSparkSessionExtension"
```

Prosedur berikut menunjukkan kepada Anda cara menggunakan tabel Delta Lake di notebook Athena untuk Spark. Jalankan setiap langkah di sel baru di notebook.

Untuk menggunakan meja Danau Delta di Athena untuk Spark

1. Tentukan konstanta yang akan digunakan di notebook.

```
DB_NAME = "NEW_DB_NAME"
TABLE_NAME = "NEW_TABLE_NAME"
TABLE_S3_LOCATION = "s3://DOC-EXAMPLE-BUCKET"
```

2. Buat Apache Spark DataFrame.

```
columns = ["language","users_count"]
data = [("Golang", 3000)]
df = spark.createDataFrame(data, columns)
```

3. Buat database.

```
spark.sql("CREATE DATABASE {} LOCATION '{}'".format(DB_NAME, TABLE_S3_LOCATION))
```

4. Buat tabel Delta Lake kosong.

```
spark.sql("""
CREATE TABLE {}.{} (
  language string,
  users_count int
) USING DELTA
""".format(DB_NAME, TABLE_NAME))
```

5. Masukkan deretan data ke dalam tabel.

```
spark.sql("""INSERT INTO {}.{} VALUES ('Golang',
 3000)""".format(DB_NAME, TABLE_NAME))
```

6. Konfirmasikan bahwa Anda dapat menanyakan tabel baru.

```
spark.sql("SELECT * FROM {}.{}".format(DB_NAME, TABLE_NAME)).show()
```

Dukungan pustaka Python di Amazon Athena untuk Apache Spark

Halaman ini menjelaskan terminologi yang digunakan dan manajemen siklus hidup yang diikuti untuk runtime, pustaka, dan paket yang digunakan di Amazon Athena untuk Apache Spark.

Ketentuan

- Amazon Athena untuk Apache Spark adalah versi khusus dari Apache Spark open source. Untuk melihat versi saat ini, jalankan perintah `print(f'{spark.version}')` di sel notebook.
- Runtime Athena adalah lingkungan tempat kode Anda berjalan. Lingkungan mencakup penerjemah Python dan perpustakaan. PySpark
- Pustaka atau paket eksternal adalah pustaka Java atau Scala JAR atau Python yang bukan bagian dari runtime Athena tetapi dapat dimasukkan dalam Athena untuk pekerjaan Spark. Paket eksternal dapat dibuat oleh Amazon atau oleh Anda.
- Paket kenyamanan adalah kumpulan paket eksternal yang dipilih oleh Athena yang dapat Anda pilih untuk disertakan dalam aplikasi Spark Anda.
- Sebuah bundel menggabungkan runtime Athena dan paket kenyamanan.

- Pustaka pengguna adalah pustaka atau paket eksternal yang Anda tambahkan secara eksplisit ke pekerjaan Athena for Spark Anda.
  - Pustaka pengguna adalah paket eksternal yang bukan bagian dari paket kenyamanan. Pustaka pengguna memerlukan pemuatan dan instalasi, seperti ketika Anda menulis beberapa `.py` file, zip mereka, dan kemudian menambahkan `.zip` file ke aplikasi Anda.
- Aplikasi Athena untuk Spark adalah pekerjaan atau pertanyaan yang Anda kirimkan ke Athena untuk Spark.

Manajemen siklus hidup

Pembuatan versi dan penghentian runtime

Komponen utama dalam runtime Athena adalah penerjemah Python. Karena Python adalah bahasa yang berkembang, versi baru dirilis secara teratur dan dukungan dihapus untuk versi yang lebih lama. Athena tidak menyarankan Anda menjalankan program dengan versi penerjemah Python yang tidak digunakan lagi dan sangat menyarankan agar Anda menggunakan runtime Athena terbaru bila memungkinkan.

Jadwal penghentian runtime Athena adalah sebagai berikut:

1. Setelah Athena memberikan runtime baru, Athena akan terus mendukung runtime sebelumnya selama 6 bulan. Selama waktu itu, Athena akan menerapkan patch keamanan dan pembaruan ke runtime sebelumnya.
2. Setelah 6 bulan, Athena akan mengakhiri dukungan untuk runtime sebelumnya. Athena tidak akan lagi menerapkan patch keamanan dan pembaruan lainnya ke runtime sebelumnya. Aplikasi Spark yang menggunakan runtime sebelumnya tidak lagi memenuhi syarat untuk dukungan teknis.
3. Setelah 12 bulan, Anda tidak akan lagi dapat memperbarui atau mengedit aplikasi Spark di workgroup yang menggunakan runtime sebelumnya. Kami menyarankan Anda memperbarui aplikasi Spark Anda sebelum periode waktu ini berakhir. Setelah periode waktu berakhir, Anda masih dapat menjalankan notebook yang ada, tetapi notebook apa pun yang masih menggunakan runtime sebelumnya akan mencatat peringatan untuk efek itu.
4. Setelah 18 bulan, Anda tidak akan lagi dapat menjalankan pekerjaan di grup kerja menggunakan runtime sebelumnya.

Pembuatan versi dan penghentian paket kenyamanan

Isi paket kenyamanan berubah seiring waktu. Athena sesekali menambahkan, menghapus, atau meningkatkan paket kenyamanan ini.

Athena menggunakan panduan berikut untuk paket kenyamanan:

- Paket kenyamanan memiliki skema versi sederhana seperti 1, 2, 3.
- Setiap versi paket kenyamanan mencakup versi spesifik dari paket eksternal. Setelah Athena membuat paket kenyamanan, paket paket kenyamanan paket eksternal dan versi yang sesuai tidak berubah.
- Athena membuat versi paket kenyamanan baru ketika menyertakan paket eksternal baru, menghapus paket eksternal, atau meningkatkan versi satu atau lebih paket eksternal.

Athena menghentikan paket kenyamanan saat tidak digunakan lagi runtime Athena yang digunakan paket tersebut. Athena dapat menghentikan paket lebih cepat untuk membatasi jumlah bundel yang didukungnya.

Jadwal penghentian paket kenyamanan mengikuti jadwal penghentian runtime Athena.

Daftar pustaka Python yang sudah diinstal sebelumnya

Pustaka Python yang sudah diinstal sebelumnya mencakup yang berikut ini.

```
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
```

Catatan

- MLLib (pustaka pembelajaran mesin Apache Spark) dan `pyspark.ml` paket tidak didukung.
- Saat `pip install` ini, tidak didukung di Athena untuk sesi Spark.

Untuk informasi tentang mengimpor pustaka Python ke Amazon Athena untuk Apache Spark, lihat. Mengimpor file dan pustaka Python ke Amazon Athena untuk Apache Spark

Mengimpor file dan pustaka Python ke Amazon Athena untuk Apache Spark

Dokumen ini memberikan contoh cara mengimpor file dan pustaka Python ke Amazon Athena untuk Apache Spark.

Pertimbangan dan batasan

- Versi Python - Saat ini, Athena untuk Spark menggunakan Python versi 3.9.16. Perhatikan bahwa paket Python sensitif terhadap versi Python minor.

- Athena untuk arsitektur Spark - Athena untuk Spark menggunakan Amazon Linux 2 pada arsitektur ARM64. Perhatikan bahwa beberapa pustaka Python tidak mendistribusikan binari untuk arsitektur ini.
- Objek bersama biner (SO) — Karena SparkContext addPyFilemetode ini tidak mendeteksi objek bersama biner, metode ini tidak dapat digunakan di Athena untuk Spark untuk menambahkan paket Python yang bergantung pada objek bersama.
- Resilient Distributed Datasets (RDD) - RDD tidak didukung.
- DataFrame.forEach — Metode PySpark DataFrame.foreach tidak didukung.

Contoh

Contoh menggunakan konvensi berikut.

- Lokasi placeholder Amazon S3. `s3://DOC-EXAMPLE-BUCKET` Ganti ini dengan lokasi bucket S3 Anda sendiri.
- Semua blok kode yang mengeksekusi dari shell Unix ditampilkan sebagai *`$directory_name`*. Misalnya, perintah `ls` dalam direktori `/tmp` dan outputnya ditampilkan sebagai berikut:

```
/tmp $ ls
```

Keluaran

```
file1 file2
```

- Menambahkan file ke buku catatan setelah menulisnya ke direktori sementara lokal
- Mengimpor file dari Amazon S3
- Menambahkan file Python dan mendaftarkan UDF
- Mengimpor file.zip Python
- Mengimpor dua versi pustaka Python sebagai modul terpisah
- Mengimpor file.zip Python dari PyPI
- Mengimpor file.zip Python dari PyPI yang memiliki dependensi

Mengimpor file teks untuk digunakan dalam perhitungan

Contoh di bagian ini menunjukkan cara mengimpor file teks untuk digunakan dalam perhitungan di buku catatan Anda di Athena untuk Spark.

Menambahkan file ke buku catatan setelah menulisnya ke direktori sementara lokal

Contoh berikut menunjukkan cara menulis file ke direktori sementara lokal, menambahkannya ke buku catatan, dan mengujinya.

```
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()
```

Keluaran

```
Calculation completed.
+---+---+-------+
| _1| _2|    col|
+---+---+-------+
|  1|  a|[aaaaa]|
|  2|  b|[bbbbb]|
+---+---+-------+
```

Mengimpor file dari Amazon S3

Contoh berikut menunjukkan cara mengimpor file dari Amazon S3 ke notebook dan mengujinya.

Untuk mengimpor file dari Amazon S3 ke notebook

1. Buat file bernama `test.txt` yang memiliki satu baris yang berisi nilai`5`.
2. Tambahkan file ke ember di Amazon S3. Contoh ini menggunakan lokasi`s3://DOC-EXAMPLE-BUCKET`.
3. Gunakan kode berikut untuk mengimpor file ke buku catatan Anda dan menguji file tersebut.

```
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()
```

Keluaran

```
Calculation completed.
+---+---+-------+
| _1| _2|    col|
+---+---+-------+
|  1|  a|[aaaaa]|
|  2|  b|[bbbbb]|
+---+---+-------+
```

Menambahkan file Python

Contoh di bagian ini menunjukkan cara menambahkan file dan pustaka Python ke buku catatan Spark Anda di Athena.

Menambahkan file Python dan mendaftarkan UDF

Contoh berikut menunjukkan cara menambahkan file Python dari Amazon S3 ke notebook Anda dan mendaftarkan UDF.

Untuk menambahkan file Python ke buku catatan Anda dan mendaftarkan UDF

1. Menggunakan lokasi Amazon S3 Anda sendiri, buat file `s3://DOC-EXAMPLE-BUCKET/file1.py` dengan konten berikut:

```
def xyz(input):
    return 'xyz  - udf ' + str(input);
```

2. Di lokasi S3 yang sama, buat file `s3://DOC-EXAMPLE-BUCKET/file2.py` dengan konten berikut:

```
from file1 import xyz
def uvw(input):
    return 'uvw -> ' + xyz(input);
```

3. Di notebook Athena for Spark Anda, jalankan perintah berikut.

```
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)
```

Keluaran

```
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... |
+---+---+--------------------+
```

Mengimpor file.zip Python

Anda dapat menggunakan Python `addPyFile` dan `import` metode untuk mengimpor file.zip Python ke notebook Anda.

> **Note**
>
> `.zip`File yang Anda impor ke Athena Spark mungkin hanya menyertakan paket Python. Misalnya, termasuk paket dengan file berbasis C tidak didukung.

Untuk mengimpor **`.zip`** file Python ke buku catatan Anda

1. Di komputer lokal Anda, di direktori desktop seperti`\tmp`, buat direktori yang disebut`moduletest`.
2. Di `moduletest` direktori, buat file bernama `hello.py` dengan konten berikut:

   ```
   def hi(input):
       return 'hi ' + str(input);
   ```
3. Di direktori yang sama, tambahkan file kosong dengan nama`__init__.py`.

   Jika Anda mencantumkan isi direktori, mereka sekarang akan terlihat seperti berikut ini.

   ```
   /tmp $ ls moduletest
   __init__.py       hello.py
   ```

4. Gunakan `zip` perintah untuk menempatkan dua file modul ke dalam file bernama`moduletest.zip`.

```
moduletest $ zip -r9 ../moduletest.zip *
```

5. Unggah `.zip` file ke bucket Anda di Amazon S3.
6. Gunakan kode berikut untuk mengimpor `.zip` file Python ke notebook Anda.

```
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()
```

Keluaran

```
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|
+---+---+----+
```

Mengimpor dua versi pustaka Python sebagai modul terpisah

Contoh kode berikut menunjukkan cara menambahkan dan mengimpor dua versi pustaka Python yang berbeda dari lokasi di Amazon S3 sebagai dua modul terpisah. Kode menambahkan setiap file pustaka dari S3, mengimpornya, dan kemudian mencetak versi pustaka untuk memverifikasi impor.

```
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__)
```

Keluaran

```
3.15.0
```

```
import simplejson_v3_17_6
print(simplejson_v3_17_6.__version__)
```

Keluaran

```
3.17.6
```

Mengimpor file.zip Python dari PyPI

Contoh ini menggunakan `pip` perintah untuk mengunduh file.zip Python dari proyek bpabel/piglatin dari Python Package Index (PyPI).

Untuk mengimpor file.zip Python dari PyPI

1. Di desktop lokal Anda, gunakan perintah berikut untuk membuat direktori yang disebut `testpiglatin` dan membuat lingkungan virtual.

   ```
   /tmp $ mkdir testpiglatin
   /tmp $ cd testpiglatin
   testpiglatin $ virtualenv .
   ```

   Keluaran

   ```
   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. Buat subdirektori bernama `unpacked` untuk menampung proyek.

```
testpiglatin $ mkdir unpacked
```

3. Gunakan `pip` perintah untuk menginstal proyek ke `unpacked` direktori.

```
testpiglatin $ bin/pip install -t $PWD/unpacked piglatin
```

Keluaran

```
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. Periksa isi direktori.

```
testpiglatin $ ls
```

Keluaran

```
bin lib pyvenv.cfg unpacked
```

5. Ubah ke `unpacked` direktori dan tampilkan isinya.

```
testpiglatin $ cd unpacked
unpacked $ ls
```

Keluaran

```
piglatin piglatin-1.0.6.dist-info
```

6. Gunakan `zip` perintah untuk menempatkan isi proyek piglatin ke dalam file bernama`library.zip`.

```
unpacked $ zip -r9 ../library.zip *
```

Keluaran

```
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. (Opsional) Gunakan perintah berikut untuk menguji impor secara lokal.

   a. Atur jalur Python ke lokasi `library.zip` file dan mulai Python.

   ```
   /home $ PYTHONPATH=/tmp/testpiglatin/library.zip
   /home $ python3
   ```

   Keluaran

   ```
   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. Impor pustaka dan jalankan perintah uji.

   ```
   >>> import piglatin
   >>> piglatin.translate('hello')
   ```

   Keluaran

   ```
   'ello-hay'
   ```

8. Gunakan perintah seperti berikut ini untuk menambahkan `.zip` file dari Amazon S3, mengimpornya ke notebook Anda di Athena, dan mengujinya.

```
sc.addPyFile('s3://DOC-EXAMPLE-BUCKET/library.zip')
```

```
import piglatin
piglatin.translate('hello')

from pyspark.sql.functions import udf
from pyspark.sql.functions import col

hi_udf = udf(piglatin.translate)

df = spark.createDataFrame([(1, "hello"), (2, "world")])

df.withColumn("col", hi_udf(col('_2'))).show()
```

Keluaran

```
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|
+---+-----+--------+
```

Mengimpor file.zip Python dari PyPI yang memiliki dependensi

Contoh ini mengimpor paket md2gemini, yang mengubah teks dalam penurunan harga ke format teks Gemini, dari PyPI. Paket ini memiliki dependensi berikut:

```
cjkwrap
mistune
wcwidth
```

Untuk mengimpor file.zip Python yang memiliki dependensi

1. Di komputer lokal Anda, gunakan perintah berikut untuk membuat direktori yang disebut `testmd2gemini` dan membuat lingkungan virtual.

```
/tmp $ mkdir testmd2gemini
/tmp $ cd testmd2gemini
```

```
testmd2gemini$ virtualenv .
```

2. Buat subdirektori bernama `unpacked` untuk menampung proyek.

```
testmd2gemini $ mkdir unpacked
```

3. Gunakan `pip` perintah untuk menginstal proyek ke `unpacked` direktori.

```
/testmd2gemini $ bin/pip install -t $PWD/unpacked md2gemini
```

Keluaran

```
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. Ubah ke `unpacked` direktori dan periksa isinya.

```
testmd2gemini $ cd unpacked
unpacked $ ls -lah
```

Keluaran

```
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. Gunakan `zip` perintah untuk menempatkan isi proyek md2gemini ke dalam file bernama. `md2gemini.zip`

```
unpacked $ zip -r9 ../md2gemini *
```

Keluaran

```
  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. (Opsional) Gunakan perintah berikut untuk menguji apakah perpustakaan berfungsi di komputer lokal Anda.

   a. Atur jalur Python ke lokasi `md2gemini.zip` file dan mulai Python.

```
/home $ PYTHONPATH=/tmp/testmd2gemini/md2gemini.zip
/home python3
```

   b. Impor pustaka dan jalankan pengujian.

```
>>> from md2gemini import md2gemini
>>> print(md2gemini('[abc](https://abc.def)'))
```

Keluaran

```
https://abc.def abc
```

7. Gunakan perintah berikut untuk menambahkan `.zip` file dari Amazon S3, mengimpornya ke notebook Anda di Athena, dan melakukan pengujian non UDF.

```
# (non udf test)
sc.addPyFile('s3://DOC-EXAMPLE-BUCKET/md2gemini.zip')
from md2gemini import md2gemini
```

```
print(md2gemini('[abc](https://abc.def)'))
```

Keluaran

```
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. Gunakan perintah berikut untuk melakukan tes UDF.

```
# (udf test)

from pyspark.sql.functions import udf
from pyspark.sql.functions import col
from md2gemini import md2gemini


hi_udf = udf(md2gemini)
df = spark.createDataFrame([(1, "[first website](https://abc.def)"), (2, "[second
 website](https://aws.com)")])
df.withColumn("col", hi_udf(col('_2'))).show()
```

Keluaran

```
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...|
+---+--------------------+--------------------+
```

Menambahkan file JAR dan konfigurasi Spark khusus

Saat membuat atau mengedit sesi di Amazon Athena untuk Apache Spark, Anda dapat menggunakannyaSifat percikanuntuk menentukan `.jar`file, paket, atau konfigurasi kustom lainnya

untuk sesi. Untuk menentukan properti Spark Anda, Anda dapat menggunakan konsol Athena,AWS CLI, atau API Athena.

Menggunakan konsol Athena untuk menentukan properti Spark

Di konsol Athena, Anda dapat menentukan properti Spark Anda saat Andamembuat notebookataumengedit sesi saat ini.

Untuk menambahkan properti diBuat notebookatauEdit detail sesikotak dialog

1. MemperluasSifat percikan.
2. Untuk menambahkan properti Anda, gunakanEdit dalam tabelatauEdit di JSONpilihan.
   - UntukEdit dalam tabelpilihan, pilihTambahkan propertiuntuk menambahkan properti, atau memilihHapusuntuk menghapus properti. GunakanKuncidanNilaikotak untuk memasukkan nama properti dan nilai-nilai mereka.
     - Untuk menambahkan kustom `.jar`file, gunakan`spark.jars`properti.
     - Untuk menentukan file paket, gunakan`spark.jars.packages`properti.
   - Untuk memasukkan dan mengedit konfigurasi Anda secara langsung, pilihEdit di JSONpilihan. Di editor teks JSON, Anda dapat melakukan tugas-tugas berikut:
     - PilihSalinuntuk menyalin teks JSON ke clipboard.
     - PilihJelasuntuk menghapus semua teks dari editor JSON.
     - Pilih ikon pengaturan (roda gigi) untuk mengonfigurasi pembungkus garis atau memilih tema warna untuk editor JSON.

Catatan

- Anda dapat mengatur properti di Athena untuk Spark, yang sama dengan pengaturanSifat percikanlangsung padaSparkConfobjek.
- Mulai semua properti Spark dengan`spark.`awalan. Properti dengan awalan lain diabaikan.
- Tidak semua properti Spark tersedia untuk konfigurasi khusus di Athena. Jika Anda mengirimkan`StartSession`permintaan yang memiliki konfigurasi terbatas, sesi gagal untuk memulai.
  - Anda tidak dapat menggunakan`spark.athena.`awalan karena dicadangkan.

MenggunakanAWS CLIatau Athena API untuk menyediakan konfigurasi kustom

Untuk menggunakanAWS CLIatau Athena API untuk menyediakan konfigurasi sesi Anda, gunakanStartSessionAksi API atausesi awalPerintah CLI. Di dalam`StartSession`permintaan, gunakan`SparkProperties`bidangEngineConfigurationkeberatan untuk meneruskan informasi konfigurasi Anda dalam format JSON. Ini memulai sesi dengan konfigurasi yang Anda tentukan. Untuk sintaks permintaan, lihatStartSessiondi dalamReferensi API Amazon Athena.

Memecahkan masalah kesalahan mulai sesi

Ketika kesalahan konfigurasi kustom terjadi selama sesi dimulai, konsol Athena untuk Spark menunjukkan spanduk pesan kesalahan. Untuk memecahkan masalah kesalahan mulai sesi, Anda dapat memeriksa perubahan status sesi atau informasi logging.

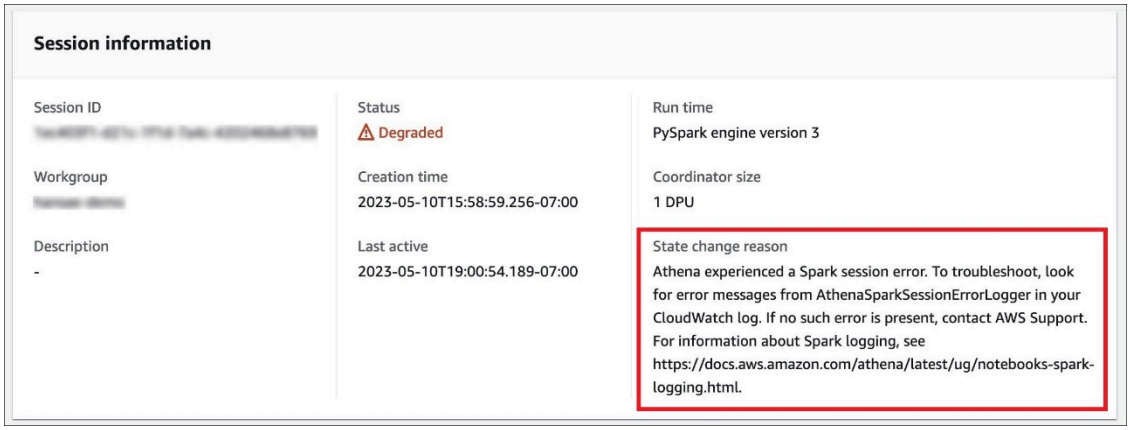
Melihat informasi perubahan status sesi

Anda bisa mendapatkan rincian tentang perubahan status sesi dari editor notebook Athena atau dari Athena API.

Untuk melihat informasi status sesi di konsol Athena

1. Di editor notebook Athena, dariSesimenu di kanan atas, pilihLihat rincian.
2. LihatSesi saat initab. YangInformasi sesibagian menunjukkan informasi seperti ID sesi, kelompok kerja, status, dan alasan perubahan negara.

   Contoh tangkapan layar berikut menunjukkan informasi diAlasan perubahan negarabagian dariInformasi sesikotak dialog untuk kesalahan sesi Spark di Athena.



Untuk melihat informasi status sesi menggunakan API Athena

- Di API Athena, Anda dapat menemukan informasi perubahan status sesi di`StateChangeReason`bidang`SessionStatus`objek.

> **Note**
>
> Setelah Anda menghentikan sesi secara manual, atau jika sesi berhenti setelah batas waktu idle (defaultnya adalah 20 menit), nilai`StateChangeReason`perubahanSesi diakhiri per permintaan.

Menggunakan pencatatan untuk memecahkan masalah kesalahan mulai sesi

Kesalahan konfigurasi kustom yang terjadi selama sesi mulai dicatat olehAmazonCloudWatch. Di dalamCloudWatchLog, mencari pesan kesalahan dari`AthenaSparkSessionErrorLogger`untuk memecahkan masalah awal sesi gagal.

Untuk informasi selengkapnya tentang Pencatatan Spark, lihatLogging peristiwa aplikasi Spark di Athena.

Untuk informasi selengkapnya tentang sesi pemecahan masalah di Athena for Spark, lihatSesi pemecahan masalah.

Format data dan penyimpanan yang didukung

Tabel berikut menunjukkan format yang didukung secara asli di Athena untuk Apache Spark.

Format data	Membaca	Menulis	Tulis kompresi
parquet	Ya	Ya	tidak ada, tidak terkompresi, tajam, gzip
orc	Ya	Ya	tidak ada, tajam, zlib, lzo

Format data	Membaca	Menulis	Tulis kompresi
json	Ya	Ya	bzip2, gzip, mengempiskan
csv	Ya	Ya	bzip2, gzip, mengempiskan
text	Ya	Ya	tidak ada, bzip2, gzip, mengempis
berkas biner	Ya	N/A	N/A

Memantau perhitungan Apache Spark denganCloudWatchmetrik

Athena menerbitkan metrik terkait perhitungan ke AmazonCloudWatchketikaPublish CloudWatch metricspilihan untuk workgroup SPARK-enabled Anda dipilih. Anda dapat membuat dasbor khusus, mengatur alarm, dan memicu metrik diCloudWatchkonsol.

Athena menerbitkan metrik berikut keCloudWatchkonsol di bawah`AmazonAthenaForApacheSpark`namespace:

- `DPUCount`- jumlah DPU yang dikonsumsi selama sesi untuk mengeksekusi perhitungan.

Metrik ini memiliki dimensi sebagai berikut:

- `SessionId`— ID sesi di mana perhitungan diajukan.
- `WorkGroup`— Nama workgroup.

Untuk melihat metrik untuk kelompok kerja berkemampuan Spark di AmazonCloudWatchkonsol

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilihMetrik,Semua metrik.
3. PilihAmazonAthenaForApacheSparknamespace.

Untuk melihat metrik dengan CLI

- Lakukan salah satu dari berikut:
  - Untuk mencantumkan metrik grup kerja yang mendukung Athena Spark, buka prompt perintah, dan gunakan perintah berikut:

```
aws cloudwatch list-metrics --namespace "AmazonAthenaForApacheSpark"
```

  - Untuk mencantumkan semua metrik yang tersedia, gunakan perintah berikut:

```
aws cloudwatch list-metrics
```

DaftarCloudWatchmetrik dan dimensi untuk perhitungan Apache Spark di Athena

Jika Anda telah mengaktifkanCloudWatchmetrik dalam grup kerja Athena berkemampuan Spark Anda, Athena mengirimkan metrik berikut keCloudWatchper kelompok kerja. Metrik menggunakan`AmazonAthenaForApacheSpark`namespace.

Nama metrik	Deskripsi
DPUCount	Jumlah DPU (unit pengolahan data) yang dikonsumsi selama sesi untuk mengeksekusi perhitungan. DPU adalah ukuran relatif daya pemrosesan yang terdiri dari 4 vCPU kapasitas komputasi dan 16 GB memori.

Metrik ini memiliki dimensi berikut.

Dimensi	Deskripsi
SessionId	ID sesi di mana perhitungan diajukan.
WorkGroup	Nama workgroup.

Mengaktifkan pemohon membayar bucket Amazon S3 di Athena untuk Spark

Ketika bucket Amazon S3 dikonfigurasi sebagai pemohon membayar, akun pengguna yang menjalankan kueri dikenakan biaya untuk akses data dan biaya transfer data yang terkait dengan kueri. Untuk informasi lebih lanjut, lihatMenggunakan Bucket Peminta Membayar untuk transfer penyimpanan dan penggunaandi dalamPanduan Pengguna Amazon S3.

Di Athena untuk Spark, pemohon membayar ember diaktifkan per sesi, bukan per kelompok kerja. Pada tingkat tinggi, memungkinkan pemohon membayar ember mencakup langkah-langkah berikut:

1. Di konsol Amazon S3, aktifkan pemohon membayar pada properti untuk bucket dan tambahkan kebijakan bucket untuk menentukan akses.
2. Di konsol IAM, buat kebijakan IAM untuk mengizinkan akses ke bucket, lalu lampirkan kebijakan ke peran IAM yang akan digunakan untuk mengakses bucket pembayaran pemohon.
3. Di Athena untuk Spark, tambahkan properti sesi untuk mengaktifkan fitur peminta membayar.

1. Aktifkan pemohon membayar pada bucket Amazon S3 dan menambahkan kebijakan bucket

Untuk mengaktifkan pemohon membayar pada bucket Amazon S3

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Dalam daftar bucket, pilih tautan untuk bucket yang ingin Anda aktifkan peminta bayar.
3. Pada halaman bucket, pilihPropertitab.
4. Gulir ke bawah kePemohon membayarbagian, dan kemudian pilihMengedit.
5. PadaEdit pemohon membayarhalaman, pilihAktifkan, dan kemudian pilihSimpan perubahan.
6. Pilih tab Izin.
7. Di bagian Kebijakan bucket, pilih Edit.
8. PadaEdit kebijakan bucketpage, terapkan kebijakan bucket yang Anda inginkan ke bucket sumber. Kebijakan contoh berikut memberikan akses ke semuaAWSkepala sekolah (`"AWS": "*"`), tetapi akses Anda bisa lebih terperinci. Misalnya, Anda mungkin ingin menentukan hanya peran IAM tertentu di akun lain.

```
{
```

```
    "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. Buat kebijakan IAM dan lampirkan ke peran IAM

Selanjutnya, Anda membuat kebijakan IAM untuk mengizinkan akses ke bucket. Kemudian Anda melampirkan kebijakan untuk peran yang akan digunakan untuk mengakses pemohon membayar ember.

Untuk membuat kebijakan IAM untuk pemohon membayar bucket dan melampirkan kebijakan ke peran

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi konsol IAM, pilihKebijakan.
3. Pilih Buat kebijakan.
4. PilihJSON.
5. Di dalamEditor kebijakan, tambahkan kebijakan seperti berikut ini:

```
{
    "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. Pilih Selanjutnya.
7. PadaTinjau dan buathalaman, masukkan nama untuk kebijakan dan deskripsi opsional, lalu pilihBuat kebijakan.
8. Di panel navigasi, pilih Peran.
9. PadaPeranhalaman, temukan peran yang ingin Anda gunakan, lalu pilih tautan nama peran.
10. Di dalamKebijakan izinbagian, pilihTambahkan izin,Lampirkan kebijakan.
11. Di dalamKebijakan izin lainnyabagian, pilih kotak centang untuk kebijakan yang Anda buat, lalu pilihTambahkan izin.

3. Tambahkan Athena untuk properti sesi Spark

Setelah mengonfigurasi bucket Amazon S3 dan izin terkait untuk pembayaran pemohon, Anda dapat mengaktifkan fitur tersebut dalam sesi Athena for Spark.

Untuk mengaktifkan pemohon membayar ember di Athena untuk sesi Spark

1. Di editor notebook, dariSesimenu di kanan atas, pilihEdit sesi.
2. MemperluasSifat percikan.
3. PilihEdit di JSON.
4. Di editor teks JSON, masukkan yang berikut ini:

```
{
  "spark.hadoop.fs.s3.useRequesterPaysHeader":"true"
}
```

5. Pilih Simpan.

Mengaktifkan enkripsi Apache Spark

Anda dapat mengaktifkan enkripsi Apache Spark di Athena. Melakukannya mengenkripsi data dalam transit antara node Spark dan juga mengenkripsi data yang disimpan secara lokal oleh Spark. Untuk meningkatkan keamanan data ini, Athena menggunakan konfigurasi enkripsi berikut:

```
spark.io.encryption.keySizeBits="256"
spark.io.encryption.keygen.algorithm="HmacSHA384"
```

Untuk mengaktifkan enkripsi Spark, Anda dapat menggunakan konsol Athena,AWS CLI, atau API Athena.

Menggunakan konsol Athena untuk mengaktifkan enkripsi Spark

Untuk membuat notebook baru yang mengaktifkan enkripsi Spark

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.
3. Lakukan salah satu dari berikut:
   - DalamPenjelajah notebook, pilihBuat notebook.
   - DalamEditor buku catatan, pilihBuat notebook, atau pilih ikon plus (+) untuk menambahkan notebook.
4. UntukNama Notebook, masukkan nama untuk notebook.
5. PerluasSifat percikanpilihan.
6. PilihAktifkan enkripsi Spark.
7. Pilih Create (Buat).

Sesi notebook yang Anda buat dienkripsi. Gunakan notebook baru seperti biasa. Ketika Anda kemudian meluncurkan sesi baru yang menggunakan notebook, sesi baru juga akan dienkripsi.

Anda juga dapat menggunakan konsol Athena untuk mengaktifkan enkripsi Spark untuk notebook yang ada.

Mengaktifkan enkripsi untuk notebook yang sudah ada

1. Buka sesi baruuntuk notebook yang dibuat sebelumnya.

2. Di editor notebook, dariSesimenu di kanan atas, pilihEdit sesi.
3. DalamEdit detail sesikotak dialog, memperluasSifat percikan.
4. PilihAktifkan enkripsi Spark.
5. Pilih Save (Simpan).

Konsol meluncurkan sesi baru yang mengaktifkan enkripsi. Sesi selanjutnya yang Anda buat untuk notebook ini juga akan mengaktifkan enkripsi.

MenggunakanAWS CLIuntuk mengaktifkan enkripsi Spark

Anda dapat menggunakanAWS CLIuntuk mengaktifkan enkripsi saat Anda meluncurkan sesi dengan menentukan properti Spark yang sesuai.

Untuk menggunakanAWS CLIuntuk mengaktifkan enkripsi Spark

1. Gunakan perintah seperti berikut ini untuk membuat objek JSON konfigurasi mesin yang menentukan properti enkripsi Spark.

```
ENGINE_CONFIGURATION_JSON=$(
  cat <<EOF
{
    "CoordinatorDpuSize": 1,
    "MaxConcurrentDpus": 20,
    "DefaultExecutorDpuSize": 1,
    "SparkProperties": {
      "spark.authenticate": "true",
      "spark.io.encryption.enabled": "true",
      "spark.network.crypto.enabled": "true"
    }
}
EOF
)
```

2. DalamAWS CLI, gunakan`athena start-session`perintah dan lulus dalam objek JSON yang Anda buat untuk`--engine-configuration`argumen, seperti pada contoh berikut:

```
aws athena start-session \
   --region "region" \
   --work-group "your-work-group" \
   --engine-configuration "$ENGINE_CONFIGURATION_JSON"
```

Menggunakan API Athena untuk mengaktifkan enkripsi Spark

Untuk mengaktifkan enkripsi Spark dengan API Athena, gunakanStartSessiontindakan dan nyaEngineConfiguration `SparkProperties`parameter untuk menentukan konfigurasi enkripsi di`StartSession`permintaan.

Mengkonfigurasi AWS Glue akses lintas akun di Athena untuk Spark

Topik ini menunjukkan bagaimana akun konsumen *666666666666* dan akun pemilik *999999999999* dapat dikonfigurasi untuk akses lintas akun. AWS Glue Ketika akun dikonfigurasi, akun konsumen dapat menjalankan kueri dari Athena untuk Spark pada database dan tabel pemilik AWS Glue .

1. Di AWS Glue, berikan akses ke peran konsumen

Di AWS Glue, pemilik membuat kebijakan yang menyediakan akses peran konsumen ke katalog AWS Glue data pemilik.

Untuk menambahkan AWS Glue kebijakan yang memungkinkan akses peran konsumen ke katalog data pemilik

1. Menggunakan akun pemilik katalog, masuk ke AWS Management Console.
2. Buka AWS Glue konsol di https://console.aws.amazon.com/glue/.
3. Di panel navigasi, perluas Katalog Data, lalu pilih Pengaturan katalog.
4. Pada halaman Setelan katalog data, di bagian Izin, tambahkan kebijakan seperti berikut ini. *Kebijakan ini memberikan peran untuk akses akun konsumen 666666666666 ke katalog data di akun pemilik 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. Konfigurasikan akun konsumen untuk akses

Di akun konsumen, buat kebijakan untuk mengizinkan akses ke pemilik AWS Glue Data Catalog, database, dan tabel, dan lampirkan kebijakan ke peran. Contoh berikut menggunakan akun konsumen *666666666666*.

Untuk membuat AWS Glue kebijakan untuk akses ke pemilik AWS Glue Data Catalog

1. Menggunakan akun konsumen, masuk ke AWS Management Console.
2. Buka konsol IAM di https://console.aws.amazon.com/iam/.
3. Di panel navigasi, perluas Manajemen akses, lalu pilih Kebijakan.
4. Pilih Buat kebijakan.
5. Pada halaman Tentukan izin, pilih JSON.
6. Di editor Kebijakan, masukkan pernyataan JSON seperti berikut ini yang memungkinkan AWS Glue tindakan pada katalog data akun pemilik.

```
{
    "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. Pilih Selanjutnya.
8. Pada halaman Tinjau dan buat, untuk nama Kebijakan, masukkan nama untuk kebijakan tersebut.
9. Pilih Buat kebijakan.

Selanjutnya, Anda menggunakan konsol IAM di akun konsumen untuk melampirkan kebijakan yang baru saja Anda buat ke peran atau peran IAM yang akan digunakan akun konsumen untuk mengakses katalog data pemilik.

Untuk melampirkan AWS Glue kebijakan ke peran di akun konsumen

1. Di panel navigasi konsol IAM akun konsumen, pilih Peran.
2. Pada halaman Peran, temukan peran yang ingin Anda lampirkan kebijakan.
3. Pilih Tambahkan izin, lalu pilih Lampirkan kebijakan.
4. Temukan kebijakan yang baru saja Anda buat.
5. Pilih kotak centang kebijakan, lalu pilih Tambahkan izin.
6. Ulangi langkah-langkah untuk menambahkan kebijakan ke peran lain yang ingin Anda gunakan.

3. Konfigurasikan sesi dan buat kueri

Di Athena Spark, di akun pemohon, menggunakan peran yang ditentukan, buat sesi untuk menguji akses dengan membuat buku catatan atau mengedit sesi saat ini. Saat Anda mengonfigurasi properti sesi, tentukan salah satu dari berikut ini:

- Pemisah katalog lem — Dengan pendekatan ini, Anda menyertakan ID akun pemilik dalam kueri Anda. Gunakan metode ini jika Anda akan menggunakan sesi untuk kueri katalog data dari pemilik yang berbeda.
- ID katalog lem — Dengan pendekatan ini, Anda menanyakan database secara langsung. Metode ini lebih nyaman jika Anda akan menggunakan sesi untuk menanyakan hanya katalog data pemilik tunggal.

Menggunakan pendekatan pemisah AWS Glue katalog

Saat Anda mengedit properti sesi, tambahkan yang berikut ini:

```
{
    "spark.hadoop.aws.glue.catalog.separator": "/"
}
```

Saat Anda menjalankan kueri di sel, gunakan sintaks seperti itu dalam contoh berikut. Perhatikan bahwa dalam `FROM` klausa, ID katalog dan pemisah diperlukan sebelum nama database.

```
df = spark.sql('SELECT requestip, uri, method, status FROM `999999999999/
mydatabase`.cloudfront_logs LIMIT 5')
df.show()
```

Menggunakan pendekatan ID AWS Glue katalog

Saat Anda mengedit properti sesi, masukkan properti berikut. Ganti *999999999999* dengan ID akun pemilik.

```
{
    "spark.hadoop.hive.metastore.glue.catalogid": "999999999999"
}
```

Saat Anda menjalankan kueri di sel, gunakan sintaks seperti berikut ini. Perhatikan bahwa dalam `FROM` klausa, ID katalog dan pemisah tidak diperlukan sebelum nama database.

```
df = spark.sql('SELECT * FROM mydatabase.cloudfront_logs LIMIT 10')
df.show()
```

Sumber daya tambahan

Akses lintas akun ke katalog AWS Glue data

Mengelola izin lintas akun menggunakan keduanya AWS Glue dan Lake Formation di Panduan AWS Lake Formation Pengembang.

Konfigurasikan akses lintas akun ke berbagi AWS Glue Data Catalog menggunakan Amazon Athena AWS dalam Pola Panduan Preskriptif.

Kuota layanan untuk Amazon Athena for Apache Spark

Kuota layanan, juga dikenal sebagaimembatasi, adalah jumlah maksimum sumber daya layanan atau operasi yang AndaAkun AWSdapat menggunakan. Untuk informasi lebih lanjut tentang kuota layanan untuk lainnyaAWSlayanan yang dapat Anda gunakan dengan Amazon Athena for Spark, lihatAWSkuota layanandi dalamReferensi Umum Amazon Web Services.

> **Note**
>
> BaruAkun AWSmungkin memiliki kuota awal yang lebih rendah yang dapat meningkat seiring waktu. Amazon Athena for Apache Spark memantau penggunaan akun di masing-masingWilayah AWS, dan kemudian secara otomatis meningkatkan kuota berdasarkan penggunaan Anda. Jika persyaratan Anda melebihi batas yang ditentukan, hubungi dukungan pelanggan.

Tabel berikut mencantumkan kuota layanan untuk Amazon Athena untuk Apache Spark.

Nama	Default	Dapat Disesuaikan	Deskripsi
Apache Spark DPU konkurensi	160	Tidak	Jumlah maksimum unit pemrosesan data (DPU) yang dapat Anda konsumsi secara bersamaan untuk perhitungan Apache Spark untuk satu akun di saat iniWilayah AWS. DPU adalah ukuran relatif daya pemrosesan yang terdiri dari 4 vCPU kapasitas komputasi dan 16 GB memori.
Apache Spark sesi DPU konkurensi	60	Tidak	Jumlah maksimum DPU yang dapat Anda konsumsi secara bersamaan untuk perhitungan Apache Spark dalam satu sesi.

API notebook Athena

Daftar berikut berisi tautan referensi ke tindakan API notebook Athena. Untuk struktur data dan tindakan API Athena lainnya, lihatReferensi API Amazon Athena.

- CreateNotebook
- CreatePresignedNotebookUrl
- DeleteNotebook
- ExportNotebook
- GetCalculationExecution
- GetCalculationExecutionCode
- GetCalculationExecutionStatus
- GetNotebookMetadata
- GetSession
- GetSessionStatus
- ImportNotebook
- ListApplicationDPUSizes
- ListCalculationExecutions
- ListExecutors
- ListNotebookMetadata
- ListNotebookSessions
- ListSessions
- StartCalculationExecution
- StartSession
- StopCalculationExecution
- TerminateSession
- UpdateNotebook
- UpdateNotebookMetadata

Masalah yang diketahui di Athena untuk Spark

Halaman ini mendokumentasikan beberapa masalah yang diketahui di Athena untuk Apache Spark.

Pengecualian argumen ilegal saat membuat tabel

Meskipun Spark tidak mengizinkan database dibuat dengan properti lokasi kosong, database di AWS Glue dapat memiliki `LOCATION` properti kosong jika dibuat di luar Spark.

Jika Anda membuat tabel dan menentukan AWS Glue database yang memiliki `LOCATION` bidang kosong, pengecualian seperti berikut dapat terjadi: IllegalArgumentException: Tidak dapat membuat jalur dari string kosong.

Misalnya, perintah berikut melempar pengecualian jika database default di AWS Glue berisi `LOCATION` bidang kosong:

```
spark.sql("create table testTable (firstName STRING)")
```

Solusi yang disarankan A — Gunakan AWS Glue untuk menambahkan lokasi ke database yang Anda gunakan.

Untuk menambahkan lokasi ke AWS Glue database

1. Masuk ke AWS Management Console dan buka AWS Glue konsol di https://console.aws.amazon.com/glue/.
2. Di panel navigasi, pilih Basis data.
3. Dalam daftar database, pilih database yang ingin Anda edit.
4. Pada halaman detail untuk database, pilih Edit.
5. Pada halaman Perbarui database, untuk Lokasi, masukkan lokasi Amazon S3.
6. Pilih Perbarui Database.

Solusi yang disarankan B — Gunakan AWS Glue database berbeda yang memiliki lokasi yang sudah ada dan valid di Amazon S3. Misalnya, jika Anda memiliki database bernama`dbWithLocation`, gunakan perintah `spark.sql("use dbWithLocation")` untuk beralih ke database itu.

Solusi yang disarankan C - Bila Anda menggunakan Spark SQL untuk membuat tabel, tentukan nilai untuk`location`, seperti pada contoh berikut.

```
spark.sql("create table testTable (firstName STRING)
       location 's3://DOC-EXAMPLE-BUCKET/'").
```

Solusi yang disarankan D — Jika Anda menentukan lokasi saat membuat tabel, tetapi masalah masih terjadi, pastikan jalur Amazon S3 yang Anda berikan memiliki garis miring ke depan. Misalnya, perintah berikut melempar pengecualian argumen ilegal:

```
spark.sql("create table testTable (firstName STRING)
       location 's3://DOC-EXAMPLE-BUCKET'")
```

Untuk memperbaikinya, tambahkan garis miring ke lokasi (misalnya,`'s3:// DOC-EXAMPLE-BUCKET/'`).

Database dibuat di lokasi workgroup

Jika Anda menggunakan perintah seperti `spark.sql('create database db')` membuat database dan tidak menentukan lokasi untuk database, Athena membuat subdirektori di lokasi workgroup Anda dan menggunakan lokasi tersebut untuk database yang baru dibuat.

Masalah dengan tabel terkelola Hive di database AWS Glue default

Jika `Location` properti database default Anda tidak kosong dan menentukan lokasi yang valid di AWS Glue Amazon S3, dan Anda menggunakan Athena for Spark untuk membuat tabel terkelola Hive di database default AWS Glue Anda, data akan ditulis ke lokasi Amazon S3 yang ditentukan dalam workgroup Athena Spark, bukan ke lokasi yang ditentukan oleh database. AWS Glue

Masalah ini terjadi karena bagaimana Apache Hive menangani basis data defaultnya. Apache Hive membuat data tabel di lokasi root gudang Hive, yang dapat berbeda dari lokasi database default yang sebenarnya.

Bila Anda menggunakan Athena for Spark untuk membuat tabel terkelola Hive di bawah database default AWS Glue, metadata AWS Glue tabel dapat menunjuk ke dua lokasi yang berbeda. Hal ini dapat menyebabkan perilaku yang tidak terduga ketika Anda mencoba `INSERT` atau `DROP TABLE` operasi.

Langkah-langkah untuk mereproduksi masalah adalah sebagai berikut:

1. Di Athena for Spark, Anda menggunakan salah satu metode berikut untuk membuat atau menyimpan tabel terkelola Hive:
   - Pernyataan SQL seperti `CREATE TABLE $tableName`
   - PySpark Perintah seperti `df.write.mode("overwrite").saveAsTable($tableName)` itu tidak menentukan `path` opsi di Dataframe API.

Pada titik ini, AWS Glue konsol mungkin menunjukkan lokasi yang salah di Amazon S3 untuk tabel.

2. Di Athena for Spark, Anda menggunakan `DROP TABLE $table_name` pernyataan untuk menjatuhkan tabel yang Anda buat.
3. Setelah Anda menjalankan `DROP TABLE` pernyataan, Anda melihat bahwa file yang mendasari di Amazon S3 masih ada.

Untuk mengatasi masalah ini, lakukan salah satu hal berikut:

Solusi A — Gunakan AWS Glue database yang berbeda saat Anda membuat tabel terkelola Hive.

Solusi B - Tentukan lokasi kosong untuk database default di AWS Glue. Kemudian, buat tabel terkelola Anda di database default.

Ketidakcocokan format file CSV dan JSON antara Athena untuk Spark dan Athena SQL

Karena masalah yang diketahui dengan Spark open source, saat Anda membuat tabel di Athena untuk Spark pada data CSV atau JSON, tabel mungkin tidak dapat dibaca dari Athena SQL, dan sebaliknya.

Misalnya, Anda dapat membuat tabel di Athena untuk Spark dengan salah satu cara berikut:

- Dengan `USING csv` sintaks berikut:

```
spark.sql('''CREATE EXTERNAL TABLE $tableName (
$colName1 $colType1,
$colName2 $colType2,
$colName3 $colType3)
USING csv
PARTITIONED BY ($colName1)
LOCATION $s3_location''')
```

- Dengan sintaks DataFrameAPI berikut:

```
df.write.format('csv').saveAsTable($table_name)
```

Karena masalah yang diketahui dengan Spark open source, kueri dari Athena SQL pada tabel yang dihasilkan mungkin tidak berhasil.

Solusi yang disarankan - Coba buat tabel di Athena untuk Spark menggunakan sintaks Apache Hive. Untuk informasi selengkapnya, lihat MEMBUAT TABEL HIVEFORMAT di dokumentasi Apache Spark.

Pemecahan Masalah Athena untuk Spark

Gunakan informasi berikut untuk memecahkan masalah yang mungkin Anda miliki saat menggunakan buku catatan dan sesi di Athena.

Topik

Memecahkan masalah grup kerja berkemampuan SPARK

Gunakan informasi berikut untuk memecahkan masalah grup kerja berkemampuan SPARK di Athena.

Sesi berhenti merespons saat menggunakan peran IAM yang ada

Jika Anda tidak membuat yang baru `AWSAthenaSparkExecutionRole` untuk grup kerja yang diaktifkan Spark dan sebagai gantinya memperbarui atau memilih peran IAM yang ada, sesi Anda mungkin berhenti merespons. Dalam hal ini, Anda mungkin perlu menambahkan kebijakan kepercayaan dan izin berikut ke peran eksekusi workgroup yang diaktifkan Spark.

Tambahkan contoh kebijakan kepercayaan berikut. Kebijakan tersebut mencakup pemeriksaan wakil yang bingung untuk peran eksekusi. Ganti nilai untuk*111122223333*,*aws-region*, dan *workgroup-name* dengan Akun AWS ID, Wilayah AWS, dan workgroup yang Anda gunakan.

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

Tambahkan kebijakan izin seperti kebijakan default berikut untuk grup kerja yang diaktifkan notebook. Ubah lokasi Akun AWS dan ID Amazon S3 placeholder agar sesuai dengan yang Anda gunakan. Ganti nilai untukDOC-EXAMPLE-BUCKET,*aws-region*,*111122223333*, dan *workgroup-name* dengan bucket Amazon S3, Akun AWS ID Wilayah AWS, dan workgroup yang Anda gunakan.

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

Menggunakan pernyataan Spark EXPLOW untuk memecahkan masalah Spark SQL

Anda dapat menggunakan `EXPLAIN` pernyataan Spark dengan Spark SQL untuk memecahkan masalah kode Spark Anda. Contoh kode dan output berikut menunjukkan penggunaan ini.

Example — Pernyataan Spark SELECT

```
spark.sql("select * from select_taxi_table").explain(True)
```

Keluaran

```
Calculation started (calculation_id=20c1ebd0-1ccf-ef14-db35-7c1844876a7e) in
(session=24c1ebcb-57a8-861e-1023-736f5ae55386).
Checking calculation status...

Calculation completed.
== Parsed Logical Plan ==
'Project [*]
+- 'UnresolvedRelation [select_taxi_table], [], false

== Analyzed Logical Plan ==
VendorID: bigint, passenger_count: bigint, count: bigint
Project [VendorID#202L, passenger_count#203L, count#204L]
+- SubqueryAlias spark_catalog.spark_demo_database.select_taxi_table
   +- Relation spark_demo_database.select_taxi_table[VendorID#202L,
       passenger_count#203L,count#204L] csv

== Optimized Logical Plan ==
```

```
Relation spark_demo_database.select_taxi_table[VendorID#202L,
passenger_count#203L,count#204L] csv

== Physical Plan ==
FileScan csv spark_demo_database.select_taxi_table[VendorID#202L,
passenger_count#203L,count#204L]
Batched: false, DataFilters: [], Format: CSV,
Location: InMemoryFileIndex(1 paths)
[s3://DOC-EXAMPLE-BUCKET/select_taxi],
PartitionFilters: [], PushedFilters: [],
ReadSchema: struct<VendorID:bigint,passenger_count:bigint,count:bigint>
```

Example — Bingkai data percikan

Contoh berikut menunjukkan bagaimana menggunakan `EXPLAIN` dengan frame data Spark.

```
taxi1_df=taxi_df.groupBy("VendorID", "passenger_count").count()
taxi1_df.explain("extended")
```

Keluaran

```
Calculation started (calculation_id=d2c1ebd1-f9f0-db25-8477-3effc001b309) in
(session=24c1ebcb-57a8-861e-1023-736f5ae55386).
Checking calculation status...

Calculation completed.
== Parsed Logical Plan ==
'Aggregate ['VendorID, 'passenger_count],
['VendorID, 'passenger_count, count(1) AS count#321L]
+- Relation [VendorID#49L,tpep_pickup_datetime#50,tpep_dropoff_datetime#51,
passenger_count#52L,trip_distance#53,RatecodeID#54L,store_and_fwd_flag#55,
PULocationID#56L,DOLocationID#57L,payment_type#58L,fare_amount#59,
extra#60,mta_tax#61,tip_amount#62,tolls_amount#63,improvement_surcharge#64,
total_amount#65,congestion_surcharge#66,airport_fee#67] parquet

== Analyzed Logical Plan ==
VendorID: bigint, passenger_count: bigint, count: bigint
Aggregate [VendorID#49L, passenger_count#52L],
[VendorID#49L, passenger_count#52L, count(1) AS count#321L]
+- Relation [VendorID#49L,tpep_pickup_datetime#50,tpep_dropoff_datetime#51,
passenger_count#52L,trip_distance#53,RatecodeID#54L,store_and_fwd_flag#55,
PULocationID#56L,DOLocationID#57L,payment_type#58L,fare_amount#59,extra#60,
mta_tax#61,tip_amount#62,tolls_amount#63,improvement_surcharge#64,
```

```
total_amount#65,congestion_surcharge#66,airport_fee#67] parquet

== Optimized Logical Plan ==
Aggregate [VendorID#49L, passenger_count#52L],
[VendorID#49L, passenger_count#52L, count(1) AS count#321L]
+- Project [VendorID#49L, passenger_count#52L]
   +- Relation [VendorID#49L,tpep_pickup_datetime#50,tpep_dropoff_datetime#51,
passenger_count#52L,trip_distance#53,RatecodeID#54L,store_and_fwd_flag#55,
PULocationID#56L,DOLocationID#57L,payment_type#58L,fare_amount#59,extra#60,
mta_tax#61,tip_amount#62,tolls_amount#63,improvement_surcharge#64,
total_amount#65,congestion_surcharge#66,airport_fee#67] parquet

== Physical Plan ==
AdaptiveSparkPlan isFinalPlan=false
+- HashAggregate(keys=[VendorID#49L, passenger_count#52L], functions=[count(1)],
output=[VendorID#49L, passenger_count#52L, count#321L])
   +- Exchange hashpartitioning(VendorID#49L, passenger_count#52L, 1000),
      ENSURE_REQUIREMENTS, [id=#531]
      +- HashAggregate(keys=[VendorID#49L, passenger_count#52L],
         functions=[partial_count(1)], output=[VendorID#49L,
         passenger_count#52L, count#326L])
         +- FileScan parquet [VendorID#49L,passenger_count#52L] Batched: true,
            DataFilters: [], Format: Parquet,
            Location: InMemoryFileIndex(1 paths)[s3://DOC-EXAMPLE-BUCKET/
            notebooks/yellow_tripdata_2016-01.parquet], PartitionFilters: [],
            PushedFilters: [],
            ReadSchema: struct<VendorID:bigint,passenger_count:bigint>
```

Acara aplikasi Logging Spark di Athena

Editor notebook Athena memungkinkan logging Jupyter, Spark, dan Python standar. Anda dapat menggunakan `df.show()` untuk menampilkan PySpark DataFrame konten atau digunakan `print("Output")` untuk menampilkan nilai dalam output sel. Output `stdoutstderr`, dan `results` output untuk perhitungan Anda ditulis ke lokasi bucket hasil kueri Anda di Amazon S3.

Logging acara aplikasi Spark ke Amazon CloudWatch

Sesi Athena Anda juga dapat menulis log ke Amazon CloudWatch di akun yang Anda gunakan.

Memahami aliran log dan grup log

CloudWatch mengatur aktivitas log ke dalam aliran log dan grup log.

Aliran log - Aliran CloudWatch log adalah urutan peristiwa log yang berbagi sumber yang sama. Setiap sumber log terpisah di CloudWatch Log membentuk aliran log terpisah.

Grup log — Di CloudWatch Log, grup log adalah grup aliran log yang berbagi pengaturan retensi, pemantauan, dan kontrol akses yang sama.

Tidak ada batas jumlah pengaliran log yang dapat tergabung dalam satu grup log.

Di Athena, saat Anda memulai sesi buku catatan untuk pertama kalinya, Athena membuat grup log CloudWatch yang menggunakan nama grup kerja berkemampuan SPARK, seperti pada contoh berikut.

```
/aws-athena/workgroup-name
```

Grup log ini menerima satu aliran log untuk setiap eksekutor di sesi Anda yang menghasilkan setidaknya satu peristiwa log. Pelaksana adalah unit komputasi terkecil yang dapat diminta oleh sesi notebook dari Athena. Di CloudWatch, nama aliran log dimulai dengan ID sesi dan ID pelaksana.

Untuk informasi selengkapnya tentang grup CloudWatch log dan aliran log, lihat Bekerja dengan grup log dan aliran log di Panduan Pengguna CloudWatch Log Amazon.

Menggunakan objek logger standar di Athena untuk Spark

Dalam sesi Athena for Spark, Anda dapat menggunakan dua objek logger standar global berikut untuk menulis log ke Amazon: CloudWatch

- athena_user_logger - Mengirim log ke saja. CloudWatch Gunakan objek ini ketika Anda ingin mencatat informasi aplikasi Spark Anda secara langsung CloudWatch, seperti pada contoh berikut.

  ```
  athena_user_logger.info("CloudWatch log line.")
  ```

  Contoh menulis peristiwa log untuk CloudWatch menyukai yang berikut:

  ```
  AthenaForApacheSpark: 2022-01-01 12:00:00,000 INFO builtins: CloudWatch log line.
  ```

- athena_shared_logger — Mengirim log yang sama baik ke maupun ke untuk tujuan dukungan. CloudWatch AWS Anda dapat menggunakan objek ini untuk berbagi log dengan tim AWS layanan untuk pemecahan masalah, seperti pada contoh berikut.

  ```
  athena_shared_logger.info("Customer debug line.")
  var = [...some variable holding customer data...]
  ```

```
athena_shared_logger.info(var)
```

Contoh mencatat debug baris dan nilai `var` variabel ke CloudWatch Log dan mengirimkan salinan dari setiap baris ke AWS Support.

> **Note**
>
> Untuk privasi Anda, kode perhitungan dan hasil Anda tidak dibagikan AWS. Pastikan panggilan Anda hanya `athena_shared_logger` menulis informasi yang ingin Anda lihat AWS Support.

Logger yang disediakan menulis peristiwa melalui Apache Log4j dan mewarisi tingkat logging antarmuka ini. Nilai tingkat log yang mungkin adalah`DEBUG`,`ERROR`,`FATAL`,`INFO`, dan `WARN` atau`WARNING`. Anda dapat menggunakan fungsi bernama yang sesuai pada logger untuk menghasilkan nilai-nilai ini.

> **Note**
>
> Jangan membalas nama `athena_user_logger` atau. `athena_shared_logger` Melakukannya membuat objek logging tidak dapat menulis CloudWatch untuk sisa sesi.

Contoh: mencatat peristiwa buku catatan ke CloudWatch

Prosedur berikut menunjukkan cara mencatat peristiwa notebook Athena ke Amazon CloudWatch Logs.

Untuk mencatat peristiwa notebook Athena ke Amazon Logs CloudWatch

1. Ikuti Memulai dengan Apache Spark di Amazon Athena untuk membuat workgroup berkemampuan Spark di Athena dengan nama unik. Tutorial ini menggunakan nama `athena-spark-example` workgroup.
2. Ikuti langkah-langkah Membuat buku catatan Anda sendiri untuk membuat buku catatan dan meluncurkan sesi baru.
3. Di editor notebook Athena, di sel notebook baru, masukkan perintah berikut:

```
athena_user_logger.info("Hello world.")
```

4. Jalankan sel.
5. Ambil ID sesi saat ini dengan melakukan salah satu hal berikut:
   - Lihat output sel (misalnya,`... session=72c24e73-2c24-8b22-14bd-443bdcd72de4`).
   - Di sel baru, jalankan perintah ajaib`%session_id`.
6. Simpan ID sesi.
7. Dengan hal Akun AWS yang sama yang Anda gunakan untuk menjalankan sesi notebook, buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
8. Di panel navigasi CloudWatch konsol, pilih Grup log.
9. Dalam daftar grup log, pilih grup log yang memiliki nama grup kerja Athena berkemampuan SPARK Anda, seperti pada contoh berikut.

   ```
   /aws-athena/athena-spark-example
   ```

   Bagian Aliran log berisi daftar satu atau beberapa tautan aliran log untuk grup kerja. Setiap nama aliran log berisi ID sesi, ID pelaksana, dan UUID unik yang dipisahkan oleh karakter garis miring maju.

   Misalnya, jika ID sesi `5ac22d11-9fd8-ded7-6542-0412133d3177` dan ID pelaksana adalah`f8c22d11-9fd8-ab13-8aba-c4100bfba7e2`, nama aliran log menyerupai contoh berikut.

   ```
   5ac22d11-9fd8-ded7-6542-0412133d3177/f8c22d11-9fd8-ab13-8aba-c4100bfba7e2/f012d7cb-cefd-40b1-90b9-67358f003d0b
   ```
10. Pilih tautan aliran log untuk sesi Anda.
11. Pada halaman Log peristiwa, lihat kolom Pesan.

    Peristiwa log untuk sel yang Anda jalankan menyerupai berikut ini:

    ```
    AthenaForApacheSpark: 2022-01-01 12:00:00,000 INFO builtins: Hello world.
    ```
12. Kembali ke editor notebook Athena.
13. Di sel baru, masukkan kode berikut. Kode mencatat variabel ke CloudWatch:

    ```
    x = 6
    athena_user_logger.warn(x)
    ```

14. Jalankan sel.
15. Kembali ke halaman peristiwa Log CloudWatch konsol untuk aliran log yang sama.
16. Aliran log sekarang berisi entri peristiwa log dengan pesan seperti berikut:

```
AthenaForApacheSpark: 2022-01-01 12:00:00,000 WARN builtins: 6
```

MenggunakanCloudTrailuntuk memecahkan masalah panggilan API notebook Athena

Untuk memecahkan masalah panggilan API notebook, Anda dapat memeriksa AthenaCloudTraillog untuk menyelidiki anomali atau menemukan tindakan yang diprakarsai oleh pengguna. Untuk informasi rinci tentang penggunaanCloudTraildengan Athena, lihatMencatat panggilan API Amazon Athena dengan AWS CloudTrail.

Contoh berikut menunjukkanCloudTrailentri log untuk API notebook Athena:

- StartSession
- TerminateSession
- ImportNotebook
- UpdateNotebook
- StartCalculationExecution

StartSession

Contoh berikut menunjukkanCloudTraillog untuk notebookStartSessionacara.

```
{
    "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

Contoh berikut menunjukkanCloudTraillog untuk notebookTerminateSessionacara.

```
{
    "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

Contoh berikut menunjukkanCloudTraillog untuk notebookImportNotebookacara. Untuk keamanan, beberapa konten disembunyikan.

```
{
    "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

Contoh berikut menunjukkanCloudTraillog untuk notebookUpdateNotebookacara. Untuk keamanan, beberapa konten disembunyikan.

```
{
    "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

Contoh berikut menunjukkanCloudTraillog untuk notebookStartCalculationExecutionacara. Untuk keamanan, beberapa konten disembunyikan.

```
{
    "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"
            }
        }
    },
```

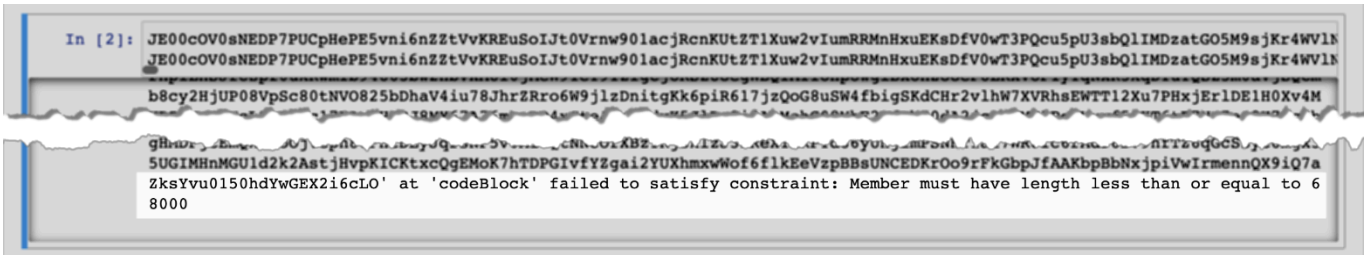
```
    "eventTime": "2022-10-14T16:52:37Z",
    "eventSource": "athena.amazonaws.com",
    "eventName": "StartCalculationExecution",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.14",
    "userAgent": "Boto3/1.24.84 Python/3.8.14 Linux/4.14.225-175.364.amzn2.aarch64
 Botocore/1.27.84",
    "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"
    }
}
```


Mengatasi batas ukuran blok kode 68k

Athena untuk Spark memiliki kode perhitungan batas ukuran blok dikenal 68000 karakter. Ketika Anda menjalankan perhitungan dengan blok kode di atas batas ini, Anda dapat menerima pesan galat berikut:

'...' di 'CodeBlock' gagal memenuhi kendala: Anggota harus memiliki panjang kurang dari atau sama dengan 68000

Gambar berikut menunjukkan kesalahan ini di editor notebook konsol Athena.



Kesalahan yang sama dapat terjadi saat Anda menggunakanAWS CLIuntuk menjalankan perhitungan yang memiliki blok kode besar, seperti pada contoh berikut.

```
aws athena start-calculation-execution \
    --session-id "{SESSION_ID}" \
    --description "{SESSION_DESCRIPTION}" \
    --code-block "{LARGE_CODE_BLOCK}"
```

Perintah memberikan pesan galat berikut:

*{LARGE_CODE_BLOCK}*di 'CodeBlock' gagal memenuhi kendala: Anggota harus memiliki panjang kurang dari atau sama dengan 68000

Solusi

Untuk mengatasi masalah ini, unggah file yang memiliki kueri atau kode perhitungan Anda ke Amazon S3. Kemudian, gunakan boto3 untuk membaca file dan menjalankan SQL atau kode Anda.

Contoh berikut mengasumsikan bahwa Anda telah mengunggah file yang memiliki kueri SQL atau kode Python ke Amazon S3.

Contoh SQL

Contoh kode berikut membaca`large_sql_query.sql`file dari bucket Amazon S3 dan kemudian menjalankan kueri besar yang berisi file tersebut.

```
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)
```

PySparkcontoh

Contoh kode berikut membaca`large_py_spark.py`file dari Amazon S3 dan kemudian menjalankan blok kode besar yang ada di file.

```
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)
```

Sesi pemecahan masalah

Gunakan informasi dalam topik ini untuk memecahkan masalah sesi.

Sesi dalam keadaan tidak sehat

Jika Anda menerima pesan kesalahan Session dalam keadaan tidak sehat. Harap buat sesi baru, hentikan sesi Anda yang ada dan buat yang baru.

Koneksi ke server notebook tidak dapat dibuat

Ketika Anda membuka buku catatan, Anda mungkin melihat pesan galat berikut:

```
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.
```

Penyebab

Ketika Athena membuka buku catatan, Athena membuat sesi dan terhubung ke notebook menggunakan URL notebook yang telah ditandatangani sebelumnya. Koneksi ke notebook menggunakan protokol WSS (WebSocketSecure).

Kesalahan dapat terjadi karena alasan berikut:

- Firewall lokal (misalnya, firewall di seluruh perusahaan) memblokir lalu lintas WSS.

- Proxy atau perangkat lunak anti-virus pada komputer lokal Anda memblokir koneksi WSS.

Solusi

Asumsikan Anda memiliki koneksi WSS di `us-east-1` Wilayah seperti berikut:

```
wss://94c2bcdf-66f9-4d17-9da6-7e7338060183.analytics-gateway.us-east-1.amazonaws.com/
api/kernels/33c78c82-b8d2-4631-bd22-1565dc6ec152/channels?session_id=
7f96a3a048ab4917b6376895ea8d7535
```

Untuk mengatasi kesalahan, gunakan salah satu strategi berikut.

- Gunakan sintaks pola wild card untuk memungkinkan daftar lalu lintas WSS pada port di `443` seluruh Wilayah AWS dan. Akun AWS

  ```
  wss://*amazonaws.com
  ```

- Gunakan sintaks pola wild card untuk memungkinkan daftar lalu lintas WSS pada port `443` dalam satu Wilayah AWS dan Akun AWS di seberang Wilayah AWS yang Anda tentukan. Contoh berikut menggunakan `us-east-1`.

  ```
  wss://*analytics-gateway.us-east-1.amazonaws.com
  ```

Tabel pemecahan masalah

Tidak dapat membuat kesalahan jalur saat membuat tabel

Pesan galat IllegalArgumentException:: Tidak dapat membuat jalur dari string kosong.

Penyebab: Kesalahan ini dapat terjadi ketika Anda menggunakan Apache Spark di Athena untuk membuat tabel dalam AWS Glue database, dan database memiliki properti kosong. `LOCATION`

Solusi yang Disarankan: Untuk informasi dan solusi lebih lanjut, lihatPengecualian argumen ilegal saat membuat tabel.

AccessDeniedException saat menanyakan tabel AWS Glue

Pesan kesalahan: pyspark.sql.utils. AnalysisException: Tidak dapat memverifikasi keberadaan database default: com.amazonaws.services.glue.model. AccessDeniedException*: User:*

arn:aws:sts: ::assumed-role/ aws-account-id- AWSAthenaSparkExecutionRole unique-identifier/- unique-identifier tidak diizinkan untuk melakukan: lem: on resource: arn:aws:glue: aws-region ::catalog karena aws-account-idtidak AthenaExecutor ada kebijakan berbasis identitas yang mengizinkan lem: action (LayananGetDatabase :; Kode Status: 400; Kode Kesalahan:; ID Permintaan: request-id; Proxy: null) GetDatabase AWSGlue AccessDeniedException

Penyebab: Peran eksekusi untuk grup kerja berkemampuan SPARK Anda tidak memiliki izin untuk mengakses sumber daya. AWS Glue

Solusi yang Disarankan: Untuk mengatasi masalah ini, berikan akses peran eksekusi ke AWS Glue sumber daya, lalu edit kebijakan bucket Amazon S3 Anda untuk memberikan akses ke peran eksekusi Anda.

Prosedur berikut menjelaskan langkah-langkah ini secara lebih rinci.

Untuk memberikan akses peran eksekusi Anda ke AWS Glue sumber daya

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Jika panel navigasi konsol tidak terlihat, pilih menu ekspansi di sebelah kiri.

3. Di panel navigasi konsol Athena, pilih Workgroups.
4. Pada halaman Workgroups, pilih link workgroup yang ingin Anda lihat.
5. Pada halaman Rincian Ikhtisar untuk grup kerja, pilih tautan ARN Peran. Tautan membuka peran eksekusi Spark di konsol IAM.

6. Di bagian Kebijakan izin, pilih nama kebijakan peran yang ditautkan.
7. Pilih Edit kebijakan, lalu pilih JSON.
8. Tambahkan AWS Glue akses ke peran. Biasanya, Anda menambahkan izin untuk `glue:GetTable` tindakan `glue:GetDatabase` dan tindakan. Untuk informasi selengkapnya tentang mengonfigurasi peran IAM, lihat Menambahkan dan menghapus izin identitas IAM di Panduan Pengguna IAM.
9. Pilih Kebijakan ulasan, lalu pilih Simpan perubahan.
10. Edit kebijakan bucket Amazon S3 Anda untuk memberikan akses ke peran eksekusi. Perhatikan bahwa Anda harus memberikan akses peran ke bucket dan objek di bucket. Untuk langkah-langkahnya, lihat Menambahkan kebijakan bucket menggunakan konsol Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Mendapatkan Dukungan

Untuk bantuan dari AWS, pilih Support, Support Center dari AWS Management Console. Untuk memudahkan pengalaman Anda, siapkan informasi berikut:

- ID kueri Athena
- ID Sesi
- ID Perhitungan

Catatan rilis

Menjelaskan fitur Amazon Athena, perbaikan, dan perbaikan bug berdasarkan tanggal rilis.

Topik

Catatan rilis Athena untuk tahun 2024

Juni 26, 2024

Diterbitkan: 2024-06-26

Kapasitas yang disediakan sekarang umumnya tersedia di Wilayah Amerika Selatan (São Paulo) dan Eropa (Spanyol). Kapasitas yang disediakan memungkinkan Anda menjalankan kueri SQL pada kapasitas komputasi yang dikelola sepenuhnya dan menyediakan kemampuan manajemen beban kerja yang membantu Anda memprioritaskan, mengontrol, dan menskalakan beban kerja interaktif terpenting Anda. Anda dapat menambahkan kapasitas kapan saja untuk meningkatkan jumlah kueri yang Anda jalankan secara bersamaan, mengontrol beban kerja mana yang menggunakan kapasitas, dan berbagi kapasitas antar beban kerja.

Untuk informasi selengkapnya, lihat Mengelola kapasitas pemrosesan kueri. Untuk informasi harga, kunjungi halaman harga Amazon Athena.

April 26, 2024

Diterbitkan: 2024-04-26

Athena merilis driver JDBC versi 3.2.0. Untuk informasi lebih lanjut tentang versi driver ini, lihatCatatan rilis Amazon Athena JDBC 3.x. Untuk mengunduh driver JDBC 3.x, lihat. Unduhan driver JDBC 3.x

April 24, 2024

Diterbitkan: 2024-04-24

Athena mengumumkan perbaikan dan peningkatan berikut.

- Parket — Athena sekarang mendukung pembacaan yang kompatibel ke belakang di Parket untuk bidang primitif berulang yang tidak dijelaskan yang tidak terkandung dalam daftar atau grup peta. Perubahan ini mencegah hasil yang salah secara diam-diam dikembalikan dan meningkatkan pesan kesalahan untuk ketidakcocokan skema.

  Untuk informasi selengkapnya, lihat Mendukung pembacaan yang kompatibel ke belakang untuk bidang primitif berulang yang tidak dianotasi di Parket di.com. GitHub
- Iceberg OPTIMIZE — Menyelesaikan masalah dengan `OPTIMIZE` kueri yang menyebabkan data hilang saat filter kunci non-partisi digunakan dalam klausa. `WHERE` Untuk informasi selengkapnya, lihat MENGOPTIMALKAN.

April 16, 2024

Diterbitkan: 2024-04-16

Gunakan fitur passthrough kueri gabungan Amazon Athena baru untuk menjalankan seluruh kueri secara langsung pada sumber data yang mendasarinya. Kueri passthrough gabungan membantu Anda memanfaatkan fungsi unik, bahasa kueri, dan kemampuan kinerja sumber data asli. Misalnya, Anda dapat menjalankan kueri Athena di DynamoDB menggunakan bahasa PartiQL. Kueri passthrough gabungan juga berguna saat Anda ingin menjalankan `SELECT` kueri yang menggabungkan, menggabungkan, atau memanggil fungsi sumber data Anda yang tidak tersedia di Athena. Menggunakan kueri passthrough dapat mengurangi jumlah data yang diproses oleh Athena dan menghasilkan waktu kueri yang lebih cepat.

Untuk informasi selengkapnya, lihat Menjalankan kueri passthrough federasi. Untuk meningkatkan konektor yang Anda gunakan hari ini ke versi terbaru, lihatMemperbarui konektor sumber data.

April 10, 2024

Diterbitkan: 2024-04-10

Athena mengumumkan fitur dan peningkatan berikut.

ODBC 1.2.3.1000 driver

Rilis driver ODBC 1.2.3.1000 untuk Athena.

Masalah yang diselesaikan:

- Masalah koneksi server proxy — Ketika server proxy digunakan tanpa sertifikat root, konektor gagal membuat koneksi.

Untuk informasi lebih lanjut, dan untuk mengunduh driver ODBC 1.x, catatan rilis, dan dokumentasi, lihat. Athena ODBC 1.x driver

Pengemudi JDBC 2.1.5

Rilis driver JBDC 2.1.5 untuk Athena.

Pembaruan dan penyempurnaan:

- Memperbarui AWS Java SDK untuk menggunakan versi 1.12.687.
- Perpustakaan Jackson yang diperbarui untuk menggunakan versi 2.16.0.
- Pustaka Logback yang diperbarui untuk menggunakan versi 1.3.14.

Untuk informasi selengkapnya, dan untuk mengunduh driver JDBC 2.x, catatan rilis, dan dokumentasi, lihat. Athena JDBC 2.x driver

April 8, 2024

Diterbitkan: 2024-04-08

Athena mengumumkan driver ODBC versi 2.0.3.0. Untuk informasi selengkapnya, lihat catatan 2.0.3.0 rilis. Untuk mengunduh driver ODBC v2 baru, lihatUnduhan driver ODBC 2.x. Untuk informasi koneksi, lihatAmazon Athena ODBC 2.x.

Maret 15, 2024

Diterbitkan: 2024-03-18

Amazon Athena mengumumkan ketersediaan Athena SQL di Wilayah Kanada Barat (Calgary).

Untuk daftar lengkap yang Layanan AWS tersedia di masing-masing Wilayah AWS, lihat AWS Layanan menurut Wilayah.

Februari 15, 2024

Diterbitkan: 2024-02-15

Athena merilis driver JDBC versi 3.1.0.

Driver Amazon Athena JDBC versi 3.1.0 menambahkan dukungan untuk Microsoft Active Directory Federation Services (AD FS) Windows Integrated Authentication dan otentikasi berbasis formulir. Rilis 3.1.0 juga mencakup perbaikan kecil dan perbaikan bug lainnya.

Untuk mengunduh driver JDBC v3, lihat. Unduhan driver JDBC 3.x

Januari 31, 2024

Diterbitkan: 2024-01-31

Athena mengumumkan fitur dan peningkatan berikut.

- Upgrade Hudi — Anda sekarang dapat menggunakan Athena SQL untuk query Hudi 0.14.0 tabel. Untuk informasi tentang menggunakan Athena SQL untuk menanyakan tabel Hudi, lihat. Menggunakan Athena untuk menanyakan kumpulan data Apache Hudi

Catatan rilis Athena untuk tahun 2023

14 Desember 2023

Diterbitkan: 2023-12-14

Athena mengumumkan perbaikan dan peningkatan berikut.

Athena merilis driver JDBC versi 2.1.3. Pengemudi menyelesaikan masalah berikut:

- Logging telah ditingkatkan untuk menghindari konflik dengan Spring Boot dan pencatatan aplikasi Gradle.
- Saat menggunakan metode `executeBatch()` JDBC untuk memasukkan catatan, driver salah memasukkan hanya satu catatan. Karena Athena tidak mendukung eksekusi batch kueri, driver sekarang melaporkan kesalahan saat Anda menggunakan. `executeBatch()` Untuk mengatasi batasan, Anda dapat mengirimkan kueri tunggal dalam satu lingkaran.

Untuk mengunduh driver JDBC baru, catatan rilis, dan dokumentasi, lihat. Athena JDBC 2.x driver

Desember 9, 2023

Diterbitkan: 2023-12-09

Merilis driver ODBC 1.2.1.1000 untuk Athena.

Fitur dan perangkat tambahan:

- Dukungan RStudio yang diperbarui - Driver ODBC sekarang mendukung RStudio di macOS.
- Dukungan katalog dan skema tunggal - Konektor sekarang dapat mengembalikan satu katalog dan skema. Untuk informasi selengkapnya, lihat panduan instalasi dan konfigurasi yang dapat diunduh.

Masalah yang diselesaikan:

- Pernyataan yang disiapkan — Ketika pernyataan yang disiapkan dengan array parameter menggunakan skema berdasarkan kolom dijalankan, konektor mengembalikan hasil kueri yang salah.
- Ukuran kolom - Ketika kolom `$file_modified_time` sistem dipilih, konektor mengembalikan ukuran kolom yang salah.
- SqlPrepare — Saat mengikat parameter yang terkait dengan `SQLPrepare` dalam `SELECT` kueri, konektor mengembalikan kesalahan.

Untuk informasi lebih lanjut, dan untuk mengunduh driver baru, catatan rilis, dan dokumentasi, lihatAthena ODBC 1.x driver.

Desember 7, 2023

Diterbitkan: 2023-12-07

Athena mengumumkan driver ODBC versi 2.0.2.1. Untuk informasi selengkapnya, lihat catatan 2.0.2.1 rilis. Untuk mengunduh driver ODBC v2 baru, lihatUnduhan driver ODBC 2.x. Untuk informasi koneksi, lihatAmazon Athena ODBC 2.x.

5 Desember 2023

Diterbitkan: 2023-12-05

Anda sekarang dapat membuat workgroup Athena SQL yang menggunakan mode otentikasi. AWS IAM Identity Center Kelompok kerja ini mendukung fitur propagasi identitas tepercaya dari IAM Identity Center. Propagasi identitas tepercaya memungkinkan identitas digunakan di seluruh layanan AWS analitik seperti Amazon Athena dan Amazon EMR Studio.

Untuk informasi selengkapnya, lihat Menggunakan IAM Identity Center mengaktifkan kelompok kerja Athena.

28 November 2023

Diterbitkan: 2023-11-28

Anda sekarang dapat melakukan kueri data di kelas penyimpanan Amazon S3 Express One Zone untuk hasil kueri yang cepat. S3 Express One Zone adalah kelas penyimpanan Zona Ketersediaan Tunggal berkinerja tinggi yang dibuat khusus untuk memberikan akses data milidetik satu digit yang konsisten untuk data yang paling sering diakses dan aplikasi yang sensitif terhadap latensi. Untuk memulai, pindahkan data Anda ke penyimpanan S3 Express One Zone dan katalogkan datanya AWS Glue Data Cataloguntuk pengalaman kueri yang mulus di Athena.

Untuk informasi selengkapnya, lihat Meminta data S3 Express One Zone.

27 November 2023

Diterbitkan: 2023-11-27

Athena mengumumkan fitur dan peningkatan berikut.

- Tampilan Glue Data Catalog — Tampilan Glue Data Catalog memberikan satu tampilan umum di seluruh AWS layanan seperti Amazon Athena dan Amazon Redshift. Dalam tampilan Glue Data Catalog, izin akses ditentukan oleh pengguna yang membuat tampilan, bukan pengguna yang menanyakan tampilan. Tampilan ini memberikan kontrol akses yang lebih besar, membantu memastikan catatan lengkap, menawarkan keamanan yang ditingkatkan, dan dapat mencegah akses ke tabel yang mendasarinya.

Untuk informasi selengkapnya, lihat Menggunakan AWS Glue Data Catalog tampilan.

- CloudTrail Dukungan danau - Anda sekarang dapat menggunakan Amazon Athena untuk menganalisis data di AWS CloudTrail Danau. AWS CloudTrail Lake adalah danau data terkelola CloudTrail yang dapat Anda gunakan untuk mengumpulkan, menyimpan, dan menganalisis log aktivitas untuk audit, keamanan, dan investigasi operasional. Untuk menanyakan log aktivitas CloudTrail Danau Anda dari Athena, Anda tidak perlu memindahkan data atau membuat pipeline pemrosesan data terpisah. Tidak diperlukan operasi ETL.

  Untuk memulai, aktifkan federasi data di CloudTrail Lake. Saat Anda membagikan metadata penyimpanan data acara CloudTrail Lake dengan AWS Glue Data Catalog, CloudTrail buat AWS Glue Data Catalog sumber daya yang diperlukan dan daftarkan datanya. AWS Lake Formation Di Lake Formation, Anda dapat menentukan pengguna dan peran yang dapat menggunakan Athena untuk menanyakan penyimpanan data acara Anda.

  Untuk informasi selengkapnya, lihat Mengaktifkan federasi kueri Danau di Panduan AWS CloudTrail Pengguna.

17 November 2023

Diterbitkan: 2023-11-17

Athena mengumumkan fitur dan peningkatan berikut.

Fitur

- Pengoptimal berbasis biaya - Athena mengumumkan ketersediaan umum pengoptimalan berbasis biaya menggunakan statistik dari. AWS Glue Untuk mengoptimalkan kueri Anda di Athena SQL, Anda dapat meminta Athena mengumpulkan statistik tingkat tabel atau kolom untuk tabel Anda. AWS Glue Jika semua tabel dalam kueri Anda memiliki statistik, Athena menggunakan statistik untuk memeriksa rencana eksekusi alternatif dan memilih salah satu yang paling mungkin menjadi yang tercepat.

  Untuk informasi selengkapnya, lihat Menggunakan pengoptimal berbasis biaya.
- Integrasi Amazon EMR Studio - Anda sekarang dapat menggunakan Athena di Amazon EMR Studio tanpa harus menggunakan konsol Athena secara langsung. Dengan integrasi Athena di Amazon EMR, Anda dapat melakukan tugas-tugas berikut:
  - Lakukan kueri Athena SQL

- Lihat hasil kueri
- Lihat riwayat kueri
- Lihat kueri yang disimpan
- Lakukan kueri parameter
- Melihat database, tabel, dan tampilan untuk katalog data

Untuk informasi selengkapnya, lihat Amazon EMR Studio dalam topik Layanan AWS Integrasi dengan Athena.

- Kontrol akses bersarang — Athena mengumumkan dukungan untuk kontrol akses Lake Formation untuk data bersarang. Di Lake Formation, Anda dapat menentukan dan menerapkan filter data pada kolom bersarang yang memiliki tipe `struct` data. Anda dapat menggunakan pemfilteran data untuk membatasi akses pengguna ke sub-struktur kolom bersarang. Untuk informasi tentang cara membuat filter data untuk data bersarang, lihat Membuat filter data di Panduan AWS Lake Formation Pengembang.
- Metrik penggunaan kapasitas yang disediakan — Athena mengumumkan metrik baru untuk pemesanan kapasitas. CloudWatch Anda dapat menggunakan metrik baru untuk melacak jumlah DPU yang telah Anda sediakan dan jumlah DPU yang digunakan oleh kueri Anda. Saat kueri selesai, Anda juga dapat melihat jumlah DPU yang digunakan kueri.

  Untuk informasi selengkapnya, lihat Memantau kueri Athena dengan metrik CloudWatch .

Perbaikan

- Perubahan pesan kesalahan - Pesan `Insufficient Lake Formation permissions` kesalahan sekarang dibaca `Table not found` atau`Schema not found`. Perubahan ini dilakukan untuk mencegah pelaku jahat menyimpulkan keberadaan tabel atau sumber daya database dari pesan kesalahan.

16 November 2023

Diterbitkan: 2023-11-16

Athena merilis driver JDBC baru yang meningkatkan pengalaman menghubungkan, menanyakan, dan memvisualisasikan data dari pengembangan SQL yang kompatibel dan aplikasi intelijen bisnis. Driver baru mudah untuk ditingkatkan. Pengemudi dapat membaca hasil kueri langsung dari Amazon S3, membuat hasil kueri tersedia untuk Anda lebih cepat.

Untuk informasi selengkapnya, lihat Athena JDBC 3.x driver.

31 Oktober 2023

Diterbitkan: 2023-10-31

Amazon Athena mengumumkan pemesanan 1 jam untuk kapasitas yang disediakan. Mulai hari ini, Anda dapat memesan dan melepaskan kapasitas yang disediakan setelah satu jam. Perubahan ini membuatnya lebih mudah untuk mengoptimalkan biaya untuk beban kerja yang permintaannya berubah seiring waktu.

Kapasitas yang disediakan adalah fitur Athena yang menyediakan kemampuan manajemen beban kerja yang membantu Anda memprioritaskan, mengontrol, dan menskalakan beban kerja interaktif Anda yang paling penting. Anda dapat menambahkan kapasitas kapan saja untuk meningkatkan jumlah kueri yang Anda jalankan secara bersamaan, mengontrol beban kerja mana yang menggunakan kapasitas, dan berbagi kapasitas antar beban kerja.

Untuk informasi selengkapnya, lihat Mengelola kapasitas pemrosesan kueri. Untuk informasi harga, kunjungi halaman Harga Amazon Athena.

25 Oktober 2023

Diterbitkan: 2023-10-26

Athena mengumumkan perbaikan dan peningkatan berikut.

paket jackson-core - Teks JSON dengan nilai numerik yang lebih besar dari 1000 karakter sekarang akan gagal. Perbaikan ini mengatasi masalah keamanan sonatype-2022-6438.

17 Oktober 2023

Diterbitkan: 2023-10-17

Athena mengumumkan driver ODBC versi 2.0.2.0. Untuk informasi selengkapnya, lihat catatan 2.0.2.0 rilis. Untuk mengunduh driver ODBC v2 baru, lihatUnduhan driver ODBC 2.x. Untuk informasi koneksi, lihatAmazon Athena ODBC 2.x.

26 September 2023

Diterbitkan: 2023-09-26

Athena mengumumkan fitur dan peningkatan berikut.

- Lake Formation membaca dukungan untuk tabel Danau Delta. Untuk informasi lebih lanjut tentang menggunakan tabel Danau Delta dengan Athena, lihat. Menanyakan tabel Delta Lake Linux Foundation

23 Agustus 2023

Diterbitkan: 2023-08-23

Amazon Athena mengumumkan ketersediaan Athena SQL di Wilayah Israel (Tel Aviv).

Untuk daftar lengkap yang Layanan AWS tersedia di masing-masing Wilayah AWS, lihat AWS Layanan menurut Wilayah.

10 Agustus 2023

Diterbitkan: 2023-08-10

Athena mengumumkan perbaikan dan peningkatan berikut.

Driver ODBC versi 2.0.1.1

Athena mengumumkan driver ODBC versi 2.0.1.1. Untuk informasi selengkapnya, lihat catatan 2.0.1.1 rilis. Untuk mengunduh driver ODBC v2 baru, lihatUnduhan driver ODBC 2.x. Untuk informasi koneksi, lihatAmazon Athena ODBC 2.x.

Driver JDBC versi 2.1.1

Athena merilis driver JDBC versi 2.1.1. Pengemudi menyelesaikan masalah berikut:

- Kesalahan yang terjadi ketika tabel dibuat dengan pernyataan yang berisi ekspresi reguler.
- Masalah yang menyebabkan parameter `ApplicationName` koneksi diterapkan secara tidak benar.

Untuk mengunduh driver JDBC baru, catatan rilis, dan dokumentasi, lihat. Menghubungkan ke Amazon Athena dengan JDBC

31 Juli 2023

Diterbitkan: 2023-07-31

Amazon Athena mengumumkan ketersediaan Athena SQL sebagai tambahan. Wilayah AWS

Rilis ini memperluas ketersediaan Athena SQL untuk mencakup Asia Pasifik (Hyderabad), Asia Pasifik (Melbourne), Eropa (Spanyol), dan Eropa (Zurich).

Untuk daftar lengkap yang Layanan AWS tersedia di masing-masing Wilayah AWS, lihat AWS Layanan menurut Wilayah.

Juli 27, 2023

Diterbitkan: 2023-07-27

Athena merilis BigQuery konektor Google versi 2023.30.1. Versi konektor ini mengurangi waktu eksekusi kueri dan menambahkan dukungan untuk kueri terhadap titik akhir BigQuery pribadi.

Untuk informasi tentang BigQuery konektor Google, lihatKonektor Google Amazon Athena BigQuery . Untuk informasi tentang memperbarui konektor sumber data yang ada, lihatMemperbarui konektor sumber data.

Juli 24, 2023

Diterbitkan: 2023-07-24

Athena mengumumkan perbaikan dan peningkatan berikut.

- Kueri dengan serikat pekerja - Meningkatkan kinerja kueri tertentu dengan serikat pekerja.
- Bergabung dengan perbandingan tipe - Memperbaiki potensi kegagalan kueri untuk `JOIN` pernyataan yang menyertakan perbandingan antara dua jenis yang berbeda.
- Subquery pada kolom bersarang - Memperbaiki masalah yang terkait dengan kegagalan kueri saat subquery berkorelasi pada kolom bersarang.
- Iceberg views - Memperbaiki masalah kompatibilitas dengan presisi kolom timestamp dalam tampilan Apache Iceberg. Tampilan gunung es yang memiliki kolom stempel waktu sekarang dapat dibaca terlepas dari apakah kolom dibuat pada mesin Athena versi 2 atau mesin Athena versi 3.

Juli 20, 2023

Diterbitkan: 2023-07-20

Athena merilis driver JDBC versi 2.1.0. Pengemudi menyertakan perangkat tambahan baru dan menyelesaikan masalah.

Penyempurnaan

Pustaka parser Jackson JSON berikut telah ditingkatkan:

- jackson-annotations 2.15.2 (sebelumnya 2.14.0)
- jackson-core 2.15.2 (sebelumnya 2.14.0)
- jackson-databind 2.15.2 (sebelumnya 2.14.0)

Masalah terselesaikan

- Memperbaiki masalah dengan meneruskan parameter array saat library sql2o digunakan.

Untuk informasi lebih lanjut, dan untuk mengunduh driver baru, catatan rilis, dan dokumentasi, lihatMenghubungkan ke Amazon Athena dengan JDBC.

13 Juli 2023

Diterbitkan: 2023-09-19

Athena mengumumkan fitur dan peningkatan berikut.

- JELASKAN ANALISIS — Menambahkan dukungan untuk antrian, analisis, perencanaan, dan waktu eksekusi ke output. `EXPLAIN ANALYZE`
- JELASKAN — `EXPLAIN` output sekarang menunjukkan statistik ketika kueri berisi agregasi.
- Parquet Hive SerDe — Menambahkan `parquet.ignore.statistics` properti untuk memungkinkan statistik pemrosesan diabaikan saat membaca data Parket. Untuk informasi, lihat Mengabaikan statistik Parket.

Untuk informasi selengkapnya tentang `EXPLAIN` dan `EXPLAIN ANALYZE`, lihat Menggunakan JELASKAN dan JELASKAN ANALISIS di Athena. Untuk informasi lebih lanjut tentang Parquet Hive SerDe, lihat. Parket SerDe

3 Juli 2023

Diterbitkan: 2023-07-25

Pada 3 Juli 2023, Athena mulai menyunting string kueri dari log. CloudTrail String query sekarang memiliki nilai`***OMITTED***`. Perubahan ini telah dilakukan untuk mencegah pengungkapan nama tabel atau nilai filter yang tidak diinginkan yang dapat mencakup informasi sensitif. Jika sebelumnya Anda mengandalkan CloudTrail log untuk mengakses string kueri lengkap, sebaiknya gunakan `Athena::GetQueryExecution` API dan meneruskan nilai `responseElements.queryExecutionId` dari log. CloudTrail Untuk informasi selengkapnya, lihat GetQueryExecutiontindakan di Referensi API Amazon Athena.

Juni 30, 2023

Diterbitkan: 2023-06-30

Editor kueri Athena sekarang mendukung saran kode typeahead untuk pengalaman penulisan kueri yang lebih cepat. Anda sekarang dapat menulis kueri SQL dengan akurasi yang ditingkatkan dan peningkatan efisiensi menggunakan fitur-fitur berikut:

- Saat Anda mengetik, saran muncul secara real time untuk kata kunci, variabel lokal, cuplikan, dan item katalog.
- Saat Anda mengetik nama database atau nama tabel diikuti dengan titik, editor dengan mudah menampilkan daftar tabel atau kolom untuk dipilih.
- Saat Anda mengarahkan kursor ke saran cuplikan, sinopsis menunjukkan gambaran singkat tentang sintaks dan penggunaan cuplikan.
- Untuk meningkatkan keterbacaan kode, kata kunci dan aturan penyorotan mereka juga telah diperbarui untuk menyelaraskan dengan sintaks terbaru Trino dan Hive.

Fitur ini diaktifkan secara default. Anda dapat mengaktifkan atau menonaktifkan fitur dalam pengaturan preferensi editor kode.

Untuk mencoba saran kode typeahead di editor kueri Athena, kunjungi konsol Athena di https://console.aws.amazon.com/athena/.

29 Juni 2023

Diterbitkan: 2023-06-29

- Athena mengumumkan driver ODBC versi 2.0.1.0. Untuk informasi selengkapnya, lihat catatan 2.0.1.0 rilis. Untuk mengunduh driver ODBC v2 baru, lihatUnduhan driver ODBC 2.x. Untuk informasi koneksi, lihatAmazon Athena ODBC 2.x.

- Athena dan fitur-fiturnya sekarang tersedia di Wilayah Timur Tengah (UEA). Untuk daftar lengkap yang Layanan AWS tersedia di masing-masing Wilayah AWS, lihat AWS Layanan menurut Wilayah.

28 Juni 2023

Diterbitkan: 2023-06-28

Anda sekarang dapat menggunakan Amazon Athena untuk menanyakan objek yang dipulihkan dari S3 Glacier Flexible Retrieval (sebelumnya Glacier) dan S3 Glacier Deep Archive kelas penyimpanan Amazon S3. Anda mengonfigurasi kemampuan ini berdasarkan per tabel. Fitur ini hanya didukung untuk tabel Apache Hive pada mesin Athena versi 3.

Untuk informasi selengkapnya, lihat Menanyakan objek Amazon S3 Glacier yang dipulihkan.

12 Juni 2023

Diterbitkan: 2023-06-12

Athena mengumumkan perbaikan dan peningkatan berikut.

- Stempel waktu Pembaca Parket - Menambahkan dukungan untuk membaca stempel waktu sebagai `bigint` (millis) untuk Pembaca Parket. Pembaruan ini memberikan paritas dengan dukungan di mesin Athena versi 2.
- JELASKAN ANALISIS - Menambahkan waktu baca input fisik ke statistik kueri dan output dari`EXPLAIN ANALYZE`. Untuk informasi tentang `EXPLAIN ANALYZE`, lihat Menggunakan JELASKAN dan JELASKAN ANALISIS di Athena.
- INSERT - Peningkatan kinerja kueri pada tabel yang ditulis dengan`INSERT`. Untuk informasi tentang `INSERT`, lihat INSERT INTO.
- Tabel Delta Lake — Memperbaiki masalah `DROP TABLE` pada tabel Delta Lake yang mencegahnya dihapus sepenuhnya saat mengalami modifikasi bersamaan.

8 Juni 2023

Diterbitkan: 2023-06-08

Amazon Athena untuk Apache Spark mengumumkan fitur-fitur baru berikut.

- Support untuk pustaka dan konfigurasi Java kustom - Anda sekarang dapat menggunakan paket Java Anda sendiri dan konfigurasi kustom untuk sesi Apache Spark Anda di Athena. Gunakan properti Spark untuk menentukan `.jar` file, paket, atau konfigurasi kustom lainnya dengan konsol Athena, API, AWS CLI atau Athena. Untuk informasi selengkapnya, lihat Menambahkan file JAR dan konfigurasi Spark khusus.
- Support untuk Apache Hudi, Apache Iceberg, dan Delta Lake tables - Athena for Spark sekarang mendukung format tabel penyimpanan data lake open-source Apache Iceberg, Apache Hudi, dan Linux Foundation Delta Lake. Untuk informasi selengkapnya, lihat Menggunakan format tabel non-Hive di Amazon Athena untuk Apache Spark dan topik individual untuk digunakanGunung Es Apache,Apache Hudi, dan Yayasan Linux Delta Lake tabel di Athena for Spark.
- Dukungan enkripsi untuk Apache Spark - Di Athena untuk Spark, Anda sekarang dapat mengaktifkan enkripsi pada data dalam perjalanan antara node Spark dan pada data lokal yang disimpan di disk oleh Spark. Untuk mengaktifkan enkripsi Spark, Anda dapat menggunakan konsol Athena, API Athena, AWS CLI atau Athena. Untuk informasi selengkapnya, lihat Mengaktifkan enkripsi Apache Spark.

Untuk informasi lebih lanjut tentang Amazon Athena untuk Apache Spark, lihat. Menggunakan Apache Spark di Amazon Athena

Juni 2, 2023

Diterbitkan: 2023-06-02

Anda sekarang dapat menghapus reservasi kapasitas di Athena dan AWS CloudFormation menggunakan templat untuk menentukan reservasi kapasitas Athena.

- Hapus reservasi kapasitas — Anda sekarang dapat menghapus reservasi kapasitas yang dibatalkan di Athena. Reservasi harus dibatalkan sebelum dapat dihapus. Menghapus reservasi kapasitas akan segera menghapus reservasi dari akun Anda. Reservasi yang dihapus tidak dapat lagi direferensikan, termasuk oleh ARN-nya. Untuk menghapus reservasi, Anda dapat menggunakan konsol Athena atau Athena API. Untuk informasi selengkapnya, lihat Menghapus reservasi kapasitas di Panduan Pengguna Amazon Athena dan DeleteCapacityReservationReferensi API Amazon Athena.
- Gunakan AWS CloudFormation templat untuk reservasi kapasitas — Anda sekarang dapat menggunakan AWS CloudFormation templat untuk menentukan reservasi kapasitas Athena menggunakan sumber daya. `AWS::Athena::CapacityReservation` Untuk informasi

selengkapnya, lihat AWS: :Athena:: CapacityReservation di AWS CloudFormation Panduan Pengguna.

Untuk informasi lebih lanjut tentang menggunakan reservasi kapasitas untuk menyediakan kapasitas Anda di Athena, lihat. Mengelola kapasitas pemrosesan kueri

25 Mei 2023

Diterbitkan: 2023-05-25

Athena telah merilis pembaruan konektor sumber data yang meningkatkan kinerja kueri federasi. Optimalisasi push-down baru dan penyaringan dinamis memungkinkan lebih banyak operasi dilakukan di database sumber daripada di Athena. Pengoptimalan ini mengurangi runtime kueri dan jumlah data yang dipindai. Perbaikan ini membutuhkan mesin Athena versi 3.

Konektor berikut telah diperbarui:

- Penyimpanan Danau Data Azure
- Sinaps Azure
- Sarang Cloudera
- Cloudera Impala
- Db2
- DynamoDB
- Google BigQuery
- Hortonworks
- MySQL
- Oracle
- PostgreSQL
- Redshift
- SAP HANA
- Kepingan salju
- SQL Server
- Teradata

Untuk informasi tentang memutakhirkan konektor sumber data Anda, lihatMemperbarui konektor sumber data.

18 Mei 2023

Diterbitkan: 2023-05-18

Anda sekarang dapat menggunakan AWS PrivateLink koneksi masuk IPv6 ke Amazon Athena.

Amazon Athena telah memperluas dukungannya untuk koneksi masuk melalui titik akhir Protokol Internet Versi 6 (IPv6) untuk disertakan. AWS PrivateLink Mulai hari ini, Anda dapat terhubung ke Athena dengan aman dan pribadi menggunakan AWS PrivateLink dari Amazon Virtual Private Cloud (Amazon VPC), selain titik akhir IPv6 publik yang sebelumnya tersedia.

Pertumbuhan Internet yang cepat melelahkan ketersediaan alamat Internet Protocol versi 4 (IPv4). IPv6 meningkatkan jumlah alamat yang tersedia beberapa kali sehingga Anda tidak lagi harus mengelola spasi alamat yang tumpang tindih di VPC Anda. Dengan rilis ini, Anda sekarang dapat menggabungkan manfaat pengalamatan IPv6 dengan keunggulan keamanan dan kinerja. AWS PrivateLink

Untuk menghubungkan secara terprogram ke AWS layanan, Anda dapat menggunakan AWS CLIatau AWS SDK untuk menentukan titik akhir. Untuk informasi selengkapnya tentang titik akhir layanan dan titik akhir layanan Athena, AWS lihat titik akhir layanan dan titik akhir Amazon Athena serta kuota di. Referensi Umum Amazon Web Services

15 Mei 2023

Diterbitkan: 2023-05-15

Athena mengumumkan rilis konektor Apache Spark DataSource V2 (DSV2) untuk DynamoDB, Logs, Metrics, dan CMDB. CloudWatch CloudWatch AWS Gunakan konektor DSV2 baru untuk menanyakan sumber data ini menggunakan Spark. Konektor DSV2 menggunakan parameter yang sama dengan konektor federasi Athena yang sesuai. Konektor DSV2 berjalan langsung pada pekerja Spark dan tidak mengharuskan Anda untuk menggunakan fungsi Lambda untuk menggunakannya.

Untuk informasi selengkapnya, lihat Konektor sumber data Athena untuk Apache Spark.

10 Mei 2023

Diterbitkan: 2023-05-10

Merilis driver ODBC 1.1.20 untuk Athena.

Fitur dan perangkat tambahan:

- Titik akhir Lake Formation mengesampingkan dukungan.
- Plugin otentikasi ADFS memiliki parameter baru untuk mengatur nilai Relying Party (). `LoginToRP`
- AWS pembaruan perpustakaan.

Perbaikan bug:

- Menyiapkan kegagalan deallocation pernyataan saat `SQLPrepare()` metode gagal dikirimkan.
- Kesalahan dalam mengikat parameter pernyataan yang disiapkan saat mengonversi tipe C ke tipe SQL.
- Kegagalan untuk mengembalikan data kapan `EXPLAIN` dan `EXPLAIN ANALYZE` kueri digunakan `SQLPrepare()` dan`SQLExecute()`.

Untuk informasi lebih lanjut, dan untuk mengunduh driver baru, catatan rilis, dan dokumentasi, lihatMenghubungkan ke Amazon Athena dengan ODBC.

8 Mei 2023

Diterbitkan: 2023-05-08

Athena mengumumkan perbaikan dan peningkatan berikut.

- Integrasi Hudi yang diperbarui - Athena telah memperbarui integrasinya dengan Apache Hudi. Anda sekarang dapat menggunakan Athena untuk menanyakan tabel Hudi 0.12.2, dan daftar metadata Hudi untuk tabel Hudi sekarang didukung. Untuk informasi selengkapnya, lihat Menggunakan Athena untuk menanyakan kumpulan data Apache Hudi dan Daftar metadata Hudi.
- Perbaikan konversi stempel waktu - Memperbaiki penanganan konversi stempel waktu ke tipe data presisi yang lebih rendah. Sebelumnya, mesin Athena versi 3 salah membulatkan nilai ke tipe target alih-alih memotongnya selama casting.

  Contoh berikut menggambarkan penanganan yang salah sebelum perbaikan.

  Contoh 1: Casting dari stempel waktu dalam mikrodetik ke milidetik

  Data sampel

```
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
```

Kueri berikut mencoba mengambil stempel waktu yang cocok dengan nilai tertentu.

```
SELECT *
FROM table
WHERE timestamps.col = timestamp'2020-06-10 15:55:23.383'
```

Kueri mengembalikan hasil sebagai berikut.

```
A, 2020-06-10 15:55:23.383
C, 2020-06-10 15:55:23.383
E, 2020-06-10 15:55:23.383
```

Sebelum perbaikan, Athena tidak memasukkan nilai `2020-06-10 15:55:23.383945` atau `2020-06-10 15:55:23.383945278` karena mereka dibulatkan ke. `2020-06-10 15:55:23.384`

Contoh 2: Casting dari stempel waktu hingga saat ini

Query berikut mengembalikan hasil yang salah.

```
SELECT date(timestamp '2020-12-31 23:59:59.999')
```

Hasil

`2021-01-01`

Sebelum perbaikan, Athena membulatkan nilainya, oleh karena itu bergerak maju. Nilai-nilai seperti itu sekarang terpotong daripada dibulatkan ke atas.

28 April 2023

Diterbitkan: 2023-04-28

Sekarang Anda dapat menggunakan reservasi kapasitas di Amazon Athena untuk menjalankan kueri SQL pada kapasitas komputasi yang dikelola sepenuhnya.

Kapasitas yang disediakan menyediakan kemampuan manajemen beban kerja yang membantu Anda memprioritaskan, mengontrol, dan menskalakan beban kerja interaktif Anda yang paling penting. Anda dapat menambahkan kapasitas kapan saja untuk meningkatkan jumlah kueri yang Anda jalankan secara bersamaan, mengontrol beban kerja mana yang menggunakan kapasitas, dan berbagi kapasitas antar beban kerja.

Untuk informasi selengkapnya, lihat Mengelola kapasitas pemrosesan kueri. Untuk informasi harga, kunjungi halaman harga Amazon Athena.

17 April 2023

Diterbitkan: 2023-04-17

Athena merilis driver JDBC versi 2.0.36. Driver menyertakan fitur-fitur baru dan menyelesaikan masalah.

Fitur baru

- Anda sekarang dapat menggunakan pengidentifikasi pihak mengandalkan yang dapat disesuaikan dengan otentikasi AD FS.
- Anda sekarang dapat menambahkan nama aplikasi yang menggunakan konektor ke string agen pengguna.

Masalah terselesaikan

- Memperbaiki kesalahan yang terjadi saat `getSchema()` digunakan untuk mengambil skema yang tidak ada.

Untuk informasi lebih lanjut, dan untuk mengunduh driver baru, catatan rilis, dan dokumentasi, lihatMenghubungkan ke Amazon Athena dengan JDBC.

April 14, 2023

Diterbitkan: 2023-06-20

Athena mengumumkan perbaikan dan peningkatan berikut.

- Saat Anda melemparkan string ke stempel waktu, ruang diperlukan antara hari dan waktu atau zona waktu. Untuk informasi selengkapnya, lihat Ruang yang diperlukan antara nilai tanggal dan waktu saat casting dari string ke stempel waktu.
- Menghapus perubahan yang melanggar cara penanganan presisi stempel waktu. Untuk menjaga konsistensi antara mesin Athena versi 2 dan mesin Athena versi 3, presisi stempel waktu sekarang defaultnya menjadi milidetik daripada mikrodetik.
- Athena sekarang secara konsisten memberlakukan akses untuk bucket keluaran kueri saat menjalankan kueri. Pastikan bahwa semua prinsip IAM yang menjalankan StartQueryExecutiontindakan memiliki GetBucketLocation izin S3: pada bucket keluaran kueri.

4 April 2023

Diterbitkan: 2023-04-04

Anda sekarang dapat menggunakan Amazon Athena untuk membuat dan menanyakan tampilan pada sumber data federasi. Gunakan satu tampilan federasi untuk menanyakan beberapa tabel eksternal atau himpunan bagian data. Ini menyederhanakan SQL yang diperlukan dan memberi Anda fleksibilitas untuk mengaburkan sumber data dari pengguna akhir yang harus menggunakan SQL untuk menanyakan data.

Untuk informasi selengkapnya, lihat Bekerja dengan pandangan dan Menjalankan kueri federasi.

30 Maret 2023

Diterbitkan: 2023-03-30

Amazon Athena mengumumkan ketersediaan Amazon Athena untuk Apache Spark sebagai tambahan. Wilayah AWS

Rilis ini memperluas ketersediaan Amazon Athena untuk Apache Spark termasuk Asia Pasifik (Mumbai), Asia Pasifik (Singapura), Asia Pasifik (Sydney), dan Eropa (Frankfurt).

Untuk informasi lebih lanjut tentang Amazon Athena untuk Apache Spark, lihat. Menggunakan Apache Spark di Amazon Athena

Maret 28, 2023

Diterbitkan: 2023-03-28

Athena mengumumkan perbaikan dan peningkatan berikut.

- Dalam tanggapan terhadap tindakan API `GetQueryExecution` dan `BatchGetQueryExecution` Athena, `subStatementType` bidang baru menampilkan jenis kueri yang dijalankan (misalnya,,,, `SELECT` `INSERT` `UNLOADCREATE_TABLE`, atau`CREATE_TABLE_AS_SELECT`).
- Memperbaiki bug di mana file manifes tidak dienkripsi dengan benar untuk operasi penulisan Apache Hive.
- Mesin Athena versi 3 sekarang benar menangani `NaN` dan `Infinity` nilai dalam fungsi. `approx_percentile` `approx_percentile`Fungsi mengembalikan perkiraan persentil untuk dataset pada persentase yang diberikan.

  Mesin Athena versi 2 salah memperlakukan `NaN` sebagai nilai yang lebih besar dari. `Infinity` Mesin Athena versi 3 sekarang menangani `NaN` dan `Infinity` sesuai dengan perlakuan nilai-nilai ini dalam fungsi analitik dan statistik lainnya. Poin-poin berikut menjelaskan perilaku baru secara lebih rinci.

  - Jika `NaN` ada dalam kumpulan data, `NaN` Athena kembali.
  - Jika tidak `NaN` ada, tetapi `Infinity` hadir, Athena memperlakukan `Infinity` sebagai jumlah yang sangat besar.
  - Jika ada beberapa `Infinity` nilai, Athena memperlakukannya sebagai jumlah yang sangat besar. Jika perlu, output Athena. `Infinity`
  - Jika satu kumpulan data memiliki keduanya - `Infinity` dan`-Double.MAX_VALUE`, dan hasil persentilnya, `-Double.MAX_VALUE` Athena kembali. `-Infinity`
  - Jika satu kumpulan data memiliki keduanya `Infinity` dan`Double.MAX_VALUE`, dan hasil persentilnya, `Double.MAX_VALUE` Athena kembali. `Infinity`
  - Untuk mengecualikan `Infinity` dan `NaN` dari perhitungan, gunakan `is_finite()` fungsi, seperti pada contoh berikut.

    ```
    approx_percentile(x, 0.5) FILTER (WHERE is_finite(x))
    ```

Maret 27, 2023

Diterbitkan: 2023-03-27

Anda sekarang dapat menentukan tingkat minimum enkripsi untuk workgroup Athena SQL di Amazon Athena. Fitur ini memastikan bahwa hasil dari semua kueri di workgroup Athena SQL dienkripsi pada

atau di atas tingkat enkripsi yang Anda tentukan. Anda dapat memilih di antara beberapa tingkat kekuatan enkripsi untuk melindungi data Anda. Untuk mengonfigurasi tingkat enkripsi minimum yang Anda inginkan, Anda dapat menggunakan konsol Athena, API AWS CLI, atau SDK.

Fitur enkripsi minimum tidak tersedia untuk grup kerja yang diaktifkan Apache Spark. Untuk informasi selengkapnya, lihat Mengkonfigurasi enkripsi minimum untuk grup kerja.

Maret 17, 2023

Diterbitkan: 2023-03-17

Athena mengumumkan perbaikan dan peningkatan berikut.

- Memperbaiki masalah dengan konektor DynamoDB Amazon Athena yang menyebabkan kueri gagal dengan pesan kesalahan hanya KeyConditionExpressions boleh berisi satu kondisi per kunci.

  Masalah ini terjadi karena mesin Athena versi 3 mengakui kesempatan untuk menekan lebih banyak jenis predikat daripada mesin Athena versi 2. Di mesin Athena versi 3, klausa seperti `some_column LIKE 'someprefix%` didorong ke bawah sebagai predikat filter yang menerapkan batas bawah dan atas pada kolom tertentu. Mesin Athena versi 2 tidak mendorong predikat ini ke bawah. Di mesin Athena versi 3, ketika `some_column` kolom kunci sortir, mesin mendorong predikat filter ke konektor DynamoDB. Predikat filter kemudian didorong lebih jauh ke layanan DynamoDB. Karena DynamoDB tidak mendukung lebih dari satu kondisi filter pada kunci pengurutan, DynamoDB mengembalikan kesalahan.

  Untuk memperbaiki masalah ini, perbarui konektor Amazon Athena DynamoDB Anda ke versi 2023.11.1. Untuk petunjuk tentang memperbarui konektor, lihatMemperbarui konektor sumber data.

8 Maret 2023

Diterbitkan: 2023-03-08

Athena mengumumkan perbaikan dan peningkatan berikut.

- Memperbaiki masalah dengan kueri gabungan yang menyebabkan nilai predikat stempel waktu dikirim sebagai mikrodetik, bukan milidetik.

15 Februari 2023

Diterbitkan: 2023-02-15

Athena mengumumkan perbaikan dan peningkatan berikut.

- Anda sekarang dapat menggunakan enkripsi sisi klien untuk mengenkripsi data di Amazon S3 untuk operasi penulisan Iceberg.
- Memperbaiki masalah yang memengaruhi enkripsi sisi server di Amazon S3 untuk operasi penulisan Iceberg.

31 Januari 2023

Diterbitkan: 2023-01-31

Anda sekarang dapat menggunakan Amazon Athena untuk menanyakan data di Google Cloud Storage. Seperti Amazon S3, Google Cloud Storage adalah layanan terkelola yang menyimpan data dalam bucket. Gunakan konektor Athena untuk Google Cloud Storage untuk menjalankan kueri federasi interaktif pada data eksternal Anda.

Untuk informasi selengkapnya, lihat Konektor Penyimpanan Awan Google Amazon Athena.

20 Januari 2023

Diterbitkan: 2023-01-20

Anda sekarang dapat melihat dokumentasi yang diperluas untuk dukungan kompresi Athena. Topik individu telah ditambahkan untukKompresi meja sarang,Kompresi tabel gunung es, danTingkat kompresi ZSTD.

Untuk informasi selengkapnya, lihat Dukungan kompresi Athena.

Januari 3, 2023

Diterbitkan: 2023-01-03

Athena mengumumkan pembaruan berikut:

- Perintah tambahan untuk metastor Hive — Anda dapat menggunakan Athena untuk terhubung ke Apache Hive Metastore yang dikelola sendiri sebagai katalog metadata dan data kueri yang disimpan di Amazon S3. Dengan rilis ini, Anda dapat menggunakan `CREATE TABLE`

`AS` (CTAS),`INSERT INTO`, dan 12 tambahan Data Definition Language (DDL) perintah untuk berinteraksi dengan Apache Hive Metastore. Anda dapat mengelola skema Hive Metastore langsung dari Athena menggunakan rangkaian kemampuan SQL yang diperluas ini.

Untuk informasi selengkapnya, lihat Menggunakan Athena Data Connector untuk Eksternal Hive Metastore.

- Driver JDBC versi 2.0.35 - Athena merilis driver JDBC versi 2.0.35. Driver JDBC 2.0.35 berisi pembaruan berikut:
  - Pengemudi sekarang menggunakan perpustakaan berikut untuk parser Jackson JSON.
    - jackson-annotations 2.14.0 (sebelumnya 2.13.2)
    - jackson-core 2.14.0 (sebelumnya 2.13.2)
    - jackson-databind 2.14.0 (sebelumnya 2.13.2.2)
  - Support untuk JDBC versi 4.1 telah dihentikan.

  Untuk informasi lebih lanjut, dan untuk mengunduh driver baru, catatan rilis, dan dokumentasi, lihatMenghubungkan ke Amazon Athena dengan JDBC.

Catatan rilis Athena untuk tahun 2022

14 Desember 2022

Diterbitkan: 2022-12-14

Anda sekarang dapat menggunakan konektor Amazon Athena untuk Kafka untuk menjalankan kueri SQL pada data streaming. Misalnya, Anda dapat menjalankan kueri analitik pada data streaming waktu nyata di Amazon Managed Streaming for Apache Kafka (Amazon MSK) dan bergabung dengan data historis di danau data Anda di Amazon S3.

Konektor Amazon Athena untuk Kafka mendukung kueri pada beberapa mesin streaming. Anda dapat menggunakan Athena untuk menjalankan kueri SQL di kluster Amazon MSK yang disediakan dan tanpa server, pada penerapan Kafka yang dikelola sendiri, dan pada streaming data di Confluent Cloud.

Untuk informasi selengkapnya, lihat Konektor MSK Amazon Athena.

Desember 2, 2022

Diterbitkan: 2022-12-02

Athena merilis driver JDBC versi 2.0.34. Driver JDBC 2.0.34 mencakup fitur-fitur baru berikut dan masalah yang diselesaikan:

- Dukungan penggunaan kembali hasil kueri — Anda sekarang dapat menggunakan kembali hasil kueri yang dieksekusi sebelumnya hingga batas waktu yang Anda tentukan alih-alih meminta Athena menghitung ulang hasil setiap kali kueri dijalankan. Untuk informasi selengkapnya, lihat Panduan Instalasi dan Konfigurasi, tersedia dari halaman unduhan JDBC, dan. Menggunakan kembali hasil kueri
- InstanceMetadata Dukungan Ec2 - Driver JDBC sekarang mendukung metode InstanceMetadata otentikasi Ec2 menggunakan profil instans IAM.
- Perbaikan pengecualian berbasis karakter - Memperbaiki pengecualian yang terjadi dengan kueri yang berisi karakter bahasa tertentu.
- Perbaikan kerentanan - Memperbaiki kerentanan yang terkait dengan AWS dependensi yang dikemas dengan konektor.

Untuk informasi lebih lanjut, dan untuk mengunduh driver baru, catatan rilis, dan dokumentasi, lihatMenghubungkan ke Amazon Athena dengan JDBC.

30 November 2022

Diterbitkan: 2022-11-30

Anda sekarang dapat secara interaktif membuat dan menjalankan aplikasi Apache Spark dan notebook yang kompatibel dengan Jupyter di Athena. Jalankan analitik data di Athena menggunakan Spark tanpa harus merencanakan, mengonfigurasi, atau mengelola sumber daya. Kirim kode Spark untuk diproses dan terima hasilnya secara langsung. Gunakan pengalaman notebook yang disederhanakan di konsol Amazon Athena untuk mengembangkan aplikasi Apache Spark menggunakan Python atau. API notebook Athena

Apache Spark di Amazon Athena tanpa server dan menyediakan penskalaan otomatis sesuai permintaan yang memberikan komputasi instan untuk memenuhi perubahan volume data dan persyaratan pemrosesan.

Untuk informasi selengkapnya, lihat Menggunakan Apache Spark di Amazon Athena.

18 November 2022

Diterbitkan: 2022-11-18

Anda sekarang dapat menggunakan konektor Amazon Athena untuk IBM Db2 untuk meminta Db2 dari Athena. Misalnya, Anda dapat menjalankan kueri analitik melalui gudang data di Db2 dan data lake di Amazon S3.

Konektor Amazon Athena Db2 memperlihatkan beberapa opsi konfigurasi melalui variabel lingkungan Lambda. Untuk informasi tentang opsi konfigurasi, parameter, string koneksi, penerapan, dan batasan, lihat. Konektor Amazon Athena IBM Db2

17 November 2022

Diterbitkan: 2022-11-17

Dukungan Apache Iceberg di mesin Athena versi 3 sekarang menawarkan fitur transaksi ACID yang disempurnakan berikut:

- Dukungan ORC dan Avro - Buat tabel Iceberg menggunakan baris Apache Avro dan Apache ORC dan format file berbasis kolom. Support untuk format ini adalah tambahan dari dukungan yang ada untuk Parket.
- MERGE INTO — Gunakan `MERGE INTO` perintah untuk menggabungkan data pada skala efisien. `MERGE INTO`menggabungkan`INSERT`,`UPDATE`, dan `DELETE` operasi menjadi satu transaksi. Ini mengurangi overhead pemrosesan dalam pipeline data Anda dan membutuhkan lebih sedikit SQL untuk menulis. Untuk informasi selengkapnya, lihat Memperbarui data tabel Gunung Es dan BERGABUNG MENJADI.
- Dukungan CTAS dan VIEW - Gunakan `CREATE TABLE AS SELECT` (CTAS) dan `CREATE VIEW` pernyataan dengan tabel Iceberg. Untuk informasi selengkapnya, lihat CREATE TABLE AS dan CREATE VIEW.
- Dukungan VACUUM - Anda dapat menggunakan `VACUUM` pernyataan untuk mengoptimalkan data lake Anda dengan menghapus snapshot dan data yang tidak lagi diperlukan. Anda dapat menggunakan fitur ini untuk meningkatkan kinerja baca dan memenuhi persyaratan peraturan seperti GDPR. Untuk informasi selengkapnya, lihat Mengoptimalkan tabel Iceberg dan VAKUM.

Fitur-fitur baru ini memerlukan mesin Athena versi 3 dan tersedia di semua Wilayah di mana Athena didukung. Anda dapat menggunakannya dengan konsol Athena, driver, atau API.

Untuk informasi tentang penggunaan Gunung Es di Athena, lihat. Menggunakan tabel Apache Iceberg

November 14, 2022

Diterbitkan: 2022-11-14

Amazon Athena sekarang mendukung titik akhir IPv6 untuk koneksi masuk yang dapat Anda gunakan untuk menjalankan fungsi Athena melalui IPv6. Anda dapat menggunakan fitur ini untuk memenuhi persyaratan kepatuhan IPv6. Ini juga menghilangkan kebutuhan akan peralatan jaringan tambahan untuk menangani terjemahan alamat antara IPv4 dan IPv6.

Untuk menggunakan fitur ini, konfigurasikan aplikasi Anda untuk menggunakan endpoint dual-stack Athena baru, yang mendukung IPv4 dan IPv6. Titik akhir dual-stack menggunakan format. `athena.region.api.aws` Misalnya, titik akhir dual-stack di Wilayah AS Timur (Virginia N.) adalah. `athena.us-east-1.api.aws`

Saat Anda membuat permintaan ke titik akhir Athena dual-stack, titik akhir akan menyelesaikan ke IPv6 atau alamat IPv4 tergantung pada protokol yang digunakan oleh jaringan dan klien Anda. Untuk menghubungkan secara terprogram ke AWS layanan, Anda dapat menggunakan AWS CLIatau AWS SDK untuk menentukan titik akhir.

Untuk informasi selengkapnya tentang titik akhir layanan, lihat titik akhir AWS layanan. Untuk mempelajari lebih lanjut tentang titik akhir layanan Athena, lihat titik akhir dan kuota Amazon Athena dalam dokumentasi. AWS

Anda dapat menggunakan titik akhir dual-stack Athena baru untuk koneksi masuk tanpa biaya tambahan. Titik akhir dual-stack umumnya tersedia di semua. Wilayah AWS

11 November 2022

Diterbitkan: 2022-11-11

Athena mengumumkan perbaikan dan peningkatan berikut.

- Kontrol akses berbutir halus Formasi Danau yang Diperluas — Anda sekarang dapat menggunakan kebijakan kontrol AWS Lake Formationakses berbutir halus dalam kueri Athena untuk data yang disimpan dalam format file atau tabel yang didukung. Anda dapat menggunakan kontrol akses berbutir halus di Lake Formation untuk membatasi akses ke data dalam hasil kueri menggunakan filter data untuk mencapai tingkat kolom, tingkat baris, dan keamanan tingkat sel. Format tabel yang didukung di Athena termasuk Apache Iceberg, Apache Hudi, dan Apache Hive. Kontrol akses berbutir halus yang diperluas tersedia di semua wilayah yang didukung oleh Athena. Dukungan tabel dan format file yang diperluas memerlukanMesin Athena versi 3, yang

menawarkan fitur baru dan peningkatan kinerja kueri, tetapi tidak mengubah cara Anda mengatur kebijakan kontrol akses berbutir halus di Lake Formation.

Penggunaan kontrol akses berbutir halus yang diperluas ini di Athena memiliki pertimbangan sebagai berikut:

- JELASKAN — Informasi penyaringan baris atau sel yang didefinisikan dalam Lake Formation dan informasi statistik kueri tidak ditampilkan dalam output `EXPLAIN` dan`EXPLAIN ANALYZE`. Untuk informasi tentang `EXPLAIN` di Athena, lihat. Menggunakan JELASKAN dan JELASKAN ANALISIS di Athena
- Metastor External Hive - Kolom tersembunyi Apache Hive tidak dapat digunakan untuk pemfilteran kontrol akses berbutir halus, dan tabel sistem tersembunyi Apache Hive tidak didukung oleh kontrol akses berbutir halus. Untuk informasi selengkapnya, lihat Pertimbangan dan batasan dalam topik Menggunakan Athena Data Connector untuk Eksternal Hive Metastore.
- Statistik kueri — Jumlah baris input dan output tingkat tahap dan informasi ukuran data tidak ditampilkan dalam statistik kueri Athena ketika kueri memiliki filter tingkat baris yang ditentukan dalam Lake Formation. Untuk informasi tentang melihat statistik untuk kueri Athena, lihat dan. Melihat statistik dan detail eksekusi untuk kueri yang diselesaikan GetQueryRuntimeStatistics
- Workgroup — Pengguna di workgroup Athena yang sama dapat melihat data yang telah dikonfigurasi oleh kontrol akses berbutir halus Lake Formation agar dapat diakses oleh workgroup. Untuk informasi tentang menggunakan Athena untuk menanyakan data yang terdaftar di Lake Formation, lihat. Menggunakan Athena untuk menanyakan data yang terdaftar AWS Lake Formation

Untuk informasi tentang penggunaan kontrol akses berbutir halus di Lake Formation, lihat Mengelola kontrol akses berbutir halus menggunakan AWS Lake Formation di Blog Big Data.AWS

- Athena Federated Query — Athena Federated Query sekarang mempertahankan casing asli nama bidang dalam objek. `struct` Sebelumnya, nama `struct` bidang secara otomatis dibuat huruf kecil.

8 November 2022

Diterbitkan: 2022-11-08

Anda sekarang dapat menggunakan hasil kueri menggunakan kembali fitur caching untuk mempercepat kueri berulang di Athena. Kueri berulang adalah kueri SQL yang identik dengan yang dikirimkan baru-baru ini yang menghasilkan hasil yang sama. Saat Anda perlu menjalankan beberapa

kueri yang identik, caching penggunaan kembali hasil dapat mengurangi waktu yang diperlukan untuk menghasilkan hasil. Hasil penggunaan kembali caching juga menurunkan biaya dengan mengurangi jumlah byte yang dipindai.

Untuk informasi selengkapnya, lihat Menggunakan kembali hasil kueri.

13 Oktober 2022

Diterbitkan: 2022-10-13

Athena mengumumkan mesin Athena versi 3.

Athena telah meningkatkan mesin kueri SQL untuk menyertakan fitur terbaru dari proyek open source Trino. Selain mendukung semua fitur mesin Athena versi 2, Athena engine versi 3 mencakup lebih dari 50 fungsi SQL baru, 30 fitur baru, dan lebih dari 90 peningkatan kinerja kueri. Dengan peluncuran hari ini, Athena juga memperkenalkan pendekatan integrasi berkelanjutan untuk manajemen perangkat lunak open source yang meningkatkan mata uang dengan proyek Trino dan Presto sehingga Anda mendapatkan akses lebih cepat ke peningkatan komunitas, terintegrasi dan disetel dalam mesin Athena.

Untuk informasi selengkapnya, lihat Mesin Athena versi 3.

10 Oktober 2022

Diterbitkan: 2022-10-10

Athena merilis driver JDBC versi 2.0.33. Driver JDBC 2.0.33 mencakup perubahan berikut:

- Versi driver baru, versi JDBC, dan properti nama plugin ditambahkan ke string user-agent di kelas penyedia kredensyal.
- Pesan kesalahan diperbaiki dan informasi yang diperlukan ditambahkan.
- Pernyataan yang disiapkan sekarang ditangani jika koneksi ditutup atau eksekusi pernyataan yang disiapkan Athena gagal.

Untuk informasi lebih lanjut, dan untuk mengunduh driver baru, catatan rilis, dan dokumentasi, lihatMenghubungkan ke Amazon Athena dengan JDBC.

September 23, 2022

Diterbitkan: 2022-09-26

Konektor Amazon Athena Neptunus sekarang mendukung pencocokan case insensitive pada nama kolom dan tabel.

- Konektor sumber data Neptunus dapat menyelesaikan nama kolom pada tabel Neptunus yang menggunakan casing bahkan jika nama kolom semuanya lebih rendah dalam tabel di. AWS Glue Untuk mengaktifkan perilaku ini, atur variabel `enable_caseinsensitivematch` lingkungan ke `true` fungsi Lambda konektor Neptunus.
- Karena hanya AWS Glue mendukung nama tabel huruf kecil, saat Anda membuat AWS Glue tabel untuk Neptunus, tentukan AWS Glue parameter tabel. `"glabel" = table_name`

Untuk informasi lebih lanjut tentang konektor Neptunus, lihat. Konektor Amazon Athena Neptunus

13 September 2022

Diterbitkan: 2022-09-13

Athena mengumumkan perbaikan dan peningkatan berikut.

- Metastore External Hive — Athena sekarang kembali `NULL` alih-alih melempar pengecualian ketika `WHERE` klausa menyertakan partisi yang tidak ada di metastore Hive eksternal (EHMS). Perilaku baru cocok dengan perilaku AWS Glue Data Catalog.
- Kueri berparameter - Nilai dalam kueri berparameter sekarang dapat dilemparkan ke tipe data. `DOUBLE`
- Apache Iceberg - Operasi tulis ke tabel Iceberg sekarang berhasil ketika Object Lock diaktifkan pada bucket Amazon S3.

31 Agustus 2022

Diterbitkan: 2022-08-31

Amazon Athena mengumumkan ketersediaan Athena dan fitur-fiturnya di Wilayah Asia Pasifik (Jakarta).

Rilis ini memperluas ketersediaan Athena di Asia Pasifik termasuk Asia Pasifik (Hong Kong), Asia Pasifik (Jakarta), Asia Pasifik (Mumbai), Asia Pasifik (Osaka), Asia Pasifik (Seoul), Asia Pasifik (Singapura), Asia Pasifik (Sydney), dan Asia Pasifik (Tokyo). Untuk daftar lengkap yang Layanan AWS tersedia di Wilayah ini dan lainnya, lihat Daftar Layanan Wilayah AWS al.

23 Agustus 2022

Diterbitkan: 2022-08-23

Rilis v2022.32.1 dari Athena Query Federation SDK mencakup perubahan berikut:

- Menambahkan dukungan ke konektor sumber data Amazon Athena Oracle untuk koneksi berbasis SSL ke instans Amazon RDS. Support terbatas pada protokol Transport Layer Security (TLS) dan otentikasi server oleh klien. Karena otentikasi timbal balik itu tidak didukung di Amazon RDS, pembaruan tidak termasuk dukungan untuk otentikasi timbal balik.

  Untuk informasi selengkapnya, lihat Konektor Amazon Athena Oracle.

3 Agustus 2022

Diterbitkan: 2022-08-03

Athena merilis driver JDBC versi 2.0.32. Driver JDBC 2.0.32 mencakup perubahan berikut:

- `User-Agent`String yang dikirim ke Athena SDK telah diperluas untuk berisi versi driver, versi spesifikasi JDBC, dan nama plugin otentikasi.
- Memperbaiki `NullPointerException` yang dilemparkan ketika tidak ada nilai yang diberikan untuk `CheckNonProxyHost` parameter.
- Memperbaiki masalah dengan `login_url` parsing di plugin BrowserSaml otentikasi.
- Memperbaiki masalah host proxy yang terjadi saat `UseProxyforIdp` parameter disetel ke`true`.

Untuk informasi lebih lanjut, dan untuk mengunduh driver baru, catatan rilis, dan dokumentasi, lihatMenghubungkan ke Amazon Athena dengan JDBC.

1 Agustus 2022

Diterbitkan: 2022-08-01

Athena mengumumkan peningkatan pada konektor sumber data bawaan Athena Query Federation SDK dan Athena. Perbaikannya meliputi:

- Penguraian struct - Memperbaiki masalah `GlueFieldLexer` parsing di Athena Query Federation SDK yang mencegah struct rumit tertentu menampilkan semua data mereka. Masalah ini memengaruhi konektor yang dibangun di Athena Query Federation SDK.

- AWS Glue tabel - Ditambahkan dukungan tambahan untuk `set` dan jenis `decimal` kolom dalam AWS Glue tabel.
- DynamoDB konektor - Ditambahkan kemampuan untuk mengabaikan casing pada nama atribut DynamoDB. Untuk informasi lebih lanjut, lihat `disable_projection_and_casing` di Parameter bagian Konektor Amazon Athena DynamoDB halaman.

Untuk informasi lebih lanjut, lihat Rilis v2022.30.2 dari Athena Query Federation di. GitHub

21 Juli 2022

Diterbitkan: 2022-07-21

Sekarang Anda dapat menganalisis dan men-debug kueri Anda menggunakan metrik kinerja dan alat analisis kueri visual interaktif di konsol Athena. Data kinerja kueri dan detail eksekusi dapat membantu Anda mengidentifikasi hambatan dalam kueri, memeriksa operator dan statistik untuk setiap tahap kueri, melacak volume data yang mengalir di antara tahapan, dan memvalidasi dampak predikat kueri. Anda sekarang dapat:

- Akses rencana eksekusi terdistribusi dan logis untuk kueri Anda dalam satu klik.
- Jelajahi operasi di setiap tahap sebelum tahap dijalankan.
- Visualisasikan kinerja kueri yang diselesaikan dengan metrik untuk waktu yang dihabiskan dalam tahap antrian, perencanaan, dan eksekusi.
- Dapatkan informasi tentang jumlah baris dan jumlah data sumber yang diproses dan output oleh kueri Anda.
- Lihat detail eksekusi granular untuk kueri Anda yang disajikan dalam konteks dan diformat sebagai grafik interaktif.
- Gunakan detail eksekusi tingkat tahap yang tepat untuk memahami aliran data melalui kueri Anda.
- Analisis data kinerja kueri secara terprogram menggunakan API baru untuk mendapatkan statistik runtime kueri, juga dirilis hari ini.

Untuk mempelajari cara menggunakan kemampuan ini pada kueri Anda, tonton tutorial video Optimalkan Kueri Amazon Athena dengan Alat Analisis Kueri Baru di saluran. AWS YouTube

Untuk dokumentasi, lihat Melihat rencana eksekusi untuk kueri SQL danMelihat statistik dan detail eksekusi untuk kueri yang diselesaikan.

Juli 11, 2022

Diterbitkan: 2022-07-11

Sekarang Anda dapat menjalankan kueri berparameter langsung dari konsol Athena atau API tanpa menyiapkan pernyataan SQL terlebih dahulu.

Saat Anda menjalankan kueri di konsol Athena yang memiliki parameter dalam bentuk tanda tanya, antarmuka pengguna sekarang meminta Anda untuk memasukkan nilai untuk parameter secara langsung. Ini menghilangkan kebutuhan untuk memodifikasi nilai literal di editor kueri setiap kali Anda ingin menjalankan kueri.

Jika Anda menggunakan API eksekusi kueri yang disempurnakan, Anda sekarang dapat memberikan parameter eksekusi dan nilainya dalam satu panggilan.

Untuk informasi selengkapnya, lihat Menggunakan kueri berparameter di panduan pengguna ini dan posting Blog AWS Big Data Menggunakan kueri berparameter Amazon Athena untuk menyediakan data sebagai layanan.

8 Juli 2022

Diterbitkan: 2022-07-08

Athena mengumumkan perbaikan dan peningkatan berikut.

- Memperbaiki masalah dengan penanganan konversi `DATE` kolom untuk SageMaker titik akhir (UDF) yang menyebabkan kegagalan kueri.

6 Juni 2022

Diterbitkan: 2022-06-06

Athena merilis driver JDBC versi 2.0.31. Driver JDBC 2.0.31 mencakup perubahan berikut:

- masalah ketergantungan log4j - Menyelesaikan pesan kesalahan kelas driver Cannot find yang disebabkan oleh ketergantungan log4j.

Untuk informasi lebih lanjut, dan untuk mengunduh driver baru, catatan rilis, dan dokumentasi, lihatMenghubungkan ke Amazon Athena dengan JDBC.

25 Mei 2022

Diterbitkan: 2022-05-25

Athena mengumumkan perbaikan dan peningkatan berikut.

- Dukungan gunung es
  - Memperkenalkan dukungan untuk kueri lintas wilayah. Sekarang Anda dapat menanyakan tabel Iceberg dalam Wilayah AWS yang berbeda dari Wilayah AWS yang Anda gunakan. Permintaan lintas wilayah tidak didukung di Wilayah China.
  - Memperkenalkan dukungan untuk konfigurasi enkripsi sisi server. Sekarang Anda dapat menggunakan SSE-S3/SSE-KMS untuk mengenkripsi data dari operasi penulisan Iceberg di Amazon S3.

  Untuk informasi lebih lanjut tentang menggunakan Apache Iceberg di Athena, lihat. Menggunakan tabel Apache Iceberg
- Rilis driver JDBC 2.0.30

  Driver JDBC 2.0.30 untuk Athena memiliki peningkatan berikut:
  - Memperbaiki masalah ras data yang memengaruhi pernyataan disiapkan berparameter.
  - Memperbaiki masalah start up aplikasi yang terjadi di lingkungan build Gradle.

  Untuk mengunduh driver JDBC 2.0.30, catatan rilis, dan dokumentasi, lihat. Menghubungkan ke Amazon Athena dengan JDBC

6 Mei 2022

Diterbitkan: 2022-05-06

Merilis driver JDBC 2.0.29 dan ODBC 1.1.17 untuk Athena.

Driver ini mencakup perubahan berikut:

- Memperbarui proses peluncuran browser plugin SAMP.

Untuk informasi lebih lanjut tentang perubahan ini, dan untuk mengunduh driver baru, catatan rilis, dan dokumentasi, lihat Menghubungkan ke Amazon Athena dengan JDBC danMenghubungkan ke Amazon Athena dengan ODBC.

22 April 2022

Diterbitkan: 2022-04-22

Athena mengumumkan perbaikan dan peningkatan berikut.

- Memperbaiki masalah pada indeks partisi dan fitur penyaringan dengan cache partisi yang terjadi ketika kondisi berikut terpenuhi:
  - `partition_filtering.enabled`Kunci diatur ke `true` dalam properti AWS Glue tabel untuk tabel.
  - Tabel yang sama digunakan beberapa kali dengan nilai filter partisi yang berbeda.

21 April 2022

Diterbitkan: 2022-04-21

Anda sekarang dapat menggunakan Amazon Athena untuk menjalankan kueri federasi pada sumber data baru, termasuk Google, Azure Synapse, dan BigQuery Snowflake. Konektor sumber data baru meliputi:

- Penyimpanan Danau Data Azure (ADLS) Gen2
- Sinaps Azure
- Sarang Cloudera
- Cloudera Impala
- Google BigQuery
- Hortonworks
- Microsoft SQL Server
- Oracle
- SAP HANA (Edisi Ekspres)
- Kepingan salju
- Teradata

Untuk daftar lengkap sumber data yang didukung oleh Athena, lihat. Konektor sumber data yang tersedia

Untuk mempermudah penelusuran sumber yang tersedia dan terhubung ke data Anda, Anda sekarang dapat mencari, mengurutkan, dan memfilter konektor yang tersedia dari layar Sumber Data yang diperbarui di konsol Athena.

Untuk mempelajari tentang menanyakan sumber federasi, lihat Menggunakan Amazon Athena dan. Menjalankan kueri federasi

13 April 2022

Diterbitkan: 2022-04-13

Athena merilis driver JDBC versi 2.0.28. Driver JDBC 2.0.28 mencakup perubahan berikut:

- Dukungan JWT - Driver sekarang mendukung token web JSON (JWT) untuk otentikasi. Untuk informasi tentang menggunakan JWT dengan driver JDBC, lihat panduan instalasi dan konfigurasi, yang dapat diunduh dari halaman driver JDBC.
- Pustaka Log4j yang diperbarui — Driver JDBC sekarang menggunakan pustaka Log4j berikut:
  - Log4j-API 2.17.1 (sebelumnya 2.17.0)
  - Log4J-Core 2.17.1 (sebelumnya 2.17.0)
  - Log4J-JCL 2.17.2
- Perbaikan lainnya - Driver baru juga mencakup perbaikan dan perbaikan bug berikut:
  - Fitur pernyataan yang disiapkan Athena sekarang tersedia melalui JDBC. Untuk informasi tentang pernyataan yang disiapkan, lihatMenggunakan kueri berparameter.
  - Athena JDBC Federasi SAMP sekarang berfungsi untuk Wilayah China.
  - Perbaikan kecil tambahan.

Untuk informasi lebih lanjut, dan untuk mengunduh driver baru, catatan rilis, dan dokumentasi, lihatMenghubungkan ke Amazon Athena dengan JDBC.

Maret 30, 2022

Diterbitkan: 2022-03-30

Athena mengumumkan perbaikan dan peningkatan berikut.

- Kueri lintas wilayah — Anda sekarang dapat menggunakan Athena untuk menanyakan data yang terletak di bucket Amazon S3 Wilayah AWS termasuk Asia Pasifik (Hong Kong), Timur Tengah

(Bahrain), Afrika (Cape Town), dan Eropa (Milan). Permintaan lintas wilayah tidak didukung di Wilayah China.

- Untuk daftar Wilayah AWS di mana Athena tersedia, lihat titik akhir dan kuota Amazon Athena.
- Untuk informasi tentang mengaktifkan Wilayah AWS yang dinonaktifkan secara default, lihat Mengaktifkan Wilayah.
- Untuk informasi tentang kueri di seluruh Wilayah, lihatMenanyakan lintas wilayah.

18 Maret 2022

Diterbitkan: 2022-03-18

Athena mengumumkan perbaikan dan peningkatan berikut.

- Penyaringan dinamis - Pemfilteran dinamis telah ditingkatkan untuk kolom integer dengan menerapkan filter secara efisien ke setiap catatan tabel yang sesuai.
- Iceberg - Memperbaiki masalah yang menyebabkan kegagalan saat menulis file Iceberg Parquet yang lebih besar dari 2GB.
- Output tidak terkompresi — CREATE TABLE pernyataan sekarang mendukung penulisan file yang tidak terkompresi. Untuk menulis file yang tidak terkompresi, gunakan sintaks berikut:
  - BUAT TABEL (file teks atau JSON) - Dalam`TBLPROPERTIES`, tentukan`write.compression = NONE`.
  - BUAT TABEL (Parket) — Dalam`TBLPROPERTIES`, tentukan`parquet.compression = UNCOMPRESSED`.
  - CREATE TABLE (ORC) - Dalam`TBLPROPERTIES`, tentukan`orc.compress = NONE`.
- Kompresi - Memperbaiki masalah dengan sisipan untuk tabel file teks yang membuat file dikompresi dalam satu format tetapi menggunakan ekstensi file format kompresi lain ketika metode kompresi non-default digunakan.
- Avro - Memperbaiki masalah yang terjadi saat membaca desimal dari jenis tetap dari file Avro.

2 Maret 2022

Diterbitkan: 2022-03-02

Athena mengumumkan fitur dan peningkatan berikut.

- Anda sekarang dapat memberikan akses kontrol penuh kepada pemilik bucket Amazon S3 atas hasil kueri saat ACL diaktifkan untuk bucket hasil kueri. Untuk informasi selengkapnya, lihat Menentukan lokasi hasil query.
- Anda sekarang dapat memperbarui kueri bernama yang ada. Untuk informasi selengkapnya, lihat Menggunakan kueri yang disimpan.

Februari 23, 2022

Diterbitkan: 2022-02-23

Athena mengumumkan perbaikan dan peningkatan kinerja berikut.

- Peningkatan penanganan memori untuk meningkatkan kinerja dan mengurangi kesalahan memori.
- Athena sekarang membaca kolom stempel waktu ORC dengan informasi zona waktu yang disimpan di footer garis dan menulis file ORC dengan zona waktu (UTC) di footer. Ini hanya memengaruhi perilaku pembacaan stempel waktu ORC jika file ORC yang akan dibaca dibuat di lingkungan zona waktu non-UTC.
- Memperbaiki perkiraan ukuran tabel symlink yang salah yang menghasilkan rencana kueri suboptimal.
- Tampilan lateral yang meledak sekarang dapat ditanyakan di konsol Athena dari sumber data metastore Hive.
- Peningkatan Amazon S3 membaca pesan kesalahan untuk menyertakan informasi kode kesalahan Amazon S3 yang lebih rinci.
- Memperbaiki masalah yang menyebabkan file keluaran dalam format ORC menjadi tidak kompatibel dengan Apache Hive 3.1.
- Memperbaiki masalah yang menyebabkan nama tabel dengan tanda kutip gagal dalam kueri DHTML dan DDL tertentu.

Februari 15, 2022

Diterbitkan: 2022-02-15

Amazon Athena telah meningkatkan kuota kueri DML aktif di semua Wilayah. AWS Kueri aktif mencakup kueri berjalan dan antrian. Dengan perubahan ini, Anda sekarang dapat memiliki lebih banyak kueri DHTML dalam keadaan aktif daripada sebelumnya.

Untuk informasi tentang kuota layanan Athena, lihat. Service Quotas Untuk kuota kueri di Wilayah tempat Anda menggunakan Athena, lihat titik akhir dan kuota Amazon Athena di. Referensi Umum AWS

Untuk memantau penggunaan kuota, Anda dapat menggunakan metrik CloudWatch penggunaan. Athena menerbitkan `ActiveQueryCount` metrik di namespace. `AWS/Usage` Untuk informasi selengkapnya, lihat Memantau metrik penggunaan Athena.

Setelah meninjau penggunaan Anda, Anda dapat menggunakan konsol Service Quotas untuk meminta peningkatan kuota. Jika sebelumnya Anda meminta kenaikan kuota untuk akun Anda, kuota yang Anda minta tetap berlaku jika melebihi kuota kuota kueri DML aktif default yang baru. Jika tidak, semua akun menggunakan default baru.

14 Februari 2022

Diterbitkan: 2022-02-14

Rilis ini menambahkan `ErrorType` subbidang ke objek AthenaErrorrespons dalam aksi GetQueryExecutionAthena API.

Sementara `ErrorCategory` bidang yang ada menunjukkan sumber umum dari kueri yang gagal (sistem, pengguna, atau lainnya), `ErrorType` bidang baru memberikan informasi yang lebih terperinci tentang kesalahan yang terjadi. Gabungkan informasi dari kedua bidang untuk mendapatkan wawasan tentang penyebab kegagalan kueri.

Untuk informasi selengkapnya, lihat Katalog kesalahan Athena.

Februari 9, 2022

Diterbitkan: 2022-02-09

Konsol Athena lama tidak lagi tersedia. Konsol baru Athena mendukung semua fitur konsol sebelumnya, tetapi dengan antarmuka modern yang lebih mudah digunakan dan menyertakan fitur baru yang meningkatkan pengalaman mengembangkan kueri, menganalisis data, dan mengelola penggunaan Anda. Untuk menggunakan konsol Athena baru, kunjungi https://console.aws.amazon.com/athena/.

8 Februari 2022

Diterbitkan: 2022-02-08

Pemilik bucket yang diharapkan — Sebagai langkah pengamanan tambahan, Anda sekarang dapat secara opsional menentukan Akun AWS ID yang Anda harapkan sebagai pemilik bucket lokasi keluaran hasil kueri di Athena. Jika ID akun pemilik bucket hasil kueri tidak cocok dengan ID akun yang Anda tentukan, upaya untuk menampilkan ke bucket akan gagal dengan kesalahan izin Amazon S3. Anda dapat membuat pengaturan ini di tingkat klien atau kelompok kerja.

Untuk informasi selengkapnya, lihat Menentukan lokasi hasil query.

28 Januari 2022

Diterbitkan: 2022-01-28

Athena mengumumkan peningkatan fitur mesin berikut.

- Apache Hudi — Kueri snapshot pada tabel Hudi Merge on Read (MoR) sekarang dapat membaca kolom stempel waktu yang memiliki tipe data. `INT64`
- Kueri UNION — Peningkatan kinerja dan pengurangan pemindaian data untuk `UNION` kueri tertentu yang memindai tabel yang sama beberapa kali.
- Kueri terputus - Peningkatan kinerja untuk kueri yang hanya memiliki nilai terpisah untuk setiap kolom partisi pada filter.
- Peningkatan proyeksi partisi
  - Beberapa nilai disjunct sekarang diizinkan pada kondisi filter untuk kolom tipe. `injected` Untuk informasi selengkapnya, lihat Jenis yang disuntikkan.
  - Peningkatan kinerja untuk kolom tipe berbasis string seperti `CHAR` atau `VARCHAR` yang hanya memiliki nilai terpisah pada filter.

Januari 13, 2022

Diterbitkan: 2022-01-13

Merilis driver JDBC 2.0.27 dan ODBC 1.1.15 untuk Athena.

Driver JDBC 2.0.27 mencakup perubahan berikut:

- Pengemudi telah diperbarui untuk mengambil katalog eksternal.
- Nomor versi driver yang diperluas sekarang disertakan dalam `user-agent` string sebagai bagian dari panggilan API Athena.

Driver ODBC 1.1.15 mencakup perubahan berikut:

- Memperbaiki masalah dengan panggilan kedua ke`SQLParamData()`.

Untuk informasi lebih lanjut tentang perubahan ini, dan untuk mengunduh driver baru, catatan rilis, dan dokumentasi, lihat Menghubungkan ke Amazon Athena dengan JDBC danMenghubungkan ke Amazon Athena dengan ODBC.

Catatan rilis Athena untuk tahun 2021

26 November 2021

Diterbitkan: 2021-11-26

Athena mengumumkan pratinjau publik transaksi ACID Athena, yang menambahkan operasi tulis, hapus, pembaruan, dan perjalanan waktu ke bahasa manipulasi data SQL (DML/bahasa manipulasi data SQL Athena). Transaksi Athena ACID memungkinkan beberapa pengguna bersamaan untuk membuat modifikasi tingkat baris yang andal pada data Amazon S3. Dibangun di atas format tabel Apache Iceberg, transaksi Athena ACID kompatibel dengan layanan dan mesin lain seperti Amazon EMR dan Apache Spark yang juga mendukung format tabel Iceberg.

Transaksi Athena ACID dan sintaks SQL yang sudah dikenal menyederhanakan pembaruan data bisnis dan peraturan Anda. Misalnya, untuk menanggapi permintaan penghapusan data, Anda dapat melakukan operasi SQL`DELETE`. Untuk membuat koreksi catatan manual, Anda dapat menggunakan satu `UPDATE` pernyataan. Untuk memulihkan data yang baru saja dihapus, Anda dapat mengeluarkan kueri perjalanan waktu menggunakan `SELECT` pernyataan. Transaksi Athena tersedia melalui konsol Athena, operasi API, dan driver ODBC dan JDBC.

Untuk informasi selengkapnya, lihat Menggunakan transaksi ACID Athena.

24 November 2021

Diterbitkan: 2021-11-24

Athena mengumumkan dukungan untuk membaca dan menulis zStandard terkompresi ORC, Parket, dan data textfile. Athena menggunakan kompresi ZStandard level 3 saat menulis data terkompresi ZStandard.

Untuk informasi tentang kompresi data di Athena, lihat. Dukungan kompresi Athena

22 November 2021

Diterbitkan: 2021-11-22

Sekarang Anda dapat mengelola AWS Step Functions alur kerja dari konsol Amazon Athena, sehingga memudahkan pembuatan pipeline pemrosesan data yang dapat diskalakan, menjalankan kueri berdasarkan logika bisnis khusus, mengotomatiskan tugas administratif dan peringatan, dan banyak lagi.

Step Functions sekarang terintegrasi dengan konsol Athena yang ditingkatkan, dan Anda dapat menggunakannya untuk melihat diagram alur kerja interaktif mesin status Anda yang memanggil Athena. Untuk memulai, pilih Alur kerja dari panel navigasi kiri. Jika Anda memiliki mesin status yang ada dengan kueri Athena, pilih mesin status untuk melihat diagram interaktif alur kerja. Jika Anda baru mengenal Step Functions, Anda dapat memulai dengan meluncurkan proyek sampel dari konsol Athena dan menyesuaikannya agar sesuai dengan kasus penggunaan Anda.

Untuk informasi selengkapnya, lihat Membangun dan mengatur pipeline ETL menggunakan Amazon Athena dan, AWS Step Functions atau lihat dokumentasi Step Functions.

18 November 2021

Diterbitkan: 2021-11-18

Athena mengumumkan fitur dan peningkatan baru.

- Support spill-to-disk untuk kueri agregasi yang berisi`DISTINCT`,`ORDER BY`, atau keduanya, seperti pada contoh berikut:

```
SELECT array_agg(orderstatus ORDER BY orderstatus)
FROM orders
GROUP BY orderpriority, custkey
```

- Mengatasi masalah penanganan memori untuk kueri yang digunakan`DISTINCT`. Untuk menghindari pesan kesalahan seperti sumber daya kueri yang habis pada faktor skala ini saat Anda menggunakan `DISTINCT` kueri, pilih kolom yang memiliki kardinalitas rendah`DISTINCT`, atau kurangi ukuran data kueri.
- Dalam `SELECT COUNT(*)` kueri yang tidak menentukan kolom tertentu, meningkatkan kinerja dan penggunaan memori dengan hanya menyimpan hitungan tanpa buffering baris.
- Memperkenalkan fungsi string berikut.

  - `translate(source, from, to)`— Mengembalikan `source` string dengan karakter yang ditemukan dalam `from` string digantikan oleh karakter yang sesuai dalam `to` string. Jika `from` string berisi duplikat, hanya yang pertama digunakan. Jika `source` karakter tidak ada dalam `from` string, `source` karakter disalin tanpa terjemahan. Jika indeks karakter yang cocok dalam `from` string lebih besar dari panjang `to` string, karakter dihilangkan dari string yang dihasilkan.
  - `concat_ws(string0, array(varchar))`— Mengembalikan rangkaian elemen dalam array menggunakan `string0` sebagai pemisah. Jika `string0` null, maka nilai kembalinya adalah null. Setiap nilai null dalam array dilewati.
- Memperbaiki bug di mana kueri gagal saat mencoba mengakses subbidang yang hilang di file. `struct` Query sekarang mengembalikan null untuk subfield yang hilang.
- Memperbaiki masalah dengan hashing yang tidak konsisten untuk tipe data desimal.
- Memperbaiki masalah yang menyebabkan sumber daya habis ketika ada terlalu banyak kolom di partisi.

17 November 2021

Diterbitkan: 2021-11-17

Amazon Athena sekarang mendukung pengindeksan partisi untuk mempercepat kueri pada tabel yang dipartisi di file. AWS Glue Data Catalog

Saat menanyakan tabel yang dipartisi, Athena mengambil dan memfilter partisi tabel yang tersedia ke subset yang relevan dengan kueri Anda. Saat data dan partisi baru ditambahkan, lebih banyak waktu diperlukan untuk memproses partisi dan runtime kueri dapat meningkat. Untuk mengoptimalkan pemrosesan partisi dan meningkatkan kinerja kueri pada tabel yang sangat dipartisi, Athena sekarang mendukung indeks partisi.AWS Glue

Untuk informasi selengkapnya, lihat AWS Glue pengindeksan partisi dan penyaringan.

November 16, 2021

Diterbitkan: 2021-11-16

Konsol Amazon Athena yang baru dan lebih baik sekarang umumnya tersedia di AWS komersial dan GovCloud wilayah di mana Athena tersedia. Konsol baru Athena mendukung semua fitur konsol sebelumnya, tetapi dengan antarmuka modern yang lebih mudah digunakan dan menyertakan fitur

baru yang meningkatkan pengalaman mengembangkan kueri, menganalisis data, dan mengelola penggunaan Anda. Anda sekarang dapat:

- Atur ulang, navigasikan ke, atau tutup beberapa tab kueri dari bilah tab kueri yang didesain ulang.
- Baca dan edit kueri dengan lebih mudah dengan pemformatan SQL dan teks yang ditingkatkan.
- Salin hasil kueri ke clipboard Anda selain mengunduh set hasil lengkap.
- Urutkan riwayat kueri, kueri tersimpan, dan grup kerja, lalu pilih kolom mana yang akan ditampilkan atau disembunyikan.
- Gunakan antarmuka yang disederhanakan untuk mengonfigurasi sumber data dan grup kerja dalam klik yang lebih sedikit.
- Tetapkan preferensi untuk menampilkan hasil kueri, riwayat kueri, pembungkus baris, dan lainnya.
- Tingkatkan produktivitas Anda dengan pintasan keyboard yang baru dan lebih baik serta dokumentasi produk yang disematkan.

Dengan pengumuman hari ini, konsol yang didesain ulang sekarang menjadi default. Untuk memberi tahu kami tentang pengalaman Anda, pilih Umpan Balik di sudut kiri bawah konsol.

Jika diinginkan, Anda dapat menggunakan konsol sebelumnya dengan masuk ke Anda Akun AWS, memilih Amazon Athena, dan membatalkan pilihan pengalaman Athena Baru dari panel navigasi di sebelah kiri.

12 November 2021

Diterbitkan: 2021-11-12

Anda sekarang dapat menggunakan Amazon Athena untuk menjalankan kueri gabungan pada sumber data yang terletak di AWS akun selain milik Anda. Hingga saat ini, kueri data ini membutuhkan sumber data dan konektornya untuk menggunakan yang Akun AWS sama dengan pengguna yang menanyakan data.

Sebagai administrator data, Anda dapat mengaktifkan kueri federasi lintas akun dengan membagikan konektor data Anda dengan akun analis data. Sebagai analis data, Anda dapat menambahkan konektor data yang telah dibagikan oleh administrator data ke akun Anda. Perubahan konfigurasi pada konektor di akun asal berlaku secara otomatis ke konektor bersama.

Untuk informasi tentang mengaktifkan kueri federasi lintas akun, lihat. Mengaktifkan kueri federasi lintas akun Untuk mempelajari tentang menanyakan sumber federasi, lihat Menggunakan Amazon Athena dan. Menjalankan kueri federasi

2 November 2021

Diterbitkan: 2021-11-02

Anda sekarang dapat menggunakan `EXPLAIN ANALYZE` pernyataan di Athena untuk melihat rencana eksekusi terdistribusi dan biaya setiap operasi untuk kueri SQL Anda.

Untuk informasi selengkapnya, lihat Menggunakan JELASKAN dan JELASKAN ANALISIS di Athena.

29 Oktober 2021

Diterbitkan: 2021-10-29

Athena merilis driver JDBC 2.0.25 dan ODBC 1.1.13 dan mengumumkan fitur dan peningkatan.

Driver JDBC dan ODBC

Dirilis JDBC 2.0.25 dan ODBC 1.1.13 driver untuk Athena. Kedua driver menawarkan dukungan untuk otentikasi multi-faktor SAMP browser, yang dapat dikonfigurasi untuk bekerja dengan penyedia SAMP 2.0 mana pun.

Driver JDBC 2.0.25 mencakup perubahan berikut:

- Support untuk otentikasi SAMP browser. Driver menyertakan plugin SAMP browser yang dapat dikonfigurasi untuk bekerja dengan penyedia SAMP 2.0 apa pun.
- Support untuk panggilan AWS Glue API. Anda dapat menggunakan `GlueEndpointOverride` parameter untuk mengganti titik AWS Glue akhir.
- Mengubah jalur `com.simba.athena.amazonaws` kelas ke`com.amazonaws`.

Driver ODBC 1.1.13 mencakup perubahan berikut:

- Support untuk otentikasi SAMP browser. Driver menyertakan plugin SAMP browser yang dapat dikonfigurasi untuk bekerja dengan penyedia SAMP 2.0 apa pun. Untuk contoh cara menggunakan plugin SAMP browser dengan driver ODBC, lihat. Mengkonfigurasi sistem masuk tunggal menggunakan ODBC, SAMP 2.0, dan Penyedia Identitas Okta
- Sekarang Anda dapat mengonfigurasi durasi sesi peran saat menggunakan ADFS, Azure AD, atau Browser Azure AD untuk autentikasi.

Untuk informasi lebih lanjut tentang ini dan perubahan lainnya, dan untuk mengunduh driver baru, catatan rilis, dan dokumentasi, lihat Menghubungkan ke Amazon Athena dengan JDBC danMenghubungkan ke Amazon Athena dengan ODBC.

Fitur dan Perbaikan

Athena mengumumkan fitur dan peningkatan berikut.

- Aturan optimasi baru telah diperkenalkan untuk menghindari pemindaian tabel duplikat dalam kasus-kasus tertentu.

4 Oktober 2021

Diterbitkan: 2021-10-04

Athena mengumumkan fitur dan peningkatan berikut.

- SQL OFFSET - `OFFSET` Klausul SQL sekarang didukung dalam pernyataan. `SELECT` Untuk informasi selengkapnya, lihat SELECT.
- CloudWatch metrik penggunaan — Athena sekarang menerbitkan metrik `ActiveQueryCount` di namespace. `AWS/Usage` Untuk informasi selengkapnya, lihat Memantau metrik penggunaan Athena.
- Perencanaan kueri - Memperbaiki bug yang dalam kasus yang jarang terjadi menyebabkan batas waktu perencanaan kueri.

September 16, 2021

Diterbitkan: 2021-09-16

Athena mengumumkan fitur dan peningkatan baru berikut.

Fitur

- Ditambahkan dukungan untuk menentukan file teks dan kompresi JSON di CTAS menggunakan properti tabel. `write_compression` Anda juga dapat menentukan `write_compression` properti di CTAS untuk format Parket dan ORC. Untuk informasi selengkapnya, lihat Properti tabel CTAS.

- Format kompresi BZIP2 sekarang didukung untuk menulis file teks dan file JSON. Untuk informasi selengkapnya tentang format kompresi di Athena, lihat. Dukungan kompresi Athena

Perbaikan

- Memperbaiki bug di mana informasi identitas gagal dikirim ke fungsi UDF Lambda.
- Memperbaiki masalah pushdown predikat dengan kondisi filter terputus.
- Memperbaiki masalah hashing untuk tipe desimal.
- Memperbaiki masalah pengumpulan statistik yang tidak perlu.
- Menghapus pesan kesalahan yang tidak konsisten.
- Peningkatan kinerja gabungan siaran dengan menerapkan pemangkasan partisi dinamis di node pekerja.
- Untuk kueri federasi:
  - Konfigurasi yang diubah untuk mengurangi terjadinya `CONSTRAINT_VIOLATION` kesalahan dalam kueri federasi.

15 September 2021

Diterbitkan: 2021-09-15

Anda sekarang dapat menggunakan konsol Amazon Athena yang didesain ulang (Pratinjau). Driver Athena JDBC baru telah dirilis.

Pratinjau Konsol Athena

Anda sekarang dapat menggunakan konsol Amazon Athena yang didesain ulang (Pratinjau) dari Wilayah AWS mana pun Athena tersedia. Konsol baru mendukung semua fitur konsol yang ada, tetapi dari antarmuka modern yang lebih mudah digunakan.

Untuk beralih ke konsol baru, masuk ke Anda Akun AWS dan pilih Amazon Athena. Dari bilah navigasi AWS konsol, pilih Beralih ke konsol baru. Untuk kembali ke konsol default, batalkan pilihan Pengalaman Athena Baru dari panel navigasi di sebelah kiri.

Mulai dengan konsol baru hari ini. Pilih Umpan Balik di pojok kiri bawah untuk memberi tahu kami tentang pengalaman Anda.

Pengemudi Athena JDBC 2.0.24

Athena mengumumkan ketersediaan driver JDBC versi 2.0.24 untuk Athena. Rilis ini memperbarui dukungan proxy untuk semua penyedia kredensional. Driver sekarang mendukung otentikasi proxy untuk semua host yang tidak didukung oleh properti `NonProxyHosts` koneksi.

Sebagai kenyamanan, rilis ini mencakup unduhan driver JDBC baik dengan maupun tanpa SDK AWS . Versi driver JDBC ini memungkinkan Anda untuk memiliki driver AWS-SDK dan Athena JDBC yang disematkan dalam proyek.

Untuk informasi selengkapnya dan untuk mengunduh driver baru, catatan rilis, dan dokumentasi, lihatMenghubungkan ke Amazon Athena dengan JDBC.

31 Agustus 2021

Diterbitkan: 2021-08-31

Athena mengumumkan penyempurnaan fitur berikut dan perbaikan bug.

- Peningkatan federasi Athena - Athena telah menambahkan dukungan untuk jenis peta dan dukungan yang lebih baik untuk tipe kompleks sebagai bagian dari Athena Query Federation SDK. Versi ini juga mencakup beberapa peningkatan memori dan pengoptimalan kinerja.
- Kategori kesalahan baru - Memperkenalkan kategori `USER` dan `SYSTEM` kesalahan dalam pesan kesalahan. Kategori ini membantu Anda membedakan antara kesalahan yang dapat Anda perbaiki sendiri (`USER`) dan kesalahan yang memerlukan bantuan dari dukungan Athena ()`SYSTEM`.
- Pesan kesalahan kueri federasi — `USER_ERROR` Kategorisasi yang diperbarui untuk kesalahan terkait kueri gabungan.
- BERGABUNG - Memperbaiki bug spill-to-disk terkait dan masalah memori untuk meningkatkan kinerja dan mengurangi kesalahan memori dalam `JOIN` operasi.

Agustus 12, 2021

Diterbitkan: 2021-08-12

Merilis driver ODBC 1.1.12 untuk Athena. Versi ini memperbaiki masalah yang terkait dengan`SQLPrepare()`,`SQLGetInfo()`, dan`EndpointOverride`.

Untuk mengunduh driver baru, catatan rilis, dan dokumentasi, lihatMenghubungkan ke Amazon Athena dengan ODBC.

6 Agustus 2021

Diterbitkan: 2021-08-06

Amazon Athena mengumumkan ketersediaan Athena dan fitur-fiturnya di Wilayah Asia Pasifik (Osaka).

Rilis ini memperluas ketersediaan Athena di Asia Pasifik termasuk Asia Pasifik (Hong Kong), Asia Pasifik (Mumbai), Asia Pasifik (Osaka), Asia Pasifik (Seoul), Asia Pasifik (Singapura), Asia Pasifik (Sydney), dan Asia Pasifik (Tokyo). Untuk daftar lengkap yang Layanan AWS tersedia di Wilayah ini dan lainnya, lihat Daftar Layanan Wilayah AWS al.

5 Agustus 2021

Diterbitkan: 2021-08-05

Anda dapat menggunakan `UNLOAD` pernyataan untuk menulis output `SELECT` kueri ke format PARQUET, ORC, AVRO, dan JSON.

Untuk informasi selengkapnya, lihat MEMBONGKAR.

30 Juli 2021

Diterbitkan: 2021-07-30

Athena mengumumkan penyempurnaan fitur berikut dan perbaikan bug.

- Pemfilteran dinamis dan pemangkasan partisi — Peningkatan meningkatkan kinerja dan mengurangi jumlah data yang dipindai dalam kueri tertentu, seperti pada contoh berikut.

  Contoh ini mengasumsikan bahwa `Table_B` adalah tabel tidak dipartisi yang memiliki ukuran file yang menambahkan hingga kurang dari 20 MB. Untuk kueri seperti ini, lebih sedikit data yang dibaca `Table_A` dan kueri selesai lebih cepat.

  ```
  SELECT *
  FROM Table_A
  JOIN Table_B ON Table_A.date = Table_B.date
  WHERE Table_B.column_A = "value"
  ```

- ORDER BY with LIMIT, DISTINCT with LIMIT — Peningkatan kinerja untuk kueri yang menggunakan `ORDER BY` atau `DISTINCT` diikuti oleh `LIMIT` klausa.

- File S3 Glacier Deep Archive — Saat Athena menanyakan tabel yang berisi campuran file S3 Glacier Deep Archive dan file Gletser non-S3, Athena sekarang melewatkan file S3 Glacier Deep Archive untuk Anda. Sebelumnya, Anda diminta untuk memindahkan file-file ini secara manual dari lokasi kueri atau kueri akan gagal. Jika Anda ingin menggunakan Athena untuk menanyakan objek di penyimpanan S3 Glacier Deep Archive, Anda harus memulihkannya. Untuk informasi selengkapnya, lihat Memulihkan objek yang diarsipkan di Panduan Pengguna Amazon S3.
- Memperbaiki bug di mana file kosong yang dibuat oleh properti `bucketed_by` tabel CTAS tidak dienkripsi dengan benar.

21 Juli 2021

Diterbitkan pada 2021-07-21

Dengan rilis Juli 2021Microsoft Power BI Desktop, Anda dapat membuat laporan dan dasbor menggunakan konektor sumber data asli untuk Amazon Athena. Konektor untuk Amazon Athena tersedia sebagai konektor standar di Power BI, mendukung DirectQuery, dan memungkinkan analisis pada kumpulan data besar dan penyegaran konten melalui Power BI Gateway.

Karena konektor menggunakan ODBC data source name (DSN) yang ada untuk menyambung ke dan menjalankan kueri pada Athena, memerlukan driver Athena ODBC. Untuk mengunduh driver ODBC terbaru, lihatMenghubungkan ke Amazon Athena dengan ODBC.

Untuk informasi selengkapnya, lihat Menggunakan konektor Amazon Athena Power BI.

16 Juli 2021

Diterbitkan pada 2021-07-16

Amazon Athena telah memperbarui integrasi dengan Apache Hudi. Hudi adalah kerangka manajemen data open-source yang digunakan untuk menyederhanakan pemrosesan data tambahan di danau data Amazon S3. Integrasi diperbarui memungkinkan Anda untuk menggunakan Athena untuk kueri Hudi 0.8.0 tabel dikelola melalui Amazon EMR, Apache Spark, Apache Hive atau layanan lain yang kompatibel. Selain itu, Athena sekarang mendukung dua fitur tambahan: kueri snapshot pada tabel Merge-on-Read (MoR) dan baca dukungan pada tabel bootstrap.

Apache Hudi menyediakan pemrosesan data level rekor yang dapat membantu Anda menyederhanakan pengembangan Perubahan Data Capture (CDC) alur, mematuhi update GDPR-driven dan menghapus, dan lebih baik mengelola streaming data dari sensor atau perangkat yang memerlukan penyisipan data dan update peristiwa. Rilis 0.8.0 membuatnya lebih mudah untuk

memigrasikan tabel Parquet besar ke Hudi tanpa menyalin data sehingga Anda dapat melakukan kueri dan menganalisisnya melalui Athena. Anda dapat menggunakan dukungan baru Athena untuk mengkueri snapshot agar memiliki tampilan yang hampir waktu nyata dari pembaruan tabel streaming Anda.

Untuk mempelajari selengkapnya tentang penggunaan Hudi dengan Athena, lihatMenggunakan Athena untuk menanyakan kumpulan data Apache Hudi.

8 Juli 2021

Diterbitkan pada 2021-07-08

Rilis driver ODBC 1.1.11 untuk Athena. Driver ODBC sekarang dapat mengautentikasi koneksi menggunakan JSON Web Token (JWT). Di Linux, nilai default untuk properti Grup kerja telah ditetapkan ke primer.

Untuk informasi selengkapnya dan untuk mengunduh driver baru, catatan rilis, dan dokumentasi, lihatMenghubungkan ke Amazon Athena dengan ODBC.

1 Juli 2021

Diterbitkan pada 2021-07-01

Pada tanggal 1 Juli 2021, penanganan khusus grup kerja pratinjau berakhir. Sementara`AmazonAthenaPreviewFunctionality`kelompok kerja mempertahankan nama mereka, mereka tidak lagi memiliki status khusus. Anda dapat terus menggunakan`AmazonAthenaPreviewFunctionality`kelompok kerja untuk melihat, memodifikasi, mengatur, dan menjalankan kueri. Namun, kueri yang menggunakan fitur yang sebelumnya ada di pratinjau sekarang tunduk pada persyaratan dan ketentuan penagihan Athena standar. Untuk informasi penagihan, lihatHarga Amazon Athena.

23 Juni 2021

Diterbitkan pada 2021-06-23

Dirilis driver JDBC 2.0.23 dan ODBC 1.1.10 untuk Athena. Kedua driver menawarkan peningkatan performa baca dan dukunganMENJELASKANdankueri parameter.

`EXPLAIN`pernyataan menunjukkan rencana eksekusi logis atau didistribusikan dari kueri SQL. kueri parameter memungkinkan kueri yang sama untuk digunakan beberapa kali dengan nilai-nilai yang berbeda disediakan pada waktu berjalan.

Rilis JDBC juga menambahkan dukungan untuk Active Directory Federation Services 2019 dan kustom endpoint menimpa pilihan untuk AWS STS. Rilis ODBC memperbaiki masalah dengan kredensial profil IAM.

Untuk informasi selengkapnya dan untuk mengunduh driver baru, catatan rilis, dan dokumentasi, lihatMenghubungkan ke Amazon Athena dengan JDBCdanMenghubungkan ke Amazon Athena dengan ODBC.

12 Mei 2021

Diterbitkan pada 2021-05-12

Anda sekarang dapat menggunakan Amazon Athena untuk mendaftarkan AWS Glue katalog dari akun selain milik Anda. Setelah mengonfigurasi izin IAM yang diperlukan AWS Glue, Anda dapat menggunakan Athena untuk menjalankan kueri lintas akun.

Untuk informasi selengkapnya, lihat Mendaftarkan akun AWS Glue Data Catalog dari akun lain dan Akses lintas akun ke katalog AWS Glue data.

10 Mei 2021

Diterbitkan pada 2021-05-10

Dirilis ODBC versi driver 1.1.9.1001 untuk Athena. Versi ini memperbaiki masalah dengan`BrowserAzureAD`jenis otentikasi saat menggunakan Azure Active Directory (AD).

Untuk mengunduh driver baru, catatan rilis, dan dokumentasi, lihatMenghubungkan ke Amazon Athena dengan ODBC.

5 Mei 2021

Diterbitkan pada 2021-05-05

Anda sekarang dapat menggunakan konektor Amazon Athena Vertica dalam kueri gabungan untuk kueri sumber data Vertica dari Athena. Misalnya, Anda dapat menjalankan kueri analitis atas data warehouse di Vertica dan danau data di Amazon S3.

Untuk menggunakan konektor Athena Vertica, kunjungi halaman AthenaVerticaConnectordi. AWS Serverless Application Repository

Konektor Amazon Athena Vertica mengekspos beberapa pilihan konfigurasi melalui variabel lingkungan Lambda. Untuk informasi tentang opsi konfigurasi, parameter, string koneksi, penerapan, dan batasan, lihat. Konektor Amazon Athena Vertica

Untuk informasi mendalam tentang menggunakan konektor Vertica, lihatQuery sumber data Vertica di Amazon Athena menggunakan Kueri Gabungan Athena SDKdiAWS Blog big data.

30 April 2021

Diterbitkan pada 2021-04-30

Driver dirilis JDBC 2.0.21 dan ODBC 1.1.9 untuk Athena. Kedua rilis mendukung otentikasi SAMP dengan Azure Active Directory (AD) dan otentikasi SAMP dengan. PingFederate Rilis JDBC juga mendukung kueri parameter. Untuk informasi tentang kueri parameter di Athena, lihatMenggunakan kueri berparameter.

Untuk mengunduh driver baru, catatan rilis, dan dokumentasi, lihatMenghubungkan ke Amazon Athena dengan JDBCdanMenghubungkan ke Amazon Athena dengan ODBC.

29 April 2021

Diterbitkan pada 2021-04-29

Amazon Athena mengumumkan ketersediaan mesin Athena versi 2 di Wilayah China (Beijing) dan China (Ningxia).

Untuk informasi tentang mesin Athena versi 2, lihatVersi mesin Athena 2.

26 April 2021

Diterbitkan pada 2021-04-26

Fungsi nilai jendela di mesin Athena versi 2 sekarang mendukung`IGNORE NULLS`dan`RESPECT NULLS`.

Untuk informasi selengkapnya, lihat.Fungsi nilaidalam dokumentasi Presto.

21 April 2021

Diterbitkan pada 2021-04-21

Amazon Athena mengumumkan ketersediaan mesin Athena versi 2 di Wilayah Eropa (Milan) dan Africa (Cape Town).

Untuk informasi tentang mesin Athena versi 2, lihatVersi mesin Athena 2.

5 April 2021

Diterbitkan pada 2021-04-05

Pernyataan EXPLAIN

Sekarang Anda dapat menggunakan`EXPLAIN`pernyataan di Athena untuk melihat rencana eksekusi untuk kueri SQL Anda.

Untuk informasi selengkapnya, lihat Menggunakan JELASKAN dan JELASKAN ANALISIS di Athena dan Memahami Athena MENJELASKAN hasil pernyataan.

SageMaker Model Machine Learning dalam Kueri SQL

Inferensi model pembelajaran mesin dengan Amazon sekarang SageMaker umumnya tersedia untuk Amazon Athena. Gunakan model machine learning dalam kueri SQL untuk menyederhanakan tugas-tugas kompleks seperti deteksi anomali, analisis kohort pelanggan, dan prediksi time-series dengan menerapkan fungsi dalam kueri SQL.

Untuk informasi selengkapnya, lihat Menggunakan Machine Learning (ML) dengan Amazon Athena.

Fungsi Ditetapkan Pengguna (UDF)

User defined functions (UDFS) sekarang umumnya tersedia untuk Athena. Gunakan UDFS untuk memanfaatkan fungsi kustom yang memproses catatan atau grup catatan dalam kueri SQL tunggal.

Untuk informasi selengkapnya, lihat Query dengan fungsi yang ditentukan pengguna.

30 Maret 2021

Diterbitkan pada 2021-03-30

Amazon Athena mengumumkan ketersediaan mesin Athena versi 2 di Wilayah Asia Pacific (Hong Kong) dan Middle East (Bahrain).

Untuk informasi tentang mesin Athena versi 2, lihat Versi mesin Athena 2.

25 Maret 2021

Diterbitkan pada 2021-03-25

Amazon Athena mengumumkan ketersediaan mesin Athena versi 2 di Eropa (Stockholm) Wilayah.

Untuk informasi tentang mesin Athena versi 2, lihatVersi mesin Athena 2.

5 Maret 2021

Diterbitkan pada 2021-03-05

Amazon Athena mengumumkan ketersediaan mesin Athena versi 2 di Wilayah Canada (Central), Eropa (Frankfurt), dan South America (Sao Paulo).

Untuk informasi tentang mesin Athena versi 2, lihatVersi mesin Athena 2.

25 Februari 2021

Diterbitkan pada 2021-02-25

Amazon Athena mengumumkan ketersediaan mesin Athena versi 2 di Wilayah Asia Pacific (Seoul), Asia Pacific (Singapore),Asia Pacific (Sydney), Eropa (London), dan Eropa (Paris).

Untuk informasi tentang mesin Athena versi 2, lihat Versi mesin Athena 2.

Catatan rilis Athena untuk tahun 2020

16 Desember 2020

Diterbitkan pada 2020-12-16

Amazon Athena mengumumkan ketersediaan mesin Athena versi 2, Athena Federated Query, dan di Wilayah tambahan. AWS PrivateLink

Mesin Athena versi 2 dan Kueri Gabungan Athena

Amazon Athena mengumumkan ketersediaan mesin Athena versi 2 dan Kueri Gabungan Athena di Wilayah Asia Pacific (Mumbai), Asia Pacific (Tokyo), Eropa (Irlandia), dan US West (N. California).

Athena versi 2 dan kueri gabungan sudah tersedia di Wilayah US East (N. Virginia), US East (Ohio), dan US West (Oregon).

Untuk informasi selengkapnya, lihat Versi mesin Athena 2 dan Menggunakan Amazon Athena.

AWS PrivateLink

AWS PrivateLink untuk Athena sekarang didukung di Wilayah Eropa (Stockholm). Untuk informasi tentang AWS PrivateLink Athena, lihat. Connect ke Amazon Athena menggunakan antarmuka VPC endpoint

24 November 2020

Diterbitkan pada 2020-11-24

Driver dirilis JDBC 2.0.16 dan ODBC 1.1.6 untuk Athena. rilis ini mendukung, di level akun, Okta Verifikasi otentikasi multifaktor (MFA). Anda juga dapat menggunakan Okta MFA untuk mengonfigurasi otentikasi SMS dan otentikasi Google Authenticator sebagai faktor.

Untuk mengunduh driver baru, catatan rilis, dan dokumentasi, lihatMenghubungkan ke Amazon Athena dengan JDBCdanMenghubungkan ke Amazon Athena dengan ODBC.

11 November 2020

Diterbitkan pada 2020-11-11

Amazon Athena mengumumkan ketersediaan umum di Wilayah US East (N. Virginia), US East (Ohio), dan US West (Oregon) untuk Athena versioning mesin 2 dan kueri gabungan.

Versi mesin Athena 2

Amazon Athena mengumumkan ketersediaan umum dari versioning mesin kueri baru, Athena versi 2, di Wilayah US East (N. Virginia), US East (Ohio), dan US West (Oregon).

Athena mesin versi 2 termasuk peningkatan performa dan kemampuan fitur baru seperti skema dukungan evolusi untuk data format Parquet, fungsi geospasial tambahan, dukungan untuk membaca skema Nest untuk mengurangi biaya, dan peningkatan performa dalam BERGABUNG dan AGREGATE operasi.

- Untuk informasi tentang peningkatan, perubahan melanggar, dan perbaikan bug, lihatVersi mesin Athena 2.

- Untuk informasi tentang cara melakukan ini, lihat Mengubah versi mesin Athena.
- Untuk informasi tentang kueri pengujian, lihatMenguji kueri sebelum upgrade versi mesin.

Kueri SQL gabungan

Anda sekarang dapat menggunakan kueri federasi Athena di Wilayah AS Timur (Virginia N.), Timur AS (Ohio), dan AS Barat (Oregon) tanpa menggunakan kelompok kerja. `AmazonAthenaPreviewFunctionality`

Menggunakan kueri SQL Gabungan untuk menjalankan kueri SQL di relasional, non-relasional, objek, dan sumber data kustom. Dengan kueri gabungan, Anda dapat mengirimkan kueri SQL tunggal yang memindai data dari beberapa sumber yang berjalan di tempat atau di-host di cloud.

Menjalankan analisis pada deployment data di seluruh aplikasi dapat menjadi rumit dan memakan waktu karena alasan berikut:

- Data yang diperlukan untuk analitik sering tersebar di relasional, kunci-nilai, dokumen, dalam memori, pencarian, grafik, objek, waktu-seri dan penyimpanan data buku besar.
- Untuk menganalisis data di sumber-sumber ini, analis membangun jaringan alur yang kompleks untuk mengekstraksi, mengubah, dan memuat ke gudang data sehingga data dapat bertanya.
- Mengakses data dari berbagai sumber memerlukan pembelajaran bahasa pemrograman baru dan konstruksi akses data.

Kueri SQL gabungan di Athena menghilangkan kompleksitas ini dengan memungkinkan pengguna untuk kueri data di tempat dari mana pun ia berada. Analis dapat menggunakan konstruksi SQL akrab untuk`JOIN`data di beberapa sumber data untuk analisis cepat, dan menyimpan hasil di Amazon S3 untuk penggunaan selanjutnya.

Konektor Sumber Data

Untuk memproses kueri gabungan, Athena menggunakan Penyambung Sumber Data Athena yang berjalan diAWS Lambda. Konektor open source dan pre-built berikut ditulis dan diuji oleh Athena. Gunakan mereka untuk menjalankan kueri SQL di Athena pada sumber data yang sesuai.

- CloudWatch
- Metrik-metrik CloudWatch
- DocumentDB

- DynamoDB
- OpenSearch
- HBase
- Neptunus
- Redis
- Aliran waktu
- Tolok Ukur TPC DS (TPC-DS)

Konektor Sumber Data Khusus

MenggunakanAthena Permintaan Gabungan SDK, developer dapat membuat konektor ke sumber data apa pun untuk memungkinkan Athena menjalankan kueri SQL terhadap sumber data tersebut. Konektor Federasi Kueri Athena memperluas manfaat kueri federasi di luar konektor yang disediakan. AWS Karena konektor berjalan AWS Lambda, Anda tidak perlu mengelola infrastruktur atau merencanakan penskalaan ke permintaan puncak.

Langkah Berikutnya

- Untuk mempelajari selengkapnya tentang fitur kueri gabungan, lihatMenggunakan Amazon Athena.
- Untuk memulai menggunakan konektor yang ada, lihatMenyebarkan konektor dan menghubungkan ke sumber Data.
- Untuk mempelajari cara membuat konektor sumber data Anda sendiri menggunakan Athena Query Federation SDK, lihat Contoh Konektor Athena aktif. GitHub

22 Oktober 2020

Diterbitkan pada 2020-10-22

Anda sekarang dapat menghubungi Athena dengan. AWS Step Functions AWS Step Functions dapat mengontrol tertentu Layanan AWS secara langsung menggunakan Bahasa Amazon States. Anda dapat menggunakan Step Functions dengan Athena untuk memulai dan menghentikan eksekusi kueri, mendapatkan hasil kueri, menjalankan ad-hoc atau dijadwalkan permintaan data, dan mengambil hasil dari danau data di Amazon S3.

Untuk informasi selengkapnya, lihat Memanggil Lambda dengan Step Functions dalam Panduan Developer AWS Step Functions .

29 Juli 2020

Diterbitkan pada 2020-07-29

Driver DBC dirilis versi 2.0.13. Rilis ini mendukung menggunakan beberapakatalog data yang terdaftar dengan Athena, layanan Okta untuk otentikasi, dan koneksi ke VPC endpoint.

Untuk mengunduh dan menggunakan versi baru driver, lihatMenghubungkan ke Amazon Athena dengan JDBC.

9 Juli 2020

Diterbitkan pada 2020-07-09

Amazon Athena menambahkan dukungan untuk kueri kumpulan data Hudi yang dipadatkan dan menambahkan AWS CloudFormation `AWS::Athena::DataCatalog` sumber daya untuk membuat, memperbarui, atau menghapus katalog data yang Anda daftarkan di Athena.

Mengkueri Apache Hudi Dataset

Apache Hudi adalah kerangka manajemen data open-source yang menyederhanakan pemrosesan data tambahan. Amazon Athena sekarang mendukung kueri tampilan baca-dioptimalkan dari set data Apache Hudi di danau data berbasis Amazon S3 Anda.

Untuk informasi selengkapnya, lihat Menggunakan Athena untuk menanyakan kumpulan data Apache Hudi.

AWS CloudFormation Sumber Daya Katalog Data

Para usar o Amazon AthenaFitur kueri gabunganuntuk meminta sumber data apa pun, Anda harus terlebih dahulu mendaftarkan katalog data Anda di Athena. Anda sekarang dapat menggunakan AWS CloudFormation `AWS::Athena::DataCatalog` sumber daya untuk membuat, memperbarui, atau menghapus katalog data yang Anda daftarkan di Athena.

Untuk informasi selengkapnya, lihat AWS::Athena::DataCatalogdi Panduan AWS CloudFormation Pengguna.

1 Juni 2020

Diterbitkan pada 2020-06-01

Menggunakan Apache Hive Metastore sebagai Metacatalog dengan Amazon Athena

Anda sekarang dapat menghubungkan Athena untuk satu atau lebih metastores Apache Hive selain AWS Glue Data Catalog Dengan Athena.

Untuk terhubung ke metastore Hive self-host, Anda memerlukan konektor metastore Athena Hive. Athena menyediakanimplementasi referensiKonektor yang dapat Anda gunakan. Konektor berjalan sebagai AWS Lambda di akun Anda.

Untuk informasi selengkapnya, lihat Menggunakan Athena Data Connector untuk Eksternal Hive Metastore.

21 Mei 2020

Diterbitkan pada 2020-05-21

Amazon Athena menambahkan dukungan untuk proyeksi partisi. Gunakan proyeksi partisi untuk mempercepat pemrosesan kueri tabel yang sangat dipartisi dan mengotomatiskan manajemen partisi. Untuk informasi selengkapnya, lihat Proyeksi partisi dengan Amazon Athena.

1 April 2020

Diterbitkan pada 2020-04-01

Selain Wilayah US East (N. Virginia), kueri gabungan Amazon Athena Menggunakan Amazon Athena, fungsi yang ditetapkan pengguna (UDFS),Inferensi Machine Learning, danEksternal Hive metastore kini tersedia dalam pratinjau di Wilayah Asia Pacific (Mumbai), Europe (Ireland), dan US West (Oregon).

11 Maret 2020

Diterbitkan pada 2020-03-11

Amazon Athena sekarang menerbitkan acara EventBridge Amazon untuk transisi status kueri. Saat kueri bertransisi antar status -- misalnya, dari Berjalan ke status terminal seperti Berhasil atau Dibatalkan - Athena menerbitkan peristiwa perubahan status kueri ke. EventBridge Peristiwa ini berisi informasi tentang transisi negara kueri. Untuk informasi selengkapnya, lihat Memantau kueri Athena dengan acara Amazon EventBridge.

6 Maret 2020

Diterbitkan pada 2020-03-06

Anda sekarang dapat membuat dan memperbarui workgroup Amazon Athena dengan menggunakan sumber daya. AWS CloudFormation `AWS::Athena::WorkGroup` Untuk informasi selengkapnya, lihat AWS::Athena::WorkGroupdi Panduan AWS CloudFormation Pengguna.

Catatan rilis Athena untuk 2019

26 November 2019

Diterbitkan pada 2019-12-17

Amazon Athena menambahkan dukungan untuk menjalankan kueri SQL di relasional, non-relasional, objek, dan sumber data kustom, memohon model machine learning dalam kueri SQL, User Defined Functions (UDFS) (Preview), menggunakan Apache Hive Metastore sebagai katalog metadata dengan Amazon Athena (Preview), dan empat kueri tambahan -terkait metrik.

Kueri SQL gabungan

Menggunakan kueri SQL Gabungan untuk menjalankan kueri SQL di relasional, non-relasional, objek, dan sumber data kustom.

Anda sekarang dapat menggunakan kueri federasi Athena untuk memindai data yang disimpan dalam sumber data relasional, non-relasional, objek, dan kustom. Dengan kueri gabungan, Anda dapat mengirimkan kueri SQL tunggal yang memindai data dari beberapa sumber yang berjalan di tempat atau di-host di cloud.

Menjalankan analisis pada deployment data di seluruh aplikasi dapat menjadi rumit dan memakan waktu karena alasan berikut:

- Data yang diperlukan untuk analitik sering tersebar di relasional, kunci-nilai, dokumen, dalam memori, pencarian, grafik, objek, waktu-seri dan penyimpanan data buku besar.
- Untuk menganalisis data di sumber-sumber ini, analis membangun jaringan alur yang kompleks untuk mengekstraksi, mengubah, dan memuat ke gudang data sehingga data dapat bertanya.
- Mengakses data dari berbagai sumber memerlukan pembelajaran bahasa pemrograman baru dan konstruksi akses data.

Kueri SQL gabungan di Athena menghilangkan kompleksitas ini dengan memungkinkan pengguna untuk kueri data di tempat dari mana pun ia berada. Analis dapat menggunakan konstruksi SQL

akrab untuk`JOIN`data di beberapa sumber data untuk analisis cepat, dan menyimpan hasil di Amazon S3 untuk penggunaan selanjutnya.

Konektor Sumber Data

Athena memproses kueri gabungan menggunakan Penyambung Sumber Data Athena yang berjalan diAWS Lambda. Gunakan konektor sumber data sumber terbuka ini untuk menjalankan kueri SQL federasi di Athena di seluruh Amazon DynamoDB, Apache HBase, Amazon Document DB, Amazon, CloudWatch CloudWatch Amazon Metrics, dan database relasional yang sesuai dengan JDBC seperti MySQL, dan PostgreSQL di bawah lisensi Apache 2.0.

Konektor Sumber Data Khusus

MenggunakanAthena Permintaan Gabungan SDK, developer dapat membuat konektor ke sumber data apa pun untuk memungkinkan Athena menjalankan kueri SQL terhadap sumber data tersebut. Konektor Federasi Kueri Athena memperluas manfaat kueri federasi di luar konektor yang disediakan. AWS Karena konektor berjalan AWS Lambda, Anda tidak perlu mengelola infrastruktur atau merencanakan penskalaan ke permintaan puncak.

Pratinjau

Athena kueri gabungan tersedia di preview di Wilayah US East (N. Virginia).

Langkah Berikutnya

- Untuk memulai pratinjau, ikuti petunjuk dalamFAQ Fitur Athena Preview.
- Untuk mempelajari selengkapnya tentang fitur kueri gabungan, lihatMenggunakan Amazon Kueri Gabungan Athena (Preview).
- Untuk memulai menggunakan konektor yang ada, lihatMenyebarkan konektor dan menghubungkan ke sumber Data.
- Untuk mempelajari cara membuat konektor sumber data Anda sendiri menggunakan Athena Query Federation SDK, lihat Contoh Konektor Athena aktif. GitHub

Meminjam Model Machine Learning dalam SQL Kueri

Anda sekarang dapat memanggil model machine learning untuk inferensi langsung dari pertanyaan Athena Anda. Kemampuan untuk menggunakan model machine learning dalam kueri SQL membuat tugas-tugas kompleks seperti deteksi anomali, analisis kohort pelanggan, dan prediksi penjualan yang sederhana seperti memohon fungsi dalam kueri SQL.

Model ML

Anda dapat menggunakan lebih dari selusin algoritma pembelajaran mesin bawaan yang disediakan oleh Amazon SageMaker, melatih model Anda sendiri, atau menemukan dan berlangganan paket model dari AWS Marketplacedan menyebarkan di Amazon SageMaker Hosting Services. Tidak ada pengaturan tambahan yang diperlukan. Anda dapat memanggil model ML ini dalam kueri SQL Anda dari konsol Athena, Athena API, dan melalui driver JDBC pratinjau Athena.

Pratinjau

Fungsionalitas ML's Athena tersedia hari ini dalam pratinjau di Wilayah AS Timur (Virginia N.).

Langkah Berikutnya

- Untuk memulai pratinjau, ikuti petunjuk dalamFAQ Fitur Athena Preview.
- Untuk mempelajari selengkapnya tentang fitur machine learning, lihatMenggunakan Machine Learning (ML) dengan Amazon Athena (Preview).

Fungsi Ditetapkan Pengguna (UDFS) (Preview)

Anda sekarang dapat menulis fungsi skalar kustom dan memanggil mereka dalam pertanyaan Athena Anda. Anda dapat menulis UDFS Anda di Jawa menggunakanAthena Permintaan Gabungan SDK. Saat UDF digunakan dalam kueri SQL diserahkan ke Athena, itu dipanggil dan berjalan padaAWS Lambda. UDFS dapat digunakan di keduaSELECTdanFILTERklausa kueri SQL. Anda dapat memanggil beberapa UDFS dalam kueri yang sama.

Pratinjau

Fungsi Athena UDF tersedia dalam mode Preview di Wilayah US East (N. Virginia).

Langkah Berikutnya

- Untuk memulai pratinjau, ikuti petunjuk dalam FAQ Fitur Pratinjau Athena.
- Untuk mempelajari selengkapnya, lihat Mengkueri dengan Fungsi Ditetapkan Pengguna (Pratinjau).
- Misalnya implementasi UDF, lihat Amazon Athena UDF Connector aktif. GitHub
- Untuk mempelajari cara menulis fungsi Anda sendiri menggunakan SDK Gabungan Kueri Athena, lihat Membuat dan Men-deploy UDF Menggunakan Lambda.

Menggunakan Apache Hive Metastore sebagai Metacatalog dengan Amazon Athena (Preview)

Anda sekarang dapat menghubungkan Athena untuk satu atau lebih Apache Hive Metastores selain AWS Glue Data Catalog Dengan Athena.

Konektor MetaStore

Untuk terhubung ke Hive Metastore self-host, Anda memerlukan konektor Athena Hive Metastore. Athena menyediakanimplementasi referensiKonektor yang dapat Anda gunakan. Konektor berjalan sebagai AWS Lambda fungsi di akun Anda. Untuk informasi selengkapnya, lihat.Menggunakan Athena Data Connector untuk eksternal Hive Metastore (Preview).

Pratinjau

Fitur Hive Metastore tersedia dalam mode Preview di Wilayah US East (N. Virginia).

Langkah Berikutnya

- Untuk memulai pratinjau, ikuti petunjuk dalamFAQ Fitur Athena Preview.
- Untuk mempelajari selengkapnya tentang fitur ini, kunjungiMenggunakan Athena Data Connector untuk eksternal Hive Metastore (Preview).

Metrik Terkait Kueri Baru

Athena sekarang menerbitkan metrik kueri tambahan yang dapat membantu Anda memahamiAmazon Athenaperforma. Athena menerbitkan metrik terkait kueri ke Amazon. CloudWatch Dalam rilis ini, Athena menerbitkan metrik kueri tambahan berikut:

- Waktu Perencanaan Kueri— Waktu yang dibutuhkan untuk merencanakan kueri. Ini termasuk waktu yang dihabiskan untuk mengambil partisi tabel dari sumber data.
- Waktu Queri antrian— Waktu yang kueri dalam antrian menunggu sumber daya.
- Waktu Pemrosesan Layanan— Waktu yang dibutuhkan untuk menulis hasil setelah mesin permintaan selesai pemrosesan.
- Total Waktu Eksekusi— Waktu Athena mengambil untuk menjalankan kueri.

Untuk menggunakan metrik kueri baru ini, Anda dapat membuat dasbor khusus, menyetel alarm, dan pemicu pada metrik CloudWatch, atau menggunakan dasbor yang telah diisi sebelumnya langsung dari konsol Athena.

Langkah Berikutnya

Untuk informasi selengkapnya, lihat Memantau Kueri Athena dengan Metrik. CloudWatch

12 November 2019

Diterbitkan pada 2019-12-17

Amazon Athena kini tersedia di Wilayah Middle East (Bahrain).

8 November 2019

Diterbitkan pada 2019-12-17

Amazon Athena kini tersedia di Wilayah US West (N. California) dan Europe (Paris).

8 Oktober 2019

Diterbitkan pada 2019-12-17

Amazon Athenasekarang memungkinkan Anda untuk terhubung langsung ke Athena melalui VPC endpoint antarmuka di Virtual Private Cloud (VPC) Anda. Dengan menggunakan fitur ini, Anda dapat mengirimkan pertanyaan Anda ke Athena dengan aman tanpa memerlukan Internet Gateway di VPC Anda.

Untuk membuat antarmuka VPC endpoint untuk terhubung ke Athena, Anda dapat menggunakan or (). AWS Management Console AWS Command Line Interface AWS CLI Untuk informasi tentang membuat titik akhir antarmuka, lihatMembuat Endpoint Interface.

Saat Anda menggunakan titik akhir VPC antarmuka, komunikasi antara VPC dan Athena API Anda aman dan tetap berada dalam jaringan. AWS Tidak ada biaya Athena tambahan untuk menggunakan fitur ini. Biaya VPC endpoint antarmuka berlaku.

Untuk mempelajari selengkapnya tentang fitur ini, lihat Terhubung ke Amazon Athena Menggunakan VPC Endpoint Antarmuka.

19 September 2019

Diterbitkan pada 2019-12-17

Amazon Athena menambahkan dukungan untuk memasukkan data baru ke tabel yang ada menggunakan`INSERT INTO`. Anda dapat memasukkan baris baru ke dalam tabel tujuan berdasarkan`SELECT`pernyataan permintaan yang berjalan pada tabel sumber, atau didasarkan pada serangkaian nilai-nilai yang disediakan sebagai bagian dari pernyataan permintaan. Format data yang didukung meliputi Avro, JSON, ORC, Parquet, dan file teks.

`INSERT INTO`pernyataan juga dapat membantu Anda menyederhanakan proses ETL Anda. Misalnya, Anda dapat menggunakan`INSERT INTO`dalam permintaan tunggal untuk memilih data dari tabel sumber yang dalam format JSON dan menulis ke tabel tujuan dalam format Parquet.

`INSERT INTO`pernyataan dibebankan berdasarkan jumlah byte yang dipindai di`SELECT`fase, mirip dengan bagaimana Athena biaya untuk`SELECT`kueri. Untuk informasi selengkapnya, lihat Harga Amazon Athena.

Untuk informasi selengkapnya tentang penggunaan`INSERT INTO`, termasuk format SerDes dan contoh yang didukung, lihat MEMASUKKAN KE dalam Panduan Pengguna Athena.

12 September 2019

Diterbitkan pada 2019-12-17

Amazon Athena kini tersedia di Wilayah Asia Pacific (Hong Kong).

16 Agustus 2019

Diterbitkan pada 2019-12-17

Amazon Athenamenambahkan dukungan untuk kueri data di Amazon S3 Requester Pays bucket.

Saat bucket Amazon S3 dikonfigurasi sebagai Pemohon Pays, pemohon, bukan pemilik bucket, membayar untuk permintaan Amazon S3 dan biaya transfer data. Di Athena, grup kerja administrator sekarang dapat mengonfigurasi pengaturan grup kerja untuk memungkinkan anggota grup kerja untuk kueri S3 Requester membayar bucket.

Untuk informasi tentang cara mengonfigurasi pengaturan Pemohon membayar untuk grup kerja Anda, lihatMembuat Workgroupdalam Panduan Pengguna Amazon Athena. Untuk informasi

selengkapnya tentang Pemohon membayar bucket, lihatPemohon Membayar Bucketdalam Panduan Developer Amazon Simple Storage Service

9 Agustus 2019

Diterbitkan pada 2019-12-17

Amazon Athena sekarang mendukung penegakanAWS Lake Formationkebijakan untuk kontrol akses berbutir halus ke basis data baru atau yang sudah ada, tabel, dan kolom yang didefinisikan dalamAWS Glue Data CatalogUntuk data yang disimpan di Amazon S3.

Anda dapat menggunakan fitur ini sebagai berikut Wilayah AWS: AS Timur (Ohio), AS Timur (Virginia N.), AS Barat (Oregon), Asia Pasifik (Tokyo), dan Eropa (Irlandia). Tidak ada biaya tambahan untuk penggunaan fitur ini.

Untuk informasi selengkapnya tentang penggunaan, lihat Menggunakan Athena untuk menanyakan data yang terdaftar AWS Lake Formation. Untuk informasi selengkapnya tentang AWS Lake Formation, lihat AWS Lake Formation.

26 Juni 2019

Amazon EKS kini tersedia di Wilayah Europe (Stockholm) (). Untuk daftar Wilayah yang didukung, lihat Wilayah AWS dan Titik Akhir.

24 Mei 2019

Dipublikasikan pada 2019-05-24

Amazon Athena sekarang tersedia di Wilayah AWS GovCloud (AS-Timur) dan AWS GovCloud (AS-Barat). Untuk daftar Wilayah yang didukung, lihat Wilayah AWS dan Titik Akhir.

5 Maret 2019

Publikasikan03-05

Amazon Athena kini tersedia di Wilayah Canada (Central). Untuk daftar Wilayah yang didukung, lihat Wilayah AWS dan Titik Akhir. Merilis versi baru dari driver ODBC dengan dukungan untuk Athena workgroups. Untuk informasi selengkapnya, lihat Catatan Rilis Driver ODBC.

Untuk mengunduh driver ODBC versi 1.0.5 dan dokumentasi, lihatMenghubungkan ke Amazon Athena dengan ODBC. Untuk informasi tentang versi ini, lihatCatatan rilis driver ODBC.

Untuk menggunakan grup kerja dengan driver ODBC, mengatur properti koneksi baru,`Workgroup`, dalam rangkaian koneksi seperti yang ditunjukkan dalam contoh berikut:

```
Driver=Simba Athena ODBC
 Driver;AwsRegion=[Region];S3OutputLocation=[S3Path];AuthenticationType=IAM
 Credentials;UID=[YourAccessKey];PWD=[YourSecretKey];Workgroup=[WorkgroupName]
```

Untuk informasi lebih lanjut, cari “workgroup” di Panduan Instalasi dan Konfigurasi Driver ODBC versi 1.0.5.Ada tidak ada perubahan ke string koneksi driver ODBC saat Anda menggunakan tanda pada grup kerja. Untuk menggunakan tanda, mutakhirkan ke versi terbaru dari driver ODBC, yang merupakan versi saat ini.

Versi driver ini memungkinkan Anda menggunakanTindakan grup kerja API Athenauntuk membuat dan mengelola grup kerja, danTindakan tanda API Athenauntuk menambahkan, membuat daftar, atau menghapus tanda pada grup kerja. Sebelum memulai, pastikan bahwa Anda memiliki izin level sumber daya di IAM untuk tindakan pada grup kerja dan tanda.

Untuk informasi selengkapnya, lihat:

- Menggunakan workgroup untuk menjalankan kueri dan Kebijakan contoh kelompok kerja.
- Menandai sumber daya Athena dan Kebijakan kontrol akses IAM berbasis tag.

Jika Anda menggunakan driver JDBC atau AWS SDK, tingkatkan ke versi terbaru driver dan SDK, yang keduanya sudah menyertakan dukungan untuk kelompok kerja dan tag di Athena. Untuk informasi selengkapnya, lihat Menghubungkan ke Amazon Athena dengan JDBC.

22 Februari 2019

Publikasikan02-22

Menambahkan dukungan tanda untuk grup kerja di Amazon Athena. Setiap tanda terdiri atas sebuah kunci dan sebuah nilai, yang keduanya Anda tentukan. Saat Anda tanda grup kerja, Anda menetapkan metadata kustom untuk itu. Anda dapat menambahkan tag ke grup kerja untuk membantu mengkategorikannya, menggunakan praktik terbaik AWS penandaan. Anda dapat menggunakan tanda untuk membatasi akses ke grup kerja, dan melacak biaya. Sebagai contoh, membuat grup kerja untuk setiap pusat biaya. Kemudian, dengan menambahkan tanda ke grup kerja ini, Anda dapat melacak pengeluaran Athena Anda untuk setiap pusat biaya. Untuk informasi selengkapnya, lihat.Menggunakan Tanda untuk PenagihandiAWS Billing and Cost Management Panduan Pengguna.

Anda dapat bekerja dengan tanda dengan menggunakan konsol Athena atau operasi API. Untuk informasi selengkapnya, lihat Menandai sumber daya Athena.

Di konsol Athena, Anda dapat menambahkan satu atau lebih tanda ke masing-masing grup kerja Anda, dan mencari berdasarkan tanda. Grup kerja adalah sumber daya yang dikontrol IAM di Athena. Dalam IAM, Anda dapat membatasi yang dapat menambahkan, menghapus, atau daftar tanda pada grup kerja yang Anda buat. Anda juga dapat menggunakan`CreateWorkGroup`API operasi yang memiliki opsional tanda parameter untuk menambahkan satu atau lebih tanda ke grup kerja. Untuk menambahkan, menghapus, atau daftar tanda, gunakan`TagResource`,`UntagResource`, dan`ListTagsForResource`. Untuk informasi selengkapnya, lihat Menggunakan operasi tag.

Untuk memungkinkan pengguna untuk menambahkan tanda saat membuat grup kerja, memastikan bahwa Anda memberikan setiap pengguna IAM izin untuk kedua`TagResource`dan`CreateWorkGroup`Tindakan API. Untuk informasi selengkapnya dan contoh tambahan, lihat Kebijakan kontrol akses IAM berbasis tag.

Tidak ada perubahan driver JDBC saat Anda menggunakan tanda pada grup kerja. Jika Anda membuat grup kerja baru dan menggunakan driver JDBC atau AWS SDK, tingkatkan ke versi terbaru driver dan SDK. Untuk informasi, lihat Menghubungkan ke Amazon Athena dengan JDBC.

18 Februari 2019

Publikasikan02-18

Menambahkan kemampuan untuk mengontrol biaya permintaan dengan menjalankan kueri dalam grup kerja. Untuk informasi, lihat Menggunakan kelompok kerja untuk mengontrol akses kueri dan biaya. Memperbaiki SerDe OpenX JSON yang digunakan di Athena, memperbaiki masalah di mana Athena tidak mengabaikan objek yang dialihkan ke kelas penyimpanan, dan menambahkan contoh untuk menanyakan log `GLACIER` Network Load Balancer.

Membuat perubahan berikut:

- Menambahkan dukungan untuk grup kerja. Menggunakan grup kerja untuk memisahkan pengguna, tim, aplikasi, atau beban kerja, dan untuk menetapkan batas pada jumlah data setiap permintaan atau seluruh grup kerja dapat memproses. Karena grup kerja bertindak sebagai sumber daya IAM, Anda dapat menggunakan izin level sumber daya untuk mengontrol akses ke grup kerja tertentu. Anda juga dapat melihat metrik terkait kueri di Amazon CloudWatch, mengontrol biaya kueri dengan mengonfigurasi batas jumlah data yang dipindai, membuat ambang batas, dan memicu tindakan, seperti alarm Amazon SNS, saat ambang batas ini dilanggar. Untuk

informasi selengkapnya, lihat Menggunakan workgroup untuk menjalankan kueri dan Mengontrol biaya dan memantau kueri dengan CloudWatch metrik dan peristiwa.

Grup kerja adalah sumber daya IAM. Untuk daftar lengkap tindakan, sumber daya, dan syarat yang berhubungan dengan grup kerja di IAM, lihat Tindakan, Sumber Daya, dan Kunci Syarat untuk Amazon Athena di Referensi Otorisasi Layanan. Sebelum Anda membuat workgroup baru, pastikan bahwa Anda menggunakan kebijakan IAM workgroup, dan. AWS kebijakan terkelola: AmazonAthenaFullAccess

Anda dapat mulai menggunakan grup kerja di konsol, dengan Operasi API grup kerja, atau dengan driver JDBC. Untuk prosedur tingkat tinggi, lihat Menyiapkan kelompok kerja. Untuk mengunduh driver JDBC dengan dukungan grup kerja, lihat Menghubungkan ke Amazon Athena dengan JDBC.

Jika Anda menggunakan grup kerja dengan driver JDBC, Anda harus mengatur nama grup kerja dalam rangkaian koneksi menggunakan parameter konfigurasi `Workgroup` seperti dalam contoh berikut:

```
jdbc:awsathena://AwsRegion=<AWSREGION>;UID=<ACCESSKEY>;
PWD=<SECRETKEY>;S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/<athena-
output>-<AWSREGION>/;
Workgroup=<WORKGROUPNAME>;
```

Tidak ada perubahan dalam cara Anda menjalankan pernyataan SQL atau membuat JDBC API panggilan ke driver. Sopir melewati nama grup kerja untuk Athena.

Untuk informasi tentang perbedaan yang diperkenalkan dengan kelompok kerja, lihat API grup kerja Athena dan Pemecahan masalah kelompok kerja.

- Peningkatan SerDe OpenX JSON yang digunakan di Athena. Perbaikan meliputi, tetapi tidak terbatas pada, berikut:
  - Support untuk `ConvertDotsInJsonKeysToUnderscores` properti. Ketika diatur ke `TRUE`, ini memungkinkan SerDe untuk mengganti titik-titik dalam nama kunci dengan garis bawah. Sebagai contoh, jika set data JSON berisi kunci dengan nama `"a.b"`, Anda dapat menggunakan properti ini untuk menentukan nama kolom menjadi `"a_b"` di Athena. Defaultnya adalah `FALSE`. Secara default, Athena tidak mengizinkan titik-titik dalam nama kolom.
  - Support untuk `case.insensitive` properti. Secara default, Athena mengharuskan semua kunci dalam set data JSON Anda menggunakan huruf kecil. Menggunakan `WITH SERDE PROPERTIES ("case.insensitive"= FALSE;)` memungkinkan Anda untuk menggunakan

nama kunci case-sensitive dalam data Anda. Defaultnya adalah `TRUE`. Ketika diatur ke`TRUE`, SerDe mengubah semua kolom huruf besar menjadi huruf kecil.

Untuk informasi selengkapnya, lihat OpenX JSON SerDe.

- Memperbaiki masalah saat Athena kembali`"access denied"`pesan kesalahan, saat diproses Amazon S3 objek yang diarsipkan ke Glacier oleh kebijakan siklus hidup Amazon S3. Sebagai hasil dari memperbaiki masalah ini, Athena mengabaikan objek transisi ke`GLACIER`kelas penyimpanan. Athena tidak mendukung kueri data dari`GLACIER`kelas penyimpanan.

  Untuk informasi selengkapnya, lihat the section called “Persyaratan untuk tabel di Athena dan data di Amazon S3” dan Transisi ke Kelas Penyimpanan GLACIER (Object Archival) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.
- Menambahkan contoh untuk kueri Network Load Balancer akses log yang menerima informasi tentang permintaan Transport Layer Security (TLS). Untuk informasi selengkapnya, lihat the section called “Network Load Balancer”.

Catatan rilis Athena untuk 2018

20 November 2018

Publikasikan2018-11-20

Merilis versi baru driver JDBC dan ODBC dengan dukungan untuk akses gabungan ke API Athena dengan AD FS dan SAML 2.0 (Security Assertion Markup Language 2.0). Untuk detailnya, lihat Catatan Rilis Driver JDBC dan Catatan Rilis Driver ODBC.

Dengan rilis ini, akses Gabungan ke Athena didukung untuk Active Directory Federation Service (AD FS 3.0). Akses dibuat melalui versi driver JDBC atau ODBC yang mendukung SAML 2.0. Untuk informasi tentang cara mengonfigurasi akses gabungan ke API Athena, lihatthe section called “Mengaktifkan akses federasi ke Athena API”.

Untuk mengunduh driver JDBC versi 2.0.6 dan dokumentasinya, lihatMenghubungkan ke Amazon Athena dengan JDBC. Untuk informasi tentang versi ini, lihatCatatan Rilis Driver JDBC.

Untuk mengunduh driver ODBC versi 1.0.4 dan dokumentasi, lihatMenghubungkan ke Amazon Athena dengan ODBC. Untuk informasi tentang versi ini,Catatan rilis driver ODBC.

Untuk informasi selengkapnya tentang dukungan SAMP 2.0 di AWS, lihat Tentang Federasi SAMP 2.0 di Panduan Pengguna IAM.

15 Oktober 2018

Publikasikan2018-10-15

Jika Anda telah memutakhirkan ke AWS Glue Data Catalog, ada dua fitur baru yang memberikan dukungan untuk:

- Enkripsi metadata Katalog Data. Jika Anda memilih untuk mengenkripsi metadata dalam Katalog Data, Anda harus menambahkan kebijakan tertentu ke Athena. Untuk informasi selengkapnya, lihat.Akses ke Metadata Terenkripsi di AWS Glue Data Catalog.
- Izin berbutir halus untuk mengakses sumber daya di file. AWS Glue Data Catalog Anda sekarang dapat menentukan kebijakan berbasis identitas (IAM) yang membatasi atau mengizinkan akses ke basis data dan tabel tertentu dari Katalog Data yang digunakan di Athena. Untuk informasi selengkapnya, lihat Akses berbutir halus ke database dan tabel di AWS Glue Data Catalog.

> **Note**
>
> Data berada di bucket Amazon S3, dan akses ke sana dikendalikan oleh. Akses ke Amazon S3 dari Athena Untuk mengakses data dalam basis data dan tabel, terus menggunakan kebijakan kontrol akses ke bucket Amazon S3 yang menyimpan data.

10 Oktober 2018

Publikasikan2018-10-10

Dukungan Athena`CREATE TABLE AS SELECT`, yang menciptakan tabel dari hasil`SELECT`. Untuk detailnya, lihatMembuat Tabel dari Hasil Kueri (CTAS).

Sebelum Anda membuat pertanyaan CTAS, penting untuk belajar tentang perilaku mereka dalam dokumentasi Athena. Ini berisi informasi tentang lokasi untuk menyimpan hasil kueri di Amazon S3, daftar format yang didukung untuk menyimpan hasil kueri CTAS, jumlah partisi yang dapat Anda buat, dan format kompresi yang didukung. Untuk informasi selengkapnya, lihat Pertimbangan dan batasan untuk kueri CTAS.

Gunakan kueri CTAS untuk:

- Membuat tabel dari hasil querydalam satu langkah.

- Buat kueri CTAS di konsol Athena, menggunakanContoh. Untuk informasi tentang sintaks, lihatBUAT TABEL SEBAGAI.
- Ubah hasil kueri ke format penyimpanan lainnya, seperti Parquet, ORC, AVRO, JSON, dan TEXTFILE Untuk informasi selengkapnya, lihat Pertimbangan dan batasan untuk kueri CTAS dan Format penyimpanan kolumnar.

6 September 2018

Publikasikan2018-09-06

Dirilis versi baru driver ODBC (versi 1.0.3). Versi baru dari driver ODBC aliran hasil secara default, bukan paging melalui mereka, memungkinkan alat intelijen bisnis untuk mengambil set big data lebih cepat. Versi ini juga mencakup perbaikan, perbaikan bug, dan dokumentasi yang diperbarui untukMenggunakan SSL dengan Server Proxy". Untuk detailnya, lihat Catatan Rilis untuk pengemudi.

Untuk memuat turun versi pemandu ODBC 1.0.3 dan dokumentasinya, lihatMenghubungkan ke Amazon Athena dengan ODBC.

Fitur hasil streaming tersedia dengan versi baru driver ODBC. Ini juga tersedia dengan driver JDBC. Untuk informasi tentang hasil streaming, lihat Panduan Instalasi dan Konfigurasi Driver ODBC, dan cari UseResultsetStreaming.

ODBC driver versi 1.0.3 adalah drop-in pengganti untuk versi sebelumnya dari driver. Kami merekomendasikan bahwa Anda bermigrasi ke driver saat ini.

> **Important**
>
> Untuk menggunakan driver ODBC versi 1.0.3, ikuti persyaratan berikut:
>
> - Jauhkan port 444 terbuka untuk lalu lintas keluar.
> - Tambahkan`athena:GetQueryResultsStream`kebijakan tindakan untuk daftar kebijakan untuk Athena. Tindakan kebijakan ini tidak terkena langsung dengan API dan hanya digunakan dengan driver ODBC dan JDBC, sebagai bagian dari dukungan hasil streaming. Untuk contoh kebijakan, lihat AWS kebijakan terkelola: AWSQuicksightAthenaAccess.

23 Agustus 2018

Publikasikan23/08/2018

Menambahkan dukungan untuk fitur terkait DDL ini dan memperbaiki beberapa bug, sebagai berikut:

- Dukungan tambahan untuk`BINARY`dan`DATE`tipe data untuk data di Parquet, dan untuk`DATE`dan`TIMESTAMP`tipe data untuk data di Avro.
- Dukungan tambahan untuk`INT`dan`DOUBLE`dalam kueri DDL.`INTEGER`adalah alias untuk`INT`, dan`DOUBLE PRECISION`adalah alias untuk`DOUBLE`.
- Peningkatan performa`DROP TABLE`dan`DROP DATABASE`kueri.
- Dihapus pembuatan`_$folder$`objek di Amazon S3 saat data bucket kosong.
- Memperbaiki masalah tempat`ALTER TABLE ADD PARTITION`melemparkan kesalahan saat tidak ada nilai partisi yang disediakan.
- Memperbaiki masalah tempat`DROP TABLE`mengabaikan nama basisdata saat memeriksa partisi setelah nama yang memenuhi syarat telah ditentukan dalam pernyataan.

Untuk selengkapnya tentang tipe data yang didukung di Athena, lihat. Tipe data di Amazon Athena

Untuk informasi tentang pemetaan tipe data yang didukung antara jenis di Athena, driver JDBC, dan tipe data Java, lihat“Jenis data”Bagian di bagianJDBC Driver Instalasi dan Panduan Konfigurasi.

16 Agustus 2018

Publikasikan2018-08-16

Merilis driver JDBC versi 2.0.5. Versi baru dari driver JDBC aliran hasil secara default, bukan paging melalui mereka, memungkinkan alat intelijen bisnis untuk mengambil set data yang besar lebih cepat. Dibandingkan dengan versi sebelumnya dari driver JDBC, ada perbaikan performa sebagai berikut:

- Sekitar 2x peningkatan performa saat mengambil kurang dari 10K baris.
- Sekitar 5-6x peningkatan performa saat mengambil lebih dari 10K baris.

Fitur hasil streaming hanya tersedia dengan driver JDBC. Ini tidak tersedia dengan driver ODBC. Anda tidak dapat menggunakannya dengan API Athena. Untuk informasi tentang hasil streaming, lihat Panduan Instalasi dan Konfigurasi Driver JDBC, dan cari. UseResultsetStreaming

Untuk mengunduh driver JDBC versi 2.0.5 dan dokumentasinya, lihatMenghubungkan ke Amazon Athena dengan JDBC.

Versi driver JDBC 2.0.5 adalah pengganti drop-in untuk versi driver sebelumnya (2.0.2). Untuk memastikan bahwa Anda dapat menggunakan driver JDBC versi 2.0.5,

menambahkan`athena:GetQueryResultsStream`kebijakan tindakan untuk daftar kebijakan untuk Athena. Tindakan kebijakan ini tidak terkena langsung dengan API dan hanya digunakan dengan driver JDBC, sebagai bagian dari dukungan hasil streaming. Untuk contoh kebijakan, lihat AWS kebijakan terkelola: AWSQuicksightAthenaAccess. Untuk informasi selengkapnya tentang migrasi dari versi 2.0.2 ke versi 2.0.5 driver, lihat Panduan Migrasi Driver JDBC.

Jika Anda bermigrasi dari driver 1.x ke driver 2.x, Anda akan perlu untuk bermigrasi konfigurasi yang ada ke konfigurasi baru. Kami sangat menyarankan Anda bermigrasi ke versi driver saat ini. Untuk informasi selengkapnya, lihat Panduan Migrasi Driver JDBC.

7 Agustus 2018

Publikasikan2018-08-07

Anda sekarang dapat menyimpan Amazon Virtual Private Cloud aliran log langsung di Amazon S3 dalam format GZIP, tempat Anda dapat mengkueri mereka di Athena. Untuk informasi, lihatMenanyakan log aliran VPC AmazondanAmazon VPC Flow Log sekarang dapat dikirim ke S3.

5 Juni 2018

Publikasikan2018-06-05

Topik

- Support untuk Views
- Perbaikan dan Pembaruan Pesan kesalahan
- Perbaikan Bug

Support untuk Views

Dukungan tambahan untuk IAM. Sekarang Anda dapat menggunakanCREATE VIEW,DESCRIBE VIEW,DROP VIEW,SHOW CREATE VIEW, danSHOW VIEWSdi Athena. Kueri yang mendefinisikan tampilan berjalan setiap kali Anda referensi tampilan dalam kueri Anda. Untuk informasi selengkapnya, lihat Bekerja dengan pandangan.

Perbaikan dan Pembaruan Pesan kesalahan

- Termasuk pustaka GSON 2.8.0 ke dalam CloudTrail SerDe, untuk menyelesaikan masalah dengan CloudTrail SerDe dan mengaktifkan penguraian string JSON.

- Peningkatan validasi skema partisi di Athena untuk Parquet, dan, dalam beberapa kasus, untuk ORC, dengan memungkinkan penataan ulang kolom. Hal ini memungkinkan Athena untuk lebih baik menangani perubahan dalam evolusi skema dari waktu ke waktu, dan dengan tabel yang ditambahkan oleh Crawler. AWS Glue Untuk informasi selengkapnya, lihat Menangani pembaruan skema.
- Menambahkan dukungan penguraian untuk`SHOW VIEWS`.
- Membuat perbaikan berikut untuk pesan kesalahan yang paling umum:
  - Mengganti pesan Kesalahan Internal dengan pesan kesalahan deskriptif ketika SerDe gagal mengurai kolom dalam kueri Athena. Sebelumnya, Athena mengeluarkan kesalahan internal dalam kasus kesalahan penguraian. Pesan kesalahan baru berbunyi:“HIVE_BAD_DATA: Kesalahan parsing nilai bidang untuk field 0: java.lang.String tidak dapat dilemparkan ke org.openx.data.JsonSerde.JSON.JSONObject”.
  - Peningkatan pesan kesalahan tentang izin tidak cukup dengan menambahkan lebih detail.

Perbaikan Bug

Diperbaiki bug berikut:

- Memperbaiki masalah yang memungkinkan penerjemahan internal `REAL` ke tipe data `FLOAT`. Ini meningkatkan integrasi dengan crawler AWS Glue yang menghasilkan tipe data `FLOAT`.
- Memperbaiki masalah saat Athena tidak mengonversi AVRO `DECIMAL` (tipe logis) ke tipe `DECIMAL`.
- Memperbaiki masalah tempat Athena tidak mengembalikan hasil untuk pertanyaan pada data Parquet dengan`WHERE`klausul yang direferensikan nilai-nilai dalam`TIMESTAMP`Jenis data.

17 Mei 2018

Dipublikasikan pada 2018-05-17

Peningkatan kuota konkurensi kueri di Athena dari lima menjadi dua puluh. Ini berarti bahwa Anda dapat mengirimkan dan menjalankan hingga dua puluh kueri `DDL` dan dua puluh kueri `SELECT` pada suatu waktu. Perhatikan bahwa kuota konkurensi terpisah untuk kueri `DDL`dan `SELECT`.

Kuota konkurensi di Athena didefinisikan sebagai jumlah pertanyaan yang dapat diserahkan ke layanan secara bersamaan. Anda dapat mengirimkan hingga dua puluh kueri dari tipe yang sama

(DDL atau SELECT) pada suatu waktu. Jika Anda mengirimkan kueri yang melebihi kuota kueri bersamaan, API Athena menampilkan pesan kesalahan.

Setelah Anda mengirimkan pertanyaan Anda ke Athena, itu memproses kueri dengan menetapkan sumber daya berdasarkan beban layanan keseluruhan dan jumlah kueri yang masuk. Kami terus memantau dan melakukan penyesuaian terhadap layanan sehingga proses kueri Anda secepat mungkin.

Untuk informasi, lihat Service Quotas. Ini adalah kuota yang dapat disesuaikan. Anda dapat menggunakan konsol Service Quotas untuk meminta penambahan kuota untuk kuota yang dapat disesuaikan.

19 April 2018

Dipublikasikan pada 2018-04-19

Merilis versi baru dari driver JDBC (versi 2.0.2) dengan dukungan untuk mengembalikan`ResultSet`data sebagai tipe data Larik, perbaikan, dan perbaikan bug. Untuk detailnya, lihat Catatan Rilis untuk pengemudi.

Untuk informasi tentang mengunduh driver JDBC baru versi 2.0.2 dan dokumentasi, lihatMenghubungkan ke Amazon Athena dengan JDBC.

Versi terbaru dari driver JDBC adalah 2.0.2. Jika Anda bermigrasi dari driver 1.x ke driver 2.x, Anda akan perlu untuk bermigrasi konfigurasi yang ada ke konfigurasi baru. Kami sangat menyarankan Anda bermigrasi ke driver saat ini.

Untuk informasi tentang perubahan yang diperkenalkan di versi baru driver, versi perbedaan, dan contoh, lihatPanduan Migrasi Driver JDBC.

6 April 2018

Dipublikasikan pada 2018-04-06

Gunakan lengkapi otomatis untuk mengetik kueri di konsol Athena.

15 Maret 2018

Publikasikan2018-03-15

Ditambahkan kemampuan untuk secara otomatis membuat tabel Athena untuk file CloudTrail log langsung dari konsol. CloudTrail Untuk informasi, lihat Menggunakan CloudTrail konsol untuk membuat tabel Athena untuk log CloudTrail .

2 Februari 2018

Publikasikan2018-02-12

Menambahkan kemampuan untuk aman offload menengah data ke disk untuk memori-intensif kueri yang menggunakan`GROUP BY`klausul. Ini meningkatkan keandalan kueri tersebut, mencegah“Sumber daya kueri habis”kesalahan.

19 Januari 2018

Publikasikan2018-01-19

Athena menggunakan Presto, mesin kueri terdistribusi open-source, untuk menjalankan kueri.

Dengan Athena, tidak ada versi untuk dikelola. Kami telah transparan mutakhirkan mesin yang mendasari di Athena untuk versi berdasarkan Presto versi 0.172. Tidak ada tindakan yang diperlukan pada akhir Anda.

Dengan mutakhirkan, Anda sekarang dapat menggunakanPresto 0.172 Fungsi dan Operator, termasukPresto 0.172 Lambda Ekspresidi Athena.

Pembaruan utama untuk rilis ini, termasuk perbaikan yang disumbangkan masyarakat, meliputi:

- Support untuk mengabaikan header. Anda dapat menggunakan`skip.header.line.count`properti saat mendefinisikan tabel, untuk memungkinkan Athena mengabaikan header. Ini didukung untuk kueri yang menggunakan dan SerDeOpenCSV, LazySimpleSerDedan bukan untuk Grok atau Regex. SerDes
- Support untuk`CHAR(n)`Jenis data`STRING`fungsi. Rentang untuk`CHAR(n)`adalah`[1.255]`, sedangkan rentang untuk`VARCHAR(n)`adalah`[1,65535]`.
- Support untuk subqueries berkorelasi.
- Support untuk ekspresi Presto Lambda dan fungsi.
- Peningkatan performa`DECIMAL`jenis dan operator.
- Support untuk agregasi terfilter, seperti`SELECT sum(col_name) FILTER`, tempat`id > 0`.
- Predikat Push-down untuk`DECIMAL`,`TINYINT`,`SMALLINT`, dan`REAL`Jenis data.

- Support untuk predikat perbandingan terkuantifikasi:`ALL`,`ANY`, dan`SOME`.
- Fungsi yang ditambahkan:`arrays_overlap()`,`array_except()`,`levenshtein_distance()`,`codepoint()`,`skew` dan`typeof()`.
- Menambahkan varian dari`from_unixtime()`fungsi yang mengambil argumen zona waktu.
- Menambahkan`bitwise_and_agg()`dan`bitwise_or_agg()`Fungsi agregasi.
- Menambahkan`xxhash64()`dan`to_big_endian_64()`fungsi.
- Menambahkan dukungan untuk melarikan diri tanda kutip ganda atau garis miring terbalik menggunakan garis miring terbalik dengan jalur JSON subscript ke`json_extract()`dan`json_extract_scalar()`fungsi. Ini mengubah semantik doa apapun menggunakan backslash, sebagai backslash sebelumnya diperlakukan sebagai karakter normal.

Untuk informasi selengkapnya tentang fungsi dan operator, lihat Kueri, fungsi, dan operator DML di panduan ini, dan Fungsi dan operator dalam dokumentasi Presto.

Athena tidak mendukung semua fitur Presto ini. Untuk informasi selengkapnya, lihat Batas.

Catatan rilis Athena untuk 2017

13 November 2017

Publikasikan2017-11-13

Menambahkan dukungan untuk menghubungkan Athena ke Driver ODBC. Untuk informasi, lihat Menghubungkan ke Amazon Athena dengan ODBC.

1 November 2017

Publikasikan2017-11-01

Menambahkan dukungan untuk mengkueri data geospasial, dan untuk wilayah Asia Pacific (Seoul), Asia Pacific (Mumbai), dan EU (London). Untuk informasi, lihat Menanyakan data geospasial dan Wilayah AWS dan Titik Akhir.

19 Oktober 2017

Publikasikan2017-10-19

Menambahkan dukungan untuk EU (Frankfurt). Untuk daftar wilayah yang didukung, lihat Wilayah AWS dan Titik Akhir.

3 Oktober 2017

Dipublikasikan pada 2017-10-03

Buat kueri bernama Athena dengan. AWS CloudFormation Untuk informasi selengkapnya, lihat AWS::Athena::NamedQuerydi Panduan AWS CloudFormation Pengguna.

25 September 2017

Publikasikan2017-09-25

Dukungan tambahan untuk Asia Pacific (Sydney). Untuk daftar wilayah yang didukung, lihat Wilayah AWS dan Titik Akhir.

Parquet

Publikasikan2017-08-14

Menambahkan integrasi dengan AWS Glue Data Catalog dan wizard migrasi untuk memperbarui dari katalog data terkelola Athena ke. AWS Glue Data Catalog Untuk informasi selengkapnya, lihat Integrasi dengan AWS Glue.

Parquet

Publikasikan2017-08-04

Menambahkan dukungan untuk Grok SerDe, yang menyediakan pencocokan pola yang lebih mudah untuk catatan dalam file teks tidak terstruktur seperti log. Untuk informasi selengkapnya, lihat Grok SerDe. Menambahkan pintasan keyboard untuk menggulir riwayat kueri menggunakan konsol (CTRL + /menggunakan Windows, CMD + /menggunakan Mac).

22 Juni 2017

Publikasikan2017-06-22

Menambahkan dukungan untuk Asia Pacific (Tokyo) dan Asia Pacific (Singapore). Untuk daftar wilayah yang didukung, lihat Wilayah AWS dan Titik Akhir.

8 Juni 2017

Dipublikasikan pada 2017-06-08

Menambahkan dukungan untuk Eropa (Irlandia). Untuk informasi lebih lanjut, lihat Wilayah AWS dan Titik Akhir.

19 Mei 2017

Publikasikan2017-05-19

Menambahkan API Amazon Athena dan AWS CLI dukungan untuk Athena; driver JDBC diperbarui ke versi 1.1.0; memperbaiki berbagai masalah.

- Amazon Athena memungkinkan pemrograman aplikasi untuk Athena. Untuk informasi selengkapnya, lihat Referensi API Amazon Athena. AWS SDK terbaru mencakup dukungan untuk Athena API. Untuk tautan ke dokumentasi dan unduhan, lihat bagian SDK dalam Alat untuk Amazon Web Services.
- AWS CLI Termasuk perintah baru untuk Athena. Untuk informasi selengkapnya, lihat Referensi API Amazon Athena.
- Sebuah driver JDBC baru 1.1.0 tersedia, yang mendukung Athena API baru serta fitur terbaru dan perbaikan bug. Unduh driver di https://downloads.athena.us-east-1.amazonaws.com/drivers/AthenaJDBC41-1.1.0.jar. Kami merekomendasikan mutakhirkan ke driver Athena JDBC terbaru; Namun, Anda masih dapat menggunakan versi driver sebelumnya. Versi driver sebelumnya tidak mendukung Athena API. Untuk informasi selengkapnya, lihat Menghubungkan ke Amazon Athena dengan JDBC.
- Tindakan khusus untuk pernyataan kebijakan di versi sebelumnya Athena telah diusangkan. Jika Anda memutakhirkan ke driver JDBC versi 1.1.0 dan memiliki pelanggan dikelola atau inline kebijakan IAM melekat pada pengguna JDBC, Anda harus memperbarui kebijakan IAM. Sebaliknya, versi sebelumnya dari driver JDBC tidak mendukung API Athena, sehingga Anda dapat menentukan hanya tindakan usang dalam kebijakan yang melekat pada versi sebelumnya pengguna JDBC. Untuk alasan ini, Anda tidak perlu memperbarui kebijakan IAM yang dikelola pelanggan atau inline.
- Tindakan khusus kebijakan ini digunakan di Athena sebelum rilis API Athena. Gunakan tindakan yang diusangkan ini dalam kebijakanhanyadengan driver JDBC lebih awal dari versi 1.1.0. Jika Anda memutakhirkan driver JDBC, mengganti pernyataan kebijakan yang memungkinkan atau menolak tindakan usang dengan tindakan API yang sesuai seperti yang tercantum atau kesalahan akan terjadi:

Tindakan Spesifik Kebijakan Usang	Aksi API Athena yang sesuai
`athena:RunQuery`	`athena:StartQueryExecution`
`athena:CancelQueryExecution`	`athena:StopQueryExecution`
`athena:GetQueryExecutions`	`athena:ListQueryExecutions`

Perbaikan

- Peningkatan batas panjang string kueri untuk 256 KB.

Perbaikan Bug

- Memperbaiki masalah yang menyebabkan hasil kueri terlihat cacat saat menggulir hasil di konsol.
- Memperbaiki masalah saat`\u0000`karakter string dalam file data Amazon S3 akan menyebabkan kesalahan.
- Memperbaiki masalah yang menyebabkan kueri untuk membatalkan kueri yang dibuat melalui driver JDBC gagal.
- Memperbaiki masalah yang menyebabkan data Amazon S3 gagal di AS Timur (Ohio). AWS CloudTrail SerDe
- Memperbaiki masalah yang menyebabkan `DROP TABLE` gagal pada tabel dipartisi.

4 April 2017

Dipublikasikan pada 2017-04-04

Menambahkan dukungan untuk enkripsi data Amazon S3 dan merilis update driver JDBC (versi 1.0.1) dengan dukungan enkripsi, peningkatan, dan perbaikan bug.

Fitur

- Menambahkan fitur enkripsi berikut:
  - Support untuk mengkueri data terenkripsi di Amazon S3.

  - Support untuk mengenkripsi hasil kueri Athena.
- Versi baru driver mendukung fitur enkripsi baru, menambahkan perbaikan, dan memperbaiki masalah.
- Menambahkan kemampuan untuk menambah, mengganti, dan mengubah kolom menggunakan`ALTER TABLE`. Untuk informasi selengkapnya, lihat Pemetaan kolom dalam dokumentasi SQL Server.
- Menambahkan dukungan untuk kueri data terkompresi LZO.

Untuk informasi selengkapnya, lihat Enkripsi diam.

Perbaikan

- performa kueri JDBC yang lebih baik dengan perbaikan halaman-ukuran, kembali 1.000 baris bukan 100.
- Menambahkan kemampuan untuk membatalkan kueri menggunakan antarmuka driver JDBC.
- Menambahkan kemampuan untuk menentukan pilihan JDBC di URL koneksi JDBC. Lihat Menghubungkan ke Amazon Athena dengan JDBC driver JDBC terbaru.
- Ditambahkan pengaturan PROXY di driver, yang sekarang dapat diatur menggunakan ClientConfigurationdalam AWS SDK for Java.

Perbaikan Bug

Diperbaiki bug berikut:

- Kesalahan throttling akan terjadi saat beberapa kueri dikeluarkan menggunakan antarmuka driver JDBC.
- Driver JDBC akan berhenti saat memproyeksikan tipe data desimal.
- Driver JDBC akan kembali setiap jenis data sebagai string, terlepas dari bagaimana tipe data didefinisikan dalam tabel. Sebagai contoh, memilih kolom didefinisikan sebagai`INT`Jenis data menggunakan`resultSet.GetObject()`akan mengembalikan`STRING`Jenis data bukan`INT`.
- Driver JDBC akan memverifikasi mandat pada saat koneksi dibuat, bukan pada saat kueri akan berjalan.
- Pertanyaan yang dilakukan melalui driver JDBC akan gagal saat skema ditentukan bersama dengan URL.

24 Maret 2017

Publikasikan2017-03-24

Ditambahkan AWS CloudTrail SerDe, peningkatan kinerja, masalah partisi tetap.

Fitur

- Menambahkan AWS CloudTrail SerDe, yang sejak itu telah digantikan oleh log Sarang JSON SerDe untuk membaca CloudTrail . Untuk informasi tentang menanyakan CloudTrail log, lihatMeminta log AWS CloudTrail.

Perbaikan

- Peningkatan performa saat memindai sejumlah besar partisi.
- Peningkatan performa pada`MSCK Repair Table`operasi.
- Menambahkan kemampuan untuk kueri data Amazon S3 yang disimpan di wilayah selain Wilayah utama Anda. Tarif transfer data antar-wilayah standar untuk Amazon S3 berlaku selain biaya Athena standar.

Perbaikan Bug

- Perbaikan bug dengan “kesalahan tabel tidak ditemukan” dapat terjadi jika tidak ada partisi yang dimuat.
- Perbaikan bug untuk menghindari memberikan pengecualian dengan kueri `ALTER TABLE ADD PARTITION IF NOT EXISTS`.
- Perbaikan bug di `DROP PARTITIONS`.

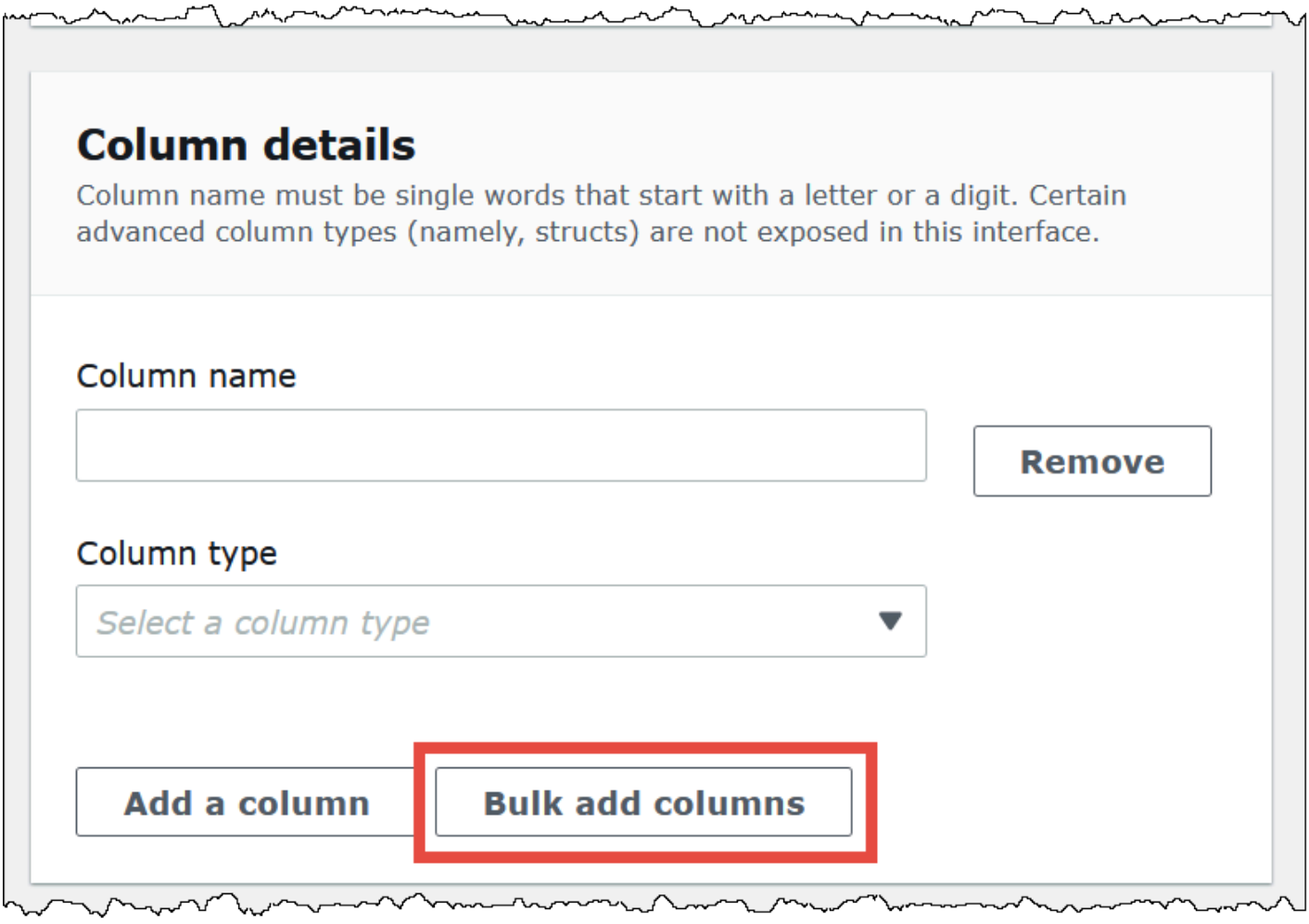
20 Februari 2017

Publikasikan2017-02-20

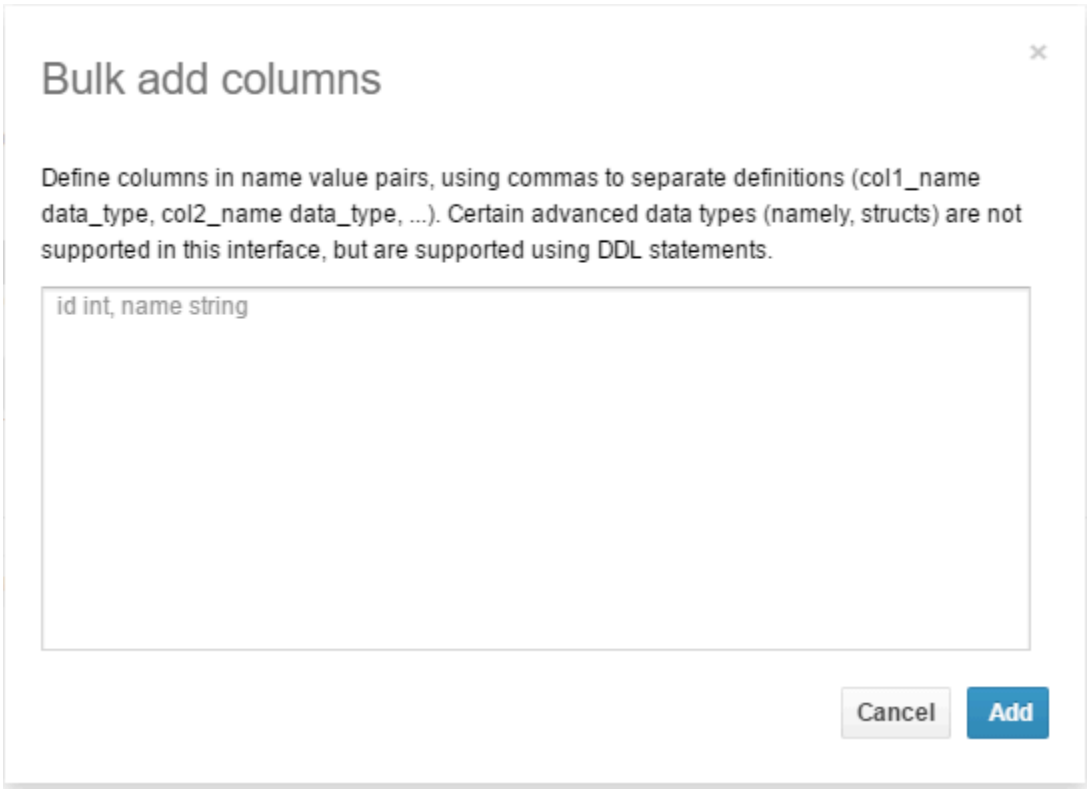
Menambahkan dukungan untuk AvroSerDe dan SerDe OpenCSV, Wilayah AS Timur (Ohio), dan kolom pengeditan massal di wizard konsol. Peningkatan performa pada tabel Parquet besar.

Fitur

- Memperkenalkan dukungan untuk yang baru SerDes:
  - AvroSerDe
  - OpenCSV untuk SerDe memproses CSV
- Peluncuran Wilayah US East (Ohio) (us-east-2) Sekarang Anda dapat menjalankan kueri di wilayah ini.
- Anda sekarang dapat menggunakan formulir data bucket Create Table From S3 untuk menentukan skema tabel secara massal. Di editor kueri, pilih Buat, data bucket S3, lalu pilih Tambahkan kolom massal di bagian Rincian kolom.



Ketik pasangan nilai nama dalam kotak teks dan pilihTambahkan.



Perbaikan

- Peningkatan performa pada tabel Parquet besar.

Riwayat dokumen

Pembaruan dokumentasi terbaru: 28 Juni 2024.

Kami juga rutin memperbarui dokumentasi untuk menjawab umpan balik yang Anda kirimkan kepada kami. Tabel berikut menjelaskan tambahan penting pada dokumentasi Amazon Athena. Tidak semua pembaruan diwakili.

Perubahan	Deskripsi	Tanggal rilis
Kebijakan `AmazonAthenaFullAccess` terkelola yang diperbarui.	Ditambahkan `glue:GetCatalogImportStatus` ke `BaseGluePermissions` bagian kebijakan AmazonAthenaFullAccessterkelola. Tindakan tambahan memungkinkan Athena menggunakan AWS Glue API yang didokumentasikan secara publik untuk mengambil status impor katalog.	Juni 18, 2024
Kebijakan `AmazonAthenaFullAccess` terkelola yang diperbarui.	`datazone:ListAccountEnvironments` Izin `datazone:ListDomains` `datazone:ListProjects` ,, dan telah ditambahkan ke kebijakan AmazonAthenaFullAccessterkelola. Tindakan yang ditambahkan memungkinkan pengguna Athena untuk bekerja dengan DataZone domain, proyek, dan lingkungan Amazon. Untuk informasi selengkapnya, lihat Menggunakan Amazon DataZone di Athena.	Januari 3, 2024
Kebijakan `AmazonAthenaFullAccess` terkelola yang diperbarui.	Ditambahkan`glue:StartColumnStatisticsTaskRun` ,`glue:GetColumnStatisticsTaskRun` , dan `glue:GetColumnStatisticsTaskRuns` izin ke kebijakan AmazonAthenaFullAccessterkelola . Tindakan yang ditambahkan memungkinkan Athena menelepon untuk mengambil statistik AWS Glue untuk fitur pengoptimal berbasis biaya. Untuk informasi selengkapnya, lihat Menggunakan pengoptimal berbasis biaya.	Januari 3, 2024

Perubahan	Deskripsi	Tanggal rilis
Dokumentasi yang ditambahkan untuk Pusat Identitas IAM mengaktifkan kelompok kerja Athena.	Anda dapat membuat workgroup Athena SQL yang menggunakan mode autentikasi IAM Identity Center. Kelompok kerja ini mendukung penggunaan identitas yang sama di seluruh AWS layanan seperti Amazon Athena dan Amazon EMR Studio. Untuk informasi selengkapnya, lihat Menggunakan IAM Identity Center mengaktifkan kelompok kerja Athena.	5 Desember 2023
Menambahkan dokumentasi untuk menanyakan data S3 Express One Zone	Anda dapat menggunakan Athena untuk menanyakan data di kelas penyimpanan Amazon S3 Express One Zone. Untuk informasi selengkapnya, lihat Meminta data S3 Express One Zone.	28 November 2023
Ditambahkan dokumentasi untuk tampilan Glue Data Catalog.	Anda dapat menggunakan tampilan Glue Data Catalog untuk memberikan satu tampilan umum di seluruh AWS layanan seperti Amazon Athena dan Amazon Redshift. Untuk informasi selengkapnya, lihat Menggunakan AWS Glue Data Catalog tampilan.	27 November 2023
Menambahkan dokumentasi untuk fitur pengoptimal berbasis biaya.	Anda dapat menggunakan statistik dari AWS Glue untuk mengoptimalkan kueri Anda di Athena SQL. Untuk informasi selengkapnya, lihat Menggunakan pengoptimal berbasis biaya.	17 November 2023
Menambahkan dokumentasi untuk driver Athena JDBC 3.x	Anda dapat menggunakan driver Athena JDBC 3.x untuk membaca hasil kueri langsung dari Amazon S3. Driver JDBC 3.x mendukung hampir semua metode otentikasi yang didukung oleh driver JDBC 2.x. Untuk informasi selengkapnya, lihat Athena JDBC 3.x driver.	16 November 2023
Ditambahkan dokumentasi untuk digunakan DataZone di Athena.	Anda dapat menggunakannya DataZone untuk menyederhanakan pengalaman Anda di seluruh layanan AWS analitik seperti Athena AWS Glue,, dan Lake Formation. Untuk informasi selengkapnya, lihat Menggunakan Amazon DataZone di Athena.	4 Oktober 2023

Perubahan	Deskripsi	Tanggal rilis
Menambahkan dokumentasi untuk pemesanan kapasitas.	Sekarang Anda dapat menggunakan reservasi kapasitas di Amazon Athena untuk menjalankan kueri SQL pada kapasitas komputasi yang dikelola sepenuhnya. Untuk informasi selengkapnya, lihat Mengelola kapasitas pemrosesan kueri.	28 April 2023
Ditambahkan dokumentasi untuk query pandangan federasi.	Anda sekarang dapat membuat dan menanyakan tampilan pada sumber data federasi di Athena. Untuk informasi selengkapnya, lihat Menanyakan pandangan federasi.	4 April 2023
Menambahkan dokumentasi tentang mencegah pelambatan di Amazon S3.	Untuk informasi selengkapnya, lihat Mencegah pelambatan Amazon S3.	24 Maret 2023
Kebijakan `AmazonAthenaFullAccess` terkelola yang diperbarui.	Ditambahkan `pricing:GetProducts` ke kebijakan AmazonAthenaFullAccessterkelola. Tindakan yang ditambahkan menyediakan akses ke AWS Billing and Cost Management. Untuk informasi selengkapnya, lihat GetProductsdi Referensi AWS Billing and Cost Management API.	Januari 25, 2023
Dokumentasi yang diperluas untuk dukungan kompresi Athena.	Topik individu ditambahkan untukKompresi meja sarang,Kompresi tabel gunung es, danTingkat kompresi ZSTD. Untuk informasi selengkapnya, lihat Dukungan kompresi Athena.	20 Januari 2023
Ditambahkan dokumentasi untuk Amazon Athena untuk Apache Spark.	Anda sekarang dapat secara interaktif membuat dan menjalankan aplikasi Apache Spark dan notebook yang kompatibel dengan Jupyter di Amazon Athena. Untuk informasi selengkapnya, lihat Menggunakan Apache Spark di Amazon Athena.	30 November 2022

Perubahan	Deskripsi	Tanggal rilis
Menambahkan dokumentasi untuk konektor Athena IBM Db2.	Anda dapat menggunakan konektor Amazon Athena untuk IBM Db2 untuk meminta Db2 dari Athena. Untuk informasi selengkapnya, lihat Konektor Amazon Athena IBM Db2	18 November 2022
Menambahkan dokumentasi untuk penggunaan kembali hasil kueri.	Saat Anda menjalankan ulang kueri di Athena, Anda sekarang dapat memilih untuk menggunakan kembali hasil kueri terakhir yang disimpan. Ini dapat meningkatkan kinerja dan mengurangi biaya dalam hal jumlah byte yang dipindai. Untuk informasi selengkapnya, lihat Menggunakan kembali hasil kueri.	8 November 2022
Diperbarui dokumentasi untuk CloudTrail log.	`CREATE TABLE` DDL untuk menanyakan CloudTrail log telah diperbarui untuk menggunakan JSON SerDe alih-alih file. CloudTrail SerDe Untuk informasi selengkapnya, lihat Meminta log AWS CloudTrail.	November 3, 2022
Ditambahkan dokumentasi untuk mesin Athena versi 3.	Untuk informasi lebih lanjut tentang mesin Athena versi 3, lihat. Mesin Athena versi 3	13 Oktober 2022
Ditambahkan tutorial tentang mengkonfigurasi SSO untuk ODBC menggunakan plugin Okta.	Konfigurasikan driver ODBC Amazon Athena dan plugin Okta untuk kemampuan masuk tunggal (SSO) menggunakan penyedia identitas Okta. Untuk informasi selengkapnya, lihat Mengkonfigurasi SSO untuk ODBC menggunakan plugin Okta dan Penyedia Identitas Okta.	23 Agustus 2022
Menambahkan dokumentasi untuk melihat rencana kueri dan statistik di konsol Athena.	Anda dapat menggunakan editor kueri Athena untuk melihat representasi grafis tentang bagaimana kueri Anda akan dijalankan dan grafik, detail, dan statistik tentang bagaimana kueri selesai dijalankan. Untuk informasi selengkapnya, lihat Melihat rencana eksekusi untuk kueri SQL dan Melihat statistik dan detail eksekusi untuk kueri yang diselesaikan.	21 Juli 2022

Perubahan	Deskripsi	Tanggal rilis
Ditambahkan dokumentasi untuk menanyakan tampilan Apache Hive di metastores Hive eksternal.	Anda dapat menggunakan Athena untuk menanyakan tampilan Apache yang dibuat di metastores Hive eksternal. Beberapa fungsi Hive tidak didukung atau memerlukan penanganan khusus. Untuk informasi selengkapnya, lihat Bekerja dengan tampilan Hive.	22 April 2022
Ditambahkan dokumentasi untuk kueri yang disimpan.	Anda dapat menggunakan fitur kueri tersimpan di Athena untuk menyimpan, mengingat, mengedit, dan mengganti nama kueri Anda. Untuk informasi selengkapnya, lihat Menggunakan kueri yang disimpan di panduan ini dan UpdateNamedQuerydi Referensi API Amazon Athena.	28 Februari 2022
Ditambahkan dokumentasi pratinjau untuk dukungan Apache Iceberg.	Athena mendukung membaca, perjalanan waktu, dan menulis kueri untuk tabel Apache Iceberg yang menggunakan format Apache Parquet untuk data dan katalog untuk metastore mereka. AWS Glue Untuk informasi selengkapnya, lihat Menggunakan tabel Apache Iceberg.	26 November 2021
Menambahkan dokumentasi untuk kueri federasi lintas akun.	Anda dapat menggunakan fitur kueri federasi lintas akun untuk menanyakan sumber data di akun lain. Untuk informasi tentang mengatur izin untuk mengaktifkan fitur ini, lihatMengaktifkan kueri federasi lintas akun.	12 November 2021
Ditambahkan dokumentasi untuk pernyataan Athena`UNLOAD`.	Gunakan `UNLOAD` pernyataan untuk menulis kueri hasil dari `SELECT` pernyataan ke format Apache Parquet, ORC, Apache Avro, dan JSON. Untuk informasi selengkapnya, lihat MEMBONGKAR.	5 Agustus 2021
Menambahkan dokumentasi untuk Athena`EXPLAIN`.	Untuk informasi selengkapnya, lihat Menggunakan JELASKAN dan JELASKAN ANALISIS di Athena dan Memahami Athena MENJELASKAN hasil pernyataan.	5 April 2021

Perubahan	Deskripsi	Tanggal rilis
Menambahkan halaman tentang pemecahan masalah dan penyetelan performa di Athena.	Untuk informasi selengkapnya, lihat Pemecahan Masalah di Athena dan Tuning kinerja di Athena.	30 Desember 2020
Menambahkan dokumentasi untuk Athena versioning mesin dan Athena mesin versi 2.	Untuk informasi selengkapnya, lihat Pembuatan versi mesin Athena.	11 November 2020
Diperbarui kueri gabungan dokumentasi untuk rilis ketersediaan umum.	Untuk informasi selengkapnya, lihat Menggunakan Amazon Athena dan Menggunakan Athena dengan tombol konteks CalledVia .	11 November 2020
Menambahkan dokumentasi untuk menggunakan driver JDBC dengan Lake Formation untuk akses Gabungan ke Athena.	Untuk informasi selengkapnya, lihat Menggunakan Lake Formation dan Athena JDBC dan ODBC driver untuk akses federasi ke Athena dan Tutorial: Mengkonfigurasi akses federasi untuk pengguna Okta ke Athena menggunakan Lake Formation dan JDBC.	25 September 2020
Menambahkan dokumentasi untuk konektor OpenSearch data Amazon Athena.	Untuk informasi selengkapnya, lihat Konektor Amazon Athena OpenSearch .	21 Juli 2020

Perubahan	Deskripsi	Tanggal rilis
Menambahkan dokumentasi untuk mengkueri set data Hudi.	Untuk informasi selengkapnya, lihat Menggunakan Athena untuk menanyakan kumpulan data Apache Hudi.	9 Juli 2020
Menambahkan dokumentasi pada kueri log server web Apache dan log server web IIS yang disimpan di Amazon S3.	Untuk informasi selengkapnya, lihat Menanyakan log Apache yang disimpan di Amazon S3 dan Menanyakan log server informasi internet (IIS) yang disimpan di Amazon S3.	8 Juli 2020
Menambahkan dokumentasi untuk rilis umum dari Athena Data Connector untuk Eksternal Hive Metastore.	Untuk informasi selengkapnya, lihat Menggunakan Athena Data Connector untuk Eksternal Hive Metastore.	1 Juni 2020
Menambahkan dokumentasi untuk menandai sumber daya katalog data.	Untuk informasi selengkapnya, lihat Menandai sumber daya Athena.	1 Juni 2020
Menambahkan dokumentasi pada proyeksi partisi.	Untuk informasi selengkapnya, lihat Proyeksi partisi dengan Amazon Athena.	21 Mei 2020
Memperbarui contoh kode Java untuk Athena.	Untuk informasi selengkapnya, lihat Sampel Kode.	11 Mei 2020

Perubahan	Deskripsi	Tanggal rilis
Menambahkan topik tentang menanyakan GuardDuty temuan Amazon.	Untuk informasi selengkapnya, lihat Menanyakan temuan Amazon GuardDuty .	19 Maret 2020
Menambahkan topik tentang menggunakan CloudWatch Acara untuk memantau transisi status kueri Athena.	Untuk informasi selengkapnya, lihat Memantau kueri Athena dengan acara Amazon EventBridge.	11 Maret 2020
Menambahkan topik tentang menanyakan log AWS Global Accelerator aliran dengan Athena.	Untuk informasi selengkapnya, lihat Memerikan log AWS Global Accelerator aliran.	6 Februari 2020

Perubahan	Deskripsi	Tanggal rilis
• Menambahkan dokumentasi menggunakan CTAS dengan INSERT INTO untuk menambahkan data dari sumber unpartitioned untuk tujuan dipartisi. • Menambahkan tautan unduhan untuk versi pratinjau 1.1.0 driver ODBC untuk Athena. • Deskripsi untuk `SHOW DATABASES LIKE` regex dikoreksi. • Dikoreksi `partitioned_by` sintaks dalam topik CTA. • Perbaikan kecil lainnya.	Pembaruan dokumentasi mencakup, tetapi tidak terbatas pada, topik-topik berikut: • Menggunakan CTAS dan INSERT INTO untuk ETL dan analisis data • Menghubungkan ke Amazon Athena dengan ODBC(Fitur pratinjau 1.1.0 sekarang disertakan dalam driver ODBC 1.1.2.) • SHOW DATABASES • CREATE TABLE AS	4 Februari 2020

Perubahan	Deskripsi	Tanggal rilis
Menambahkan dokumentasi menggunakan CTAS dengan INSERT INTO untuk menambahkan data dari sumber dipartisi untuk tujuan dipartisi.	Untuk informasi selengkapnya, lihat Menggunakan CTAS dan INSERT INTO untuk bekerja di sekitar batas partisi 100.	22 Januari 2020
Kueri informasi lokasi hasil diperbarui.	Athena tidak lagi menciptakan 'default' hasil kueri lokasi. Untuk informasi selengkapnya, lihat Menentukan lokasi hasil query.	20 Januari 2020
Menambahkan topik tentang menanyakan. AWS Glue Data Catalog Informasi terbaru tentang kuota layanan (sebelumnya “batas layanan”) di Athena.	Untuk informasi selengkapnya, lihat topik berikut. • Mengkueri AWS Glue Data Catalog • Service Quotas	17 Januari 2020
Topik dikoreksi pada SerDe OpenCSV untuk dicatat bahwa `TIMESTAMP` jenis harus ditentukan dalam format numerik UNIX.	Untuk informasi selengkapnya, lihat OpenCSV untuk SerDe memproses CSV.	15 Januari 2020

Perubahan	Deskripsi	Tanggal rilis
Topik keamanan diperbarui pada enkripsi untuk dicatat bahwa Athena tidak mendukung kunci asimetris.	Athena hanya mendukung kunci simetris untuk membaca dan menulis data. Untuk informasi selengkapnya, lihat Opsi enkripsi Amazon S3 yang didukung.	8 Januari 2020
Menambahkan informasi tentang akses lintas akun ke bucket Amazon S3 yang dienkripsi dengan kunci khusus. AWS KMS	Untuk informasi selengkapnya, lihat Akses lintas akun ke bucket yang dienkripsi dengan kunci khusus AWS KMS.	13 Desember 2019
Menambahkan dokumentasi untuk mengkueri gabungan, metastores Hive eksternal, machine learning, dan fungsi yang ditetapkan pengguna. Menambahkan CloudWatch metrik baru.	Untuk informasi selengkapnya, lihat topik berikut. • Menggunakan Amazon Athena • Konektor sumber data yang tersedia • Menggunakan Athena Data Connector untuk Eksternal Hive Metastore • Menggunakan Machine Learning (ML) dengan Amazon Athena • Query dengan fungsi yang ditentukan pengguna • Daftar CloudWatch metrik dan dimensi untuk Athena	26 November 2019

Perubahan	Deskripsi	Tanggal rilis
Menambahkan bagian untuk baru INSERT INTO perintah dan diperbarui hasil kueri informasi lokasi untuk mendukung data manifest file.	Untuk informasi selengkapnya, lihat INSERT INTO dan Bekerja dengan hasil kueri, kueri terbaru, dan file keluaran.	18 September 2019
Ditambahkan bagian untuk antarmuka VPC endpoint () PrivateLink dukungan. Driver JDBC diperbarui. Informasi terbaru tentang log aliran VPC yang diperkaya.	Lihat informasi selengkapnya di Connect ke Amazon Athena menggunakan antarmuka VPC endpoint, Menanyakan log aliran VPC Amazon, dan Menghubungkan ke Amazon Athena dengan JDBC.	11 September 2019
Menambahkan bagian tentang mengintegrasikan dengan AWS Lake Formation.	Untuk informasi selengkapnya, lihat Menggunakan Athena untuk menanyakan data yang terdaftar AWS Lake Formation.	26 Juni 2019
Bagian keamanan diperbarui untuk konsistensi dengan AWS layanan.	Untuk informasi selengkapnya, lihat Keamanan Amazon Athena.	26 Juni 2019

Perubahan	Deskripsi	Tanggal rilis
Ditambahkan bagian pada query AWS WAF log.	Untuk informasi selengkapnya, lihat Meminta log AWS WAF.	31 Mei 2019
Merilis versi baru dari driver ODBC dengan dukungan untuk Athena workgroups.	Untuk mengunduh versi driver ODBC versi 1.0.5 dan dokumentasi, lihat Menghubungkan ke Amazon Athena dengan ODBC. Ada tidak ada perubahan ke string koneksi driver ODBC saat Anda menggunakan tanda pada grup kerja. Untuk menggunakan tanda, mutakhirkan ke versi terbaru dari driver ODBC, yang merupakan versi saat ini. Versi driver ini memungkinkan Anda menggunakanTindakan grup kerja API Athenauntuk membuat dan mengelola grup kerja, danTindakan tanda API Athenauntuk menambahkan, membuat daftar, atau menghapus tanda pada grup kerja. Sebelum memulai, pastikan Anda memiliki izin level sumber daya di IAM untuk tindakan pada grup kerja dan tanda.	5 Maret 2019
Menambahkan dukungan tanda untuk grup kerja di Amazon Athena.	Setiap tanda terdiri atas sebuah kunci dan sebuah nilai, yang keduanya Anda tentukan. Saat Anda tanda grup kerja, Anda menetapkan metadata kustom untuk itu. Sebagai contoh, membuat grup kerja untuk setiap pusat biaya. Kemudian, dengan menambahkan tanda ke grup kerja ini, Anda dapat melacak pengeluaran Athena Anda untuk setiap pusat biaya. Untuk informasi selengkapnya, lihat Menggunakan tag untuk penagihan di Panduan AWS Billing and Cost Management Pengguna.	22 Februari 2019

Perubahan	Deskripsi	Tanggal rilis
Peningkatan SerDe OpenX JSON yang digunakan di Athena.	Perbaikan meliputi, tetapi tidak terbatas pada, berikut: • Support untuk`ConvertDotsInJsonKeysToUnderscores` properti. Ketika diatur ke`TRUE`, ini memungkinkan SerDe untuk mengganti titik-titik dalam nama kunci dengan garis bawah. Sebagai contoh, jika set data JSON berisi kunci dengan nama`"a.b"`, Anda dapat menggunakan properti ini untuk menentukan nama kolom menjadi`"a_b"`di Athena. Defaultnya adalah `FALSE`. Secara default, Athena tidak mengizinkan titik-titik dalam nama kolom. • Support untuk`case.insensitive` properti. Secara default, Athena mengharuskan semua kunci dalam set data JSON Anda menggunakan huruf kecil. Menggunakan`WITH SERDE PROPERTIES ("case.insensitive"= FALSE;)` memungkinkan Anda untuk menggunakan nama kunci case-sensitive dalam data Anda. Defaultnya adalah `TRUE`. Ketika diatur ke`TRUE`, SerDe mengubah semua kolom huruf besar menjadi huruf kecil. Untuk informasi selengkapnya, lihat OpenX JSON SerDe.	18 Februari 2019

Perubahan	Deskripsi	Tanggal rilis
Menambahkan dukungan untuk grup kerja.	Menggunakan grup kerja untuk memisahkan pengguna, tim, aplikasi, atau beban kerja, dan untuk menetapkan batas pada jumlah data setiap permintaan atau seluruh grup kerja dapat memproses. Karena grup kerja bertindak sebagai sumber daya IAM, Anda dapat menggunakan izin level sumber daya untuk mengontrol akses ke grup kerja tertentu. Anda juga dapat melihat metrik terkait kueri di Amazon CloudWatch, mengontrol biaya kueri dengan mengonfigurasi batas jumlah data yang dipindai, membuat ambang batas, dan memicu tindakan, seperti alarm Amazon SNS, saat ambang batas ini dilanggar. Untuk informasi selengkapnya, lihat Menggunakan workgroup untuk menjalankan kueri dan Mengontrol biaya dan memantau kueri dengan CloudWatch metrik dan peristiwa.	18 Februari 2019
Menambahkan dukungan untuk menganalisis log dari Network Load Balancer.	Menambahkan contoh pertanyaan Athena untuk menganalisis log dari Network Load Balancer. Log ini menerima informasi rinci tentang permintaan Transport Layer Security (TLS) yang dikirim ke Network Load Balancer. Anda dapat menggunakan log akses ini untuk menganalisis pola lalu lintas dan memecahkan masalah. Untuk informasi, lihat the section called “Network Load Balancer”.	24 Januari 2019

Perubahan	Deskripsi	Tanggal rilis
Merilis versi baru driver JDBC dan ODBC dengan dukungan untuk akses gabungan ke API Athena dengan AD FS dan SAML 2.0 (Security Assertion Markup Language 2.0).	Dengan peluncuran driver ini, akses gabungan ke Athena didukung untuk Active Directory Federation Service (AD FS 3.0). Akses dibuat melalui versi driver JDBC atau ODBC yang mendukung SAML 2.0. Untuk informasi tentang cara mengonfigurasi akses gabungan ke API Athena, lihat the section called “Mengaktifkan akses federasi ke Athena API”.	10 November 2018
Menambahkan dukungan untuk kontrol akses terperinci ke basis data dan tabel di Athena. Selain itu, menambahkan kebijakan di Athena yang memungkinkan Anda mengenkripsi basis data dan tabel metadata dalam Katalog Data.	Menambahkan dukungan untuk membuat kebijakan berbasis identitas (IAM) yang menyediakan kontrol akses berbutir halus ke sumber daya di AWS Glue Data Catalog, seperti database dan tabel yang digunakan di Athena. Selain itu, Anda dapat mengenkripsi basis data dan tabel metadata dalam Katalog Data, dengan menambahkan kebijakan khusus untuk Athena. Lihat perinciannya di Akses berbutir halus ke database dan tabel di AWS Glue Data Catalog.	15 Oktober 2018

Perubahan	Deskripsi	Tanggal rilis
Menambahkan dukungan untuk`CREATE TABLE AS SELECT`. Membuat perbaikan lain dalam dokumentasi.	Menambahkan dukungan untuk`CREATE TABLE AS SELECT`. LihatMembuat tabel dari hasil query (CTAS),Pertimbangan dan batasan untuk kueri CTAS, danContoh kueri CTAS.	10 Oktober 2018
Dirilis ODBC driver versi 1.0.3 dengan dukungan untuk streaming hasil bukannya mengambil mereka di halaman. Membuat perbaikan lain dalam dokumentasi.	Versi driver ODBC 1.0.3 mendukung streaming hasil dan juga mencakup perbaikan, perbaikan bug, dan dokumentasi diperbarui untukMenggunakan SSL dengan Server Proxy". Untuk memuat turun versi driver ODBC 1.0.3 dan dokumentasinya, lihat Menghubungkan ke Amazon Athena dengan ODBC.	6 September 2018

Perubahan	Deskripsi	Tanggal rilis
Merilis driver JDBC versi 2.0.5 dengan dukungan default untuk hasil streaming bukannya mengambil mereka di halaman. Membuat perbaikan lain dalam dokumentasi.	Merilis driver JDBC 2.0.5 dengan dukungan default untuk hasil streaming bukannya mengambil mereka di halaman. Untuk informasi, lihat Menghubungkan ke Amazon Athena dengan JDBC.	16 Agustus 2018
Diperbarui dokumentasi untuk mengkueri Amazon Virtual Private Cloud aliran log, yang dapat disimpan langsung di Amazon S3 dalam format GZIP. Contoh diperbarui untuk mengkueri ALB log.	Diperbarui dokumentasi untuk mengkueri Amazon Virtual Private Cloud aliran log, yang dapat disimpan langsung di Amazon S3 dalam format GZIP. Untuk informasi, lihat Menanyakan log aliran VPC Amazon. Contoh diperbarui untuk mengkueri ALB log. Untuk informasi, lihat Meminta log Application Load Balancer.	7 Agustus 2018

Perubahan	Deskripsi	Tanggal rilis
Dukungan tambahan untuk IAM. Menambahkan pedoman untuk manipulasi skema untuk berbagai format penyimpanan data.	Dukungan tambahan untuk IAM. Untuk informasi, lihat Bekerja dengan pandangan. Diperbarui panduan ini dengan panduan tentang penanganan skema update untuk berbagai format penyimpanan data. Untuk informasi, lihat Menangani pembaruan skema.	5 Juni 2018
Peningkatan standar permintaan concurrency batas 5-20.	Anda dapat mengirimkan dan menjalankan hingga dua puluh`DDL`kueri dan dua puluh`SELECT`pertanyaan pada suatu waktu. Untuk informasi, lihat Service Quotas.	17 Mei 2018
Menambahkan tab kueri, dan kemampuan untuk mengonfigurasi lengkapi otomatis di Editor Kueri.	Menambahkan tab kueri, dan kemampuan untuk mengonfigurasi lengkapi otomatis di Editor Kueri. Untuk informasi, lihat Memulai.	8 Mei 2018
Merilis driver JDBC versi 2.0.2.	Merilis versi baru driver JDBC (versi 2.0.2). Untuk informasi, lihat Menghubungkan ke Amazon Athena dengan JDBC.	19 April 2018
Menambahkan lengkapi otomatis untuk mengetik kueri di konsol Athena.	Menambahkan lengkapi otomatis untuk mengetik kueri di konsol Athena.	6 April 2018

Perubahan	Deskripsi	Tanggal rilis
Ditambahkan kemampuan untuk membuat tabel Athena untuk file CloudTrail log langsung dari konsol. CloudTrail	Ditambahkan kemampuan untuk secara otomatis membuat tabel Athena untuk file CloudTrail log langsung dari konsol. CloudTrail Untuk informasi, lihat Menggunakan CloudTrail konsol untuk membuat tabel Athena untuk log CloudTrail .	15 Maret 2018
Menambahkan dukungan untuk membongkar data perantara ke disk dengan aman untuk mengkueri dengan`GROUP BY`.	Menambahkan kemampuan untuk aman offload menengah data ke disk untuk memori-intensif permintaan yang menggunakan`GROUP BY`Klausul. Ini meningkatkan keandalan kueri tersebut, mencegah“Sumber daya permintaan habis”kesalahan. Untuk informasi selengkapnya, lihat catatan rilis untuk2 Februari 2018.	2 Februari 2018
Menambahkan dukungan untuk Presto versi 0.172.	Upgrade mesin yang mendasari di Amazon Athena untuk versi berdasarkan versi Presto 0.172. Untuk informasi selengkapnya, lihat catatan rilis untuk19 Januari 2018.	19 Januari 2018
Menambahkan dukungan untuk Driver ODBC.	Menambahkan dukungan untuk menghubungkan Athena ke Driver ODBC. Untuk informasi selengkapnya, lihatMenghubungkan ke Amazon Athena dengan ODBC.	13 November 2017
Menambahkan dukungan untuk wilayah Asia Pacific (Seoul), Asia Pacific (Mumbai), dan Eropa (London). Menambahkan dukungan untuk mengkueri data geospasial.	Menambahkan dukungan untuk mengkueri data geospasial, dan untuk Asia Pacific (Seoul), Asia Pacific (Mumbai), Eropa (London). Untuk informasi, lihat Mengkueri data geospasial dan Wilayah AWS dan titik akhir.	1 November 2017

Perubahan	Deskripsi	Tanggal rilis
Menambahkan dukungan untuk Eropa (Frankfurt).	Menambahkan dukungan untuk Eropa (Frankfurt). Untuk daftar wilayah yang didukung, lihat Wilayah AWS dan titik akhir.	19 Oktober 2017
Ditambahkan dukungan untuk pertanyaan bernama Athena dengan. AWS CloudFormation	Ditambahkan dukungan untuk membuat kueri bernama Athena dengan. AWS CloudFormation Untuk informasi selengkapnya, lihat AWS::Athena::NamedQuerydi Panduan AWS CloudFormation Pengguna.	3 Oktober 2017
Menambahkan dukungan untuk Asia Pacific (Sydney).	Dukungan tambahan untuk Asia Pacific (Sydney). Untuk daftar wilayah yang didukung, lihat Wilayah AWS dan titik akhir.	25 September 2017
Menambahkan bagian ke panduan ini untuk menanyakan Layanan AWS log dan berbagai jenis data, termasuk peta, array, data bersarang, dan data yang berisi JSON.	Menambahkan contoh untukMeminta log Layanan AWSdan untuk mengkueri berbagai jenis data di Athena. Untuk informasi, lihat Menjalankan kueri SQL menggunakan Amazon Athena.	5 September 2017
Menambahkan dukungan untuk AWS Glue Data Catalog.	Menambahkan integrasi dengan AWS Glue Data Catalog dan wizard migrasi untuk memperbarui dari katalog data terkelola Athena ke. AWS Glue Data Catalog Untuk informasi selengkapnya, lihatIntegrasi dengan AWS GluedanAWS Glue.	14 Agustus 2017

Perubahan	Deskripsi	Tanggal rilis
Ditambahkan dukungan untuk Grok SerDe.	Menambahkan dukungan untuk Grok SerDe, yang menyediakan pencocokan pola yang lebih mudah untuk catatan dalam file teks tidak terstruktur seperti log. Untuk informasi lebih lanjut, lihat Grok SerDe. Menambahk an pintasan keyboard untuk menggulir riwayat kueri menggunakan konsol.	4 Agustus 2017
Menambahk an dukungan untuk Asia Pacific (Tokyo).	Menambahkan dukungan untuk Asia Pacific (Tokyo) dan Asia Pacific (Singapore). Untuk daftar wilayah yang didukung, lihat Wilayah AWS dan titik akhir.	22 Juni 2017
Menambahkan dukungan untuk Eropa (Irlandia).	Menambahkan dukungan untuk Eropa (Irlandia). Untuk informasi lebih lanjut, lihat Wilayah AWS dan titik akhir.	8 Juni 2017
Menambahkan API dan dukungan Amazon Athena. AWS CLI	Menambahkan API Amazon Athena dan AWS CLI dukungan untuk Athena. Update driver JDBC ke versi 1.1.0.	19 Mei 2017
Menambahkan dukungan untuk enkripsi data Amazon S3.	Menambahkan dukungan untuk enkripsi data Amazon S3 dan merilis update driver JDBC (versi 1.0.1) dengan dukungan enkripsi, perbaikan, dan perbaikan bug. Untuk informasi selengkapnya, lihat Enkripsi diam.	4 April 2017

Perubahan	Deskripsi	Tanggal rilis
Ditambahkan AWS CloudTrail SerDe.	Ditambahkan AWS CloudTrail SerDe, peningkatan kinerja, masalah partisi tetap. • AWS CloudTrail SerDe Telah digantikan oleh log Sarang JSON SerDe untuk membaca CloudTrail . Untuk informasi tentang menanyakan CloudTrail log, lihatMeminta log AWS CloudTrail. • Peningkatan performa saat memindai sejumlah besar partisi. • Peningkatan performa pada`MSCK Repair Table`operasi. • Menambahkan kemampuan untuk mengkueri data Amazon S3 yang disimpan di wilayah selain wilayah utama Anda. Tarif transfer data antar-wilayah standar untuk Amazon S3 berlaku selain biaya Athena standar.	24 Maret 2017
Menambahkan dukungan untuk US East (Ohio).	Menambahkan dukungan untuk AvroSerDe dan OpenCSV untuk SerDe memproses CSV, US East (Ohio), dan kolom pengeditan massal di wizard konsol. Peningkatan performa pada tabel Parquet besar.	20 Februari 2017
	Rilis awal dariPanduan Pengguna Amazon Athena.	Selasa, 07 Nopember 2016

AWS Glosarium

Untuk AWS terminologi terbaru, lihat AWS glosarium di Referensi.Glosarium AWS

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.