Guia do usuário

Amazon Athena

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Athena: Guia do usuário

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

As marcas comerciais e imagens comerciais da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestigie a Amazon. Todas as outras marcas comerciais que não são propriedade da Amazon pertencem aos respectivos proprietários, os quais podem ou não ser afiliados, estar conectados ou ser patrocinados pela Amazon.

Table of Contents

O que é o Amazon Athena? 1
Quando devo usar o Athena? 1
Amazon Athena 2
Amazon EMR 2
Amazon Redshift 3
Integrações de AWS service (Serviço da AWS) ao Athena 4
Configuração 9
Cadastre-se em uma Conta da AWS 9
Criar um usuário com acesso administrativo 10
Conceder acesso programático 11
Anexar políticas gerenciadas do Athena 13
Acessar o Athena 14
Como usar o SQL do Athena 16
Noções básicas sobre tabelas, bancos de dados e catálogos de dados 17
Conceitos básicos 20
Pré-requisitos 20
Etapa 1: criar um banco de dados 20
Etapa 2: Criar uma tabela 24
Etapa 3: consultar dados 29
Salvar consultas 32
Atalhos de teclado e sugestões de digitação antecipada 32
Conectar a outras origens de dados 33
Conectar a origens de dados 33
Integração com AWS Glue 34
Usar um metastore do Hive 53
Usar a consulta federada do Amazon Athena 90
Políticas do IAM de acesso a catálogos de dados 368
Gerenciar origens de dados 375
Usar o DataZone 377
Conectar-se ao Amazon Athena com drivers ODBC e JDBC 379
Conectar-se ao Athena com JDBC 380
Conectar-se ao Athena com ODBC 428
Criar bancos de dados e tabelas 572
Criar bancos de dados 573

Criar tabelas ... 576
Nomes de tabelas, bancos de dados e colunas ... 581
Palavras-chave reservadas ... 583
Local da tabela no Amazon S3 ... 585
Formatos de armazenamento colunar ... 588
Converter em formatos colunares ... 590
Particionar dados ... 590
Projeção de partições ... 598
Criar uma tabela a partir de resultados de consultas (CTAS) ... 622
Considerações e limitações de consultas CTAS ... 623
Executar consultas CTAS no console ... 625
Particionamento e bucketing ... 628
Exemplos de CTAS ... 633
Usar CTAS e INSERT INTO para ETL ... 639
Contornar o limite de 100 partições ... 647
Referência de SerDe ... 652
Usar um SerDe ... 652
SerDes e formatos de dados compatíveis ... 654
Executar consultas ... 705
Visualizar planos de consultas ... 706
Resultados da consulta e consultas recentes ... 712
Reutilização dos resultados da consulta ... 730
Visualizar estatísticas de consultas ... 735
Trabalhar com visualizações ... 741
Usar consultas salvas ... 758
Utilizar consultas parametrizadas ... 760
Otimizador baseado em custos ... 769
Consulta do S3 Express One Zone ... 775
Consultar o S3 Glacier ... 777
Tratamento de atualizações do esquema ... 779
Consultar matrizes ... 793
Consultar dados geoespaciais ... 819
Consultar JSON ... 845
Usar ML com o Athena ... 856
Consultas com UDFs ... 859
Realizar consultas entre regiões ... 871

Consulta no AWS Glue Data Catalog ... 872
Consulta de logs do AWS service (Serviço da AWS) ... 880
Consultar logs do servidor Web ... 961
Usar transações ACID ... 972
Consulta de tabelas Delta Lake ... 973
Consultar conjuntos de dados do Hudi ... 978
Usar tabelas Iceberg ... 988
Segurança ... 1013
Proteção de dados ... 1014
Gerenciamento de identidade e acesso ... 1029
Registrar em log e monitoramento ... 1098
Validação de conformidade ... 1104
Resiliência ... 1105
Segurança da infraestrutura ... 1105
Análise de configuração e vulnerabilidade ... 1109
Usar o Athena com o Lake Formation ... 1109
Gerenciamento do workload ... 1172
Usar grupos de trabalho para controlar o acesso a consultas e os custos ... 1172
Como gerenciar a capacidade de processamento de consulta ... 1238
Ajuste de performance ... 1256
Suporte à compactação ... 1280
Marcar recursos ... 1289
Service Quotas ... 1305
Versionamento do mecanismo do Athena ... 1308
Alterar versões do mecanismo do Athena ... 1309
Referência da versão do mecanismo do Athena ... 1314
Referência do SQL para o Athena ... 1348
Tipos de dados no Athena ... 1348
Consultas, funções e operadores em DML ... 1358
Instruções DDL ... 1419
Considerações e limitações ... 1476
Solução de problemas ... 1478
CREATE TABLE AS SELECT (CTAS) ... 1479
Problemas no arquivo de dados ... 1479
Tabelas do Linux Foundation Delta Lake ... 1481
Consultas federadas ... 1482

Erros relacionados ao JSON ... 1483
MSCK REPAIR TABLE ... 1485
Problemas de saída ... 1485
Problemas do Parquet ... 1486
Problemas de particionamento ... 1487
Permissões ... 1489
Problemas de sintaxe da consulta ... 1491
Problemas de tempo limite de consulta ... 1493
Problemas de controle de utilização ... 1494
Visões ... 1494
Grupos de trabalho ... 1495
Recursos adicionais do ... 1495
Catálogo de erros do Athena ... 1496
Exemplos de código ... 1502
Constantes ... 1503
Criar um cliente para acessar o Athena ... 1504
Iniciar a execução da consulta ... 1504
Interromper a execução da consulta ... 1508
Listar execuções de consulta ... 1510
Criar uma consulta nomeada ... 1511
Excluir uma consulta nomeada ... 1513
Listar consultas nomeadas ... 1515
Uso do Apache Spark ... 1517
Considerações e limitações ... 1517
Conceitos básicos ... 1519
Criação de um grupo de trabalho habilitado para Spark no Athena ... 1519
Como abrir o explorador de cadernos e alternar grupos de trabalho ... 1524
Como executar o caderno de exemplo ... 1525
Como editar os detalhes da sessão ... 1526
Como visualizar detalhes da sessão e dos cálculos ... 1528
Encerrar uma sessão ... 1529
Como criar seu próprio caderno ... 1529
Como abrir um caderno criado anteriormente ... 1531
Como trabalhar com cadernos ... 1531
Sessões e cálculos ... 1532
Uso do editor de cadernos do Athena ... 1532

Magics 1536
Gerenciamento de arquivos de cadernos 1546
Usar formatos de tabela que não sejam do Hive 1548
Suporte à biblioteca Python 1553
Definições 1553
Gerenciamento de ciclo de vida 1554
Bibliotecas Python 1555
Importação de arquivos e de bibliotecas 1556
Adicionar arquivos JAR e configuração personalizada 1569
Usar o console do Athena 1570
Usar a AWS CLI ou a API do Athena 1571
Solução de problemas 1571
Formatos de dados e de armazenamento compatíveis 1572
Monitoramento de cálculos do Apache Spark 1573
Lista de métricas e dimensões do CloudWatch para cálculos do Apache Spark no Athena 1574
Habilitar buckets de pagamentos pelo solicitante 1575
1. Habilitar pagamentos pelo solicitante em um bucket do Amazon S3 e adicionar uma política de bucket 1575
2. Criar uma política do IAM e anexá-la ao perfil do IAM 1576
3. Adicionar uma propriedade de sessão do Athena para Spark 1577
Habilitar a criptografia do Spark 1578
Console do Athena 1578
AWS CLI 1579
API do Athena 1580
Acesso ao catálogo entre contas 1580
1. No AWS Glue, forneça acesso aos perfis do consumidor 1580
2. Configurar a conta do consumidor para acesso 1581
3. Configurar uma sessão e criar uma consulta 1583
Recursos adicionais do 1584
Cotas de serviço 1584
APIs para cadernos do Athena 1585
Problemas conhecidos 1586
Exceção para argumento inválido ao criar uma tabela 1586
Banco de dados criado em um local do grupo de trabalho 1587
Problemas com tabelas gerenciadas pelo Hive no banco de dados do AWS Glue padrão . 1587

Incompatibilidade de formatos de arquivo CSV e JSON entre o Athena para Spark e o Athena SQL ... 1589
Solução de problemas ... 1589
Grupos de trabalho habilitados para Spark ... 1590
Uso de EXPLAIN do Spark ... 1593
Registro em log de eventos da aplicação ... 1595
Uso do CloudTrail para chamadas de API de cadernos ... 1599
Limite de tamanho do bloco de código ... 1606
Sessões ... 1608
Tabelas ... 1609
Obter suporte ... 1612
Notas de release ... 1613
2024 ... 1613
26 de abril de 2024 ... 1613
24 de abril de 2024 ... 1613
16 de abril de 2024 ... 1614
10 de abril de 2024 ... 1614
8 de abril de 2024 ... 1615
15 de março de 2024 ... 1615
15 de fevereiro de 2024 ... 1615
31 de janeiro de 2024 ... 1616
2023 ... 1616
14 de dezembro de 2023 ... 1616
9 de dezembro de 2023 ... 1616
7 de dezembro de 2023 ... 1617
5 de dezembro de 2023 ... 1617
28 de novembro de 2023 ... 1618
27 de novembro de 2023 ... 1618
17 de novembro de 2023 ... 1619
16 de novembro de 2023 ... 1620
31 de outubro de 2023 ... 1620
25 de outubro de 2023 ... 1621
17 de outubro de 2023 ... 1621
26 de setembro de 2023 ... 1621
23 de agosto de 2023 ... 1622
10 de agosto de 2023 ... 1622

31 de julho de 2023 1622
27 de julho de 2023 1623
24 de julho de 2023 1623
20 de julho de 2023 1623
13 de julho de 2023 1624
3 de julho de 2023 1624
30 de junho de 2023 1625
29 de junho de 2023 1625
28 de junho de 2023 1626
12 de junho de 2023 1626
8 de junho de 2023 1626
2 de junho de 2023 1627
25 de maio de 2023 1628
18 de maio de 2023 1629
15 de maio de 2023 1629
10 de maio de 2023 1629
8 de maio de 2023 1630
28 de abril de 2023 1632
17 de abril de 2023 1632
14 de abril de 2023 1633
4 de abril de 2023 1633
30 de março de 2023 1633
28 de março de 2023 1634
27 de março de 2023 1635
17 de março de 2023 1635
8 de março de 2023 1636
15 de fevereiro de 2023 1636
31 de janeiro de 2023 1636
20 de janeiro de 2023 1636
3 de janeiro de 2023 1637
2022 1637
14 de dezembro de 2022 1637
2 de dezembro de 2022 1638
30 de novembro de 2022 1638
18 de novembro de 2022 1639
17 de novembro de 2022 1639

14 de novembro de 2022 1640
11 de novembro de 2022 1641
8 de novembro de 2022 1642
13 de outubro de 2022 1642
10 de outubro de 2022 1643
23 de setembro de 2022 1643
13 de setembro de 2022 1643
31 de agosto de 2022 1644
23 de agosto de 2022 1644
3 de agosto de 2022 1644
1º de agosto de 2022 1645
21 de julho de 2022 1645
11 de julho de 2022 1646
8 de julho de 2022 1647
6 de junho de 2022 1647
25 de maio de 2022 1647
6 de maio de 2022 1648
22 de abril de 2022 1648
21 de abril de 2022 1649
13 de abril de 2022 1649
30 de março de 2022 1650
18 de março de 2022 1650
2 de março de 2022 1651
23 de fevereiro de 2022 1651
15 de fevereiro de 2022 1652
14 de fevereiro de 2022 1653
9 de fevereiro de 2022 1653
8 de fevereiro de 2022 1653
28 de janeiro de 2022 1653
13 de janeiro de 2022 1654
2021 1655
26 de novembro de 2021 1655
24 de novembro de 2021 1655
22 de novembro de 2021 1655
18 de novembro de 2021 1656
17 de novembro de 2021 1657

16 de novembro de 2021 1657
12 de novembro de 2021 1658
2 de novembro de 2021 1659
29 de outubro de 2021 1659
4 de outubro de 2021 1660
16 de setembro de 2021 1660
15 de setembro de 2021 1661
31 de agosto de 2021 1662
12 de agosto de 2021 1662
6 de agosto de 2021 1663
5 de agosto de 2021 1663
30 de julho de 2021 1663
21 de julho de 2021 1664
16 de julho de 2021 1664
8 de julho de 2021 1665
1º de julho de 2021 1665
23 de junho de 2021 1666
12 de maio de 2021 1666
10 de maio de 2021 1666
5 de maio de 2021 1666
30 de abril de 2021 1667
29 de abril de 2021 1667
26 de abril de 2021 1667
21 de abril de 2021 1668
5 de abril de 2021 1668
30 de março de 2021 1669
25 de março de 2021 1669
5 de março de 2021 1669
25 de fevereiro de 2021 1669
2020 1670
16 de dezembro de 2020 1670
24 de novembro de 2020 1670
11 de novembro de 2020 1670
22 de outubro de 2020 1673
29 de julho de 2020 1673
9 de julho de 2020 1673

1 de junho de 2020 1674
21 de maio de 2020 1674
1 de abril de 2020 1675
11 de março de 2020 1675
6 de março de 2020 1675
2019 1675
26 de novembro de 2019 1675
12 de novembro de 2019 1680
8 de novembro de 2019 1680
8 de outubro de 2019 1680
19 de setembro de 2019 1680
12 de setembro de 2019 1681
16 de agosto de 2019 1681
9 de agosto de 2019 1682
26 de junho de 2019 1682
24 de maio de 2019 1682
05 de março de 2019 1682
22 de fevereiro de 2019 1683
18 de fevereiro de 2019 1684
2018 1686
20 de novembro de 2018 1686
15 de outubro de 2018 1687
10 de outubro de 2018 1687
6 de setembro de 2018 1688
23 de agosto de 2018 1689
16 de agosto de 2018 1689
7 de agosto de 2018 1690
5 de junho de 2018 1690
17 de maio de 2018 1692
19 de abril de 2018 1692
6 de abril de 2018 1693
15 de março de 2018 1693
2 de fevereiro de 2018 1693
19 de janeiro de 2018 1693
2017 1694
13 de novembro de 2017 1694

1 de novembro de 2017 1695
19 de outubro de 2017 1695
3 de outubro de 2017 1695
25 de setembro de 2017 1695
14 de agosto de 2017 1695
4 de agosto de 2017 1696
22 de junho de 2017 1696
8 de junho de 2017 1696
19 de maio de 2017 1696
4 de abril de 2017 1698
24 de março de 2017 1699
20 de fevereiro de 2017 1700
Histórico do documento 1703
Glossário da AWS 1728

O que é o Amazon Athena?

O Amazon Athena é um serviço de consultas interativas que facilita a análise de dados diretamente no Amazon Simple Storage Service (Amazon S3) usando SQL padrão. Com algumas ações no AWS Management Console, você pode direcionar o Athena para os dados armazenados no Amazon S3 e começar a usar o SQL padrão para executar consultas ad-hoc e obter resultados em segundos.

Para ter mais informações, consulte Conceitos básicos.

O Amazon Athena também facilita a execução interativa de análises de dados usando o Apache Spark sem a necessidade de planejamento, configuração ou gerenciamento de recursos. Ao executar aplicações do Apache Spark no Athena, você envia o código Spark para processamento e recebe os resultados de forma direta. Use a experiência simplificada de cadernos no console do Amazon Athena para desenvolver aplicações do Apache Spark usando Python ou APIs para cadernos do Athena.

Para ter mais informações, consulte Conceitos básicos do Apache Spark no Amazon Athena.

O Athena SQL e o Apache Spark no Amazon Athena são uma tecnologia sem servidor, portanto, não há infraestrutura para configuração ou gerenciamento, e você paga apenas pelas consultas executadas. O Athena é escalado automaticamente, executando consultas em paralelo, o que acelera os resultados mesmo em conjuntos de dados grandes e consultas complexas.

Tópicos

- Quando devo usar o Athena?
- Integrações de AWS service (Serviço da AWS) ao Athena
- Configuração
- Acessar o Athena

Quando devo usar o Athena?

Serviços de consulta como o Amazon Athena, data warehouses como o Amazon Redshift e frameworks sofisticadas de processamento de dados, como o Amazon EMR, atendem a diferentes necessidades e casos de uso. As orientações a seguir podem ajudar você a escolher um ou mais serviços com base nas suas necessidades.

Amazon Athena

O Athena ajuda a analisar dados desestruturados, semiestruturados e estruturados armazenados no Amazon S3. Entre os exemplos estão formatos de dados CSV, JSON ou colunares, como Apache Parquet e Apache ORC. Você pode usar o Athena para executar consultas ad-hoc com o ANSI SQL, sem necessidade de agregar ou carregar os dados no Athena.

O Athena se integra ao Amazon QuickSight para facilitar a visualização de dados. Você pode usar o Athena para gerar relatórios ou explorar dados com ferramentas de business intelligence ou clientes SQL conectados com um driver JDBC ou ODBC. Para obter mais informações, consulte O que é o Amazon QuickSight no Guia do usuário do Amazon QuickSight e Conectar-se ao Amazon Athena com drivers ODBC e JDBC.

O Athena se integra ao AWS Glue Data Catalog, que oferece armazenamento de metadados persistente para seus dados no Amazon S3. Isso permite criar tabelas e consultar dados no Athena com base em um armazenamento central de metadados disponível em sua conta da Amazon Web Services e integrado ao ETL e aos recursos de descoberta de dados do AWS Glue. Para obter mais informações, consulte Integração com o AWS Glue e O que é o AWS Glue no Guia do desenvolvedor do AWS Glue.

O Amazon Athena facilita a execução de consultas interativas com dados diretamente no Amazon S3, sem exigir a formatação de dados ou o gerenciamento da infraestrutura. Por exemplo, o Athena é útil quando você deseja executar uma consulta rápida em logs da Web para solucionar um problema de performance no seu site. Com o Athena, é possível começar rapidamente: basta definir uma tabela para os seus dados e começar a consultar usando SQL padrão.

Você deve usar o Amazon Athena quando deseja executar consultas SQL assistemáticas interativas em dados no Amazon S3 sem ter que gerenciar infraestruturas ou clusters. O Amazon Athena fornece a maneira mais fácil de executar consultas assistemáticas para dados no Amazon S3, sem a necessidade de configurar ou gerenciar servidores.

Para obter uma lista de Serviços da AWS que o Athena utiliza ou se integra, consulte the section called “Integrações de AWS service (Serviço da AWS) ao Athena”.

Amazon EMR

O Amazon EMR torna simples e econômico executar frameworks de processamento altamente distribuídas, como Hadoop, Spark e Presto, quando comparadas a implantações on-premises. O Amazon EMR é flexível: é possível executar aplicações e códigos personalizados e definir

parâmetros específicos de computação, memória, armazenamento e aplicações para otimizar seus requisitos analíticos.

Além de executar consultas SQL, o Amazon EMR pode executar várias tarefas de processamento de dados de aumento na escala na horizontal para aplicações como machine learning, análises de gráficos, transformação de dados, dados de transmissão e praticamente qualquer coisa que você possa codificar. Você deve utilizar o Amazon EMR se usar código personalizado para processar e analisar conjuntos de dados extremamente grandes com as frameworks de processamento de big data mais recentes, como Spark, Hadoop, Presto ou Hbase. O Amazon EMR oferece controle total sobre a configuração dos seus clusters e do software instalado neles.

É possível usar o Amazon Athena para consultar dados processados com o uso do Amazon EMR. O Amazon Athena oferece suporte a vários dos mesmos formatos de dados que o Amazon EMR. O catálogo de dados do Athena é compatível com o metastore do Hive. Se você usa o EMR e já tem um metastore do Hive, pode executar suas instruções DDL no Amazon Athena e consultar seus dados imediatamente, sem afetar os trabalhos do Amazon EMR.

Amazon Redshift

Um data warehouse como o Amazon Redshift é a melhor opção quando você precisa reunir dados de várias origens diferentes (como sistemas de inventário, sistemas financeiros e sistemas de vendas a varejo) em um formato comum e armazená-los por longos períodos de tempo. Se quiser criar relatórios comerciais sofisticados com base em dados históricos, um data warehouse como o Amazon Redshift é a melhor escolha. O mecanismo de consulta no Amazon Redshift foi otimizado para ter uma performace especialmente boa na execução de consultas complexas que unem várias tabelas de banco de dados muito grandes. Se você precisar executar consultas com dados altamente estruturados com muitas uniões em muitas tabelas muito grandes, escolha o Amazon Redshift.

Para obter mais informações de quando usar o Athena, consulte os seguintes recursos:

- Guia de decisão para serviços de análise em AWS no Centro de recursos de conceitos básicos
- Quando usar o Athena em comparação com outros serviços de big data nas Perguntas frequentes sobre o Amazon Athena
- Visão geral do Amazon Athena
- Recursos do Amazon Athena
- Perguntas frequentes sobre o Amazon Athena
- Publicações de blog do Amazon Athena

Integrações de AWS service (Serviço da AWS) ao Athena

Você pode usar o Athena para consultar dados dos Serviços da AWS listados nesta seção. Para ver as regiões com suporte para cada serviço, consulte Regiões e endpoints no Referência geral da Amazon Web Services.

Serviços da AWS integrados ao Athena

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

Para obter informações sobre cada integração, consulte as seções a seguir.

AWS CloudFormation

Reserva de capacidade

Tópico de referência: AWS::Athena::CapacityReservation no Guia do usuário do AWS CloudFormation

Especifica uma reserva de capacidade com o nome fornecido e o número de unidades de processamento de dados solicitadas. Para obter mais informações, consulte Como gerenciar a capacidade de processamento de consulta no Guia do usuário do Amazon Athena e CreateCapacityReservation na Amazon Athena API Reference.

Catálogo de dados

Tópico de referência: AWS::Athena::DataCatalog no Guia do usuário do AWS CloudFormation

Especifique um catálogo de dados do Athena, incluindo nome, descrição, tipo, parâmetros e etiquetas. Para obter mais informações, consulte Noções básicas sobre tabelas, bancos de dados e catálogos de dados no Guia do usuário do Amazon Athena e CreateDataCatalog na Amazon Athena API Reference.

Consulta nomeada

Tópico de referência: AWS::Athena::NamedQuery no Guia do usuário do AWS CloudFormation

Especifique as consultas nomeadas com AWS CloudFormation e execute-as no Athena. As consultas nomeadas permitem mapear um nome para uma consulta e executá-la como consulta salva no console do Athena. Para obter mais informações, consulte Usar consultas salvas no Guia do usuário do Amazon Athena e CreateNamedQuery na Amazon Athena API Reference.

Instrução preparada

Tópico de referência: AWS::Athena::PreparedStatement no Guia do usuário do AWS CloudFormation

Especifica uma instrução preparada para uso com consultas SQL no Athena. Uma instrução preparada contém espaços reservados de parâmetros dos quais os valores são fornecidos no runtime. Para obter mais informações, consulte Utilizar consultas parametrizadas no Guia do usuário do Amazon Athena e CreatePreparedStatement na Amazon Athena API Reference.

WorkGroup

Tópico de referência: AWS::Athena::WorkGroup no Guia do usuário do AWS CloudFormation

Especifique grupos de trabalho do Athena usando o AWS CloudFormation. Use os grupos de trabalho do Athena para isolar as consultas que são suas ou do seu grupo das demais consultas na mesma conta. Para obter mais informações, consulte Usar grupos de trabalho para controlar o acesso a consultas e os custos no Manual do usuário do Amazon Athena e CreateWorkGroup na Referência de API do Amazon Athena.

Amazon CloudFront

Tópico de referência: Consultar os logs do Amazon CloudFront

Use o Athena para consultar logs do Amazon CloudFront. Para obter mais informações sobre o uso do CloudFront, consulte o Guia do desenvolvedor do Amazon CloudFront.

AWS CloudTrail

Tópico de referência: Consultar os logs do AWS CloudTrail

O uso do Athena com os logs do CloudTrail é uma ótima maneira de melhorar a análise das atividades dos serviços da AWS. Por exemplo, é possível usar consultas para identificar tendências e isolar ainda mais a atividade por atributos, como endereço IP de origem ou usuário. É possível criar tabelas para consultar os logs diretamente no console do CloudTrail e usá-las para executar as consultas no Athena. Para ter mais informações, consulte Usar o console do CloudTrail para criar uma tabela do Athena com logs do CloudTrail .

Amazon DataZone

Tópico de referência: Usar o Amazon DataZone no Athena

Use o Amazon DataZone para compartilhar, pesquisar e descobrir dados em grande escala, ultrapassando os limites organizacionais. O DataZone simplifica sua experiência em todos os serviços de análise da AWS, como o Athena, o AWS Glue e o AWS Lake Formation. Se você tiver grandes quantidades de dados em diferentes fontes de dados, poderá usar o Amazon DataZone para criar agrupamentos de pessoas, dados e ferramentas baseados em casos de uso comercial.

No Athena, você pode usar o editor de consultas para acessar e consultar os ambientes do DataZone. Para ter mais informações, consulte Usar o Amazon DataZone no Athena.

Elastic Load Balancing

Tópico de referência: Consultar logs do Application Load Balancer

Consultar logs do Application Load Balancer permite consultar a origem do tráfego, a latência e os bytes transferidos de e para instâncias do Elastic Load Balancing e aplicativos de backend. Para ter mais informações, consulte Consultar logs do Application Load Balancer.

Tópico de referência: Consultar logs do Classic Load Balancer

Consulte os logs do Classic Load Balancer para analisar e entender os padrões de tráfego de e para as instâncias e aplicações de backend do Elastic Load Balancing. Você pode ver a origem do tráfego, da latência e dos bytes transferidos. Para obter mais informações, consulte Criar a tabela para logs do ELB.

Amazon EMR Studio

Tópico de referência: Use the Amazon Athena SQL editor in EMR Studio

É possível usar o Athena em um EMR Studio para desenvolver e executar consultas interativas. Isso possibilita que você use o EMR Studio para realizar análises SQL no Athena usando a mesma interface do Amazon EMR que é usada para Spark, Scala e outras workloads. Com a integração do Athena no EMR Studio, você pode executar as seguintes tarefas:

- Fazer consultas SQL do Athena
- Visualizar resultados da consulta
- Visualizar o histórico de consultas
- Visualizar as consultas salvas
- Fazer consultas parametrizadas
- Visualizar bancos de dados, tabelas e visualizações de um catálogo de dados

Os seguintes atributos do Athena não estão disponíveis no Amazon EMR Studio:

- Recursos administrativos, como a criação ou a atualização de grupos de trabalho, fontes de dados ou reservas de capacidade do Athena.
- Athena para o Spark ou para cadernos do Spark
- Integração do DataZone
- Step Functions

A integração do EMR Studio com o Athena está disponível em todas as Regiões da AWS nas quais o EMR Studio e o Athena estão disponíveis. Para obter mais informações sobre como usar o Athena no EMR Studio, consulte Use the Amazon Athena SQL editor in EMR Studio no Guia de gerenciamento do Amazon EMR.

AWS Glue Data Catalog

Tópico de referência: Integração com AWS Glue

O Athena se integra ao AWS Glue Data Catalog, que oferece armazenamento de metadados persistente para seus dados no Amazon S3. Isso permite criar tabelas e consultar dados no Athena com base em um armazenamento central de metadados disponível em sua conta da Amazon Web Services e integrado ao ETL e aos recursos de descoberta de dados do AWS Glue. Para obter mais informações, consulte Integração com AWS Glue e O que é o AWS Glue? no Guia do desenvolvedor do AWS Glue.

AWS Identity and Access Management (IAM)

Tópico de referência: Ações no Amazon Athena

Você pode usar as ações de API do Athena nas políticas de permissão do IAM. Para obter mais informações, consulte Ações no Amazon Athena e Gerenciamento de identidade e acesso no Athena.

Amazon QuickSight

Tópico de referência: Conectar-se ao Amazon Athena com drivers ODBC e JDBC

O Athena se integra ao Amazon QuickSight para facilitar a visualização de dados. Você pode usar o Athena para gerar relatórios ou explorar dados com ferramentas de business intelligence ou clientes SQL conectados com um driver JDBC ou ODBC. Para obter mais informações sobre o Amazon QuickSight, consulte O que é o Amazon QuickSight no Manual do usuário do Amazon QuickSight. Para obter informações sobre como usar os drivers JDBC e ODBC com o Athena, consulte Conectar-se ao Amazon Athena com drivers ODBC e JDBC.

Inventário do Amazon S3

Tópico de referência: Consultar o inventário com o Athena no Guia do usuário do Amazon Simple Storage Service

Você pode usar o Amazon Athena para consultar o inventário do Amazon S3 com o SQL padrão. Você pode usar o inventário do Amazon S3 para auditar e gerar relatórios de status da replicação e criptografia dos seus objetos para atender às necessidades dos negócios, de compatibilidade e regulatórias. Para obter mais informações, consulte Inventário do Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

AWS Step Functions

Tópico de referência: Chamar o Athena com o Step Functions no Guia do desenvolvedor do AWS Step Functions

Chame o Athena com o AWS Step Functions. O AWS Step Functions pode controlar a seleção de Serviços da AWS diretamente no Amazon States Language. Você pode usar o Step Functions com o Athena para iniciar e interromper a execução da consulta, acessar os resultados da consulta, executar consultas de dados específicas ou agendadas e recuperar os resultados de data lakes no Amazon S3. O perfil do Step Functions deve ter permissões para usar o Athena. Para mais informações, consulte o Guia do desenvolvedor do AWS Step Functions.

Vídeo: Orquestrar consultas do Amazon Athena com o AWS Step Functions

O vídeo a seguir mostra como usar o Amazon Athena e o AWS Step Functions para executar uma consulta do Athena agendada regularmente e gerar um relatório correspondente.

Orquestrar consultas do Amazon Athena com o AWS Step Functions

Para ver um exemplo que usa o Step Functions e o Amazon EventBridge para orquestrar o AWS Glue DataBrew, o Athena e o Amazon QuickSight, consulte Orquestrar um trabalho do AWS Glue DataBrew e uma consulta do Amazon Athena com o AWS Step Functions no blog de big data da AWS.

Inventário do AWS Systems Manager

Tópico de referência: Consultar dados de inventário de várias regiões e contas no Guia do usuário do AWS Systems Manager

O inventário do AWS Systems Manager se integra ao Amazon Athena para ajudar você a consultar dados de inventário de várias contas e Regiões da AWS. Para mais informações, consulte o Guia do usuário do AWS Systems Manager.

Amazon Virtual Private Cloud

Tópico de referência: Consultar os logs de fluxo do Amazon VPC

Os logs de fluxo do Amazon Virtual Private Cloud capturam informações do tráfego de IP de entrada e saída das interfaces de rede em uma VPC. Consulte os logs no Athena para investigar os padrões de tráfego de rede e identificar ameaças e riscos em toda a rede do Amazon VPC. Para obter mais informações sobre o Amazon VPC, consulte o Manual do usuário do Amazon VPC.

Configuração

Se você já se cadastrou na Amazon Web Services, pode começar a usar o Amazon Athena imediatamente. Se você ainda não se cadastrou na AWS ou precisa de ajuda para começar, conclua as tarefas a seguir.

Cadastre-se em uma Conta da AWS

Se você ainda não tem Conta da AWS, siga as etapas a seguir para criar uma.

Para se cadastrar em uma Conta da AWS

1. Abra https://portal.aws.amazon.com/billing/signup.

2. Siga as instruções on-line.

   Parte do procedimento de inscrição envolve receber uma chamada telefônica e digitar um código de verificação no teclado do telefone.

   Quando você se cadastra em uma Conta da AWS, um Usuário raiz da conta da AWS é criado. O usuário raiz tem acesso a todos os Serviços da AWS e recursos na conta. Como uma prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que requerem o acesso de usuário-raiz.

A AWS envia um e-mail de confirmação depois que o processo de cadastramento é concluído. A qualquer momento, é possível visualizar as atividades da conta atual e gerenciar sua conta acessando https://aws.amazon.com/ e selecionando Minha conta.

Criar um usuário com acesso administrativo

Depois de se cadastrar em uma Conta da AWS, proteja seu Usuário raiz da conta da AWS, habilite o AWS IAM Identity Center e crie um usuário administrativo para não usar o usuário raiz em tarefas cotidianas.

Proteger seu Usuário raiz da conta da AWS

1. Faça login no AWS Management Console como o proprietário da conta ao escolher a opção Usuário raiz e inserir o endereço de e-mail da Conta da AWS. Na próxima página, digite sua senha.

   Para obter ajuda ao fazer login usando o usuário raiz, consulte Fazer login como usuário raiz no Guia do usuário do Início de Sessão da AWS.

2. Ative a autenticação multifator (MFA) para o usuário raiz.c

   Para obter instruções, consulte Habilitar um dispositivo MFA virtual para o usuário raiz de sua conta da Conta da AWS para seu (console) no Guia do usuário do IAM.

Criar um usuário com acesso administrativo

1. Habilitar o IAM Identity Center.

   Para obter instruções, consulte Habilitar AWS IAM Identity Center no Guia do usuário do AWS IAM Identity Center.

2. No Centro de Identidade do IAM, conceda o acesso administrativo para um usuário.

   Para obter um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte Configurar o acesso dos usuários com o Diretório do Centro de Identidade do IAM padrão no Guia do usuário do AWS IAM Identity Center.

Iniciar sessão como o usuário com acesso administrativo

- Para fazer login com seu usuário do Centro de Identidade do IAM, use a URL de login que foi enviada ao seu endereço de e-mail quando você criou o usuário do Centro do Usuário do IAM.

  Para obter ajuda com o login utilizando um usuário do Centro de Identidade do IAM, consulte Fazer login no portal de acesso da AWS, no Guia do usuário do Início de Sessão da AWS.

Atribuir acesso para usuários adicionais

1. No Centro de Identidade do IAM, crie um conjunto de permissões que siga as práticas recomendadas de aplicação de permissões com privilégio mínimo.

   Para obter instruções, consulte Criar um conjunto de permissões no Guia do usuário do AWS IAM Identity Center.
2. Atribua usuários para um grupo e, em seguida, atribua o acesso de autenticação única ao grupo.

   Para obter instruções, consulte Adicionar grupos no Guia do usuário do AWS IAM Identity Center.

Conceder acesso programático

Os usuários precisam de acesso programático se quiserem interagir com a AWS de fora do AWS Management Console. A forma de conceder acesso programático depende do tipo de usuário que está acessando a AWS.

Para conceder acesso programático aos usuários, selecione uma das seguintes opções:

Qual usuário precisa de acesso programático?	Para	Por
Identificação da força de trabalho (Usuários gerenciados no Centro de Identidade do IAM)	Use credenciais temporárias para assinar solicitações programáticas para a AWS CLI, os SDKs da AWS ou as APIs da AWS.	Siga as instruções da interface que deseja utilizar. • Para a AWS CLI, consulte Configuração da AWS CLI para usar o AWS IAM Identity Center no AWS Command Line Interface Guia do usuário da . • Para os SDKs da AWS, ferramentas e APIs da AWS, consulte Autenticação do Centro de Identidade do IAM no Guia de referência de ferramentas e SDKs da AWS.
IAM	Use credenciais temporárias para assinar solicitações programáticas para a AWS CLI, os SDKs da AWS ou as APIs da AWS.	Siga as instruções em Como usar credenciais temporárias com recursos da AWS no Guia do usuário do IAM.
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitações programáticas para a AWS CLI, os SDKs da AWS ou as APIs da AWS.	Siga as instruções da interface que deseja utilizar. • Para a AWS CLI, consulte Autenticação usando as credenciais de usuário do IAM no Guia do usuário da AWS Command Line Interface. • Para as ferramentas e SDKs da AWS, consulte

Qual usuário precisa de acesso programático?	Para	Por
		Autenticação usando as credenciais de longo prazo no Guia de referência de ferramentas e SDKs da AWS. • Para as APIs da AWS, consulte Gerenciamento de chaves de acesso de usuários do IAM no Guia do usuário do IAM.

Anexar políticas gerenciadas do Athena

As políticas gerenciadas do Athena concedem permissões para uso dos recursos do Athena. É possível vincular essas políticas gerenciadas a um ou mais perfis do IAM, os quais os usuários podem assumir para usar o Athena.

Um perfil do IAM é uma identidade do IAM que você pode criar em sua conta que tem permissões específicas. Um perfil do IAM é semelhante a um usuário do IAM porque é uma identidade da AWS com políticas de permissão que determinam o que ela pode e não pode fazer na AWS. No entanto, em vez de ser exclusivamente associada a uma pessoa, o propósito do perfil é ser assumido por qualquer pessoa que precisar dele. Além disso, um perfil não tem credenciais de longo prazo padrão associadas a ele, como senha ou chaves de acesso. Em vez disso, quando você assumir um perfil, ele fornecerá credenciais de segurança temporárias para sua sessão de perfil.

Para obter mais informações sobre os perfis, consulte Perfis do IAM e Criação de perfis do IAM no Guia do usuário do IAM.

Para criar um perfil que conceda acesso ao Athena, vincule as políticas gerenciadas do Athena ao perfil. Há duas políticas gerenciadas do Athena: `AmazonAthenaFullAccess` e `AWSQuicksightAthenaAccess`. Essas políticas concedem permissões ao Athena para consultar o Amazon S3 e para gravar os resultados das consultas em um bucket separado em seu nome. Para ver o conteúdo dessas políticas do Athena, consulte Politicas gerenciadas pela AWS para o Amazon Athena.

Para obter as etapas para vincular as políticas gerenciadas do Athena a um perfil, acesse Adicionar permissões de identidade do IAM (console) no Guia do usuário do IAM e adicione as políticas gerenciadas `AmazonAthenaFullAccess` e `AWSQuicksightAthenaAccess` ao perfil criado.

> **Note**
>
> Você pode precisar de permissões adicionais para acessar o conjunto de dados subjacente no Amazon S3. Se você não for o proprietário da conta ou tiver acesso restrito a um bucket, entre em contato com o proprietário do bucket para obter acesso usando uma política de bucket baseada em recursos ou entre em contato com o administrador da conta para obter acesso usando uma política baseada em perfis. Para ter mais informações, consulte Acesso ao Amazon S3. Se o conjunto de dados ou os resultados da consulta do Athena estiverem criptografados, talvez sejam necessárias outras permissões. Para ter mais informações, consulte Criptografia inativa.

Acessar o Athena

Você pode acessar o Athena usando o AWS Management Console, uma conexão JDBC ou ODBC, a API do Athena, a CLI do Athena, o AWS SDK ou o AWS Tools for Windows PowerShell.

- Para começar a usar o SQL do Athena com o console, consulte Conceitos básicos.
- Para começar a criar cadernos compatíveis com Jupyter e aplicações Apache Spark que usam Python, consulte Uso do Apache Spark no Amazon Athena.
- Para saber como usar drivers JDBC ou ODBC, consulte Conectar-se ao Amazon Athena com JDBC e Conectar-se ao Amazon Athena com ODBC.
- Para usar a API do Athena, consulte a Referência de API do Amazon Athena.
- Para usar a CLI, instale a AWS CLI e digite `aws athena help` na linha de comando para ver os comandos disponíveis. Para obter informações sobre os comandos disponíveis, consulte a Referência da linha de comando do Amazon Athena.
- Para usar o AWS SDK for Java 2.x, consulte a seção sobre o Athena da referência de API do AWS SDK for Java 2.x, os exemplos do Java V2 para o Athena no GitHub.com e o Guia do desenvolvedor do AWS SDK for Java 2.x.
- Para usar o AWS SDK for .NET, consulte o namespace `Amazon.Athena` na Referência de API do AWS SDK for .NET, os Exemplos do Athena no .NET no GitHub.com e o Guia do desenvolvedor do AWS SDK for .NET.

- Para usar o AWS Tools for Windows PowerShell, consulte a referência do cmdlet AWS Tools for PowerShell - Amazon Athena, a página do portal do AWS Tools for PowerShell e o Manual do usuário do AWS Tools for Windows PowerShell.
- Para obter informações sobre endpoints de serviço do Athena aos quais você pode se conectar de maneira programática, consulte Endpoints e cotas do Amazon Athena no Referência geral da Amazon Web Services.

Como usar o SQL do Athena

É possível usar o SQL do Athena para consultar os dados no Amazon S3 localmente ao usar o AWS Glue Data Catalog, um metastore do Hive externo ou consultas federadas usando uma variedade de conectores criados previamente para outras fontes de dados.

Você também pode:

- Conectar-se a ferramentas de business intelligence e outras aplicações usando os drivers JDBC e ODBC do Athena.
- Consultar logs de serviço da AWS.
- Consultar tabelas do Apache Iceberg, incluindo consultas de passagem de tempo e conjuntos de dados do Apache Hudi.
- Consultar dados geoespaciais.
- Consultar usando a inferência de machine learning do Amazon SageMaker.
- Consultar usando suas próprias funções definidas pelo usuário.
- Acelerar o processamento de consultas de tabelas altamente particionadas e automatizar o gerenciamento de partições usando a projeção de partições.

Tópicos

- Noções básicas sobre tabelas, bancos de dados e catálogos de dados
- Conceitos básicos
- Conectar a origens de dados
- Conectar-se ao Amazon Athena com drivers ODBC e JDBC
- Criar bancos de dados e tabelas
- Criar uma tabela a partir de resultados de consultas (CTAS)
- Referência de SerDe
- Executar consultas SQL usando o Amazon Athena
- Usar transações ACID do Athena
- Segurança do Amazon Athena
- Gerenciamento do workload
- Versionamento do mecanismo do Athena

- Referência do SQL para o Athena
- Solução de problemas no Athena
- Exemplos de código

Noções básicas sobre tabelas, bancos de dados e catálogos de dados

No Athena, os catálogos, os bancos de dados e as tabelas são contêineres para as definições de metadados que definem um esquema para os dados de origem subjacentes.

O Athena utiliza os seguintes termos para se referir às hierarquias de objetos de dados:

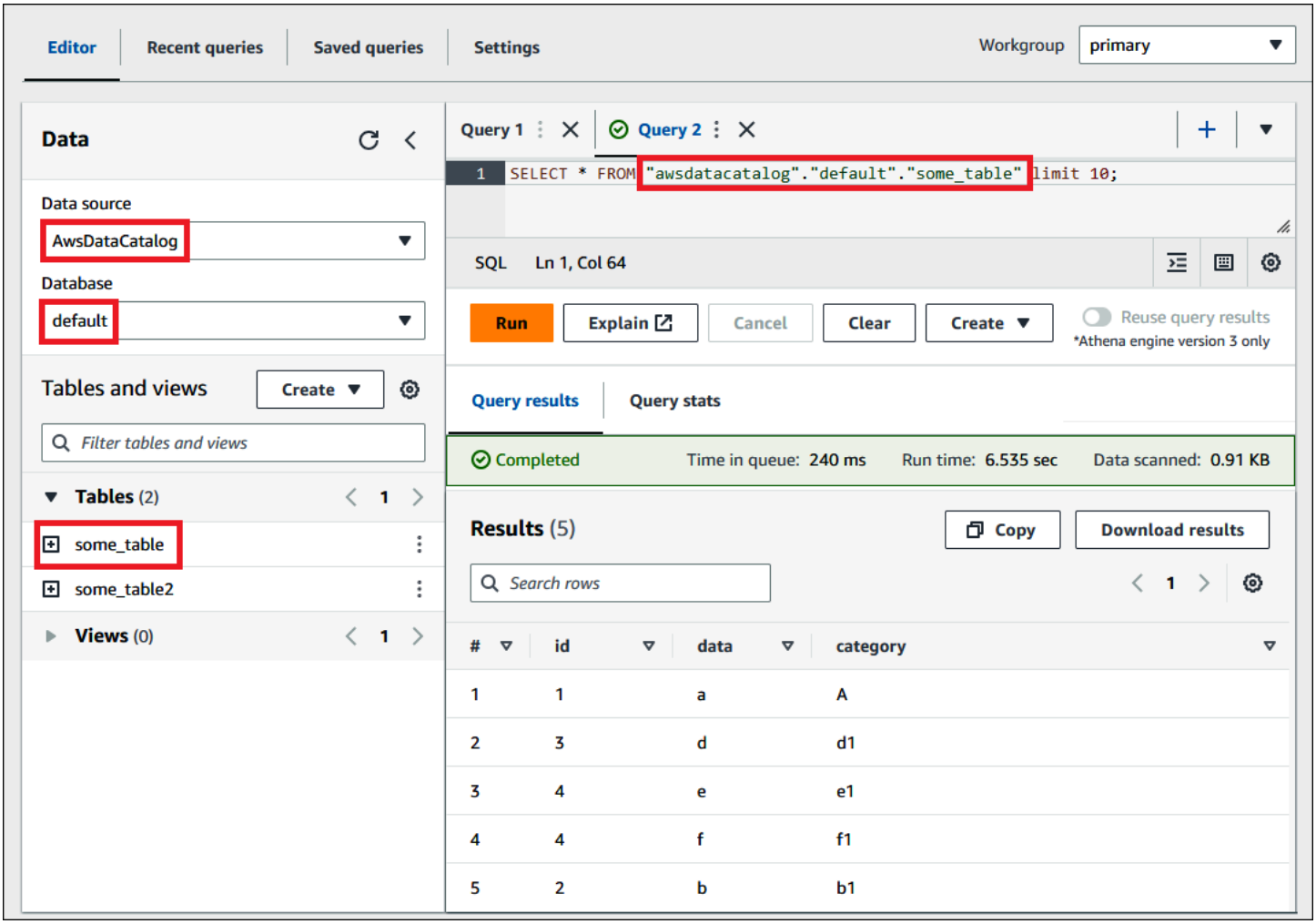
- Fonte de dados: um grupo de bancos de dados
- Banco de dados: um grupo de tabelas
- Tabela: dados organizados como um grupo de linhas ou colunas

Às vezes, esses objetos também são chamados por nomes alternativos, mas equivalentes, como:

- Às vezes uma fonte de dados é denominada catálogo.
- Às vezes um banco de dados é denominado esquema.

> **Note**
>
> A terminologia pode variar nas fontes de dados federadas usadas com o Athena. Para ter mais informações, consulte Qualificadores de nomes do Athena e de tabelas federadas.

O exemplo de consulta no console do Athena a seguir usa a fonte de dados `awsdatacatalog`, o banco de dados `default` e a tabela `some_table`.



Para cada conjunto de dados, deve existir uma tabela no Athena. Os metadados na tabela informam ao Athena onde os dados estão localizados no Amazon S3 e especificam a estrutura dos dados, por exemplo, nomes de coluna, tipos de dados e o nome da tabela. Os bancos de dados são um agrupamento lógico de tabelas e também mantêm somente metadados e informações do esquema de um conjunto de dados.

Para cada conjunto de dados que você deseja consultar, o Athena deve ter uma tabela subjacente que ele usará para obter e retornar os resultados das consultas. Portanto, antes de consultar os dados, uma tabela deve ser registrada no Athena. O registro ocorre quando você cria tabelas automática ou manualmente.

É possível criar uma tabela automaticamente usando um crawler do AWS Glue. Para obter mais informações sobre o AWS Glue e os crawlers, consulte Integração com o AWS Glue. Quando o AWS Glue cria uma tabela, ele a registra no próprio Catálogo de dados do AWS Glue. O Athena usa o

Catálogo de dados do AWS Glue para armazenar e recuperar esses metadados, usando-o quando você executa consultas para analisar o conjunto de dados subjacente.

Seja qual for o modo de criação da tabela, o processo de criação de tabelas registra o conjunto de dados no Athena. Esse registro ocorre no AWS Glue Data Catalog e permite que o Athena execute consultas nos dados. No editor de consultas do Athena, esse catálogo (ou fonte de dados) é referenciado com o rótulo `AwsDataCatalog`.

Após criar uma tabela, use as instruções SQL SELECT para consultá-la e obter os locais de arquivo específicos para seus dados de origem. Os resultados das consultas são armazenados no Amazon S3 no local de resultados de consultas especificado.

O Catálogo de dados do AWS Glue fica acessível em toda a sua conta da Amazon Web Services. Outros Serviços da AWS podem compartilhar o Catálogo de dados do AWS Glue, portanto, você pode ver os bancos de dados e as tabelas criados em toda a organização usando o Athena e vice-versa.

- Para criar uma tabela manualmente:
  - Use o console do Athena para executar o Create Table Wizard (Assistente de criação de tabela).
  - Use o console do Athena para escrever instruções DDL do Hive no editor de consultas.
  - Use a API ou a CLI do Athena para executar uma string de consulta SQL com instruções DDL.
  - Use o driver JDBC ou ODBC do Athena.

Quando você cria tabelas e bancos de dados manualmente, o Athena usa as instruções Data Definition Language (DDL – Linguagem de definição de dados) do HiveQL, como `CREATE TABLE`, `CREATE DATABASE` e `DROP TABLE` em segundo plano para criar tabelas e bancos de dados no AWS Glue Data Catalog.

Para começar, use um tutorial no console do Athena ou leia um guia detalhado na documentação do Athena.

- Para usar o tutorial no console do Athena, escolha o ícone de informações no canto superior direito do console e escolha a guia Tutorial.
- Para ver um tutorial detalhado sobre como criar uma tabela e gravar consultas no editor de consultas do Athena, consulte Conceitos básicos.

Conceitos básicos

Este tutorial orienta você a usar o Amazon Athena para consultar dados. Você criará uma tabela com base nos dados de exemplo armazenados no Amazon Simple Storage Service, consultará a tabela e verificará os resultados da consulta.

O tutorial usa recursos dinâmicos. Sendo assim, há cobrança pelas consultas que você executa. Não há cobrança pelos dados de exemplo no local usados neste tutorial, mas se você carregar seus próprios arquivos de dados no Amazon S3, haverá cobrança.

Pré-requisitos

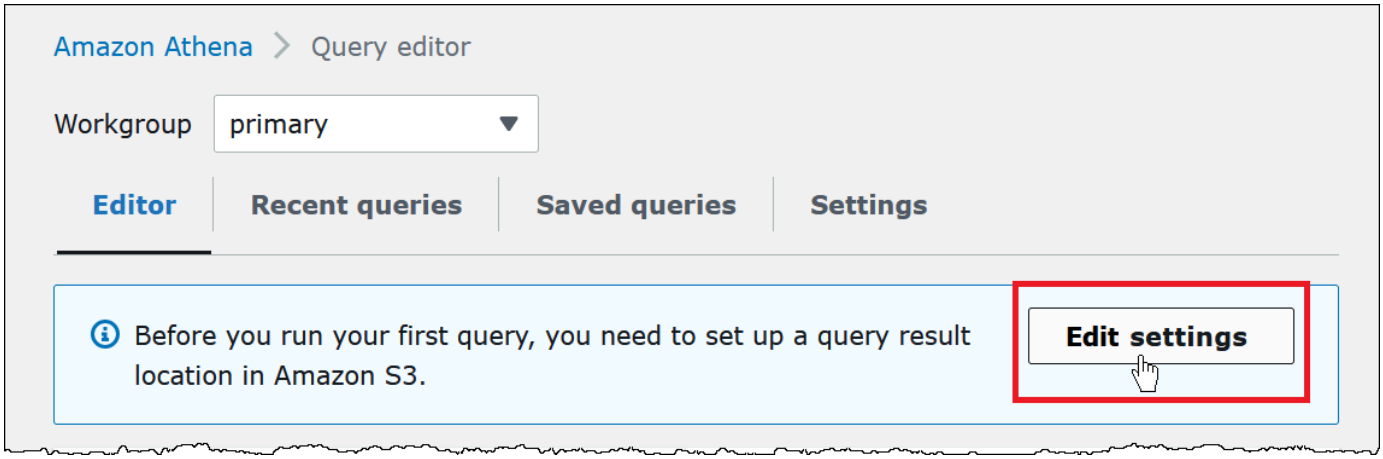
- Se ainda não tiver feito isso, cadastre-se para um Conta da AWS.
- Usando a mesma Região da AWS (por exemplo, Oeste dos EUA (Oregon)) e a conta que você usa no Athena, siga as etapas para criar um bucket no Amazon S3 para armazenar os resultados das consultas do Athena. Você configurará esse bucket para ser o local de saída da consulta.

Etapa 1: criar um banco de dados

Primeiro você precisa criar um banco de dados no Athena.

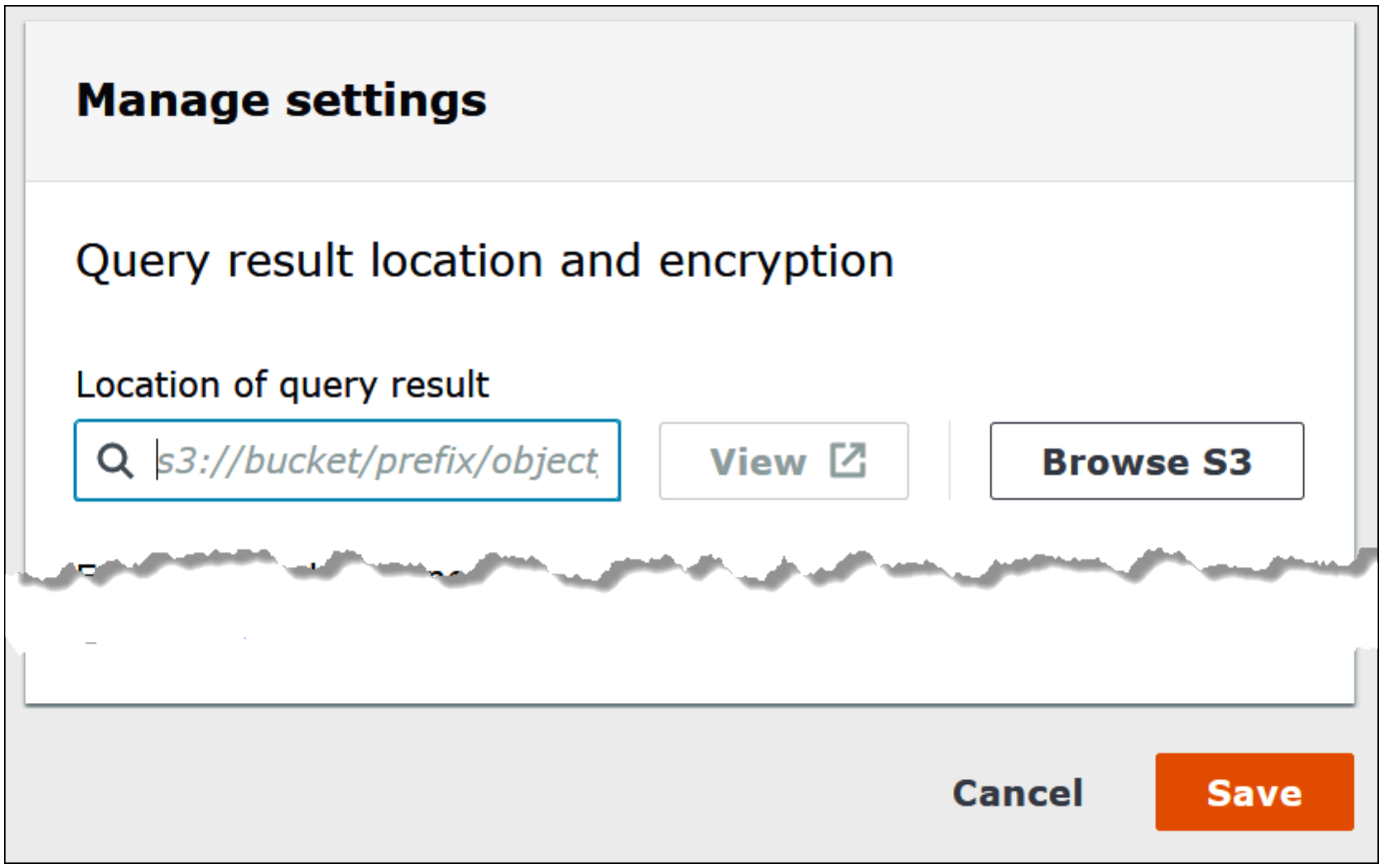
Para criar um banco de dados do Athena

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se esta for a primeira vez que você acessa o console do Athena em sua Região da AWS, escolha Explore the query editor (Explorar o editor de consultas) para abrir o editor de consultas. Do contrário, o Athena é aberto no editor de consultas.
3. Escolha Edit Settings (Editar configurações) para configurar um local para os resultados de consultas no Amazon S3.

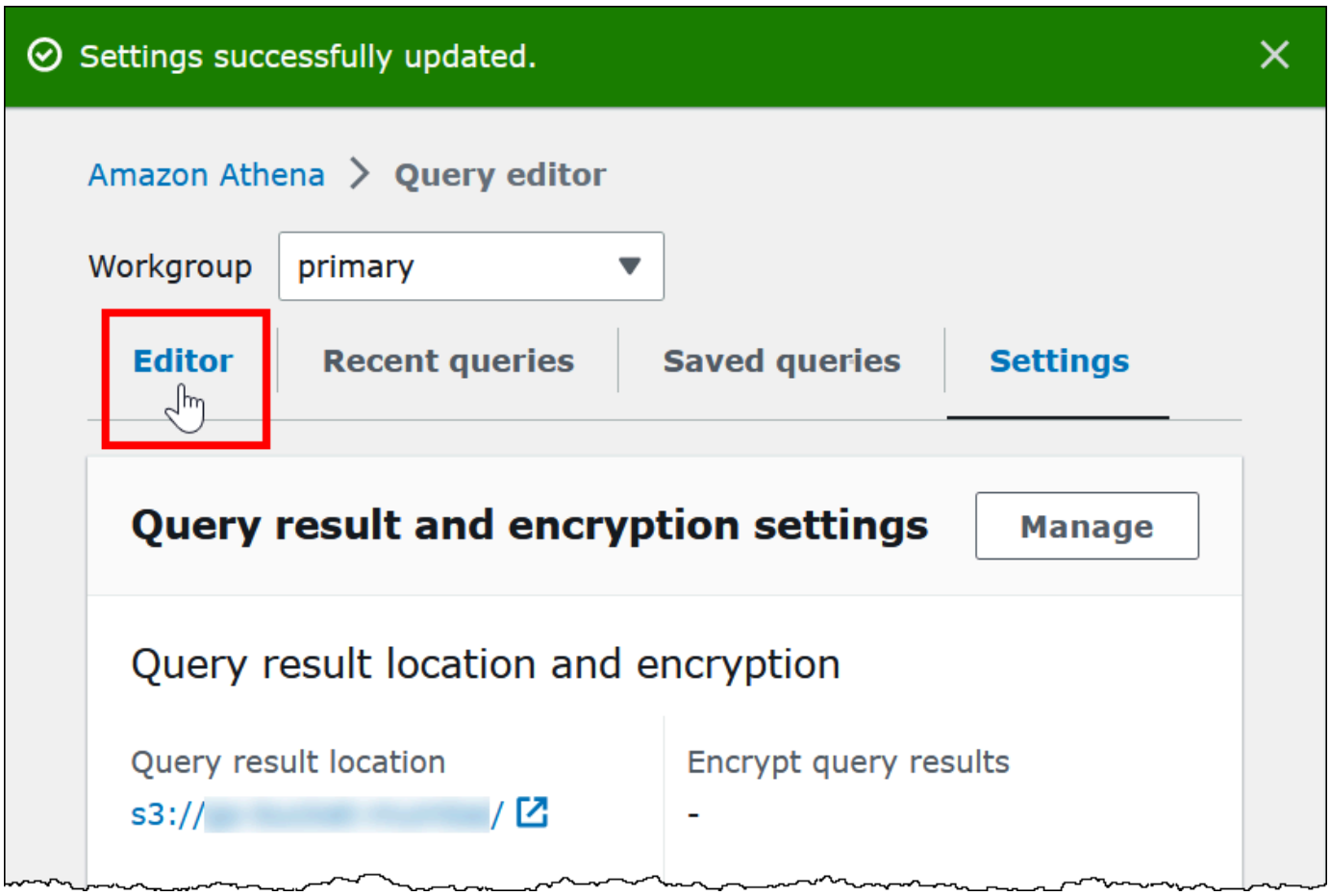


4. Em Manage settings (Gerenciar configurações), faça um dos seguintes procedimentos:

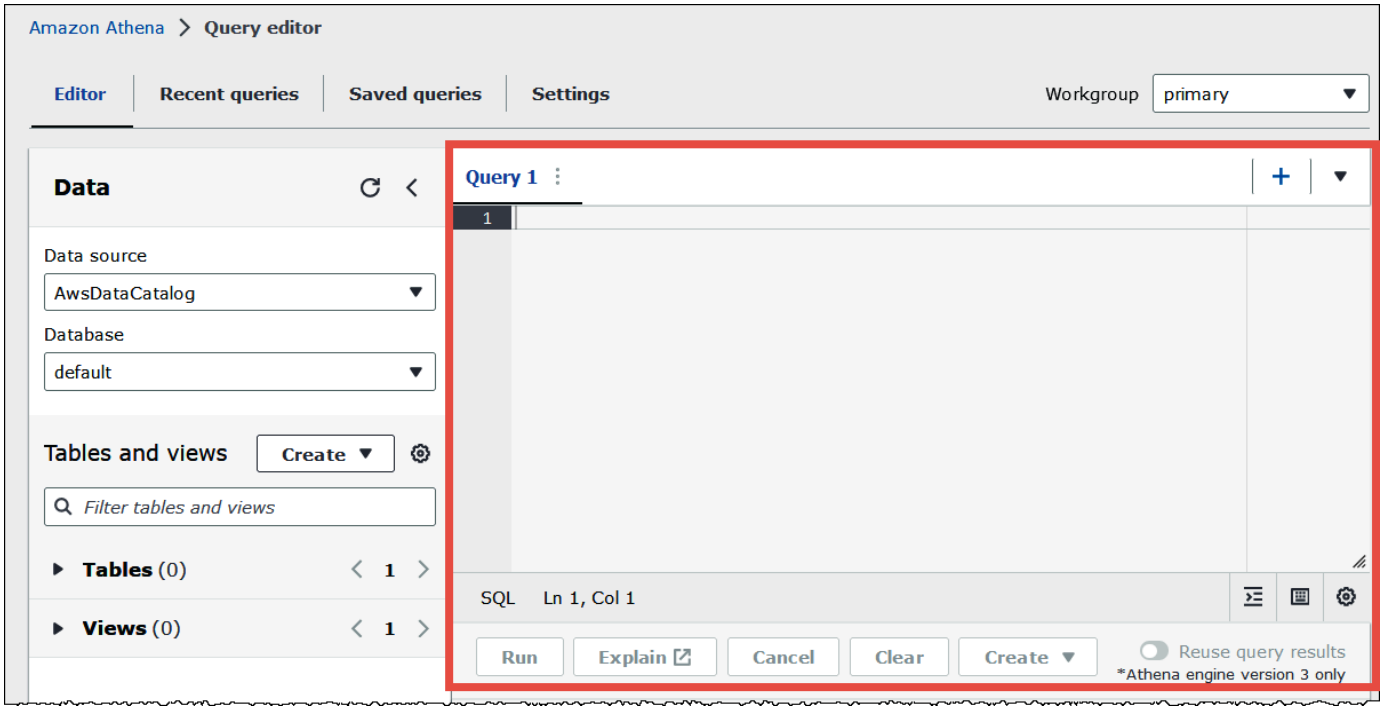
   - Na caixa de texto Query result location (Localização dos resultados da consulta), insira o caminho para o bucket criado no Amazon S3 para resultados de consultas. Adicione o prefixo `s3://` ao caminho.
   - Escolha Browse S3 (Navegar no S3), escolha o bucket do Amazon S3 que você criou na região atual e escolha Choose (Escolher).



5. Escolha Salvar.
6. Selecione Editor para alternar para o editor de consultas.



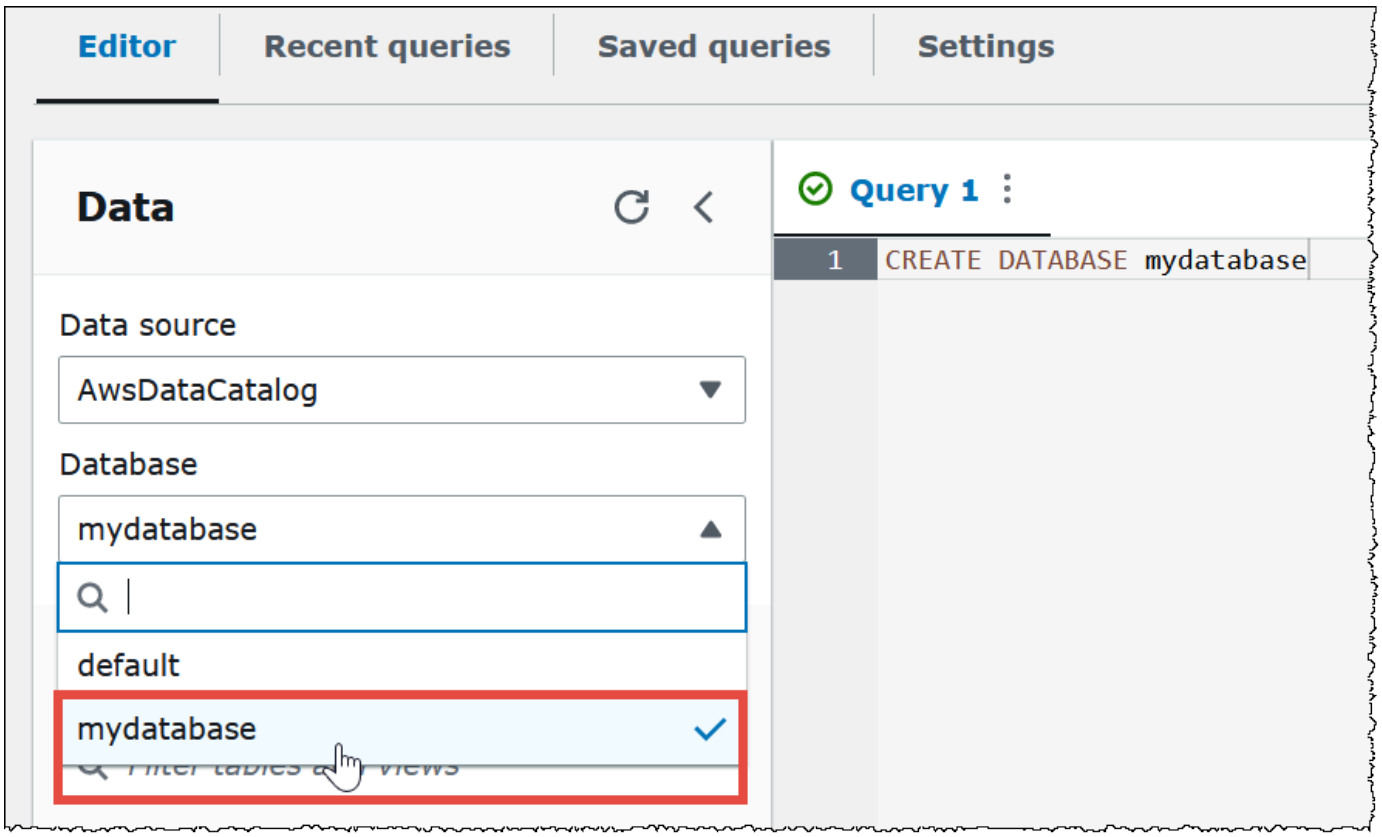
7. À direita do painel de navegação, você pode usar o editor de consultas do Athena para inserir e executar as consultas e instruções.



8. Para criar um banco de dados denominado `mydatabase`, insira a instrução CREATE DATABASE a seguir.

```
CREATE DATABASE mydatabase
```

9. Selecione Run (Executar) ou pressione **`Ctrl+ENTER`**.
10. Na lista Database (Banco de dados) à esquerda, escolha `mydatabase` para torná-lo seu banco de dados atual.



Etapa 2: Criar uma tabela

Agora que você tem um banco de dados, pode criar uma tabela do Athena para ele. A tabela criada será baseada nos dados de log de exemplo do Amazon CloudFront no local `s3://athena-examples-`*`myregion`*`/cloudfront/plaintext/`, em que *`myregion`* é a sua Região da AWS atual.

Os dados de log de exemplo estão no formato Tab-Separated Values (TSV – Valores separados por tabulação), o que significa que um caractere de tabulação é usado como delimitador para separar os campos. Veja no exemplo abaixo a aparência dos dados. Para legibilidade, as guias no trecho foram convertidas em espaços e o campo final foi reduzido.

```
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[...]
```

Para permitir que o Athena leia esses dados, você pode criar uma instrução `CREATE EXTERNAL TABLE` direta, semelhante à apresentada abaixo. A instrução que cria a tabela define as colunas que são mapeadas para os dados, especifica como os dados são delimitados e o local do Amazon S3 que contém os dados de exemplo. Observe que, como o Athena espera fazer a varredura de todos os arquivos em uma pasta, a cláusula `LOCATION` especifica um local de pasta do Amazon S3 e não um arquivo específico.

Não use esse exemplo ainda, pois ele tem uma limitação importante que será explicada em breve.

```
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/';
```

O exemplo cria uma tabela chamada `cloudfront_logs` e especifica um nome e tipo de dados para cada campo. Esses campos se tornam as colunas da tabela. Como `date` é uma palavra reservada, ela é escapada com caracteres de acento grave (`). `ROW FORMAT DELIMITED` significa que o Athena usará uma biblioteca padrão chamada LazySimpleSerDe para fazer o trabalho real de análise dos dados. O exemplo também especifica que os campos são separados por tabulação (`FIELDS TERMINATED BY '\t'`) e que cada registro no arquivo termina com um caractere de nova linha (`LINES TERMINATED BY '\n`). Por fim, a cláusula `LOCATION` especifica o caminho no Amazon S3 onde estão localizados os dados reais que serão lidos.

Se tiver seus próprios dados separados por tabulação ou vírgula, você poderá usar uma instrução `CREATE TABLE` como a do exemplo que acabamos de apresentar, desde que seus campos não contenham informações aninhadas. No entanto, se você tiver uma coluna como `ClientInfo`,

contendo informações aninhadas que usem um delimitador diferente, será necessário adotar outra abordagem.

Como extrair dados do campo ClientInfo

Observando os dados do exemplo, aqui está um exemplo completo da situação final do campo `ClientInfo`:

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

Como você pode ver, esse é um campo multivalor. Como a instrução `CREATE TABLE` do exemplo apresentado especifica guias como delimitadores de campo, ela não pode decompor em colunas separadas componentes separados dentro do campo `ClientInfo`. Portanto, é necessário empregar uma nova instrução `CREATE TABLE`.

Para criar colunas com base nos valores dentro do campo `ClientInfo`, você pode usar uma expressão regular (regex) que contenha grupos de regex. Os grupos de regex especificados tornam-se as colunas separadas da tabela. Para usar uma regex na instrução `CREATE TABLE`, use a sintaxe conforme mostrado a seguir. Essa sintaxe instrui o Athena a usar a biblioteca Regex SerDe e a expressão regular que você especificar.

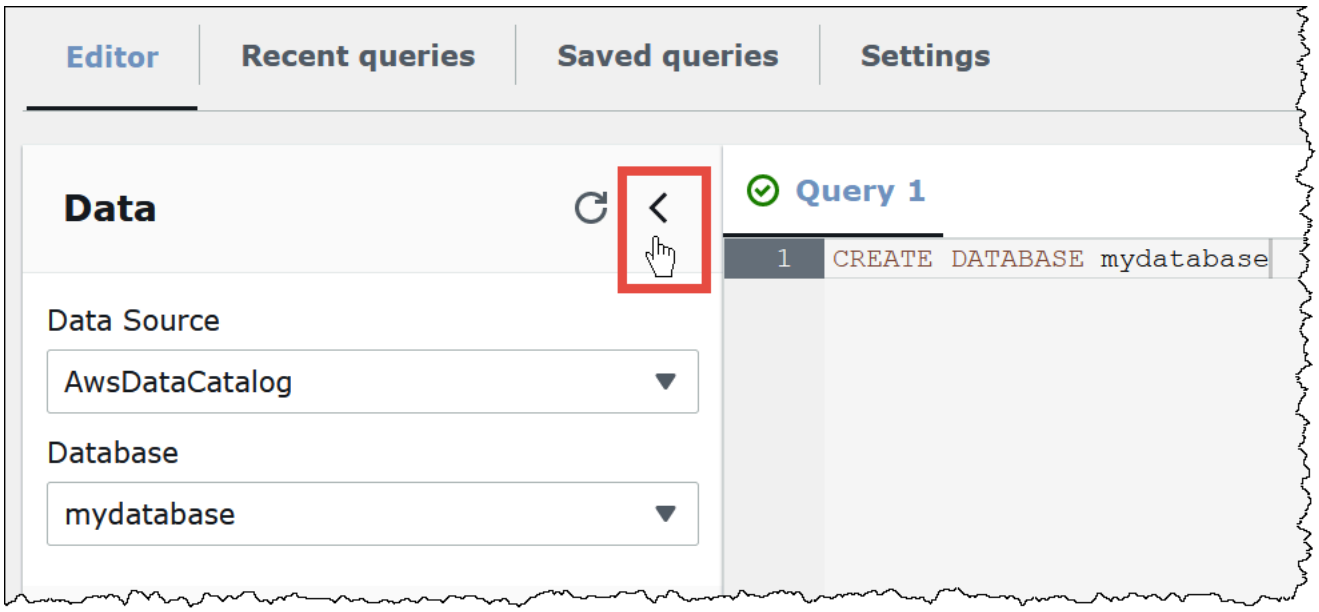
```
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
  WITH SERDEPROPERTIES ("input.regex" = "regular_expression")
```

As expressões regulares podem ser úteis para criar tabelas com base em dados complexos CSV ou TSV, mas podem ser difíceis de escrever e manter. No entanto, há outras bibliotecas que você pode usar para formatos como JSON, Parquet e ORC. Para obter mais informações, consulte SerDes e formatos de dados compatíveis.

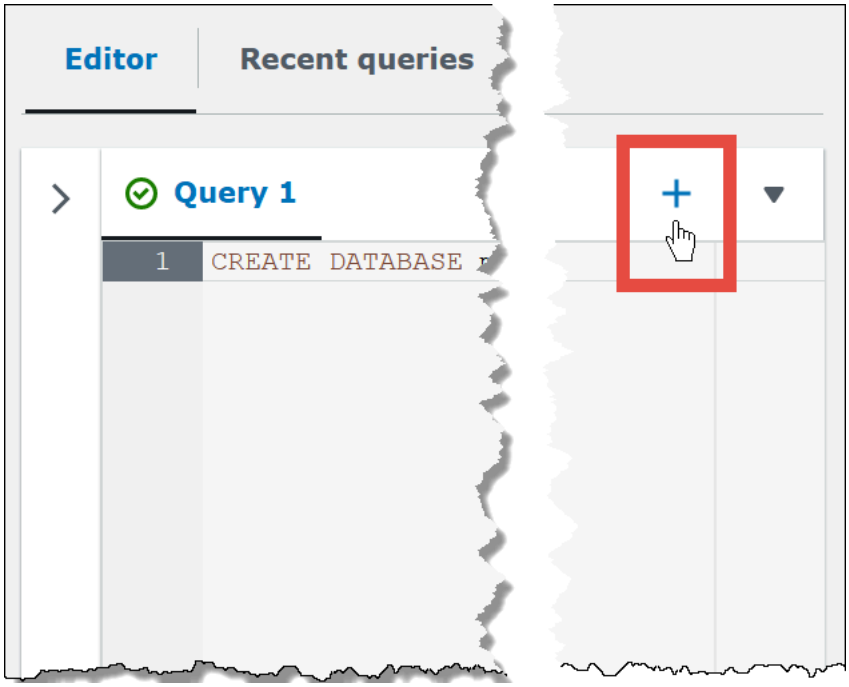
Agora você está pronto para criar a tabela no editor de consultas do Athena. A instrução `CREATE TABLE` e a regex já foram incluídas para você.

Para criar uma tabela no Athena

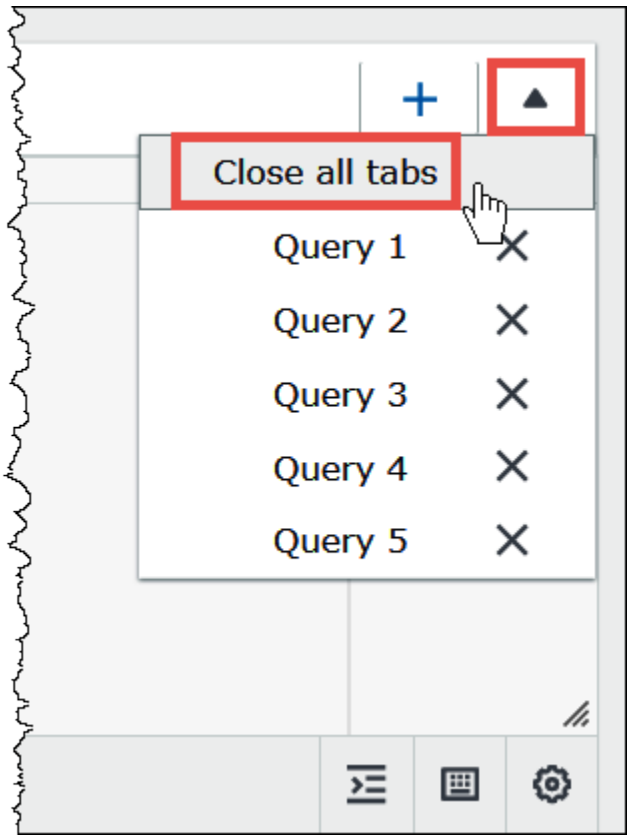
1. No painel de navegação, para Database (Banco de dados), certifique-se de que `mydatabase` esteja selecionado.
2. Para ter mais espaço no editor de consultas, escolha o ícone de seta para recolher o painel de navegação.



3. Para criar uma guia para um nova consulta, escolha o sinal de adição (+) no editor de consultas. É possível ter até dez guias de consulta abertas por vez.



4. Para fechar uma ou mais guias de consulta, escolha a seta ao lado do sinal de mais. Para fechar todas as guias de uma só vez, escolha a seta e escolha Close all tabs (Fechar todas as guias).



5. No painel de consultas, digite a instrução `CREATE EXTERNAL TABLE` a seguir. A regex divide as informações de sistema operacional, navegador e versão do navegador do campo `ClientInfo` nos dados de log.

```
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. Na instrução `LOCATION`, substitua *myregion* pela Região da AWS que você está usando (por exemplo, `us-west-1`).
7. Escolha Executar.

   A tabela `cloudfront_logs` é criada e aparece na lista de Tables (Tabelas) para o banco de dados `mydatabase`.

Etapa 3: consultar dados

Agora que você criou a tabela `cloudfront_logs` no Athena com base nos dados do Amazon S3, pode executar as consultas SQL na tabela e ver os resultados no Athena. Para obter mais informações sobre como usar o SQL no Athena, consulte Referência do SQL para o Athena.

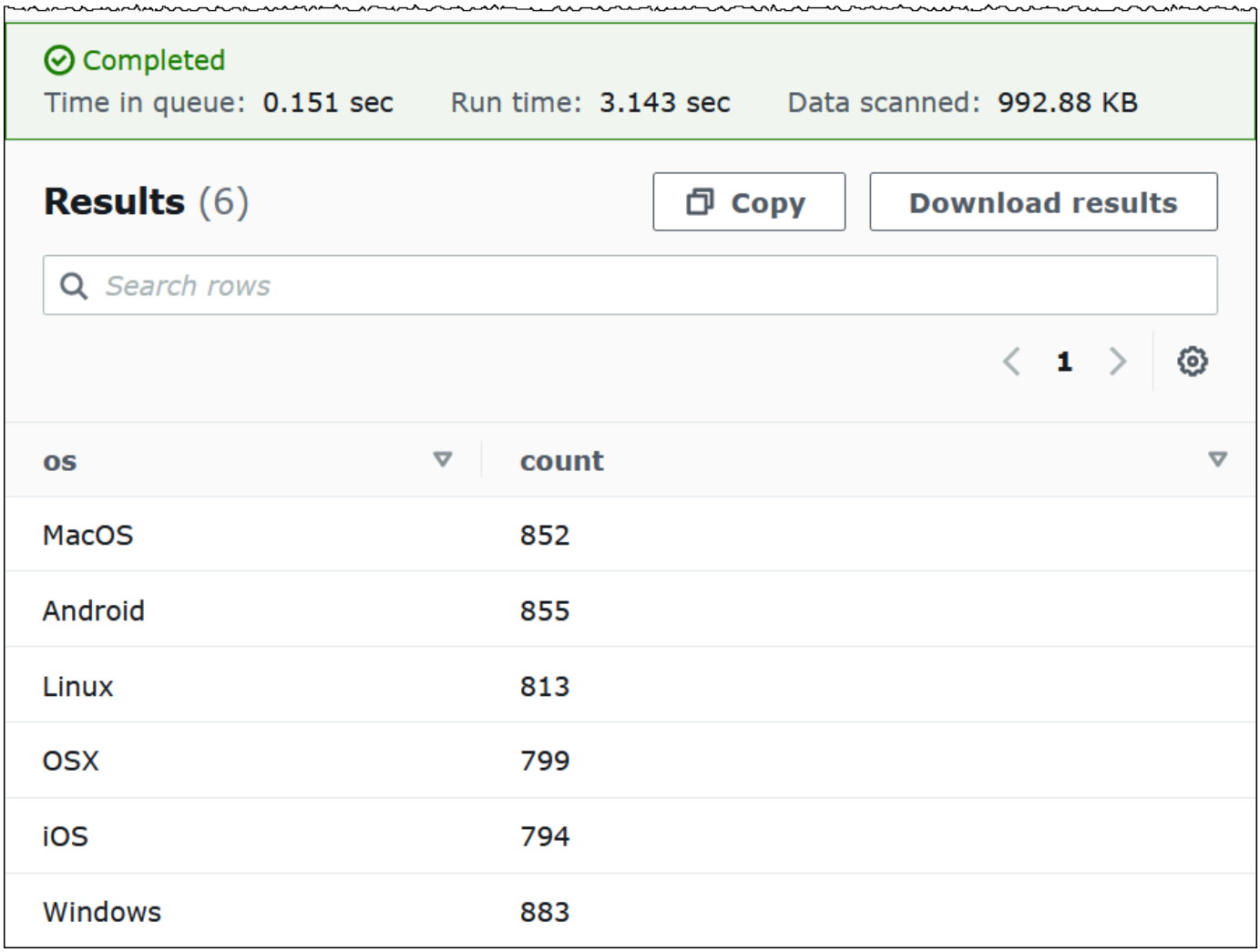
Para executar uma consulta

1. Clique no sinal de mais (+) para abrir uma nova guia de consulta e insira a instrução SQL a seguir no painel de consulta.

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

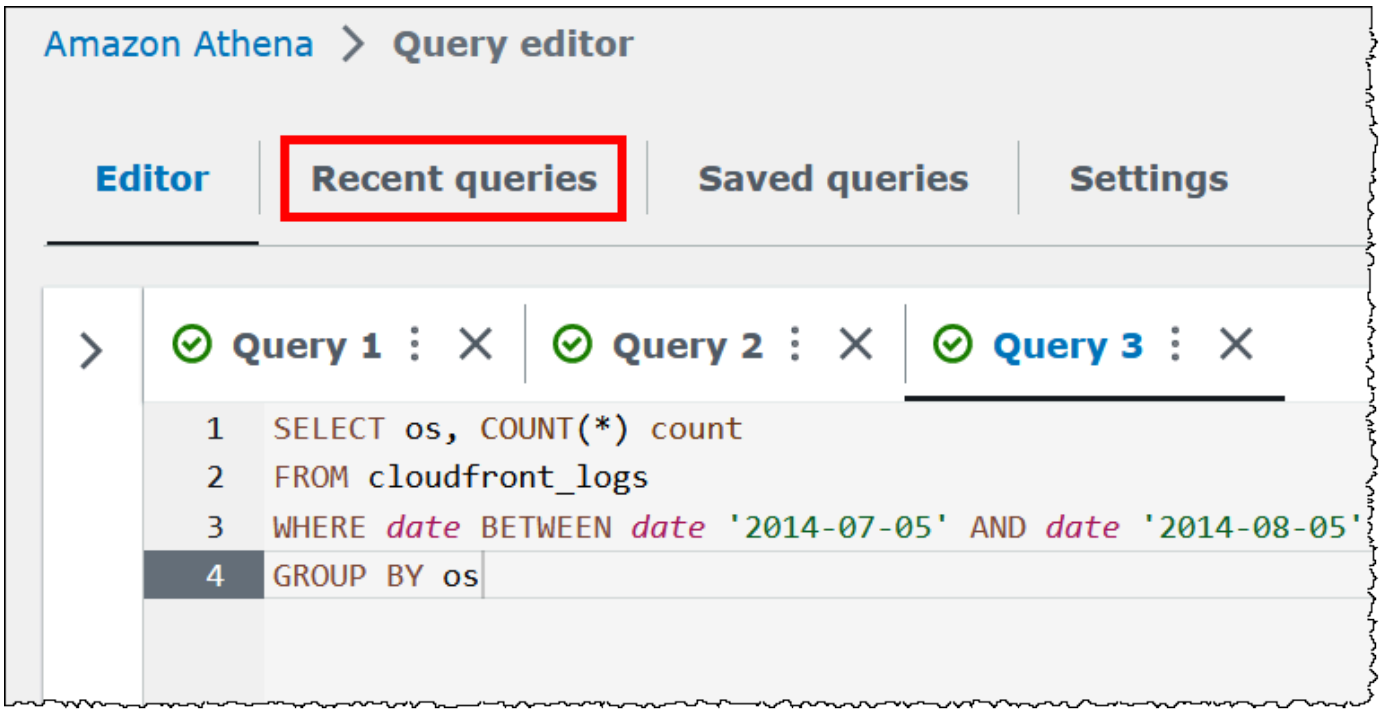
2. Escolha Executar.

   Os resultados são semelhantes ao seguintes:

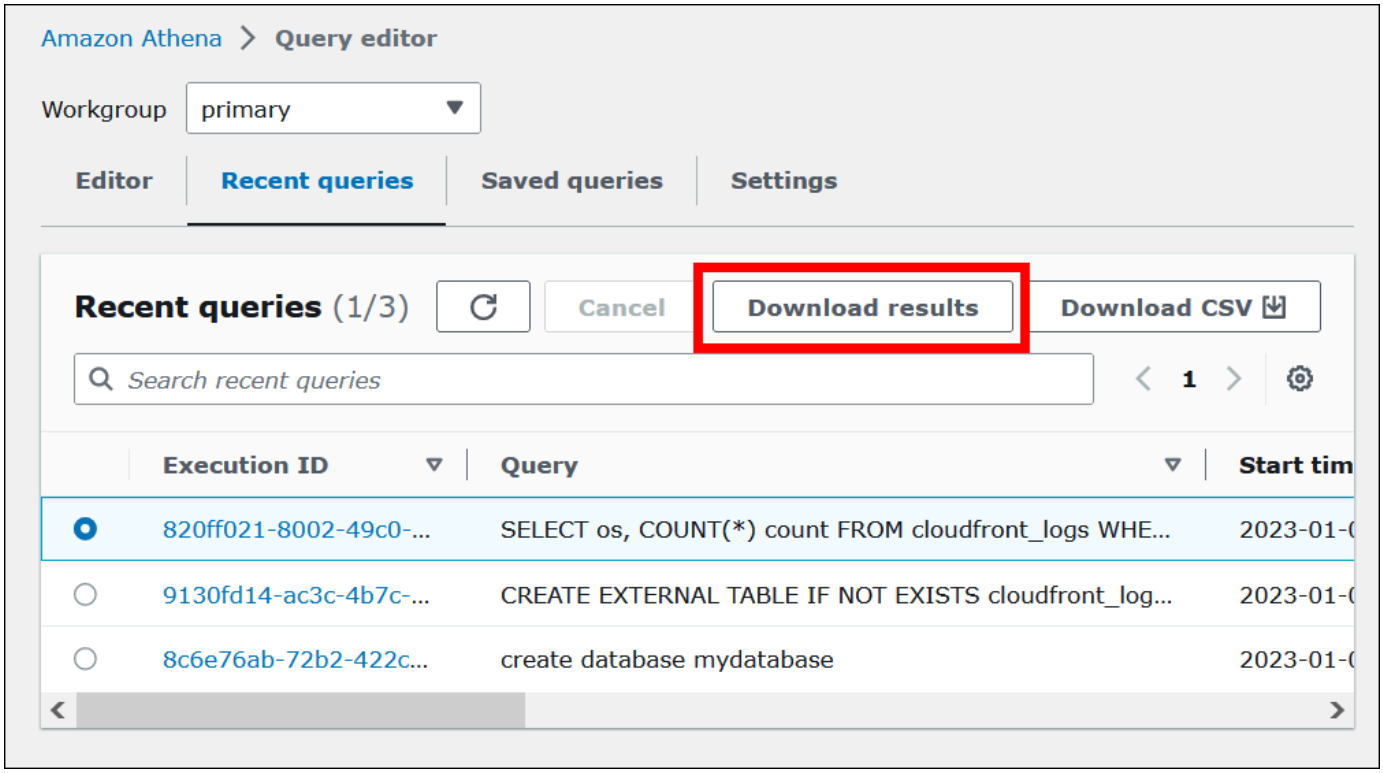


3. Para salvar os resultados da consulta em um arquivo `.csv`, escolha Download results (Baixar resultados).

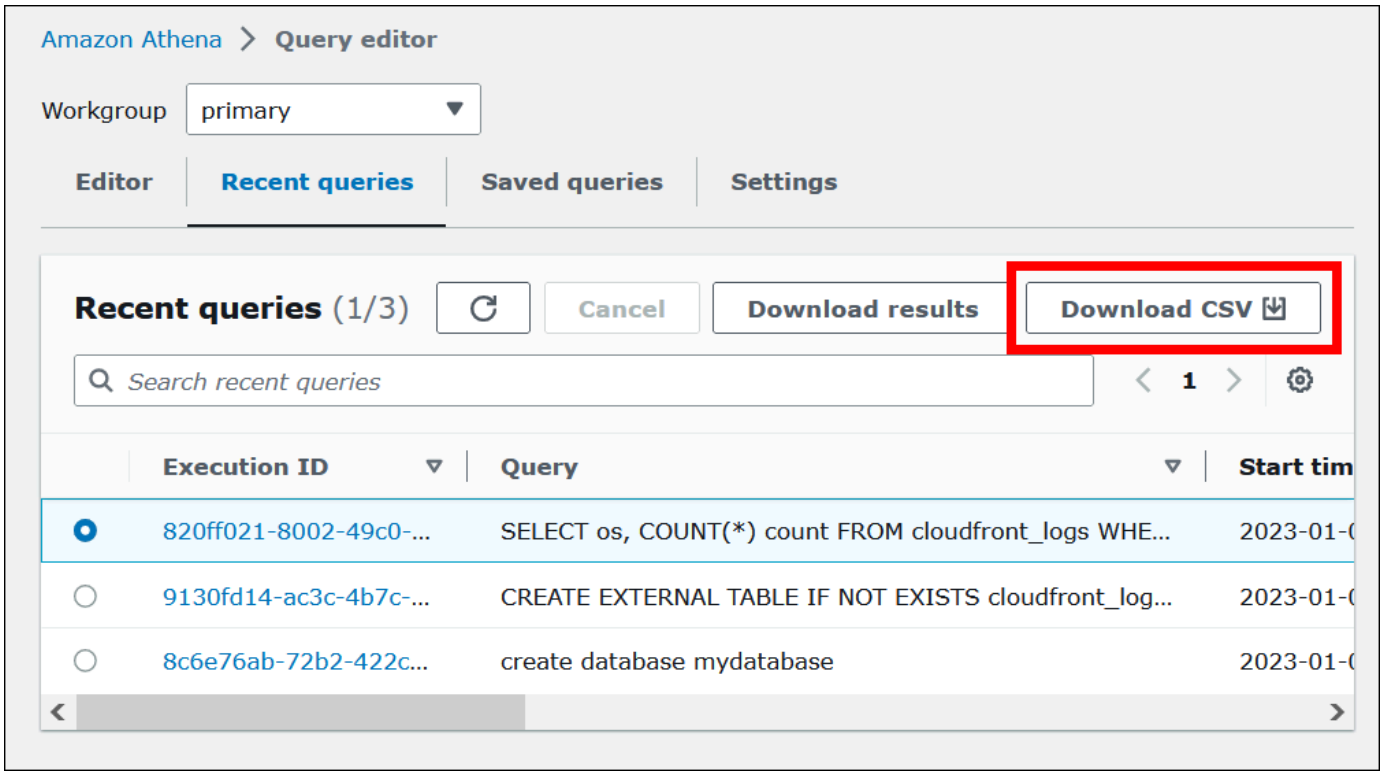
4. Para visualizar ou executar consultas anteriores, escolha Recent queries (Consultas recentes).



5. Para baixar os resultados de uma consulta anterior da guia Recent queries (Consultas recentes), selecione a consulta e escolha Download results (Baixar resultados). As consultas são retidas por 45 dias.



6. Para baixar uma ou mais strings de consulta SQL recentes como um arquivo CSV, escolha Download CSV (Baixar como CSV).



Para obter mais informações, consulte Trabalhar com resultados de consultas, consultas recentes e arquivos de saída.

Salvar consultas

É possível salvar as consultas criadas ou editadas no editor de consultas com um nome. O Athena armazena essas consultas na guia Saved queries (Consultas salvas). Você pode usar a guia Saved queries (Consultas salvas) para chamar novamente, executar, renomear ou excluir consultas salvas. Para obter mais informações, consulte Usar consultas salvas.

Atalhos de teclado e sugestões de digitação antecipada

O editor de consultas Athena fornece vários atalhos de teclado para ações como executar uma consulta, formatar uma consulta, operações de linha e localizar e substituir. Para obter mais informações e uma lista completa de atalhos, consulte Melhore a produtividade usando atalhos de teclado no editor de consultas Amazon Athena no AWSBlog Big Data.

O editor de consultas do Athena é compatível com sugestões de código de digitação antecipada para proporcionar uma experiência de criação de consultas mais rápida. Para ajudar você a escrever consultas SQL com maior precisão e eficiência, ele oferece os seguintes atributos:

- Enquanto você digita, as sugestões são exibidas em tempo real para palavras-chave, variáveis locais, trechos e itens de catálogo.
- Quando você digita um nome de banco de dados ou nome de tabela seguido por um ponto, o editor exibe, de maneira conveniente, uma lista de tabelas ou colunas para escolher.
- Quando você passa o mouse sobre uma sugestão de trecho, uma sinopse exibe uma breve visão geral da sintaxe e do uso do trecho.
- Para melhorar a legibilidade do código, as palavras-chave e as respectivas regras de destaque também foram atualizadas para se alinharem à sintaxe mais recente do Trino e do Hive.

Esse recurso está habilitado por padrão. Para ativar ou desativar o recurso, use Preferências do editor de código (ícone de engrenagem) na parte inferior direita da janela do editor de consultas.

Conectar a outras origens de dados

Este tutorial usou uma origem dos dados em formato CSV no Amazon S3. Para obter mais informações sobre como usar o Athena com o AWS Glue, consulte Usar o AWS Glue para se conectar a origens de dados no Amazon S3. É possível também conectar o Athena a uma variedade de origens de dados usando os drivers ODBC e JDBC, os metastores externos do Hive e os conectores de origem dos dados do Athena. Para obter mais informações, consulte Conectar a origens de dados.

Conectar a origens de dados

Você pode usar o Amazon Athena para consultar os dados armazenados em diferentes locais e formatos em um conjunto de dados. Esse conjunto de dados pode estar em CSV, JSON, Avro, Parquet ou outro formato.

As tabelas e os bancos de dados com os quais você trabalha no Athena para executar consultas são baseados em metadados. Metadados são dados sobre os dados subjacentes em seu conjunto de dados. A forma como esses metadados descrevem seu conjunto de dados é chamada de esquema. Por exemplo, um nome de tabela, os nomes de coluna na tabela e o tipo de dados de cada coluna são esquemas, salvos como metadados, que descrevem um conjunto de dados subjacente. No Athena, chamamos um sistema para organizar metadados de catálogo de dados ou de metastore. A

combinação de um conjunto de dados e o catálogo de dados que o descreve é chamada de fonte de dados.

A relação dos metadados com um conjunto de dados subjacente depende do tipo de fonte de dados com a qual você trabalha. Fontes de dados relacionais como MySQL, PostgreSQL e SQL Server integram totalmente os metadados ao conjunto de dados. Nesses sistemas, os metadados são gravados com maior frequência quando os dados são gravados. Outras fontes de dados, como aquelas criadas usando o Hive, permitem definir metadados em tempo real ao ler o conjunto de dados. O conjunto de dados pode estar em uma variedade de formatos, por exemplo, CSV, JSON, Parquet ou Avro.

O Athena tem compatibilidade nativa com o AWS Glue Data Catalog. O AWS Glue Data Catalog é um catálogo de dados desenvolvido com base em outros conjuntos e origens de dados, como Amazon S3, Amazon Redshift e Amazon DynamoDB. Você também pode conectar o Athena a outras origens de dados usando uma variedade de conectores.

Tópicos

- Integração com AWS Glue
- Usar o conector de dados do Athena para metastore externo do Hive
- Usar a consulta federada do Amazon Athena
- Políticas do IAM de acesso a catálogos de dados
- Gerenciar origens de dados
- Usar o Amazon DataZone no Athena

Integração com AWS Glue

O AWS Glue é um AWS service (Serviço da AWS) Extract, Transform, and Load (ETL – Extrair, transformar e carregar) totalmente gerenciado. Um dos seus recursos principais é analisar e categorizar dados. Você pode usar os crawlers do AWS Glue para inferir automaticamente o esquema de banco de dados e de tabela dos dados no Amazon S3 e armazenar os metadados associados no AWS Glue Data Catalog.

O Athena usa o AWS Glue Data Catalog para armazenar e recuperar metadados de tabela para os dados do Amazon S3 em sua conta da Amazon Web Services. Os metadados da tabela permitem que o mecanismo de consulta do Athena saiba como localizar, ler e processar os dados que você deseja consultar.

Para criar o esquema de banco de dados e de tabela no AWS Glue Data Catalog, você pode executar um crawler do AWS Glue no Athena em uma origem dos dados ou executar consultas de Data Definition Language (DDL – Linguagem de definição de dados) diretamente no editor de consultas do Athena. Depois disso, usando o esquema de banco de dados e de tabela que você criou, será possível usar as consultas de Data Manipulation (DML – Manipulação de dados) no Athena para consultar os dados.

Agora é possível registrar um AWS Glue Data Catalog de uma conta diferente da sua. Depois de configurar as permissões do IAM necessárias para o AWS Glue, você poderá usar o Athena para executar consultas entre contas. Para ter mais informações, consulte Acesso aos catálogos de dados do AWS Glue entre contas.

Para obter mais informações sobre o AWS Glue Data Catalog, consulte Data Catalog e crawlers no AWS Glue no Guia do desenvolvedor do AWS Glue.

São feitas cobranças separadas pelo uso do AWS Glue. Para obter mais informações, consulte Preços do AWS Glue.

Tópicos

- Usar o AWS Glue para se conectar a origens de dados no Amazon S3
- Registrar um AWS Glue Data Catalog de outra conta
- Práticas recomendadas ao usar o Athena com o AWS Glue
- Usar a AWS CLI para recriar um banco de dados do AWS Glue e suas tabelas

Usar o AWS Glue para se conectar a origens de dados no Amazon S3

O Athena pode se conectar aos dados armazenados no Amazon S3 usando o AWS Glue Data Catalog para armazenar metadados, como nomes de tabela e de coluna. Depois que a conexão é feita, os bancos de dados, as tabelas e as visualizações são exibidas no editor de consultas do Athena.

Para definir informações de esquemas a serem usadas pelo AWS Glue, é possível criar um crawler do AWS Glue para recuperar as informações automaticamente ou adicionar manualmente uma tabela e inserir as informações do esquema.

Criar um crawler do AWS Glue

É possível criar um crawler começando no console do Athena e usando o console do AWS Glue de forma integrada. Ao criar o crawler, você especifica um local de dados no Amazon S3 para crawling.

Para criar um crawler no AWS Glue começando do console do Athena

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. No editor de consultas, ao lado de Tabelas e visualizações, escolha Criar e, em seguida, selecione Crawler do AWS Glue.
3. No console do AWS Glue, na página Add crawler (Adicionar crawler), siga as etapas para criar um crawler. Para obter mais informações, consulte Usar crawlers do AWS Glue neste guia e Como preencher o AWS Glue Data Catalog no Guia do desenvolvedor do AWS Glue.

> **Note**
>
> O Athena não reconhece os padrões de exclusão que você especifica para um crawler do AWS Glue. Por exemplo, se você tem um bucket do Amazon S3 com os arquivos `.csv` e `.json` e exclui os arquivos `.json` do crawler, o Athena consulta os dois grupos de arquivos. Para evitar isso, coloque os arquivos que você deseja excluir em um local diferente.

Adicionar uma tabela usando um formulário

O procedimento a seguir mostra como usar o console do Athena para adicionar uma tabela usando o formulário Create Table From S3 bucket data (Criar tabela a partir de bucket do S3).

Como adicionar uma tabela e inserir informações de esquema usando um formulário

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. No editor de consultas, ao lado de Tables and views (Tabelas e visualizações), escolha Create (Criar) e, em seguida, escolha S3 bucket data (Dados do bucket do S3).
3. No formulário Create Table From S3 bucket data (Criar tabela a partir de dados de bucket do S3), em Table name (Nome da tabela), insira um nome para a tabela.
4. Em Database configuration (Configuração do banco de dados), escolha um banco de dados existente ou crie um.
5. Em Location of Input Data Set (Local do conjunto de dados de entrada), especifique o caminho no Amazon S3 para a pasta que contém o conjunto de dados que você deseja processar. Não inclua um nome de arquivo no caminho. O Athena verifica todos os arquivos na pasta especificada. Se seus dados já estiverem particionados (p. ex.,

s3://DOC-EXAMPLE-BUCKET/logs/year=2004/month=12/day=11/), insira somente o caminho base (p. ex., s3://DOC-EXAMPLE-BUCKET/logs/).

6. Em Data format (Formato de dados), escolha entre as seguintes opções:

   - Em Table type (Tipo de tabela), escolha Apache Hive, Apache Iceberg ou Delta Lake. O Athena usa o tipo de tabela Apache Hive como padrão. Para obter informações sobre como consultar tabelas do Apache Iceberg no Athena, consulte Usar tabelas do Apache Iceberg. Para obter informações sobre como usar tabelas do Delta Lake no Athena, consulte Consultar tabelas do Linux Foundation Delta Lake.
   - Em File format (Formato de arquivo), escolha o formato de arquivo ou log dos seus dados.
     - Na opção Text File with Custom Delimiters (Arquivo de texto com delimitadores personalizados), especifique um Field terminator (Terminador de campo) (ou seja, um delimitador de coluna). Opcionalmente, você pode especificar um terminador de coleção que marque o fim de um tipo de matriz ou um terminador de coleção que marque o fim de um tipo de dados de mapa.
   - Biblioteca SerDe: uma biblioteca SerDe (serializador-desserializador) analisa um formato de dados específico para que o Athena possa criar uma tabela para ele. Para a maioria dos formatos, uma biblioteca SerDe padrão é escolhida para você. Para os formatos a seguir, escolha uma biblioteca de acordo com seus requisitos:
     - Apache Web Logs: escolha a biblioteca RegexSerDe ou GrokserDe. Para RegexSerDe, forneça uma expressão regular na caixa Regex definition (Definição Regex). Para GrokserDe, forneça uma série de expressões regulares nomeadas para a propriedade SerDe `input.format`. As expressões regulares nomeadas são mais fáceis de ler e manter do que as expressões regulares. Para ter mais informações, consulte Consultar logs do Apache armazenados no Amazon S3.
     - CSV: escolha LazySimpleSerDe se seus dados separados por vírgula não contiverem valores entre aspas duplas ou se usarem o formato `java.sql.Timestamp`. Escolha OpenCSVSerDe se os dados incluírem aspas ou usarem o formato numérico UNIX para `TIMESTAMP` (p. ex., `1564610311`). Para obter mais informações, consulte LazySimpleSerDe para arquivos CSV, TSV e com delimitação personalizada e OpenCSVSerDe para processar CSV.
     - JSON: escolha a biblioteca OpenX ou Hive JSON SerDe. Os dois formatos esperam que cada documento JSON esteja em uma única linha de texto e que os campos não sejam separados por caracteres de nova linha. O OpenX SerDe oferece algumas propriedades

adicionais. Para obter mais informações sobre essas propriedades, consulte OpenX JSON SerDe. Para obter mais informações sobre o Hive SerDe, consulte Hive JSON SerDe.

Para obter mais informações sobre como usar as bibliotecas SerDe no Athena, consulte SerDes e formatos de dados compatíveis.

7. Em SerDe properties (Propriedades de SerDe), adicione, edite ou remova propriedades e valores de acordo com a biblioteca SerDe que você estiver usando e seus requisitos.

   - Para adicionar uma propriedade SerDe, escolha Add SerDe property (Adicionar propriedade SerDe).
   - No campo Name (Nome), insira o nome da propriedade.
   - No campo Value (Valor), insira um valor para a propriedade.
   - Para remover uma propriedade SerDe, escolha Remove (Remover).

8. Em Table properties (Propriedades da tabela), escolha ou edite as propriedades da tabela de acordo com seus requisitos.

   - Em Propriedades da tabela (Compactação de gravação), escolha uma opção de compactação. A disponibilidade da opção de compactação de gravação e das opções de compactação disponíveis depende do formato dos dados. Para ter mais informações, consulte Suporte a compactação no Athena.
   - Para Encryption (Criptografia), selecione Encrypted data set (Conjunto de dados criptografados) se os dados subjacentes estiverem criptografados no Amazon S3. Essa opção define a propriedade `has_encrypted_data` da tabela como verdadeira na instrução `CREATE TABLE`.

9. Em Column details (Detalhes da coluna), insira os nomes e os tipos de dados das colunas que você deseja adicionar à tabela.

   - Para adicionar mais colunas uma de cada vez, escolha Add a column (Adicionar uma coluna).
   - Para adicionar rapidamente mais colunas, escolha Bulk add columns (Adicionar colunas em massa). Na caixa de texto, insira uma lista separada por vírgulas de colunas no formato *`column_name data_type`*, *`column_name data_type`*[, …] e escolha Add (Adicionar).

10. (Opcional) Em Partition details (Detalhes da partição), adicione um ou mais nomes de colunas e tipos de dados. O particionamento mantém os dados relacionados juntos com base nos valores das colunas e pode ajudar a reduzir a quantidade de dados digitalizados por consulta. Para obter informações sobre particionamento, consulte Particionar dados no Athena.

11. (Opcional) Em Bucketing, você pode especificar uma ou mais colunas que tenham linhas que você deseja agrupar e, em seguida, colocar essas linhas em vários buckets. Isso permite que você consulte somente o bucket que deseja ler quando o valor das colunas agrupadas for especificado.

    - Para Buckets, selecione uma ou mais colunas que tenham um grande número de valores exclusivos (p. ex., uma chave primária) e que sejam frequentemente usadas para filtrar os dados em suas consultas.
    - Em Number of buckets (Número de buckets), insira um número que permita que os arquivos tenham o tamanho ideal. Para obter mais informações, consulte Top 10 Performance Tuning Tips for Amazon Athena (As 10 melhores dicas para ajustar o desempenho do Amazon Athena) no AWS Big Data Blog.
    - Para especificar suas colunas agrupadas, a instrução `CREATE TABLE` usará a seguinte sintaxe:

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

    > **Note**
    >
    > A opção Bucketing (Agregar em bucket) não está disponível para o tipo de tabela do Iceberg.

12. A caixa Preview table query (Previsualizar consulta da tabela) mostra a instrução `CREATE TABLE` gerada pelas informações inseridas no formulário. A instrução de visualização não pode ser editada diretamente. Para alterar a instrução, modifique os campos no formulário acima da visualização ou crie a instrução diretamente no editor de consultas em vez de usar o formulário.
13. Selecione Create table (Criar tabela) para executar a instrução gerada no editor de consultas e criar a tabela.

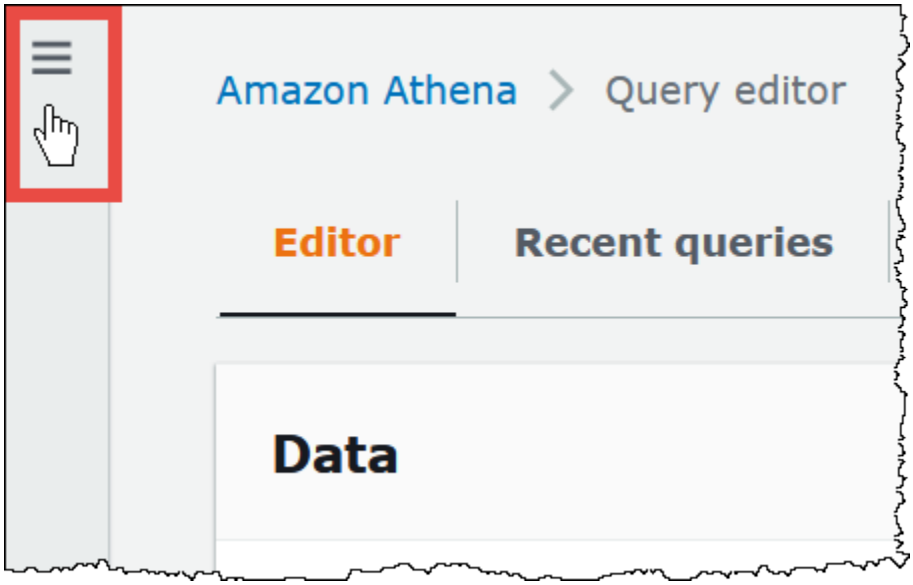
Registrar um AWS Glue Data Catalog de outra conta

Você pode usar o recurso de catálogo do AWS Glue entre contas do Athena para registrar um catálogo do AWS Glue de outra conta sua. Depois de configurar as permissões do IAM necessárias para o AWS Glue e registrar o catálogo como um recurso `DataCatalog` no Athena, você poderá usar o Athena para executar consultas entre contas. Para obter informações sobre como configurar as permissões necessárias, consulte Acesso aos catálogos de dados do AWS Glue entre contas.

O procedimento a seguir mostra como usar o console do Athena para configurar um AWS Glue Data Catalog em uma conta da Amazon Web Services diferente da sua origem dos dados.

Para registrar um AWS Glue Data Catalog de outra conta

1. Siga as etapas em Acesso aos catálogos de dados do AWS Glue entre contas para garantir que você tenha permissões para consultar o catálogo de dados na outra conta.
2. Abra o console do Athena em https://console.aws.amazon.com/athena/.
3. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.



4. Escolha Data sources (Origens de dados).
5. No canto superior direito do console, escolha Create data source (Criar fonte de dados).
6. Na página Choose a data source (Escolher uma fonte de dados), para Data sources (Fontes de dados), escolha S3 -AWS Glue Data Catalog e escolha Next (Próximo).
7. Na página Inserir detalhes da fonte de dados, na seção AWS Glue Data Catalog, para Escolher um AWS Glue Data Catalog, escolha AWS Glue Data Catalog em outra conta.
8. Em Data source details (Detalhes da origemdos dados), forneça as seguintes informações:

   - Data source name (Nome da origem dos dados): insira o nome que você deseja usar em suas consultas SQL para fazer referência ao catálogo de dados na outra conta.
   - Description (Descrição): insira uma descrição do catálogo de dados na outra conta (opcional).

- Catalog ID (ID do catálogo): insira o ID de 12 dígitos da conta da Amazon Web Services à qual o catálogo de dados pertence. O ID da conta da Amazon Web Services é o ID do catálogo.

9. (Opcional) Para Tags, adicione pares de chave-valor que você queira associar com a origem dos dados. Para obter mais informações sobre tags, consulte Etiquetar recursos do Athena.
10. Escolha Próximo.
11. Na página Review and create (Revisar e criar), analise as informações fornecidas e selecione Create data source (Criar fonte de dados). A página Data source details (Detalhes da fonte de dados) lista os bancos de dados e etiquetas do catálogo de dados que você registrou.
12. Escolha Data sources (Origens de dados). O catálogo de dados que você registrou está listado na coluna Data source name (Nome da fonte de dados).
13. Para visualizar ou editar as informações do catálogo de dados, escolha o catálogo e selecione Actions (Ações), Edit (Editar).
14. Para excluir o novo catálogo de dados, escolha o catálogo e selecione Actions (Ações) Delete (Excluir).

Para saber mais, consulte Consultar entre contas AWS Glue Data Catalog usando o Amazon Athena no AWSBlog Big Data.

Práticas recomendadas ao usar o Athena com o AWS Glue

Ao usar o Athena com o AWS Glue Data Catalog, é possível usar o AWS Glue para criar bancos de dados e tabelas (esquema) para serem consultados no Athena ou usar o Athena para criar um esquema e usá-lo no AWS Glue e nos serviços relacionados. Este tópico apresenta considerações e melhores práticas durante o uso de um dos métodos.

Nos bastidores, o Athena usa o Trino para processar as instruções DML e o Hive para processar as instruções DDL que criam e modificam o esquema. Com essas tecnologias, existem algumas convenções que devem ser seguidas para que o Athena e o AWS Glue funcionem bem juntos.

Neste tópico

- Nomes de bancos de dados, tabelas e colunas
- Usar crawlers do AWS Glue
  - Programar um crawler para manter a sincronização entre o AWS Glue Data Catalog e o Amazon S3

    - Usar várias origens de dados com crawlers
    - Sincronizar o esquema da partição para evitar “HIVE_PARTITION_SCHEMA_MISMATCH”
    - Atualizar metadados de tabelas
- Trabalhar com arquivos CSV
    - Dados CSV entre aspas
    - Arquivos CSV com cabeçalhos
- Indexação e filtragem de partições do AWS Glue
- Trabalhar com dados geoespaciais
- Usar trabalhos do AWS Glue para ETL com o Athena
    - Criar tabelas usando o Athena para trabalhos ETL do AWS Glue
    - Com usar trabalhos de ETL para otimizar a performance da consulta
    - Converter tipos de dados SMALLINT e TINYINT em INT ao converter em ORC
    - Automatizar trabalhos do AWS Glue para ETL

Nomes de bancos de dados, tabelas e colunas

Ao criar um esquema no AWS Glue para consulta no Athena, considere o seguinte:

- Os caracteres aceitáveis para nomes de bancos de dados, nomes de tabelas e nomes de colunas no AWS Glue devem ser strings em UTF-8. A string não pode ter menos do que 1 ou mais de 255 bytes de comprimento. Os caracteres que podem ser usados incluem espaços e são definidos pelo seguinte padrão de string de linha única:

```
[\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\t]*
```

- Atualmente, o padrão regex AWS Glue permite a adição de espaços iniciais ao início dos nomes. Como pode ser difícil detectar esses espaços iniciais e eles podem causar problemas de usabilidade após a criação, evite criar nomes de objetos que contenham espaços iniciais.
- Se você usar um modelo AWS::Glue::Database do AWS CloudFormation para criar um banco de dados do AWS Glue e não especificar um nome de banco de dados, o AWS Glue gera automaticamente um nome de banco de dados no formato *resource_name–random_string* que não é compatível com o Athena.
- Você pode usar o Gerenciador de Catálogos do AWS Glue para renomear colunas, mas não nomes de tabelas ou nomes de banco de dados. Para resolver essa limitação, é necessário usar

uma definição do banco de dados antigo para criar um banco de dados com o novo nome. Em seguida, use as definições das tabelas do banco de dados antigo para recriar as tabelas no novo banco de dados. Para isso, você pode usar a AWS CLI ou o SDK do AWS Glue. Para obter as etapas, consulte Usar a AWS CLI para recriar um banco de dados do AWS Glue e suas tabelas.

Para obter mais informações sobre bancos de dados e tabelas no AWS Glue, consulte Bancos de dados e Tabelas no Guia do desenvolvedor do AWS Glue.

Usar crawlers do AWS Glue

Os crawlers do AWS Glue ajudam a descobrir o esquema para conjuntos de dados e registrá-los no catálogo de dados do AWS Glue. Os crawlers passam pelos dados e determinam o esquema. Além disso, o crawler pode detectar e registrar partições. Para obter mais informações, consulte Definir crawlers no Guia do desenvolvedor do AWS Glue. Tabelas de dados que foram rastreadas com sucesso podem ser consultadas no Athena.

> **Note**
>
> O Athena não reconhece os padrões de exclusão que você especifica para um crawler do AWS Glue. Por exemplo, se você tem um bucket do Amazon S3 com os arquivos `.csv` e `.json` e exclui os arquivos `.json` do crawler, o Athena consulta os dois grupos de arquivos. Para evitar isso, coloque os arquivos que você deseja excluir em um local diferente.

Programar um crawler para manter a sincronização entre o AWS Glue Data Catalog e o Amazon S3

Os crawlers do AWS Glue podem ser configurados para serem executados em uma programação ou sob demanda. Para obter mais informações, consulte Programações baseadas em hora para trabalhos e crawlers no Guia do desenvolvedor do AWS Glue.

Se você tiver dados que chegam a uma tabela particionada em um horário fixo, poderá configurar um crawler do AWS Glue para ser executado de acordo com a programação para detectar e atualizar as partições da tabela. Isso pode eliminar a necessidade de executar um comando `MSCK REPAIR` possivelmente longo e caro ou executar manualmente um comando `ALTER TABLE ADD PARTITION`. Para obter mais informações, consulte Partições de tabela no Guia do desenvolvedor do AWS Glue.

Usar várias origens de dados com crawlers

Quando um crawler do AWS Glue verifica o Amazon S3 e detecta vários diretórios, ele usa uma heurística para determinar onde a raiz de uma tabela está na estrutura do diretório e quais diretórios são partições da tabela. Em alguns casos, quando o esquema detectado em dois ou mais diretórios é semelhante, o crawler pode tratá-lo como partições, em vez de tabelas à parte. Uma maneira de ajudar o crawler a descobrir tabelas individuais é adicionar o diretório raiz de cada tabela como um armazenamento de dados para o crawler.

As seguintes partições no Amazon S3 são um exemplo:

```
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
```

Se o esquema de `table1` e `table2` for semelhante, e uma única origem dos dados for definida como `s3://DOC-EXAMPLE-BUCKET/folder1/` no AWS Glue, o crawler poderá criar uma única tabela com duas colunas de partição: uma com `table1` e `table2` e outra com `partition1` a `partition5`.

Para fazer com que o crawler do AWS Glue crie duas tabelas separadas, defina o crawler para ter duas fontes de dados, `s3://DOC-EXAMPLE-BUCKET/folder1/table1/` e `s3://DOC-EXAMPLE-BUCKET/folder1/table2`, conforme mostrado no procedimento a seguir.

Para adicionar um armazenamento de dados do S3 a um crawler existente no AWS Glue

1. Faça login no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, escolha Rastreadores.
3. Escolha o link para o seu crawler e, em seguida, escolha Edit (Editar).
4. Em Step 2: Choose data sources and classifiers (Etapa 2: Escolher fontes de dados e classificadores), escolha Edit(Editar).
5. Em Data sources (Fontes de dados), escolha Add a data source (Adicionar uma fonte de dados).
6. Na caixa de diálogo Add data source (Adicionar fonte de dados), em S3 path (Caminho do S3), escolha Browse (Procurar).
7. Escolha o bucket que deseja usar e, em seguida, escolha Choose (Escolher).

A fonte de dados que você adicionou aparece na lista Data sources (Fontes de dados).

8. Escolha Próximo.
9. Na página Configure security settings (Definir configurações de segurança), crie ou escolha um perfil do IAM para o crawler e, em seguida, escolha Next (Próximo).
10. Certifique-se de que o caminho do S3 termine em uma barra à direita e, em seguida, escolha Add an S3 data source (Adicionar uma fonte de dados do S3).
11. Na página Set output and scheduling (Definir saída e programação), em Output configuration (Configuração da saída), escolha o banco de dados de destino.
12. Escolha Próximo.
13. Na página Review and update (Revisar e atualizar), revise as escolhas feitas. Para editar uma etapa, escolha Edit (Editar).
14. Escolha Atualizar.

Sincronizar o esquema da partição para evitar “HIVE_PARTITION_SCHEMA_MISMATCH”

Para cada tabela no AWS Glue Data Catalog que tenha colunas de partição, o esquema é armazenado no nível de tabela e para cada partição individual dentro da tabela. O esquema de partições é preenchido por um crawler do AWS Glue com base no exemplo de dados lido dentro da partição. Para ter mais informações, consulte Usar várias origens de dados com crawlers.

Quando o Athena executa uma consulta, ela valida o esquema da tabela e o esquema de todas as partições necessárias para a consulta. A validação compara os tipos de dados da coluna em ordem e verifica se eles correspondem às colunas que se sobrepõem. Isso evita operações inesperadas, como adicionar ou remover colunas no meio de uma tabela. Se o Athena detectar que o esquema de uma partição é diferente do esquema da tabela, o Athena talvez não possa processar a consulta e emita uma falha com `HIVE_PARTITION_SCHEMA_MISMATCH`.

Existem algumas maneiras de corrigir esse problema. Primeiro, se os dados tiverem sido adicionados acidentalmente, você poderá remover os arquivos de dados que causam a diferença no esquema, ignorar a partição e rastrear novamente os dados. Segundo, você pode descartar a partição individual e executar `MSCK REPAIR` no Athena para recriá-la usando o esquema da tabela. Essa segunda opção só funcionará se você tiver certeza de que o esquema aplicado continuará lendo os dados corretamente.

Atualizar metadados de tabelas

Depois de um rastreamento, o crawler do AWS Glue atribui automaticamente determinados metadados para ajudar a torná-los compatíveis com outras tecnologias externas, como Apache Hive, Presto e Spark. Às vezes, o crawler pode atribuir incorretamente propriedades de metadados. Corrija manualmente as propriedades no AWS Glue antes de consultar a tabela usando o Athena. Para obter mais informações, consulte Exibir e editar detalhes da tabela no Guia do desenvolvedor do AWS Glue.

O AWS Glue pode atribuir indevidamente metadados quando um arquivo CSV tem aspas em torno de cada campo de dados, processando a propriedade `serializationLib` incorretamente. Para ter mais informações, consulte Dados CSV entre aspas.

Trabalhar com arquivos CSV

Às vezes, os arquivos CSV têm aspas em valores de dados destinados a cada coluna e talvez haja valores de cabeçalho incluídos em arquivos CSV, que não fazem parte dos dados a serem analisados. Ao usar o AWS Glue para criar um esquema com base nesses arquivos, siga as orientações desta seção.

Dados CSV entre aspas

Você pode ter um arquivo CSV com campos de dados entre aspas duplas, como no seguinte exemplo:

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

Para executar uma consulta no Athena em uma tabela criada com base em um arquivo CSV que tenha valores entre aspas, você deve modificar as propriedades da tabela no AWS Glue para usar o OpenCSVSerDe. Para obter mais informações sobre o OpenCSV SerDe, consulte OpenCSVSerDe para processar CSV.

Para editar as propriedades da tabela no console do AWS Glue

1. No painel de navegação do console do AWS Glue, escolha Tables.
2. Escolha o link da tabela que deseja editar e, em seguida, escolha Actions (Ações) e Edit table (Editar tabela).
3. Na página Edit table (Editar tabela), faça as seguintes alterações:

- Em Serialization lib (Biblioteca de serialização), insira `org.apache.hadoop.hive.serde2.OpenCSVSerde`.
- Em Serde parameters (Parâmetros do Serde), insira os seguintes valores para as chaves `escapeChar`, `quoteChar` e `separatorChar`:
  - Em `escapeChar`, insira uma barra invertida (****).
  - Em `quoteChar`, insira aspas duplas (**"**).
  - Em `separatorChar`, insira uma vírgula (**,**).

4. Escolha Salvar.

Para obter mais informações, consulte Exibir e editar detalhes da tabela no Guia do desenvolvedor do AWS Glue.

Atualizar as propriedades da tabela do AWS Glue de forma programática

Você pode usar a operação da API AWS Glue UpdateTable ou o comando da CLI update-table para modificar o bloco `SerDeInfo` na definição da tabela, conforme mostrado no exemplo de JSON a seguir.

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

Arquivos CSV com cabeçalhos

Quando você define uma tabela no Athena com uma instrução `CREATE TABLE`, pode usar a propriedade de tabela `skip.header.line.count` para ignorar os cabeçalhos nos dados CSV, conforme mostrado no exemplo a seguir.

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

Se preferir, remova os cabeçalhos CSV com antecedência para que as informações do cabeçalho não sejam incluídas nos resultados da consulta do Athena. Uma maneira de fazer isso é usar os trabalhos do AWS Glue, que excutam o trabalho de Extract, Transform, and Load (ETL - Extração, transformação e carga). Você pode escrever scripts no AWS Glue usando uma linguagem que é uma extensão do dialeto PySpark Python. Para obter mais informações, consulte Criação de trabalhos no AWS Glue no Guia do desenvolvedor do AWS Glue.

O exemplo a seguir mostra uma função em um script do AWS Glue que escreve um quadro dinâmico usando `from_options` e define a opção de formato `writeHeader` como falsa, o que remove as informações do cabeçalho:

```
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")
```

Indexação e filtragem de partições do AWS Glue

Quando o Athena consulta tabelas particionadas, ele recupera e filtra as partições de tabela disponíveis para o subconjunto relevante para a sua consulta. À medida que novos dados e partições são adicionados, mais tempo torna-se necessário para processar as partições, e o runtime da consulta pode aumentar. Se você tiver uma tabela com um grande número de partições que aumenta ao longo do tempo, considere o uso de indexação e filtragem de partições do AWS Glue. A indexação de partições permite que o Athena otimize o processamento das partições e melhore a performance das consulta em tabelas altamente particionadas. A configuração da filtragem de partições nas propriedades de uma tabela é um processo de duas etapas:

1. Criar um índice de partição em AWS Glue.
2. Habilitar a filtragem de partições para a tabela.

Criar um índice de partição

Para conhecer as etapas da criação de um índice de partição no AWS Glue, consulte Trabalhar com índices de partição no Guia do desenvolvedor do AWS Glue. Para saber quais são as limitações dos índices de partição no AWS Glue, consulte a seção Sobre índices de partição na mesma página.

Habilitar filtragem de partição

Para habilitar a filtragem de partição para a tabela, você deve definir uma nova propriedade de tabela no AWS Glue. Para conhecer as etapas da definição de propriedades de tabela no AWS Glue,

consulte a página Configurar a projeção de partições. Quando você editar os detalhes da tabela no AWS Glue, adicione o seguinte par de chave-valor à seção Table properties (Propriedades de tabela):

- Para Key (Chave), adicione `partition_filtering.enabled`
- Para Value (Valor), adicione `true`

Você pode desabilitar a filtragem de partições nessa tabela a qualquer momento definindo o valor `partition_filtering.enabled` como `false`.

Depois de concluir as etapas acima, você pode retornar ao console do Athena para consultar os dados.

Para mais informações sobre usar a indexação e filtragem de partição, consulte Melhorar o desempenho de consultas Amazon Athena usando índices de partição AWS Glue Data Catalog no AWSBlog de Big Data.

Trabalhar com dados geoespaciais

O AWS Glue não oferece suporte nativo a Well-known Text (WKT – Texto bem conhecido), Well-Known Binary (WKB – Binário bem conhecido) ou outros tipos de dados PostGIS. O classificador do AWS Glue analisa dados geoespaciais e os classifica usando tipos de dados compatíveis para o formato, como `varchar` para CSV. Assim como ocorre com outras tabelas do AWS Glue, pode ser necessário atualizar as propriedades das tabelas criadas a partir de dados geoespaciais para permitir que o Athena analise esses tipos de dados no estado em que se encontram. Para obter mais informações, consulte Usar crawlers do AWS Glue e Trabalhar com arquivos CSV. O Athena pode não ser capaz de analisar alguns tipos de dados geoespaciais em tabelas do AWS Glue no estado em que se encontram. Para obter mais informações sobre como trabalhar com dados geoespaciais no Athena, consulte Consultar dados geoespaciais.

Usar trabalhos do AWS Glue para ETL com o Athena

Os trabalhos do AWS Glue realizam operações de ETL. Um trabalho do AWS Glue executa um script que extrai dados de fontes, transforma os dados e os carrega em destinos. Para obter mais informações, consulte Criação de trabalhos no AWS Glue no Guia do desenvolvedor do AWS Glue.

Criar tabelas usando o Athena para trabalhos ETL do AWS Glue

As tabelas que você cria no Athena devem ter uma propriedade de tabela adicionada chamada `classification`, que identifica o formato dos dados. Isso permite que o AWS Glue use as tabelas

para trabalhos de ETL. Os valores de classificação podem ser `avro`, `csv`, `json`, `orc`, `parquet` ou `xml`. Veja abaixo um exemplo da instrução `CREATE TABLE` no Athena:

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

Se a propriedade da tabela não tiver sido adicionada quando a tabela foi criada, a propriedade poderá ser adicionada com o console do AWS Glue.

Para adicionar a propriedade de classificação da tabela usando o console do AWS Glue

1. Faça login no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação do console, escolha Tables (Tabelas).
3. Escolha o link da tabela que deseja editar e, em seguida, escolha Actions (Ações) e Edit table (Editar tabela).
4. Role para baixo até a seção Table properties (Propriedades da tabela).
5. Escolha Adicionar.
6. Em Chave, digite **`classification`**.
7. Em Value (Valor), insira um tipo de dado (por exemplo, **`json`**).
8. Escolha Salvar.

   Na seção Table details (Detalhes da tabela), o tipo de dado que você inseriu aparece no campo Classification (Classificação) da tabela.

Para obter mais informações, consulte Trabalhar com tabelas no Guia do desenvolvedor do AWS Glue.

Com usar trabalhos de ETL para otimizar a performance da consulta

Os trabalhos do AWS Glue podem ajudar a transformar os dados em um formato que otimiza a performance das consultas no Athena. Os formatos de dados têm um grande impacto na performance e nos custos das consultas no Athena.

É recomendável usar os formatos de dados Parquet e ORC. O AWS Glue permite a gravação nesses dois formatos de dados, o que pode facilitar e agilizar a transformação de dados em um formato ideal para o Athena. Para obter mais informações sobre esses formatos e outras maneiras de melhorar o desempenho, leia Principais dicas para ajuste do desempenho do Amazon Athena.

Converter tipos de dados SMALLINT e TINYINT em INT ao converter em ORC

Para reduzir as chances de o Athena não conseguir ler os tipos de dados `SMALLINT` e `TINYINT` produzidos por um trabalho de ETL do AWS Glue, converta `SMALLINT` e `TINYINT` em `INT` ao usar o assistente ou ao escrever um script para um trabalho de ETL.

Automatizar trabalhos do AWS Glue para ETL

Você pode configurar trabalhos de ETL do AWS Glue para serem executados automaticamente com base em gatilhos. Esse recurso é ideal quando há dados de fora da AWS sendo enviados para um bucket do Amazon S3 em um formato inadequado para consultas no Athena. Para obter mais informações, consulte Iniciar trabalhos do AWS Glue usando gatilhos no Guia do desenvolvedor do AWS Glue.

Usar a AWS CLI para recriar um banco de dados do AWS Glue e suas tabelas

Não é possível renomear um banco de dados do AWS Glue diretamente, mas você pode copiar a definição, modificar a definição e usá-la para recriar o banco de dados com um nome diferente. Da mesma forma, é possível copiar as definições das tabelas do banco de dados antigo, modificar as definições e usar as definições modificadas para recriar as tabelas no novo banco de dados.

> **Note**
>
> O método apresentado não copia o particionamento da tabela.

O procedimento a seguir para o Windows pressupõe que a AWS CLI esteja configurada para saída JSON. Para alterar o formato de saída padrão na AWS CLI, execute `aws configure`.

Para copiar um banco de dados do AWS Glue usando a AWS CLI

1. Em um prompt de comando, execute o comando da AWS CLI a seguir para recuperar a definição do banco de dados do AWS Glue que você deseja copiar.

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

Para obter mais informações sobre o comando `get-database`, consulte get-database.

2. Salve a saída JSON em um arquivo com o nome do novo banco de dados (por exemplo, *new_database_name*`.json`) na área de trabalho.
3. Abra o arquivo *new_database_name*`.json` em um editor de textos.
4. No arquivo JSON, execute as seguintes etapas:

   a. Remova a entrada externa `{ "Database":` e a chave de fechamento correspondente `}` no final do arquivo.

   b. Altere a entrada `Name` para o nome do novo banco de dados.

   c. Remova o campo `CatalogId`.
5. Salve o arquivo.
6. Em um prompt de comando, execute o comando da AWS CLI a seguir para usar o arquivo de definição de banco de dados modificado para criar o banco de dados com o novo nome.

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

   Para obter mais informações sobre o comando `create-database`, consulte create-database. Para obter informações sobre como usar parâmetros da AWS CLI em um arquivo, consulte Carregar os parâmetros da AWS CLI de um arquivo no Guia do usuário da AWS Command Line Interface.
7. Para verificar se o novo banco de dados foi criado no AWS Glue, execute o seguinte comando:

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

Agora você está pronto para obter a definição de uma tabela que deseja copiar para o novo banco de dados, modificar a definição e usar a definição modificada para recriar a tabela no novo banco de dados. Esse procedimento não altera o nome da tabela.

Para copiar uma tabela do AWS Glue usando a AWS CLI

1. Em um prompt de comando, execute o seguinte comando da AWS CLI.

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

   Para obter mais informações sobre o comando `get-table`, consulte get-table.

2. Salve a saída JSON em um arquivo com o nome da tabela (por exemplo, *table_name*.json) na área de trabalho do Windows.
3. Abra o arquivo em um editor de textos.
4. No arquivo JSON, remova a entrada externa `{"Table":` e a chave de fechamento correspondente `}` no final do arquivo.
5. No arquivo JSON, remova as seguintes entradas e os respectivos valores:
   - `DatabaseName`: esta entrada não é necessária porque o comando `create-table` da CLI usa o parâmetro `--database-name`.
   - `CreateTime`
   - `UpdateTime`
   - `CreatedBy`
   - `IsRegisteredWithLakeFormation`
   - `CatalogId`
   - `VersionId`
6. Salve o arquivo de definição da tabela.
7. Em um prompt de comando, execute o seguinte comando da AWS CLI para recriar a tabela no novo banco de dados:

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

   Para obter mais informações sobre o comando `create-table`, consulte create-table.

   A tabela agora aparece no novo banco de dados no AWS Glue e pode ser consultada no Athena.
8. Repita as etapas para copiar cada tabela adicional para o novo banco de dados no AWS Glue.

Usar o conector de dados do Athena para metastore externo do Hive

Você pode usar o conector de dados do Amazon Athena para o metastore externo do Hive para consultar conjuntos de dados no Amazon S3 que usam um metastore do Apache Hive. Não é necessária nenhuma migração de metadados para o AWS Glue Data Catalog. No console de gerenciamento do Athena, configure uma função do Lambda para se comunicar com o metastore do Hive que está em sua VPC privada e conectá-la ao metastore. A conexão do Lambda com o

metastore do Hive é protegida por um canal privado do Amazon VPC e não usa a Internet pública. Você pode usar seu próprio código de função do Lambda ou a implementação padrão do conector de dados do Athena para o metastore externo do Hive.

Tópicos

- Visão geral de recursos
- Fluxo de trabalho
- Considerações e limitações
- Conectar o Athena a um metastore do Apache Hive
- Usar o AWS Serverless Application Repository para implantar um conector de origem de dados do Hive
- Conectar o Athena a um metastore do Hive usando uma função de execução do IAM existente
- Configurar o Athena para usar um conector de metastore do Hive implantado
- Usar um nome de origem de dados padrão em consultas do metastore externo do Hive
- Usar visualizações do Hive
- Usar a AWS CLI com metastores do Hive
- Implementação de referência

Visão geral de recursos

Com o conector de dados do Athena para metastore externo do Hive, você pode executar as seguintes tarefas:

- Use o console do Athena para registrar catálogos personalizados e usá-los para executar consultas.
- Defina funções do Lambda para diferentes metastores externos do Hive e adicione-as às consultas do Athena.
- Use o AWS Glue Data Catalog e os metastores externos do Hive na mesma consulta do Athena.
- Especifique um catálogo no contexto de execução da consulta como o catálogo padrão atual. Isso remove a necessidade de prefixar nomes de catálogo para nomes de banco de dados em suas consultas. Em vez de usar a sintaxe `catalog.database.table`, você pode usar `database.table`.
- Use uma variedade de ferramentas para executar consultas que fazem referência a metastores externos do Hive. Você pode usar o console do Athena, a AWS CLI, o AWS SDK, as APIs do

Athena e os drivers JDBC e ODBC atualizados do Athena. Os drivers atualizados são compatíveis com catálogos personalizados.

Suporte à API

O conector de dados do Athena para metastore externo do Hive permite operações da API de registro de catálogo e da API de metadados.

- Registro do catálogo: registre catálogos personalizados para metastores externos do Hive e origens de dados federadas.
- Metadados: use as APIs de metadados para fornecer informações de banco de dados e tabela para o AWS Glue e qualquer catálogo que você registrar no Athena.
- Cliente JAVA SDK do Athena: use as APIs de registro de catálogo, as APIs de metadados e o suporte para catálogos na operação `StartQueryExecution` no cliente Java SDK atualizado do Athena.

Implementação de referência

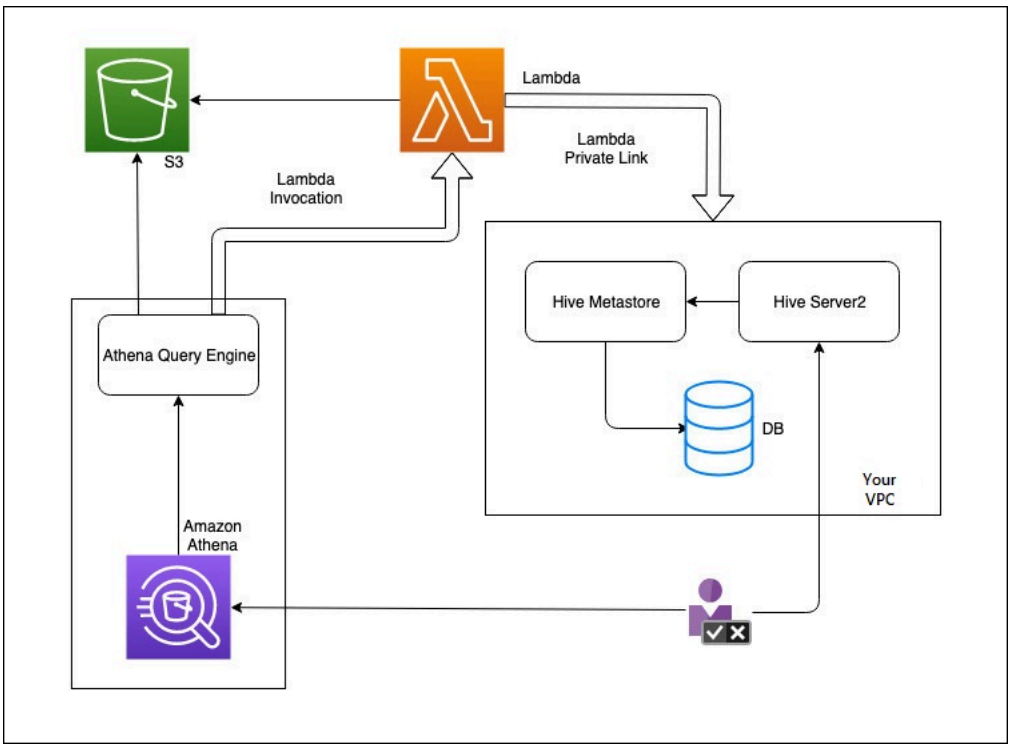
O Athena oferece uma implementação de referência para a função do Lambda que se conecta a metastores externos do Hive. A implementação de referência é fornecida no GitHub como um projeto de código aberto no metastore do Athena Hive.

A implementação de referência está disponível como os dois aplicativos do AWS SAM a seguir no AWS Serverless Application Repository (SAR). Você pode usar qualquer uma dessas aplicações no SAR para criar as próprias funções do Lambda.

- **`AthenaHiveMetastoreFunction`**: arquivo Uber `.jar` da função do Lambda. Um "uber" JAR (também conhecido como fat JAR ou JAR com dependências) é um arquivo `.jar` com um programa Java e as respectivas dependências em um único arquivo.
- **`AthenaHiveMetastoreFunctionWithLayer`**: a camada do Lambda e o arquivo `.jar` fino da função do Lambda.

Fluxo de trabalho

O diagrama a seguir mostra como o Athena interage com o metastore externo do Hive.



Neste fluxo de trabalho, o metastore do Hive conectado ao banco de dados está dentro da sua VPC. Você usa o Hive Server2 para gerenciar sua metastore do Hive usando a CLI do Hive.

O fluxo de trabalho para uso de metastores externos do Hive do Athena inclui as etapas a seguir.

1. Crie uma função do Lambda que conecte o Athena ao metastore do Hive que reside em sua VPC.
2. Registre um nome de catálogo exclusivo para seu metastore do Hive e um nome de função correspondente em sua conta.
3. Ao executar uma consulta DML ou DDL do Athena que usa o nome do catálogo, o mecanismo de consulta do Athena chama o nome da função do Lambda que você associou ao nome do catálogo.
4. Ao usar AWS PrivateLink, a função do Lambda se comunica com o metastore externo do Hive em sua VPC e recebe respostas às solicitações de metadados. O Athena usa os metadados do metastore externo do Hive da mesma forma que usa os metadados do AWS Glue Data Catalog padrão.

Considerações e limitações

Ao usar o conector de dados do Athena para metastore externo do Hive, considere os seguintes pontos:

- É possível usar CTAS para criar uma tabela em um metastore do Hive externo.
- É possível usar INSERT INTO para inserir dados em um metastore do Hive externo.
- O suporte de DDL para metastore externo do Hive está limitado às instruções a seguir.
  - 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
  - CRIAR TABELA
  - CREATE TABLE AS
  - DESCRIBE TABLE
  - DROP DATABASE
  - DESCARTAR TABELA
  - SHOW COLUMNS
  - SHOW CREATE TABLE
  - SHOW PARTITIONS
  - SHOW SCHEMAS
  - SHOW TABLES
  - SHOW TBLPROPERTIES
- O número máximo de catálogos registrados que é possível ter é 1.000.
- A autenticação Kerberos para o metastore do Hive não é compatível.
- Para usar o driver JDBC com um metastore externo do Hive ou com consultas federadas, inclua `MetadataRetrievalMethod=ProxyAPI` na string de conexão do JDBC. Para obter informações sobre o driver JDBC, consulte Conectar-se ao Amazon Athena com JDBC.
- As colunas ocultas do Hive `$path`, `$bucket`, `$file_size`, `$file_modified_time`, `$partition` e `$row_id` não podem ser usadas para filtragem de controle de acesso detalhada.

- Não há suporte para o controle de acesso detalhado para tabelas de sistema ocultas do Hive, como *example_table*$partitions ou *example_table*$properties.

Permissões

Conectores de dados predefinidos e personalizados podem exigir acesso aos recursos a seguir para funcionar corretamente. Verifique as informações do conector que você usa para confirmar se você configurou a VPC corretamente. Para obter informações sobre as permissões do IAM necessárias para executar consultas e criar um conector de origem dos dados no Athena, consulte Permitir acesso a um conector de dados do Athena para metastore externo do Hive e Permitir acesso da função do Lambda aos metastores externos do Hive.

- Amazon S3: além de gravar os resultados das consultas no local específico do Athena no Amazon S3, os conectores de dados gravam em um bucket de vazamento no Amazon S3. São necessárias conectividade e permissões para esse local do Amazon S3. Para obter mais informações, consulte Local de vazamento no Amazon S3 mais adiante neste tópico.
- Athena: o acesso é necessário para conferir o status da consulta e evitar verificação em excesso.
- AWS Glue: o acesso será necessário se o conector usar AWS Glue para metadados complementares ou primários.
- AWS Key Management Service
- Políticas: o metastore do Hive, o Athena Query Federation e as UDFs exigem políticas além da Política gerenciada pela AWS: AmazonAthenaFullAccess. Para ter mais informações, consulte Gerenciamento de identidade e acesso no Athena.

Local de vazamento no Amazon S3

Devido ao limite de tamanho de resposta da função do Lambda, as respostas que o ultrapassam são vazadas para um local no Amazon S3 especificado quando você cria a função do Lambda. O Athena lê essas respostas diretamente do Amazon S3.

> **Note**
>
> O Athena não remove os arquivos de resposta do Amazon S3. Recomendamos configurar uma política de retenção para excluir arquivos de resposta automaticamente.

Conectar o Athena a um metastore do Apache Hive

Para conectar o Athena a um metastore do Apache Hive, crie e configure uma função do Lambda. Para uma implementação básica, execute todas as etapas necessárias no console de gerenciamento do Athena.

> **Note**
>
> O procedimento a seguir requer que você tenha permissão para criar uma função personalizada do IAM para a função do Lambda. Se você não tiver permissão para criar uma função personalizada, poderá usar a implementação de referência do Athena para criar uma função do Lambda separadamente e, depois, usar o console do AWS Lambda para escolher uma função existente do IAM para essa função. Para ter mais informações, consulte Conectar o Athena a um metastore do Hive usando uma função de execução do IAM existente.

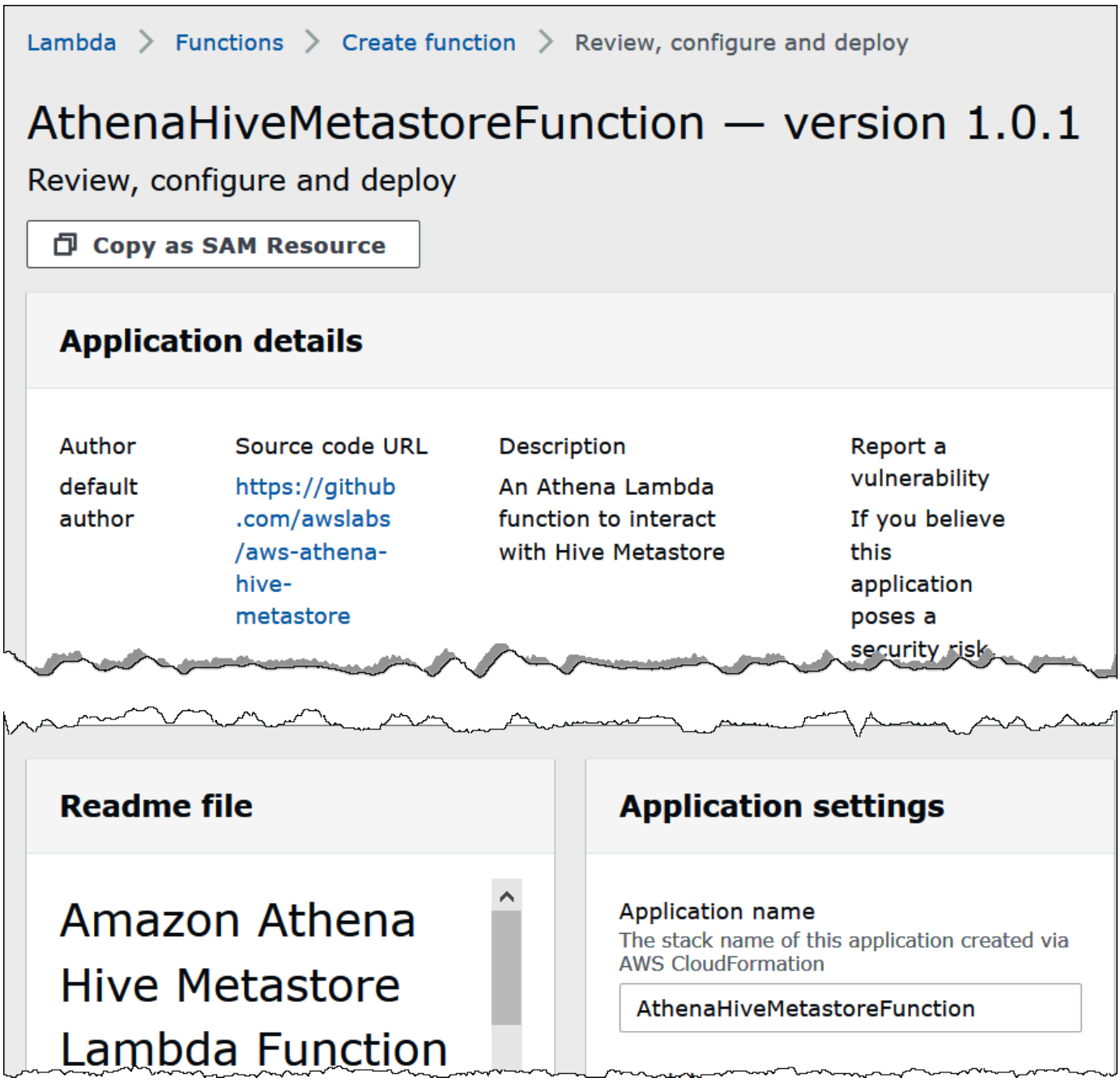
Para conectar o Athena a um metastore do Hive

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.

3. Escolha Data sources (Origens de dados).
4. No canto superior direito do console, escolha Create data source (Criar origem dos dados).

5. Na página Choose a data source (Escolher uma origem dos dados), em Data sources (Origens de dados), escolha S3 - Apache Hive metastore.
6. Escolha Próximo.
7. Na seção Data source details (Detalhes da origem dos dados), em Data source name (Nome da origem dos dados), insira o nome que deseja usar em suas instruções SQL ao consultar a origem dos dados pelo Athena. O nome pode ter até 127 caracteres e deve ser exclusivo na sua conta. Ele não poderá ser alterado após a criação. Os caracteres válidos são a-z, A-Z, 0-9, _ (sublinhado), @ (arroba) e - (hífen). Os nomes `awsdatacatalog`, `hive`, `jmx` e `system` são reservados pelo Athena e não podem ser usados como nomes de origens dos dados.
8. Em Função do Lambda, escolha Criar função do Lambda e, em seguida, Criar uma nova função do Lambda no AWS Lambda

   A página AthenaHiveMetastoreFunction é aberta no console do AWS Lambda. A página inclui informações detalhadas sobre o conector.

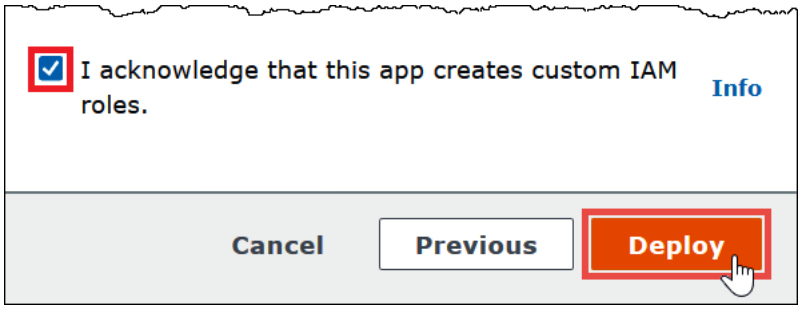


9. Em Application settings (Configurações da aplicação), insira os parâmetros para a função do Lambda.

   - LambdaFuncName: especifique um nome para a função. Por exemplo, myHiveMetastore.
   - SpillLocation: especifique um local do Amazon S3 nessa conta para armazenar os metadados de vazamento, caso o tamanho da resposta da função do Lambda exceda 4 MB.
   - HMSUris: insira o URI do host do metastore do Hive que usa o protocolo Thrift na porta 9083. Use a sintaxe `thrift://<host_name>:9083`.

- LambdaMemory: especifique um valor entre 128 MB e 3008 MB. A função do Lambda aloca os ciclos de CPU proporcionalmente à quantidade de memória que você configura. O padrão é 1024.
- LambdaTimeout: especifique o tempo máximo de execução de invocação do Lambda permitido em segundos de 1 a 900 (900 segundos são 15 minutos). O padrão é 300 segundos (5 minutos).
- VPCSecurityGroupIds: insira uma lista separada por vírgulas de IDs de grupo de segurança da VPC para o metastore do Hive.
- VPCSubnetIds: insira uma lista separada por vírgulas de IDs de sub-rede da VPC para o metastore do Hive.

10. Selecione I acknowledge that this app creates custom IAM roles (Eu reconheço que este aplicativo cria funções personalizadas do IAM e, em seguida, escolha Deploy (Implantar).



Quando a implantação for concluída, sua função será exibida na lista de aplicações do Lambda. Agora que a função do metastore do Hive foi implantada em sua conta, você pode configurar o Athena para usá-la.

11. Retorne à página Enter data source details (Inserir detalhes da origem dos dados) do console do Athena.
12. Na seção Lambda function (Função do Lambda), escolha o ícone de atualização ao lado da caixa de pesquisa de funções do Lambda. Atualizar a lista de funções disponíveis faz com que a função recém-criada apareça na lista.
13. Escolha o nome da função que você acabou de criar no console do Lambda. O ARN da função do Lambda é exibido.
14. (Opcional) Para Tags, adicione pares de chave-valor a associar com essa origem dos dados. Para obter mais informações sobre tags, consulte Etiquetar recursos do Athena.
15. Escolha Próximo.
16. Na página Review and create (Revisar e criar), analise os detalhes da origem dos dados e escolha Create data source (Criar origem dos dados).

17. A seção Data source details (Detalhes da origem dos dados) da página de sua origem dos dados mostra informações sobre o novo conector.

    Agora você poderá usar o Data source name (Nome da origem dos dados) especificado para fazer referência ao metastore do Hive em suas consultas SQL no Athena. Nas consultas SQL, use a seguinte sintaxe de exemplo, substituindo `hms-catalog-1` pelo nome do catálogo especificado anteriormente.

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

18. Para obter informações sobre como visualizar, editar ou excluir as origens dos dados criadas, consulte Gerenciar origens de dados.

Usar o AWS Serverless Application Repository para implantar um conector de origem de dados do Hive

Para implantar um conector de origem dos dados do Athena para Hive, você pode usar o AWS Serverless Application Repository em vez de começar com o console do Athena. Use AWS Serverless Application Repository para encontrar o conector que deseja usar, forneça os parâmetros que o conector exige e implante o conector em sua conta. Depois de implantar o conector, você usa o console do Athena para disponibilizar a origem dos dados para o Athena.

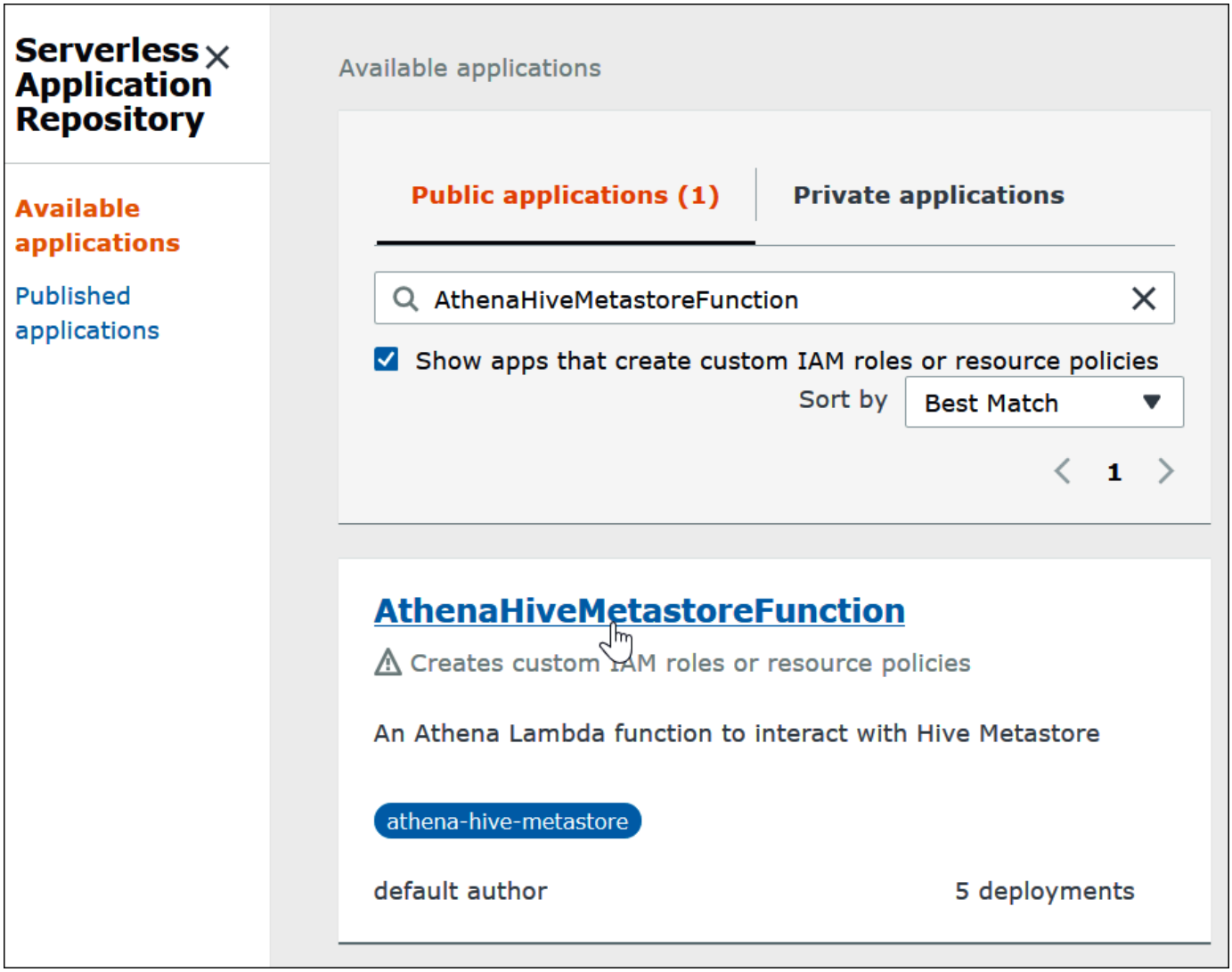
Como usar o AWS Serverless Application Repository para implantar um conector de fonte de dados do Hive em sua conta

1. Faça login no AWS Management Console e abra o Repositório de aplicativos sem servidor.
2. No painel de navegação, escolha Aplicativos disponíveis.
3. Selecione a opção Show apps that create custom IAM roles or resource policies (Mostrar aplicações que criam funções personalizadas do IAM ou políticas de recursos).
4. Na caixa de pesquisa, insira **`Hive`**. Os conectores exibidos incluem estes dois:

    - AthenaHiveMetastoreFunction: arquivo Uber `.jar` da função do Lambda.
    - AthenaHiveMetastoreFunctionWithLayer: a camada do Lambda e o arquivo `.jar` fino da função do Lambda.

Os dois aplicativos têm a mesma funcionalidade e diferem apenas em sua implementação. É possível usar qualquer um para criar uma função do Lambda que conecta o Athena ao seu metastore do Hive.

5. Escolha o nome do conector que você deseja usar. Este tutorial usa AthenaHiveMetastoreFunction.



6. Em Application settings (Configurações da aplicação), insira os parâmetros para a função do Lambda.

   - LambdaFuncName: especifique um nome para a função. Por exemplo, myHiveMetastore.
   - SpillLocation: especifique um local do Amazon S3 nessa conta para armazenar os metadados de vazamento, caso o tamanho da resposta da função do Lambda exceda 4 MB.

- HMSUris: insira o URI do host do metastore do Hive que usa o protocolo Thrift na porta 9083. Use a sintaxe `thrift://<host_name>:9083`.
- LambdaMemory: especifique um valor entre 128 MB e 3008 MB. A função do Lambda aloca os ciclos de CPU proporcionalmente à quantidade de memória que você configura. O padrão é 1024.
- LambdaTimeout: especifique o tempo máximo de execução de invocação do Lambda permitido em segundos de 1 a 900 (900 segundos são 15 minutos). O padrão é 300 segundos (5 minutos).
- VPCSecurityGroupIds: insira uma lista separada por vírgulas de IDs de grupo de segurança da VPC para o metastore do Hive.
- VPCSubnetIds: insira uma lista separada por vírgulas de IDs de sub-rede da VPC para o metastore do Hive.

7. Na parte inferior direita da página Application details (Detalhes da aplicação), selecione I acknowledge that this app creates custom IAM roles (Eu reconheço que esta aplicação cria funções personalizadas do IAM) e escolha Deploy (Implantar).

Neste ponto, você pode configurar o Athena para usar sua função do Lambda para se conectar ao metastore do Hive. Para obter as etapas, consulte Configurar o Athena para usar um conector de metastore do Hive implantado.

Conectar o Athena a um metastore do Hive usando uma função de execução do IAM existente

Para conectar seu metastore externo do Hive ao Athena com uma função do Lambda que usa uma função do IAM existente, você pode usar a implementação de referência do Athena do conector Athena para o metastore externo do Hive.

Veja abaixo as três etapas principais:

1. Clonar e criar: clone a implementação de referência do Athena e crie o arquivo JAR que contém o código da função do Lambda.
2. Console do AWS Lambda: no console do AWS Lambda, crie uma função do Lambda, atribua a ela uma função de execução do IAM existente e carregue o código da função que você gerou.
3. Console do Amazon Athena: no console do Amazon Athena, crie um nome da origem dos dados que você pode usar para referenciar o metastore externo do Hive em suas consultas do Athena.

Se você já tiver permissões para criar uma função personalizada do IAM, poderá usar um fluxo de trabalho mais simples que use o console do Athena e o AWS Serverless Application Repository para criar e configurar uma função do Lambda. Para ter mais informações, consulte Conectar o Athena a um metastore do Apache Hive.

Pré-requisitos

- O Git deve estar instalado no sistema.
- Você deve ter o Apache Maven instalado.
- Você tem uma função de execução do IAM que pode atribuir à função do Lambda. Para ter mais informações, consulte Permitir acesso da função do Lambda aos metastores externos do Hive.

Clonar e criar a função do Lambda

O código da função para a implementação de referência do Athena é um projeto do Maven localizado no GitHub em awslabs/aws-athena-hive-metastore. Para obter informações detalhadas sobre o projeto, consulte o arquivo README correspondente no GitHub ou o tópico Implementação de referência nesta documentação.

Para clonar e criar o código da função do Lambda

1. Insira o seguinte comando para clonar a implementação de referência do Athena:

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

2. Execute este comando para criar o arquivo `.jar` para a função do Lambda:

   ```
   mvn clean install
   ```

   Após a compilação bem-sucedida do projeto, o seguinte arquivo `.jar` será criado na pasta de destino do projeto:

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

   Na próxima seção, use o console do AWS Lambda para carregar esse arquivo em sua conta da Amazon Web Services.

Criar e configurar a função do Lambda no console do AWS Lambda

Nesta seção, use o console do AWS Lambda para criar uma função que aplique uma função de execução do IAM existente. Depois de configurar uma VPC para a função, carregue o código da função e configure as variáveis de ambiente dela.

Criar a função do Lambda

Nesta etapa, crie uma função no console do AWS Lambda que use uma função do IAM existente.

Para criar uma função do Lambda que usa uma função do IAM existente

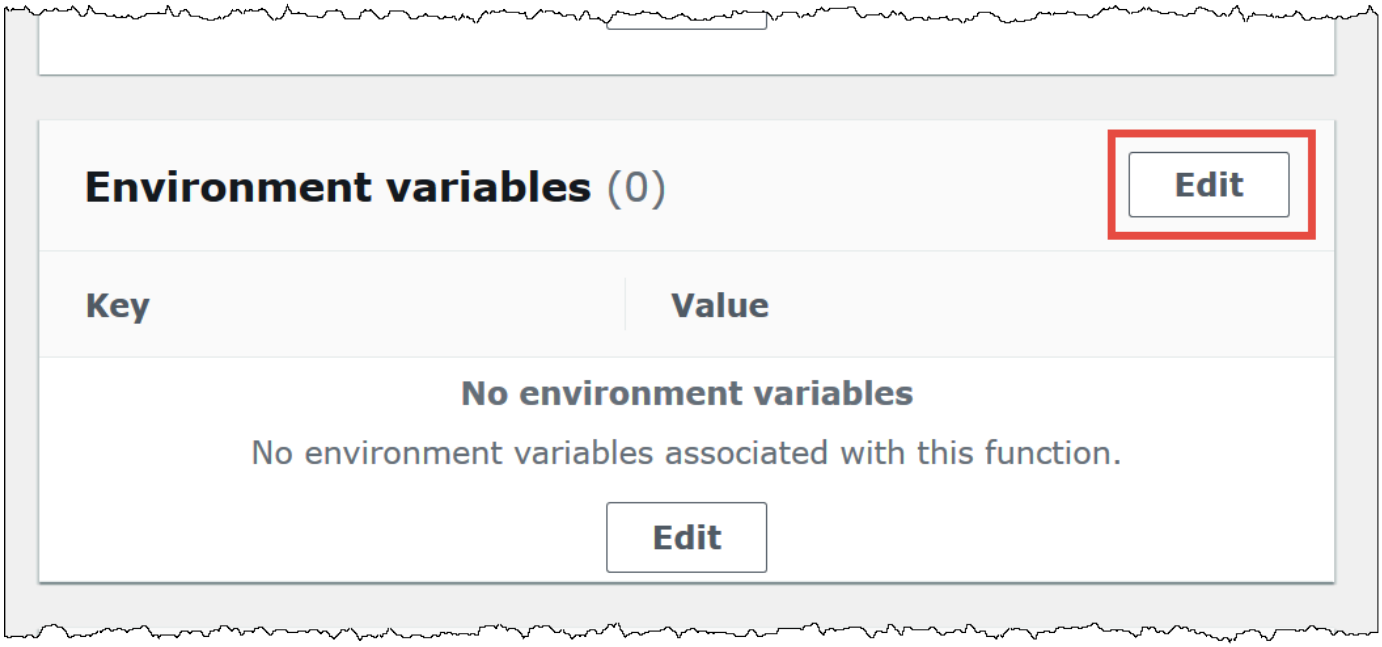
1. Faça login no AWS Management Console e abra o console do AWS Lambda em https://console.aws.amazon.com/lambda/.
2. No painel de navegação, escolha Funções.
3. Escolha Create function (Criar função).
4. Escolha Author from scratch (Criar do zero).
5. Em Function name (Nome da função), insira o nome da sua função do Lambda (por exemplo, **`EHMSBasedLambda`**).
6. Em Runtime (Tempo de execução), escolha Java 8.
7. Em Permissions (Permissões), expanda Change default execution role (Alterar função de execução padrão).
8. Para Execution role (Função de execução), selecione Use an existing role (Usar uma função existente).
9. Em Existing role (Função existente), escolha a função de execução do IAM que sua função do Lambda usará com o Athena (este exemplo usa uma função chamada `AthenaLambdaExecutionRole`).
10. Expanda Advanced settings (Configurações avançadas).
11. Selecione Enable Network (Habilitar rede).
12. Em VPC, escolha a VPC à qual sua função terá acesso.
13. Em Subnets (Sub-redes), escolha as sub-redes VPC para o Lambda usar.
14. Em Security groups (Grupos de segurança), escolha os grupos de segurança da VPC para o Lambda usar.
15. Escolha a opção Criar função. O console do AWS Lambda abre a página de configuração da sua função e começa a criá-la.

Carregar o código e configurar a função do Lambda

Quando o console informar que sua função foi criada com êxito, você estará pronto para carregar o código da função e configurar as variáveis de ambiente.

Para carregar o código da função do Lambda e configurar as variáveis de ambiente

1. No console do Lambda, certifique-se de que você está na guia Code (Código) da página da função que você especificou.
2. Em Code source (Fonte do código), escolha Upload from (Carregar de) e escolha .zip or jar file (Arquivo .zip ou .jar).
3. Carregue o arquivo `hms-lambda-func-1.0-SNAPSHOT-withdep.jar` que você gerou.
4. Em sua página da função Lambda, escolha a guia Configuration (Configuração).
5. No painel à esquerda, escolha Environment variables (Variáveis de ambiente).
6. Em Variáveis de ambiente, selecione Editar.

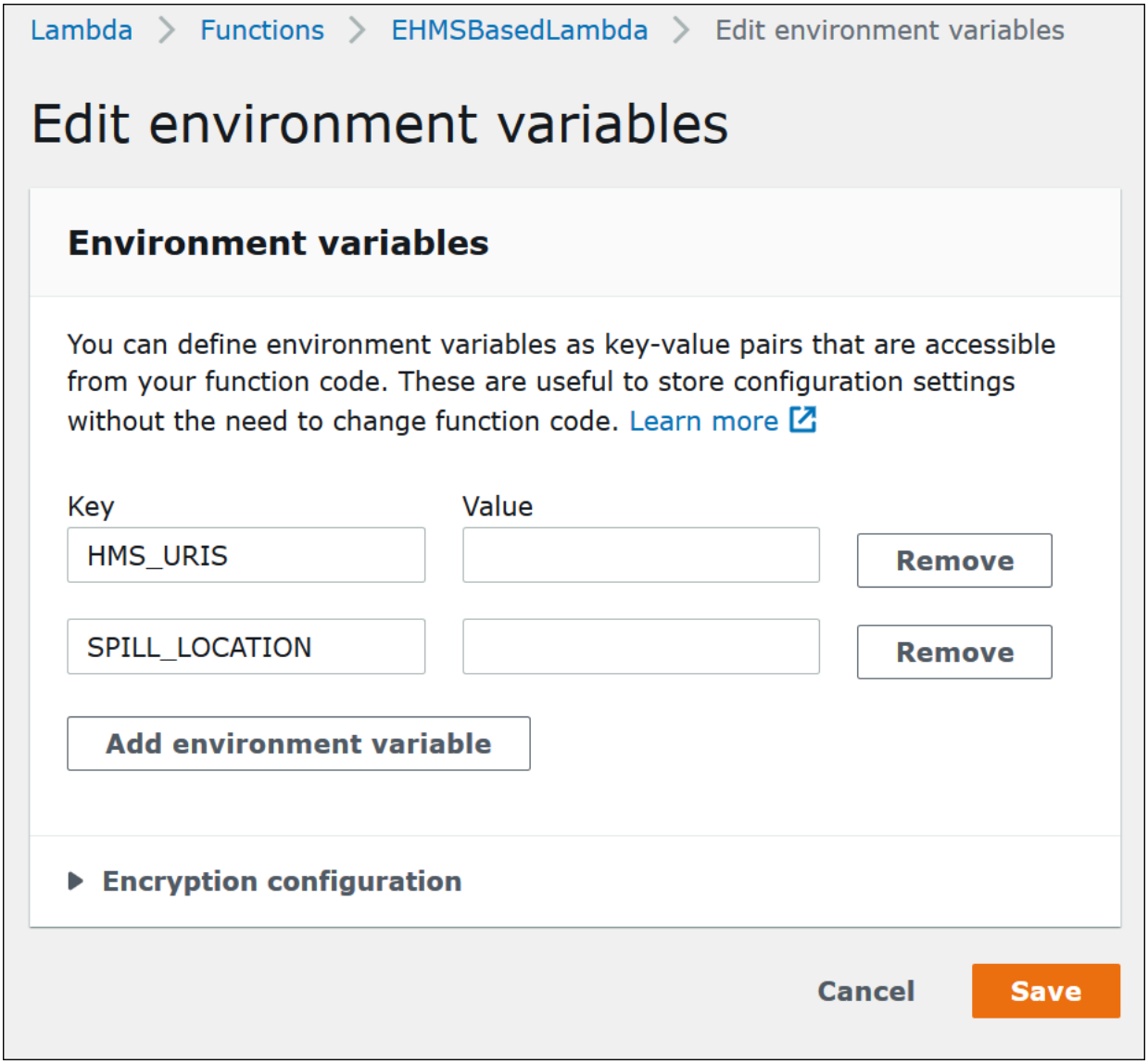


7. Na página Edit environment variables (Editar variáveis de ambiente), use a opção Add environment variable (Adicionar variável de ambiente) para adicionar as seguintes chaves e valores da variável de ambiente:

   - HMS_URIS: use a sintaxe a seguir para inserir o URI do host do metastore do Hive que usa o protocolo Thrift na porta 9083.

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

- SPILL_LOCATION: especifique um local do Amazon S3 na sua conta da Amazon Web Services para armazenar os metadados de vazamento, caso o tamanho da resposta da função do Lambda exceda 4 MB.



8. Escolha Salvar.

Neste ponto, você está pronto para configurar o Athena para usar sua função do Lambda para se conectar ao metastore do Hive. Para obter as etapas, consulte Configurar o Athena para usar um conector de metastore do Hive implantado.

Configurar o Athena para usar um conector de metastore do Hive implantado

Depois de implantar um conector de origem dos dados do Lambda em sua conta, como `AthenaHiveMetastoreFunction`, você pode configurar o Athena para usá-lo. Para isso, crie um nome de origem dos dados que faça referência a sua metastore externa do Hive para usar em suas consultas do Athena.

Para conectar o Athena ao metastore do Hive usando uma função do Lambda existente

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.

3. Escolha Data sources (Origens de dados).
4. Na página Data sources (Origens de dados), escolha Create data source (Criar origem dos dados).
5. Na página Choose a data source (Escolher uma origem dos dados), em Data sources (Origens de dados), escolha S3 - Apache Hive metastore.
6. Escolha Próximo.
7. Na seção Data source details (Detalhes da origem dos dados), em Data source name (Nome da origem dos dados), insira o nome que deseja usar em suas instruções SQL ao consultar a origem dos dados pelo Athena (por exemplo, `MyHiveMetastore`). O nome pode ter até 127 caracteres e deve ser exclusivo na sua conta. Ele não poderá ser alterado após a criação. Os caracteres válidos são a-z, A-Z, 0-9, _ (sublinhado), @ (arroba) e - (hífen). Os nomes

`awsdatacatalog`, `hive`, `jmx` e `system` são reservados pelo Athena e não podem ser usados como nomes de origens dos dados.

8. Na seção Connection details (Detalhes da conexão), use a caixa Select or enter a Lambda function (Selecionar ou inserir uma função do Lambda) para escolher o nome da função que você acabou de criar. O ARN da função do Lambda é exibido.
9. (Opcional) Para Tags, adicione pares de chave-valor a associar com essa origem dos dados. Para obter mais informações sobre tags, consulte Etiquetar recursos do Athena.
10. Escolha Próximo.
11. Na página Review and create (Revisar e criar), analise os detalhes da origem dos dados e escolha Create data source (Criar origem dos dados).
12. A seção Data source details (Detalhes da origem dos dados) da página de sua origem dos dados mostra informações sobre o novo conector.

    Agora você poderá usar o Data source name (Nome da origem dos dados) especificado para fazer referência ao metastore do Hive em suas consultas SQL no Athena.

    Nas consultas SQL, use a seguinte sintaxe de exemplo, substituindo `ehms-catalog` pelo nome da origem dos dados especificada anteriormente.

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

13. Para visualizar, editar ou excluir as fontes de dados criadas, consulte Gerenciar origens de dados.

Usar um nome de origem de dados padrão em consultas do metastore externo do Hive

Ao executar consultas DML e DDL em metastores externos do Hive, você pode simplificar a sintaxe da consulta omitindo o nome do catálogo se esse nome estiver selecionado no editor de consulta. Certas restrições se aplicam a essa funcionalidade.

Instruções DML

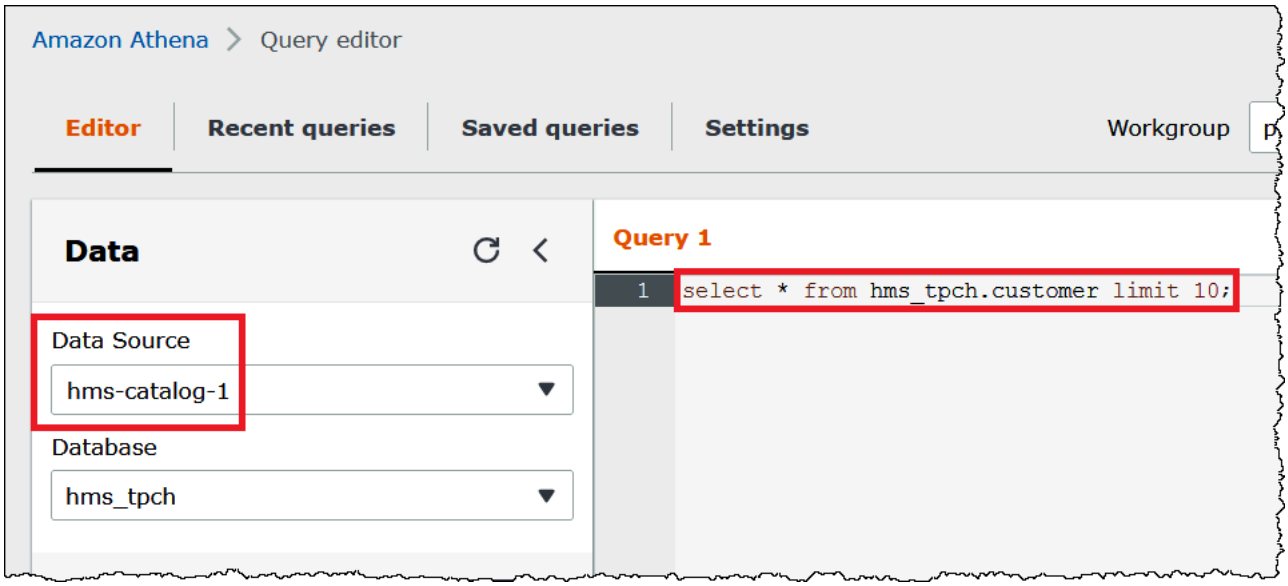
Como executar consultas com catálogos registrados

1. Você pode colocar o nome da origem dos dados antes do banco de dados usando a sintaxe `[[`*`data_source_name`*`].`*`database_name`*`].`*`table_name`*, como no exemplo a seguir.

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

2. Quando a origem dos dados que você deseja usar já estiver selecionada no editor de consultas, você poderá omitir o nome da consulta, como no exemplo a seguir.

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



3. Quando você usa várias origens dos dados em uma consulta, pode omitir somente o nome da origem dos dados padrão e especificar o nome completo de qualquer origem dos dados não padrão.

   Por exemplo, suponha que `AwsDataCatalog` esteja selecionado como origem dos dados padrão no editor de consultas. A instrução `FROM` no trecho de consulta a seguir qualifica totalmente os nomes das duas primeiras origens dos dados, mas omite o nome da terceira porque ela está no catálogo de dados do AWS Glue.

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

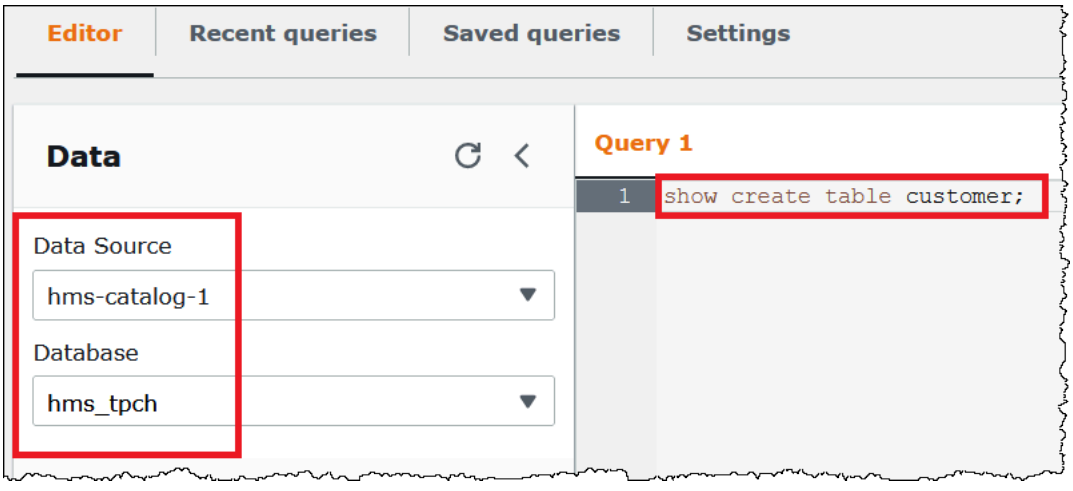
Instruções DDL

As instruções DDL do Athena a seguir permitem prefixos de nome de catálogo. Prefixos de nome de catálogo em outras instruções DDL causam erros de sintaxe.

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

Assim como acontece com as instruções DML, você pode omitir a origem dos dados e os prefixos do banco de dados da consulta quando a origem dos dados e o banco de dados são selecionados no editor de consultas.

Na imagem a seguir, a origem dos dados `hms-catalog-1` e o banco de dados `hms_tpch` são selecionados no editor de consultas. A instrução `show create table customer` é bem-sucedida mesmo que o prefixo `hms-catalog-1` e o nome do banco de dados `hms_tpch` sejam omitidos da consulta em si.



Especificar uma origem de dados padrão em uma string de conexão JDBC

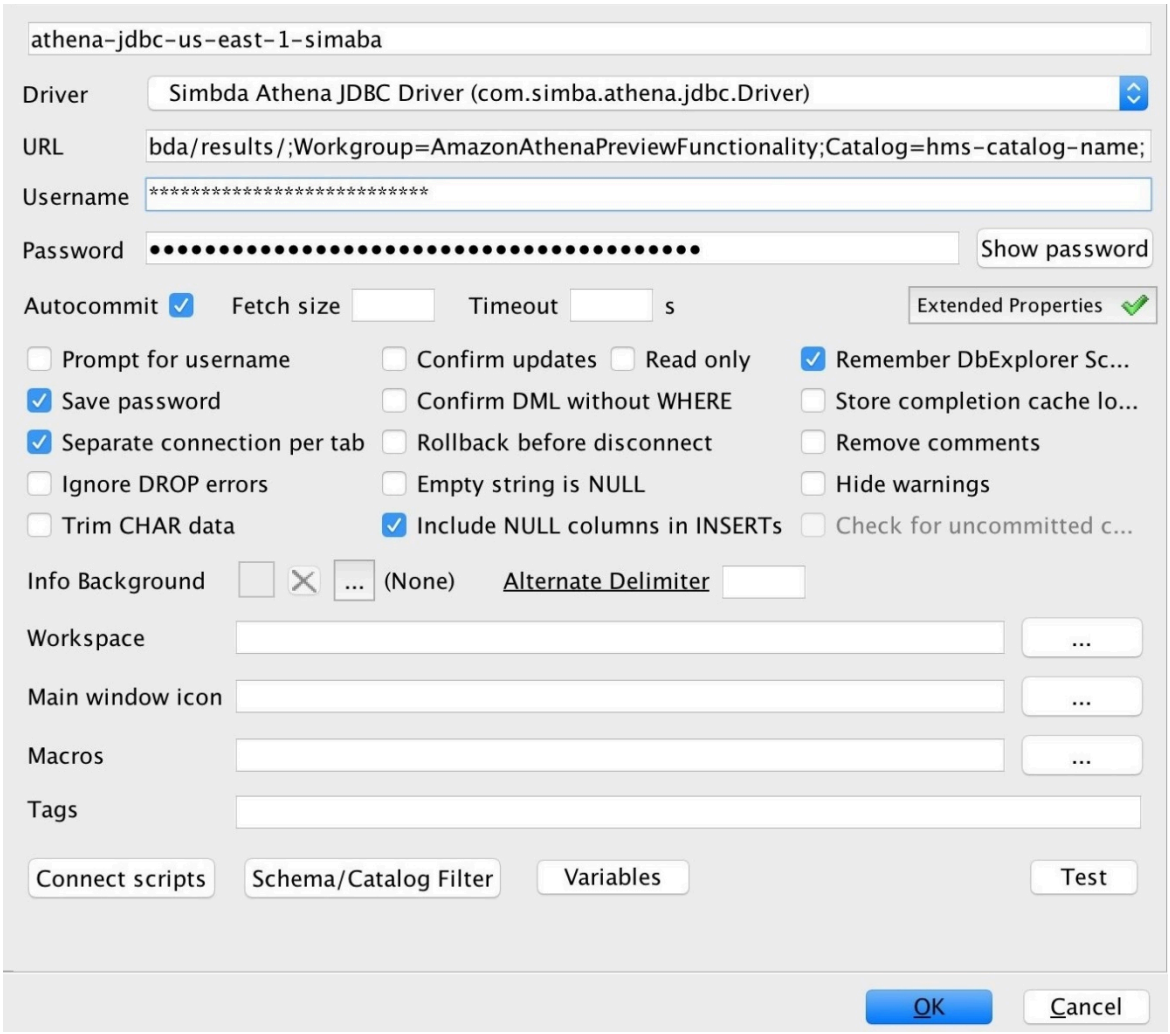
Ao usar o driver JDBC do Athena para conectar o Athena a um metastore externo do Hive, você pode usar o parâmetro `Catalog` para especificar o nome da origem de dados padrão na string de conexão em um editor SQL, como o SQL Workbench.

> **Note**
>
> Para baixar os drivers JDBC do Athena mais recentes, consulte Usar o Athena com o driver JDBC.

A cadeia de conexão a seguir especifica a origem dos dados padrão*hms-catalog-name*.

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

A imagem a seguir mostra um exemplo de URL de conexão JDBC como configurado no SQL Workbench.



Usar visualizações do Hive

Você pode usar o Athena para consultar visualizações existentes em seus metastores externos do Apache Hive. O Athena traduz as visualizações dinamicamente em tempo de execução sem alterar a visualização original ou armazenar a tradução.

Por exemplo, suponha que você tenha uma visualização do Hive que usa uma sintaxe semelhante a `LATERAL VIEW explode()`, não compatível com o Athena, como mostrado a seguir:

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

O Athena traduz a string de consulta da visualização do Hive em uma instrução que o Athena pode executar, como esta:

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

Para obter informações sobre como conectar um metastore externo do Hive ao Athena, consulte Usar o conector de dados do Athena para metastore externo do Hive.

Considerações e limitações

Ao consultar visualizações do Hive no Athena, considere os seguintes pontos:

- O Athena não oferece suporte à criação de visualizações do Hive. Você pode criar visualizações do Hive no seu metastore externo do Hive, que pode então ser consultado usando o Athena.
- O Athena não oferece suporte a UDFs personalizadas para visualizações do Hive.
- Devido a um problema conhecido no console do Athena, as visualizações do Hive aparecem na lista de tabelas, e não na lista de visualizações.
- Embora o processo de tradução seja automático, certas funções do Hive não são compatíveis com exibições do Hive ou exigem um tratamento especial. Para obter mais informações, consulte a seção a seguir.

Limitações de suporte às funções do Hive

Esta seção destaca as funções do Hive que não são compatíveis com visualizações do Hive no Athena ou que exigem um tratamento especial. Como o Athena atualmente oferece suporte principalmente a funções do Hive 2.2.0, funções disponíveis somente em versões superiores (como Hive 4.0.0) não são compatíveis. Para obter uma lista completa de funções do Hive, consulte o manual de idioma de UDFs do Hive.

Funções agregadas

Funções agregadas que exigem tratamento especial

A função agregada a seguir para visualizações do Hive exige um tratamento especial.

- Avg: em vez de `avg(INT i)`, use `avg(CAST(i AS DOUBLE))`.

Funções agregadas sem suporte

O Athena não oferece suporte às funções agregadas do Hive a seguir para visualizações do Hive.

```
covar_pop
histogram_numeric
ntile
percentile
percentile_approx
```

O Athena não oferece suporte a funções de regressão como `regr_count`, `regr_r2` e `regr_sxx` para visualizações do Hive.

Funções de data sem suporte

O Athena não oferece suporte às funções de data do Hive a seguir para visualizações do 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)
```

Funções de mascaramento sem suporte

O Athena não oferece suporte a funções de mascaramento do Hive como `mask()` e `mask_first_n()` para visualizações do Hive.

Funções diversas

Funções diversas que exigem tratamento especial

As funções diversas para visualizações do Hive a seguir exigem tratamento especial.

- md5: o Athena oferece suporte a `md5(binary)`, mas não a `md5(varchar)`.
- explode: o Athena oferece suporte a `explode` quando usada com a seguinte sintaxe:

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

- posexplode: o Athena oferece suporte a `posexplode` quando usada com a seguinte sintaxe:

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

  Na saída `(pos, val)`, o Athena trata a coluna `pos` como `BIGINT`. Por isso, poderá ser necessário converter a coluna `pos` em `BIGINT` para evitar uma visualização obsoleta. O exemplo a seguir ilustra essa técnica.

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

Funções diversas sem suporte

O Athena não oferece suporte às funções do Hive a seguir para visualizações do Hive.

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

Operadores

Operadores que exigem tratamento especial

Os operadores para visualizações do Hive a seguir exigem tratamento especial.

- Operador de módulo (%): como o tipo `DOUBLE` implicitamente converte em `DECIMAL(x,y)`, a sintaxe a seguir pode gerar uma mensagem de erro View is stale (Visualização obsoleta):

```
a_double % 1.0 AS column
```

  Para contornar esse problema, use `CAST`, como no exemplo a seguir.

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

- Operador de divisão (/): no Hive , `int` dividido por `int` gera um `double`. No Athena, a mesma operação gera um `int` truncado.

Operadores sem suporte

O Athena não oferece suporte aos operadores a seguir para visualizações do Hive.

~A: bitwise `NOT`

A ^ b: bitwise `XOR`

A & b: bitwise `AND`

A | b: bitwise `OR`

A **<=>** b: retorna o mesmo resultado que o operador igual a (=) para operandos não nulos. Retornará `TRUE` se ambos forem `NULL`, ou `FALSE` se um deles for `NULL`.

Funções de string

Funções de string que exigem tratamento especial

As funções de string para visualizações do Hive a seguir exigem tratamento especial.

- chr(bigint|double a): o Hive permite argumentos negativos; o Athena não.
- instr(string str, string substr): como o mapeamento do Athena para a função `instr` retorna `BIGINT` em vez de `INT`, use a seguinte sintaxe:

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

  Sem essa etapa, a visualização será considerada obsoleta.
- length(string a): como o mapeamento do Athena para a função `length` retorna `BIGINT` em vez de `INT`, use a seguinte sintaxe para que a visualização não seja considerada obsoleta:

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

Funções de string sem suporte

O Athena não oferece suporte às funções de string do Hive a seguir para visualizações do 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)
```

Funções XPath sem suporte

O Athena não oferece suporte a funções XPath do Hive como `xpath`, `xpath_short` e `xpath_int` para visualizações do Hive.

Solução de problemas

Ao usar visualizações do Hive no Athena, você poderá encontrar os seguintes problemas:

- View ***<view name>*** is stale (A visualização <nome da visualização> está obsoleta): esta mensagem geralmente indica uma não correspondência de tipo entre a visualização no Hive e no Athena. Se a mesma função no manual de idioma de UDFs do Hive e na documentação sobre funções e operadores do Presto tiver assinaturas diferentes, tente converter o tipo de dado sem correspondência.
- Function not registered (Função não registrada): o Athena atualmente não oferece suporte à função. Para obter mais informações, consulte as seções anteriores neste documento.

Usar a AWS CLI com metastores do Hive

É possível usar os comandos da CLI `aws athena` para gerenciar os catálogos de dados do metastore do Hive que você usa com o Athena. Depois de definir um ou mais catálogos para usar com o Athena, você poderá referenciar esses catálogos em seus comandos DDL e DML `aws athena`.

Usar a AWS CLI para gerenciar catálogos de metastore do Hive

Registrar um catálogo: create-data-catalog

Para registrar um catálogo de dados, use o comando `create-data-catalog`. Use o parâmetro `name` para especificar o nome que você deseja usar para o catálogo. Insira o ARN da função do Lambda na opção `metadata-function` do argumento `parameters`. Para criar tags para o novo catálogo, use o parâmetro `tags` com um ou mais pares de argumentos `Key=`*`key`*`,Value=`*`value`* separados por espaço.

O exemplo a seguir registra o catálogo de metastore do Hive chamado `hms-catalog-1`. O comando foi formatado para legibilidade.

```
$ 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
```

Mostrar detalhes do catálogo: get-data-catalog

Para mostrar os detalhes de um catálogo, passe o nome do catálogo para o comando `get-data-catalog`, como no exemplo a seguir.

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

O resultado de exemplo a seguir está em formato 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"
        }
    }
}
```

Listar catálogos registrados: list-data-catalogs

Para listar os catálogos registrados, use o comando `list-data-catalogs` e, opcionalmente, especifique uma região, como no exemplo a seguir. Os catálogos listados sempre incluem o AWS Glue.

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

O resultado de exemplo a seguir está em formato 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"
            }
        }
    ]
}
```

Atualizar um catálogo: update-data-catalog

Para atualizar um catálogo de dados, use o comando `update-data-catalog`, como no exemplo a seguir. O comando foi formatado para legibilidade.

```
$ 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
```

Excluir um catálogo: delete-data-catalog

Para excluir um catálogo de dados, use o comando `delete-data-catalog`, como no exemplo a seguir.

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

Mostrar detalhes do banco de dados: get-database

Para mostrar os detalhes de um banco de dados, passe o nome do catálogo e do banco de dados para o comando `get-database`, como no exemplo a seguir.

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

O resultado de exemplo a seguir está em formato JSON.

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

Listar bancos de dados em um catálogo: list-databases

Para listar os bancos de dados em um catálogo, use o comando `list-databases` e, opcionalmente, especifique uma região, como no exemplo a seguir.

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

O resultado de exemplo a seguir está em formato JSON.

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

Mostrar detalhes da tabela: get-table-metadata

Para mostrar os metadados de uma tabela, incluindo nomes de coluna e tipos de dados, passe o nome do catálogo, banco de dados e o nome da tabela para o comando `get-table-metadata`, como no exemplo a seguir.

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

O resultado de exemplo a seguir está em formato 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"
            }
        }
}
```

Visualizar metadados de todas as tabelas em um banco de dados: list-table-metadata

Para mostrar os metadados de todas as tabelas em um banco de dados, passe o nome do catálogo e do nome do banco de dados para o comando `list-table-metadata`. O comando `list-table-metadata` é semelhante ao comando `get-table-metadata`, exceto que você não especifica um nome de tabela. Para limitar o número de resultados, você pode usar a opção `--max-results`, como no exemplo a seguir.

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

O resultado de exemplo a seguir está em formato 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
}
```

Executar instruções DDL e DML

Ao usar a AWS CLI para executar instruções DDL e DML, você pode passar o nome do catálogo de metastore do Hive de uma das duas maneiras:

- Diretamente para as declarações que são compatíveis com ele.
- Para o parâmetro `Catalog` de `--query-execution-context`.

Instruções DDL

O exemplo a seguir passa o nome do catálogo diretamente como parte da instrução DDL `show create table`. O comando foi formatado para legibilidade.

```
$ 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"
```

A instrução DDL `show create table` de exemplo a seguir usa o parâmetro `Catalog` de `--query-execution-context` para passar o nome do catálogo de metastore do Hive `hms-catalog-1`. O comando foi formatado para legibilidade.

```
$ 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"
```

Instruções DML

O exemplo da instrução DML `select` a seguir passa o nome do catálogo diretamente para a consulta. O comando foi formatado para legibilidade.

```
$ 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"
```

O seguinte exemplo da instrução DML `select` usa o parâmetro `Catalog` de `--query-execution-context` para passar o nome do catálogo de metastore do Hive `hms-catalog-1`. O comando foi formatado para legibilidade.

```
$ 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"
```

Implementação de referência

O Athena oferece uma implementação de referência do seu conector para metastore externo do Hive no site GitHub.com: https://github.com/awslabs/aws-athena-hive-metastore.

A implementação de referência é um projeto Apache Maven que tem os seguintes módulos:

- **`hms-service-api`**: contém as operações de API entre a função do Lambda e os clientes dos serviços do Athena. Essas operações da API são definidas na interface do `HiveMetaStoreService`. Como este é um contrato de serviço, você não deve alterar nada neste módulo.
- **`hms-lambda-handler`**: um conjunto de manipuladores padrão do Lambda que processa todas as chamadas de API de metastore do Hive. A classe `MetadataHandler` é o dispatcher para todas as chamadas da API. Você não precisa alterar este pacote.
- **`hms-lambda-func`**: um exemplo de função do Lambda que tem os componentes a seguir.
  - **`HiveMetaStoreLambdaFunc`**: um exemplo de função do Lambda que estende `MetadataHandler`.
  - **`ThriftHiveMetaStoreClient`**: um cliente Thrift que se comunica com o metastore do Hive. Esse cliente foi escrito para o Hive 2.3.0. Se você usar uma versão diferente do Hive, talvez seja necessário atualizar essa classe para garantir que os objetos de resposta sejam compatíveis.
  - **`ThriftHiveMetaStoreClientFactory`**: controla o comportamento da função do Lambda. Por exemplo, você pode fornecer seu próprio conjunto de provedores de manipuladores substituindo o método `getHandlerProvider()`.
  - `hms.properties`: configura a função do Lambda. A maioria dos casos requer a atualização de apenas as duas propriedades a seguir.
    - `hive.metastore.uris`: o URI do metastore do Hive no formato `thrift://<host_name>:9083`.
    - `hive.metastore.response.spill.location`: o local do Amazon S3 para armazenar objetos de resposta quando os tamanhos excedem um determinado limite (por exemplo, 4 MB). O limite é definido na propriedade `hive.metastore.response.spill.threshold`. Não é recomendável alterar o valor padrão.

> **Note**
>
> Essas duas propriedades podem ser substituídas pelas variáveis de ambiente do Lambda `HMS_URIS` e `SPILL_LOCATION`. Use essas variáveis em vez de recompilar o código-fonte da função do Lambda para usar a função com um metastore do Hive ou local de vazamento diferente.

- **`hms-lambda-layer`**: um projeto de montagem do Maven que coloca `hms-service-api`, `hms-lambda-handler` e suas dependências em um arquivo `.zip`. O arquivo `.zip` é registrado como uma camada do Lambda para uso por várias funções do Lambda.
- **`hms-lambda-rnp`**: registra as respostas de uma função do Lambda e, em seguida, usa-as para reproduzir a resposta. É possível usar esse modelo para simular respostas do Lambda para testes.

Criar seus próprios artefatos

A maioria dos casos de uso não exige que você modifique a implementação de referência. Se necessário, você mesmo pode modificar o código-fonte, criar os artefatos e carregá-los em um local do Amazon S3.

Antes de criar os artefatos, atualize as propriedades `hive.metastore.uris` e `hive.metastore.response.spill.location` no arquivo `hms.properties` no módulo `hms-lambda-func`.

Para criar os artefatos, você deve ter o Apache Maven instalado e executar o comando `mvn install`. Isso gera o arquivo `.zip` de camada na pasta de saída chamada `target` no módulo `hms-lambda-layer` e o arquivo `.jar` da função do Lambda no módulo `hms-lambd-func`.

Usar a consulta federada do Amazon Athena

Se você tiver dados em origens diferentes do Amazon S3, poderá usar a consulta federada do Athena para consultá-los no local ou criar pipelines para extrair os dados de várias origens e armazená-los no Amazon S3. Com a consulta federada do Athena, é possível executar consultas SQL nos dados armazenados em origens relacionais, não relacionais, de objetos e personalizadas.

O Athena usa conectores de origem dos dados operados no AWS Lambda para executar as consultas federadas. Um conector de origem dos dados é uma parte do código que converte entre sua origem dos dados de destino e o Athena. Pense em um conector como uma extensão do mecanismo de consulta do Athena. Os conectores de origem dos dados predefinidos do Athena foram desenvolvidos para origens de dados, como Amazon CloudWatch Logs, Amazon DynamoDB, Amazon DocumentDB e Amazon RDS, e para origens de dados relacionais compatíveis com JDBC, como MySQL e PostgreSQL, na licença do Apache 2.0. Também é possível usar o Athena Query Federation SDK para escrever conectores personalizados. Para escolher, configurar e implantar um conector de origem dos dados em sua conta, você pode usar os consoles do Athena e do Lambda ou o AWS Serverless Application Repository. Depois que implantar conectores de fonte de dados, o conector será associado a um catálogo que pode ser especificado nas consultas SQL. É possível

combinar instruções SQL de vários catálogos e abranger várias fontes de dados com uma única consulta.

Quando uma consulta é enviada para uma origem dos dados, o Athena invoca o conector correspondente para identificar as partes das tabelas que precisam ser lidas, gerencia o paralelismo e envia os predicados de filtro. Com base no usuário que envia a consulta, os conectores podem fornecer ou restringir o acesso a elementos de dados específicos. Os conectores usam o Apache Seta como o formato para retornar dados solicitados em uma consulta, o que permite que os conectores sejam implementados em linguagens como C, C++, Java, Python e Rust. Como os conectores são processados no Lambda, eles podem ser usados para acessar os dados de qualquer origem dos dados na nuvem ou on-premises que seja acessível pelo Lambda.

Para escrever o próprio conector de origem dos dados, você pode usar o Athena Query Federation SDK para personalizar um dos conectores predefinidos disponíveis e mantidos pelo Amazon Athena. Você pode modificar uma cópia do código-fonte do repositório do GitHub e usar a ferramenta de publicação do conector para criar seu próprio pacote do AWS Serverless Application Repository.

> **Note**
>
> Desenvolvedores de terceiro podem ter usado o Athena Query Federation SDK para escrever conectores de origem dos dados. Para problemas de suporte ou licenciamento com esses conectores de origem dos dados, entre em contato com o provedor dos conectores. Esses conectores não foram testados nem contam com suporte da AWS.

Para obter uma lista de conectores de origem dos dados escritos e testados pelo Athena, consulte Conectores de fonte de dados disponíveis.

Para obter informações sobre como gravar o próprio conector de origem de dados, consulte Example Athena connector (Exemplo de conector do Athena) no GitHub.

Considerações e limitações

- Versões do mecanismo: a consulta federada do Athena é permitida apenas no mecanismo Athena versão 2 e posteriores. Para obter informações sobre as versões do mecanismo do Athena, consulte Versionamento do mecanismo do Athena.
- Visualizações: é possível criar e consultar visualizações em fontes de dados federadas. As visualizações federadas são armazenadas no AWS Glue, não na fonte de dados subjacente. Para ter mais informações, consulte Consultar visualizações federadas.

- Operações de gravação: operações de gravação como INSERT INTO não são suportadas. Tentar fazer isso pode gerar a mensagem de erro: This operation is currently not supported for external catalogs (Atualmente, esta operação não é suportada para catálogos externos).
- Preço: para obter informações de preço, consulte Preços do Amazon Athena.

  Driver JDBC: para usar o driver JDBC com consultas federadas ou um metastore externo do Hive, inclua `MetadataRetrievalMethod=ProxyAPI` na string de conexão JDBC. Para obter informações sobre o driver JDBC, consulte Conectar-se ao Amazon Athena com JDBC.
- Secrets Manager: para usar o recurso de consulta federada do Athena com AWS Secrets Manager, você deve configurar um endpoint privado do Amazon VPC para o Secrets Manager. Para obter mais informações, consulte Criação de um endpoint privado da VPC para o Secrets Manager no Guia do usuário do AWS Secrets Manager.

Conectores de fonte de dados podem exigir acesso aos recursos a seguir para funcionar corretamente. Se você usar um conector predefinido, verifique as informações do conector para garantir que tenha configurado sua VPC corretamente. Além disso, garanta que os principais do IAM que executam consultas e criam conectores tenham os privilégios para as ações necessárias. Para ter mais informações, consulte Exemplo de políticas de permissões do IAM para permitir consulta federada do Athena.

- Amazon S3: além de gravar os resultados das consultas no local específico do Athena no Amazon S3, os conectores de dados gravam em um bucket de vazamento no Amazon S3. São necessárias conectividade e permissões para esse local do Amazon S3.
- Athena: as origens de dados precisam de conectividade com o Athena e vice-versa para conferir o status da consulta e evitar a verificação em excesso.
- AWS Glue Data Catalog: conectividade e permissões serão necessárias se o conector usar o Catálogo de dados para metadados complementares ou primários.

Vídeos

Assista aos vídeos a seguir para saber mais como usar a consulta federada do Athena.

Vídeo: Analisar os resultados da consulta federada do Amazon Athena no Amazon QuickSight

O vídeo a seguir demonstra como analisar os resultados de uma consulta federada do Athena no Amazon QuickSight.

Analyze results of federated query in Amazon Athena in Amazon QuickSight (Analisar os resultados da consulta federada do Amazon Athena no Amazon QuickSight)

Vídeo: Game Analytics Pipeline

O vídeo a seguir mostra como implantar um pipeline de dados escalável sem servidor para ingerir, armazenar e analisar dados de telemetria de jogos e serviços usando as consultas federadas do Amazon Athena.

Game Analytics Pipeline (Pipeline de análises de jogos)

Conectores de fonte de dados disponíveis

Esta seção lista os conectores de origem dos dados predefinidos do Athena que podem ser usados para consultar uma variedade de origens de dados externas ao Amazon S3. Para usar um conector em suas consultas do Athena, configure-o e implante-o em sua conta.

Considerações e limitações

- Alguns conectores pré-construídos exigem que você crie uma VPC e um grupo de segurança antes de poder usar o conector. Para obter informações sobre como criar VPCs, consulte Criar uma VPC para um conector de origem de dados (Criar uma VPC para um conector de origem de dados).
- Para usar o recurso de consulta federada do Athena com o AWS Secrets Manager, configure um endpoint privado do Amazon VPC para o Secrets Manager. Para obter mais informações, consulte Criação de um endpoint privado da VPC para o Secrets Manager no Guia do usuário do AWS Secrets Manager.
- Para conectores não compatíveis com passagem direta de predicados, as consultas que incluem um predicado demoram muito mais para serem executadas. Para conjuntos de dados pequenos, muito poucos dados são examinados, e as consultas levam em média cerca de 2 minutos. No entanto, para grandes conjuntos de dados, muitas consultas podem expirar.
- Algumas fontes de dados federadas usam terminologia para fazer referência a objetos de dados que são diferentes do Athena. Para ter mais informações, consulte Qualificadores de nomes do Athena e de tabelas federadas.
- Para conectores que não oferecem suporte à paginação ao listar tabelas, o serviço Web poderá atingir o tempo limite se o banco de dados tiver muitas tabelas e muitos metadados. Os seguintes conectores fornecem suporte à paginação para tabelas de listagem:

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

Mais informações

- Para obter informações sobre como implantar um conector de origem dos dados do Athena, consulte Implantar um conector de fonte de dados.
- Para obter informações sobre consultas que usam conectores de fonte de dados do Athena, consulte Executar consultas federadas.
- Para obter informações detalhadas sobre os conectores de origem dos dados do Athena, consulte Available connectors (Conectores disponíveis) no GitHub.

Conectores de origem de dados do Athena

- Conector do Amazon Athena para o Azure Data Lake Storage (ADLS) Gen2
- Conector do Amazon Athena para o Azure Synapse
- Conector do Amazon Athena para o Cloudera Hive
- Conector do Amazon Athena para o Cloudera Impala
- Conector do Amazon Athena para o CloudWatch
- Conector do Amazon Athena para o CloudWatch Metrics
- Conector do AWS CMDB no Amazon Athena
- Conector IBM Db2 do Amazon Athena
- Conector IBM Db2 AS/400 (Db2 iSeries) para Amazon Athena
- Conector do DocumentDB no Amazon Athena
- Conector do Amazon Athena para o DynamoDB
- Conector do Amazon Athena para o Google BigQuery

- Conector Google Cloud Storage para Amazon Athena
- Conector do Amazon Athena para o HBase
- Conector do Amazon Athena para a Hortonworks
- Conector do Apache Kafka do Amazon Athena
- Conector MSK do Amazon Athena
- Conector do Amazon Athena para o MySQL
- Conector do Amazon Athena para o Neptune
- Conector do Amazon Athena para o OpenSearch
- Conector do Amazon Athena para Oracle
- Conector do Amazon Athena para o PostgreSQL
- Conector do Amazon Athena para o Redis
- Conector do Amazon Athena para o Redshift
- Conector do Amazon Athena para o SAP HANA
- Conector do Amazon Athena para o Snowflake
- Conector do Amazon Athena para o Microsoft SQL Server
- Conector do Amazon Athena para o Teradata
- Conector do Amazon Athena para o Timestream
- Conector do Amazon Athena para o TPC Benchmark DS (TPC-DS)
- Conector do Amazon Athena para o Vertica

> **Note**
>
> O AthenaJdbcConnector (versão mais recente 2022.4.1) foi descontinuado. Ao invés dele, use um conector específico para banco de dados como aqueles para MySQL, Redshift ou PostgreSQL.

Conector do Amazon Athena para o Azure Data Lake Storage (ADLS) Gen2

O conector do Amazon Athena para o Azure Data Lake Storage (ADLS) Gen2 permite que o Amazon Athena execute consultas SQL em dados armazenados no ADLS. O Athena não pode acessar diretamente os arquivos armazenados no data lake.

- Fluxo de trabalho – O conector implementa a interface JDBC, que usa o driver `com.microsoft.sqlserver.jdbc.SQLServerDriver`. O conector envia consultas para o mecanismo do Azure Synapse, que então acessa o data lake.
- Manipulação de dados e S3 – Normalmente, o conector Lambda consulta dados diretamente sem transferência para o Amazon S3. No entanto, quando os dados retornados pela função Lambda excedem os limites do Lambda, os dados são gravados no bucket de vazamento do Amazon S3 que você especifica para que o Athena possa ler o excesso.
- Autenticação AAD – O AAD pode ser usado como um método de autenticação para o conector Azure Synapse. Para usar o AAD, a string de conexão JDBC usada pelo conector deve conter os parâmetros de URL `authentication=ActiveDirectoryServicePrincipal`, `AADSecurePrincipalId` e`AADSec`
  Esses parâmetros podem ser passados diretamente ou pelo Secrets Manager.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Limitações

- Não há suporte para operações de gravação de DDL.
- Em uma configuração de multiplexador, o prefixo e o bucket de derramamento são compartilhados em todas as instâncias do banco de dados.
- Quaisquer limites relevantes do Lambda. Para obter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do AWS Lambda.
- Em condições de filtro, você deve converter os tipos de dados date e timestamp para os tipos de dados apropriados.

Termos

Os termos a seguir são relativos ao conector do Azure Data Lake Storage Gen2.

- Instância do banco de dados: qualquer instância de um banco de dados implantado on-premises, no Amazon EC2 ou no Amazon RDS.

- Manipulador: um manipulador Lambda que acessa sua instância de banco de dados. Um manipulador pode ser para metadados ou para registros de dados.
- Manipulador de metadados: um manipulador Lambda que recupera metadados da sua instância de banco de dados.
- Manipulador de registros: um manipulador Lambda que recupera registros de dados da sua instância de banco de dados.
- Manipulador composto: um manipulador Lambda que recupera tanto metadados quanto registros de dados da sua instância de banco de dados.
- Propriedade ou parâmetro: uma propriedade do banco de dados usada pelos manipuladores para extrair informações do banco de dados. Você configura essas propriedades como variáveis de ambiente do Lambda.
- String de conexão: uma string de texto usada para estabelecer uma conexão com uma instância de banco de dados.
- Catálogo: um catálogo não AWS Glue registrado no Athena que é um prefixo obrigatório para a propriedade `connection_string`.
- Manipulador de multiplexação: um manipulador Lambda que pode aceitar e usar várias conexões de banco de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector do Azure Data Lake Storage Gen2.

String de conexão

Use uma string de conexão JDBC no seguinte formato para se conectar a uma instância de banco de dados.

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

Uso de um manipulador de multiplexação

É possível usar um multiplexador para se conectar a várias instâncias de banco de dados com uma única função do Lambda. As solicitações são encaminhadas por nome do catálogo. Use as seguintes classes no Lambda.

Manipulador	Classe
Manipulador composto	`DataLakeGen2MuxCompositeHandler`
Manipulador de metadados	`DataLakeGen2MuxMetadataHandler`
Manipulador de registros	`DataLakeGen2MuxRecordHandler`

Parâmetros do manipulador de multiplexação

Parâmetro	Descrição
`$`*`catalog`*`_connection_string`	Obrigatório. Uma string de conexão de instância de banco de dados. Prefixe a variável de ambiente com o nome do catálogo usado no Athena. Por exemplo, se o catálogo registrado no Athena for `mydatalakegentwocatalog` , então o nome da variável de ambiente será `mydatalakegentwocatalog_connection_string` .
`default`	Obrigatório. A string de conexão padrão. Essa string é usada quando o catálogo for `lambda:${` *`AWS_LAMBDA_FUNCTION_NAME`* `}`.

As propriedades de exemplo a seguir são para uma função do Lambda MUX do DataLakeGen2 que ofereça suporte a duas instâncias de banco de dados: `datalakegentwo1` (o padrão) e `datalakegentwo2`.

Propriedade	Valor
`default`	`datalakegentwo://jdbc:sqlserver://adlsgentwo1.`*`hostname`*`:`*`port`*`;databaseName=`*`database_name`*`;${`*`secret1_name`*`}`

Propriedade	Valor
`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 }`

Fornecimento de credenciais

Para fornecer um nome de usuário e uma senha para seu banco de dados na string de conexão JDBC, é possível usar as propriedades da string de conexão ou o AWS Secrets Manager.

- String de conexão: um nome de usuário e uma senha podem ser especificados como propriedades na string de conexão do JDBC.

  > **Important**
  >
  > Como prática recomendada de segurança, não use credenciais codificadas em suas variáveis de ambiente ou strings de conexão. Para obter informações sobre como mover seus segredos codificados para o AWS Secrets Manager, consulte Mover segredos codificados para o AWS Secrets Manager no Guia do usuário do AWS Secrets Manager.

- AWS Secrets Manager: para usar o recurso Athena Federated Query com o AWS Secrets Manager, a VPC conectada à sua função do Lambda deve ter acesso à Internet ou um endpoint da VPC para se conectar ao Secrets Manager.

  É possível colocar o nome de um segredo no AWS Secrets Manager na sua string de conexão JDBC. O conector substitui o nome secreto pelos valores de `username` e `password` do Secrets Manager.

  Para instâncias de banco de dados do Amazon RDS, esse suporte é totalmente integrado. Se você usa o Amazon RDS, é altamente recomendável usar o AWS Secrets Manager e rotação de credenciais. Se seu banco de dados não usar o Amazon RDS, armazene as credenciais em JSON no seguinte formato:

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

Exemplo de string de conexão com nome secreto

A string a seguir tem o nome secreto `${secret1_name}`.

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

O conector usa o nome secreto para recuperar segredos e fornecer o nome de usuário e a senha, como no exemplo a seguir.

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

Uso de um único manipulador de conexão

É possível usar os seguintes manipuladores de metadados e registros de conexão única para se conectar a uma única instância do Azure Data Lake Storage Gen2.

Tipo de manipulador	Classe
Manipulador composto	`DataLakeGen2CompositeHandler`
Manipulador de metadados	`DataLakeGen2MetadataHandler`
Manipulador de registros	`DataLakeGen2RecordHandler`

Parâmetros do manipulador de conexão única

Parâmetro	Descrição
`default`	Obrigatório. A string de conexão padrão.

Os manipuladores de conexão únicos oferecem suporte a uma instância de banco de dados e devem fornecer um parâmetro de string de conexão `default`. Todas as outras strings de conexão são ignoradas.

O exemplo de propriedade a seguir é para uma única instância do Azure Data Lake Storage Gen2 compatível com uma função do Lambda.

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

Parâmetros de derramamento

O SDK do Lambda pode derramar dados no Amazon S3. Todas as instâncias do banco de dados acessadas pela mesma função do Lambda derramam no mesmo local.

Parâmetro	Descrição
`spill_bucket`	Obrigatório. Nome do bucket de derramamento.
`spill_prefix`	Obrigatório. Prefixo de chave do bucket de derramamento.
`spill_put_request_headers`	(Opcional) Um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.

Suporte ao tipo de dados

A tabela a seguir mostra os correspondentes tipos de dados do ADLS Gen2 e do Arrow.

ADLS Gen2	Arrow
bit	TINYINT

ADLS Gen2	Arrow
tinyint	SMALLINT
smallint	SMALLINT
int	INT
bigint	BIGINT
decimal	DECIMAL
numeric	FLOAT8
smallmoney	FLOAT8
money	DECIMAL
float[24]	FLOAT4
float[53]	FLOAT8
real	FLOAT4
datetime	Date(MILLISECOND)
datetime2	Date(MILLISECOND)
smalldatetime	Date(MILLISECOND)
date	Date(DAY)
horário	VARCHAR
datetimeoffset	Date(MILLISECOND)
char[n]	VARCHAR
varchar[n/max]	VARCHAR

Partições e divisões

O Azure Data Lake Storage Gen2 usa armazenamento de blobs Gen2 compatível com Hadoop para armazenar arquivos de dados. Os dados desses arquivos são consultados no mecanismo do Azure Synapse. O mecanismo Azure Synapse trata os dados do Gen2 armazenados nos sistemas de arquivos como tabelas externas. As partições são implementadas com base no tipo de dados. Se os dados já tiverem sido particionados e distribuídos no sistema de armazenamento do Gen2, o conector recuperará os dados como uma única divisão.

Performance

O conector do Azure Data Lake Storage Gen2 apresenta uma performance de consulta mais lenta quando executa várias consultas ao mesmo tempo e está sujeito a controle de utilização.

O conector do Athena para o Azure Data Lake Storage Gen2 realiza a passagem direta de predicados para diminuir os dados examinados pela consulta. Predicados simples e expressões complexas são passados diretamente ao conector para reduzir a quantidade de dados examinados e o runtime de execução da consulta.

Predicados

Um predicado é uma expressão na cláusula `WHERE` de uma consulta SQL, que avalia para um valor booleano e filtra as linhas com base em várias condições. O conector do Athena para o Azure Data Lake Storage Gen2 pode combinar essas expressões e passá-las diretamente ao Azure Data Lake Storage Gen2 para melhorar a funcionalidade e reduzir a quantidade de dados examinados.

Os seguintes operadores do conector do Athena para o Azure Data Lake Storage Gen2 são compatíveis com passagem de predicados:

- Booleanos: E, OU, NÃO
- Igualdade:EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, NULL_IF, IS_NULL
- Aritméticos: ADICIONAR, SUBTRAIR, MULTIPLICAR, DIVIDIR, MÓDULO, NEGAR
- Outros:LIKE_PATTERN, IN

Exemplo de passagem direta combinada

Para ter recursos aprimorados de consulta, combine os tipos de passagem direta, como no seguinte exemplo:

```
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%');
```

Consultas de passagem

O conector do Azure Data Lake Storage Gen2 é compatível com consultas de passagem. As consultas de passagem usam uma função de tabela para enviar sua consulta completa para execução na fonte de dados.

Para usar consultas de passagem com o Azure Data Lake Storage Gen2, você pode empregar a seguinte sintaxe:

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

O exemplo de consulta a seguir envia uma consulta para uma fonte de dados no Azure Data Lake Storage Gen2. A consulta seleciona todas as colunas na tabela `customer`, limitando os resultados a 10.

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

Informações de licença

Ao usar esse conector, você reconhece a inclusão de componentes de terceiros, cuja lista pode ser encontrada no arquivo pom.xml desse conector, e concorda com os termos das respectivas licenças de terceiros fornecidas no arquivo LICENSE.txt em GitHub.com.

Recursos adicionais do

Para obter as informações mais recentes sobre versões do driver JDBC, consulte o arquivo pom.xml do conector do Azure Data Lake Storage Gen2 em GitHub.com.

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o Azure Synapse

O conector do Amazon Athena para o Azure Synapse Analytics permite que o Amazon Athena execute consultas SQL em seus bancos de dados do Azure Synapse usando JDBC.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Limitações

- Não há suporte para operações de gravação de DDL.
- Em uma configuração de multiplexador, o prefixo e o bucket de derramamento são compartilhados em todas as instâncias do banco de dados.
- Quaisquer limites relevantes do Lambda. Para obter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do AWS Lambda.
- Em condições de filtro, você deve converter os tipos de dados `Date` e `Timestamp` para o tipo de dados apropriado.
- Para pesquisar valores negativos dos tipos `Real` e `Float`, use o operador <= ou >=.
- Não há suporte para os tipos de dados `binary`, `varbinary`, `image` e `rowversion`.

Termos

Os termos a seguir estão relacionados ao conector Synapse.

- Instância do banco de dados: qualquer instância de um banco de dados implantado on-premises, no Amazon EC2 ou no Amazon RDS.
- Manipulador: um manipulador Lambda que acessa sua instância de banco de dados. Um manipulador pode ser para metadados ou para registros de dados.
- Manipulador de metadados: um manipulador Lambda que recupera metadados da sua instância de banco de dados.
- Manipulador de registros: um manipulador Lambda que recupera registros de dados da sua instância de banco de dados.

- Manipulador composto: um manipulador Lambda que recupera tanto metadados quanto registros de dados da sua instância de banco de dados.
- Propriedade ou parâmetro: uma propriedade do banco de dados usada pelos manipuladores para extrair informações do banco de dados. Você configura essas propriedades como variáveis de ambiente do Lambda.
- String de conexão: uma string de texto usada para estabelecer uma conexão com uma instância de banco de dados.
- Catálogo: um catálogo não AWS Glue registrado no Athena que é um prefixo obrigatório para a propriedade `connection_string`.
- Manipulador de multiplexação: um manipulador Lambda que pode aceitar e usar várias conexões de banco de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector Synapse.

String de conexão

Use uma string de conexão JDBC no seguinte formato para se conectar a uma instância de banco de dados.

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

Uso de um manipulador de multiplexação

É possível usar um multiplexador para se conectar a várias instâncias de banco de dados com uma única função do Lambda. As solicitações são encaminhadas por nome do catálogo. Use as seguintes classes no Lambda.

Manipulador	Classe
Manipulador composto	`SynapseMuxCompositeHandler`
Manipulador de metadados	`SynapseMuxMetadataHandler`
Manipulador de registros	`SynapseMuxRecordHandler`

Parâmetros do manipulador de multiplexação

Parâmetro	Descrição
`$catalog_connection_string`	Obrigatório. Uma string de conexão de instância de banco de dados. Prefixe a variável de ambiente com o nome do catálogo usado no Athena. Por exemplo, se o catálogo registrado no Athena for `mysynapsecatalog`, então o nome da variável de ambiente será `mysynapsecatalog_connection_string`.
`default`	Obrigatório. A string de conexão padrão. Essa string é usada quando o catálogo for `lambda:${AWS_LAMBDA_FUNCTION_NAME}`.

As propriedades de exemplo a seguir são para uma função do Lambda MUX do Synapse que ofereça suporte a duas instâncias de banco de dados: `synapse1` (o padrão) e `synapse2`.

Propriedade	Valor
`default`	`synapse://jdbc:synapse://synapse1.hostname:port;databaseName=<database_name>;${secret1_name}`
`synapse_catalog1_connection_string`	`synapse://jdbc:synapse://synapse1.hostname:port;databaseName=<database_name>;${secret1_name}`
`synapse_catalog2_connection_string`	`synapse://jdbc:synapse://synapse2.hostname:port;databaseName=<database_name>;${secret2_name}`

Fornecimento de credenciais

Para fornecer um nome de usuário e uma senha para seu banco de dados na string de conexão JDBC, é possível usar as propriedades da string de conexão ou o AWS Secrets Manager.

- String de conexão: um nome de usuário e uma senha podem ser especificados como propriedades na string de conexão do JDBC.

  > **Important**
  >
  > Como prática recomendada de segurança, não use credenciais codificadas em suas variáveis de ambiente ou strings de conexão. Para obter informações sobre como mover seus segredos codificados para o AWS Secrets Manager, consulte Mover segredos codificados para o AWS Secrets Manager no Guia do usuário do AWS Secrets Manager.

- AWS Secrets Manager: para usar o recurso Athena Federated Query com o AWS Secrets Manager, a VPC conectada à sua função do Lambda deve ter acesso à Internet ou um endpoint da VPC para se conectar ao Secrets Manager.

  É possível colocar o nome de um segredo no AWS Secrets Manager na sua string de conexão JDBC. O conector substitui o nome secreto pelos valores de `username` e `password` do Secrets Manager.

  Para instâncias de banco de dados do Amazon RDS, esse suporte é totalmente integrado. Se você usa o Amazon RDS, é altamente recomendável usar o AWS Secrets Manager e rotação de credenciais. Se seu banco de dados não usar o Amazon RDS, armazene as credenciais em JSON no seguinte formato:

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

Exemplo de string de conexão com nome secreto

A string a seguir tem o nome do segredo ${secret_name}.

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

O conector usa o nome secreto para recuperar segredos e fornecer o nome de usuário e a senha, como no exemplo a seguir.

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

Uso de um único manipulador de conexão

É possível usar os seguintes metadados de conexão única e manipuladores de registros para se conectar a uma única instância do Synapse.

Tipo de manipulador	Classe
Manipulador composto	`SynapseCompositeHandler`
Manipulador de metadados	`SynapseMetadataHandler`
Manipulador de registros	`SynapseRecordHandler`

Parâmetros do manipulador de conexão única

Parâmetro	Descrição
`default`	Obrigatório. A string de conexão padrão.

Os manipuladores de conexão únicos oferecem suporte a uma instância de banco de dados e devem fornecer um parâmetro de string de conexão `default`. Todas as outras strings de conexão são ignoradas.

O exemplo de propriedade a seguir é para uma única instância do Synapse com suporte em uma função do Lambda.

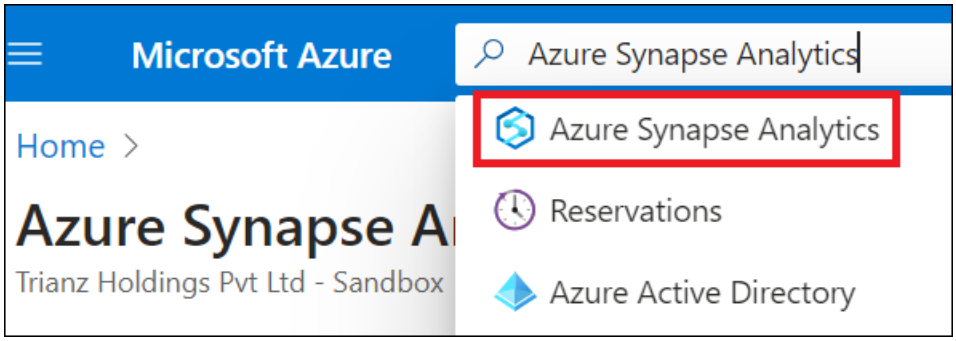
Propr de	Valor
defa	`synapse://jdbc:sqlserver://hostname:port;databaseName= <database_name> ;${secret_name }`

Autenticação da Configuração do Active Directory

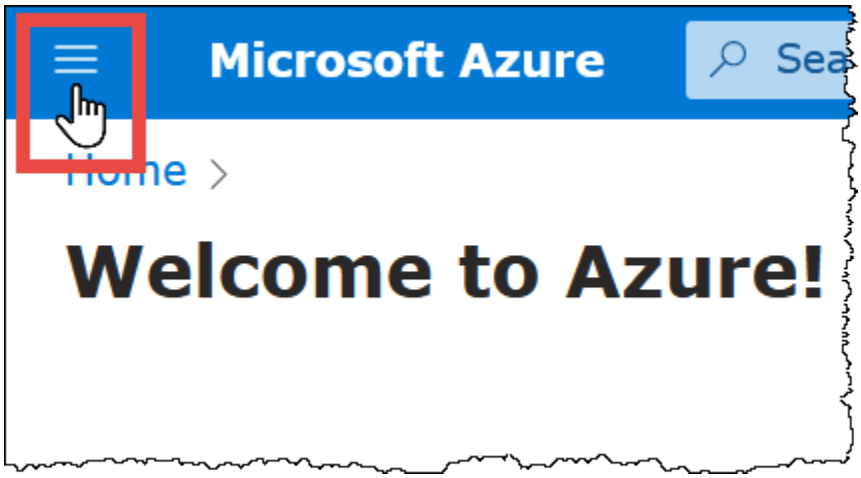
O conector Amazon Athena Azure Synapse oferece suporte à Autenticação do Microsoft Active Directory. Antes de começar, você deve configurar um usuário administrativo no portal do Microsoft Azure e usá-lo AWS Secrets Manager para criar um segredo.

Para configurar o usuário administrativo do Active Directory:

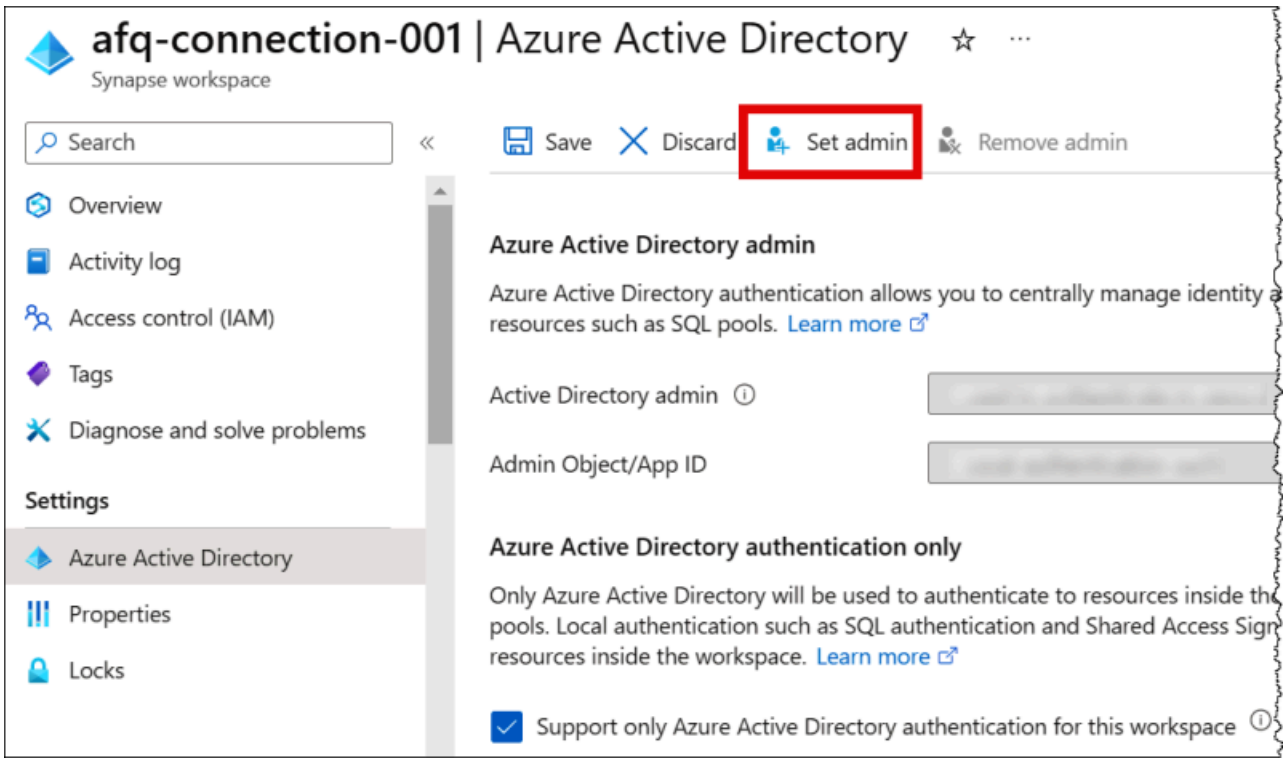
1. Usando uma conta que tenha privilégios administrativos, faça login no portal Microsoft Azure em https://portal.azure.com/.
2. Na caixa de pesquisa, insira Azure Synapse Analytics e escolha Azure Synapse Analytics.



3. Abra o menu à esquerda.



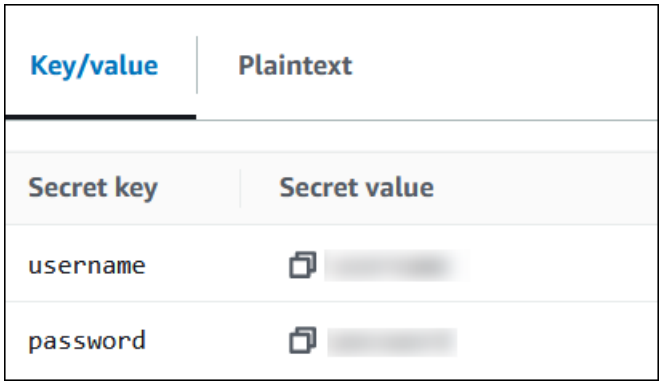
4. No painel de navegação, escolha Azure Active Directory.
5. Na guia Definir administrador, defina o administrador do Active Directory como um usuário novo ou existente.



6. Em AWS Secrets Manager, armazene as credenciais de nome de usuário e senha do administrador. Para obter informações sobre como criar um segredo no Secrets Manager, consulte Criação de um segredo do AWS Secrets Manager.

Para visualizar o segredo no Secrets Manager

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. No painel de navegação, escolha Secrets (Segredos).
3. Na página Secrets (Segredos), selecione o link do seu segredo.
4. Na página de detalhes do seu segredo, escolha Retrieve secret value (Recuperar valor do segredo).



Modificando a string de conexão

Para habilitar a Autenticação do Active Directory para o conector, modifique a string de conexão usando a seguinte sintaxe:

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

Usando o ActiveDirectoryServicePrincipal

O conector do Amazon Athena para o Azure Synapse também oferece suporte ao `ActiveDirectoryServicePrincipal`. Para habilitar isso, modifique a string de conexão da seguinte forma:

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

Para `secret_name`, especifique o ID do aplicativo ou cliente como o nome de usuário e o segredo de uma identidade de serviço principal como a senha.

Parâmetros de derramamento

O SDK do Lambda pode derramar dados no Amazon S3. Todas as instâncias do banco de dados acessadas pela mesma função do Lambda derramam no mesmo local.

Parâmetro	Descrição
`spill_bucket`	Obrigatório. Nome do bucket de derramamento.
`spill_prefix`	Obrigatório. Prefixo de chave do bucket de derramamento.
`spill_put_request_headers`	(Opcional) Um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.

Suporte ao tipo de dados

A tabela a seguir mostra os tipos de dados correspondentes para Synapse e Apache Arrow.

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

Synapse	Arrow
char[n]	VARCHAR
varchar[n/max]	VARCHAR
nchar[n]	VARCHAR
nvarchar[n/max]	VARCHAR

Partições e divisões

Uma partição é representada por uma única coluna de partição do tipo `varchar`. O Synapse oferece suporte a particionamento por intervalo, portanto, o particionamento é implementado extraindo a coluna da partição e o intervalo da partição das tabelas de metadados do Synapse. Esses valores de intervalo são usados para criar as divisões.

Performance

A seleção de um subconjunto de colunas diminui significativamente o runtime da consulta. O conector apresenta um controle de utilização significativo devido à simultaneidade.

O conector do Athena para o Synapse realiza a passagem direta de predicados para diminuir os dados examinados pela consulta. Predicados simples e expressões complexas são passados diretamente ao conector para reduzir a quantidade de dados examinados e o runtime de execução da consulta.

Predicados

Um predicado é uma expressão na cláusula `WHERE` de uma consulta SQL, que avalia para um valor booleano e filtra as linhas com base em várias condições. O conector do Athena para o Synapse pode combinar essas expressões e passá-las diretamente ao Synapse para melhorar a funcionalidade e reduzir a quantidade de dados examinados.

Os seguintes operadores do conector do Athena para o Synapse são compatíveis com a passagem direta de predicados:

- Booleanos: E, OU, NÃO
- Igualdade:EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, NULL_IF, IS_NULL

- Aritméticos: ADICIONAR, SUBTRAIR, MULTIPLICAR, DIVIDIR, MÓDULO, NEGAR
- Outros:LIKE_PATTERN, IN

Exemplo de passagem direta combinada

Para ter recursos aprimorados de consulta, combine os tipos de passagem direta, como no seguinte exemplo:

```
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%');
```

Consultas de passagem

O conector Synapse é compatível com consultas de passagem. As consultas de passagem usam uma função de tabela para enviar sua consulta completa para execução na fonte de dados.

Para usar consultas de passagem com o Synapse, você pode empregar a seguinte sintaxe:

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

O exemplo de consulta a seguir envia uma consulta para uma fonte de dados no Synapse. A consulta seleciona todas as colunas na tabela `customer`, limitando os resultados a 10.

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

Informações de licença

Ao usar esse conector, você reconhece a inclusão de componentes de terceiros, cuja lista pode ser encontrada no arquivo pom.xml desse conector, e concorda com os termos das respectivas licenças de terceiros fornecidas no arquivo LICENSE.txt em GitHub.com.

Recursos adicionais do

- Para ver um artigo que mostra como usar o Amazon QuickSight e a consulta federada Amazon Athena para criar painéis e visualizações em dados armazenados nos bancos de dados do Microsoft Azure Synapse, consulte Execute análises multinuvem usando o Amazon QuickSight, a consulta federada Amazon Athena e o Microsoft Azure Synapse no AWSBlog Big Data.
- Para obter as informações mais recentes sobre a versão do driver JDBC, consulte o arquivo pom.xml do conector Synapse em GitHub.com.
- Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o Cloudera Hive

O conector do Amazon Athena para o Cloudera Hive permite que o Athena execute consultas SQL na distribuição Hadoop do Cloudera Hive. O conector transforma suas consultas SQL do Athena na sintaxe equivalente do HiveQL.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.
- Configura uma VPC e um grupo de segurança antes de usar esse conector. Para ter mais informações, consulte Criar uma VPC para um conector de origem de dados.

Limitações

- Não há suporte para operações de gravação de DDL.
- Em uma configuração de multiplexador, o prefixo e o bucket de derramamento são compartilhados em todas as instâncias do banco de dados.
- Quaisquer limites relevantes do Lambda. Para obter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do AWS Lambda.

Termos

Os termos a seguir estão relacionados ao conector Cloudera Hive.

- Instância do banco de dados: qualquer instância de um banco de dados implantado on-premises, no Amazon EC2 ou no Amazon RDS.
- Manipulador: um manipulador Lambda que acessa sua instância de banco de dados. Um manipulador pode ser para metadados ou para registros de dados.
- Manipulador de metadados: um manipulador Lambda que recupera metadados da sua instância de banco de dados.
- Manipulador de registros: um manipulador Lambda que recupera registros de dados da sua instância de banco de dados.
- Manipulador composto: um manipulador Lambda que recupera tanto metadados quanto registros de dados da sua instância de banco de dados.
- Propriedade ou parâmetro: uma propriedade do banco de dados usada pelos manipuladores para extrair informações do banco de dados. Você configura essas propriedades como variáveis de ambiente do Lambda.
- String de conexão: uma string de texto usada para estabelecer uma conexão com uma instância de banco de dados.
- Catálogo: um catálogo não AWS Glue registrado no Athena que é um prefixo obrigatório para a propriedade `connection_string`.
- Manipulador de multiplexação: um manipulador Lambda que pode aceitar e usar várias conexões de banco de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector Cloudera Hive.

String de conexão

Use uma string de conexão JDBC no seguinte formato para se conectar a uma instância de banco de dados.

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

Uso de um manipulador de multiplexação

É possível usar um multiplexador para se conectar a várias instâncias de banco de dados com uma única função do Lambda. As solicitações são encaminhadas por nome do catálogo. Use as seguintes classes no Lambda.

Manipulador	Classe
Manipulador composto	`HiveMuxCompositeHandler`
Manipulador de metadados	`HiveMuxMetadataHandler`
Manipulador de registros	`HiveMuxRecordHandler`

Parâmetros do manipulador de multiplexação

Parâmetro	Descrição
`$`*`catalog`*`_connection_string`	Obrigatório. Uma string de conexão de instância de banco de dados. Prefixe a variável de ambiente com o nome do catálogo usado no Athena. Por exemplo, se o catálogo registrado no Athena for `myhivecatalog` , então o nome da variável de ambiente será `myhivecatalog_connection_string` .
`default`	Obrigatório. A string de conexão padrão. Essa string é usada quando o catálogo for `lambda:${` *`AWS_LAMBDA_FUNCTION_NAME`* `}`.

As propriedades de exemplo a seguir são para uma função do Lambda MUX do Hive que ofereça suporte a duas instâncias de banco de dados: `hive1` (o padrão) e `hive2`.

Propriedade	Valor
`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`

Fornecimento de credenciais

Para fornecer um nome de usuário e uma senha para seu banco de dados na string de conexão JDBC, é possível usar as propriedades da string de conexão ou o AWS Secrets Manager.

- String de conexão: um nome de usuário e uma senha podem ser especificados como propriedades na string de conexão do JDBC.

  > **Important**
  >
  > Como prática recomendada de segurança, não use credenciais codificadas em suas variáveis de ambiente ou strings de conexão. Para obter informações sobre como mover seus segredos codificados para o AWS Secrets Manager, consulte Mover segredos codificados para o AWS Secrets Manager no Guia do usuário do AWS Secrets Manager.

- AWS Secrets Manager: para usar o recurso Athena Federated Query com o AWS Secrets Manager, a VPC conectada à sua função do Lambda deve ter acesso à Internet ou um endpoint da VPC para se conectar ao Secrets Manager.

  É possível colocar o nome de um segredo no AWS Secrets Manager na sua string de conexão JDBC. O conector substitui o nome secreto pelos valores de `username` e `password` do Secrets Manager.

  Para instâncias de banco de dados do Amazon RDS, esse suporte é totalmente integrado. Se você usa o Amazon RDS, é altamente recomendável usar o AWS Secrets Manager e rotação de credenciais. Se seu banco de dados não usar o Amazon RDS, armazene as credenciais em JSON no seguinte formato:

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

Exemplo de string de conexão com nome secreto

A string a seguir tem o nome secreto `${Test/RDS/hive1}`.

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

O conector usa o nome secreto para recuperar segredos e fornecer o nome de usuário e a senha, como no exemplo a seguir.

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

Atualmente, o conector Cloudera Hive reconhece as propriedades do JDBC `UID` e `PWD`.

Uso de um único manipulador de conexão

É possível usar os seguintes metadados de conexão única e manipuladores de registros para se conectar a uma única instância do Cloudera Hive.

Tipo de manipulador	Classe
Manipulador composto	`HiveCompositeHandler`
Manipulador de metadados	`HiveMetadataHandler`
Manipulador de registros	`HiveRecordHandler`

Parâmetros do manipulador de conexão única

Parâmetro	Descrição
`default`	Obrigatório. A string de conexão padrão.

Os manipuladores de conexão únicos oferecem suporte a uma instância de banco de dados e devem fornecer um parâmetro de string de conexão `default`. Todas as outras strings de conexão são ignoradas.

O exemplo de propriedade a seguir é para uma única instância do Cloudera Hive com suporte em uma função do Lambda.

Propriedade	Valor
padrão	`hive://jdbc:hive2://hive1:10000/default?secret=${Test/RDS/hive1}`

Parâmetros de derramamento

O SDK do Lambda pode derramar dados no Amazon S3. Todas as instâncias do banco de dados acessadas pela mesma função do Lambda derramam no mesmo local.

Parâmetro	Descrição
`spill_bucket`	Obrigatório. Nome do bucket de derramamento.
`spill_prefix`	Obrigatório. Prefixo de chave do bucket de derramamento.
`spill_put_request_headers`	(Opcional) Um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.

Suporte ao tipo de dados

A tabela a seguir mostra os correspondentes tipos de dados do JDBC, do Cloudera Hive e do Arrow.

JDBC	Cloudera Hive	Arrow
Booleano	Booleano	Bit
Inteiro	TINYINT	Tiny
Short	SMALLINT	Smallint
Inteiro	INT	Int
Longo	BIGINT	Bigint
float	float4	Float4
Double	float8	Float8
Data	date	Data/Dia

JDBC	Cloudera Hive	Arrow
Timestamp	timestamp	Date Milli
String	VARCHAR	Varchar
Bytes	bytes	Varbinary
BigDecimal	Decimal	Decimal
ARRAY	N/D (ver nota)	Lista

> **Note**
>
> Atualmente, o Cloudera Hive não oferece suporte para os tipos agregados `ARRAY`, `MAP`, `STRUCT` ou `UNIONTYPE`. As colunas de tipos agregados são tratadas como colunas `VARCHAR` pelo SQL.

Partições e divisões

As partições são usadas para determinar como gerar divisões para o conector. O Athena constrói uma coluna sintética do tipo `varchar` que representa o esquema de particionamento da tabela para ajudar o conector a gerar divisões. O conector não modifica a definição real da tabela.

Performance

O Cloudera Hive oferece suporte a partições estáticas. O conector do Athena para o Cloudera Hive pode recuperar dados dessas partições em paralelo. Se você quiser consultar conjuntos de dados muito grandes com distribuição uniforme de partições, o particionamento estático é altamente recomendado. O conector Cloudera Hive é resiliente ao controle de utilização devido à simultaneidade.

O conector do Athena para o Cloudera Hive executa a passagem direta de predicados para diminuir os dados examinados pela consulta. Cláusulas `LIMIT`, predicados simples e expressões complexas são passados diretamente ao conector para reduzir a quantidade de dados examinados e o runtime de execução da consulta.

Cláusulas LIMIT

Uma instrução `LIMIT N` reduz os dados examinados pela consulta. Com a passagem direta de `LIMIT N`, o conector só retorna `N` linhas para o Athena.

Predicados

Um predicado é uma expressão na cláusula `WHERE` de uma consulta SQL, que avalia para um valor booleano e filtra as linhas com base em várias condições. O conector do Athena para o Cloudera Hive pode combinar essas expressões e passá-las diretamente ao Cloudera Hive para melhorar a funcionalidade e reduzir a quantidade de dados examinados.

Os seguintes operadores do conector do Athena para o Cloudera Hive são compatíveis com a passagem direta de predicados:

- Booleanos: E, OU, NÃO
- Igualdade: EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_NULL
- Aritméticos: ADICIONAR, SUBTRAIR, MULTIPLICAR, DIVIDIR, MÓDULO, NEGAR
- Outros:LIKE_PATTERN, IN

Exemplo de passagem direta combinada

Para ter recursos aprimorados de consulta, combine os tipos de passagem direta, como no seguinte exemplo:

```
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;
```

Consultas de passagem

O conector Cloudera Hive é compatível com consultas de passagem. As consultas de passagem usam uma função de tabela para enviar sua consulta completa para execução na fonte de dados.

Para usar consultas de passagem com o Cloudera Hive, você pode empregar a seguinte sintaxe:

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

O exemplo de consulta a seguir envia uma consulta para uma fonte de dados no Cloudera Hive. A consulta seleciona todas as colunas na tabela `customer`, limitando os resultados a 10.

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

Informações de licença

Ao usar esse conector, você reconhece a inclusão de componentes de terceiros, cuja lista pode ser encontrada no arquivo pom.xml desse conector, e concorda com os termos das respectivas licenças de terceiros fornecidas no arquivo LICENSE.txt em GitHub.com.

Recursos adicionais do

Para obter as informações mais recentes sobre a versão do driver JDBC, consulte o arquivo pom.xml do conector Cloudera Hive em GitHub.com.

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o Cloudera Impala

O conector do Amazon Athena para o Cloudera Impala permite que o Athena execute consultas SQL na distribuição do Cloudera Impala. O conector transforma suas consultas SQL do Athena na sintaxe equivalente do Impala.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.
- Configura uma VPC e um grupo de segurança antes de usar esse conector. Para ter mais informações, consulte Criar uma VPC para um conector de origem de dados.

Limitações

- Não há suporte para operações de gravação de DDL.
- Em uma configuração de multiplexador, o prefixo e o bucket de derramamento são compartilhados em todas as instâncias do banco de dados.
- Quaisquer limites relevantes do Lambda. Para obter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do AWS Lambda.

Termos

Os termos a seguir estão relacionados ao conector Cloudera Impala.

- Instância do banco de dados: qualquer instância de um banco de dados implantado on-premises, no Amazon EC2 ou no Amazon RDS.
- Manipulador: um manipulador Lambda que acessa sua instância de banco de dados. Um manipulador pode ser para metadados ou para registros de dados.
- Manipulador de metadados: um manipulador Lambda que recupera metadados da sua instância de banco de dados.
- Manipulador de registros: um manipulador Lambda que recupera registros de dados da sua instância de banco de dados.
- Manipulador composto: um manipulador Lambda que recupera tanto metadados quanto registros de dados da sua instância de banco de dados.
- Propriedade ou parâmetro: uma propriedade do banco de dados usada pelos manipuladores para extrair informações do banco de dados. Você configura essas propriedades como variáveis de ambiente do Lambda.
- String de conexão: uma string de texto usada para estabelecer uma conexão com uma instância de banco de dados.
- Catálogo: um catálogo não AWS Glue registrado no Athena que é um prefixo obrigatório para a propriedade `connection_string`.
- Manipulador de multiplexação: um manipulador Lambda que pode aceitar e usar várias conexões de banco de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector Cloudera Impala.

String de conexão

Use uma string de conexão JDBC no formato a seguir para se conectar a um cluster do Impala.

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

Uso de um manipulador de multiplexação

É possível usar um multiplexador para se conectar a várias instâncias de banco de dados com uma única função do Lambda. As solicitações são encaminhadas por nome do catálogo. Use as seguintes classes no Lambda.

Manipulador	Classe
Manipulador composto	`ImpalaMuxCompositeHandler`
Manipulador de metadados	`ImpalaMuxMetadataHandler`
Manipulador de registros	`ImpalaMuxRecordHandler`

Parâmetros do manipulador de multiplexação

Parâmetro	Descrição
`$catalog_connection_string`	Obrigatório. Uma string de conexão do cluster do Impala para um catálogo do Athena. Prefixe a variável de ambiente com o nome do catálogo usado no Athena. Por exemplo, se o catálogo registrado no Athena for `myimpalacatalog` , então o nome da variável de ambiente será `myimpalacatalog_connection_string` .
`default`	Obrigatório. A string de conexão padrão. Essa string é usada quando o catálogo for `lambda:${ AWS_LAMBDA_FUNCTION_NAME }`.

As propriedades de exemplo a seguir são para uma função do Lambda MUX do Impala que ofereça suporte a duas instâncias de banco de dados: `impala1` (o padrão) e `impala2`.

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

Fornecimento de credenciais

Para fornecer um nome de usuário e uma senha para seu banco de dados na string de conexão JDBC, é possível usar as propriedades da string de conexão ou o AWS Secrets Manager.

- String de conexão: um nome de usuário e uma senha podem ser especificados como propriedades na string de conexão do JDBC.

  > **Important**
  >
  > Como prática recomendada de segurança, não use credenciais codificadas em suas variáveis de ambiente ou strings de conexão. Para obter informações sobre como mover seus segredos codificados para o AWS Secrets Manager, consulte Mover segredos codificados para o AWS Secrets Manager no Guia do usuário do AWS Secrets Manager.

- AWS Secrets Manager: para usar o recurso Athena Federated Query com o AWS Secrets Manager, a VPC conectada à sua função do Lambda deve ter acesso à Internet ou um endpoint da VPC para se conectar ao Secrets Manager.

  É possível colocar o nome de um segredo no AWS Secrets Manager na sua string de conexão JDBC. O conector substitui o nome secreto pelos valores de `username` e `password` do Secrets Manager.

Para instâncias de banco de dados do Amazon RDS, esse suporte é totalmente integrado. Se você usa o Amazon RDS, é altamente recomendável usar o AWS Secrets Manager e rotação de credenciais. Se seu banco de dados não usar o Amazon RDS, armazene as credenciais em JSON no seguinte formato:

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

Exemplo de string de conexão com nome secreto

A string a seguir tem o nome secreto `${Test/impala1host}`.

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

O conector usa o nome secreto para recuperar segredos e fornecer o nome de usuário e a senha, como no exemplo a seguir.

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

Atualmente, o Cloudera Impala reconhece as propriedades do JDBC `UID` e `PWD`.

Uso de um único manipulador de conexão

É possível usar os seguintes metadados de conexão única e manipuladores de registros para se conectar a uma única instância do Cloudera Impala.

Tipo de manipulador	Classe
Manipulador composto	`ImpalaCompositeHandler`
Manipulador de metadados	`ImpalaMetadataHandler`
Manipulador de registros	`ImpalaRecordHandler`

Parâmetros do manipulador de conexão única

Parâmetro	Descrição
`default`	Obrigatório. A string de conexão padrão.

Os manipuladores de conexão únicos oferecem suporte a uma instância de banco de dados e devem fornecer um parâmetro de string de conexão `default`. Todas as outras strings de conexão são ignoradas.

O exemplo de propriedade a seguir é para uma única instância do Cloudera Impala com suporte em uma função do Lambda.

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

Parâmetros de derramamento

O SDK do Lambda pode derramar dados no Amazon S3. Todas as instâncias do banco de dados acessadas pela mesma função do Lambda derramam no mesmo local.

Parâmetro	Descrição
`spill_bucket`	Obrigatório. Nome do bucket de derramamento.
`spill_prefix`	Obrigatório. Prefixo de chave do bucket de derramamento.
`spill_put_request_headers`	(Opcional) Um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.

Suporte ao tipo de dados

A tabela a seguir mostra os correspondentes tipos de dados do JDBC, do Cloudera Impala e do Arrow.

JDBC	Cloudera Impala	Arrow
Booleano	Booleano	Bit
Inteiro	TINYINT	Tiny
Short	SMALLINT	Smallint
Inteiro	INT	Int
Longo	BIGINT	Bigint
float	float4	Float4
Double	float8	Float8
Data	date	Data/Dia
Timestamp	timestamp	Date Milli
String	VARCHAR	Varchar
Bytes	bytes	Varbinary
BigDecimal	Decimal	Decimal
ARRAY	N/D (ver nota)	Lista

Note

Atualmente, o Cloudera Impala não oferece suporte para os tipos agregados `ARRAY`, `MAP`, `STRUCT` ou `UNIONTYPE`. As colunas de tipos agregados são tratadas como colunas `VARCHAR` pelo SQL.

Partições e divisões

As partições são usadas para determinar como gerar divisões para o conector. O Athena constrói uma coluna sintética do tipo `varchar` que representa o esquema de particionamento da tabela para ajudar o conector a gerar divisões. O conector não modifica a definição real da tabela.

Performance

O Cloudera Impala oferece suporte a partições estáticas. O conector do Athena para o Cloudera Impala pode recuperar dados dessas partições em paralelo. Se você quiser consultar conjuntos de dados muito grandes com distribuição uniforme de partições, o particionamento estático é altamente recomendado. O conector Cloudera Impala é resiliente ao controle de utilização devido à simultaneidade.

O conector do Athena para o Cloudera Impala executa a passagem direta de predicados para diminuir os dados examinados pela consulta. Cláusulas `LIMIT`, predicados simples e expressões complexas são passados diretamente ao conector para reduzir a quantidade de dados examinados e o runtime de execução da consulta.

Cláusulas LIMIT

Uma instrução `LIMIT N` reduz os dados examinados pela consulta. Com a passagem direta de `LIMIT N`, o conector só retorna `N` linhas para o Athena.

Predicados

Um predicado é uma expressão na cláusula `WHERE` de uma consulta SQL, que avalia para um valor booleano e filtra as linhas com base em várias condições. O conector do Athena para o Cloudera Impala pode combinar essas expressões e passá-las diretamente ao Cloudera Impala para melhorar a funcionalidade e reduzir a quantidade de dados examinados.

Os seguintes operadores do conector do Athena para o Cloudera Impala são compatíveis com a passagem direta de predicados:

- Booleanos: E, OU, NÃO
- Igualdade: EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Aritméticos: ADICIONAR, SUBTRAIR, MULTIPLICAR, DIVIDIR, MÓDULO, NEGAR
- Outros:LIKE_PATTERN, IN

Exemplo de passagem direta combinada

Para ter recursos aprimorados de consulta, combine os tipos de passagem direta, como no seguinte exemplo:

```
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;
```

Consultas de passagem

O conector Cloudera Impala é compatível com consultas de passagem. As consultas de passagem usam uma função de tabela para enviar sua consulta completa para execução na fonte de dados.

Para usar consultas de passagem com o Cloudera Impala, você pode empregar a seguinte sintaxe:

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

O exemplo de consulta a seguir envia uma consulta para uma fonte de dados no Cloudera Impala. A consulta seleciona todas as colunas na tabela `customer`, limitando os resultados a 10.

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

Informações de licença

Ao usar esse conector, você reconhece a inclusão de componentes de terceiros, cuja lista pode ser encontrada no arquivo pom.xml desse conector, e concorda com os termos das respectivas licenças de terceiros fornecidas no arquivo LICENSE.txt em GitHub.com.

Recursos adicionais do

Para obter as informações mais recentes sobre a versão do driver JDBC, consulte o arquivo pom.xml do conector Cloudera Impala em GitHub.com.

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o CloudWatch

O conector do CloudWatch no Amazon Athena permite que o Amazon Athena se comunique com o CloudWatch para que você possa consultar os dados de log com SQL.

O conector mapeia os LogGroups como esquemas e cada LogStream como uma tabela. O conector também mapeia uma exibição `all_log_streams` especial que contém todos os LogStreams no LogGroup. Essa exibição permite consultar todos os logs em um LogGroup de uma só vez, em vez de pesquisar cada LogStream individualmente.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector CloudWatch.

- spill_bucket: especifica o bucket do Amazon S3 para dados que excedem os limites da função do Lambda.
- spill_prefix: (opcional) assume como padrão uma subpasta no `spill_bucket` especificado chamado `athena-federation-spill`. Recomendamos que você configure um ciclo de vida de armazenamento do Amazon S3 neste local para excluir derramamentos anteriores a um número predeterminado de dias ou horas.
- spill_put_request_headers: (opcional) um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.
- kms_key_id: (opcional) por padrão, todos os dados transmitidos para o Amazon S3 são criptografados usando o modo de criptografia autenticado AES-GCM e uma chave gerada aleatoriamente. Para que sua função do Lambda use chaves de criptografia mais fortes geradas pelo KMS, como `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, é possível especificar um ID de chave do KMS.

- disable_spill_encryption: (opcional) quando definido como `True`, desativa a criptografia do derramamento. É padronizado como `False`, para que os dados transmitidos para o S3 sejam criptografados usando o AES-GCM — usando uma chave gerada aleatoriamente ou o KMS para gerar chaves. Desativar a criptografia do derramamento pode melhorar a performance, especialmente se o local do derramamento usar criptografia no lado do servidor.

O conector também oferece suporte a Controle de congestionamento AIMD para lidar com eventos de limitação do CloudWatch por meio da construção `ThrottlingInvoker` do SDK do Amazon Athena Query Federation. É possível ajustar o comportamento do controle de utilização padrão definindo qualquer uma das seguintes variáveis de ambiente opcionais:

- throttle_initial_delay_ms: o atraso inicial da chamada aplicado após o primeiro evento de congestionamento. O padrão é de 10 milissegundos.
- throttle_max_delay_ms: o atraso máximo entre as chamadas. É possível derivar o TPS dividindo-o em 1000 ms. O padrão é de 1000 milissegundos.
- throttle_decrease_factor: o fator pelo qual o Athena reduz a taxa de chamadas. O padrão é de 0,5
- throttle_increase_ms: a taxa na qual o Athena diminui o atraso da chamada. O padrão é de 10 milissegundos.

Bancos de dados e tabelas

O conector CloudWatch do Athena mapeia seus LogGroups como esquemas (ou seja, bancos de dados) e cada LogStream como uma tabela. O conector também mapeia uma exibição `all_log_streams` especial que contém todos os LogStreams no LogGroup. Essa exibição permite consultar todos os logs em um LogGroup de uma só vez, em vez de pesquisar cada LogStream individualmente.

Cada tabela mapeada pelo conector CloudWatch do Athena possui o esquema a seguir. Esse esquema corresponde aos campos fornecidos pelo CloudWatch Logs.

- log_stream: um `VARCHAR` que contém o nome do LogStream de onde a linha pertence.
- time: um `INT64` que contém a época em que a linha de log foi gerada.
- message: um `VARCHAR` que contém a mensagem de log.

Exemplos

O exemplo a seguir mostra como realizar uma consulta SELECT em um LogStream especificado.

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

O exemplo a seguir mostra como usar a visualização all_log_streams para executar uma consulta em todos os LogStreams em um LogGroup especificado.

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

Permissões obrigatórias

Os detalhes completos sobre as políticas do IAM exigidas por esse conector podem ser encontrados na seção Policies do arquivo athena-cloudwatch.yaml. A lista a seguir resume as permissões necessárias.

- Acesso de gravação do Amazon S3: o conector requer acesso de gravação a um local no Amazon S3 para mostrar resultados de grandes consultas.
- Athena GetQueryExecution: o conector usa esta permissão para falhar rapidamente quando a consulta upstream do Athena é encerrada.
- CloudWatch Logs Read/Write (Leitura/gravação do CloudWatch Logs): o conector usa esta permissão para ler seus dados de log e gravar seus logs de diagnóstico.

Performance

O conector CloudWatch do Athena tenta otimizar as consultas em relação ao CloudWatch paralelizando as varreduras dos fluxos de log necessários para sua consulta. Para determinados filtros de período de tempo, a redução de predicados é realizada tanto na função do Lambda quanto no CloudWatch Logs.

Para obter a melhor performance, use somente letras minúsculas em nomes de grupos de logs e nomes de fluxos de logs. O uso de maiúsculas e minúsculas mistas faz com que o conector execute uma pesquisa que não diferencia maiúsculas de minúsculas e é mais computacionalmente intensiva.

Consultas de passagem

O conector do CloudWatch suporta consultas de passagem que usam a sintaxe de consulta do CloudWatch Logs Insights. Para obter mais informações sobre o CloudWatch Logs Insights, consulte Analisar dados de log com o CloudWatch Logs Insights no Guia do usuário do Amazon CloudWatch Logs.

Para criar consultas de passagem com o CloudWatch, use a seguinte sintaxe:

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

O exemplo a seguir filtra a consulta de passagem do CloudWatch para o campo `duration` quando ele não é igual a 1000.

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

Informações de licença

O projeto do conector CloudWatch do Amazon Athena é licenciado sob a Licença Apache-2.0.

Recursos adicionais do

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o CloudWatch Metrics

O conector das métricas do CloudWatch no Amazon Athena permite que o Amazon Athena consulte dados de métricas do CloudWatch com SQL.

Para obter informações sobre a publicação de métricas de consulta do próprio Athena no CloudWatch, consulte Controlar custos e monitorar consultas com métricas e eventos do CloudWatch.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector CloudWatch Metrics.

- spill_bucket: especifica o bucket do Amazon S3 para dados que excedem os limites da função do Lambda.
- spill_prefix: (opcional) assume como padrão uma subpasta no `spill_bucket` especificado chamado `athena-federation-spill`. Recomendamos que você configure um ciclo de vida de armazenamento do Amazon S3 neste local para excluir derramamentos anteriores a um número predeterminado de dias ou horas.
- spill_put_request_headers: (opcional) um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.
- kms_key_id: (opcional) por padrão, todos os dados transmitidos para o Amazon S3 são criptografados usando o modo de criptografia autenticado AES-GCM e uma chave gerada aleatoriamente. Para que sua função do Lambda use chaves de criptografia mais fortes geradas pelo KMS, como `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, é possível especificar um ID de chave do KMS.
- disable_spill_encryption: (opcional) quando definido como `True`, desativa a criptografia do derramamento. É padronizado como `False`, para que os dados transmitidos para o S3 sejam criptografados usando o AES-GCM — usando uma chave gerada aleatoriamente ou o KMS para gerar chaves. Desativar a criptografia do derramamento pode melhorar a performance, especialmente se o local do derramamento usar criptografia no lado do servidor.

O conector também oferece suporte a Controle de congestionamento AIMD para lidar com eventos de limitação do CloudWatch por meio da construção `ThrottlingInvoker` do SDK do Amazon Athena Query Federation. É possível ajustar o comportamento do controle de utilização padrão definindo qualquer uma das seguintes variáveis de ambiente opcionais:

- throttle_initial_delay_ms: o atraso inicial da chamada aplicado após o primeiro evento de congestionamento. O padrão é de 10 milissegundos.
- throttle_max_delay_ms: o atraso máximo entre as chamadas. É possível derivar o TPS dividindo-o em 1000 ms. O padrão é de 1000 milissegundos.
- throttle_decrease_factor: o fator pelo qual o Athena reduz a taxa de chamadas. O padrão é de 0,5
- throttle_increase_ms: a taxa na qual o Athena diminui o atraso da chamada. O padrão é de 10 milissegundos.

Bancos de dados e tabelas

O conector CloudWatch Metrics do Athena mapeia seus namespaces, dimensões, métricas e valores métricos em duas tabelas em um único esquema chamado `default`.

A tabela de métricas

A tabela `metrics` contém as métricas disponíveis conforme definido exclusivamente por uma combinação de namespace, conjunto e nome. A tabela `metrics` contém as colunas a seguir.

- namespace: um `VARCHAR` contendo o namespace.
- metric_name: um `VARCHAR` contendo o nome da métrica.
- dimensions: uma `LIST` de objetos `STRUCT` compostos por `dim_name (VARCHAR)` e `dim_value (VARCHAR)`.
- statistic: uma `LIST` de estatísticas `VARCH` (por exemplo, `p90`, `AVERAGE`, ...) disponíveis para a métrica.

A tabela metric_samples

A tabela `metric_samples` contém as amostras métricas disponíveis para cada métrica na tabela `metrics`. A tabela `metric_samples` contém as colunas a seguir.

- namespace: um `VARCHAR` que contém o namespace.
- metric_name: um `VARCHAR` que contém o nome da métrica.

- dimensions: uma `LIST` de objetos `STRUCT` compostos por `dim_name (VARCHAR)` e `dim_value (VARCHAR)`.
- dim_name: um campo `VARCHAR` de conveniência que você pode usar para filtrar facilmente por um único nome de dimensão.
- dim_value: um campo `VARCHAR` de conveniência que você pode usar para filtrar facilmente por um único valor de dimensão.
- period: um campo `INT` que representa o "período" da métrica em segundos (por exemplo, uma métrica de 60 segundos).
- timestamp: um campo `BIGINT` que representa o tempo de época em segundos para o qual a amostra métrica se destina.
- value: um `FLOAT8` campo que contém o valor da amostra.
- statistic: um `VARCHAR` que contém o tipo estatístico da amostra (por exemplo, `AVERAGE` ou `p90`).

Permissões obrigatórias

Os detalhes completos sobre as políticas do IAM exigidas por esse conector podem ser encontrados na seção `Policies` do arquivo athena-cloudwatch-metrics.yaml. A lista a seguir resume as permissões necessárias.

- Acesso de gravação do Amazon S3: o conector requer acesso de gravação a um local no Amazon S3 para mostrar resultados de grandes consultas.
- Athena GetQueryExecution: o conector usa esta permissão para falhar rapidamente quando a consulta upstream do Athena é encerrada.
- CloudWatch Metrics ReadOnly (Métricas do CloudWatch somente para leitura): o conector usa esta permissão para consultar seus dados de métricas.
- CloudWatch Logs Write: o conector usa este acesso para gravar seus registros de diagnóstico.

Performance

O conector CloudWatch Metrics do Athena tenta otimizar as consultas em relação ao CloudWatch Metrics paralelizando as varreduras dos fluxos de log necessários para sua consulta. Para determinados filtros de período de tempo, métricas, namespaces e dimensões, a redução de predicados é realizada tanto na função do Lambda quanto no CloudWatch Logs.

Informações de licença

O projeto do conector CloudWatch Metrics do Amazon Athena é licenciado sob a Licença Apache-2.0.

Recursos adicionais do

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do AWS CMDB no Amazon Athena

O conector do Amazon Athena para o AWS CMDB permite que o Athena se comunique com vários serviços da AWS para que você possa consultá-los com SQL.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector AWS CMDB.

- spill_bucket: especifica o bucket do Amazon S3 para dados que excedem os limites da função do Lambda.
- spill_prefix: (opcional) assume como padrão uma subpasta no `spill_bucket` especificado chamado `athena-federation-spill`. Recomendamos que você configure um ciclo de vida de armazenamento do Amazon S3 neste local para excluir derramamentos anteriores a um número predeterminado de dias ou horas.
- spill_put_request_headers: (opcional) um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.
- kms_key_id: (opcional) por padrão, todos os dados transmitidos para o Amazon S3 são criptografados usando o modo de criptografia autenticado AES-GCM e uma chave gerada aleatoriamente. Para que sua função do Lambda use chaves de criptografia mais fortes geradas

pelo KMS, como `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, é possível especificar um ID de chave do KMS.

- disable_spill_encryption: (opcional) quando definido como `True`, desativa a criptografia do derramamento. É padronizado como `False`, para que os dados transmitidos para o S3 sejam criptografados usando o AES-GCM — usando uma chave gerada aleatoriamente ou o KMS para gerar chaves. Desativar a criptografia do derramamento pode melhorar o desempenho, especialmente se o local do derramamento usar criptografia no lado do servidor.
- default_ec2_image_owner: (opcional) quando definido, controla o proprietário padrão da imagem do Amazon EC2 que filtra Imagens de máquina da Amazon (AMIs). Se você não definir esse valor e sua consulta na tabela de imagens do EC2 não incluir um filtro para o proprietário, seus resultados incluirão todas as imagens públicas.

Bancos de dados e tabelas

O conector AWS CMDB do Athena disponibiliza os seguintes bancos de dados e tabelas para consultar seu inventário de recursos da AWS. Para obter mais informações sobre as colunas disponíveis em cada tabela, execute uma declaração `DESCRIBE` *`database`*`.`*`table`* usando o console ou a API do Athena.

- ec2: esse banco de dados contém recursos relacionados ao Amazon EC2, incluindo os seguintes.

- ebs_volumes: contém detalhes dos seus volumes do Amazon EBS.
- ec2_instances: contém detalhes das suas instâncias do EC2.
- ec2_images: contém detalhes das suas imagens do EC2.
- routing_tables: contém detalhes das suas tabelas de roteamento de VPC.
- security_groups: contém detalhes dos seus grupos de segurança.
- subnets: contém detalhes das suas sub-redes VPC.
- vpcs: contém detalhes das suas VPCs.

- emr: este banco de dados contém recursos relacionados ao Amazon EMR, incluindo os seguintes.

- emr_clusters: contém detalhes dos seus clusters do EMR.

- rds: este banco de dados contém recursos relacionados ao Amazon RDS, incluindo os seguintes.

- rds_instances: contém detalhes das suas instâncias do RDS.

- s3: este banco de dados contém recursos relacionados ao RDS, incluindo os seguintes.

- buckets: contém detalhes dos buckets do Amazon S3.
- objects: contém detalhes dos seus objetos do Amazon S3, excluindo seu conteúdo.

Permissões obrigatórias

Os detalhes completos sobre as políticas do IAM exigidas por esse conector podem ser encontrados na seção `Policies` do arquivo athena-aws-cmdb.yaml. A lista a seguir resume as permissões necessárias.

- Acesso de gravação do Amazon S3: o conector requer acesso de gravação a um local no Amazon S3 para mostrar resultados de grandes consultas.
- Athena GetQueryExecution: o conector usa esta permissão para falhar rapidamente quando a consulta upstream do Athena é encerrada.
- S3 List (Listar S3): o conector usa esta permissão para listar seus buckets e objetos do Amazon S3.
- EC2 Describe (Descrever EC2): o conector usa esta permissão para descrever recursos, como suas instâncias do Amazon EC2, grupos de segurança, VPCs e volumes do Amazon EBS.
- EMR Describe / List (Descrever / listar EMR): o conector usa esta permissão para descrever seus clusters do EMR.
- RDS Describe (Descrever RDS): o conector usa esta permissão para descrever suas instâncias do RDS.

Performance

Atualmente, o conector AWS CMDB do Athena não oferece suporte a verificações paralelas. A redução de predicados é realizada dentro da função do Lambda. Sempre que possível, predicados parciais são enviados para os serviços que estão sendo consultados. Por exemplo, uma consulta para obter os detalhes de uma instância específica do Amazon EC2 chama a API do EC2 com o ID de instância específico para executar uma operação de descrição direcionada.

Informações de licença

O projeto do conector AWS CMDB do Amazon Athena é licenciado sob a Licença Apache-2.0.

Recursos adicionais do

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector IBM Db2 do Amazon Athena

O conector do Amazon Athena para Db2 possibilita que o Amazon Athena execute consultas SQL em seus bancos de dados do IBM Db2 usando o JDBC.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.
- Configura uma VPC e um grupo de segurança antes de usar esse conector. Para ter mais informações, consulte Criar uma VPC para um conector de origem de dados.

Limitações

- Não há suporte para operações de gravação de DDL.
- Em uma configuração de multiplexador, o prefixo e o bucket de derramamento são compartilhados em todas as instâncias do banco de dados.
- Quaisquer limites relevantes do Lambda. Para obter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do AWS Lambda.
- Em condições de filtro, você deve converter os tipos de dados date e timestamp para os tipos de dados apropriados.

Termos

Os termos a seguir estão relacionados ao conector Db2.

- Instância do banco de dados: qualquer instância de um banco de dados implantado on-premises, no Amazon EC2 ou no Amazon RDS.

- Manipulador: um manipulador Lambda que acessa sua instância de banco de dados. Um manipulador pode ser para metadados ou para registros de dados.
- Manipulador de metadados: um manipulador Lambda que recupera metadados da sua instância de banco de dados.
- Manipulador de registros: um manipulador Lambda que recupera registros de dados da sua instância de banco de dados.
- Manipulador composto: um manipulador Lambda que recupera tanto metadados quanto registros de dados da sua instância de banco de dados.
- Propriedade ou parâmetro: uma propriedade do banco de dados usada pelos manipuladores para extrair informações do banco de dados. Você configura essas propriedades como variáveis de ambiente do Lambda.
- String de conexão: uma string de texto usada para estabelecer uma conexão com uma instância de banco de dados.
- Catálogo: um catálogo não AWS Glue registrado no Athena que é um prefixo obrigatório para a propriedade `connection_string`.
- Manipulador de multiplexação: um manipulador Lambda que pode aceitar e usar várias conexões de banco de dados.

Parâmetros

Use as variáveis de ambiente do Lambda apresentadas nesta seção para configurar o conector Db2.

String de conexão

Use uma string de conexão JDBC no seguinte formato para se conectar a uma instância de banco de dados.

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

Uso de um manipulador de multiplexação

É possível usar um multiplexador para se conectar a várias instâncias de banco de dados com uma única função do Lambda. As solicitações são encaminhadas por nome do catálogo. Use as seguintes classes no Lambda.

Manipulador	Classe
Manipulador composto	`Db2MuxCompositeHandler`
Manipulador de metadados	`Db2MuxMetadataHandler`
Manipulador de registros	`Db2MuxRecordHandler`

Parâmetros do manipulador de multiplexação

Parâmetro	Descrição
`$`*`catalog`*`_connection_string`	Obrigatório. Uma string de conexão de instância de banco de dados. Prefixe a variável de ambiente com o nome do catálogo usado no Athena. Por exemplo, se o catálogo registrado no Athena for `mydbtwocatalog` , então o nome da variável de ambiente será `mydbtwocatalog_connection_string` .
`default`	Obrigatório. A string de conexão padrão. Essa string é usada quando o catálogo for `lambda:${` *`AWS_LAMBDA_FUNCTION_NAME`* `}`.

As propriedades exemplificadas a seguir são para uma função MUX do Lambda para o Db2 compatível com duas instâncias de banco de dados: `dbtwo1` (o padrão) e `dbtwo2`.

Propriedade	Valor
`default`	`dbtwo://jdbc:db2://dbtwo1.hostname:port/` *`database_name`* `:${`*`secret1_name`*`}`
`dbtwo_catalog1_connection_string`	`dbtwo://jdbc:db2://dbtwo1. hostname:port/` *`database_name`* `:${`*`secret1_name`*`}`
`dbtwo_catalog2_connection_string`	`dbtwo://jdbc:db2://dbtwo2. hostname:port/` *`database_name`* `:${`*`secret2_name`*`}`

Fornecimento de credenciais

Para fornecer um nome de usuário e uma senha para seu banco de dados na string de conexão JDBC, é possível usar as propriedades da string de conexão ou o AWS Secrets Manager.

- String de conexão: um nome de usuário e uma senha podem ser especificados como propriedades na string de conexão do JDBC.

  > **Important**
  >
  > Como prática recomendada de segurança, não use credenciais codificadas em suas variáveis de ambiente ou strings de conexão. Para obter informações sobre como mover seus segredos codificados para o AWS Secrets Manager, consulte Mover segredos codificados para o AWS Secrets Manager no Guia do usuário do AWS Secrets Manager.

- AWS Secrets Manager: para usar o recurso Athena Federated Query com o AWS Secrets Manager, a VPC conectada à sua função do Lambda deve ter acesso à Internet ou um endpoint da VPC para se conectar ao Secrets Manager.

  É possível colocar o nome de um segredo no AWS Secrets Manager na sua string de conexão JDBC. O conector substitui o nome secreto pelos valores de `username` e `password` do Secrets Manager.

  Para instâncias de banco de dados do Amazon RDS, esse suporte é totalmente integrado. Se você usa o Amazon RDS, é altamente recomendável usar o AWS Secrets Manager e rotação de credenciais. Se seu banco de dados não usar o Amazon RDS, armazene as credenciais em JSON no seguinte formato:

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

Exemplo de string de conexão com nome secreto

A string a seguir tem o nome secreto `${secret_name}`.

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

O conector usa o nome secreto para recuperar segredos e fornecer o nome de usuário e a senha, como no exemplo a seguir.

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

Uso de um único manipulador de conexão

É possível usar os metadados de conexão única e os manipuladores de registros a seguir para se conectar a uma única instância do Db2.

Tipo de manipulador	Classe
Manipulador composto	`Db2CompositeHandler`
Manipulador de metadados	`Db2MetadataHandler`
Manipulador de registros	`Db2RecordHandler`

Parâmetros do manipulador de conexão única

Parâmetro	Descrição
`default`	Obrigatório. A string de conexão padrão.

Os manipuladores de conexão únicos oferecem suporte a uma instância de banco de dados e devem fornecer um parâmetro de string de conexão `default`. Todas as outras strings de conexão são ignoradas.

A propriedade exemplificada a seguir é para uma única instância do Db2 compatível em uma função do Lambda.

Propriedade	Valor
`default`	dbtwo://jdbc:db2://hostname:port/*database_name* :${secret_name}

Parâmetros de derramamento

O SDK do Lambda pode derramar dados no Amazon S3. Todas as instâncias do banco de dados acessadas pela mesma função do Lambda derramam no mesmo local.

Parâmetro	Descrição
`spill_bucket`	Obrigatório. Nome do bucket de derramamento.
`spill_prefix`	Obrigatório. Prefixo de chave do bucket de derramamento.
`spill_put_request_headers`	(Opcional) Um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.

Suporte ao tipo de dados

A tabela a seguir mostra os correspondentes tipos de dados do JDBC e do Arrow.

Db2	Arrow
CHAR	VARCHAR
VARCHAR	VARCHAR
DATA	DATEDAY
TIME	VARCHAR
TIMESTAMP	DATEMILLI
DATETIME	DATEMILLI
BOOLEAN	BOOL
SMALLINT	SMALLINT

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

Partições e divisões

Uma partição é representada por uma ou mais colunas de partição do tipo `varchar`. O conector Db2 cria partições usando os esquemas de organização a seguir.

- Distribuir por hash
- Particionar por intervalo
- Organizar por dimensões

O conector recupera detalhes da partição, como o número de partições e o nome da coluna de uma ou mais tabelas de metadados do Db2. As divisões são criadas com base no número de partições identificadas.

Performance

O conector do Athena para o Db2 executa a passagem direta de predicados para diminuir os dados examinados pela consulta. Cláusulas `LIMIT`, predicados simples e expressões complexas são passados diretamente ao conector para reduzir a quantidade de dados examinados e o runtime de execução da consulta.

Cláusulas LIMIT

Uma instrução `LIMIT N` reduz os dados examinados pela consulta. Com a passagem direta de `LIMIT N`, o conector só retorna `N` linhas para o Athena.

Predicados

Um predicado é uma expressão na cláusula `WHERE` de uma consulta SQL, que avalia para um valor booleano e filtra as linhas com base em várias condições. O conector do Athena para o Db2 pode combinar essas expressões e passá-las diretamente ao Db2 para melhorar a funcionalidade e reduzir a quantidade de dados examinados.

Os seguintes operadores do conector do Athena para o Db2 são compatíveis com a passagem de predicados:

- Booleanos: E, OU, NÃO
- Igualdade: EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, IS_NULL
- Aritméticos: ADICIONAR, SUBTRAIR, MULTIPLICAR, DIVIDIR, MÓDULO, NEGAR
- Outros:LIKE_PATTERN, IN

Exemplo de passagem direta combinada

Para ter recursos aprimorados de consulta, combine os tipos de passagem direta, como no seguinte exemplo:

```
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;
```

Consultas de passagem

O conector Db2 é compatível com consultas de passagem. As consultas de passagem usam uma função de tabela para enviar sua consulta completa para execução na fonte de dados.

Para usar consultas de passagem com o Db2, você pode empregar a seguinte sintaxe:

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

```
        ))
```

O exemplo de consulta a seguir envia uma consulta para uma fonte de dados no Db2. A consulta seleciona todas as colunas na tabela `customer`, limitando os resultados a 10.

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

Informações de licença

Ao usar esse conector, você reconhece a inclusão de componentes de terceiros, cuja lista pode ser encontrada no arquivo pom.xml desse conector, e concorda com os termos das respectivas licenças de terceiros fornecidas no arquivo LICENSE.txt em GitHub.com.

Recursos adicionais do

Para obter as informações mais recentes sobre a versão do driver JDBC, consulte o arquivo pom.xml para o conector Db2 em GitHub.com.

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector IBM Db2 AS/400 (Db2 iSeries) para Amazon Athena

O conector do Amazon Athena para Db2 AS/400 habilita o Amazon Athena a executar consultas SQL em seus bancos de dados IBM Db2 AS/400 (Db2 iSeries) usando o JDBC.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.
- Configura uma VPC e um grupo de segurança antes de usar esse conector. Para ter mais informações, consulte Criar uma VPC para um conector de origem de dados.

Limitações

- Não há suporte para operações de gravação de DDL.

- Em uma configuração de multiplexador, o prefixo e o bucket de derramamento são compartilhados em todas as instâncias do banco de dados.
- Quaisquer limites relevantes do Lambda. Para obter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do AWS Lambda.
- Em condições de filtro, você deve converter os tipos de dados date e timestamp para os tipos de dados apropriados.

Termos

Os termos a seguir estão relacionados ao conector Db2 AS/400.

- Instância do banco de dados: qualquer instância de um banco de dados implantado on-premises, no Amazon EC2 ou no Amazon RDS.
- Manipulador: um manipulador Lambda que acessa sua instância de banco de dados. Um manipulador pode ser para metadados ou para registros de dados.
- Manipulador de metadados: um manipulador Lambda que recupera metadados da sua instância de banco de dados.
- Manipulador de registros: um manipulador Lambda que recupera registros de dados da sua instância de banco de dados.
- Manipulador composto: um manipulador Lambda que recupera tanto metadados quanto registros de dados da sua instância de banco de dados.
- Propriedade ou parâmetro: uma propriedade do banco de dados usada pelos manipuladores para extrair informações do banco de dados. Você configura essas propriedades como variáveis de ambiente do Lambda.
- String de conexão: uma string de texto usada para estabelecer uma conexão com uma instância de banco de dados.
- Catálogo: um catálogo não AWS Glue registrado no Athena que é um prefixo obrigatório para a propriedade `connection_string`.
- Manipulador de multiplexação: um manipulador Lambda que pode aceitar e usar várias conexões de banco de dados.

Parâmetros

Use as variáveis de ambiente do Lambda apresentadas nesta seção para configurar o conector Db2 AS/400.

String de conexão

Use uma string de conexão JDBC no seguinte formato para se conectar a uma instância de banco de dados.

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

Uso de um manipulador de multiplexação

É possível usar um multiplexador para se conectar a várias instâncias de banco de dados com uma única função do Lambda. As solicitações são encaminhadas por nome do catálogo. Use as seguintes classes no Lambda.

Manipulador	Classe
Manipulador composto	`Db2MuxCompositeHandler`
Manipulador de metadados	`Db2MuxMetadataHandler`
Manipulador de registros	`Db2MuxRecordHandler`

Parâmetros do manipulador de multiplexação

Parâmetro	Descrição
`$`*`catalog`*`_connection_string`	Obrigatório. Uma string de conexão de instância de banco de dados. Prefixe a variável de ambiente com o nome do catálogo usado no Athena. Por exemplo, se o catálogo registrado no Athena for `mydb2as400catalog` , então o nome da variável de ambiente será `mydb2as400catalog_connection_string` .
`default`	Obrigatório. A string de conexão padrão. Essa string é usada quando o catálogo for `lambda:${` *`AWS_LAMBDA_FUNCTION_NAME`* `}`.

As propriedades exemplificadas a seguir são para uma função MUX do Lambda para o Db2 compatível com duas instâncias de banco de dados: `db2as4001` (o padrão) e `db2as4002`.

Propriedade	Valor
`default`	`db2as400://jdbc:as400://` *`<ip_address>`* `;`*`<properties>`* `;:${`*`<secret name>`*`};`
`db2as400_catalog1_connection_string`	`db2as400://jdbc:as400://db2as4001.hostname/:${` *`secret1_name`* `}`
`db2as400_catalog2_connection_string`	`db2as400://jdbc:as400://db2as4002.hostname/:${` *`secret2_name`* `}`
`db2as400_catalog3_connection_string`	`db2as400://jdbc:as400://` *`<ip_address>`* `;user=`*`<username>`* `;password=` *`<password>`* `;`*`<properties>`* `;`

Fornecimento de credenciais

Para fornecer um nome de usuário e uma senha para seu banco de dados na string de conexão JDBC, é possível usar as propriedades da string de conexão ou o AWS Secrets Manager.

- String de conexão: um nome de usuário e uma senha podem ser especificados como propriedades na string de conexão do JDBC.

  > **Important**
  >
  > Como prática recomendada de segurança, não use credenciais codificadas em suas variáveis de ambiente ou strings de conexão. Para obter informações sobre como mover seus segredos codificados para o AWS Secrets Manager, consulte Mover segredos codificados para o AWS Secrets Manager no Guia do usuário do AWS Secrets Manager.

- AWS Secrets Manager: para usar o recurso Athena Federated Query com o AWS Secrets Manager, a VPC conectada à sua função do Lambda deve ter acesso à Internet ou um endpoint da VPC para se conectar ao Secrets Manager.

É possível colocar o nome de um segredo no AWS Secrets Manager na sua string de conexão JDBC. O conector substitui o nome secreto pelos valores de `username` e `password` do Secrets Manager.

Para instâncias de banco de dados do Amazon RDS, esse suporte é totalmente integrado. Se você usa o Amazon RDS, é altamente recomendável usar o AWS Secrets Manager e rotação de credenciais. Se seu banco de dados não usar o Amazon RDS, armazene as credenciais em JSON no seguinte formato:

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

Exemplo de string de conexão com nome secreto

A string a seguir tem o nome secreto `${secret_name}`.

```
db2as400://jdbc:as400://<ip_address>;<properties>;:${<secret_name>};
```

O conector usa o nome secreto para recuperar segredos e fornecer o nome de usuário e a senha, como no exemplo a seguir.

```
db2as400://jdbc:as400://<ip_address>;user=<username>;password=<password>;<properties>;
```

Uso de um único manipulador de conexão

É possível usar os metadados de conexão única e os manipuladores de registros a seguir para se conectar a uma única instância do Db2 AS/400.

Tipo de manipulador	Classe
Manipulador composto	`Db2CompositeHandler`
Manipulador de metadados	`Db2MetadataHandler`
Manipulador de registros	`Db2RecordHandler`

Parâmetros do manipulador de conexão única

Parâmetro	Descrição
`default`	Obrigatório. A string de conexão padrão.

Os manipuladores de conexão únicos oferecem suporte a uma instância de banco de dados e devem fornecer um parâmetro de string de conexão `default`. Todas as outras strings de conexão são ignoradas.

A propriedade exemplificada a seguir é para uma única instância do Db2 AS/400 compatível em uma função do Lambda.

Propriedade	Valor
`default`	`db2as400://jdbc:as400://` *`<ip_address>`* `;`*`<properties>`* `;: ${`*`<secret_name>`* `};`

Parâmetros de derramamento

O SDK do Lambda pode derramar dados no Amazon S3. Todas as instâncias do banco de dados acessadas pela mesma função do Lambda derramam no mesmo local.

Parâmetro	Descrição
`spill_bucket`	Obrigatório. Nome do bucket de derramamento.
`spill_prefix`	Obrigatório. Prefixo de chave do bucket de derramamento.
`spill_put_request_headers`	(Opcional) Um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.

Suporte ao tipo de dados

A tabela a seguir mostra os correspondentes tipos de dados do JDBC e do Apache Arrow.

Db2 AS/400	Arrow
CHAR	VARCHAR
VARCHAR	VARCHAR
DATA	DATEDAY
TIME	VARCHAR
TIMESTAMF	DATEMILLI
DATETIME	DATEMILLI
BOOLEAN	BOOL
SMALLINT	SMALLINT
INTEGER	INT
BIGINT	BIGINT
DECIMAL	DECIMAL
REAL	FLOAT8
DOUBLE	FLOAT8
DECFLOAT	FLOAT8

Partições e divisões

Uma partição é representada por uma ou mais colunas de partição do tipo `varchar`. O conector Db2 AS/400 cria partições usando os esquemas de organização a seguir.

- Distribuir por hash

- Particionar por intervalo
- Organizar por dimensões

O conector recupera detalhes da partição, como o número de partições e o nome da coluna, de uma ou mais tabelas de metadados do Db2 AS/400. As divisões são criadas com base no número de partições identificadas.

Performance

Para melhorar o desempenho, use o pushdown de predicados para consultar o Athena, como nos exemplos a seguir.

```
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'
```

Consultas de passagem

O conector Db2 AS/400 é compatível com consultas de passagem. As consultas de passagem usam uma função de tabela para enviar sua consulta completa para execução na fonte de dados.

Para usar consultas de passagem com o Db2 AS/400, você pode empregar a seguinte sintaxe:

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

O exemplo de consulta a seguir envia uma consulta para uma fonte de dados no Db2 AS/400. A consulta seleciona todas as colunas na tabela `customer`, limitando os resultados a 10.

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

Informações de licença

Ao usar esse conector, você reconhece a inclusão de componentes de terceiros, cuja lista pode ser encontrada no arquivo pom.xml desse conector, e concorda com os termos das respectivas licenças de terceiros fornecidas no arquivo LICENSE.txt em GitHub.com.

Recursos adicionais do

Para obter as informações mais recentes sobre a versão do driver JDBC, consulte o arquivo pom.xml para o conector Db2 AS/400 em GitHub.com.

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do DocumentDB no Amazon Athena

O conector Amazon Athena para o DocumentDB permite que o Athena se comunique com suas instâncias do DocumentDB para que você possa consultar seus dados do DocumentDB com SQL. O conector também funciona com qualquer endpoint compatível com o MongoDB.

Diferentemente dos armazenamentos de dados relacionais tradicionais, as coleções do Amazon DocumentDB não têm um esquema definido. O DocumentDB não tem um armazenamento de metadados. Cada entrada em uma coleção do DocumentDB pode ter diferentes campos e tipos de dados.

O conector DocumentDB oferece suporte a dois mecanismos para gerar informações do esquema da tabela: inferência básica de esquema e metadados do AWS Glue Data Catalog.

A inferência de esquema é o padrão. Essa opção examina um pequeno número de documentos em sua coleção, forma uma união de todos os campos e força campos que têm tipos de dados não sobrepostos. Essa opção funciona bem para coleções que têm, em sua maioria, entradas uniformes.

Para coleções com uma maior variedade de tipos de dados, o conector oferece suporte à recuperação de metadados do AWS Glue Data Catalog. Se o conector vê um banco de dados do AWS Glue e uma tabela que correspondam aos nomes de seu banco de dados e coleção do DocumentDB, ele obtém suas informações de esquema da tabela AWS Glue correspondente. Quando você cria sua tabela AWS Glue, recomendamos que você a torne um superconjunto de todos os campos que você talvez queira acessar da sua coleção do DocumentDB.

Se você tiver o Lake Formation habilitado em sua conta, o perfil do IAM para seu conector Lambda federado para Athena que você implantou no AWS Serverless Application Repository deve ter acesso de leitura ao AWS Glue Data Catalog no Lake Formation.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector DocumentdDB.

- spill_bucket: especifica o bucket do Amazon S3 para dados que excedem os limites da função do Lambda.
- spill_prefix: (opcional) assume como padrão uma subpasta no `spill_bucket` especificado chamado `athena-federation-spill`. Recomendamos que você configure um ciclo de vida de armazenamento do Amazon S3 neste local para excluir derramamentos anteriores a um número predeterminado de dias ou horas.
- spill_put_request_headers: (opcional) um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.
- kms_key_id: (opcional) por padrão, todos os dados transmitidos para o Amazon S3 são criptografados usando o modo de criptografia autenticado AES-GCM e uma chave gerada aleatoriamente. Para que sua função do Lambda use chaves de criptografia mais fortes geradas pelo KMS, como `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, é possível especificar um ID de chave do KMS.
- disable_spill_encryption: (opcional) quando definido como `True`, desativa a criptografia do derramamento. É padronizado como `False`, para que os dados transmitidos para o S3 sejam criptografados usando o AES-GCM — usando uma chave gerada aleatoriamente ou o KMS para gerar chaves. Desativar a criptografia do derramamento pode melhorar a performance, especialmente se o local do derramamento usar criptografia no lado do servidor.
- disable_glue: (opcional) se estiver presente e definido como verdadeiro, o conector não tentará recuperar metadados complementares do AWS Glue.
- glue_catalog: (opcional) use essa opção para especificar um catálogo do AWS Glue entre contas. Por padrão, o conector tenta obter metadados de sua própria conta do AWS Glue.

- default_docdb: se estiver presente, especifica uma string de conexão do DocumentDB a ser usada quando não existir uma variável de ambiente específica do catálogo.
- disable_projection_and_casing: (opcional) desativa a projeção e a capitalização. Use se quiser consultar tabelas do Amazon DocumentDB que usam nomes de colunas que diferenciam letras maiúsculas de minúsculas. O parâmetro `disable_projection_and_casing` usa os seguintes valores para especificar o comportamento do mapeamento de capitalização e coluna:
  - false (falso): essa é a configuração padrão. A projeção está ativada e o conector espera que todos os nomes das colunas estejam usando letras minúsculas.
  - true (verdadeiro): desativa a projeção e o uso de maiúsculas e minúsculas. Ao usar o parâmetro `disable_projection_and_casing`, lembre-se dos seguintes pontos:
    - O uso do parâmetro pode resultar em um maior uso de largura de banda. Além disso, se a função do Lambda não estiver na mesma Região da AWS que a fonte de dados, custos mais altos de transferência padrão entre regiões da AWS serão gerados como resultado do maior uso de largura de banda. Para obter mais informações sobre os custos de transferência entre regiões, consulte Cobranças de transferência de dados da AWS para arquiteturas com servidor e com tecnologia sem servidor no blog da rede de parceiros da AWS.
    - Como um número maior de bytes é transferido, e o maior número de bytes exige um maior tempo de desserialização, a latência geral pode aumentar.
- enable_case_insensitive_match: (opcional) quando `true`, realiza pesquisas sem distinção entre maiúsculas e minúsculas em nomes de esquemas e de tabelas no Amazon DocumentDB. O padrão é `false`. Use se sua consulta contiver nomes de esquemas ou de tabelas com letras maiúsculas.

Especificação de strings de conexão

É possível fornecer uma ou mais propriedades que definem os detalhes da conexão do DocumentDB para as instâncias do DocumentDB que você usar com o conector. Para fazer isso, defina uma variável de ambiente Lambda que corresponda ao nome do catálogo que você deseja usar no Athena. Por exemplo, suponha que você queira usar as seguintes consultas para consultar duas instâncias diferentes do DocumentDB no Athena:

```
SELECT * FROM "docdb_instance_1".database.table
```

```
SELECT * FROM "docdb_instance_2".database.table
```

Antes de usar essas duas instruções de SQL, você deverá adicionar duas variáveis de ambiente à sua função do Lambda: `docdb_instance_1` e `docdb_instance_2`. O valor para cada uma deverá ser uma string de conexão do DocumentDB no seguinte formato:

```
mongodb://:@:/?ssl=true&ssl_ca_certs=rds-combined-ca-bundle.pem&replicaSet=rs0
```

Uso de segredos

Opcionalmente, é possível usar o AWS Secrets Manager para obter parte ou todo o valor dos detalhes da string de conexão. Para usar o recurso Athena Federated Query com o Secrets Manager, a VPC conectada à sua função do Lambda deve ter acesso à Internet ou um endpoint da VPC para se conectar ao Secrets Manager.

Se você usar a sintaxe `${my_secret}` para colocar o nome de um segredo do Secrets Manager na string de conexão, o conector substituirá `${my_secret}` por seu valor em texto sem formatação do Secrets Manager exatamente. Os segredos devem ser armazenados como um segredo de texto sem formatação com o valor de *`<username>`*`:`*`<password>`*. Os segredos armazenados como `{username:`*`<username>`*`,password:`*`<password>`*`}` não serão passados corretamente para a string de conexão.

Os segredos também podem ser usados para toda a string de conexão, e o nome de usuário e a senha podem ser definidos no segredo.

Por exemplo, suponha que você defina a variável de ambiente Lambda para `docdb_instance_1` com o seguinte valor:

```
mongodb://${docdb_instance_1_creds}@myhostname.com:123/?ssl=true&ssl_ca_certs=rds-
combined-ca-bundle.pem&replicaSet=rs0
```

O SDK do Athena Query Federation tenta automaticamente recuperar um segredo chamado `docdb_instance_1_creds` do Secrets Manager e injetar esse valor no lugar de `${docdb_instance_1_creds}`. Qualquer parte da string de conexão que esteja delimitada pela combinação de caracteres `${ }` será é interpretada como um segredo do Secrets Manager. Se você especificar um nome secreto que o conector não consiga encontrar no Secrets Manager, o conector não substituirá o texto.

Configuração de bancos de dados e tabelas no AWS Glue

Como o recurso de inferência de esquema integrado do conector examina um número limitado de documentos e oferece suporte apenas um subconjunto de tipos de dados, talvez você queira usar o AWS Glue para metadados, em vez disso.

Para habilitar uma tabela do AWS Glue para uso com o Amazon DocumentDB, você deve ter um banco de dados do AWS Glue e uma tabela para o banco de dados DocumentDB e a coleção para a qual você deseja fornecer metadados complementares.

Para usar uma tabela AWS Glue para metadados complementares

1. Quando você edita a tabela e o banco de dados no console do AWS Glue, adicione a seguinte propriedade de tabela:

   - timestream-metadata-flag: essa propriedade indica ao conector DocumentDB que o conector pode usar a tabela para metadados complementares. É possível fornecer qualquer valor para `docdb-metadata-flag`, desde que a propriedade `docdb-metadata-flag` esteja presente na lista de propriedades da tabela.

2. (Opcional) Adicione a propriedade de tabela sourceTable. Essa propriedade define o nome da tabela de origem no Amazon DocumentDB. Use essa propriedade caso as regras de nomenclatura de tabelas do AWS Glue impeçam que você crie uma tabela do AWS Glue com o mesmo nome da sua tabela do Amazon DocumentDB. Por exemplo, não é permitido usar letras maiúsculas em nomes de tabelas do AWS Glue, mas é permitido usá-las em nomes de tabelas do Amazon DocumentDB.

3. (Opcional) Adicione a propriedade de tabela columnMapping. Essa propriedade define mapeamentos de nomes de colunas. Use essa propriedade caso as regras de nomenclatura de colunas do AWS Glue impeçam que você crie uma tabela do AWS Glue com os mesmos nomes de coluna da tabela do Amazon DocumentDB. Ela pode ser útil porque letras maiúsculas são permitidas nos nomes de colunas do Amazon DocumentDB, mas não são permitidas nos nomes de colunas do AWS Glue.

   Espera-se que o valor da propriedade `columnMapping` seja um conjunto de mapeamentos no formato `col1=Col1,col2=Col2`.

> **Note**
>
> O mapeamento de colunas só é aplicável a nomes de colunas de nível superior e não a campos aninhados.

Após adicionar a propriedade de tabela `columnMapping` do AWS Glue, é possível remover a variável de ambiente `disable_projection_and_casing` do Lambda.

4. Use os tipos de dados apropriados para o AWS Glue, conforme listado neste documento.

Suporte ao tipo de dados

Esta seção lista os tipos de dados que o conector DocumentDB usa para inferência de esquema e os tipos de dados quando metadados do AWS Glue forem usados.

Tipos de dados de inferência de esquema

O recurso de inferência de esquema do conector DocumentDB tenta inferir valores como pertencentes a um dos seguintes tipos de dados. A tabela mostra os tipos de dados correspondentes para o Amazon DocumentDB, Java e Apache Arrow.

Apache Arrow	Java ou DocDB
VARCHAR	String
INT	Inteiro
BIGINT	Longo
BIT	Booleano
FLOAT4	Float
FLOAT8	Double
TIMESTAMPSEC	Data
VARCHAR	ObjectId

Apache Arrow	Java ou DocDB
LIST	Lista
STRUCT	Documento

Tipos de dados do AWS Glue

Se você usar o AWS Glue para metadados complementares, será possível poderá configurar os tipos de dados a seguir. A tabela mostra os correspondentes tipos de dados do AWS Glue e do Apache Arrow.

AWS Glue	Apache Arrow
int	INT
bigint	BIGINT
double	FLOAT8
float	FLOAT4
boolean	BIT
binary	VARBINARY
string	VARCHAR
Lista	LIST
struct	STRUCT

Permissões obrigatórias

Os detalhes completos sobre as políticas do IAM exigidas por esse conector podem ser encontrados na seção `Policies` do arquivo athena-docdb.yaml. A lista a seguir resume as permissões necessárias.

- Acesso de gravação do Amazon S3: o conector requer acesso de gravação a um local no Amazon S3 para mostrar resultados de grandes consultas.
- Athena GetQueryExecution: o conector usa esta permissão para falhar rapidamente quando a consulta upstream do Athena é encerrada.
- AWS Glue Data Catalog: o conector DocumentDB requer acesso somente de leitura ao AWS Glue Data Catalog para obter informações do esquema.
- CloudWatch Logs: o conector requer acesso ao CloudWatch Logs para armazenar registros.
- Acesso de leitura do AWS Secrets Manager: se você optar por armazenar os detalhes do endpoint do DocumentDB no Secrets Manager, deverá conceder ao conector acesso a esses segredos.
- Acesso à VPC: o conector exige a capacidade de conectar e desconectar interfaces à sua VPC para que ela possa se conectar a ela e se comunicar com suas instâncias do DocumentDB.

Performance

No momento, o conector Amazon DocumentDB do Athena não é compatível com verificações paralelas, mas tenta empilhar predicados como parte de suas consultas ao DocumentDB, e predicados em relação a índices na coleção do DocumentDB resultam em, significantemente, menos dados verificados.

A função do Lambda executa o empilhamento de projeções para diminuir os dados verificados pela consulta. No entanto, selecionar um subconjunto de colunas, às vezes, resulta em um runtime de consulta mais longo. As cláusulas `LIMIT` reduzem a quantidade de dados verificados, mas se você não fornecer um predicado, deverá aguardar que as consultas `SELECT` com uma cláusula `LIMIT` verifiquem, no mínimo, 16 MB de dados.

Consultas de passagem

O conector do Amazon DocumentDB para Athena é compatível com consultas de passagem e é baseado em NoSQL. Para obter mais informações sobre como fazer consultas ao Amazon DocumentDB, consulte Como fazer consultas no Guia do desenvolvedor do Amazon DocumentDB.

Para usar consultas de passagem com o Amazon DocumentDB, use a seguinte sintaxe:

```
SELECT * FROM TABLE(
        system.query(
            database => 'database_name',
            collection => 'collection_name',
            filter => '{query_syntax}'
```

```
        ))
```

O exemplo a seguir consulta o banco de dados `example` da coleção `TPCDS`, filtrando todos os livros com o título Bill of Rights.

```
SELECT * FROM TABLE(
        system.query(
            database => 'example',
            collection => 'tpcds',
            filter => '{title: "Bill of Rights"}'
        ))
```

Recursos adicionais do

- Para um artigo sobre o uso da Consulta federada do Amazon Athena para conectar um banco de dados MongoDB ao Amazon QuickSight para criar painéis e visualizações, consulte Visualize dados do MongoDB do Amazon QuickSight usando a consulta federada Amazon Athena no AWSBlog de Big Data.
- Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o DynamoDB

O conector do DynamoDB no Amazon Athena permite que o Amazon Athena se comunique com o DynamoDB para que você possa consultar suas tabelas com SQL. Operações de gravação como INSERT INTO não são suportadas.

Se você tiver o Lake Formation habilitado em sua conta, o perfil do IAM para seu conector Lambda federado para Athena que você implantou no AWS Serverless Application Repository deve ter acesso de leitura ao AWS Glue Data Catalog no Lake Formation.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector DynamoDB.

- spill_bucket: especifica o bucket do Amazon S3 para dados que excedem os limites da função do Lambda.
- spill_prefix: (opcional) assume como padrão uma subpasta no `spill_bucket` especificado chamado `athena-federation-spill`. Recomendamos que você configure um ciclo de vida de armazenamento do Amazon S3 neste local para excluir derramamentos anteriores a um número predeterminado de dias ou horas.
- spill_put_request_headers: (opcional) um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.
- kms_key_id: (opcional) por padrão, todos os dados transmitidos para o Amazon S3 são criptografados usando o modo de criptografia autenticado AES-GCM e uma chave gerada aleatoriamente. Para que sua função do Lambda use chaves de criptografia mais fortes geradas pelo KMS, como `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, é possível especificar um ID de chave do KMS.
- disable_spill_encryption: (opcional) quando definido como `True`, desativa a criptografia do derramamento. É padronizado como `False`, para que os dados transmitidos para o S3 sejam criptografados usando o AES-GCM — usando uma chave gerada aleatoriamente ou o KMS para gerar chaves. Desativar a criptografia do derramamento pode melhorar a performance, especialmente se o local do derramamento usar criptografia no lado do servidor.
- disable_glue: (opcional) se estiver presente e definido como verdadeiro, o conector não tentará recuperar metadados complementares do AWS Glue.
- glue_catalog: (opcional) use essa opção para especificar um catálogo do AWS Glue entre contas. Por padrão, o conector tenta obter metadados de sua própria conta do AWS Glue.
- disable_projection_and_casing: (opcional) desativa a projeção e a capitalização. Use se quiser consultar tabelas do DynamoDB que tenham maiúsculas e minúsculas em seus nomes de coluna e não quiser especificar uma propriedade `columnMapping` em sua tabela AWS Glue.

  O parâmetro `disable_projection_and_casing` usa os seguintes valores para especificar o comportamento do mapeamento de capitalização e coluna:

- auto: desativa a projeção e a capitalização quando um tipo anteriormente sem suporte é detectado e o mapeamento do nome da coluna não está definido na tabela. Essa é a configuração padrão.
- always (sempre): desativa a projeção e a capitalização incondicionalmente. Isso é útil quando você tem maiúsculas e minúsculas nos nomes das colunas do DynamoDB, mas não quer especificar nenhum mapeamento de nome de coluna.

Ao usar o parâmetro `disable_projection_and_casing`, lembre-se dos seguintes pontos:

- O uso do parâmetro pode resultar em um maior uso de largura de banda. Além disso, se a função do Lambda não estiver na mesma Região da AWS que a fonte de dados, custos mais altos de transferência padrão entre regiões da AWS serão gerados como resultado do maior uso de largura de banda. Para obter mais informações sobre os custos de transferência entre regiões, consulte Cobranças de transferência de dados da AWS para arquiteturas com servidor e com tecnologia sem servidor no blog da rede de parceiros da AWS.
- Como um número maior de bytes é transferido, e o maior número de bytes exige um maior tempo de desserialização, a latência geral pode aumentar.

Configuração de bancos de dados e tabelas no AWS Glue

Como a capacidade incorporada de inferência de esquema do conector é limitada, talvez você queira usar o AWS Glue para metadados. Para fazer isso, você deve ter um banco de dados e uma tabela no AWS Glue. Para habilitá-los para uso com o DynamoDB, você deve editar suas propriedades.

Para editar as propriedades do banco de dados no console do AWS Glue

1. Faça login no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. Escolha a guia Databases (Bancos de dados).

   Na página Databases (Bancos de dados), você pode editar um banco de dados existente ou escolher Add database (Adicionar banco de dados) para criar um.
3. Na lista de bancos de dados, selecione o link do banco de dados que deseja editar.
4. Selecione a opção Editar.
5. Na página Update a database (Atualizar um banco de dados), em Location (Local), adicione a string **`dynamo-db-flag`**. Essa palavra-chave indica que o banco de dados contém tabelas que o conector DynamoDB para Athena está usando para metadados complementares e

é necessária para bancos de dados do AWS Glue diferentes de `default`. A propriedade `dynamo-db-flag` é útil para filtrar bancos de dados em contas com muitos bancos de dados.

Para editar as propriedades da tabela no console do AWS Glue

1. Faça login no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. Escolha a guia Tabelas.

   Na guia Tabelas, edite uma tabela existente. Para obter informações sobre como adicionar tabelas manualmente ou com um crawler, consulte Trabalhar com tabelas no console do AWS Glue no Guia do desenvolvedor do AWS Glue.
3. Na lista de tabelas, escolha o link para a tabela que você deseja editar.
4. Selecione Actions (Ações), Edit (Editar).
5. Na página Edit table (Editar tabela), na seção Table properties (Propriedades de tabela), adicione as seguintes propriedades de tabela, conforme solicitado: Se você usar o crawler do DynamoDB do AWS Glue, essas propriedades serão definidas automaticamente.

   - DynamoDB: string que indica ao conector DynamoDB do Athena que a tabela pode ser usada para metadados complementares. Insira a string `dynamodb` nas propriedades da tabela em um campo chamado classification (classificação) (correspondência exata).

     > **Note**
     >
     > A página Definir propriedades da tabela, que é parte do processo de criação de tabelas no console no AWS Glue, tem uma seção Formato dos dados com um campo Classificação. Você não pode inserir nem escolher o `dynamodb` aqui. Em vez disso, depois de criar a tabela, siga as etapas para editar a tabela e inserir `classification` e `dynamodb` como um par de chave/valor na seção Propriedades da tabela.

   - sourceTable: propriedade de tabela opcional que define o nome da tabela de origem no DynamoDB. Use-a caso as regras de nomenclatura de tabelas do AWS Glue impeçam que você crie uma tabela do AWS Glue com o mesmo nome da sua tabela do DynamoDB. Por exemplo, letras maiúsculas não são permitidas em nomes de tabelas do AWS Glue, mas são permitidas nos nomes de tabelas do DynamoDB.

- columnMapping: propriedade de tabela opcional que define mapeamentos de nomes de colunas. Use-a caso as regras de nomenclatura de colunas do AWS Glue impeçam que você crie uma tabela do AWS Glue com os mesmos nomes de coluna da sua tabela do DynamoDB. Por exemplo, letras maiúsculas não são permitidas em nomes de colunas do AWS Glue, mas são permitidas nos nomes de colunas do DynamoDB. Espera-se que o valor da propriedade esteja no formato col1=Col1,col2=Col2. Observe que o mapeamento de colunas só é aplicável a nomes de colunas de nível superior e não a campos aninhados.
- defaultTimeZone: propriedade de tabela opcional que é aplicada a valores `date` ou `datetime` que não tenham um fuso horário explícito. Definir esse valor é uma boa prática para evitar discrepâncias entre o fuso horário padrão da fonte de dados e o fuso horário da sessão do Athena.
- datetimeFormatMapping: propriedade de tabela opcional que especifica o formato de `date` ou `datetime` a ser usado ao analisar dados de uma coluna do AWS Glue de tipo de dado `date` ou `timestamp`. Se essa propriedade não for especificada, o conector tentará inferir um formato ISO-8601. Se o conector não puder inferir o formato `date` ou `datetime`, ou analisar a string bruta, então o valor será omitido do resultado.

  O valor de `datetimeFormatMapping` deve estar no formato `col1=someformat1,col2=someformat2`. Veja a seguir alguns exemplos de formatos:

  ```
  yyyyMMdd'T'HHmmss
  ddMMyyyy'T'HH:mm:ss
  ```

  Se sua coluna tiver valores `date` ou `datetime` sem um fuso horário, e você desejar usar a coluna na cláusula `WHERE`, defina a propriedade `datetimeFormatMapping` para a coluna.

6. Se você definir suas colunas manualmente, certifique-se de usar os tipos de dados apropriados. Se você usou um crawler, valide as colunas e os tipos que o crawler descobriu.

Permissões obrigatórias

Revise a seção `Policies` do arquivo athena-dynamodb.yaml para obter detalhes completos sobre as políticas do IAM que esse conector exige. A lista a seguir resume as permissões necessárias.

- Acesso de gravação do Amazon S3: o conector requer acesso de gravação a um local no Amazon S3 para mostrar resultados de grandes consultas.

- Athena GetQueryExecution: o conector usa esta permissão para falhar rapidamente quando a consulta upstream do Athena é encerrada.
- AWS Glue Data Catalog: o conector DynamoDB requer acesso somente de leitura ao AWS Glue Data Catalog para obter informações do esquema.
- CloudWatch Logs: o conector requer acesso ao CloudWatch Logs para armazenar registros.
- Acesso de leitura do DynamoDB: o conector usa as operações da API `DescribeTable`, `ListSchemas`, `ListTables`, `Query` e `Scan`.

Performance

O conector Athena DynamoDB oferece suporte a verificações paralelas e tentativas de reduzir predicados como parte de suas consultas do DynamoDB. Um predicado de chave de hash com `X` valores distintos resulta em `X` chamadas de consulta para o DynamoDB. Todos os outros cenários de predicados resultam em um número `Y` de chamadas de exame, onde `Y` é determinado heuristicamente com base no tamanho da tabela e na throughput provisionada. Porém, selecionar um subconjunto de colunas resulta, algumas vezes, em um runtime mais longo.

Cláusulas `LIMIT` e predicados simples são passados diretamente e podem reduzir a quantidade de dados examinados e reduzir o runtime de execução da consulta.

Cláusulas LIMIT

Uma instrução `LIMIT N` reduz os dados examinados pela consulta. Com a passagem direta de `LIMIT N`, o conector só retorna `N` linhas para o Athena.

Predicados

Um predicado é uma expressão na cláusula `WHERE` de uma consulta SQL, que avalia para um valor booleano e filtra as linhas com base em várias condições. Para melhorar a funcionalidade e reduzir a quantidade de dados examinados, o conector do Athena para o DynamoDB pode combinar essas expressões e passá-las diretamente ao DynamoDB.

Os seguintes operadores do conector do Athena para o DynamoDB são compatíveis com a passagem direta de predicados:

- Booleano: E
- Igualdade: EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_NULL

Exemplo de passagem direta combinada

Para ter recursos aprimorados de consulta, combine os tipos de passagem direta, como no seguinte exemplo:

```
SELECT *
FROM my_table
WHERE col_a > 10 and col_b < 10
LIMIT 10
```

Para ver um artigo sobre como usar o pushdown de predicado para melhorar o desempenho em consultas federadas, incluindo DynamoDB, consulte Melhorar as consultas federadas com o pushdown de predicados no Amazon Athena no AWSBlog de Big Data.

Consultas de passagem

O conector do DynamoDB é compatível com consultas de passagem e usa a sintaxe partiQL. Não há compatibilidade com a operação de API GetItem do DynamoDB. Para obter informações sobre como consultar o DynamoDB usando o PartiQL, consulte Instruções selecionadas do partiQL para o DynamoDB no Guias do desenvolvedor do Amazon DynamoDB.

Para usar consultas de passagem com o DynamoDB, use a seguinte sintaxe:

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

O seguinte exemplo de consulta de passagem do DynamoDB usa partiQL para retornar uma lista de dispositivos Fire TV Stick com uma propriedade `DateWatched` posterior a 24/12/22.

```
SELECT * FROM TABLE(
        system.query(
           query => 'SELECT Devices
                       FROM WatchList
                       WHERE Devices.FireStick.DateWatched[0] > '12/24/22''
        ))
```

Solução de problemas

Vários filtros em uma coluna de chave de classificação

Mensagem de erro: KeyConditionExpressions deve conter apenas uma condição por chave

Causa: este problema pode ocorrer no mecanismo Athena versão 3 em consultas que tenham um filtro limitado inferior e superior em uma coluna de chave de classificação do DynamoDB. Como o DynamoDB não oferece suporte a mais de uma condição de filtro em uma chave de classificação, ocorre um erro quando o conector tenta propagar uma consulta que tenha as duas condições aplicadas.

Solução: atualize o conector para a versão 2023.11.1 ou posterior. Para obter instruções sobre como atualizar um conector, consulte Atualizar um conector de fonte de dados.

Custos

Os custos de uso do conector dependem dos recursos subjacentes da AWS que são usados. Como as consultas que usam verificações podem consumir um grande número de unidades de capacidade de leitura (RCUs), considere as informações para Definição de preços do Amazon DynamoDB com cuidado.

Recursos adicionais do

- Para uma introdução ao uso do conector Amazon Athena DynamoDB, consulte Acesse, consulte e una tabelas Amazon DynamoDB usando o Athena no guia AWSPadrões de Orientação Prescritiva.
- Para ver um artigo sobre o uso do conector Amazon Athena DynamoDB com Amazon DynamoDB, Athena e Amazon QuickSight para criar um painel de governança simples, consulte Consulte tabelas do Amazon DynamoDB entre contas usando a consulta federada Amazon Athena no AWSBlog Big Data.
- Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o Google BigQuery

O conector do Amazon Athena para o Google BigQuery permite que o Amazon Athena execute consultas SQL em seus dados do Google BigQuery.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de

dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Limitações

- As funções do Lambda têm um valor máximo de tempo limite de 15 minutos. Cada divisão executa uma consulta no BigQuery e precisa terminar com tempo suficiente para armazenar os resultados para que o Athena leia. Se a função do Lambda atingir o tempo limite, a consulta falhará.
- O Google BigQuery diferencia maiúsculas e minúsculas. O conector tenta corrigir a capitalização dos nomes dos conjuntos de dados e dos nomes das tabelas, mas não faz nenhuma correção de capitalização nos ID de projetos. Isso é necessário porque o Athena coloca em minúsculas todos os metadados. Essas correções geram muitas chamadas extras para o Google BigQuery.
- Não há suporte para o tipo de dado binário.
- Devido à simultaneidade e aos limites de cota do Google BigQuery, o conector pode encontrar problemas de limite de cota do Google. Para evitar esses problemas, imponha o máximo possível de restrições ao Google BigQuery. Para obter informações sobre cotas do BigQuery, consulte Cotas e limites na documentação do Google BigQuery.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector Google BigQuery.

- spill_bucket: especifica o bucket do Amazon S3 para dados que excedem os limites da função do Lambda.
- spill_prefix: (opcional) assume como padrão uma subpasta no `spill_bucket` especificado chamado `athena-federation-spill`. Recomendamos que você configure um ciclo de vida de armazenamento do Amazon S3 neste local para excluir derramamentos anteriores a um número predeterminado de dias ou horas.
- spill_put_request_headers: (opcional) um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.
- kms_key_id: (opcional) por padrão, todos os dados transmitidos para o Amazon S3 são criptografados usando o modo de criptografia autenticado AES-GCM e uma chave gerada aleatoriamente. Para que sua função do Lambda use chaves de criptografia mais fortes geradas

pelo KMS, como `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, é possível especificar um ID de chave do KMS.

- disable_spill_encryption: (opcional) quando definido como `True`, desativa a criptografia do derramamento. É padronizado como `False`, para que os dados transmitidos para o S3 sejam criptografados usando o AES-GCM — usando uma chave gerada aleatoriamente ou o KMS para gerar chaves. Desativar a criptografia do derramamento pode melhorar o desempenho, especialmente se o local do derramamento usar criptografia no lado do servidor.
- gcp_project_id: o ID do projeto (não o nome do projeto) que contém os conjuntos de dados que o conector deve ler (por exemplo, `semiotic-primer-1234567`).
- secret_manager_gcp_creds_name: o nome do segredo no AWS Secrets Manager que contém suas credenciais do BigQuery no formato JSON (por exemplo, `GoogleCloudPlatformCredentials`).
- big_query_endpoint: (opcional) o URL de um endpoint privado do BigQuery. Use esse parâmetro quando quiser acessar o BigQuery por meio de um endpoint privado.

Divisões e exibições

Como o conector BigQuery usa a API Storage Read do BigQuery para consultar tabelas, e a API Storage do BigQuery não é compatível com exibições, o conector usa o cliente do BigQuery com uma única divisão para exibições.

Performance

Para consultar tabelas, o conector BigQuery usa a API Storage Read do BigQuery, que usa um protocolo baseado em RPC que fornece acesso rápido ao armazenamento gerenciado do BigQuery. Para obter mais informações sobre a API BigQuery Storage Read, consulte Use the BigQuery Storage Read API to read table data na documentação do Google Cloud.

A seleção de um subconjunto de colunas acelera o runtime da consulta e reduz os dados verificados de forma significativa. O conector está sujeito a falhas de consulta à medida que a simultaneidade aumenta e, de forma geral, é um conector lento.

O conector do Athena para o Google BigQuery executa a passagem direta de predicados para diminuir os dados examinados pela consulta. Cláusulas `LIMIT`, cláusulas `ORDER BY`, predicados simples e expressões complexas são passados diretamente ao conector para reduzir a quantidade de dados examinados e o runtime de execução da consulta.

Cláusulas LIMIT

Uma instrução `LIMIT N` reduz os dados examinados pela consulta. Com a passagem direta de `LIMIT N`, o conector só retorna `N` linhas para o Athena.

Principais N consultas

Uma das `N` principais consultas especifica uma ordenação do conjunto de resultados e um limite no número de linhas retornadas. Você pode usar esse tipo de consulta para determinar os `N` principais valores máximos ou `N` principais valores mínimos para os conjuntos de dados. Com os `N` pushdown principais, o conector só retorna `N` linhas para o Athena.

Predicados

Um predicado é uma expressão na cláusula `WHERE` de uma consulta SQL, que avalia para um valor booleano e filtra as linhas com base em várias condições. O conector do Athena para o Google BigQuery pode combinar essas expressões e passá-las diretamente ao Google BigQuery para melhorar a funcionalidade e reduzir a quantidade de dados examinados.

Os seguintes operadores do conector do Athena para o Google BigQuery são compatíveis com a passagem direta de predicados:

- Booleanos: E, OU, NÃO
- Igualdade: EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Aritméticos: ADICIONAR, SUBTRAIR, MULTIPLICAR, DIVIDIR, MÓDULO, NEGAR
- Outros:LIKE_PATTERN, IN

Exemplo de passagem direta combinada

Para ter recursos aprimorados de consulta, combine os tipos de passagem direta, como no seguinte exemplo:

```
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;
```

Consultas de passagem

O conector Google BigQuery é compatível com consultas de passagem. As consultas de passagem usam uma função de tabela para enviar sua consulta completa para execução na fonte de dados.

Para usar consultas de passagem com o Google BigQuery, você pode empregar a seguinte sintaxe:

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

O exemplo de consulta a seguir envia uma consulta para uma fonte de dados no Google BigQuery. A consulta seleciona todas as colunas na tabela `customer`, limitando os resultados a 10.

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

Informações de licença

O projeto do conector Google BigQuery do Amazon Athena é licenciado sob a Licença Apache-2.0.

Ao usar esse conector, você reconhece a inclusão de componentes de terceiros, cuja lista pode ser encontrada no arquivo pom.xml desse conector, e concorda com os termos das respectivas licenças de terceiros fornecidas no arquivo LICENSE.txt em GitHub.com.

Recursos adicionais do

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector Google Cloud Storage para Amazon Athena

O conector Google Cloud Storage para Amazon Athena permite que o Amazon Athena execute consultas em arquivos Parquet e CSV armazenados em um bucket do Google Cloud Storage (GCS). Depois de agrupar um ou mais arquivos Parquet ou CSV em uma pasta não particionada ou particionada em um bucket do GCS, você poderá organizá-los em uma tabela de banco de dados do AWS Glue.

Se você tiver o Lake Formation habilitado em sua conta, o perfil do IAM para seu conector Lambda federado para Athena que você implantou no AWS Serverless Application Repository deve ter acesso de leitura ao AWS Glue Data Catalog no Lake Formation.

Pré-requisitos

- Configure um banco de dados e uma tabela do AWS Glue que correspondam ao seu bucket e suas pastas no Google Cloud Storage. Para ver as etapas, consulte Configuração de bancos de dados e tabelas no AWS Glue mais adiante neste documento.
- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Limitações

- Não há suporte para operações de gravação de DDL.
- Quaisquer limites relevantes do Lambda. Para obter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do AWS Lambda.
- Atualmente, o conector é compatível somente com o tipo `VARCHAR` de colunas de partição (`string` ou `varchar` em um esquema de tabela do AWS Glue). Outros tipos de campo de partição geram erros quando você os consulta no Athena.

Termos

Os termos a seguir estão relacionados ao conector GCS.

- Manipulador: um manipulador Lambda que acessa seu bucket do GCS. Um manipulador pode ser para metadados ou para registros de dados.
- Manipulador de metadados: um manipulador Lambda que recupera metadados do seu bucket do GCS.
- Manipulador de registros: um manipulador Lambda que recupera registros de dados do seu bucket do GCS.
- Manipulador composto: um manipulador Lambda que recupera tanto metadados quanto registros de dados do seu bucket do GCS.

Tipos de arquivos compatíveis

O conector GCS é compatível com os tipos de arquivo Parquet e CSV.

> **Note**
>
> Certifique-se de não colocar os arquivos CSV e Parquet no mesmo bucket ou caminho do GCS. Isso pode resultar em um erro de execução se houver tentativa de ler os arquivos Parquet como CSV ou vice-versa.

Parâmetros

Use as variáveis de ambiente do Lambda apresentadas nesta seção para configurar o conector GCS.

- spill_bucket: especifica o bucket do Amazon S3 para dados que excedem os limites da função do Lambda.
- spill_prefix: (opcional) assume como padrão uma subpasta no `spill_bucket` especificado chamado `athena-federation-spill`. Recomendamos que você configure um ciclo de vida de armazenamento do Amazon S3 neste local para excluir derramamentos anteriores a um número predeterminado de dias ou horas.
- spill_put_request_headers: (opcional) um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.
- kms_key_id: (opcional) por padrão, todos os dados transmitidos para o Amazon S3 são criptografados usando o modo de criptografia autenticado AES-GCM e uma chave gerada aleatoriamente. Para que sua função do Lambda use chaves de criptografia mais fortes geradas pelo KMS, como `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, é possível especificar um ID de chave do KMS.
- disable_spill_encryption: (opcional) quando definido como `True`, desativa a criptografia do derramamento. É padronizado como `False`, para que os dados transmitidos para o S3 sejam criptografados usando o AES-GCM — usando uma chave gerada aleatoriamente ou o KMS para gerar chaves. Desativar a criptografia do derramamento pode melhorar a performance, especialmente se o local do derramamento usar criptografia no lado do servidor.

- secret_manager_gcp_creds_name: o nome do segredo no AWS Secrets Manager que contém suas credenciais do GCS no formato JSON (por exemplo, `GoogleCloudPlatformCredentials`).

Configuração de bancos de dados e tabelas no AWS Glue

Como a capacidade de inferência de esquema integrada do conector GCS é limitada, recomendamos que você use o AWS Glue para seus metadados. Os procedimentos a seguir mostram como criar um banco de dados e uma tabela do AWS Glue que você possa acessar a partir do Athena.

Criação de um banco de dados no AWS Glue

Você pode usar o console do AWS Glue para criar um banco de dados para uso com o conector GCS.

Para criar um banco de dados no AWS Glue

1. Faça login no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. Escolha Databases (Bancos de dados) no painel de navegação.
3. Selecione Add database (Adicionar banco de dados).
4. Em Name (Nome), insira um nome para o banco de dados que você deseja usar com o conector GCS.
5. Em Location (Local), especifique `s3://google-cloud-storage-flag.` Esse local informa ao conector GCS que o banco de dados do AWS Glue contém tabelas para dados do GCS a serem consultados no Athena. O conector reconhece bancos de dados no Athena que têm esse sinalizador e ignora bancos de dados que não o têm.
6. Selecione Criar banco de dados.

Criar uma tabela no AWS Glue

Agora você pode criar uma tabela para o banco de dados. Ao criar uma tabela do AWS Glue para usar com o conector GCS, você deve especificar metadados adicionais.

Para criar uma tabela no console do AWS Glue

1. No painel de navegação do console do AWS Glue, escolha Tables (Tabelas).

2. Na página Tables (Tabelas), escolha Add table (Adicionar tabela).
3. Na página Set table properties (Definir propriedades da tabela), insira as informações a seguir.

   - Name (Tabela): um nome exclusivo para a tabela.
   - Banco de dados: escolha o banco de dados do AWS Glue que você criou para o conector GCS.
   - Include path (Incluir caminho): Na seção Data store (Armazenamento de dados), em Include path (Incluir caminho), insira a localização do URI para GCS prefixada por `gs://` (por exemplo, `gs://gcs_table/data/`). Se você tiver uma ou mais pastas de partição, não as inclua no caminho.

     > **Note**
     >
     > Quando você insere um caminho de table que não é do `s3://` tabela, o console do AWS Glue mostra um erro. Você pode ignorar esse erro. A tabela será criada com êxito.

   - Formato de dados: para Classification (Classificação), selecione CSV ou Parquet.
4. Escolha Next (próximo).
5. Na página Choose or define schema (Escolher ou definir esquema), definir um esquema de tabela é altamente recomendado, mas não obrigatório. Se você não definir um esquema, o conector GCS tentará inferir um esquema para você.

   Execute um destes procedimentos:

   - Se você quiser que o conector GCS tente inferir um esquema para você, escolha Next (Próximo) e, em seguida, escolha Create (Criar).
   - Para definir um esquema você mesmo, siga as etapas da próxima seção.

Definir um esquema de tabela no AWS Glue

Definir um esquema de tabela no AWS Glue requer mais etapas, mas oferece maior controle sobre o processo de criação da tabela.

Para definir um esquema para sua tabela no AWS Glue

1. Na página Choose or define schema (Escolher ou definir esquema), escolha Add (Adicionar).

2. Use a caixa de diálogo Add schema entry (Adicionar entrada de esquema) para fornecer um nome de coluna e um tipo de dados.
3. Para designar a coluna como uma coluna de partição, selecione a opção Set as partition key (Definir como chave de partição).
4. Escolha Save (Salvar) para salvar a alteração.
5. Selecione Add (Adicionar) para adicionar outra coluna.
6. Ao terminar de adicionar contas, escolha Next (Próximo).
7. Na página Review and create (Revisar e criar), revise a configuração e selecione Create (Criar).
8. Se o esquema contiver informações sobre partições, siga as etapas na próxima seção para adicionar um padrão de partição às propriedades da tabela em AWS Glue.

Adicionar um padrão de partição às propriedades da tabela no AWS Glue

Se seus buckets do GCS tiverem partições, você deverá adicionar o padrão de partição às propriedades da tabela no AWS Glue.

Para adicionar informações de partição às propriedades da tabela do AWS Glue

1. Na página de detalhes da tabela que você criou em AWS Glue, escolha Actions (Ações), Edit table (Editar tabela).
2. Na página Edit table (Editar tabela), role para baixo até a seção Table properties (Propriedades da tabela).
3. Escolha Add (Adicionar) para adicionar uma chave de partição.
4. Em Chave, digite **`partition.pattern`**. Essa chave define o padrão do caminho da pasta.
5. Em Value (Valor), insira um padrão de caminho de pasta como **`StateName=${statename}/ZipCode=${zipcode}/`**, em que **`statename`** e **`zipcode`** delimitados por **`${}`** são nomes de colunas de partição. O conector GCS é compatível com esquemas de partição Hive e não Hive.
6. Quando terminar, escolha Salvar.
7. Para visualizar as propriedades da tabela que você acabou de criar, escolha a guia Advanced properties (Propriedades avançadas).

Nesse ponto, você pode navegar até o console do Athena. O banco de dados e a tabela que você criou no AWS Glue estão disponíveis para consulta no Athena.

Suporte ao tipo de dados

As tabelas a seguir mostram os tipos de dados com suporte para CSV e Parquet.

CSV

Natureza dos dados	Tipo de dados inferido
Os dados parecem um número	BIGINT
Os dados parecem uma string	VARCHAR
Os dados parecem de ponto flutuante (flutuante, duplo ou decimal)	DOUBLE
Os dados parecem uma data	Timestamp
Dados que contêm valores verdadeiros/falsos	BOOL

Parquet

PARQUET	Athena (Seta)
BINARY	VARCHAR
BOOLEAN	BOOL
DOUBLE	DOUBLE
ENUM	VARCHAR
FIXED_LEN_BYTE_ARRAY	DECIMAL
FLOAT	FLOAT (32 bits)
INT32	1. INT32 2. DATEDAY (quando o tipo lógico da coluna Parquet é DATE)
INT64	1. INT64

PARQUET	Athena (Seta)
	2. TIMESTAMP (quando o tipo lógico da coluna Parquet é TIMESTAMP)
INT96	Timestamp
MAP	MAP
STRUCT	STRUCT
LIST	LIST

Permissões obrigatórias

Os detalhes completos sobre as políticas do IAM exigidas por esse conector podem ser encontrados na seção `Policies` do arquivo athena-gcs.yaml. A lista a seguir resume as permissões necessárias.

- Acesso de gravação do Amazon S3: o conector requer acesso de gravação a um local no Amazon S3 para mostrar resultados de grandes consultas.
- Athena GetQueryExecution: o conector usa esta permissão para falhar rapidamente quando a consulta upstream do Athena é encerrada.
- AWS Glue Data Catalog: o conector GCS requer acesso somente de leitura ao AWS Glue Data Catalog para obter informações do esquema.
- CloudWatch Logs: o conector requer acesso ao CloudWatch Logs para armazenar registros.

Performance

Quando o esquema da tabela contém campos de partição e a propriedade `partition.pattern` da tabela está configurada corretamente, você pode incluir o campo de partição na cláusula `WHERE` de suas consultas. Para essas consultas, o conector GCS usa as colunas de partição para refinar o caminho da pasta GCS e evitar a varredura de arquivos desnecessários nas pastas do GCS.

Para conjuntos de dados Parquet, selecionar um subconjunto de colunas resulta em menos dados sendo verificados. Isso geralmente resulta em um tempo de execução de consulta mais curto quando a projeção de coluna é aplicada.

Para conjuntos de dados CSV, a projeção de colunas não é compatível e não reduz a quantidade de dados que estão sendo digitalizados.

As cláusulas `LIMIT` reduzem a quantidade de dados verificados, mas se você não fornecer um predicado, deverá aguardar que as consultas `SELECT` com uma cláusula `LIMIT` verifiquem, no mínimo, 16 MB de dados. O conector GCS examina mais dados para conjuntos de dados maiores do que para conjuntos de dados menores, independentemente da cláusula `LIMIT` aplicada. Por exemplo, a consulta `SELECT * LIMIT 10000` examina mais dados em busca de um conjunto de dados subjacente maior do que de um menor.

Informações de licença

Ao usar esse conector, você reconhece a inclusão de componentes de terceiros, cuja lista pode ser encontrada no arquivo pom.xml desse conector, e concorda com os termos das respectivas licenças de terceiros fornecidas no arquivo LICENSE.txt em GitHub.com.

Recursos adicionais do

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o HBase

O conector do HBase no Amazon Athena permite que o Amazon Athena se comunique com as instâncias do Apache HBase para que você possa consultar os dados do HBase com SQL.

Diferentemente dos armazenamentos de dados relacionais tradicionais, as coleções do HBase não têm um esquema definido. O HBase não tem um armazenamento de metadados. Cada entrada em uma coleção do HBase pode ter diferentes campos e tipos de dados.

O conector HBase oferece suporte a dois mecanismos para gerar informações do esquema da tabela: inferência básica de esquema e metadados do AWS Glue Data Catalog.

A inferência de esquema é o padrão. Essa opção examina um pequeno número de documentos em sua coleção, forma uma união de todos os campos e força campos que têm tipos de dados não sobrepostos. Essa opção funciona bem para coleções que têm, em sua maioria, entradas uniformes.

Para coleções com uma maior variedade de tipos de dados, o conector oferece suporte à recuperação de metadados do AWS Glue Data Catalog. Se o conector vê um banco de dados do AWS Glue e uma tabela que correspondam aos nomes de seu namespace do HBase e nomes de coleção, ele obtém suas informações de esquema da tabela AWS Glue correspondente. Quando você cria sua tabela AWS Glue, recomendamos que você a torne um superconjunto de todos os campos que você talvez queira acessar da sua coleção do HBase.

Se você tiver o Lake Formation habilitado em sua conta, o perfil do IAM para seu conector Lambda federado para Athena que você implantou no AWS Serverless Application Repository deve ter acesso de leitura ao AWS Glue Data Catalog no Lake Formation.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector HBase.

- spill_bucket: especifica o bucket do Amazon S3 para dados que excedem os limites da função do Lambda.
- spill_prefix: (opcional) assume como padrão uma subpasta no `spill_bucket` especificado chamado `athena-federation-spill`. Recomendamos que você configure um ciclo de vida de armazenamento do Amazon S3 neste local para excluir derramamentos anteriores a um número predeterminado de dias ou horas.
- spill_put_request_headers: (opcional) um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.
- kms_key_id: (opcional) por padrão, todos os dados transmitidos para o Amazon S3 são criptografados usando o modo de criptografia autenticado AES-GCM e uma chave gerada aleatoriamente. Para que sua função do Lambda use chaves de criptografia mais fortes geradas pelo KMS, como `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, é possível especificar um ID de chave do KMS.
- disable_spill_encryption: (opcional) quando definido como `True`, desativa a criptografia do derramamento. É padronizado como `False`, para que os dados transmitidos para o S3 sejam criptografados usando o AES-GCM — usando uma chave gerada aleatoriamente ou o KMS para gerar chaves. Desativar a criptografia do derramamento pode melhorar a performance, especialmente se o local do derramamento usar criptografia no lado do servidor.
- disable_glue: (opcional) se estiver presente e definido como verdadeiro, o conector não tentará recuperar metadados complementares do AWS Glue.

- glue_catalog: (opcional) use essa opção para especificar um catálogo do AWS Glue entre contas. Por padrão, o conector tenta obter metadados de sua própria conta do AWS Glue.
- default_hbase: se estiver presente, especifica uma string de conexão do HBase a ser usada quando não existir uma variável de ambiente específica do catálogo.

Especificação de strings de conexão

É possível fornecer uma ou mais propriedades que definem os detalhes da conexão do HBase para as instâncias do HBase que você usar com o conector. Para fazer isso, defina uma variável de ambiente Lambda que corresponda ao nome do catálogo que você deseja usar no Athena. Por exemplo, suponha que você queira usar as seguintes consultas para consultar duas instâncias diferentes do HBase no Athena:

```
SELECT * FROM "hbase_instance_1".database.table
```

```
SELECT * FROM "hbase_instance_2".database.table
```

Antes de usar essas duas instruções de SQL, você deverá adicionar duas variáveis de ambiente à sua função do Lambda: `hbase_instance_1` e `hbase_instance_2`. O valor para cada uma deverá ser uma string de conexão do HBase no seguinte formato:

```
master_hostname:hbase_port:zookeeper_port
```

Uso de segredos

Opcionalmente, é possível usar o AWS Secrets Manager para obter parte ou todo o valor dos detalhes da string de conexão. Para usar o recurso Athena Federated Query com o Secrets Manager, a VPC conectada à sua função do Lambda deve ter acesso à Internet ou um endpoint da VPC para se conectar ao Secrets Manager.

Se você usar a sintaxe `${my_secret}` para colocar o nome de um segredo do Secrets Manager em sua string de conexão, o conector substituirá o nome do segredo pelos valores de seu nome de usuário e senha do Secrets Manager.

Por exemplo, suponha que você defina a variável de ambiente Lambda para `hbase_instance_1` com o seguinte valor:

```
${hbase_host_1}:${hbase_master_port_1}:${hbase_zookeeper_port_1}
```

O SDK do Athena Query Federation tenta automaticamente recuperar um segredo chamado `hbase_instance_1_creds` do Secrets Manager e injetar esse valor no lugar de `${hbase_instance_1_creds}`. Qualquer parte da string de conexão que esteja delimitada pela combinação de caracteres `${ }` será é interpretada como um segredo do Secrets Manager. Se você especificar um nome secreto que o conector não consiga encontrar no Secrets Manager, o conector não substituirá o texto.

Configuração de bancos de dados e tabelas no AWS Glue

A inferência de esquema incorporada do conector oferece suporte somente a valores serializados no HBase, como strings (por exemplo, `String.valueOf(int)`). Como a capacidade incorporada de inferência de esquema do conector é limitada, talvez você queira usar o AWS Glue para metadados, em vez disso. Para habilitar uma tabela do AWS Glue para uso com o HBase, é preciso ter um banco de dados do AWS Glue e uma tabela com nomes que correspondam ao namespace do HBase e à tabela para a qual você deseja fornecer metadados complementares. O uso das convenções de nomenclatura da família de colunas do HBase é opcional, mas não obrigatório.

Para usar uma tabela AWS Glue para metadados complementares

1. Quando você editar a tabela e o banco de dados no console do AWS Glue, adicione as seguintes propriedades de tabela:

   - hbase-metadata-flag: essa propriedade indica ao conector HBase que o conector pode usar a tabela para metadados complementares. É possível fornecer qualquer valor para `hbase-metadata-flag`, desde que a propriedade `hbase-metadata-flag` esteja presente na lista de propriedades da tabela.
   - hbase-native-storage-flag: use esse sinalizador para alternar os dois modos de serialização de valores com suporte pelo conector. Por padrão, quando esse campo não estiver presente, o conector assumirá que todos os valores estão armazenados no HBase como strings. Como tal, ele tentará analisar tipos de dados como `INT`, `BIGINT` e `DOUBLE` do HBase como strings. Se esse campo for definido com qualquer valor na tabela emAWS Glue, o conector muda para o modo de armazenamento “nativo” e tenta ler`INT`,`BIGINT`,`BIT`, e`DOUBLE`como bytes usando as seguintes funções:

     ```
     ByteBuffer.wrap(value).getInt()
     ByteBuffer.wrap(value).getLong()
     ByteBuffer.wrap(value).get()
     ByteBuffer.wrap(value).getDouble()
     ```

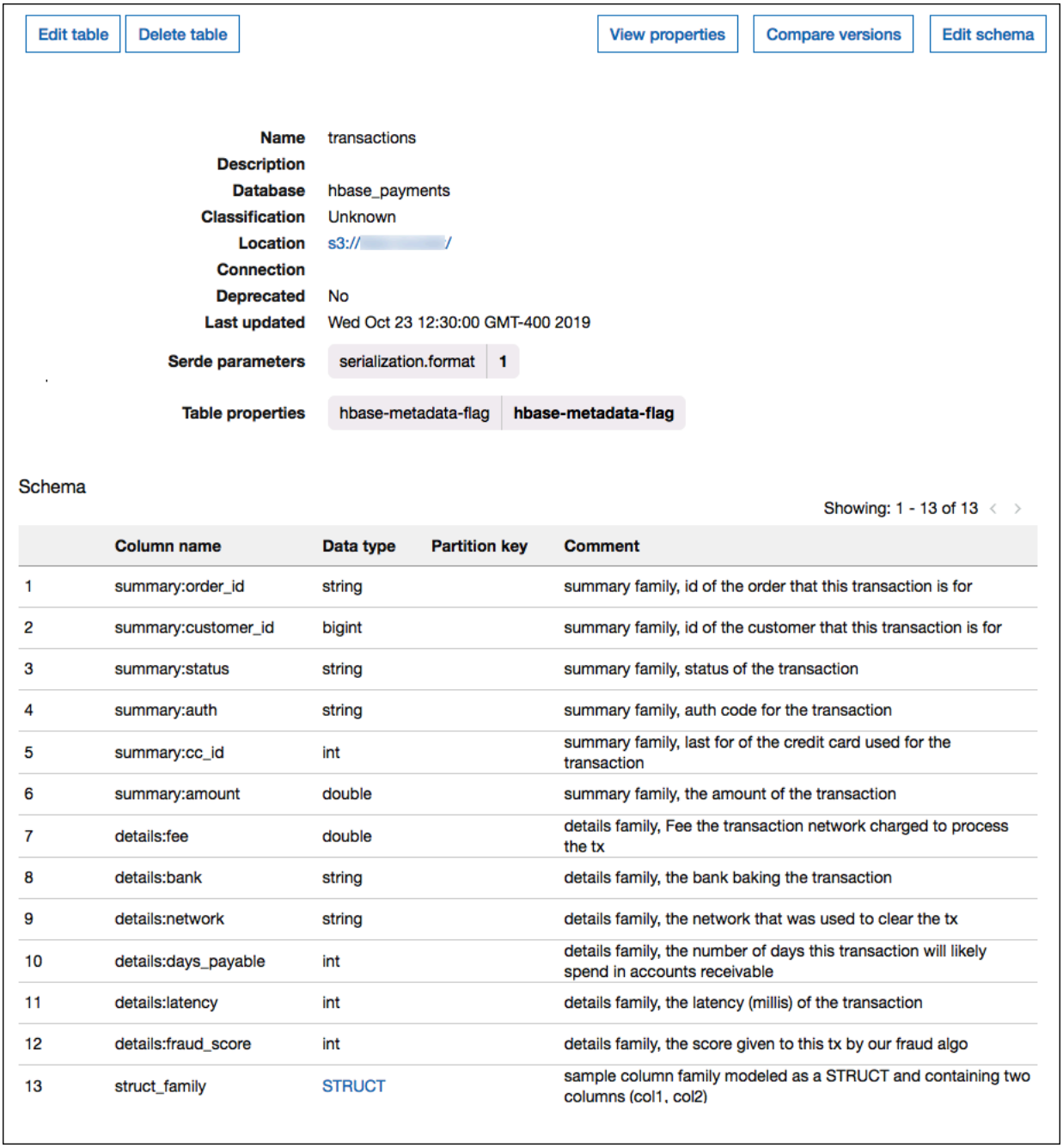
2. Use os tipos de dados apropriados para o AWS Glue, conforme listado neste documento.

Modelagem de famílias de colunas

O conector Athena HBase oferece suporte a duas formas de modelagem de famílias de colunas do HBase: nomenclatura totalmente qualificada (plana), como `family:column` ou usando objetos `STRUCT`.

No modelo `STRUCT`, o nome do campo `STRUCT` deve corresponder à família da coluna, e filhos do `STRUCT` devem corresponder aos nomes das colunas da família. No entanto, como ainda não há suporte total para a redução de predicados e as leituras colunares com tipos complexos como `STRUCT`, o uso de `STRUCT`, atualmente, não é recomendado.

A imagem a seguir mostra uma tabela configurada no AWS Glue que usa uma combinação das duas abordagens.



	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)

Suporte ao tipo de dados

O conector recupera todos os valores do HBase como o tipo básico byte. Em seguida, com base em como você definiu suas tabelas no Catálogo de dados do AWS Glue, ele mapeia os valores em um dos tipos de dados do Apache Arrow na tabela a seguir.

Tipo de dados do AWS Glue	Tipo de dados Apache Arrow
int	INT
bigint	BIGINT
double	FLOAT8
float	FLOAT4
boolean	BIT
binary	VARBINARY
string	VARCHAR

> **Note**
>
> Se você não usar AWS Glue para complementar seus metadados, a inferência do esquema do conector usará somente os tipos de dados `BIGINT`, `FLOAT8` e `VARCHAR`.

Permissões obrigatórias

Os detalhes completos sobre as políticas do IAM exigidas por esse conector podem ser encontrados na seção `Policies` do arquivo athena-hbase.yaml. A lista a seguir resume as permissões necessárias.

- Acesso de gravação do Amazon S3: o conector requer acesso de gravação a um local no Amazon S3 para mostrar resultados de grandes consultas.
- Athena GetQueryExecution: o conector usa esta permissão para falhar rapidamente quando a consulta upstream do Athena é encerrada.
- AWS Glue Data Catalog: o conector HBase requer acesso somente de leitura ao AWS Glue Data Catalog para obter informações do esquema.
- CloudWatch Logs: o conector requer acesso ao CloudWatch Logs para armazenar registros.
- Acesso de leitura do AWS Secrets Manager: se você optar por armazenar os detalhes do endpoint do HBase no Secrets Manager, deverá conceder ao conector acesso a esses segredos.

- Acesso à VPC: o conector exige a capacidade de conectar e desconectar interfaces à sua VPC para que ela possa se conectar a ela e se comunicar com suas instâncias do HBase.

Performance

O conector Athena HBase tenta paralelizar as consultas em sua instância do HBase lendo cada servidor da região em paralelo. O conetor do Athena para o HBase realiza a passagem direta de predicados para diminuir os dados examinados pela consulta.

A função do Lambda também executa o empilhamento de projeções para diminuir os dados verificados pela consulta. No entanto, selecionar um subconjunto de colunas, às vezes, resulta em um runtime de consulta mais longo. As cláusulas `LIMIT` reduzem a quantidade de dados verificados, mas se você não fornecer um predicado, deverá aguardar que as consultas `SELECT` com uma cláusula `LIMIT` verifiquem, no mínimo, 16 MB de dados.

O HBase está propenso a falhas na consulta e tempos de execução variáveis para as consultas. Pode ser necessário repetir as consultas diversas vezes para que elas sejam bem-sucedidas. O conector HBase é resiliente ao controle de utilização devido à simultaneidade.

Consultas de passagem

O conector do HBase é compatível com consultas de passagem e é baseado em NoSQL. Para obter informações sobre como consultar o Apache HBase, consulte Como consultar o HBase na documentação do Apache.

Para usar consultas de passagem com o HBase, use a seguinte sintaxe:

```
SELECT * FROM TABLE(
        system.query(
            database => 'database_name',
            collection => 'collection_name',
            filter => '{query_syntax}'
        ))
```

O exemplo a seguir de consulta de passagem do HBase filtra funcionários com 24 ou 30 anos na coleção `employee` do banco de dados `default`.

```
SELECT * FROM TABLE(
        system.query(
            DATABASE => 'default',
```

```
           COLLECTION => 'employee',
           FILTER => 'SingleColumnValueFilter(''personaldata'', ''age'', =,
''binary:30'')' ||
                     ' OR SingleColumnValueFilter(''personaldata'', ''age'', =,
''binary:24'')'
       ))
```

Informações de licença

O projeto do conector HBase do Amazon Athena é licenciado sob a Licença Apache-2.0.

Recursos adicionais do

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para a Hortonworks

O conector do Amazon Athena para a Hortonworks permite que o Amazon Athena execute consultas SQL na plataforma de dados Hortonworks da Cloudera. O conector transforma suas consultas SQL do Athena na sintaxe equivalente do HiveQL.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Limitações

- Não há suporte para operações de gravação de DDL.
- Em uma configuração de multiplexador, o prefixo e o bucket de derramamento são compartilhados em todas as instâncias do banco de dados.
- Quaisquer limites relevantes do Lambda. Para obter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do AWS Lambda.

Termos

Os termos a seguir estão relacionados ao conector Hortonworks Hive.

- Instância do banco de dados: qualquer instância de um banco de dados implantado on-premises, no Amazon EC2 ou no Amazon RDS.
- Manipulador: um manipulador Lambda que acessa sua instância de banco de dados. Um manipulador pode ser para metadados ou para registros de dados.
- Manipulador de metadados: um manipulador Lambda que recupera metadados da sua instância de banco de dados.
- Manipulador de registros: um manipulador Lambda que recupera registros de dados da sua instância de banco de dados.
- Manipulador composto: um manipulador Lambda que recupera tanto metadados quanto registros de dados da sua instância de banco de dados.
- Propriedade ou parâmetro: uma propriedade do banco de dados usada pelos manipuladores para extrair informações do banco de dados. Você configura essas propriedades como variáveis de ambiente do Lambda.
- String de conexão: uma string de texto usada para estabelecer uma conexão com uma instância de banco de dados.
- Catálogo: um catálogo não AWS Glue registrado no Athena que é um prefixo obrigatório para a propriedade `connection_string`.
- Manipulador de multiplexação: um manipulador Lambda que pode aceitar e usar várias conexões de banco de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector Hortonworks Hive.

String de conexão

Use uma string de conexão JDBC no seguinte formato para se conectar a uma instância de banco de dados.

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

Uso de um manipulador de multiplexação

É possível usar um multiplexador para se conectar a várias instâncias de banco de dados com uma única função do Lambda. As solicitações são encaminhadas por nome do catálogo. Use as seguintes classes no Lambda.

Manipulador	Classe
Manipulador composto	`HiveMuxCompositeHandler`
Manipulador de metadados	`HiveMuxMetadataHandler`
Manipulador de registros	`HiveMuxRecordHandler`

Parâmetros do manipulador de multiplexação

Parâmetro	Descrição
`$`*`catalog`*`_connection_string`	Obrigatório. Uma string de conexão de instância de banco de dados. Prefixe a variável de ambiente com o nome do catálogo usado no Athena. Por exemplo, se o catálogo registrado no Athena for `myhivecatalog` , então o nome da variável de ambiente será `myhivecatalog_connection_string` .
`default`	Obrigatório. A string de conexão padrão. Essa string é usada quando o catálogo for `lambda:${` *`AWS_LAMBDA_FUNCTION_NAME`* `}`.

As propriedades de exemplo a seguir são para uma função do Lambda MUX do Hive que ofereça suporte a duas instâncias de banco de dados: `hive1` (o padrão) e `hive2`.

Propriedade	Valor
`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`

Fornecimento de credenciais

Para fornecer um nome de usuário e uma senha para seu banco de dados na string de conexão JDBC, é possível usar as propriedades da string de conexão ou o AWS Secrets Manager.

- String de conexão: um nome de usuário e uma senha podem ser especificados como propriedades na string de conexão do JDBC.

  > **Important**
  >
  > Como prática recomendada de segurança, não use credenciais codificadas em suas variáveis de ambiente ou strings de conexão. Para obter informações sobre como mover seus segredos codificados para o AWS Secrets Manager, consulte Mover segredos codificados para o AWS Secrets Manager no Guia do usuário do AWS Secrets Manager.

- AWS Secrets Manager: para usar o recurso Athena Federated Query com o AWS Secrets Manager, a VPC conectada à sua função do Lambda deve ter acesso à Internet ou um endpoint da VPC para se conectar ao Secrets Manager.

  É possível colocar o nome de um segredo no AWS Secrets Manager na sua string de conexão JDBC. O conector substitui o nome secreto pelos valores de `username` e `password` do Secrets Manager.

  Para instâncias de banco de dados do Amazon RDS, esse suporte é totalmente integrado. Se você usa o Amazon RDS, é altamente recomendável usar o AWS Secrets Manager e rotação de credenciais. Se seu banco de dados não usar o Amazon RDS, armazene as credenciais em JSON no seguinte formato:

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

Exemplo de string de conexão com nome secreto

A string a seguir tem o nome secreto `${Test/RDS/hive1host}`.

```
hive://jdbc:hive2://hive1host:10000/default?...&${Test/RDS/hive1host}&...
```

O conector usa o nome secreto para recuperar segredos e fornecer o nome de usuário e a senha, como no exemplo a seguir.

```
hive://jdbc:hive2://hive1host:10000/default?...&UID=sample2&PWD=sample2&...
```

Atualmente, o conector Hortonworks Hive reconhece as propriedades do JDBC `UID` e `PWD`.

Uso de um único manipulador de conexão

É possível usar os seguintes metadados de conexão única e manipuladores de registros para se conectar a uma única instância do Hortonworks Hive.

Tipo de manipulador	Classe
Manipulador composto	`HiveCompositeHandler`
Manipulador de metadados	`HiveMetadataHandler`
Manipulador de registros	`HiveRecordHandler`

Parâmetros do manipulador de conexão única

Parâmetro	Descrição
`default`	Obrigatório. A string de conexão padrão.

Os manipuladores de conexão únicos oferecem suporte a uma instância de banco de dados e devem fornecer um parâmetro de string de conexão `default`. Todas as outras strings de conexão são ignoradas.

O exemplo de propriedade a seguir é para uma única instância do Hortonworks Hive com suporte em uma função do Lambda.

Propriedade	Valor
`default`	`hive://jdbc:hive2://hive1host:10000/default?secret=${Test/RDS/hive1host}`

Parâmetros de derramamento

O SDK do Lambda pode derramar dados no Amazon S3. Todas as instâncias do banco de dados acessadas pela mesma função do Lambda derramam no mesmo local.

Parâmetro	Descrição
`spill_bucket`	Obrigatório. Nome do bucket de derramamento.
`spill_prefix`	Obrigatório. Prefixo de chave do bucket de derramamento.
`spill_put_request_headers`	(Opcional) Um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.

Suporte ao tipo de dados

A tabela a seguir mostra os correspondentes tipos de dados do JDBC, do Hortonworks Hive e do Arrow.

JDBC	Hortonworks Hive	Arrow
Booleano	Booleano	Bit
Inteiro	TINYINT	Tiny
Short	SMALLINT	Smallint
Inteiro	INT	Int
Longo	BIGINT	Bigint
float	float4	Float4
Double	float8	Float8

JDBC	Hortonworks Hive	Arrow
Data	date	Data/Dia
Timestamp	timestamp	Date Milli
String	VARCHAR	Varchar
Bytes	bytes	Varbinary
BigDecimal	Decimal	Decimal
ARRAY	N/D (ver nota)	Lista

> **Note**
>
> Atualmente, o Hortonworks Hive não oferece suporte para os tipos agregados `ARRAY`, `MAP`, `STRUCT` ou `UNIONTYPE`. As colunas de tipos agregados são tratadas como colunas `VARCHAR` pelo SQL.

Partições e divisões

As partições são usadas para determinar como gerar divisões para o conector. O Athena constrói uma coluna sintética do tipo `varchar` que representa o esquema de particionamento da tabela para ajudar o conector a gerar divisões. O conector não modifica a definição real da tabela.

Performance

O Hortonworks Hive oferece suporte a partições estáticas. O conector do Athena para o Hortonworks Hive pode recuperar dados dessas partições em paralelo. Se você quiser consultar conjuntos de dados muito grandes com distribuição uniforme de partições, o particionamento estático é altamente recomendado. A seleção de um subconjunto de colunas acelera o runtime da consulta e reduz os dados verificados de forma significativa. O conector Hortonworks Hive é resiliente ao controle de utilização devido à simultaneidade.

O conector do Athena para o Hortonworks Hive executa a passagem direta de predicados para diminuir os dados examinados pela consulta. Cláusulas `LIMIT`, predicados simples e expressões

complexas são passados para o conector para reduzir a quantidade de dados examinados e o runtime de execução da consulta.

Cláusulas LIMIT

Uma instrução `LIMIT N` reduz os dados examinados pela consulta. Com a passagem direta de `LIMIT N`, o conector só retorna `N` linhas para o Athena.

Predicados

Um predicado é uma expressão na cláusula `WHERE` de uma consulta SQL, que avalia para um valor booleano e filtra as linhas com base em várias condições. O conector do Athena para o Hortonworks Hive pode combinar essas expressões e passá-las diretamente ao Hortonworks Hive para melhorar a funcionalidade e reduzir a quantidade de dados examinados.

Os seguintes operadores do conector do Athena para o Hortonworks Hive são compatíveis com a passagem direta de predicados:

- Booleanos: E, OU, NÃO
- Igualdade: EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_NULL
- Aritméticos: ADICIONAR, SUBTRAIR, MULTIPLICAR, DIVIDIR, MÓDULO, NEGAR
- Outros:LIKE_PATTERN, IN

Exemplo de passagem direta combinada

Para ter recursos aprimorados de consulta, combine os tipos de passagem direta, como no seguinte exemplo:

```
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;
```

Consultas de passagem

O conector Hortonworks Hive é compatível com consultas de passagem. As consultas de passagem usam uma função de tabela para enviar sua consulta completa para execução na fonte de dados.

Para usar consultas de passagem com o Hortonworks Hive, você pode empregar a seguinte sintaxe:

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

O exemplo de consulta a seguir envia uma consulta para uma fonte de dados no Hortonworks Hive. A consulta seleciona todas as colunas na tabela `customer`, limitando os resultados a 10.

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

Informações de licença

Ao usar esse conector, você reconhece a inclusão de componentes de terceiros, cuja lista pode ser encontrada no arquivo pom.xml desse conector, e concorda com os termos das respectivas licenças de terceiros fornecidas no arquivo LICENSE.txt em GitHub.com.

Recursos adicionais do

Para obter as informações sobre a versão do driver JDBC mais recentes, consulte o arquivo pom.xml do conector Hortonworks Hive em GitHub.com.

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Apache Kafka do Amazon Athena

O conector do Amazon Athena para o Apache Kafka possibilita que o Amazon Athena execute consultas SQL em seus tópicos do Apache Kafka. Use esse conector para visualizar os tópicos e as mensagens do Apache Kafka no Athena como tabelas e linhas, respectivamente.

Pré-requisitos

Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Limitações

- Não há suporte para operações de gravação de DDL.
- Quaisquer limites relevantes do Lambda. Para obter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do AWS Lambda.
- Em condições de filtro, você deve converter os tipos de dados date e timestamp para os tipos de dados apropriados.
- Os tipos de dados de data e carimbo de data/hora não são compatíveis com o tipo de arquivo CSV e são tratados como valores varchar.
- Não há suporte para o mapeamento em campos JSON aninhados. O conector mapeia somente os campos de nível superior.
- O conector não é compatível com tipos complexos. Tipos complexos são interpretados como strings.
- Para extrair ou trabalhar com valores JSON complexos, use as funções relacionadas ao JSON disponíveis no Athena. Para ter mais informações, consulte Extrair dados do JSON.
- O conector não oferece suporte ao acesso a metadados de mensagens do Kafka.

Termos

- Manipulador de metadados: um manipulador Lambda que recupera metadados da sua instância de banco de dados.
- Manipulador de registros: um manipulador Lambda que recupera registros de dados da sua instância de banco de dados.
- Manipulador composto: um manipulador Lambda que recupera tanto metadados quanto registros de dados da sua instância de banco de dados.
- Endpoint do Kafka: uma string de texto que estabelece uma conexão com uma instância do Kafka.

Compatibilidade de cluster

O conector do Kafka pode ser usado com os seguintes tipos de cluster:

- Kafka dedicado: uma conexão direta com o Kafka (autenticada ou não autenticada).
- Confluent - Uma conexão direta com o Confluent Kafka. Para obter informações sobre como usar o Athena com dados do Confluent Kafka, consulte Visualizar dados do Confluent no Amazon QuickSight usando o Amazon Athena no Blog de Business Intelligence. AWS

Conectando-se ao Confluent

A conexão com o Confluent requer as etapas a seguir:

1. Gere uma chave de API do Confluent.
2. Armazene o nome de usuário e a senha da chave da API Confluent em AWS Secrets Manager.
3. Forneça o nome secreto da variável de `secrets_manager_secret` ambiente no conector Kafka.
4. Siga as etapas na seção Como configurar o conector do Kafka deste documento.

Métodos de autenticação compatíveis

O conector é compatível com os métodos de autenticação a seguir.

- SSL
- SASL/SCRAM
- SASL/PLAIN
- SASL/PLAINTEXT
- NO_AUTH
- Plataforma Kafka e Confluent autogerenciada - SSL, SASL/SCRAM, SASL/PLAINTEXT, NO_AUTH
- Kafka autogerenciado e Confluent Cloud - SASL/PLAIN

Para ter mais informações, consulte Configurar a autenticação do conector do Kafka para o Athena.

Formatos de dados de entrada compatíveis

O conector é compatível com os seguintes formatos de dados de entrada:

- JSON
- CSV

Parâmetros

Use as variáveis de ambiente do Lambda mencionadas nesta seção para configurar o conector do Kafka para o Athena.

- auth_type: especifica o tipo de autenticação do cluster. O conector é compatível com os tipos de autenticação a seguir:
  - NO_AUTH: conexão direta ao Kafka (por exemplo, um cluster do Kafka implantado em uma instância do EC2 que não usa autenticação).
  - SASL_SSL_PLAIN: esse método usa o protocolo de segurança `SASL_SSL` e o mecanismo SASL `PLAIN`. Para obter mais informações, consulte Configuração do SASL na documentação do Apache Kafka.
  - SASL_PLAINTEXT_PLAIN: esse método usa o protocolo de segurança `SASL_PLAINTEXT` e o mecanismo SASL `PLAIN`. Para obter mais informações, consulte Configuração do SASL na documentação do Apache Kafka.
  - SASL_SSL_SCRAM_SHA512: você pode usar esse tipo de autenticação para controlar o acesso aos clusters do Apache Kafka. Esse método armazena o nome do usuário e a senha no AWS Secrets Manager. O segredo deve estar associado ao cluster do Kafka. Para obter mais informações, consulte Autenticação usando SASL/SCRAM na documentação do Apache Kafka.
  - SASL_PLAINTEXT_SCRAM_SHA512: este método usa o protocolo de segurança `SASL_PLAINTEXT` e o mecanismo `SCRAM_SHA512 SASL`. Esse método usa seu nome de usuário e a senha armazenados no AWS Secrets Manager. Para obter mais informações, consulte a seção Configuração do SASL na documentação do Apache Kafka.
  - SSL: a autenticação SSL usa os arquivos de armazenamento de chaves e de armazenamento confiável para se conectar ao cluster do Apache Kafka. Você deve gerar os arquivos de armazenamento confiável e de armazenamento de chaves, carregá-los em um bucket do Amazon S3 e fornecer a referência ao Amazon S3 ao implantar o conector. O armazenamento de chaves, o armazenamento confiável e a chave SSL serão armazenados no AWS Secrets Manager. Seu cliente deve fornecer a chave secreta da AWS ao implantar o conector. Para obter mais informações, consulte Criptografia e autenticação usando SSL na documentação do Apache Kafka.

    Para ter mais informações, consulte Configurar a autenticação do conector do Kafka para o Athena.
- certificates_s3_reference: o local do Amazon S3 que contém os certificados (os arquivos de armazenamento de chaves e de armazenamento confiável).
- disable_spill_encryption: (opcional) quando definido como `True`, desativa a criptografia do derramamento. É padronizado como `False`, para que os dados transmitidos para o S3 sejam criptografados usando o AES-GCM — usando uma chave gerada aleatoriamente ou o KMS

para gerar chaves. Desativar a criptografia do derramamento pode melhorar a performance, especialmente se o local do derramamento usar criptografia no lado do servidor.

- kafka_endpoint: os detalhes do endpoint a serem fornecidos para o Kafka.
- secrets_manager_secret: o nome do segredo da AWS no qual as credenciais são salvas.
- Parâmetros de vazamento: as funções do Lambda armazenam temporariamente dados (“de vazamentos”) que não cabem na memória do Amazon S3. Todas as instâncias do banco de dados acessadas pela mesma função do Lambda derramam no mesmo local. Use os parâmetros na tabela a seguir para especificar o local do vazamento.

Parâmetro	Descrição
`spill_bucket`	Obrigatório. O nome do bucket do Amazon S3 no qual a função do Lambda pode realizar o vazamento dos dados.
`spill_prefix`	Obrigatório. O prefixo que acompanha o bucket de vazamento no qual a função do Lambda pode realizar o vazamento dos dados.
`spill_put_request_headers`	(Opcional) Um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.

- Subnet IDs - Um ou mais IDs de sub-rede que correspondem à sub-rede que a função do Lambda pode usar para acessar sua fonte de dados.
  - Cluster Kafka público ou cluster padrão do Confluent Cloud — Associe o conector a uma sub-rede privada que tenha um gateway NAT.
  - Cluster do Confluent Cloud com conectividade privada - Associe o conector a uma sub-rede privada que tenha uma rota para o cluster Confluent Cloud.
    - Para o AWS Transit Gateway, as sub-redes devem estar em uma VPC conectada ao mesmo gateway de trânsito que o Confluent Cloud usa.
    - No caso de VPC Peering, as sub-redes devem estar em uma VPC que está emparelhada com a VPC do Confluent Cloud.

- Para AWS PrivateLink, as sub-redes devem estar em uma VPC que tenha uma rota para os endpoints da VPC que se conectam ao Confluent Cloud.

> **Note**
>
> Se você implantar o conector em uma VPC para acessar recursos privados e também quiser estabelecer uma conexão com um serviço acessível ao público, como o Confluent, deverá associar o conector a uma sub-rede privada que tenha um gateway NAT. Para obter mais informações, consulte Gateways NAT no Guia do usuário da Amazon VPC.

Suporte ao tipo de dados

A tabela a seguir apresenta os tipos de dados correspondentes compatíveis com o Kafka e o Apache Arrow.

Kafka	Arrow
CHAR	VARCHAR
VARCHAR	VARCHAR
TIMESTAMP	MILLISECOND
DATA	DAY
BOOLEAN	BOOL
SMALLINT	SMALLINT
INTEGER	INT
BIGINT	BIGINT
DECIMAL	FLOAT8
DOUBLE	FLOAT8

Partições e divisões

Os tópicos do Kafka são divididos em partições. Cada partição é ordenada. Cada mensagem em uma partição tem um ID incremental denominado deslocamento. Cada partição do Kafka é separada em diversas divisões para o processamento paralelo. Os dados estão disponíveis para o período de retenção configurado nos clusters do Kafka.

Práticas recomendadas

Como prática recomendada, use o empilhamento de predicados ao consultar o Athena, como mostrado nos exemplos a seguir.

```
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'
```

Como configurar o conector do Kafka

Antes de usar o conector, você deve configurar seu cluster do Apache Kafka. Use o registro de esquemas do AWS Glue para definir seu esquema e configurar a autenticação para o conector.

Ao trabalhar com o registro de esquemas do AWS Glue, preste atenção aos seguintes pontos:

- Certifique-se de que o texto no campo Description (Descrição) do registro de esquemas do AWS Glue inclua a string `{AthenaFederationKafka}`. Essa string de marcação é necessária para os registros do AWS Glue usados com o conector do Kafka para o Amazon Athena.
- Para obter a melhor performance, use somente letras minúsculas para nomes de banco de dados e nomes de tabela. O uso de maiúsculas e minúsculas mistas faz com que o conector execute uma pesquisa que não diferencia maiúsculas de minúsculas e é mais computacionalmente intensiva.

Para configurar o ambiente do Apache Kafka e o registro de esquemas do AWS Glue

1. Configure o ambiente do Apache Kafka.

2. Faça o upload do arquivo de descrição do tópico do Kafka (ou seja, seu esquema) no formato JSON para o registro de esquemas do AWS Glue. Para obter mais informações, consulte Integração com o registro de esquemas do AWS Glue no Guia do desenvolvedor do AWS Glue. Para obter esquemas de exemplo, consulte a seção a seguir.

Exemplos de esquema para o registro de esquemas do AWS Glue

Use o formato dos exemplos apresentados nessa seção ao fazer upload de seu esquema para o registro de esquemas do AWS Glue.

Exemplo de esquema: tipo JSON

No exemplo a seguir, o esquema a ser criado no registro de esquemas do AWS Glue especifica `json` como o valor para `dataFormat` e usa `datatypejson` para `topicName`.

> **Note**
>
> O valor para `topicName` deve usar a mesma capitalização que o nome do tópico no 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"
      }
    ]
  }
}
```

Exemplo de esquema: tipo CSV

No exemplo a seguir, o esquema a ser criado no registro de esquemas do AWS Glue especifica `csv` como o valor para `dataFormat` e usa `datatypecsvbulk` para `topicName`. O valor para `topicName` deve usar a mesma capitalização que o nome do tópico no 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"
      }
    ]
```

```
  }
}
```

Configurar a autenticação do conector do Kafka para o Athena

É possível usar uma variedade de métodos para autenticar o cluster do Apache Kafka, como SSL, SASL/SCRAM, SASL/PLAIN e SASL/PLAINTEXT.

A tabela a seguir mostra os tipos de autenticação para o conector, e o protocolo de segurança e o mecanismo SASL para cada um. Para obter mais informações, consulte a seção Segurança da documentação do Apache Kafka.

auth_type	security.protocol	sasl.mechanism	Compatibilidade do tipo de cluster
SASL_SSL_PLAIN	SASL_SSL	PLAIN	• Kafka autogerenciado • Plataforma Confluent • Confluent Cloud
SASL_PLAINTEXT_PLAIN	SASL_PLAINTEXT	PLAIN	• Kafka autogerenciado • Plataforma Confluent
SASL_SSL_SCRAM_SHA512	SASL_SSL	SCRAM-SHA-512	• Kafka autogerenciado • Plataforma Confluent
SASL_PLAINTEXT_SCRAM_SHA512	SASL_PLAINTEXT	SCRAM-SHA-512	• Kafka autogerenciado • Plataforma Confluent
SSL	SSL	N/D	• Kafka autogerenciado • Plataforma Confluent

SSL

Se o cluster for autenticado por SSL, você deverá gerar os arquivos de armazenamento confiável e de armazenamento de chaves, e fazer o upload deles no bucket do Amazon S3. Você deverá fornecer essa referência do Amazon S3 ao implantar o conector. O armazenamento de chaves, o armazenamento confiável e a chave SSL serão armazenados no AWS Secrets Manager. Você fornecerá a chave secreta da AWS ao implantar o conector.

Para obter informações sobre como criar um segredo no Secrets Manager, consulte Criação de um segredo do AWS Secrets Manager.

Para usar esse tipo de autenticação, defina as variáveis de ambiente conforme mostrado na tabela a seguir.

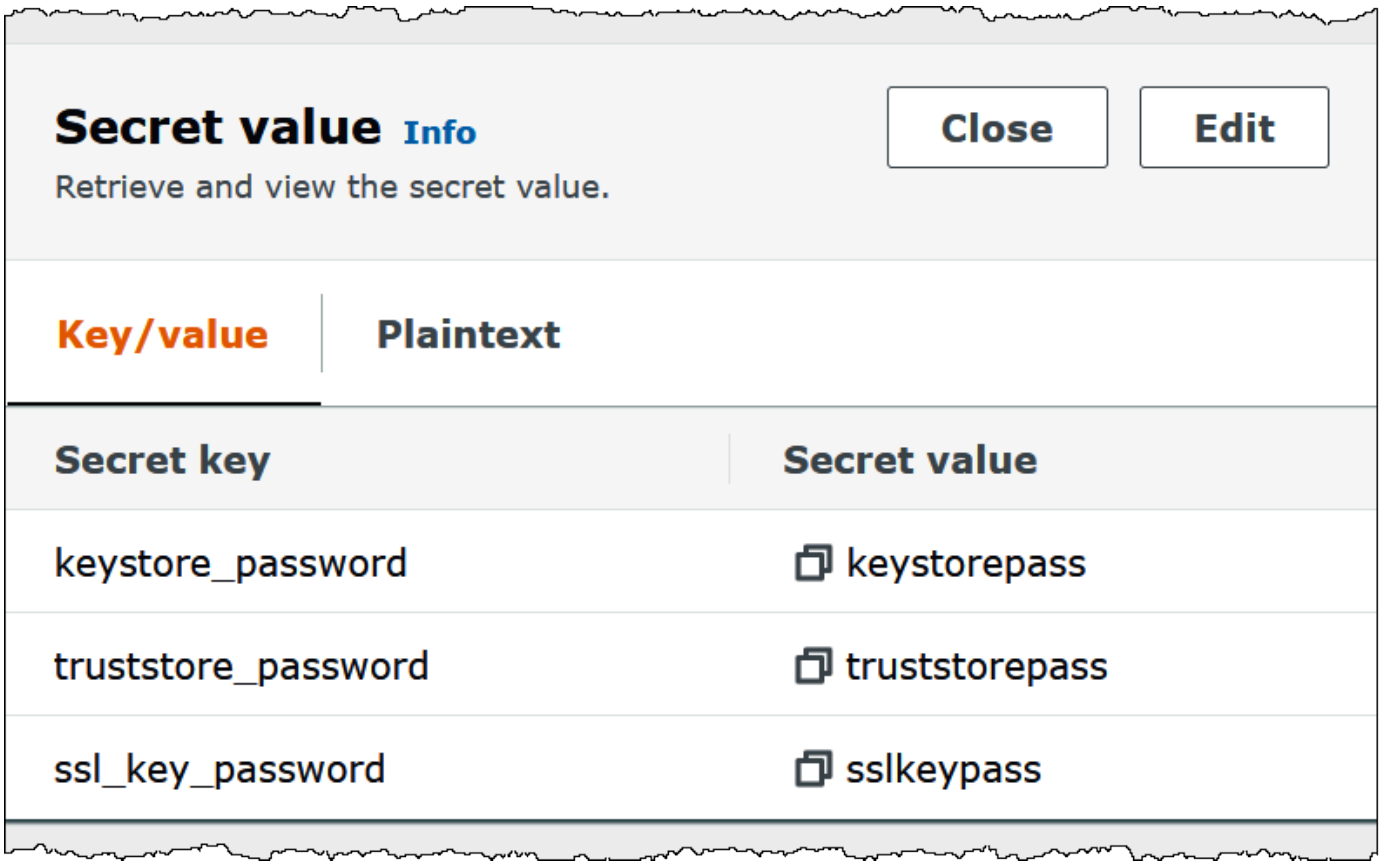
Parâmetro	Valor
`auth_type`	`SSL`
`certificates_s3_reference`	O local do Amazon S3 que contém os certificados.
`secrets_manager_secret`	O nome da chave secreta da AWS.

Após a criação de um segredo no Secrets Manager, é possível visualizá-lo no console do Secrets Manager.

Para visualizar o segredo no Secrets Manager

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. No painel de navegação, escolha Secrets (Segredos).
3. Na página Secrets (Segredos), selecione o link do seu segredo.
4. Na página de detalhes do seu segredo, escolha Retrieve secret value (Recuperar valor do segredo).

   A imagem a seguir apresenta um exemplo de segredo com três pares chave/valor: `keystore_password`, `truststore_password` e `ssl_key_password`.



Para obter mais informações sobre como usar SSL com o Kafka, consulte Criptografia e autenticação usando SSL na documentação do Apache Kafka.

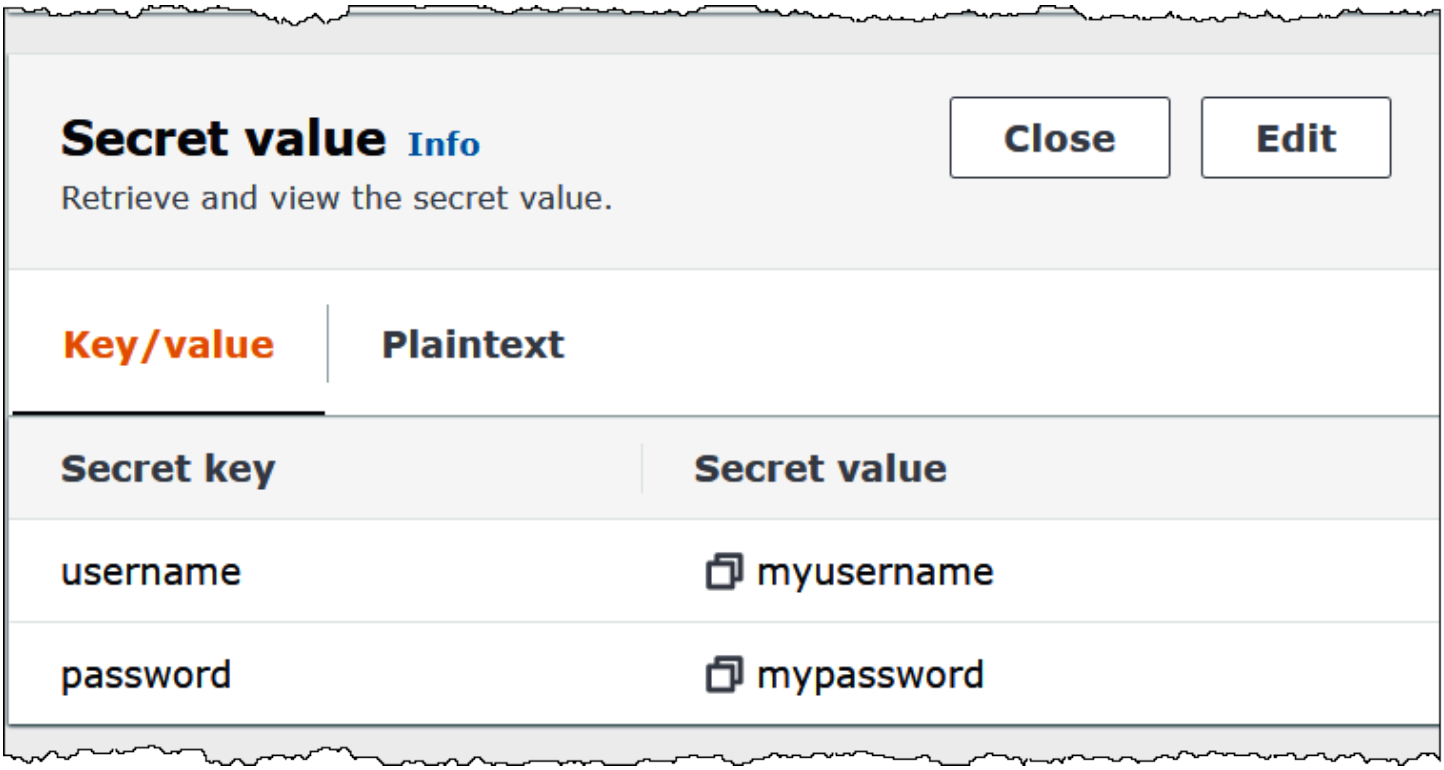
SASL/SCRAM

Se o cluster usar a autenticação SCRAM, forneça a chave do Secrets Manager associada ao cluster ao implantar o conector. As credenciais da AWS do usuário (chave secreta e chave de acesso) serão usadas para realizar a autenticação com o cluster.

Defina as variáveis de ambiente conforme mostrado na tabela a seguir.

Parâmetro	Valor
`auth_type`	`SASL_SSL_SCRAM_SHA512`
`secrets_manager_secret`	O nome da chave secreta da AWS.

A imagem a seguir apresenta um exemplo de segredo no console do Secrets Manager com dois pares chave/valor: um para `username` e outro para `password`.



Para obter mais informações sobre como usar o SASL/SCRAM com o Kafka, consulte Autenticação usando SASL/SCRAM na documentação do Apache Kafka.

Informações de licença

Ao usar esse conector, você reconhece a inclusão de componentes de terceiros, cuja lista pode ser encontrada no arquivo pom.xml desse conector, e concorda com os termos das respectivas licenças de terceiros fornecidas no arquivo LICENSE.txt em GitHub.com.

Recursos adicionais do

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector MSK do Amazon Athena

O conector do Amazon Athena para o Amazon MSK possibilita que o Amazon Athena execute consultas SQL em seus tópicos do Apache Kafka. Use esse conector para visualizar os tópicos e as mensagens do Apache Kafka no Athena como tabelas e linhas, respectivamente. Para obter informações adicionais, consulte Analyze real-time streaming data in Amazon MSK with Amazon

Athena (Analisar dados de streaming em tempo real no Amazon MSK com o Amazon Athena) no blog de big data da AWS.

Pré-requisitos

Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Limitações

- Não há suporte para operações de gravação de DDL.
- Quaisquer limites relevantes do Lambda. Para obter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do AWS Lambda.
- Em condições de filtro, você deve converter os tipos de dados date e timestamp para os tipos de dados apropriados.
- Os tipos de dados de data e carimbo de data/hora não são compatíveis com o tipo de arquivo CSV e são tratados como valores varchar.
- Não há suporte para o mapeamento em campos JSON aninhados. O conector mapeia somente os campos de nível superior.
- O conector não é compatível com tipos complexos. Tipos complexos são interpretados como strings.
- Para extrair ou trabalhar com valores JSON complexos, use as funções relacionadas ao JSON disponíveis no Athena. Para ter mais informações, consulte Extrair dados do JSON.
- O conector não oferece suporte ao acesso a metadados de mensagens do Kafka.

Termos

- Manipulador de metadados: um manipulador Lambda que recupera metadados da sua instância de banco de dados.
- Manipulador de registros: um manipulador Lambda que recupera registros de dados da sua instância de banco de dados.
- Manipulador composto: um manipulador Lambda que recupera tanto metadados quanto registros de dados da sua instância de banco de dados.
- Endpoint do Kafka: uma string de texto que estabelece uma conexão com uma instância do Kafka.

Compatibilidade de cluster

O conector MSK pode ser usado com os seguintes tipos de cluster:

- Cluster provisionado pelo MSK: você especifica, monitora e escala manualmente a capacidade do cluster.
- Cluster do MSK Serverless: fornece uma capacidade sob demanda que é escalada automaticamente conforme a escalabilidade de E/S da aplicação.
- Kafka dedicado: uma conexão direta com o Kafka (autenticada ou não autenticada).

Métodos de autenticação compatíveis

O conector é compatível com os métodos de autenticação a seguir.

- SASL/IAM
- SSL
- SASL/SCRAM
- SASL/PLAIN
- SASL/PLAINTEXT
- NO_AUTH

  Para ter mais informações, consulte Configuração da autenticação para o conector MSK do Athena.

Formatos de dados de entrada compatíveis

O conector é compatível com os seguintes formatos de dados de entrada:

- JSON
- CSV

Parâmetros

Use as variáveis de ambiente do Lambda mencionadas nessa seção para configurar o conector MSK do Athena.

- auth_type: especifica o tipo de autenticação do cluster. O conector é compatível com os tipos de autenticação a seguir:
  - NO_AUTH: conexão direta ao Kafka sem autenticação (por exemplo, um cluster do Kafka implantado em uma instância do EC2 que não usa autenticação).
  - SASL_SSL_PLAIN: esse método usa o protocolo de segurança `SASL_SSL` e o mecanismo SASL `PLAIN`.
  - SASL_PLAINTEXT_PLAIN: esse método usa o protocolo de segurança `SASL_PLAINTEXT` e o mecanismo SASL `PLAIN`.

    > **Note**
    >
    > Os tipos de autenticação `SASL_SSL_PLAIN` e `SASL_PLAINTEXT_PLAIN` são compatíveis com o Apache Kafka, mas não no Amazon MSK.

  - SASL_SSL_AWS_MSK_IAM: o controle de acesso do IAM para o Amazon MSK permite que você gerencie a autenticação e a autorização para o cluster do MSK. As credenciais da AWS do seu usuário (chave secreta e chave de acesso) serão usadas para se conectar ao cluster. Para obter mais informações, consulte IAM access control (Controle de acesso do IAM) no Guia do desenvolvedor do Amazon Managed Streaming for Apache Kafka.
  - SASL_SSL_SCRAM_SHA512: você pode usar esse tipo de autenticação para controlar o acesso aos seus clusters do Amazon MSK. Esse método armazena o nome do usuário e a senha no AWS Secrets Manager. O segredo deve estar associado ao cluster do Amazon MSK. Para obter mais informações, consulte Setting up SASL/SCRAM authentication for an Amazon MSK cluster (Configuração da autenticação SASL/SCRAM para um cluster do Amazon MSK) no Guia do desenvolvedor do Amazon Managed Streaming for Apache Kafka.
  - SSL: a autenticação SSL usa os arquivos de armazenamento de chaves e de armazenamento confiável para se conectar ao cluster do Amazon MSK. Você deve gerar os arquivos de armazenamento confiável e de armazenamento de chaves, carregá-los em um bucket do Amazon S3 e fornecer a referência ao Amazon S3 ao implantar o conector. O armazenamento de chaves, o armazenamento confiável e a chave SSL serão armazenados no AWS Secrets Manager. Seu cliente deve fornecer a chave secreta da AWS ao implantar o conector. Para obter mais informações, consulte Mutual TLS authentication (Autenticação mútua TLS) no Guia do desenvolvedor do Amazon Managed Streaming for Apache Kafka.

    Para ter mais informações, consulte Configuração da autenticação para o conector MSK do Athena.

- certificates_s3_reference: o local do Amazon S3 que contém os certificados (os arquivos de armazenamento de chaves e de armazenamento confiável).
- disable_spill_encryption: (opcional) quando definido como `True`, desativa a criptografia do derramamento. É padronizado como `False`, para que os dados transmitidos para o S3 sejam criptografados usando o AES-GCM — usando uma chave gerada aleatoriamente ou o KMS para gerar chaves. Desativar a criptografia do derramamento pode melhorar a performance, especialmente se o local do derramamento usar criptografia no lado do servidor.
- kafka_endpoint: os detalhes do endpoint a serem fornecidos para o Kafka. Por exemplo, para um cluster do Amazon MSK, você fornece um URL de inicialização para o cluster.
- secrets_manager_secret: o nome do segredo da AWS no qual as credenciais são salvas. Esse parâmetro não é necessário para a autenticação do IAM.
- Parâmetros de vazamento: as funções do Lambda armazenam temporariamente dados (“de vazamentos”) que não cabem na memória do Amazon S3. Todas as instâncias do banco de dados acessadas pela mesma função do Lambda derramam no mesmo local. Use os parâmetros na tabela a seguir para especificar o local do vazamento.

Parâmetro	Descrição
`spill_bucket`	Obrigatório. O nome do bucket do Amazon S3 no qual a função do Lambda pode realizar o vazamento dos dados.
`spill_prefix`	Obrigatório. O prefixo que acompanha o bucket de vazamento no qual a função do Lambda pode realizar o vazamento dos dados.
`spill_put_request_headers`	(Opcional) Um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.

Suporte ao tipo de dados

A tabela a seguir apresenta os tipos de dados correspondentes compatíveis com o Kafka e o Apache Arrow.

Kafka	Arrow
CHAR	VARCHAR
VARCHAR	VARCHAR
TIMESTAMP	MILLISECOND
DATA	DAY
BOOLEAN	BOOL
SMALLINT	SMALLINT
INTEGER	INT
BIGINT	BIGINT
DECIMAL	FLOAT8
DOUBLE	FLOAT8

Partições e divisões

Os tópicos do Kafka são divididos em partições. Cada partição é ordenada. Cada mensagem em uma partição tem um ID incremental denominado deslocamento. Cada partição do Kafka é separada em diversas divisões para o processamento paralelo. Os dados estão disponíveis para o período de retenção configurado nos clusters do Kafka.

Práticas recomendadas

Como prática recomendada, use o empilhamento de predicados ao consultar o Athena, como mostrado nos exemplos a seguir.

```
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'
```

Configuração do conector MSK

Antes de usar o conector, você deve configurar seu cluster do Amazon MSK. Use o registro de esquemas do AWS Glue para definir seu esquema e configurar a autenticação para o conector.

> **Note**
>
> Se você implantar o conector em uma VPC para acessar recursos privados e também quiser estabelecer uma conexão com um serviço acessível ao público, como o Confluent, deverá associar o conector a uma sub-rede privada que tenha um gateway NAT. Para obter mais informações, consulte Gateways NAT no Guia do usuário da Amazon VPC.

Ao trabalhar com o registro de esquemas do AWS Glue, preste atenção aos seguintes pontos:

- Certifique-se de que o texto no campo Description (Descrição) do registro de esquemas do AWS Glue inclua a string `{AthenaFederationMSK}`. Essa string de marcação é necessária para os registros do AWS Glue usados com o conector MSK do Amazon Athena.
- Para obter a melhor performance, use somente letras minúsculas para nomes de banco de dados e nomes de tabela. O uso de maiúsculas e minúsculas mistas faz com que o conector execute uma pesquisa que não diferencia maiúsculas de minúsculas e é mais computacionalmente intensiva.

Para configurar o ambiente do Amazon MSK e o registro de esquemas do AWS Glue

1. Configure o ambiente do Amazon MSK. Para obter mais informações e etapas, consulte Setting up Amazon MSK (Configuração do Amazon MSK) e Getting started using Amazon MSK (Comece a usar o Amazon MSK) no Guia do desenvolvedor do Amazon Managed Streaming for Apache Kafka.
2. Faça o upload do arquivo de descrição do tópico do Kafka (ou seja, seu esquema) no formato JSON para o registro de esquemas do AWS Glue. Para obter mais informações, consulte Integração com o registro de esquemas do AWS Glue no Guia do desenvolvedor do AWS Glue. Para obter esquemas de exemplo, consulte a seção a seguir.

Exemplos de esquema para o registro de esquemas do AWS Glue

Use o formato dos exemplos apresentados nessa seção ao fazer upload de seu esquema para o registro de esquemas do AWS Glue.

Exemplo de esquema: tipo JSON

No exemplo a seguir, o esquema a ser criado no registro de esquemas do AWS Glue especifica `json` como o valor para `dataFormat` e usa `datatypejson` para `topicName`.

> **Note**
>
> O valor para `topicName` deve usar a mesma capitalização que o nome do tópico no 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"
      }
    ]
  }
}
```

Exemplo de esquema: tipo CSV

No exemplo a seguir, o esquema a ser criado no registro de esquemas do AWS Glue especifica `csv` como o valor para `dataFormat` e usa `datatypecsvbulk` para `topicName`. O valor para `topicName` deve usar a mesma capitalização que o nome do tópico no 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"
      }
    ]
  }
}
```

Configuração da autenticação para o conector MSK do Athena

É possível usar uma variedade de métodos para autenticar seu cluster do Amazon MSK, incluindo IAM, SSL, SCRAM e Kafka dedicado.

A tabela a seguir mostra os tipos de autenticação para o conector, e o protocolo de segurança e o mecanismo SASL para cada um. Para obter mais informações, consulte Authentication and authorization for Apache Kafka APIs (Autenticação e autorização para APIs do Apache Kafka) no Guia do desenvolvedor do Amazon Managed Streaming for Apache Kafka.

auth_type	security.protocol	sasl.mechanism
`SASL_SSL_PLAIN`	`SASL_SSL`	`PLAIN`
`SASL_PLAINTEXT_PLAIN`	`SASL_PLAINTEXT`	`PLAIN`
`SASL_SSL_AWS_MSK_IAM`	`SASL_SSL`	`AWS_MSK_IAM`
`SASL_SSL_SCRAM_SHA512`	`SASL_SSL`	`SCRAM-SHA-512`
`SSL`	`SSL`	N/D

> **Note**
>
> Há suporte para os tipos de autenticação `SASL_SSL_PLAIN` e `SASL_PLAINTEXT_PLAIN` no Apache Kafka, mas não no Amazon MSK.

SASL/IAM

Se o cluster usar a autenticação do IAM, você deverá configurar a política do IAM para o usuário ao configurar o cluster. Para obter mais informações, consulte IAM access control (Controle de acesso do IAM) no Guia do desenvolvedor do Amazon Managed Streaming for Apache Kafka.

Para usar esse tipo de autenticação, defina a variável de ambiente do Lambda `auth_type` como `SASL_SSL_AWS_MSK_IAM` para o conector.

SSL

Se o cluster for autenticado por SSL, você deverá gerar os arquivos de armazenamento confiável e de armazenamento de chaves, e fazer o upload deles no bucket do Amazon S3. Você deverá fornecer essa referência do Amazon S3 ao implantar o conector. O armazenamento de chaves, o armazenamento confiável e a chave SSL serão armazenados no AWS Secrets Manager. Você fornecerá a chave secreta da AWS ao implantar o conector.

Para obter informações sobre como criar um segredo no Secrets Manager, consulte Criação de um segredo do AWS Secrets Manager.

Para usar esse tipo de autenticação, defina as variáveis de ambiente conforme mostrado na tabela a seguir.

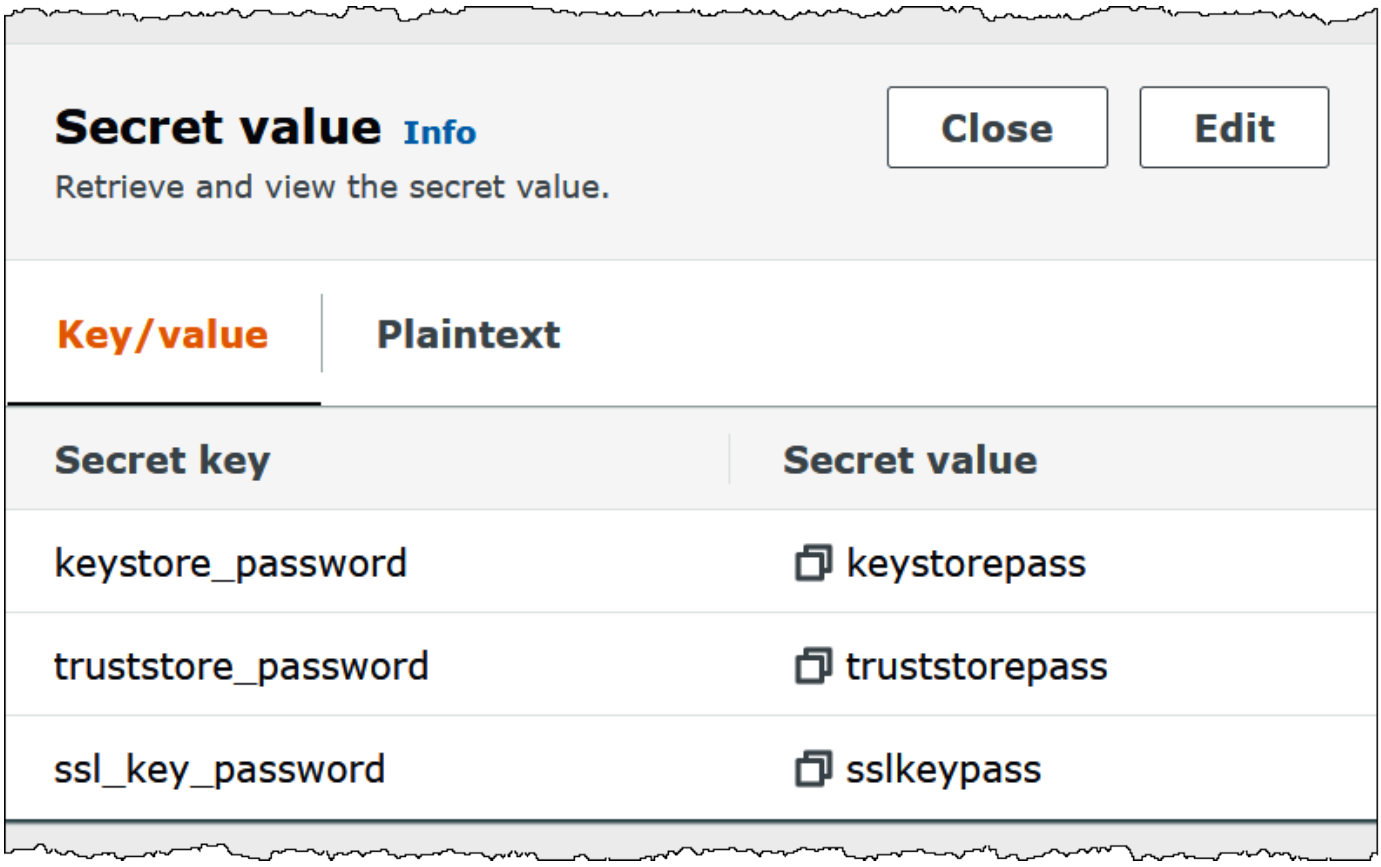
Parâmetro	Valor
`auth_type`	`SSL`
`certificates_s3_reference`	O local do Amazon S3 que contém os certificados.
`secrets_manager_secret`	O nome da chave secreta da AWS.

Após a criação de um segredo no Secrets Manager, é possível visualizá-lo no console do Secrets Manager.

Para visualizar o segredo no Secrets Manager

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. No painel de navegação, escolha Secrets (Segredos).
3. Na página Secrets (Segredos), selecione o link do seu segredo.
4. Na página de detalhes do seu segredo, escolha Retrieve secret value (Recuperar valor do segredo).

   A imagem a seguir apresenta um exemplo de segredo com três pares chave/valor: `keystore_password`, `truststore_password` e `ssl_key_password`.



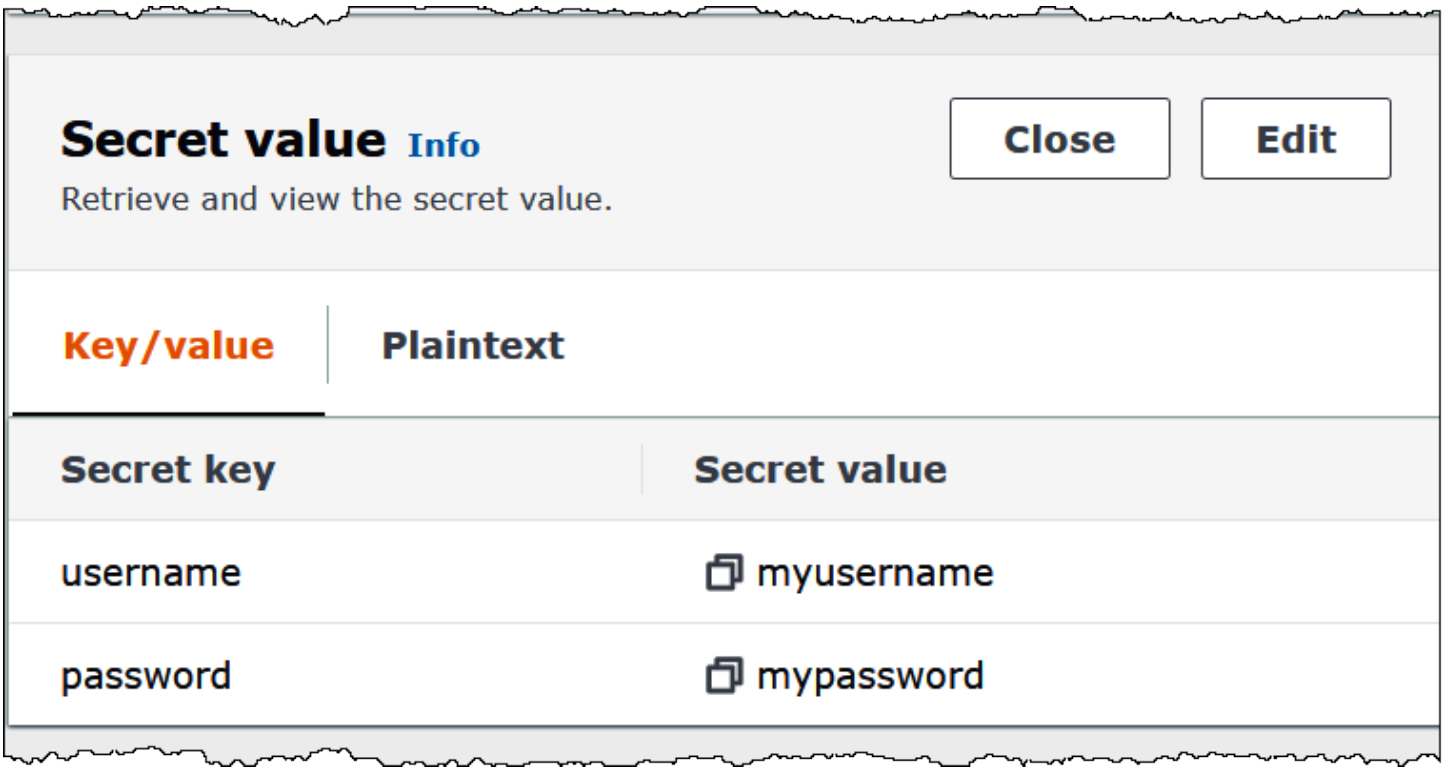
SASL/SCRAM

Se o cluster usar a autenticação SCRAM, forneça a chave do Secrets Manager associada ao cluster ao implantar o conector. As credenciais da AWS do usuário (chave secreta e chave de acesso) serão usadas para realizar a autenticação com o cluster.

Defina as variáveis de ambiente conforme mostrado na tabela a seguir.

Parâmetro	Valor
`auth_type`	`SASL_SSL_SCRAM_SHA512`
`secrets_manager_secret`	O nome da chave secreta da AWS.

A imagem a seguir apresenta um exemplo de segredo no console do Secrets Manager com dois pares chave/valor: um para `username` e outro para `password`.



Informações de licença

Ao usar esse conector, você reconhece a inclusão de componentes de terceiros, cuja lista pode ser encontrada no arquivo pom.xml desse conector, e concorda com os termos das respectivas licenças de terceiros fornecidas no arquivo LICENSE.txt em GitHub.com.

Recursos adicionais do

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o MySQL

O conector MySQL do Lambda no Amazon Athena permite que o Amazon Athena acesse bancos de dados MySQL.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.
- Configura uma VPC e um grupo de segurança antes de usar esse conector. Para ter mais informações, consulte Criar uma VPC para um conector de origem de dados.

Limitações

- Não há suporte para operações de gravação de DDL.
- Em uma configuração de multiplexador, o prefixo e o bucket de derramamento são compartilhados em todas as instâncias do banco de dados.
- Quaisquer limites relevantes do Lambda. Para obter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do AWS Lambda.
- Como o Athena converte as consultas em minúsculas, os nomes das tabelas do MySQL devem estar em minúsculas. Por exemplo, consultas do Athena em uma tabela chamada `myTable` falharão.

Termos

Os termos a seguir estão relacionados ao conector MySQL.

- Instância do banco de dados: qualquer instância de um banco de dados implantado on-premises, no Amazon EC2 ou no Amazon RDS.
- Manipulador: um manipulador Lambda que acessa sua instância de banco de dados. Um manipulador pode ser para metadados ou para registros de dados.
- Manipulador de metadados: um manipulador Lambda que recupera metadados da sua instância de banco de dados.
- Manipulador de registros: um manipulador Lambda que recupera registros de dados da sua instância de banco de dados.
- Manipulador composto: um manipulador Lambda que recupera tanto metadados quanto registros de dados da sua instância de banco de dados.
- Propriedade ou parâmetro: uma propriedade do banco de dados usada pelos manipuladores para extrair informações do banco de dados. Você configura essas propriedades como variáveis de ambiente do Lambda.
- String de conexão: uma string de texto usada para estabelecer uma conexão com uma instância de banco de dados.
- Catálogo: um catálogo não AWS Glue registrado no Athena que é um prefixo obrigatório para a propriedade `connection_string`.
- Manipulador de multiplexação: um manipulador Lambda que pode aceitar e usar várias conexões de banco de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector MySQL.

String de conexão

Use uma string de conexão JDBC no seguinte formato para se conectar a uma instância de banco de dados.

```
mysql://${jdbc_connection_string}
```

> **Note**
>
> Se você receber o erro java.sql.SQLException: Zero date value prohibited (Valor de data zero proibido) ao fazer uma consulta `SELECT` em uma tabela do MySQL, adicione o seguinte parâmetro à sua string de conexão:
>
> ```
> zeroDateTimeBehavior=convertToNull
> ```
>
> Para obter mais informações, consulte: Erro 'Valor de data zero proibido' ao tentar selecionar em tabela do MySQL em GitHub.com.

Uso de um manipulador de multiplexação

É possível usar um multiplexador para se conectar a várias instâncias de banco de dados com uma única função do Lambda. As solicitações são encaminhadas por nome do catálogo. Use as seguintes classes no Lambda.

Manipulador	Classe
Manipulador composto	`MySqlMuxCompositeHandler`
Manipulador de metadados	`MySqlMuxMetadataHandler`
Manipulador de registros	`MySqlMuxRecordHandler`

Parâmetros do manipulador de multiplexação

Parâmetro	Descrição
`$catalog_connection_string`	Obrigatório. Uma string de conexão de instância de banco de dados. Prefixe a variável de ambiente com o nome do catálogo usado no Athena. Por exemplo, se o catálogo registrado no Athena for `mymysqlcatalog`, então o nome da variável de ambiente será `mymysqlcatalog_connection_string`.
`default`	Obrigatório. A string de conexão padrão. Essa string é usada quando o catálogo for `lambda:${AWS_LAMBDA_FUNCTION_NAME}`.

As propriedades de exemplo a seguir são para uma função do Lambda MUX do MySql que ofereça suporte a duas instâncias de banco de dados: `mysql1` (o padrão) e `mysql2`.

Propriedade	Valor
`default`	`mysql://jdbc:mysql://mysql2.host:3333/default? user=sample2&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=sample2&password=sample2`

Fornecimento de credenciais

Para fornecer um nome de usuário e uma senha para seu banco de dados na string de conexão JDBC, é possível usar as propriedades da string de conexão ou o AWS Secrets Manager.

- String de conexão: um nome de usuário e uma senha podem ser especificados como propriedades na string de conexão do JDBC.

  > **Important**
  >
  > Como prática recomendada de segurança, não use credenciais codificadas em suas variáveis de ambiente ou strings de conexão. Para obter informações sobre como mover seus segredos codificados para o AWS Secrets Manager, consulte Mover segredos codificados para o AWS Secrets Manager no Guia do usuário do AWS Secrets Manager.

- AWS Secrets Manager: para usar o recurso Athena Federated Query com o AWS Secrets Manager, a VPC conectada à sua função do Lambda deve ter acesso à Internet ou um endpoint da VPC para se conectar ao Secrets Manager.

  É possível colocar o nome de um segredo no AWS Secrets Manager na sua string de conexão JDBC. O conector substitui o nome secreto pelos valores de `username` e `password` do Secrets Manager.

  Para instâncias de banco de dados do Amazon RDS, esse suporte é totalmente integrado. Se você usa o Amazon RDS, é altamente recomendável usar o AWS Secrets Manager e rotação de credenciais. Se seu banco de dados não usar o Amazon RDS, armazene as credenciais em JSON no seguinte formato:

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

Exemplo de string de conexão com nome secreto

A string a seguir tem o nome secreto `${Test/RDS/MySql1}`.

```
mysql://jdbc:mysql://mysql1.host:3306/default?...&${Test/RDS/MySql1}&...
```

O conector usa o nome secreto para recuperar segredos e fornecer o nome de usuário e a senha, como no exemplo a seguir.

```
mysql://jdbc:mysql://mysql1host:3306/default?...&user=sample2&password=sample2&...
```

Atualmente, o conector MySQL reconhece as propriedades `user` e `password` do JDBC.

Uso de um único manipulador de conexão

É possível usar os seguintes metadados de conexão única e manipuladores de registros para se conectar a uma única instância do MySQL.

Tipo de manipulador	Classe
Manipulador composto	`MySqlCompositeHandler`
Manipulador de metadados	`MySqlMetadataHandler`
Manipulador de registros	`MySqlRecordHandler`

Parâmetros do manipulador de conexão única

Parâmetro	Descrição
`default`	Obrigatório. A string de conexão padrão.

Os manipuladores de conexão únicos oferecem suporte a uma instância de banco de dados e devem fornecer um parâmetro de string de conexão `default`. Todas as outras strings de conexão são ignoradas.

O exemplo de propriedade a seguir é para uma única instância do MySQL com suporte em uma função do Lambda.

Proprieda de	Valor
`default`	`mysql://mysql1.host:3306/default?secret=Test/RDS/` MySQL1

Parâmetros de derramamento

O SDK do Lambda pode derramar dados no Amazon S3. Todas as instâncias do banco de dados acessadas pela mesma função do Lambda derramam no mesmo local.

Parâmetro	Descrição
`spill_bucket`	Obrigatório. Nome do bucket de derramamento.
`spill_prefix`	Obrigatório. Prefixo de chave do bucket de derramamento.
`spill_put_request_headers`	(Opcional) Um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.

Suporte ao tipo de dados

A tabela a seguir mostra os correspondentes tipos de dados do JDBC e do Arrow.

JDBC	Arrow
Booleano	Bit
Inteiro	Tiny
Short	Smallint
Inteiro	Int
Longo	Bigint
float	Float4
Double	Float8
Data	Data/Dia
Timestamp	Date Milli
String	Varchar

JDBC	Arrow
Bytes	Varbinary
BigDecimal	Decimal
ARRAY	Lista

Partições e divisões

As partições são usadas para determinar como gerar divisões para o conector. O Athena constrói uma coluna sintética do tipo `varchar` que representa o esquema de particionamento da tabela para ajudar o conector a gerar divisões. O conector não modifica a definição real da tabela.

Performance

O MySQL oferece suporte a partições nativas. O conector do Athena para o Lambda pode recuperar dados dessas partições em paralelo. Se você quiser consultar conjuntos de dados muito grandes com distribuição uniforme de partições, o particionamento nativo é altamente recomendado.

O conector do Athena para o MySQL executa a passagem direta de predicados para diminuir os dados examinados pela consulta. Cláusulas `LIMIT`, predicados simples e expressões complexas são passados diretamente ao conector para reduzir a quantidade de dados examinados e o runtime de execução da consulta.

Cláusulas LIMIT

Uma instrução `LIMIT N` reduz os dados examinados pela consulta. Com a passagem direta de `LIMIT N`, o conector só retorna `N` linhas para o Athena.

Predicados

Um predicado é uma expressão na cláusula `WHERE` de uma consulta SQL, que avalia para um valor booleano e filtra as linhas com base em várias condições. O conector do Athena para o MySQL pode combinar essas expressões e passá-las diretamente ao MySQL para melhorar a funcionalidade e reduzir a quantidade de dados examinados.

Os seguintes operadores do conector do Athena para o MySQL são compatíveis com a passagem direta de predicados:

- Booleanos: E, OU, NÃO

- Igualdade: EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Aritméticos: ADICIONAR, SUBTRAIR, MULTIPLICAR, DIVIDIR, MÓDULO, NEGAR
- Outros:LIKE_PATTERN, IN

Exemplo de passagem direta combinada

Para ter recursos aprimorados de consulta, combine os tipos de passagem direta, como no seguinte exemplo:

```
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;
```

Para ver um artigo sobre como usar o pushdown de predicado para melhorar o desempenho em consultas federadas, incluindo MySQL, consulte Melhorar as consultas federadas com o pushdown de predicados no Amazon Athena no AWSBlog de Big Data.

Consultas de passagem

O conector MySQL é compatível com consultas de passagem. As consultas de passagem usam uma função de tabela para enviar sua consulta completa para execução na fonte de dados.

Para usar consultas de passagem com o MySQL, você pode empregar a seguinte sintaxe:

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

O exemplo de consulta a seguir envia uma consulta para uma fonte de dados no MySQL. A consulta seleciona todas as colunas na tabela `customer`, limitando os resultados a 10.

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

```
        ) )
```

Informações de licença

Ao usar esse conector, você reconhece a inclusão de componentes de terceiros, cuja lista pode ser encontrada no arquivo pom.xml desse conector, e concorda com os termos das respectivas licenças de terceiros fornecidas no arquivo LICENSE.txt em GitHub.com.

Recursos adicionais do

Para obter as informações mais recentes sobre a versão do driver JDBC, consulte o arquivo pom.xml do conector MySQL em GitHub.com.

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o Neptune

O Amazon Neptune é um serviço de banco de dados de grafos rápido, confiável e totalmente gerenciado que facilita a criação e a execução de aplicações que trabalham com conjuntos de dados altamente conectados. O mecanismo de banco de dados de grafos especificamente projetado e de alta performance armazena bilhões de relacionamentos de maneira ideal e consulta gráficos com latência de poucos milissegundos. Para obter mais informações, consulte o Manual do usuário do Neptune.

O conector do Neptune no Amazon Athena permite que o Athena se comunique com a instância de banco de dados de grafos do Neptune, tornando seus dados de grafos do Neptune acessíveis para consultas SQL.

Se você tiver o Lake Formation habilitado em sua conta, o perfil do IAM para seu conector Lambda federado para Athena que você implantou no AWS Serverless Application Repository deve ter acesso de leitura ao AWS Glue Data Catalog no Lake Formation.

Pré-requisitos

O uso do conector Neptune requer as três etapas a seguir.

- Configuração de um cluster do Neptune
- Configuração de um AWS Glue Data Catalog
- Implantação do conector em sua Conta da AWS. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados. Para obter detalhes adicionais específicos sobre a implantação

do conector Neptune, consulte Implantação do conector Neptune no Amazon Athena em GitHub.com.

Limitações

No momento, o conector Neptune tem a seguinte limitação.

- A projeção de colunas, incluindo a chave primária (ID), não é compatível.

Configuração de um cluster do Neptune

Se você não tiver um cluster do Amazon Neptune e um conjunto de dados de gráfico de propriedades existentes que gostaria de usar, você deverá configurar um.

Verifique se você tem um gateway da Internet e um gateway NAT na VPC que hospeda seu cluster Neptune. As sub-redes privadas que a função do Lambda do conector Neptune usa devem ter uma rota para a Internet por meio desse gateway NAT. A função do Lambda do conector Neptune usa o gateway NAT para se comunicar com o AWS Glue.

Para obter instruções sobre como configurar um novo cluster do Neptune e carregá-lo com um conjunto de dados de amostra, consulte Exemplo de configuração do cluster Neptune em GitHub.com.

Configuração de um AWS Glue Data Catalog

Diferentemente dos armazenamentos de dados relacionais tradicionais, os nós e bordas do banco de dados gráfico do Neptune não usam um esquema definido. Cada entrada pode ter diferentes campos e tipos de dados. No entanto, como o conector Neptune recupera metadados do AWS Glue Data Catalog, você deve criar um banco de dados do AWS Glue que tenha tabelas com o esquema necessário. Depois de criar o banco de dados e as tabelas do AWS Glue, o conector poderá preencher a lista de tabelas disponíveis para consulta no Athena.

Habilitar a correspondência de colunas sem distinção entre maiúsculas e minúsculas

Para resolver os nomes das colunas da tabela do Neptune com a capitalização correta, mesmo quando os nomes das colunas estão em minúsculas no AWS Glue, é possível configurar o conector do Neptune para correspondência sem distinção entre maiúsculas e minúsculas.

Para habilitar esse atributo, defina a variável de ambiente da função do Lambda do conector Neptune `enable_caseinsensitivematch` como `true`.

Especificar o parâmetro de tabela glabel do AWS Glue para nomes de tabelas com distinção entre maiúsculas e minúsculas

Como o AWS Glue é compatível apenas com nomes de tabelas em letras minúsculas, é importante especificar o parâmetro `glabel` da tabela do AWS Glue ao criar uma tabela do AWS Glue para o Neptune e que a tabela do Neptune inclua distinção entre maiúsculas e minúsculas.

Na definição da tabela do AWS Glue, inclua o parâmetro `glabel` e defina seu valor para o nome da tabela com a capitalização original. Isso garante que a capitalização correta seja preservada quando o AWS Glue interagir com a tabela do Neptune. O exemplo a seguir define o valor de `glabel` para o nome de tabela `Airport`.

```
glabel = Airport
```

Table properties (3)

Key	Value
separatorChar	,
componenttype	vertex
glabel	Airport

Para obter mais informações sobre como configurar um AWS Glue Data Catalog para trabalhar com o Neptune, consulte Set up AWS Glue Catalog em GitHub.com.

Performance

O conector do Athena para o Neptune realiza a passagem direta de predicados para diminuir os dados examinados pela consulta. No entanto, os predicados que usam a chave primária conduzem a falhas na consulta. As cláusulas `LIMIT` reduzem a quantidade de dados verificados, mas se você não fornecer um predicado, deverá aguardar que as consultas `SELECT` com uma cláusula `LIMIT` verifiquem, no mínimo, 16 MB de dados. O conector Neptune é resiliente ao controle de utilização devido à simultaneidade.

Consultas de passagem

O conector Neptune é compatível com consultas de passagem. Você pode usar esse atributo para executar consultas Gremlin em gráficos de propriedades e executar consultas SPARQL em dados RDF.

Para usar consultas de passagem com o Neptune, use a seguinte sintaxe:

```
SELECT * FROM TABLE(
        system.query(
            DATABASE => 'database_name',
            COLLECTION => 'collection_name',
            QUERY => 'query_string'
        ))
```

O exemplo a seguir filtra a consulta de passagem do Neptune para aeroportos com o código ATL.

```
SELECT * FROM TABLE(
        system.query(
            DATABASE => 'graph-database',
            COLLECTION => 'airport',
            QUERY => 'g.V().has(\"airport\", \"code\", \"ATL\").valueMap()',
        ))
```

Recursos adicionais do

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o OpenSearch

OpenSearch Service

O conector do OpenSearch no Amazon Athena permite que o Amazon Athena se comunique com suas instâncias do OpenSearch para que você possa usar o SQL para consultar os dados do OpenSearch.

> **Note**
>
> Devido a um problema conhecido, o conector do OpenSearch não pode ser usado com uma VPC.

Se você tiver o Lake Formation habilitado em sua conta, o perfil do IAM para seu conector Lambda federado para Athena que você implantou no AWS Serverless Application Repository deve ter acesso de leitura ao AWS Glue Data Catalog no Lake Formation.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Termos

Os termos a seguir estão relacionados ao conector OpenSearch.

- Domínio: um nome que esse conector associa ao endpoint da sua instância do OpenSearch. O domínio também é usado como nome do banco de dados. Para instâncias do OpenSearch definidas no Amazon OpenSearch Service, o domínio é detectável automaticamente. Para outras instâncias, você deve fornecer um mapeamento entre o nome de domínio e o endpoint.
- Índice: uma tabela de banco de dados definida em sua instância do OpenSearch.
- Mapeamento: se um índice for uma tabela de banco de dados, o mapeamento será seu esquema (ou seja, as definições de seus campos e atributos).

  Esse conector oferece suporte à recuperação de metadados da instância do OpenSearch e do AWS Glue Data Catalog. Se o conector encontrar um banco de dados e tabela do AWS Glue que correspondam aos nomes de domínio e índice do OpenSearch, o conector tentará usá-los para definição de esquema. Recomendamos que você crie sua tabela do AWS Glue para que seja um superconjunto de todos os campos em seu índice do OpenSearch.
- Documento: um registro em uma tabela do banco de dados.
- Fluxo de dados: dados baseados em tempo que são compostos por vários índices de apoio. Para obter mais informações, consulte Fluxos de dados na documentação do OpenSearch e Conceitos básicos de fluxos de dados no Guia do desenvolvedor do Amazon OpenSearch Service.

> **Note**
>
> Como os índices do fluxo de dados são criados e gerenciados internamente pelo OpenSearch, o conector escolhe o mapeamento do esquema com base no primeiro índice disponível. Por isso, é altamente recomendável configurar uma tabela do AWS Glue como fonte suplementar de metadados. Para ter mais informações, consulte Configuração de bancos de dados e tabelas no AWS Glue.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector OpenSearch.

- spill_bucket: especifica o bucket do Amazon S3 para dados que excedem os limites da função do Lambda.
- spill_prefix: (opcional) assume como padrão uma subpasta no `spill_bucket` especificado chamado `athena-federation-spill`. Recomendamos que você configure um ciclo de vida de armazenamento do Amazon S3 neste local para excluir derramamentos anteriores a um número predeterminado de dias ou horas.
- spill_put_request_headers: (opcional) um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.
- kms_key_id: (opcional) por padrão, todos os dados transmitidos para o Amazon S3 são criptografados usando o modo de criptografia autenticado AES-GCM e uma chave gerada aleatoriamente. Para que sua função do Lambda use chaves de criptografia mais fortes geradas pelo KMS, como `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, é possível especificar um ID de chave do KMS.
- disable_spill_encryption: (opcional) quando definido como `True`, desativa a criptografia do derramamento. É padronizado como `False`, para que os dados transmitidos para o S3 sejam criptografados usando o AES-GCM — usando uma chave gerada aleatoriamente ou o KMS para gerar chaves. Desativar a criptografia do derramamento pode melhorar a performance, especialmente se o local do derramamento usar criptografia no lado do servidor.
- disable_glue: (opcional) se estiver presente e definido como verdadeiro, o conector não tentará recuperar metadados complementares do AWS Glue.
- query_timeout_cluster: o tempo limite, em segundos, para consultas de integridade do cluster usadas na geração de verificações paralelas.
- query_timeout_search: o tempo limite, em segundos, para consultas de pesquisa usadas na recuperação de documentos de um índice.
- auto_discover_endpoint: booleano. O padrão é `true`. Quando você usar o Amazon OpenSearch Service e definir esse parâmetro como verdadeiro, o conector poderá descobrir automaticamente seus domínios e endpoints chamando as operações de API de descrição ou lista apropriadas no OpenSearch Service. Para qualquer outro tipo de instância do OpenSearch (por exemplo, auto-hospedada), você deverá especificar os endpoints de domínio associados na variável

`domain_mapping`. Se `auto_discover_endpoint=true`, o conector usará credenciais da AWS para se autenticar no OpenSearch Service. Caso contrário, o conector recuperará as credenciais de nome de usuário e senha do AWS Secrets Manager por meio da variável `domain_mapping`.

- domain_mapping: usado somente quando `auto_discover_endpoint` for definido como falso, e estabelece o mapeamento entre nomes de domínio e seus endpoints associados. A variável `domain_mapping` pode acomodar vários endpoints do OpenSearch no formato a seguir:

```
domain1=endpoint1,domain2=endpoint2,domain3=endpoint3,...
```

Para fins de autenticação em um endpoint do OpenSearch, o conector oferece suporte a strings de substituição injetadas usando o formato `${SecretName}:` com nome de usuário e senha recuperados do AWS Secrets Manager. Os dois pontos (:) no final da expressão servem como separador do resto do endpoint.

> **Important**
>
> Como prática recomendada de segurança, não use credenciais codificadas em suas variáveis de ambiente ou strings de conexão. Para obter informações sobre como mover seus segredos codificados para o AWS Secrets Manager, consulte Mover segredos codificados para o AWS Secrets Manager no Guia do usuário do AWS Secrets Manager.

O exemplo a seguir usa o segredo `opensearch-creds`.

```
movies=https://${opensearch-creds}:search-movies-ne...qu.us-east-1.es.amazonaws.com
```

Em runtime, `${opensearch-creds}` será renderizado como nome de usuário e senha, como no exemplo a seguir.

```
movies=https://myusername@mypassword:search-movies-ne...qu.us-east-1.es.amazonaws.com
```

No parâmetro `domain_mapping`, cada par de domínio e endpoint pode usar um segredo diferente. O segredo em si deve ser especificado no formato *`nome_do_usuário`*@*`senha`*. Embora a senha possa conter sinais @ incorporados, o primeiro @ serve como um separador de *`nome_do_usuário`*.

Também é importante observar que a vírgula (,) e o sinal de igual (=) são usados por esse conector como separadores para os pares de endpoints do domínio. Por esse motivo, você não deve usá-los em nenhum lugar do segredo armazenado.

Configuração de bancos de dados e tabelas no AWS Glue

O conector obtém informações de metadados usando o AWS Glue ou o OpenSearch. É possível configurar uma tabela do AWS Glue como fonte de definição de metadados complementar. Para ativar esse recurso, defina um banco de dados do AWS Glue e uma tabela que correspondam ao domínio e ao índice da fonte que você está complementando. O conector também pode aproveitar as definições de metadados armazenadas na instância do OpenSearch recuperando o mapeamento para o índice especificado.

Definição de metadados para matrizes no OpenSearch

O OpenSearch não tem um tipo de dados de matriz dedicado. Qualquer campo pode conter zero ou mais valores, desde que sejam do mesmo tipo de dados. Se você quiser usar o OpenSearch como sua fonte de definição de metadados, você deve definir uma propriedade `_meta` para todos os índices usados com o Athena para os campos que devam ser considerados uma lista ou matriz. Se você não conseguir concluir essa etapa, as consultas retornarão somente o primeiro elemento no campo da lista. Quando você especificar a propriedade `_meta`, os nomes dos campos deverão ser totalmente qualificados para estruturas JSON aninhadas (por exemplo, `address.street`, em que `street` é um campo aninhado dentro de uma estrutura `address`).

O exemplo a seguir define as listas `actor` e `genre` na tabela `movies`.

```
PUT movies/_mapping
{
  "_meta": {
    "actor": "list",
    "genre": "list"
  }
}
```

Tipos de dados

O conector OpenSearch pode extrair definições de metadados do AWS Glue ou da instância do OpenSearch. O conector usa o mapeamento na tabela a seguir para converter as definições em tipos de dados do Apache Arrow, incluindo os pontos indicados na seção a seguir.

OpenSearch	Apache Arrow	AWS Glue
text, keyword, binary	VARCHAR	string
longo	BIGINT	bigint
scaled_float	BIGINT	SCALED_FLOAT(...)
inteiro	INT	int
curto	SMALLINT	smallint
byte	TINYINT	tinyint
double	FLOAT8	double
float, half_float	FLOAT4	float
boolean	BIT	boolean
date, date_nanos	DATEMILLI	timestamp
Estrutura JSON	STRUCT	STRUCT
_meta (para obter mais informações, consulte a seção Definição de metadados para matrizes no OpenSearch.)	LIST	ARRAY

Notas sobre tipos de dados

- No momento, o conector oferece suporte somente os tipos de dados do OpenSearch e do AWS Glue listados na tabela anterior.
- Um `scaled_float` é um número de ponto flutuante escalado por um fator fixo de escala dupla e representado como um `BIGINT` no Apache Arrow. Por exemplo, 0,756 com um fator de escala de 100 é arredondado para 76.
- Para definir um `scaled_float` no AWS Glue, você deverá selecionar o tipo de coluna `array` e declarar o campo usando o formato SCALED_FLOAT(*`fator_de_escala`*).

Os exemplos a seguir são válidos:

```
SCALED_FLOAT(10.51)
SCALED_FLOAT(100)
SCALED_FLOAT(100.0)
```

Os exemplos a seguir não são válidos:

```
SCALED_FLOAT(10.)
SCALED_FLOAT(.5)
```

- Ao converter de `date_nanos` para `DATEMILLI`, os nanossegundos serão arredondados para o milissegundo mais próximo. Valores válidos para `date` e `date_nanos` incluem, entre outros, os seguintes formatos:

```
"2020-05-18T10:15:30.123456789"
"2020-05-15T06:50:01.123Z"
"2020-05-15T06:49:30.123-05:00"
1589525370001 (epoch milliseconds)
```

- Um `binary` do OpenSearch é uma representação de string de um valor binário codificado usando `Base64`, e é convertido em um `VARCHAR`.

Execução de consultas de SQL

Veja a seguir exemplos de consultas DDL que você pode usar com esse conector. Nos exemplos, *`nome_da_função`* corresponde ao nome da função do Lambda *`domínio`* é o nome do domínio que você deseja consultar e *`índice`* é o nome do seu índice.

```
SHOW DATABASES in `lambda:function_name`
```

```
SHOW TABLES in `lambda:function_name`.domain
```

```
DESCRIBE `lambda:function_name`.domain.index
```

Performance

O conector OpenSearch do Athena oferece suporte a verificações paralelas baseadas em fragmentos. O conector usa as informações de integridade do cluster recuperadas da instância do OpenSearch para gerar várias solicitações para uma consulta de pesquisa de documentos. As solicitações são divididas para cada fragmento e executadas simultaneamente.

O conector também reduz os predicados como parte de suas consultas de pesquisa de documentos. O exemplo de consulta e predicado a seguir mostra como o conector usa a redução de predicado.

Consulta

```
SELECT * FROM "lambda:elasticsearch".movies.movies
WHERE year >= 1955 AND year <= 1962 OR year = 1996
```

Predicado

```
(_exists_:year) AND year:([1955 TO 1962] OR 1996)
```

Consultas de passagem

O conector OpenSearch é compatível com consultas de passagem e usa a linguagem Query DSL. Para obter mais informações sobre consultas com o Query DSL, consulte Query DSL na documentação do Elasticsearch ou Query DSL na documentação do OpenSearch.

Para usar as consultas de passagem com o conector OpenSearch, use a seguinte sintaxe:

```
SELECT * FROM TABLE(
        system.query(
            schema => 'schema_name',
            index => 'index_name',
            query => "{query_string}"
        ))
```

O seguinte exemplo de consulta de passagem do OpenSearch filtra funcionários com status de emprego ativo no índice `employee` do esquema `default`.

```
SELECT * FROM TABLE(
        system.query(
            schema => 'default',
```

```
            index => 'employee',
            query => "{ ''bool'':{''filter'':{''term'':{''status'': ''active''}}}}"
        ))
```

Recursos adicionais do

- Para ver um artigo sobre o uso do conector Amazon Athena OpenSearch para consultar dados no Amazon OpenSearch Service e no Amazon S3 em uma única consulta, consulte Consulte dados no Amazon OpenSearch Service usando SQL do Amazon Athena no AWSBlog de Big Data.
- Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para Oracle

O conector do Amazon Athena para Oracle permite que o Amazon Athena execute consultas SQL em dados armazenados no Oracle em execução on-premises, no Amazon EC2 ou no Amazon RDS. Você também pode usar o conector para consultar dados no Oracle Exadata.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Limitações

- Não há suporte para operações de gravação de DDL.
- Em uma configuração de multiplexador, o prefixo e o bucket de derramamento são compartilhados em todas as instâncias do banco de dados.
- Quaisquer limites relevantes do Lambda. Para obter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do AWS Lambda.

Termos

Os termos a seguir estão relacionados ao conector Oracle.

- Instância do banco de dados: qualquer instância de um banco de dados implantado on-premises, no Amazon EC2 ou no Amazon RDS.

- Manipulador: um manipulador Lambda que acessa sua instância de banco de dados. Um manipulador pode ser para metadados ou para registros de dados.
- Manipulador de metadados: um manipulador Lambda que recupera metadados da sua instância de banco de dados.
- Manipulador de registros: um manipulador Lambda que recupera registros de dados da sua instância de banco de dados.
- Manipulador composto: um manipulador Lambda que recupera tanto metadados quanto registros de dados da sua instância de banco de dados.
- Propriedade ou parâmetro: uma propriedade do banco de dados usada pelos manipuladores para extrair informações do banco de dados. Você configura essas propriedades como variáveis de ambiente do Lambda.
- String de conexão: uma string de texto usada para estabelecer uma conexão com uma instância de banco de dados.
- Catálogo: um catálogo não AWS Glue registrado no Athena que é um prefixo obrigatório para a propriedade `connection_string`.
- Manipulador de multiplexação: um manipulador Lambda que pode aceitar e usar várias conexões de banco de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector Oracle.

String de conexão

Use uma string de conexão JDBC no seguinte formato para se conectar a uma instância de banco de dados.

```
oracle://${jdbc_connection_string}
```

> **Note**
>
> Se sua senha contiver caracteres especiais (por exemplo, `some.password`), coloque-a entre aspas duplas quando passá-la para a string de conexão (por exemplo, `"some.password"`). Não fazê-lo pode resultar em um erro de URL inválido especificado.

Uso de um manipulador de multiplexação

É possível usar um multiplexador para se conectar a várias instâncias de banco de dados com uma única função do Lambda. As solicitações são encaminhadas por nome do catálogo. Use as seguintes classes no Lambda.

Manipulador	Classe
Manipulador composto	`OracleMuxCompositeHandler`
Manipulador de metadados	`OracleMuxMetadataHandler`
Manipulador de registros	`OracleMuxRecordHandler`

Parâmetros do manipulador de multiplexação

Parâmetro	Descrição
`$`*`catalog`*`_connection_string`	Obrigatório. Uma string de conexão de instância de banco de dados. Prefixe a variável de ambiente com o nome do catálogo usado no Athena. Por exemplo, se o catálogo registrado no Athena for `myoraclecatalog` , então o nome da variável de ambiente será `myoraclecatalog_connection_string` .
`default`	Obrigatório. A string de conexão padrão. Essa string é usada quando o catálogo for `lambda:${` *`AWS_LAMBDA_FUNCTION_NAME`* `}`.

As propriedades de exemplo a seguir são para uma função do Lambda MUX do Oracle que ofereça suporte a duas instâncias de banco de dados: `oracle1` (o padrão) e `oracle2`.

Propriedade	Valor
`default`	`oracle://jdbc:oracle:thin:${Test/RDS/Oracle1}@//oracle1.hostname:port/servicename`

Propriedade	Valor
`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`

Fornecimento de credenciais

Para fornecer um nome de usuário e uma senha para seu banco de dados na string de conexão JDBC, é possível usar as propriedades da string de conexão ou o AWS Secrets Manager.

- String de conexão: um nome de usuário e uma senha podem ser especificados como propriedades na string de conexão do JDBC.

  > **Important**
  >
  > Como prática recomendada de segurança, não use credenciais codificadas em suas variáveis de ambiente ou strings de conexão. Para obter informações sobre como mover seus segredos codificados para o AWS Secrets Manager, consulte Mover segredos codificados para o AWS Secrets Manager no Guia do usuário do AWS Secrets Manager.

- AWS Secrets Manager: para usar o recurso Athena Federated Query com o AWS Secrets Manager, a VPC conectada à sua função do Lambda deve ter acesso à Internet ou um endpoint da VPC para se conectar ao Secrets Manager.

  É possível colocar o nome de um segredo no AWS Secrets Manager na sua string de conexão JDBC. O conector substitui o nome secreto pelos valores de `username` e `password` do Secrets Manager.

  Para instâncias de banco de dados do Amazon RDS, esse suporte é totalmente integrado. Se você usa o Amazon RDS, é altamente recomendável usar o AWS Secrets Manager e rotação de credenciais. Se seu banco de dados não usar o Amazon RDS, armazene as credenciais em JSON no seguinte formato:

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

> **Note**
>
> Se sua senha contiver caracteres especiais (por exemplo, `some.password`), coloque-a entre aspas duplas quando armazená-la no Secrets Manager (por exemplo, `"some.password"`). Não fazê-lo pode resultar em um erro de URL inválido especificado.

Exemplo de string de conexão com nome secreto

A string a seguir tem o nome secreto `${Test/RDS/Oracle}`.

```
oracle://jdbc:oracle:thin:${Test/RDS/Oracle}@//hostname:port/servicename
```

O conector usa o nome secreto para recuperar segredos e fornecer o nome de usuário e a senha, como no exemplo a seguir.

```
oracle://jdbc:oracle:thin:username/password@//hostname:port/servicename
```

Atualmente, o conector Oracle reconhece as propriedades `UID` e `PWD` do JDBC.

Uso de um único manipulador de conexão

É possível usar os seguintes metadados de conexão única e manipuladores de registros para se conectar a uma única instância do Oracle.

Tipo de manipulador	Classe
Manipulador composto	`OracleCompositeHandler`
Manipulador de metadados	`OracleMetadataHandler`
Manipulador de registros	`OracleRecordHandler`

Parâmetros do manipulador de conexão única

Parâmetro	Descrição
`default`	Obrigatório. A string de conexão padrão.
`IsFIPSEnabled`	Opcional. Definido como `true` quando o modo FIPS estiver ativado. O padrão é `false`.

Os manipuladores de conexão únicos oferecem suporte a uma instância de banco de dados e devem fornecer um parâmetro de string de conexão `default`. Todas as outras strings de conexão são ignoradas.

O conector oferece suporte a conexões baseadas em SSL a instâncias do Amazon RDS. O suporte é limitado ao protocolo Transport Layer Security (TLS) e à autenticação do servidor pelo cliente. Não há suporte para a autenticação mútua no Amazon RDS. A segunda linha na tabela abaixo mostra a sintaxe para o uso de SSL.

O exemplo de propriedade a seguir é para uma única instância do Oracle com suporte em uma função do Lambda.

Propriedade	Valor
`default`	`oracle://jdbc:oracle:thin:${Test/RDS/Oracle}@//hostname:port/servicename`
	`oracle://jdbc:oracle:thin:${Test/RDS/Oracle}@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS) (HOST=<HOST_NAME>)(PORT=))(CONNECT_DATA=(SID=))(SECURITY=(SSL_SERVER_CERT_DN=)))`

Parâmetros de derramamento

O SDK do Lambda pode derramar dados no Amazon S3. Todas as instâncias do banco de dados acessadas pela mesma função do Lambda derramam no mesmo local.

Parâmetro	Descrição
`spill_bucket`	Obrigatório. Nome do bucket de derramamento.
`spill_prefix`	Obrigatório. Prefixo de chave do bucket de derramamento.
`spill_put_request_headers`	(Opcional) Um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.

Suporte ao tipo de dados

A tabela a seguir mostra os correspondentes tipos de dados do JDBC, do Oracle e do Arrow.

JDBC	Oracle	Arrow
Booleano	boolean	Bit
Inteiro	N/D	Tiny
Short	smallint	Smallint
Inteiro	inteiro	Int
Longo	bigint	Bigint
float	float4	Float4
Double	float8	Float8
Data	date	Data/Dia
Timestamp	timestamp	Date Milli
String	text	Varchar

JDBC	Oracle	Arrow
Bytes	bytes	Varbinary
BigDecimal	numeric(p,s)	Decimal
ARRAY	N/D (ver nota)	Lista

Partições e divisões

As partições são usadas para determinar como gerar divisões para o conector. O Athena constrói uma coluna sintética do tipo `varchar` que representa o esquema de particionamento da tabela para ajudar o conector a gerar divisões. O conector não modifica a definição real da tabela.

Performance

O Oracle oferece suporte a partições nativas. O conector do Athena para o Oracle pode recuperar dados dessas partições em paralelo. Se você quiser consultar conjuntos de dados muito grandes com distribuição uniforme de partições, o particionamento nativo é altamente recomendado.
A seleção de um subconjunto de colunas acelera o runtime da consulta e reduz os dados verificados de forma significativa. O conector Oracle é resiliente ao controle de utilização devido à simultaneidade. No entanto, os tempos de execução das consultas tendem a ser longos.

O conector do Athena para Oracle realiza a passagem direta de predicados para diminuir os dados examinados pela consulta. Predicados simples e expressões complexas são passados diretamente ao conector para reduzir a quantidade de dados examinados e o runtime de execução da consulta.

Predicados

Um predicado é uma expressão na cláusula `WHERE` de uma consulta SQL, que avalia para um valor booleano e filtra as linhas com base em várias condições. O conector do Athena para o Oracle pode combinar essas expressões e passá-las diretamente ao Oracle para melhorar a funcionalidade e reduzir a quantidade de dados examinados.

Os seguintes operadores do conector do Athena para o Oracle são compatíveis com a passagem de predicados:

- Booleanos: E, OU, NÃO
- Igualdade: EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_NULL

- Aritméticos: ADICIONAR, SUBTRAIR, MULTIPLICAR, DIVIDIR, NEGAR
- Outros:LIKE_PATTERN, IN

Exemplo de passagem direta combinada

Para ter recursos aprimorados de consulta, combine os tipos de passagem direta, como no seguinte exemplo:

```
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%');
```

Consultas de passagem

O conector Oracle é compatível com consultas de passagem. As consultas de passagem usam uma função de tabela para enviar sua consulta completa para execução na fonte de dados.

Para usar consultas de passagem com o Oracle, você pode empregar a seguinte sintaxe:

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

O exemplo de consulta a seguir envia uma consulta para uma fonte de dados no Oracle. A consulta seleciona todas as colunas na tabela `customer`.

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

Informações de licença

Ao usar esse conector, você reconhece a inclusão de componentes de terceiros, cuja lista pode ser encontrada no arquivo pom.xml desse conector, e concorda com os termos das respectivas licenças de terceiros fornecidas no arquivo LICENSE.txt em GitHub.com.

Recursos adicionais do

Para obter as informações mais recentes sobre a versão do driver JDBC, consulte o arquivo pom.xml do conector Oracle em GitHub.com.

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o PostgreSQL

O conector PostgreSQL do Amazon Athena permite que o Athena acesse seus bancos de dados PostgreSQL.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Limitações

- Não há suporte para operações de gravação de DDL.
- Em uma configuração de multiplexador, o prefixo e o bucket de derramamento são compartilhados em todas as instâncias do banco de dados.
- Quaisquer limites relevantes do Lambda. Para obter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do AWS Lambda.

Termos

Os termos a seguir estão relacionados ao conector PostgreSQL.

- Instância do banco de dados: qualquer instância de um banco de dados implantado on-premises, no Amazon EC2 ou no Amazon RDS.
- Manipulador: um manipulador Lambda que acessa sua instância de banco de dados. Um manipulador pode ser para metadados ou para registros de dados.
- Manipulador de metadados: um manipulador Lambda que recupera metadados da sua instância de banco de dados.
- Manipulador de registros: um manipulador Lambda que recupera registros de dados da sua instância de banco de dados.

- Manipulador composto: um manipulador Lambda que recupera tanto metadados quanto registros de dados da sua instância de banco de dados.
- Propriedade ou parâmetro: uma propriedade do banco de dados usada pelos manipuladores para extrair informações do banco de dados. Você configura essas propriedades como variáveis de ambiente do Lambda.
- String de conexão: uma string de texto usada para estabelecer uma conexão com uma instância de banco de dados.
- Catálogo: um catálogo não AWS Glue registrado no Athena que é um prefixo obrigatório para a propriedade `connection_string`.
- Manipulador de multiplexação: um manipulador Lambda que pode aceitar e usar várias conexões de banco de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector PostgreSQL.

String de conexão

Use uma string de conexão JDBC no seguinte formato para se conectar a uma instância de banco de dados.

```
postgres://${jdbc_connection_string}
```

Uso de um manipulador de multiplexação

É possível usar um multiplexador para se conectar a várias instâncias de banco de dados com uma única função do Lambda. As solicitações são encaminhadas por nome do catálogo. Use as seguintes classes no Lambda.

Manipulador	Classe
Manipulador composto	`PostGreSqlMuxCompositeHandler`
Manipulador de metadados	`PostGreSqlMuxMetadataHandler`
Manipulador de registros	`PostGreSqlMuxRecordHandler`

Parâmetros do manipulador de multiplexação

Parâmetro	Descrição
`$catalog_connection_string`	Obrigatório. Uma string de conexão de instância de banco de dados. Prefixe a variável de ambiente com o nome do catálogo usado no Athena. Por exemplo, se o catálogo registrado no Athena for `mypostgrescatalog` , então o nome da variável de ambiente será `mypostgrescatalog_connection_string` .
`default`	Obrigatório. A string de conexão padrão. Essa string é usada quando o catálogo for `lambda:${ AWS_LAMBDA_FUNCTION_NAME }`.

As propriedades de exemplo a seguir são para uma função do Lambda MUX do PostgreSQL que ofereça suporte a duas instâncias de banco de dados: `postgres1` (o padrão) e `postgres2`.

Propriedade	Valor
`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`

Fornecimento de credenciais

Para fornecer um nome de usuário e uma senha para seu banco de dados na string de conexão JDBC, é possível usar as propriedades da string de conexão ou o AWS Secrets Manager.

- String de conexão: um nome de usuário e uma senha podem ser especificados como propriedades na string de conexão do JDBC.

  > **Important**
  >
  > Como prática recomendada de segurança, não use credenciais codificadas em suas variáveis de ambiente ou strings de conexão. Para obter informações sobre como mover seus segredos codificados para o AWS Secrets Manager, consulte Mover segredos codificados para o AWS Secrets Manager no Guia do usuário do AWS Secrets Manager.

- AWS Secrets Manager: para usar o recurso Athena Federated Query com o AWS Secrets Manager, a VPC conectada à sua função do Lambda deve ter acesso à Internet ou um endpoint da VPC para se conectar ao Secrets Manager.

  É possível colocar o nome de um segredo no AWS Secrets Manager na sua string de conexão JDBC. O conector substitui o nome secreto pelos valores de `username` e `password` do Secrets Manager.

  Para instâncias de banco de dados do Amazon RDS, esse suporte é totalmente integrado. Se você usa o Amazon RDS, é altamente recomendável usar o AWS Secrets Manager e rotação de credenciais. Se seu banco de dados não usar o Amazon RDS, armazene as credenciais em JSON no seguinte formato:

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

Exemplo de string de conexão com nome secreto

A string a seguir tem o nome secreto `${Test/RDS/PostGres1}`.

```
postgres://jdbc:postgresql://postgres1.host:5432/default?...&${Test/RDS/PostGres1}&...
```

O conector usa o nome secreto para recuperar segredos e fornecer o nome de usuário e a senha, como no exemplo a seguir.

```
postgres://jdbc:postgresql://postgres1.host:5432/
default?...&user=sample2&password=sample2&...
```

Atualmente, o conector PostgreSQL reconhece as propriedades `user` e `password` do JDBC.

Habilitação do SSL

Para oferecer suporte a SSL em sua conexão PostgreSQL, acrescente o seguinte à string de conexão:

```
&sslmode=verify-ca&sslfactory=org.postgresql.ssl.DefaultJavaSSLFactory
```

Exemplo

O exemplo de string de conexão a seguir não usa SSL.

```
postgres://jdbc:postgresql://example-asdf-aurora-postgres-endpoint:5432/asdf?
user=someuser&password=somepassword
```

Para habilitar o SSL, modifique a string da forma mostrada a seguir.

```
postgres://jdbc:postgresql://example-asdf-aurora-postgres-
endpoint:5432/asdf?user=someuser&password=somepassword&sslmode=verify-
ca&sslfactory=org.postgresql.ssl.DefaultJavaSSLFactory
```

Uso de um único manipulador de conexão

É possível usar os seguintes metadados de conexão única e manipuladores de registros para se conectar a uma única instância do PostgreSQL.

Tipo de manipulador	Classe
Manipulador composto	`PostGreSqlCompositeHandler`
Manipulador de metadados	`PostGreSqlMetadataHandler`
Manipulador de registros	`PostGreSqlRecordHandler`

Parâmetros do manipulador de conexão única

Parâmetro	Descrição
`default`	Obrigatório. A string de conexão padrão.

Os manipuladores de conexão únicos oferecem suporte a uma instância de banco de dados e devem fornecer um parâmetro de string de conexão `default`. Todas as outras strings de conexão são ignoradas.

O exemplo de propriedade a seguir é para uma única instância do PostgreSQL com suporte em uma função do Lambda.

Propriedade	Valor
`default`	`postgres://jdbc:postgresql://postgres1.host:5432/default?secret=${Test/RDS/PostgreSQL1}`

Parâmetros de derramamento

O SDK do Lambda pode derramar dados no Amazon S3. Todas as instâncias do banco de dados acessadas pela mesma função do Lambda derramam no mesmo local.

Parâmetro	Descrição
`spill_bucket`	Obrigatório. Nome do bucket de derramamento.
`spill_prefix`	Obrigatório. Prefixo de chave do bucket de derramamento.
`spill_put_request_headers`	(Opcional) Um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.

Suporte ao tipo de dados

A tabela a seguir mostra os correspondentes tipos de dados do JDBC, do PostgreSQL e do Arrow.

JDBC	PostGreSQL	Arrow
Booleano	Booleano	Bit
Inteiro	N/D	Tiny
Short	smallint	Smallint
Inteiro	inteiro	Int
Longo	bigint	Bigint
float	float4	Float4
Double	float8	Float8
Data	date	Data/Dia
Timestamp	timestamp	Date Milli
String	text	Varchar
Bytes	bytes	Varbinary
BigDecimal	numeric(p,s)	Decimal
ARRAY	N/D (ver nota)	Lista

Note

Há suporte para o tipo ARRAY no conector PostgreSQL com as seguintes restrições: não há suporte para matrizes multidimensionais (`<data_type>[][]` ou matrizes aninhadas). Colunas com tipos de dados ARRAY sem suporte serão convertidas em uma matriz de elementos de string (`array<varchar>`).

Partições e divisões

As partições são usadas para determinar como gerar divisões para o conector. O Athena constrói uma coluna sintética do tipo `varchar` que representa o esquema de particionamento da tabela para ajudar o conector a gerar divisões. O conector não modifica a definição real da tabela.

Performance

O PostgreSQL oferece suporte a partições nativas. O conector do Athena para o PostgreSQL pode recuperar dados dessas partições em paralelo. Se você quiser consultar conjuntos de dados muito grandes com distribuição uniforme de partições, o particionamento nativo é altamente recomendado.

O conector do Athena para o PostgreSQL executa a passagem direta de predicados para diminuir os dados examinados pela consulta. Cláusulas `LIMIT`, predicados simples e expressões complexas são passados diretamente ao conector para reduzir a quantidade de dados examinados e o runtime de execução da consulta. Porém, selecionar um subconjunto de colunas resulta, algumas vezes, em um runtime mais longo.

Cláusulas LIMIT

Uma instrução `LIMIT N` reduz os dados examinados pela consulta. Com a passagem direta de `LIMIT N`, o conector só retorna `N` linhas para o Athena.

Predicados

Um predicado é uma expressão na cláusula `WHERE` de uma consulta SQL, que avalia para um valor booleano e filtra as linhas com base em várias condições. O conector do Athena para o PostgreSQL pode combinar essas expressões e passá-las diretamente ao PostgreSQL para melhorar a funcionalidade e reduzir a quantidade de dados examinados.

Os seguintes operadores do conector do Athena para o PostgreSQL são compatíveis com a passagem direta de predicados:

- Booleanos: E, OU, NÃO
- Igualdade: EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Aritméticos: ADICIONAR, SUBTRAIR, MULTIPLICAR, DIVIDIR, MÓDULO, NEGAR
- Outros:LIKE_PATTERN, IN

Exemplo de passagem direta combinada

Para ter recursos aprimorados de consulta, combine os tipos de passagem direta, como no seguinte exemplo:

```
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;
```

Consultas de passagem

O conector PostgreSQL é compatível com consultas de passagem. As consultas de passagem usam uma função de tabela para enviar sua consulta completa para execução na fonte de dados.

Para usar consultas de passagem com o PostgreSQL, você pode empregar a seguinte sintaxe:

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

O exemplo de consulta a seguir envia uma consulta para uma fonte de dados no PostgreSQL. A consulta seleciona todas as colunas na tabela `customer`, limitando os resultados a 10.

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

Recursos adicionais do

Para obter as informações mais recentes sobre a versão do driver JDBC, consulte o arquivo pom.xml do conector PostgreSQL em GitHub.com.

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o Redis

O conector do Redis no Amazon Athena permite que o Amazon Athena se comunique com as instâncias do Redis para que você possa consultar os dados do Redis com SQL. É possível usar o AWS Glue Data Catalog para mapear os pares de chave-valor do Redis em tabelas virtuais.

Diferente dos armazenamentos de dados relacionais tradicionais, o Redis não tem o conceito de tabela ou coluna. Em vez disso, o Redis oferece padrões de acesso de valores-chave em que a chave é essencialmente uma `string` e o valor é uma `string`, `z-set` ou hmap.

É possível usar o AWS Glue Data Catalog para criar um esquema e configurar tabelas virtuais. As propriedades especiais da tabela informam ao conector do Redis do Athena como mapear suas chaves e valores do Redis em uma tabela. Para obter mais informações, consulte Configuração de bancos de dados e tabelas no AWS Glue adiante neste documento.

Se você tiver o Lake Formation habilitado em sua conta, o perfil do IAM para seu conector Lambda federado para Athena que você implantou no AWS Serverless Application Repository deve ter acesso de leitura ao AWS Glue Data Catalog no Lake Formation.

O conector do Amazon Athena Redis é compatível com o Amazon MemoryDB para Redis e Amazon ElastiCache para Redis.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.
- Configura uma VPC e um grupo de segurança antes de usar esse conector. Para ter mais informações, consulte Criar uma VPC para um conector de origem de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector do Redis.

- spill_bucket: especifica o bucket do Amazon S3 para dados que excedem os limites da função do Lambda.
- spill_prefix: (opcional) assume como padrão uma subpasta no `spill_bucket` especificado chamado `athena-federation-spill`. Recomendamos que você configure um ciclo de vida de

armazenamento do Amazon S3 neste local para excluir derramamentos anteriores a um número predeterminado de dias ou horas.

- spill_put_request_headers: (opcional) um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.
- kms_key_id: (opcional) por padrão, todos os dados transmitidos para o Amazon S3 são criptografados usando o modo de criptografia autenticado AES-GCM e uma chave gerada aleatoriamente. Para que sua função do Lambda use chaves de criptografia mais fortes geradas pelo KMS, como `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, é possível especificar um ID de chave do KMS.
- disable_spill_encryption: (opcional) quando definido como `True`, desativa a criptografia do derramamento. É padronizado como `False`, para que os dados transmitidos para o S3 sejam criptografados usando o AES-GCM — usando uma chave gerada aleatoriamente ou o KMS para gerar chaves. Desativar a criptografia do derramamento pode melhorar a performance, especialmente se o local do derramamento usar criptografia no lado do servidor.
- glue_catalog: (opcional) use essa opção para especificar um catálogo do AWS Glue entre contas. Por padrão, o conector tenta obter metadados de sua própria conta do AWS Glue.

Configuração de bancos de dados e tabelas no AWS Glue

Para habilitar uma tabela do AWS Glue para uso com o Redis, é possível definir as seguintes propriedades da tabela na tabela: `redis-endpoint`, `redis-value-type` e `redis-keys-zset` ou `redis-key-prefix`.

Além disso, qualquer banco de dados do AWS Glue que contenha tabelas do Redis deve ter um `redis-db-flag` na propriedade URI do banco de dados. Para definir a propriedade da URI `redis-db-flag`, use o console do AWS Glue para editar o banco de dados.

A lista a seguir descreve as propriedades da tabela.

- redis-endpoint: (obrigatório) o *`hostname`*`:`*`port`*`:`*`senha`* do servidor Redis que contém dados para essa tabela (por exemplo, `athena-federation-demo.cache.amazonaws.com:6379`). Como alternativa, é possível armazenar o endpoint, ou parte do endpoint, em AWS Secrets Manager, usando ${*`Secret_Name`*} como o valor da propriedade da tabela.

> **Note**
>
> Para usar o recurso de consulta federada do Athena com o AWS Secrets Manager, a VPC conectada à sua função do Lambda deve ter acesso à Internet ou um endpoint da VPC para se conectar ao Secrets Manager.

- redis-keys-zset: (necessário se `redis-key-prefix` não for usado) uma lista separada por vírgulas de chaves cujo valor é um zset (por exemplo, `active-orders,pending-orders`). Cada um dos valores no zset é tratado como uma chave que faz parte da tabela. A propriedade `redis-keys-zset` ou a propriedade `redis-key-prefix` devem ser definidas.
- redis-key-prefix: (necessário se `redis-keys-zset` não for usado) uma lista separada por vírgulas de prefixos de chave para verificar os valores na tabela (por exemplo, `accounts-*,acct-`). A propriedade `redis-key-prefix` ou a propriedade `redis-keys-zset` devem ser definidas.
- redis-value-type: (obrigatório) define como os valores das chaves definidos por `redis-key-prefix` ou `redis-keys-zset` são mapeados para sua tabela. Um literal é mapeado para uma única coluna. Um zset também é mapeado para uma única coluna, mas cada chave pode armazenar muitas linhas. Um hash permite que cada chave seja uma linha com várias colunas (por exemplo, um hash, literal ou zset).
- redis-ssl-flag: (opcional) quando for `True`, cria uma conexão Redis que usa SSL/TLS. O padrão é `False`.
- redis-cluster-flag: (opcional) quando for `True`, permite o suporte para instâncias Redis em cluster. O padrão é `False`.
- redis-db-number: (opcional) aplica-se somente a instâncias autônomas sem cluster. Defina esse número (por exemplo, 1, 2 ou 3) para ler de um banco de dados Redis não padrão. O padrão é o banco de dados lógico Redis 0. Esse número não se refere a um banco de dados no Athena ou no AWS Glue, mas a um banco de dados lógico Redis. Para obter mais informações, consulte Índice SELECT na documentação do Redis.

Tipos de dados

O conector Redis é compatível com os seguintes tipos de dados: Os fluxos do Redis não são compatíveis.

- String

- Hash
- Conjunto classificados (ZSet)

Todos os valores do Redis são recuperados como tipo de dados `string`. Em seguida, eles são convertidos em um dos seguintes tipos de dados do Apache Arrow, com base em como suas tabelas são definidas no AWS Glue Data Catalog.

Tipo de dados do AWS Glue	Tipo de dados Apache Arrow
int	INT
string	VARCHAR
bigint	BIGINT
double	FLOAT8
float	FLOAT4
smallint	SMALLINT
tinyint	TINYINT
boolean	BIT
binary	VARBINARY

Permissões obrigatórias

Os detalhes completos sobre as políticas do IAM exigidas por esse conector podem ser encontrados na seção `Policies` do arquivo athena-redis.yaml. A lista a seguir resume as permissões necessárias.

- Acesso de gravação do Amazon S3: o conector requer acesso de gravação a um local no Amazon S3 para mostrar resultados de grandes consultas.
- Athena GetQueryExecution: o conector usa esta permissão para falhar rapidamente quando a consulta upstream do Athena é encerrada.

- AWS Glue Data Catalog: o conector do Redis requer acesso somente de leitura ao AWS Glue Data Catalog para obter informações do esquema.
- CloudWatch Logs: o conector requer acesso ao CloudWatch Logs para armazenar registros.
- Acesso de leitura do AWS Secrets Manager: se você optar por armazenar os detalhes do endpoint do Redis no Secrets Manager, deverá conceder ao conector acesso a esses segredos.
- Acesso à VPC: o conector exige a capacidade de conectar e desconectar interfaces à sua VPC para que possa se conectar a ela e se comunicar com suas instâncias do Redis.

Performance

O conector do Redis do Athena tenta paralelizar as consultas com sua instância do Redis de acordo com o tipo de tabela que você definiu (por exemplo, chaves zset ou chaves de prefixo).

O conector do Athena para Redis realiza a passagem direta de predicados para diminuir os dados examinados pela consulta. No entanto, as consultas que contêm um predicado na chave primária apresentam falhas relacionadas ao tempo limite. As cláusulas `LIMIT` reduzem a quantidade de dados verificados, mas se você não fornecer um predicado, deverá aguardar que as consultas `SELECT` com uma cláusula `LIMIT` verifiquem, no mínimo, 16 MB de dados. O conector Redis é resiliente ao controle de utilização devido à simultaneidade.

Informações de licença

O projeto do conector do Redis do Amazon Athena é licenciado sob a Licença Apache-2.0.

Recursos adicionais do

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o Redshift

O conector Redshift do Amazon Athena permite que o Amazon Athena acesse bancos de dados do Amazon Redshift e do Amazon Redshift sem servidor, incluindo visualizações do RedShift sem servidor. Você pode se conectar a qualquer serviço usando as configurações da string de conexão JDBC descritas nesta página.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de

dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Limitações

- Não há suporte para operações de gravação de DDL.
- Em uma configuração de multiplexador, o prefixo e o bucket de derramamento são compartilhados em todas as instâncias do banco de dados.
- Quaisquer limites relevantes do Lambda. Para obter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do AWS Lambda.
- Como o Redshift não oferece suporte a partições externas, todos os dados especificados por uma consulta são recuperados sempre.

Termos

Os termos a seguir estão relacionados ao conector Redshift.

- Instância do banco de dados: qualquer instância de um banco de dados implantado on-premises, no Amazon EC2 ou no Amazon RDS.
- Manipulador: um manipulador Lambda que acessa sua instância de banco de dados. Um manipulador pode ser para metadados ou para registros de dados.
- Manipulador de metadados: um manipulador Lambda que recupera metadados da sua instância de banco de dados.
- Manipulador de registros: um manipulador Lambda que recupera registros de dados da sua instância de banco de dados.
- Manipulador composto: um manipulador Lambda que recupera tanto metadados quanto registros de dados da sua instância de banco de dados.
- Propriedade ou parâmetro: uma propriedade do banco de dados usada pelos manipuladores para extrair informações do banco de dados. Você configura essas propriedades como variáveis de ambiente do Lambda.
- String de conexão: uma string de texto usada para estabelecer uma conexão com uma instância de banco de dados.
- Catálogo: um catálogo não AWS Glue registrado no Athena que é um prefixo obrigatório para a propriedade `connection_string`.

- Manipulador de multiplexação: um manipulador Lambda que pode aceitar e usar várias conexões de banco de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector Redshift.

String de conexão

Use uma string de conexão JDBC no seguinte formato para se conectar a uma instância de banco de dados.

```
redshift://${jdbc_connection_string}
```

Uso de um manipulador de multiplexação

É possível usar um multiplexador para se conectar a várias instâncias de banco de dados com uma única função do Lambda. As solicitações são encaminhadas por nome do catálogo. Use as seguintes classes no Lambda.

Manipulador	Classe
Manipulador composto	`RedshiftMuxCompositeHandler`
Manipulador de metadados	`RedshiftMuxMetadataHandler`
Manipulador de registros	`RedshiftMuxRecordHandler`

Parâmetros do manipulador de multiplexação

Parâmetro	Descrição
$*catalog*_connection_string	Obrigatório. Uma string de conexão de instância de banco de dados. Prefixe a variável de ambiente com o nome do catálogo usado no Athena. Por exemplo, se o catálogo registrado no Athena for `myredshiftcatalog` , então o nome da variável

Parâmetro	Descrição
	de ambiente será `myredshiftcatalog_connection_string` .
`default`	Obrigatório. A string de conexão padrão. Essa string é usada quando o catálogo for `lambda:${ AWS_LAMBDA_FUNCTION_NAME }`.

As propriedades do exemplo a seguir são para uma função do Lambda MUX do Redshift que ofereça suporte a duas instâncias de banco de dados: `redshift1` (o padrão) e `redshift2`.

Propriedade	Valor
`default`	`redshift://jdbc:redshift://redshift1.host:5439/dev?user=sample2&password=sample2`
`redshift_catalog1_connection_string`	`redshift://jdbc:redshift://redshift1.host:3306/default?${Test/RDS/Redshift1}`
`redshift_catalog2_connection_string`	`redshift://jdbc:redshift://redshift2.host:3333/default?user=sample2&password=sample2`

Fornecimento de credenciais

Para fornecer um nome de usuário e uma senha para seu banco de dados na string de conexão JDBC, é possível usar as propriedades da string de conexão ou o AWS Secrets Manager.

- String de conexão: um nome de usuário e uma senha podem ser especificados como propriedades na string de conexão do JDBC.

> **Important**
>
> Como prática recomendada de segurança, não use credenciais codificadas em suas variáveis de ambiente ou strings de conexão. Para obter informações sobre como mover

seus segredos codificados para o AWS Secrets Manager, consulte Mover segredos codificados para o AWS Secrets Manager no Guia do usuário do AWS Secrets Manager.

- AWS Secrets Manager: para usar o recurso Athena Federated Query com o AWS Secrets Manager, a VPC conectada à sua função do Lambda deve ter acesso à Internet ou um endpoint da VPC para se conectar ao Secrets Manager.

  É possível colocar o nome de um segredo no AWS Secrets Manager na sua string de conexão JDBC. O conector substitui o nome secreto pelos valores de `username` e `password` do Secrets Manager.

  Para instâncias de banco de dados do Amazon RDS, esse suporte é totalmente integrado. Se você usa o Amazon RDS, é altamente recomendável usar o AWS Secrets Manager e rotação de credenciais. Se seu banco de dados não usar o Amazon RDS, armazene as credenciais em JSON no seguinte formato:

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

Exemplo de string de conexão com nome secreto

A string a seguir tem o nome secreto ${Test/RDS/ `Redshift1`}.

```
redshift://jdbc:redshift://redshift1.host:3306/default?...&${Test/RDS/Redshift1}&...
```

O conector usa o nome secreto para recuperar segredos e fornecer o nome de usuário e a senha, como no exemplo a seguir.

```
redshift://jdbc:redshift://redshift1.host:3306/
default?...&user=sample2&password=sample2&...
```

Atualmente, o conector do Redshift reconhece as propriedades `user` e `password` do JDBC.

Parâmetros de derramamento

O SDK do Lambda pode derramar dados no Amazon S3. Todas as instâncias do banco de dados acessadas pela mesma função do Lambda derramam no mesmo local.

Parâmetro	Descrição
`spill_bucket`	Obrigatório. Nome do bucket de derramamento.
`spill_prefix`	Obrigatório. Prefixo de chave do bucket de derramamento.
`spill_put_request_headers`	(Opcional) Um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.

Suporte ao tipo de dados

A tabela a seguir mostra os correspondentes tipos de dados do JDBC e do Apache Arrow.

JDBC	Arrow
Booleano	Bit
Inteiro	Tiny
Short	Smallint
Inteiro	Int
Longo	Bigint
float	Float4
Double	Float8
Data	Data/Dia
Timestamp	Date Milli
String	Varchar

JDBC	Arrow
Bytes	Varbinary
BigDecimal	Decimal
ARRAY	Lista

Partições e divisões

O Redshift não oferece suporte a partições externas. Para obter mais informações sobre problemas relativos à performance, consulte Performance.

Performance

O conector do Athena para o Redshift executa a passagem direta de predicados para diminuir os dados examinados pela consulta. Cláusulas `LIMIT`, cláusulas `ORDER BY`, predicados simples e expressões complexas são passados diretamente ao conector para reduzir a quantidade de dados examinados e o runtime de execução da consulta. Porém, selecionar um subconjunto de colunas resulta, algumas vezes, em um runtime mais longo. O Amazon Redshift é particularmente suscetível à lentidão na execução de consultas quando você executa várias consultas simultaneamente.

Cláusulas LIMIT

Uma instrução `LIMIT N` reduz os dados examinados pela consulta. Com a passagem direta de `LIMIT N`, o conector só retorna `N` linhas para o Athena.

Principais N consultas

Uma das `N` principais consultas especifica uma ordenação do conjunto de resultados e um limite no número de linhas retornadas. Você pode usar esse tipo de consulta para determinar os `N` principais valores máximos ou `N` principais valores mínimos para os conjuntos de dados. Com os `N` pushdown principais, o conector só retorna `N` linhas para o Athena.

Predicados

Um predicado é uma expressão na cláusula `WHERE` de uma consulta SQL, que avalia para um valor booleano e filtra as linhas com base em várias condições. O conector do Athena para o Redshift pode combinar essas expressões e passá-las diretamente ao Redshift para melhorar a funcionalidade e reduzir a quantidade de dados examinados.

Os seguintes operadores do conector do Athena para o Redshift são compatíveis com a passágem de predicados:

- Booleanos: E, OU, NÃO
- Igualdade: EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Aritméticos: ADICIONAR, SUBTRAIR, MULTIPLICAR, DIVIDIR, MÓDULO, NEGAR
- Outros:LIKE_PATTERN, IN

Exemplo de passagem direta combinada

Para ter recursos aprimorados de consulta, combine os tipos de passagem direta, como no seguinte exemplo:

```
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;
```

Para ver um artigo sobre como usar o pushdown de predicado para melhorar o desempenho em consultas federadas, incluindo Amazon Redshift, consulte Melhorar as consultas federadas com o pushdown de predicados no Amazon Athena no AWSBlog de Big Data.

Consultas de passagem

O conector Redshift é compatível com consultas de passagem. As consultas de passagem usam uma função de tabela para enviar sua consulta completa para execução na fonte de dados.

Para usar consultas de passagem com o Redshift, você pode empregar a seguinte sintaxe:

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

O exemplo de consulta a seguir envia uma consulta para uma fonte de dados no Redshift. A consulta seleciona todas as colunas na tabela `customer`, limitando os resultados a 10.

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

Recursos adicionais do

Para obter as informações mais recentes sobre a versão do driver JDBC, consulte o arquivo pom.xml do conector do Redshift em GitHub.com.

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o SAP HANA

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Limitações

- Não há suporte para operações de gravação de DDL.
- Em uma configuração de multiplexador, o prefixo e o bucket de derramamento são compartilhados em todas as instâncias do banco de dados.
- Quaisquer limites relevantes do Lambda. Para obter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do AWS Lambda.
- No SAP HANA, os nomes dos objetos são convertidos em maiúsculas quando armazenados no banco de dados do SAP HANA. No entanto, como os nomes entre aspas diferenciam maiúsculas e minúsculas, é possível que duas tabelas tenham o mesmo nome em maiúsculas e minúsculas (por exemplo, `EMPLOYEE` e `employee`).

  Na Athena Federated Query, os nomes das tabelas de esquema são fornecidos para a função do Lambda em letras minúsculas. Para resolver esse problema, é possível fornecer dicas de consulta `@schemaCase` para recuperar os dados das tabelas que tenham nomes que diferenciem maiúsculas de minúsculas. A seguir estão dois exemplos de consultas com dicas de consulta.

  ```
  SELECT *
  ```

```
FROM "lambda:saphanaconnector".SYSTEM."MY_TABLE@schemaCase=upper&tableCase=upper"
```

```
SELECT *
FROM "lambda:saphanaconnector".SYSTEM."MY_TABLE@schemaCase=upper&tableCase=lower"
```

Termos

Os termos a seguir estão relacionados ao conector SAP HANA.

- Instância do banco de dados: qualquer instância de um banco de dados implantado on-premises, no Amazon EC2 ou no Amazon RDS.
- Manipulador: um manipulador Lambda que acessa sua instância de banco de dados. Um manipulador pode ser para metadados ou para registros de dados.
- Manipulador de metadados: um manipulador Lambda que recupera metadados da sua instância de banco de dados.
- Manipulador de registros: um manipulador Lambda que recupera registros de dados da sua instância de banco de dados.
- Manipulador composto: um manipulador Lambda que recupera tanto metadados quanto registros de dados da sua instância de banco de dados.
- Propriedade ou parâmetro: uma propriedade do banco de dados usada pelos manipuladores para extrair informações do banco de dados. Você configura essas propriedades como variáveis de ambiente do Lambda.
- String de conexão: uma string de texto usada para estabelecer uma conexão com uma instância de banco de dados.
- Catálogo: um catálogo não AWS Glue registrado no Athena que é um prefixo obrigatório para a propriedade `connection_string`.
- Manipulador de multiplexação: um manipulador Lambda que pode aceitar e usar várias conexões de banco de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector SAP HANA.

String de conexão

Use uma string de conexão JDBC no seguinte formato para se conectar a uma instância de banco de dados.

```
saphana://${jdbc_connection_string}
```

Uso de um manipulador de multiplexação

É possível usar um multiplexador para se conectar a várias instâncias de banco de dados com uma única função do Lambda. As solicitações são encaminhadas por nome do catálogo. Use as seguintes classes no Lambda.

Manipulador	Classe
Manipulador composto	`SaphanaMuxCompositeHandler`
Manipulador de metadados	`SaphanaMuxMetadataHandler`
Manipulador de registros	`SaphanaMuxRecordHandler`

Parâmetros do manipulador de multiplexação

Parâmetro	Descrição
$*catalog*_connection_string	Obrigatório. Uma string de conexão de instância de banco de dados. Prefixe a variável de ambiente com o nome do catálogo usado no Athena. Por exemplo, se o catálogo registrado no Athena for `mysaphanacatalog`, então o nome da variável de ambiente será `mysaphanacatalog_connection_string`.
`default`	Obrigatório. A string de conexão padrão. Essa string é usada quando o catálogo for `lambda:${` *AWS_LAMBDA_FUNCTION_NAME* `}`.

As propriedades de exemplo a seguir são para uma função do Lambda MUX do Saphana que ofereça suporte a duas instâncias de banco de dados: `saphana1` (o padrão) e `saphana2`.

Propriedade	Valor
`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`

Fornecimento de credenciais

Para fornecer um nome de usuário e uma senha para seu banco de dados na string de conexão JDBC, é possível usar as propriedades da string de conexão ou o AWS Secrets Manager.

- String de conexão: um nome de usuário e uma senha podem ser especificados como propriedades na string de conexão do JDBC.

  > **Important**
  >
  > Como prática recomendada de segurança, não use credenciais codificadas em suas variáveis de ambiente ou strings de conexão. Para obter informações sobre como mover seus segredos codificados para o AWS Secrets Manager, consulte Mover segredos codificados para o AWS Secrets Manager no Guia do usuário do AWS Secrets Manager.

- AWS Secrets Manager: para usar o recurso Athena Federated Query com o AWS Secrets Manager, a VPC conectada à sua função do Lambda deve ter acesso à Internet ou um endpoint da VPC para se conectar ao Secrets Manager.

É possível colocar o nome de um segredo no AWS Secrets Manager na sua string de conexão JDBC. O conector substitui o nome secreto pelos valores de `username` e `password` do Secrets Manager.

Para instâncias de banco de dados do Amazon RDS, esse suporte é totalmente integrado. Se você usa o Amazon RDS, é altamente recomendável usar o AWS Secrets Manager e rotação de credenciais. Se seu banco de dados não usar o Amazon RDS, armazene as credenciais em JSON no seguinte formato:

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

Exemplo de string de conexão com nome secreto

A string a seguir tem o nome secreto `${Test/RDS/Saphana1}`.

```
saphana://jdbc:sap://saphana1.host:port/?${Test/RDS/Saphana1}&...
```

O conector usa o nome secreto para recuperar segredos e fornecer o nome de usuário e a senha, como no exemplo a seguir.

```
saphana://jdbc:sap://saphana1.host:port/?user=sample2&password=sample2&...
```

Atualmente, o conector SAP HANA reconhece as propriedades `user` e `password` do JDBC.

Uso de um único manipulador de conexão

É possível usar os seguintes metadados de conexão única e manipuladores de registros para se conectar a uma única instância do SAP HANA.

Tipo de manipulador	Classe
Manipulador composto	`SaphanaCompositeHandler`
Manipulador de metadados	`SaphanaMetadataHandler`
Manipulador de registros	`SaphanaRecordHandler`

Parâmetros do manipulador de conexão única

Parâmetro	Descrição
`default`	Obrigatório. A string de conexão padrão.

Os manipuladores de conexão únicos oferecem suporte a uma instância de banco de dados e devem fornecer um parâmetro de string de conexão `default`. Todas as outras strings de conexão são ignoradas.

O exemplo de propriedade a seguir é para uma única instância SAP HANA com suporte em uma função do Lambda.

Propriedade	Valor
`default`	`saphana://jdbc:sap://saphana1.host:port/?secret=Test/RDS/Saphana1`

Parâmetros de derramamento

O SDK do Lambda pode derramar dados no Amazon S3. Todas as instâncias do banco de dados acessadas pela mesma função do Lambda derramam no mesmo local.

Parâmetro	Descrição
`spill_bucket`	Obrigatório. Nome do bucket de derramamento.
`spill_prefix`	Obrigatório. Prefixo de chave do bucket de derramamento.
`spill_put_request_headers`	(Opcional) Um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.

Suporte ao tipo de dados

A tabela a seguir mostra os correspondentes tipos de dados do JDBC e do Apache Arrow.

JDBC	Arrow
Booleano	Bit
Inteiro	Tiny
Short	Smallint
Inteiro	Int
Longo	Bigint
float	Float4
Double	Float8
Data	Data/Dia
Timestamp	Date Milli
String	Varchar
Bytes	Varbinary
BigDecimal	Decimal
ARRAY	Lista

Conversões de tipos de dados

Além das conversões de JDBC para Arrow, o conector realiza algumas outras conversões para tornar a fonte SAP HANA e os tipos de dados do Athena compatíveis. Essas conversões ajudam a garantir que as consultas sejam executadas com êxito. A tabela a seguir mostra essas conversões:

Tipo de dados de origem (SAP HANA)	Tipo de dados convertidos (Athena)
DECIMAL	BIGINT
INTEGER	INT
DATA	DATEDAY
TIMESTAMP	DATEMILLI

Todos os outros tipos de dados sem suporte são convertidos em `VARCHAR`.

Partições e divisões

Uma partição é representada por uma única coluna de partição do tipo `Integer`. A coluna contém os nomes das partições definidas em uma tabela do SAP HANA. Para uma tabela que não tenha nomes de partição, * será retornado, o que equivale a uma única partição. Uma partição é equivalente a uma divisão.

Nome	Tipo	Descrição
PART_ID	Inteiro	Partição nomeada no SAP HANA.

Performance

O SAP HANA oferece suporte a partições nativas. O conector do Athena para o SAP HANA pode recuperar dados dessas partições em paralelo. Se você quiser consultar conjuntos de dados muito grandes com distribuição uniforme de partições, o particionamento nativo é altamente recomendado. A seleção de um subconjunto de colunas acelera o runtime da consulta e reduz os dados verificados de forma significativa. O conector apresenta um controle de utilização significativo e, às vezes, falhas na consulta devido à simultaneidade.

O conector do Athena para o SAP HANA executa a passagem direta de predicados para diminuir os dados examinados pela consulta. Cláusulas `LIMIT`, predicados simples e expressões complexas são passados diretamente ao conector para reduzir a quantidade de dados examinados e o runtime de execução da consulta.

Cláusulas LIMIT

Uma instrução `LIMIT N` reduz os dados examinados pela consulta. Com a passagem direta de `LIMIT N`, o conector só retorna `N` linhas para o Athena.

Predicados

Um predicado é uma expressão na cláusula `WHERE` de uma consulta SQL, que avalia para um valor booleano e filtra as linhas com base em várias condições. O conector do Athena para o SAP HANA pode combinar essas expressões e passá-las diretamente ao SAP HANA para melhorar a funcionalidade e reduzir a quantidade de dados examinados.

Os seguintes operadores do conector do Athena para o SAP HANA são compatíveis com a passagem direta de predicados:

- Booleanos: E, OU, NÃO
- Igualdade: EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Aritméticos: ADICIONAR, SUBTRAIR, MULTIPLICAR, DIVIDIR, MÓDULO, NEGAR
- Outros:LIKE_PATTERN, IN

Exemplo de passagem direta combinada

Para ter recursos aprimorados de consulta, combine os tipos de passagem direta, como no seguinte exemplo:

```
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;
```

Consultas de passagem

O conector SAP HANA é compatível com consultas de passagem. As consultas de passagem usam uma função de tabela para enviar sua consulta completa para execução na fonte de dados.

Para usar consultas de passagem com o SAP HANA, você pode empregar a seguinte sintaxe:

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

O exemplo de consulta a seguir envia uma consulta para uma fonte de dados no SAP HANA. A consulta seleciona todas as colunas na tabela `customer`, limitando os resultados a 10.

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

Informações de licença

Ao usar esse conector, você reconhece a inclusão de componentes de terceiros, cuja lista pode ser encontrada no arquivo pom.xml desse conector, e concorda com os termos das respectivas licenças de terceiros fornecidas no arquivo LICENSE.txt em GitHub.com.

Recursos adicionais do

Para obter as informações mais recentes sobre a versão do driver JDBC, consulte o arquivo pom.xml do conector SAP HANA em GitHub.com.

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o Snowflake

O conector do Amazon Athena para o Snowflake permite que o Amazon Athena execute consultas SQL nos dados armazenados no seu banco de dados SQL Snowflake ou em instâncias do RDS usando JDBC.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Limitações

- Não há suporte para operações de gravação de DDL.
- Em uma configuração de multiplexador, o prefixo e o bucket de derramamento são compartilhados em todas as instâncias do banco de dados.
- Quaisquer limites relevantes do Lambda. Para obter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do AWS Lambda.
- No momento, as visualizações do Snowflake com uma única divisão são compatíveis.
- No Snowflake, como os nomes dos objetos fazem distinção entre maiúsculas e minúsculas, duas tabelas podem ter o mesmo nome em maiúsculas e minúsculas (por exemplo, `EMPLOYEE` e `employee`). Na Athena Federated Query, os nomes das tabelas de esquema são fornecidos para a função do Lambda em letras minúsculas. Para resolver esse problema, é possível fornecer dicas de consulta `@schemaCase` para recuperar os dados das tabelas que tenham nomes que diferenciem maiúsculas de minúsculas. A seguir estão dois exemplos de consultas com dicas de consulta.

```
SELECT *
FROM "lambda:snowflakeconnector".SYSTEM."MY_TABLE@schemaCase=upper&tableCase=upper"
```

```
SELECT *
FROM "lambda:snowflakeconnector".SYSTEM."MY_TABLE@schemaCase=upper&tableCase=lower"
```

Termos

Os termos a seguir estão relacionados ao conector Snowflake.

- Instância do banco de dados: qualquer instância de um banco de dados implantado on-premises, no Amazon EC2 ou no Amazon RDS.
- Manipulador: um manipulador Lambda que acessa sua instância de banco de dados. Um manipulador pode ser para metadados ou para registros de dados.
- Manipulador de metadados: um manipulador Lambda que recupera metadados da sua instância de banco de dados.
- Manipulador de registros: um manipulador Lambda que recupera registros de dados da sua instância de banco de dados.

- Manipulador composto: um manipulador Lambda que recupera tanto metadados quanto registros de dados da sua instância de banco de dados.
- Propriedade ou parâmetro: uma propriedade do banco de dados usada pelos manipuladores para extrair informações do banco de dados. Você configura essas propriedades como variáveis de ambiente do Lambda.
- String de conexão: uma string de texto usada para estabelecer uma conexão com uma instância de banco de dados.
- Catálogo: um catálogo não AWS Glue registrado no Athena que é um prefixo obrigatório para a propriedade `connection_string`.
- Manipulador de multiplexação: um manipulador Lambda que pode aceitar e usar várias conexões de banco de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector Snowflake.

String de conexão

Use uma string de conexão JDBC no seguinte formato para se conectar a uma instância de banco de dados.

```
snowflake://${jdbc_connection_string}
```

Uso de um manipulador de multiplexação

É possível usar um multiplexador para se conectar a várias instâncias de banco de dados com uma única função do Lambda. As solicitações são encaminhadas por nome do catálogo. Use as seguintes classes no Lambda.

Manipulador	Classe
Manipulador composto	`SnowflakeMuxCompositeHandler`
Manipulador de metadados	`SnowflakeMuxMetadataHandler`
Manipulador de registros	`SnowflakeMuxRecordHandler`

Parâmetros do manipulador de multiplexação

Parâmetro	Descrição
$*catalog*_connection_string	Obrigatório. Uma string de conexão de instância de banco de dados. Prefixe a variável de ambiente com o nome do catálogo usado no Athena. Por exemplo, se o catálogo registrado no Athena for mysnowflakecatalog , então o nome da variável de ambiente será mysnowflakecatalog_connection_string .
default	Obrigatório. A string de conexão padrão. Essa string é usada quando o catálogo for lambda:${ *AWS_LAMBDA_FUNCTION_NAME* }.

As propriedades de exemplo a seguir são para uma função do Lambda MUX do Snowflake que ofereça suporte a duas instâncias de banco de dados: snowflake1 (o padrão) e snowflake2.

Propriedade	Valor
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

Fornecimento de credenciais

Para fornecer um nome de usuário e uma senha para seu banco de dados na string de conexão JDBC, é possível usar as propriedades da string de conexão ou o AWS Secrets Manager.

- String de conexão: um nome de usuário e uma senha podem ser especificados como propriedades na string de conexão do JDBC.

  > **Important**
  >
  > Como prática recomendada de segurança, não use credenciais codificadas em suas variáveis de ambiente ou strings de conexão. Para obter informações sobre como mover seus segredos codificados para o AWS Secrets Manager, consulte Mover segredos codificados para o AWS Secrets Manager no Guia do usuário do AWS Secrets Manager.

- AWS Secrets Manager: para usar o recurso Athena Federated Query com o AWS Secrets Manager, a VPC conectada à sua função do Lambda deve ter acesso à Internet ou um endpoint da VPC para se conectar ao Secrets Manager.

  É possível colocar o nome de um segredo no AWS Secrets Manager na sua string de conexão JDBC. O conector substitui o nome secreto pelos valores de `username` e `password` do Secrets Manager.

  Para instâncias de banco de dados do Amazon RDS, esse suporte é totalmente integrado. Se você usa o Amazon RDS, é altamente recomendável usar o AWS Secrets Manager e rotação de credenciais. Se seu banco de dados não usar o Amazon RDS, armazene as credenciais em JSON no seguinte formato:

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

Exemplo de string de conexão com nome secreto

A string a seguir tem o nome secreto `${Test/RDS/Snowflake1}`.

```
snowflake://jdbc:snowflake://snowflake1.host:port/?
warehouse=warehousename&db=db1&schema=schema1${Test/RDS/Snowflake1}&...
```

O conector usa o nome secreto para recuperar segredos e fornecer o nome de usuário e a senha, como no exemplo a seguir.

```
snowflake://jdbc:snowflake://snowflake1.host:port/
warehouse=warehousename&db=db1&schema=schema1&user=sample2&password=sample2&...
```

Atualmente, o Snowflake reconhece as propriedades `user` e `password` do JDBC. Ele também aceita o nome do usuário e a senha no formato *`nome de usuário`*`/`*`senha`* sem as chaves `user` ou `password`.

Uso de um único manipulador de conexão

É possível usar os seguintes metadados de conexão única e manipuladores de registros para se conectar a uma única instância do Snowflake.

Tipo de manipulador	Classe
Manipulador composto	`SnowflakeCompositeHandler`
Manipulador de metadados	`SnowflakeMetadataHandler`
Manipulador de registros	`SnowflakeRecordHandler`

Parâmetros do manipulador de conexão única

Parâmetro	Descrição
`default`	Obrigatório. A string de conexão padrão.

Os manipuladores de conexão únicos oferecem suporte a uma instância de banco de dados e devem fornecer um parâmetro de string de conexão `default`. Todas as outras strings de conexão são ignoradas.

O exemplo de propriedade a seguir é para uma única instância do Snowflake com suporte em uma função do Lambda.

Propriedade	Valor
default	snowflake://jdbc:snowflake://snowflake1.host:port/?secret=Test/RDS/Snowflake1

Parâmetros de derramamento

O SDK do Lambda pode derramar dados no Amazon S3. Todas as instâncias do banco de dados acessadas pela mesma função do Lambda derramam no mesmo local.

Parâmetro	Descrição
spill_bucket	Obrigatório. Nome do bucket de derramamento.
spill_prefix	Obrigatório. Prefixo de chave do bucket de derramamento.
spill_put_request_headers	(Opcional) Um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação putObject do Amazon S3 usada para o derramamento (por exemplo, {"x-amz-server-side-encryption" : "AES256"}). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.

Suporte ao tipo de dados

A tabela a seguir mostra os correspondentes tipos de dados do JDBC e do Apache Arrow.

JDBC	Arrow
Booleano	Bit
Inteiro	Tiny
Short	Smallint
Inteiro	Int

JDBC	Arrow
Longo	Bigint
float	Float4
Double	Float8
Data	Data/Dia
Timestamp	Date Milli
String	Varchar
Bytes	Varbinary
BigDecimal	Decimal
ARRAY	Lista

Conversões de tipos de dados

Além das conversões de JDBC para Arrow, o conector realiza algumas outras conversões para tornar a fonte do Snowflake e os tipos de dados do Athena compatíveis. Essas conversões ajudam a garantir que as consultas sejam executadas com êxito. A tabela a seguir mostra essas conversões:

Tipo de dados de origem (Snowflake)	Tipo de dados convertidos (Athena)
TIMESTAMP	TIMESTAMPMILLI
DATA	TIMESTAMPMILLI
INTEGER	INT
DECIMAL	BIGINT
TIMESTAMP_NTZ	TIMESTAMPMILLI

Todos os outros tipos de dados sem suporte são convertidos em `VARCHAR`.

Partições e divisões

As partições são usadas para determinar como gerar divisões para o conector. O Athena constrói uma coluna sintética do tipo `varchar` que representa o esquema de particionamento da tabela para ajudar o conector a gerar divisões. O conector não modifica a definição real da tabela.

Para criar essa coluna sintética e as partições, o Athena exige que uma chave primária seja definida. No entanto, como o Snowflake não impõe restrições de chave primária, você mesmo deve impor a exclusividade. Caso você não faça isso, o Athena vai ter como padrão uma divisão única.

Performance

Para obter um desempenho ideal, use filtros nas consultas sempre que possível. Além disso, é altamente recomendável o particionamento nativo para recuperar grandes conjuntos de dados com distribuição uniforme de partições. A seleção de um subconjunto de colunas acelera o runtime da consulta e reduz os dados verificados de forma significativa. O conector Snowflake é resiliente ao controle de utilização devido à simultaneidade.

O conector do Athena para o Snowflake executa a passagem direta de predicados para diminuir os dados examinados pela consulta. Cláusulas `LIMIT`, predicados simples e expressões complexas são passados diretamente ao conector para reduzir a quantidade de dados examinados e o runtime de execução da consulta.

Cláusulas LIMIT

Uma instrução `LIMIT N` reduz os dados examinados pela consulta. Com a passagem direta de `LIMIT N`, o conector só retorna `N` linhas para o Athena.

Predicados

Um predicado é uma expressão na cláusula `WHERE` de uma consulta SQL, que avalia para um valor booleano e filtra as linhas com base em várias condições. O conector do Athena para o Snowflake pode combinar essas expressões e passá-las diretamente ao Snowflake para melhorar a funcionalidade e reduzir a quantidade de dados examinados.

Os seguintes operadores do conector do Athena para o Snowflake são compatíveis com a passagem direta de predicados:

- Booleanos: E, OU, NÃO
- Igualdade: EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL

- Aritméticos: ADICIONAR, SUBTRAIR, MULTIPLICAR, DIVIDIR, MÓDULO, NEGAR
- Outros:LIKE_PATTERN, IN

Exemplo de passagem direta combinada

Para ter recursos aprimorados de consulta, combine os tipos de passagem direta, como no seguinte exemplo:

```
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;
```

Consultas de passagem

O conector Snowflake é compatível com consultas de passagem. As consultas de passagem usam uma função de tabela para enviar sua consulta completa para execução na fonte de dados.

Para usar consultas de passagem com o Snowflake, você pode empregar a seguinte sintaxe:

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

O exemplo de consulta a seguir envia uma consulta para uma fonte de dados no Snowflake. A consulta seleciona todas as colunas na tabela `customer`, limitando os resultados a 10.

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

Informações de licença

Ao usar esse conector, você reconhece a inclusão de componentes de terceiros, cuja lista pode ser encontrada no arquivo pom.xml desse conector, e concorda com os termos das respectivas licenças de terceiros fornecidas no arquivo LICENSE.txt em GitHub.com.

Recursos adicionais do

Para obter as informações mais recentes sobre a versão do driver JDBC, consulte o arquivo pom.xml do conector Snowflake em GitHub.com.

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o Microsoft SQL Server

O conector do Amazon Athena para o Microsoft SQL Server permite que o Amazon Athena execute consultas SQL em dados armazenados no Microsoft SQL Server usando JDBC.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Limitações

- Não há suporte para operações de gravação de DDL.
- Em uma configuração de multiplexador, o prefixo e o bucket de derramamento são compartilhados em todas as instâncias do banco de dados.
- Quaisquer limites relevantes do Lambda. Para obter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do AWS Lambda.
- Em condições de filtro, você deve converter os tipos de dados `Date` e `Timestamp` para o tipo de dados apropriado.
- Para pesquisar valores negativos dos tipos `Real` e `Float`, use o operador <= ou >=.
- Não há suporte para os tipos de dados `binary`, `varbinary`, `image` e `rowversion`.

Termos

Os termos a seguir estão relacionados ao conector SQL Server.

- Instância do banco de dados: qualquer instância de um banco de dados implantado on-premises, no Amazon EC2 ou no Amazon RDS.

- Manipulador: um manipulador Lambda que acessa sua instância de banco de dados. Um manipulador pode ser para metadados ou para registros de dados.
- Manipulador de metadados: um manipulador Lambda que recupera metadados da sua instância de banco de dados.
- Manipulador de registros: um manipulador Lambda que recupera registros de dados da sua instância de banco de dados.
- Manipulador composto: um manipulador Lambda que recupera tanto metadados quanto registros de dados da sua instância de banco de dados.
- Propriedade ou parâmetro: uma propriedade do banco de dados usada pelos manipuladores para extrair informações do banco de dados. Você configura essas propriedades como variáveis de ambiente do Lambda.
- String de conexão: uma string de texto usada para estabelecer uma conexão com uma instância de banco de dados.
- Catálogo: um catálogo não AWS Glue registrado no Athena que é um prefixo obrigatório para a propriedade `connection_string`.
- Manipulador de multiplexação: um manipulador Lambda que pode aceitar e usar várias conexões de banco de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector SQL Server.

String de conexão

Use uma string de conexão JDBC no seguinte formato para se conectar a uma instância de banco de dados.

```
sqlserver://${jdbc_connection_string}
```

Uso de um manipulador de multiplexação

É possível usar um multiplexador para se conectar a várias instâncias de banco de dados com uma única função do Lambda. As solicitações são encaminhadas por nome do catálogo. Use as seguintes classes no Lambda.

Manipulador	Classe
Manipulador composto	`SqlServerMuxCompositeHandler`
Manipulador de metadados	`SqlServerMuxMetadataHandler`
Manipulador de registros	`SqlServerMuxRecordHandler`

Parâmetros do manipulador de multiplexação

Parâmetro	Descrição
`$`*`catalog`*`_connection_string`	Obrigatório. Uma string de conexão de instância de banco de dados. Prefixe a variável de ambiente com o nome do catálogo usado no Athena. Por exemplo, se o catálogo registrado no Athena for `mysqlservercatalog` , então o nome da variável de ambiente será `mysqlservercatalog_connection_string` .
`default`	Obrigatório. A string de conexão padrão. Essa string é usada quando o catálogo for `lambda:${` *`AWS_LAMBDA_FUNCTION_NAME`* `}`.

As propriedades de exemplo a seguir são para uma função do Lambda MUX do SqlServer que ofereça suporte a duas instâncias de banco de dados: `sqlserver1` (o padrão) e `sqlserver2`.

Propriedade	Valor
`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`*`}`

Propriedade	Valor
`sqlserver_catalog2_connection_string`	`sqlserver://jdbc:sqlserver://sqlserver2. hostname:port;databaseName= <database_name> ;${secret2_name }`

Fornecimento de credenciais

Para fornecer um nome de usuário e uma senha para seu banco de dados na string de conexão JDBC, é possível usar as propriedades da string de conexão ou o AWS Secrets Manager.

- String de conexão: um nome de usuário e uma senha podem ser especificados como propriedades na string de conexão do JDBC.

  > **Important**
  >
  > Como prática recomendada de segurança, não use credenciais codificadas em suas variáveis de ambiente ou strings de conexão. Para obter informações sobre como mover seus segredos codificados para o AWS Secrets Manager, consulte Mover segredos codificados para o AWS Secrets Manager no Guia do usuário do AWS Secrets Manager.

- AWS Secrets Manager: para usar o recurso Athena Federated Query com o AWS Secrets Manager, a VPC conectada à sua função do Lambda deve ter acesso à Internet ou um endpoint da VPC para se conectar ao Secrets Manager.

  É possível colocar o nome de um segredo no AWS Secrets Manager na sua string de conexão JDBC. O conector substitui o nome secreto pelos valores de `username` e `password` do Secrets Manager.

  Para instâncias de banco de dados do Amazon RDS, esse suporte é totalmente integrado. Se você usa o Amazon RDS, é altamente recomendável usar o AWS Secrets Manager e rotação de credenciais. Se seu banco de dados não usar o Amazon RDS, armazene as credenciais em JSON no seguinte formato:

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

Exemplo de string de conexão com nome secreto

A string a seguir tem o nome secreto `${secret_name}`.

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

O conector usa o nome secreto para recuperar segredos e fornecer o nome de usuário e a senha, como no exemplo a seguir.

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

Uso de um único manipulador de conexão

É possível usar os seguintes metadados de conexão única e manipuladores de registros para se conectar a uma única instância do SQL Server.

Tipo de manipulador	Classe
Manipulador composto	`SqlServerCompositeHandler`
Manipulador de metadados	`SqlServerMetadataHandler`
Manipulador de registros	`SqlServerRecordHandler`

Parâmetros do manipulador de conexão única

Parâmetro	Descrição
`default`	Obrigatório. A string de conexão padrão.

Os manipuladores de conexão únicos oferecem suporte a uma instância de banco de dados e devem fornecer um parâmetro de string de conexão `default`. Todas as outras strings de conexão são ignoradas.

O exemplo de propriedade a seguir é para uma única instância do SQL Server com suporte em uma função do Lambda.

Propriedade	Valor
default	`sqlserver://jdbc:sqlserver://` *hostname*`:`*port*`;database Name=` *<database_name>* `;${`*secret_name*` }`

Parâmetros de derramamento

O SDK do Lambda pode derramar dados no Amazon S3. Todas as instâncias do banco de dados acessadas pela mesma função do Lambda derramam no mesmo local.

Parâmetro	Descrição
`spill_bucket`	Obrigatório. Nome do bucket de derramamento.
`spill_prefix`	Obrigatório. Prefixo de chave do bucket de derramamento.
`spill_put_request_headers`	(Opcional) Um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.

Suporte ao tipo de dados

A tabela a seguir mostra os tipos de dados correspondentes para SQL Server e Apache Arrow.

SQL Server	Arrow
bit	TINYINT
tinyint	SMALLINT
smallint	SMALLINT
int	INT

SQL Server	Arrow
bigint	BIGINT
decimal	DECIMAL
numeric	FLOAT8
smallmoney	FLOAT8
money	DECIMAL
float[24]	FLOAT4
float[53]	FLOAT8
real	FLOAT4
datetime	Date(MILLISECOND)
datetime2	Date(MILLISECOND)
smalldatetime	Date(MILLISECOND)
date	Date(DAY)
horário	VARCHAR
datetimeoffset	Date(MILLISECOND)
char[n]	VARCHAR
varchar[n/max]	VARCHAR
nchar[n]	VARCHAR
nvarchar[n/max]	VARCHAR
text	VARCHAR
ntext	VARCHAR

Partições e divisões

Uma partição é representada por uma única coluna de partição do tipo `varchar`. No caso do conector SQL Server, uma função de partição determina como as partições serão aplicadas na tabela. As informações sobre a função de partição e o nome da coluna são recuperadas da tabela de metadados do SQL Server. Em seguida, uma consulta personalizada obtém a partição. As divisões são criadas com base no número de partições distintas recebidas.

Performance

A seleção de um subconjunto de colunas acelera o runtime da consulta e reduz os dados verificados de forma significativa. O conector SQL Server é resiliente ao controle de utilização devido à simultaneidade.

O concetor do Athena para o SQL Server realiza a passagem direta de predicados para diminuir os dados examinados pela consulta. Predicados simples e expressões complexas são passados diretamente ao conector para reduzir a quantidade de dados examinados e o runtime de execução da consulta.

Predicados

Um predicado é uma expressão na cláusula `WHERE` de uma consulta SQL, que avalia para um valor booleano e filtra as linhas com base em várias condições. O conector do Athena para o SQL Server pode combinar essas expressões e passá-las diretamente ao SQL Server para melhorar a funcionalidade e reduzir a quantidade de dados examinados.

Os seguintes operadores do conector do Athena para o SQL Server são compatíveis com a passagem direta de predicados:

- Booleanos: E, OU, NÃO
- Igualdade: EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, IS_DISTINCT_FROM, NULL_IF, IS_NULL
- Aritméticos: ADICIONAR, SUBTRAIR, MULTIPLICAR, DIVIDIR, MÓDULO, NEGAR
- Outros:LIKE_PATTERN, IN

Exemplo de passagem direta combinada

Para ter recursos aprimorados de consulta, combine os tipos de passagem direta, como no seguinte exemplo:

```
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%');
```

Consultas de passagem

O conector SQL Server é compatível com consultas de passagem. As consultas de passagem usam uma função de tabela para enviar sua consulta completa para execução na fonte de dados.

Para usar consultas de passagem com o SQL Server, você pode empregar a seguinte sintaxe:

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

O exemplo de consulta a seguir envia uma consulta para uma fonte de dados no SQL Server. A consulta seleciona todas as colunas na tabela `customer`, limitando os resultados a 10.

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

Informações de licença

Ao usar esse conector, você reconhece a inclusão de componentes de terceiros, cuja lista pode ser encontrada no arquivo pom.xml desse conector, e concorda com os termos das respectivas licenças de terceiros fornecidas no arquivo LICENSE.txt em GitHub.com.

Recursos adicionais do

Para obter as informações mais recentes sobre a versão do driver JDBC, consulte o arquivo pom.xml do conector SQL Server em GitHub.com.

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o Teradata

O conector do Amazon Athena para o Teradata permite que o Amazon Athena execute consultas SQL em dados armazenados nos seus bancos de dados Teradata.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Limitações

- Não há suporte para operações de gravação de DDL.
- Em uma configuração de multiplexador, o prefixo e o bucket de derramamento são compartilhados em todas as instâncias do banco de dados.
- Quaisquer limites relevantes do Lambda. Para obter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do AWS Lambda.

Termos

Os termos a seguir estão relacionados ao conector Teradata.

- Instância do banco de dados: qualquer instância de um banco de dados implantado on-premises, no Amazon EC2 ou no Amazon RDS.
- Manipulador: um manipulador Lambda que acessa sua instância de banco de dados. Um manipulador pode ser para metadados ou para registros de dados.
- Manipulador de metadados: um manipulador Lambda que recupera metadados da sua instância de banco de dados.
- Manipulador de registros: um manipulador Lambda que recupera registros de dados da sua instância de banco de dados.
- Manipulador composto: um manipulador Lambda que recupera tanto metadados quanto registros de dados da sua instância de banco de dados.
- Propriedade ou parâmetro: uma propriedade do banco de dados usada pelos manipuladores para extrair informações do banco de dados. Você configura essas propriedades como variáveis de ambiente do Lambda.

- String de conexão: uma string de texto usada para estabelecer uma conexão com uma instância de banco de dados.
- Catálogo: um catálogo não AWS Glue registrado no Athena que é um prefixo obrigatório para a propriedade `connection_string`.
- Manipulador de multiplexação: um manipulador Lambda que pode aceitar e usar várias conexões de banco de dados.

Pré-requisito da camada do Lambda

Para usar o conector do Teradata com o Athena, você deve criar uma camada do Lambda que inclua o driver JDBC do Teradata. Uma camada do Lambda é um arquivo `.zip` que contém código adicional para uma função do Lambda. Quando você implanta o conector do Teradata em sua conta, você especifica o ARN da camada. Isso anexa a camada do Lambda com o driver JDBC do Teradata ao conector Teradata para que você possa usá-lo com o Athena.

Para obter mais informações sobre camadas do Lambda, consulte Criar e compartilhar camadas do Lambda no Guia do desenvolvedor do AWS Lambda.

Criar uma camada do Lambda para o conector do Teradata

1. Navegue até a página de download do driver JDBC do Teradata em https://downloads.teradata.com/download/connectivity/jdbc-driver.
2. Baixe o driver JDBC do Teradata. O site exige que você crie uma conta e aceite um contrato de licença para baixar o arquivo.
3. Extraia o arquivo `terajdbc4.jar` do arquivo baixado.
4. Crie a estrutura de pastas descrita a seguir e coloque o arquivo `.jar` nela.

   ```
   java\lib\terajdbc4.jar
   ```
5. Crie um arquivo `.zip` com toda a estrutura de pastas que contém o arquivo `terajdbc4.jar`.
6. Faça login no AWS Management Console e abra o console do AWS Lambda em https://console.aws.amazon.com/lambda/.
7. No painel de navegação, escolha Layers (Camadas) e Create layer (Criar uma camada).
8. Em Name (Nome), insira um nome para a camada (por exemplo, `TeradataJava11LambdaLayer`).
9. Certifique-se de que a opção Upload a .zip file (Carregar um arquivo .zip) esteja selecionada.

10. Escolha Upload (Carregar) e, em seguida, carregue a pasta compactada que contém o driver JDBC do Teradata.
11. Escolha Criar.
12. Na página de detalhes da camada, copie o ARN da camada escolhendo o ícone da área de transferência no topo da página.
13. Salve o ARN para referência.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector Teradata.

String de conexão

Use uma string de conexão JDBC no seguinte formato para se conectar a uma instância de banco de dados.

```
teradata://${jdbc_connection_string}
```

Uso de um manipulador de multiplexação

É possível usar um multiplexador para se conectar a várias instâncias de banco de dados com uma única função do Lambda. As solicitações são encaminhadas por nome do catálogo. Use as seguintes classes no Lambda.

Manipulador	Classe
Manipulador composto	`TeradataMuxCompositeHandler`
Manipulador de metadados	`TeradataMuxMetadataHandler`
Manipulador de registros	`TeradataMuxRecordHandler`

Parâmetros do manipulador de multiplexação

Parâmetro	Descrição
`$catalog_connection_string`	Obrigatório. Uma string de conexão de instância de banco de dados. Prefixe a variável de ambiente com o nome do catálogo usado no Athena. Por exemplo, se o catálogo registrado no Athena for `myteradatacatalog` , então o nome da variável de ambiente será `myteradatacatalog_connection_string` .
`default`	Obrigatório. A string de conexão padrão. Essa string é usada quando o catálogo for `lambda:${ AWS_LAMBDA_FUNCTION_NAME }`.

As propriedades de exemplo a seguir são para uma função do Lambda MUX do Teradata que ofereça suporte a duas instâncias de banco de dados: `teradata1` (o padrão) e `teradata2`.

Propriedade	Valor
`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`

Fornecimento de credenciais

Para fornecer um nome de usuário e uma senha para seu banco de dados na string de conexão JDBC, é possível usar as propriedades da string de conexão ou o AWS Secrets Manager.

- String de conexão: um nome de usuário e uma senha podem ser especificados como propriedades na string de conexão do JDBC.

  > **Important**
  >
  > Como prática recomendada de segurança, não use credenciais codificadas em suas variáveis de ambiente ou strings de conexão. Para obter informações sobre como mover seus segredos codificados para o AWS Secrets Manager, consulte Mover segredos codificados para o AWS Secrets Manager no Guia do usuário do AWS Secrets Manager.

- AWS Secrets Manager: para usar o recurso Athena Federated Query com o AWS Secrets Manager, a VPC conectada à sua função do Lambda deve ter acesso à Internet ou um endpoint da VPC para se conectar ao Secrets Manager.

  É possível colocar o nome de um segredo no AWS Secrets Manager na sua string de conexão JDBC. O conector substitui o nome secreto pelos valores de `username` e `password` do Secrets Manager.

  Para instâncias de banco de dados do Amazon RDS, esse suporte é totalmente integrado. Se você usa o Amazon RDS, é altamente recomendável usar o AWS Secrets Manager e rotação de credenciais. Se seu banco de dados não usar o Amazon RDS, armazene as credenciais em JSON no seguinte formato:

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

Exemplo de string de conexão com nome secreto

A string a seguir tem o nome secreto `${Test/RDS/Teradata1}`.

```
teradata://jdbc:teradata1.host/TMODE=ANSI,CHARSET=UTF8,DATABASE=TEST,${Test/RDS/
Teradata1}&...
```

O conector usa o nome secreto para recuperar segredos e fornecer o nome de usuário e a senha, como no exemplo a seguir.

```
teradata://jdbc:teradata://teradata1.host/
TMODE=ANSI,CHARSET=UTF8,DATABASE=TEST,...&user=sample2&password=sample2&...
```

Atualmente, o Teradata reconhece as propriedades `user` e `password` do JDBC. Ele também aceita o nome do usuário e a senha no formato *`nome de usuário`*`/`*`senha`* sem as chaves `user` ou `password`.

Uso de um único manipulador de conexão

É possível usar os seguintes metadados de conexão única e manipuladores de registros para se conectar a uma única instância Teradata.

Tipo de manipulador	Classe
Manipulador composto	`TeradataCompositeHandler`
Manipulador de metadados	`TeradataMetadataHandler`
Manipulador de registros	`TeradataRecordHandler`

Parâmetros do manipulador de conexão única

Parâmetro	Descrição
`default`	Obrigatório. A string de conexão padrão.

Os manipuladores de conexão únicos oferecem suporte a uma instância de banco de dados e devem fornecer um parâmetro de string de conexão `default`. Todas as outras strings de conexão são ignoradas.

O exemplo de propriedade a seguir é para uma única instância do Teradata com suporte em uma função do Lambda.

Propriedade	Valor
`default`	`teradata://jdbc:teradata://teradata1.host/TMODE=ANSI,CHARSET=UTF8,DATABASE=TEST,secret=Test/RDS/Teradata1`

Parâmetros de derramamento

O SDK do Lambda pode derramar dados no Amazon S3. Todas as instâncias do banco de dados acessadas pela mesma função do Lambda derramam no mesmo local.

Parâmetro	Descrição
`spill_bucket`	Obrigatório. Nome do bucket de derramamento.
`spill_prefix`	Obrigatório. Prefixo de chave do bucket de derramamento.
`spill_put_request_headers`	(Opcional) Um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.

Suporte ao tipo de dados

A tabela a seguir mostra os correspondentes tipos de dados do JDBC e do Apache Arrow.

JDBC	Arrow
Booleano	Bit
Inteiro	Tiny
Short	Smallint

JDBC	Arrow
Inteiro	Int
Longo	Bigint
float	Float4
Double	Float8
Data	Data/Dia
Timestamp	Date Milli
String	Varchar
Bytes	Varbinary
BigDecimal	Decimal
ARRAY	Lista

Partições e divisões

Uma partição é representada por uma única coluna de partição do tipo `Integer`. A coluna contém os nomes das partições definidas em uma tabela do Teradata. Para uma tabela que não tenha nomes de partição, * será retornado, o que equivale a uma única partição. Uma partição é equivalente a uma divisão.

Nome	Tipo	Descrição
Partição	Inteiro	Partição nomeada no Teradata.

Performance

O Teradata oferece suporte a partições nativas. O conector do Athena para o Teradata pode recuperar dados dessas partições em paralelo. Se você quiser consultar conjuntos de dados muito grandes com distribuição uniforme de partições, o particionamento nativo é altamente recomendado.

A seleção de um subconjunto de colunas diminui significativamente o runtime da consulta. O conector apresenta alguns controles de utilização devido à simultaneidade.

O conector do Athena para o Teradata realiza a passagem direta de predicados para diminuir os dados examinados pela consulta. Predicados simples e expressões complexas são passados diretamente ao conector para reduzir a quantidade de dados examinados e o runtime de execução da consulta.

Predicados

Um predicado é uma expressão na cláusula `WHERE` de uma consulta SQL que é avaliado como um valor booleano e filtra as linhas com base em várias condições. O conector do Athena para o Teradata pode combinar essas expressões e passá-las diretamente ao Synapse para melhorar a funcionalidade e reduzir a quantidade de dados examinados.

Os seguintes operadores do conector do Athena para o Teradata são compatíveis com a passagem direta de predicados:

- Booleanos: E, OU, NÃO
- Igualdade:EQUAL, NOT_EQUAL, LESS_THAN, LESS_THAN_OR_EQUAL, GREATER_THAN, GREATER_THAN_OR_EQUAL, NULL_IF, IS_NULL
- Aritméticos: ADICIONAR, SUBTRAIR, MULTIPLICAR, DIVIDIR, MÓDULO, NEGAR
- Outros:LIKE_PATTERN, IN

Exemplo de passagem direta combinada

Para ter recursos aprimorados de consulta, combine os tipos de passagem direta, como no seguinte exemplo:

```
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%');
```

Consultas de passagem

O conector Teradata é compatível com consultas de passagem. As consultas de passagem usam uma função de tabela para enviar sua consulta completa para execução na fonte de dados.

Para usar consultas de passagem com o Teradata, você pode empregar a seguinte sintaxe:

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

O exemplo de consulta a seguir envia uma consulta para uma fonte de dados no Teradata. A consulta seleciona todas as colunas na tabela `customer`, limitando os resultados a 10.

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

Informações de licença

Ao usar esse conector, você reconhece a inclusão de componentes de terceiros, cuja lista pode ser encontrada no arquivo pom.xml desse conector, e concorda com os termos das respectivas licenças de terceiros fornecidas no arquivo LICENSE.txt em GitHub.com.

Recursos adicionais do

Para obter as informações mais recentes sobre a versão do driver JDBC, consulte o arquivo pom.xml do conector Teradata em GitHub.com.

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o Timestream

O conector do Amazon Athena para o Timestream permite que o Amazon Athena se comunique com Amazon Timestream, tornando seus dados de séries temporais acessíveis pelo Amazon Athena. Se preferir, use o AWS Glue Data Catalog como uma fonte de metadados complementares.

O Amazon Timestream é um banco de dados de séries temporais rápido, escalável, totalmente gerenciado e criado para fins específicos que facilita o armazenamento e a análise de trilhões de pontos de dados de séries temporais por dia. O Timestream economiza tempo e custo de gerenciamento do ciclo de vida dos dados de séries temporais mantendo os dados recentes na memória e movendo os dados históricos para um nível de armazenamento com otimização de custo conforme as políticas definidas pelo usuário.

Se você tiver o Lake Formation habilitado em sua conta, o perfil do IAM para seu conector Lambda federado para Athena que você implantou no AWS Serverless Application Repository deve ter acesso de leitura ao AWS Glue Data Catalog no Lake Formation.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector Timestream.

- spill_bucket: especifica o bucket do Amazon S3 para dados que excedem os limites da função do Lambda.
- spill_prefix: (opcional) assume como padrão uma subpasta no `spill_bucket` especificado chamado `athena-federation-spill`. Recomendamos que você configure um ciclo de vida de armazenamento do Amazon S3 neste local para excluir derramamentos anteriores a um número predeterminado de dias ou horas.
- spill_put_request_headers: (opcional) um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.
- kms_key_id: (opcional) por padrão, todos os dados transmitidos para o Amazon S3 são criptografados usando o modo de criptografia autenticado AES-GCM e uma chave gerada aleatoriamente. Para que sua função do Lambda use chaves de criptografia mais fortes geradas pelo KMS, como `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, é possível especificar um ID de chave do KMS.
- disable_spill_encryption: (opcional) quando definido como `True`, desativa a criptografia do derramamento. É padronizado como `False`, para que os dados transmitidos para o S3 sejam criptografados usando o AES-GCM — usando uma chave gerada aleatoriamente ou o KMS para gerar chaves. Desativar a criptografia do derramamento pode melhorar a performance, especialmente se o local do derramamento usar criptografia no lado do servidor.
- glue_catalog: (opcional) use essa opção para especificar um catálogo do AWS Glue entre contas. Por padrão, o conector tenta obter metadados de sua própria conta do AWS Glue.

Configuração de bancos de dados e tabelas no AWS Glue

Se preferir, use o AWS Glue Data Catalog como uma fonte de metadados complementares. Para habilitar uma tabela do AWS Glue para uso com o Timestream, é preciso ter um banco de dados do AWS Glue e uma tabela com nomes que correspondam ao banco de dados Timestream e à tabela para a qual você deseja fornecer metadados complementares.

> **Note**
>
> Para obter a melhor performance, use somente letras minúsculas para nomes de banco de dados e nomes de tabela. O uso de maiúsculas e minúsculas mistas faz com que o conector execute uma pesquisa que não diferencia maiúsculas de minúsculas e é mais computacionalmente intensiva.

Para configurar a tabela do AWS Glue para uso com o Timestream é preciso definir suas propriedades de tabela em AWS Glue.

Para usar uma tabela do AWS Glue para metadados complementares

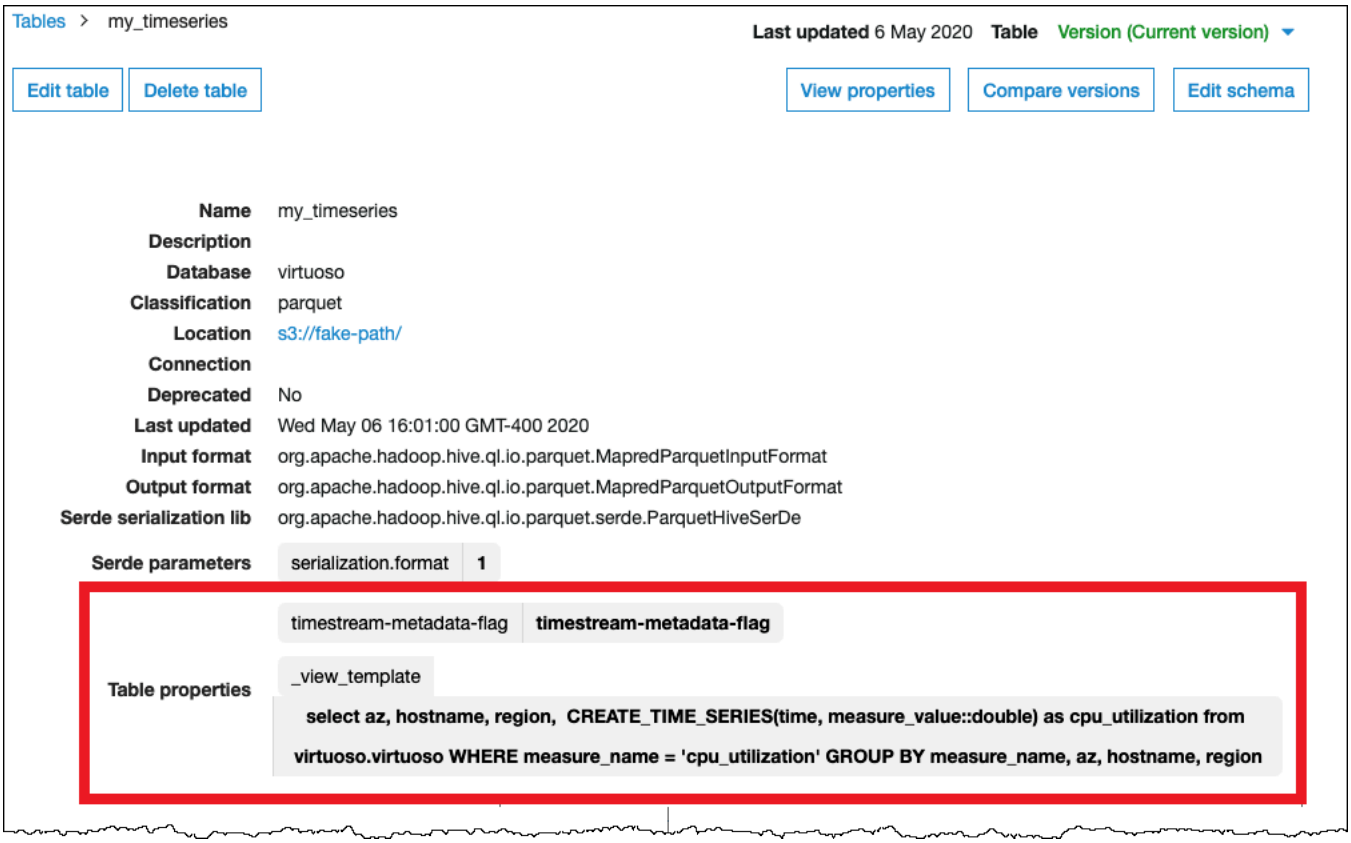
1. Edite a tabela no console do AWS Glue para adicionar as seguintes propriedades da tabela:
   - timestream-metadata-flag: essa propriedade indica ao conector Timestream que o conector pode usar a tabela para metadados complementares. É possível fornecer qualquer valor para `timestream-metadata-flag`, desde que a propriedade `timestream-metadata-flag` esteja presente na lista de propriedades da tabela.
   - _view_template: quando você usa o AWS Glue para metadados complementares, é possível usar essa propriedade da tabela e especificar qualquer Timestream SQL como visualização. O conector Timestream do Athena usa o SQL da visualização junto com o SQL do Athena para executar sua consulta. Isso é útil se você quiser usar um recurso do Timestream SQL que, de outra forma, não está disponível no Athena.
2. Use os tipos de dados apropriados para o AWS Glue, conforme listado neste documento.

Tipos de dados

Atualmente, o conector Timestream oferece suporte somente a um subconjunto dos tipos de dados disponíveis no Timestream, especificamente: os valores escalares `varchar`, `double` e `timestamp`.

Para consultar o tipo de dados da `timeseries`, é preciso configurar uma visualização nas propriedades da tabela do AWS Glue que use a função `CREATE_TIME_SERIES` do Timestream. Você também precisa fornecer um esquema para a visualização que use a sintaxe `ARRAY<STRUCT<time:timestamp,measure_value::double:double>>` como o tipo de qualquer uma de suas colunas de séries temporais. Certifique-se de substituir `double` pelo tipo escalar apropriado para sua tabela.

A imagem a seguir mostra um exemplo de propriedades da tabela do AWS Glue configuradas para definir uma visualização sobre uma série temporal.



Permissões obrigatórias

Os detalhes completos sobre as políticas do IAM exigidas por esse conector podem ser encontrados na seção `Policies` do arquivo athena-timestream.yaml. A lista a seguir resume as permissões necessárias.

- Acesso de gravação do Amazon S3: o conector requer acesso de gravação a um local no Amazon S3 para mostrar resultados de grandes consultas.
- Athena GetQueryExecution: o conector usa esta permissão para falhar rapidamente quando a consulta upstream do Athena é encerrada.

- AWS Glue Data Catalog: o conector Timestream requer acesso somente de leitura ao AWS Glue Data Catalog para obter informações do esquema.
- CloudWatch Logs: o conector requer acesso ao CloudWatch Logs para armazenar registros.
- Acesso ao Timestream: para executar consultas no Timestream.

Performance

Recomendamos o uso da cláusula `LIMIT` para limitar os dados retornados (não os dados verificados) a menos de 256 MB com a finalidade de garantir que as consultas interativas tenham uma boa performance.

O conector do Athena para o Timestream realiza a passagem direta de predicados para diminuir os dados examinados pela consulta. As cláusulas `LIMIT` reduzem a quantidade de dados examinados, mas, se você não fornecer um predicado, deverá esperar que as consultas `SELECT` com uma cláusula `LIMIT` examinem, pelo menos, 16 MB de dados. A seleção de um subconjunto de colunas acelera o runtime da consulta e reduz os dados verificados de forma significativa. O conector Timestream é resiliente ao controle de utilização devido à simultaneidade.

Consultas de passagem

O conector Timestream é compatível com consultas de passagem. As consultas de passagem usam uma função de tabela para enviar sua consulta completa para execução na fonte de dados.

Para usar consultas de passagem com o Timestream, você pode empregar a seguinte sintaxe:

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

O exemplo de consulta a seguir envia uma consulta para uma fonte de dados no Timestream. A consulta seleciona todas as colunas na tabela `customer`, limitando os resultados a 10.

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

Informações de licença

O projeto do conector Timestream do Amazon Athena é licenciado sob a Licença Apache-2.0.

Recursos adicionais do

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o TPC Benchmark DS (TPC-DS)

O conector TPC-DS no Amazon Athena permite que o Amazon Athena se comunique com uma fonte de dados do TPC Benchmark DS gerada aleatoriamente para uso em benchmarking e testes funcionais do Athena Federation. O conector TPC-DS no Athena gera um banco de dados compatível com TPC-DS em um dos quatro fatores de escala. Não recomendamos o uso desse conector como alternativa aos testes de desempenho de data lake baseados no Amazon S3.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.

Parâmetros

Use as variáveis de ambiente do Lambda nesta seção para configurar o conector TPC-DS.

- spill_bucket: especifica o bucket do Amazon S3 para dados que excedem os limites da função do Lambda.
- spill_prefix: (opcional) assume como padrão uma subpasta no `spill_bucket` especificado chamado `athena-federation-spill`. Recomendamos que você configure um ciclo de vida de armazenamento do Amazon S3 neste local para excluir derramamentos anteriores a um número predeterminado de dias ou horas.
- spill_put_request_headers: (opcional) um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.
- kms_key_id: (opcional) por padrão, todos os dados transmitidos para o Amazon S3 são criptografados usando o modo de criptografia autenticado AES-GCM e uma chave gerada

aleatoriamente. Para que sua função do Lambda use chaves de criptografia mais fortes geradas pelo KMS, como `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`, é possível especificar um ID de chave do KMS.

- disable_spill_encryption: (opcional) quando definido como `True`, desativa a criptografia do derramamento. É padronizado como `False`, para que os dados transmitidos para o S3 sejam criptografados usando o AES-GCM — usando uma chave gerada aleatoriamente ou o KMS para gerar chaves. Desativar a criptografia do derramamento pode melhorar o desempenho, especialmente se o local do derramamento usar criptografia no lado do servidor.

Teste de bancos de dados e tabelas

O conector TPC-DS no Athena gera um banco de dados compatível com TPC-DS em um dos quatro fatores de escala `tpcds1`, `tpcds10`, `tpcds100`, `tpcds250` ou `tpcds1000`.

Resumo de tabelas

Para obter uma lista completa de tabelas e colunas de dados de teste, execute a consulta `SHOW TABLES` ou `DESCRIBE TABLE`. O resumo de tabelas a seguir é fornecido para sua conveniência.

1. call_center
2. catalog_page
3. catalog_returns
4. catalog_sales
5. customer
6. customer_address
7. customer_demographics
8. date_dim
9. dbgen_version
10. household_demographics
11. income_band
12. Inventário do
13. item
14. promotion
15. razão
16. ship_mode

17. armazenamento
18. store_returns
19. store_sales
20. time_dim
21. warehouse
22. web_page
23. web_returns
24. web_sales
25. web_site

Para consultas do TPC-DS que sejam compatíveis com esse esquema e dados gerados, consulte o diretório athena-tpcds/src/main/resources/query no GitHub.

Consulta de exemplo

O exemplo de consulta `SELECT` a seguir consulta o catálogo `tpcds` para dados demográficos de clientes em países específicos.

```
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
```

Permissões obrigatórias

Os detalhes completos sobre as políticas do IAM exigidas por esse conector podem ser encontrados na seção `Policies` do arquivo athena-tpcds.yaml. A lista a seguir resume as permissões necessárias.

- Acesso de gravação do Amazon S3: o conector requer acesso de gravação a um local no Amazon S3 para mostrar resultados de grandes consultas.
- Athena GetQueryExecution: o conector usa esta permissão para falhar rapidamente quando a consulta upstream do Athena é encerrada.

Performance

O conector TPC-DS do Athena tenta paralelizar as consultas com base no fator de escala que você escolher. A redução de predicados é realizada dentro da função do Lambda.

Informações de licença

O projeto do conector TPC-DS do Amazon Athena é licenciado sob a Licença Apache-2.0.

Recursos adicionais do

Para obter mais informações sobre esse conector, visite o site correspondente em GitHub.com.

Conector do Amazon Athena para o Vertica

O Vertica é uma plataforma de banco de dados em colunas que pode ser implantada na nuvem ou on-premises e que oferece suporte a data warehouses em escala de exabytes. Você pode usar o conector do Vertica para Amazon Athena em consultas federadas para consultar origens de dados do Vertica usando o Athena. Por exemplo, você pode executar consultas analíticas em um data warehouse no Vertica e um data lake no Amazon S3.

Pré-requisitos

- Implante o conector na sua Conta da AWS usando o console do Athena ou o AWS Serverless Application Repository. Para obter mais informações, consulte Implantar um conector de fonte de dados ou Usar o AWS Serverless Application Repository para implantar um conector de origem de dados.
- Configura uma VPC e um grupo de segurança antes de usar esse conector. Para ter mais informações, consulte Criar uma VPC para um conector de origem de dados.

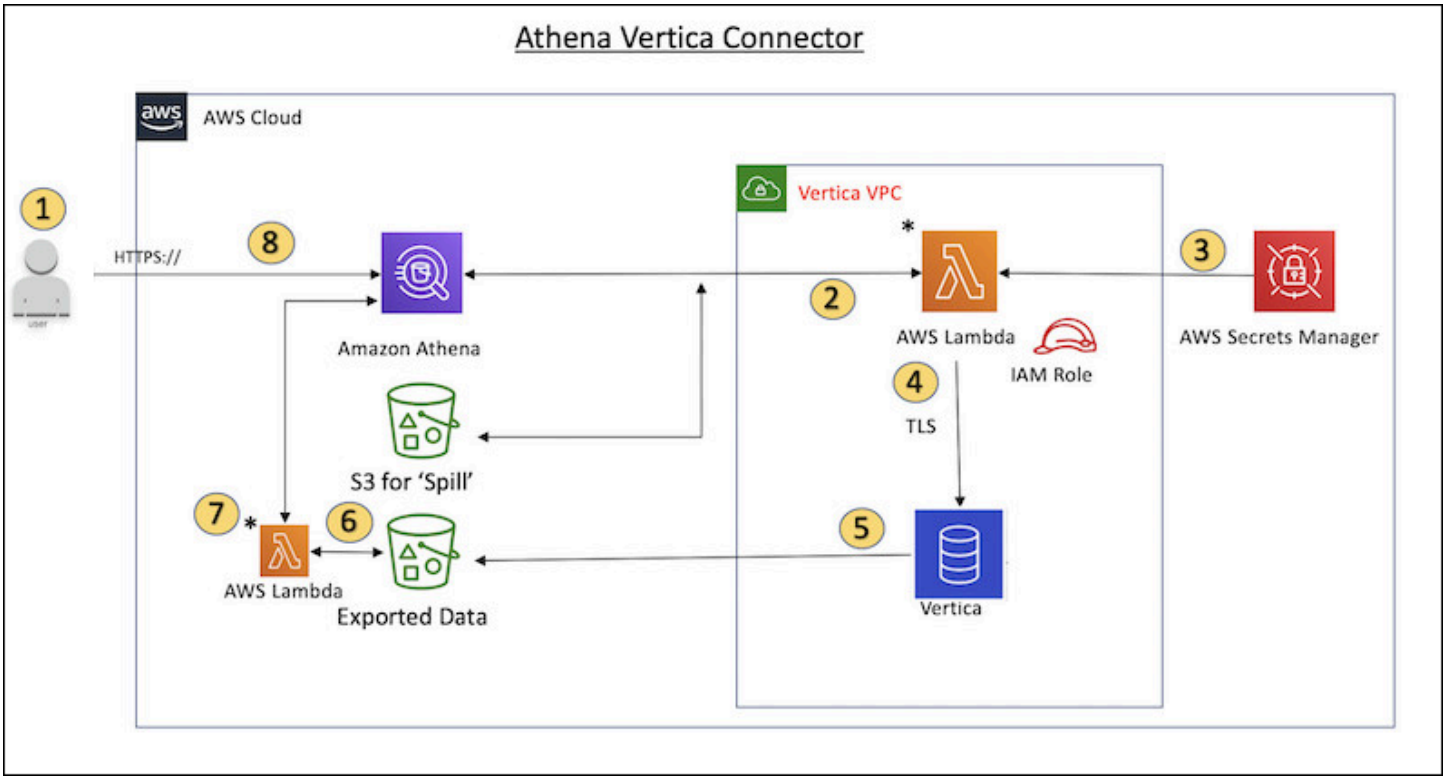
Limitações

- Como o conector Vertica do Athena usa Amazon S3 Select para ler arquivos Parquet do Amazon S3, o desempenho do conector pode ser lento. Ao consultar tabelas grandes, recomendamos usar uma consulta CREATE TABLE AS (SELECT ...) e predicados de SQL.

- Atualmente, devido a um problema conhecido na consulta federada do Athena, o conector faz com que o Vertica exporte todas as colunas da tabela consultada para o Amazon S3, mas somente as colunas consultadas estão visíveis nos resultados no console do Athena.
- Não há suporte para operações de gravação de DDL.
- Quaisquer limites relevantes do Lambda. Para obter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do AWS Lambda.

Fluxo de trabalho

O diagrama a seguir mostra o fluxo de trabalho de uma consulta que use o conector Vertica.



1. Uma consulta SQL é emitida em uma ou mais tabelas no Vertica.
2. O conector analisa a consulta SQL para enviar a parte relevante para o Vertica por meio da conexão do JDBC.
3. As strings de conexão usam o nome de usuário e a senha armazenados no AWS Secrets Manager para obter acesso ao Vertica.
4. O conector envolve a consulta SQL com um comando EXPORT do Vertica, como no exemplo a seguir.

```
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. O Vertica processa a consulta SQL e envia o conjunto de resultados para um bucket do Amazon S3. Para uma melhor throughput, o Vertica usa a opção EXPORT para paralelizar a operação de gravação de vários arquivos Parquet.
6. O Athena examina o bucket do Amazon S3 para determinar o número de arquivos a serem lidos para o conjunto de resultados.
7. O Athena faz várias chamadas para a função do Lambda e usa o Amazon S3 Select para ler os arquivos Parquet do conjunto de resultados. Várias chamadas permitem que o Athena paralelize a leitura dos arquivos do Amazon S3 e alcance uma throughput de até 100 GB por segundo.
8. O Athena processa os dados retornados do Vertica com dados examinados do data lake e retorna o resultado.

Termos

Os termos a seguir estão relacionados ao conector Vertica.

- Instância do banco de dados: qualquer instância de um banco de dados Vertica implantado no Amazon EC2.
- Manipulador: um manipulador Lambda que acessa sua instância de banco de dados. Um manipulador pode ser para metadados ou para registros de dados.
- Manipulador de metadados: um manipulador Lambda que recupera metadados da sua instância de banco de dados.
- Manipulador de registros: um manipulador Lambda que recupera registros de dados da sua instância de banco de dados.
- Manipulador composto: um manipulador Lambda que recupera tanto metadados quanto registros de dados da sua instância de banco de dados.
- Propriedade ou parâmetro: uma propriedade do banco de dados usada pelos manipuladores para extrair informações do banco de dados. Você configura essas propriedades como variáveis de ambiente do Lambda.

- String de conexão: uma string de texto usada para estabelecer uma conexão com uma instância de banco de dados.
- Catálogo: um catálogo não AWS Glue registrado no Athena que é um prefixo obrigatório para a propriedade `connection_string`.

Parâmetros

O conector do Amazon Athena para o Vertica expõe as opções de configuração por meio das variáveis de ambiente do Lambda. É possível usar as seguintes variáveis de ambiente do Lambda para definir o conector.

- AthenaCatalogName: nome da função do Lambda
- ExportBucket: o bucket do Amazon S3 para onde os resultados de consultas do Vertica são exportados.
- SpillBucket: o nome do bucket do Amazon S3 no qual essa função pode derramar dados.
- SpillPrefix: o prefixo para a localização do `SpillBucket` onde essa função pode derramar dados.
- SecurityGroupIds: um ou mais IDs que correspondem ao grupo de segurança que deve ser aplicado à função do Lambda (por exemplo, `sg1`, `sg2` ou `sg3`).
- SubnetIds: uma ou mais IDs de sub-rede que correspondem à sub-rede que a função do Lambda pode usar para acessar sua fonte de dados (por exemplo, `subnet1` ou `subnet2`).
- SecretNameOrPrefix: o nome ou prefixo de um conjunto de nomes no Secrets Manager ao qual essa função tem acesso (por exemplo, `vertica-*`)
- VerticaConnectionString: os detalhes da conexão do Vertica a serem usados por padrão se nenhuma conexão específica do catálogo for definida. A string pode opcionalmente usar a sintaxe do AWS Secrets Manager (por exemplo, `${secret_name}`).
- VPC ID: o ID da VPC a ser associada à função do Lambda.

String de conexão

Use uma string de conexão JDBC no seguinte formato para se conectar a uma instância de banco de dados.

```
vertica://jdbc:vertica://host_name:port/database?user=vertica-
username&password=vertica-password
```

Uso de um único manipulador de conexão

É possível usar os seguintes metadados de conexão única e manipuladores de registros para se conectar a uma única instância do Vertica.

Tipo de manipulador	Classe
Manipulador composto	`VerticaCompositeHandler`
Manipulador de metadados	`VerticaMetadataHandler`
Manipulador de registros	`VerticaRecordHandler`

Parâmetros do manipulador de conexão única

Parâmetro	Descrição
`default`	Obrigatório. A string de conexão padrão.

Os manipuladores de conexão únicos oferecem suporte a uma instância de banco de dados e devem fornecer um parâmetro de string de conexão `default`. Todas as outras strings de conexão são ignoradas.

Fornecimento de credenciais

Para fornecer um nome de usuário e uma senha para seu banco de dados na string de conexão JDBC, é possível usar as propriedades da string de conexão ou o AWS Secrets Manager.

- String de conexão: um nome de usuário e uma senha podem ser especificados como propriedades na string de conexão do JDBC.

> **Important**
>
> Como prática recomendada de segurança, não use credenciais codificadas em suas variáveis de ambiente ou strings de conexão. Para obter informações sobre como mover

seus segredos codificados para o AWS Secrets Manager, consulte Mover segredos codificados para o AWS Secrets Manager no Guia do usuário do AWS Secrets Manager.

- AWS Secrets Manager: para usar o recurso Athena Federated Query com o AWS Secrets Manager, a VPC conectada à sua função do Lambda deve ter acesso à Internet ou um endpoint da VPC para se conectar ao Secrets Manager.

  É possível colocar o nome de um segredo no AWS Secrets Manager na sua string de conexão JDBC. O conector substitui o nome secreto pelos valores de `username` e `password` do Secrets Manager.

  Para instâncias de banco de dados do Amazon RDS, esse suporte é totalmente integrado. Se você usa o Amazon RDS, é altamente recomendável usar o AWS Secrets Manager e rotação de credenciais. Se seu banco de dados não usar o Amazon RDS, armazene as credenciais em JSON no seguinte formato:

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

Exemplo de string de conexão com nomes secretos

A string a seguir tem os nomes secretos ${`vertica-username`} e ${`vertica-password`}.

```
vertica://jdbc:vertica://host_name:port/database?user=${vertica-username}&password=
${vertica-password}
```

O conector usa o nome secreto para recuperar segredos e fornecer o nome de usuário e a senha, como no exemplo a seguir.

```
vertica://jdbc:vertica://host_name:port/database?user=sample-user&password=sample-
password
```

Atualmente, o conector Vertica reconhece as propriedades `vertica-username` e `vertica-password` do JDBC.

Parâmetros de derramamento

O SDK do Lambda pode derramar dados no Amazon S3. Todas as instâncias do banco de dados acessadas pela mesma função do Lambda derramam no mesmo local.

Parâmetro	Descrição
`spill_bucket`	Obrigatório. Nome do bucket de derramamento.
`spill_prefix`	Obrigatório. Prefixo de chave do bucket de derramamento.
`spill_put_request_headers`	(Opcional) Um mapa codificado em JSON de cabeçalhos e valores de solicitações para a solicitação `putObject` do Amazon S3 usada para o derramamento (por exemplo, `{"x-amz-server-side-encryption" : "AES256"}`). Para outros cabeçalhos possíveis, consulte PutObject na Referência da API do Amazon Simple Storage Service.

Suporte ao tipo de dados

A tabela a seguir mostra os tipos de dados com suporte pelo conector Vertica.

Booleano
BigInt
Short
Inteiro
Longo
Float
Double
Data
Varchar
Bytes
BigDecimal

Booleano
TimeStamp como Varchar

Performance

A função do Lambda realiza o empilhamento de projeções para diminuir os dados verificados pela consulta. As cláusulas `LIMIT` reduzem a quantidade de dados verificados, mas se você não fornecer um predicado, deverá aguardar que as consultas `SELECT` com uma cláusula `LIMIT` verifiquem, no mínimo, 16 MB de dados. O conector Vertica é resiliente ao controle de utilização devido à simultaneidade.

Consultas de passagem

O conector Vertica é compatível com consultas de passagem. As consultas de passagem usam uma função de tabela para enviar sua consulta completa para execução na fonte de dados.

Para usar consultas de passagem com o Vertica, você pode empregar a seguinte sintaxe:

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

O exemplo de consulta a seguir envia uma consulta para uma fonte de dados no Vertica. A consulta seleciona todas as colunas na tabela `customer`, limitando os resultados a 10.

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

Informações de licença

Ao usar esse conector, você reconhece a inclusão de componentes de terceiros, cuja lista pode ser encontrada no arquivo pom.xml desse conector, e concorda com os termos das respectivas licenças de terceiros fornecidas no arquivo LICENSE.txt em GitHub.com.

Recursos adicionais do

Para obter as informações mais recentes sobre a versão do driver JDBC, consulte o arquivo pom.xml do conector Vertica em GitHub.com.

Para obter mais informações sobre esse conector, consulte o site correspondente em GitHub.com e Consulta de uma fonte de dados do Vertica no Amazon Athena usando o SDK Athena Federated Query no Blog de big data da AWS.

Implantar um conector de fonte de dados

A preparação para criar consultas federadas é um processo de duas partes: implantar um conector de origem dos dados da função do Lambda e conectar a função do Lambda a uma origem dos dados. Nesse processo, você dá um nome à função do Lambda que possa escolher posteriormente no console do Athena e dá um nome ao conector que você possa referenciar em suas consultas SQL.

> **Note**
>
> Para usar o recurso de consulta federada do Athena com o AWS Secrets Manager, configure um endpoint privado do Amazon VPC para o Secrets Manager. Para obter mais informações, consulte Criação de um endpoint privado da VPC para o Secrets Manager no Guia do usuário do AWS Secrets Manager.

Tópicos

- Usar o console do Athena
- Usar o AWS Serverless Application Repository para implantar um conector de origem de dados
- Criar uma VPC para um conector de origem de dados
- Habilitar consultas federadas entre contas
- Atualizar um conector de fonte de dados

Usar o console do Athena

Para escolher, especificar um nome e implantar um conector de origem dos dados, use os consoles do Athena e do Lambda em um processo integrado.

Como implantar um conector de fonte de dados

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.

3. No painel de navegação, escolha Origens dos dados.
4. Na página Data sources (Origens de dados), escolha Create data source (Criar origem dos dados).
5. Em Choose a data source (Escolher uma origem dos dados), escolha a origem dos dados que o Athena deve consultar, considerando as seguintes diretrizes:

   - Escolha uma opção de consulta federada que corresponda à sua origem dos dados. O Athena tem conectores de origem dos dados pré-criados que você pode configurar para origens que incluem o MySQL, o Amazon DocumentDB e o PostgreSQL.
   - Escolha S3 - AWS Glue Data Catalog se quiser consultar dados no Amazon S3 e não estiver usando um metastore do Apache Hive ou uma das outras opções de origem dos dados de consulta federada nesta página. O Athena usa o AWS Glue Data Catalog para armazenar metadados e informações de esquemas para origens dos dados no Amazon S3. Essa é a opção padrão (não federada). Para ter mais informações, consulte Usar o AWS Glue para se conectar a origens de dados no Amazon S3.
   - Selecione S3 - Apache Hive metastore para consultar conjuntos de dados no Amazon S3 que usam um metastore do Apache Hive. Para obter mais informações sobre essa opção, consulte Conectar o Athena a um metastore do Apache Hive.

- Escolha Custom or shared connector (Conector personalizado ou compartilhado) se quiser criar seu próprio conector de origem dos dados para usar com o Athena. Para obter informações sobre como escrever um conector de origem dos dados, consulte Desenvolver um conector de fonte de dados usando o SDK do Athena Query Federation.

Este tutorial escolhe o Amazon CloudWatch Logs como origem dos dados federada.

6. Escolha Próximo.
7. Na página Enter data source details (Inserir detalhes da origem dos dados), em Data source name (Nome da origem dos dados), insira o nome que deseja usar em suas instruções SQL ao consultar a origem dos dados pelo Athena (por exemplo, `CloudWatchLogs`). O nome pode ter até 127 caracteres e deve ser exclusivo na sua conta. Ele não poderá ser alterado após a criação. Os caracteres válidos são a-z, A-Z, 0-9, _ (sublinhado), @ (arroba) e - (hífen). Os nomes `awsdatacatalog`, `hive`, `jmx` e `system` são reservados pelo Athena e não podem ser usados como nomes de origens dos dados.
8. Em Lambda function (Função do Lambda), escolha Create Lambda function (Criar função do Lambda). A página de função do conector escolhido será aberta no console do AWS Lambda. A página inclui informações detalhadas sobre o conector.
9. Em Application settings (Configurações da aplicação), leia atentamente a descrição de cada configuração de aplicação e insira os valores de acordo com os seus requisitos.

   As configurações de aplicação exibidas variam dependendo do conector da sua origem dos dados. As configurações mínimas necessárias são:

   - AthenaCatalogName: um nome para a função do Lambda, em letras minúsculas, que indica a origem dos dados desejada, como `cloudwatchlogs`.
   - SpillBucket: um bucket do Amazon S3 em sua conta para armazenar os dados que excedem os limites de tamanho de resposta da função do Lambda.

   > **Note**
   >
   > Dados derramados não são reutilizados em execuções subsequentes e podem ser excluídos com segurança depois de 12 horas. O Athena não os exclui para você. Para gerenciar esses objetos, considere adicionar uma política de ciclo de vida de objetos que exclua dados antigos do seu bucket de derramamento do Amazon S3. Para obter

mais informações, consulte Gerenciar ciclo de vida de armazenamento no Manual do usuário do Amazon S3.

10. Selecione Reconheço que este aplicativo cria perfis personalizadas do IAM e políticas de recursos. Para obter mais informações, escolha o link Informações.
11. Escolha Implantar. Quando a implantação for concluída, a função do Lambda será exibida seção Resources (Recursos) no console do Lambda.

Conectar à fonte de dados

Depois de implantar o conector da origem dos dados em sua conta, você poderá conectar o Athena a ele.

Para conectar o Athena a uma origem dos dados usando um conector que você implantou em sua conta

1. Retorne à página Enter data source details (Inserir detalhes da origem dos dados) do console do Athena.
2. Na seção Connection details (Detalhes da conexão), escolha o ícone de atualização ao lado da caixa de pesquisa Select or enter a Lambda function (Selecionar ou inserir uma função do Lambda).
3. Escolha o nome da função que você acabou de criar no console do Lambda. O ARN da função do Lambda é exibido.
4. (Opcional) Para Tags, adicione pares de chave-valor a associar com essa origem dos dados. Para obter mais informações sobre tags, consulte Etiquetar recursos do Athena.
5. Escolha Próximo.
6. Na página Review and create (Revisar e criar), analise os detalhes da origem dos dados e escolha Create data source (Criar origem dos dados).
7. A seção Data source details (Detalhes da origem dos dados) da página de sua origem dos dados mostra informações sobre o novo conector. Agora é possível usar o conector em suas consultas do Athena.

   Para obter informações sobre como usar conectores de dados em consultas, acesse Executar consultas federadas.

Usar o AWS Serverless Application Repository para implantar um conector de origem de dados

Para implantar um conector de origem dos dados, você pode usar o AWS Serverless Application Repository em vez de começar com o console do Athena. Use AWS Serverless Application Repository para encontrar o conector que deseja usar, forneça os parâmetros que o conector exige e implante o conector em sua conta. Depois de implantar o conector, você usa o console do Athena para disponibilizar a origem dos dados para o Athena.

Implantar o conector em sua conta

Como usar o AWS Serverless Application Repository para implantar um conector de fonte de dados em sua conta

1. Faça login no AWS Management Console e abra o Repositório de aplicativos sem servidor.
2. No painel de navegação, escolha Aplicativos disponíveis.
3. Selecione a opção Show apps that create custom IAM roles or resource policies (Mostrar aplicações que criam funções personalizadas do IAM ou políticas de recursos).
4. Na caixa de pesquisa, digite o nome do conector. Para obter uma lista de conectores de dados predefinidos do Athena, consulte Conectores de fonte de dados disponíveis.
5. Escolha o nome do conector. Ao escolher um conector, a página Application details (Detalhes da aplicação) da função do Lambda no console do AWS Lambda é aberta.
6. No lado direito da página de detalhes, em Application settings (Configurações da aplicação), preencha as informações necessárias. As configurações mínimas necessárias são mostradas a seguir. Para obter informações sobre as opções configuráveis restantes de conectores de dados criados pelo Athena, consulte o tópico Available Connectors (Conectores disponíveis) no GitHub.

   - AthenaCatalogName: um nome para a função do Lambda em letras minúsculas que indica a origem dos dados desejada, como `cloudwatchlogs`.
   - SpillBucket: especifique um bucket do Amazon S3 em sua conta para receber os dados de qualquer carga útil de resposta grande que exceda os limites de tamanho de resposta da função do Lambda.
7. Selecione I acknowledge that this app creates custom IAM roles and resource policies (Reconheço que este aplicativo cria funções personalizadas do IAM e políticas de recursos). Para obter mais informações, escolha o link Informações.
8. Na parte inferior direita da página Configurações da aplicação selecione Implantar. Quando a implantação for concluída, a função do Lambda será exibida seção Resources (Recursos) no console do Lambda.

Disponibilizar o conector no Athena

Agora você já pode usar o console do Athena para disponibilizar o conector da origem dos dados para o Athena.

Para disponibilizar o conector da origem dos dados para o Athena

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.

3. No painel de navegação, escolha Origens dos dados.
4. Na página Data sources (Origens de dados), escolha Create data source (Criar origem dos dados).
5. Em Choose a data source (Escolher uma origem dos dados), escolha a origem dos dados para a qual você criou um conector no AWS Serverless Application Repository. Este tutorial usa o Amazon CloudWatch Logs como origem dos dados federada.
6. Escolha Próximo.
7. Na página Enter data source details (Inserir detalhes da origem dos dados), em Data source name (Nome da origem dos dados), insira o nome que deseja usar em suas instruções SQL ao consultar a origem dos dados pelo Athena (por exemplo, `CloudWatchLogs`). O nome pode ter até 127 caracteres e deve ser exclusivo na sua conta. Ele não poderá ser alterado após a criação. Os caracteres válidos são a-z, A-Z, 0-9, _ (sublinhado), @ (arroba) e - (hífen). Os nomes `awsdatacatalog`, `hive`, `jmx` e `system` são reservados pelo Athena e não podem ser usados como nomes de origens dos dados.

8. Na seção Connection details (Detalhes da conexão), use a caixa Select or enter a Lambda function (Selecionar ou inserir uma função do Lambda) para escolher o nome da função que você acabou de criar. O ARN da função do Lambda é exibido.
9. (Opcional) Para Tags, adicione pares de chave-valor a associar com essa origem dos dados. Para obter mais informações sobre tags, consulte Etiquetar recursos do Athena.
10. Escolha Próximo.
11. Na página Review and create (Revisar e criar), analise os detalhes da origem dos dados e escolha Create data source (Criar origem dos dados).
12. A seção Data source details (Detalhes da origem dos dados) da página de sua origem dos dados mostra informações sobre o novo conector. Agora é possível usar o conector em suas consultas do Athena.

    Para obter informações sobre como usar conectores de dados em consultas, acesse Executar consultas federadas.

Criar uma VPC para um conector de origem de dados

Alguns conectores de origem dos dados do Athena exigem uma VPC e um grupo de segurança. Este tópico mostra como criar uma VPC com uma sub-rede e um grupo de segurança para a VPC. Como parte desse processo, você recupera os IDs da VPC, da sub-rede e do grupo de segurança criados. Esses IDs são necessários quando você configura o conector para uso com o Athena.

Para criar uma VPC para um conector de origem de dados do Athena

1. Faça login no AWS Management Console e abra o console do Amazon VPC em https://console.aws.amazon.com/vpc/.
2. No painel de navegação, confirme que a opção New VPC Experience (Experiência da nova VPC) está selecionada.
3. Escolha Criar VPC.
4. Em VPC Settings (Configurações da VPC), escolha VPC and more (VPC e mais) em Resources to create (Recursos a serem criados).
5. Na opção Auto-generate (Gerar automaticamente), insira o valor que será usado para gerar as etiquetas de nome para todos os recursos na VPC.
6. Escolha Criar VPC.
7. Escolha View VPC (Visualizar VPC).

8. Na seção Details (Detalhes), em VPC ID (ID da VPC), copie o ID da VPC para referência posterior.

Agora você já pode recuperar o ID de sub-rede da VPC que acabou de criar.

Para recuperar o ID de sub-rede da VPC

1. No painel de navegação do console da VPC, escolha Subnets (Sub-redes).
2. Escolha o nome correspondente à sub-rede que você criou.
3. Na seção Details (Detalhes), em Subnet ID (ID da sub-rede), copie o ID da sub-rede para referência posterior.

Em seguida, crie um grupo de segurança para a VPC.

Para criar um grupo de segurança para a VPC

1. No painel de navegação do console da VPC, escolha Security (Segurança), Security groups (Grupos de segurança).
2. Escolha Create security group (Criar grupo de segurança).
3. Na página Create Security Group (Criar grupo de segurança), insira as informações a seguir:
   - Em Security group name (Nome do grupo de segurança), insira um nome para o grupo de segurança.
   - Em Description (Descrição), insira uma descrição para o grupo de segurança. Este campo é obrigatório.
   - Em VPC, insira o ID da VPC criada por você para o conector da origem dos dados.
   - Em Inbound rules (Regras de entrada) e Outbound rules (Regras de saída), adicione todas as regras de entrada e saída que precisar.
4. Escolha Create security group (Criar grupo de segurança).
5. Na página Details (Detalhes) do grupo de segurança, copie o Security group ID (ID do grupo de segurança) para referência posterior.

Habilitar consultas federadas entre contas

A consulta federada permite consultar as origens de dados que não são do Amazon S3 usando os conectores de origem de dados implantados no AWS Lambda. O recurso de consulta federada

entre contas permite que a função do Lambda e as origens de dados a serem consultadas estejam localizadas em contas diferentes.

Como administrador de dados, você pode habilitar consultas federadas entre contas compartilhando seu conector de dados com uma conta de analista de dados ou, como analista de dados, usando um ARN do Lambda compartilhado de um administrador de dados para adicionar à sua conta. Quando alterações de configuração são feitas em um conector na conta de origem, a configuração atualizada é aplicada automaticamente às instâncias compartilhadas do conector nas outras contas de usuário.

Considerações e limitações

- O recurso de consulta federada entre contas está disponível para conectores de dados de metastore não Hive que usam uma origem de dados baseada no Lambda.
- O recurso não está disponível para o tipo de origem de dados AWS Glue Data Catalog. Para obter informações sobre o acesso entre contas aos AWS Glue Data Catalogs, consulte Acesso aos catálogos de dados do AWS Glue entre contas.
- Se a resposta da função do Lambda do conector excede o limite de tamanho de resposta do Lambda de 6 MB, o Athena automaticamente criptografa, agrupa e distribui a resposta para um bucket do Amazon S3 que você configurou. A entidade que executa a consulta do Athena deve ter acesso ao local do vazamento para que o Athena leia os dados vazados. Recomendamos definir uma política de ciclo de vida do Amazon S3 para excluir objetos do local do vazamento, pois os dados não serão necessários após a conclusão da consulta.
- Não há suporte para o uso de consultas federadas em Regiões da AWS.

Permissões obrigatórias

- Para que a Conta A de administrador de dados compartilhe uma função do Lambda com a Conta B de analista de dados, a Conta B requer a função invoke do Lambda e acesso ao bucket de vazamento. Consequentemente, a Conta A deve adicionar uma política baseada em recursos à função do Lambda e acesso do principal ao bucket de vazamento no Amazon S3.
  1. A política a seguir concede as permissões da função invoke do Lambda à Conta B em uma função do Lambda na conta 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. A política a seguir concede acesso ao bucket de vazamento a entidade principal na conta 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. Se a função do Lambda criptografar o bucket de vazamento com uma chave AWS KMS, em vez de usar a criptografia padrão oferecida pelo SDK da federação, a política de chave AWS KMS na conta A deverá conceder acesso ao usuário na Conta B, como no exemplo a seguir.

```
{
    "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.
}
```

- Para que a Conta A compartilhe seu conector com a Conta B, a Conta B deve criar uma função chamada `AthenaCrossAccountCreate-`*`account-A-id`*, que a Conta A assume chamando a ação de API AssumeRole do AWS Security Token Service.

  A política a seguir, que permite a ação `CreateDataCatalog`, deve ser criada na Conta B e adicionada à função `AthenaCrossAccountCreate-`*`account-A-id`* que a Conta B cria para a Conta A.

```
{
 "Effect": "Allow",
 "Action": "athena:CreateDataCatalog",
 "Resource": "arn:aws:athena:*:account-B-id:datacatalog/*"
}
```

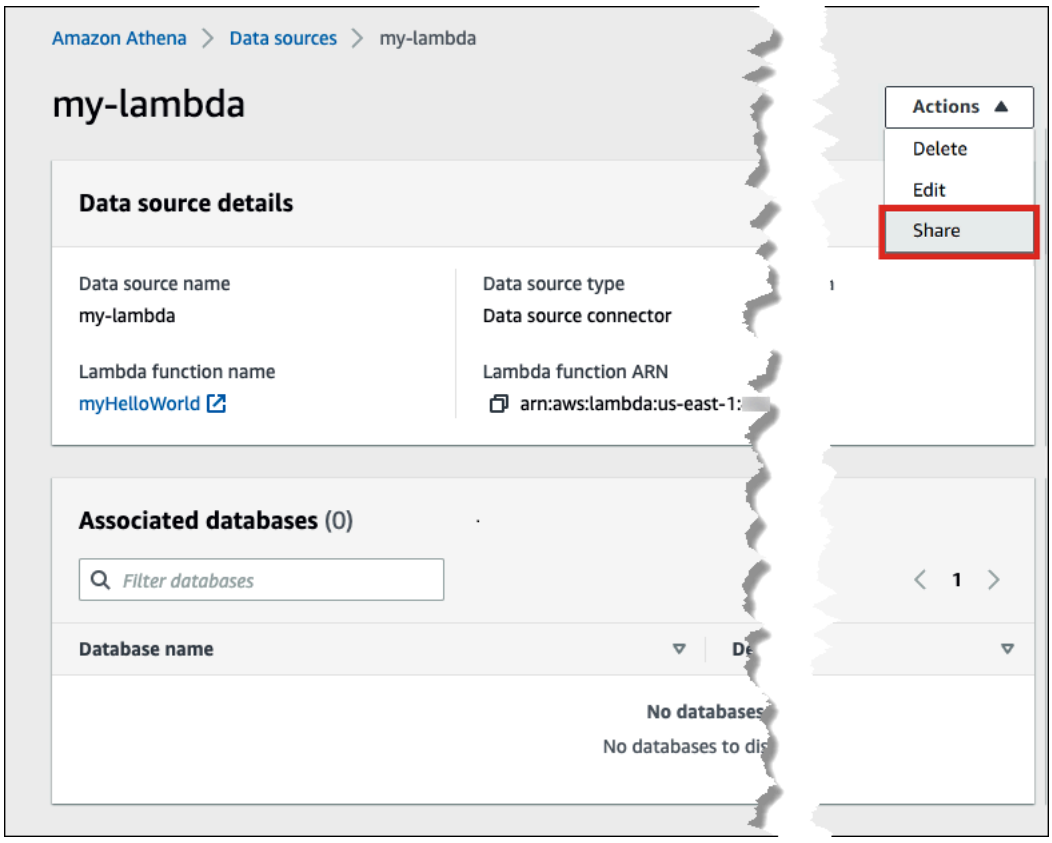
Compartilhar uma origem de dados na conta A com a conta B

Depois que as permissões estiverem em vigor, você pode usar a página Data sources (Origens de dados) no console do Athena para compartilhar um conector de dados em sua conta (Conta A) com outra conta (Conta B). A conta A mantém total controle e propriedade do conector. Quando a conta A faz alterações de configuração no conector, a configuração atualizada se aplica ao conector compartilhado na conta B.

Para compartilhar uma origem de dados do Lambda na conta A com a conta B

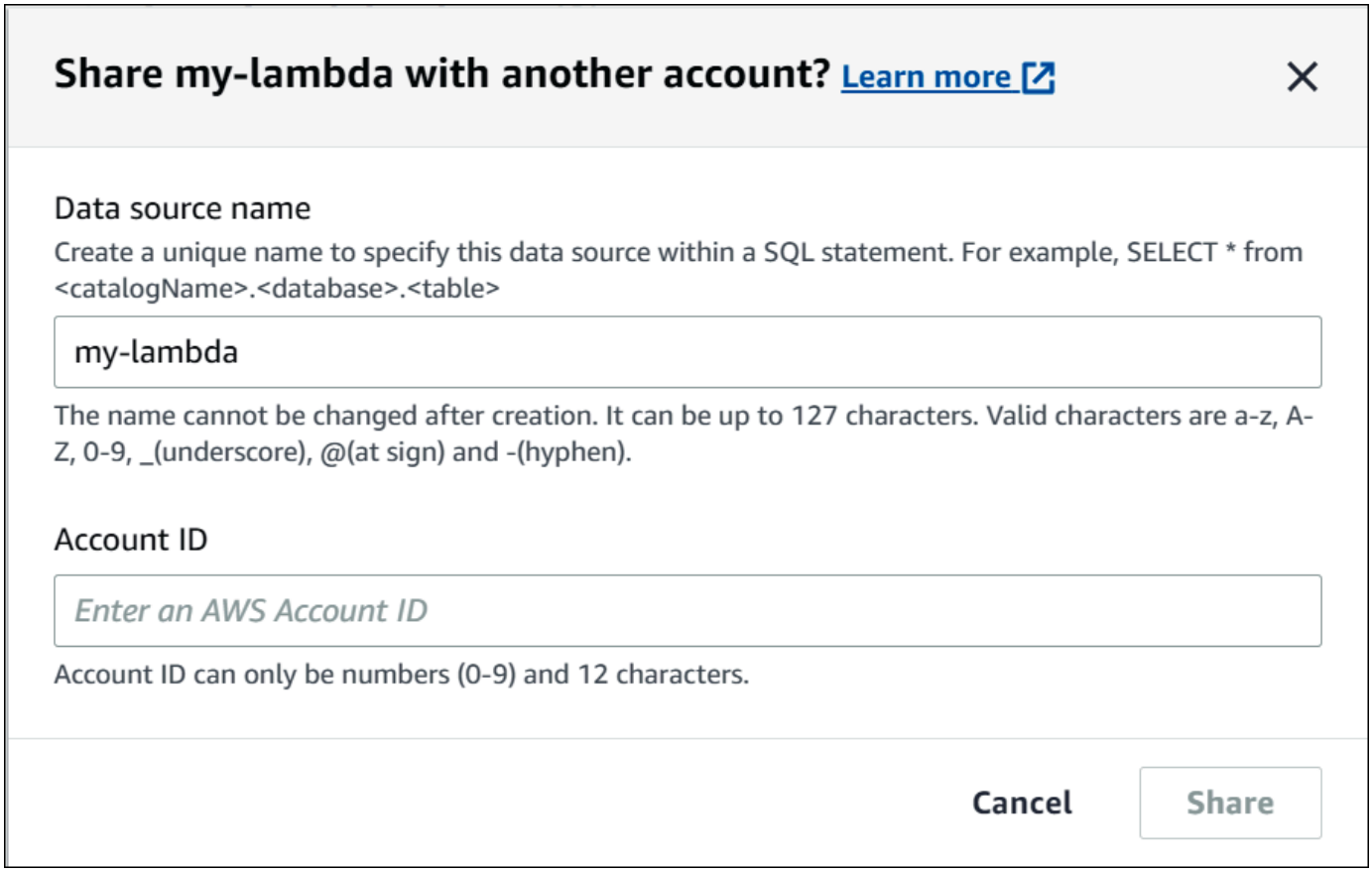
1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.

3. Escolha Data sources (Origens de dados).
4. Na página Data sources (Origens de dados), escolha o link do conector que deseja compartilhar.
5. Na página de detalhes de uma origem dos dados do Lambda, escolha a opção Share (Compartilhar) no canto superior direito.



6. Na caixa de diálogo Share ***nome do Lambda*** with another account (Compartilhar nome-do-Lambda com outra conta), insira as informações necessárias.

   - Para Data source name (Nome da origem dos dados), insira o nome da origem de dados copiada como você deseja que apareça na outra conta.
   - Para Account ID, (ID da conta), insira o ID da conta com a qual deseja compartilhar sua origem dos dados (neste caso, Conta B).



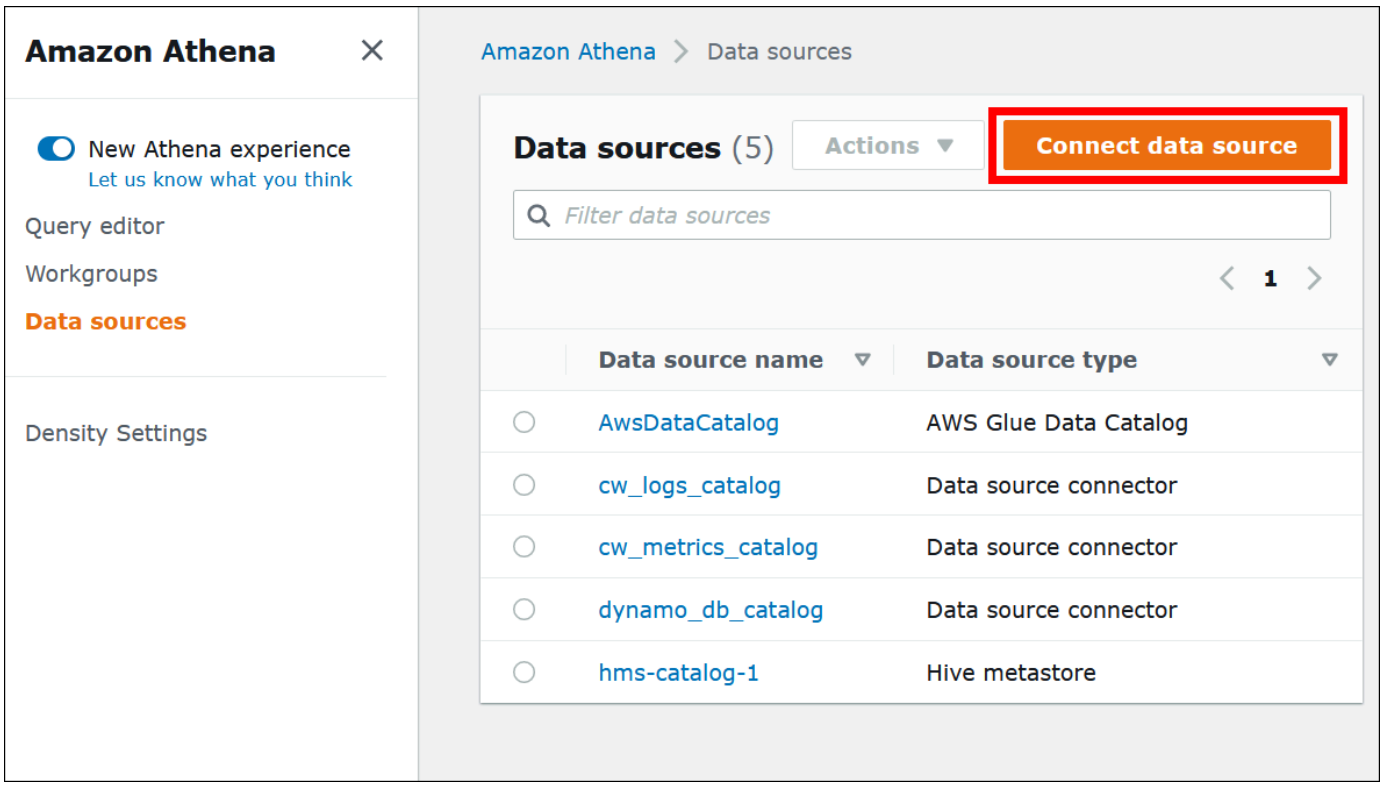
7. Selecione Share. O conector de dados compartilhados que você especificou é criado na Conta B. As alterações de configuração do conector na Conta A se aplicam ao conector na Conta B.

Adicionar uma origem de dados compartilhada da conta A à conta B

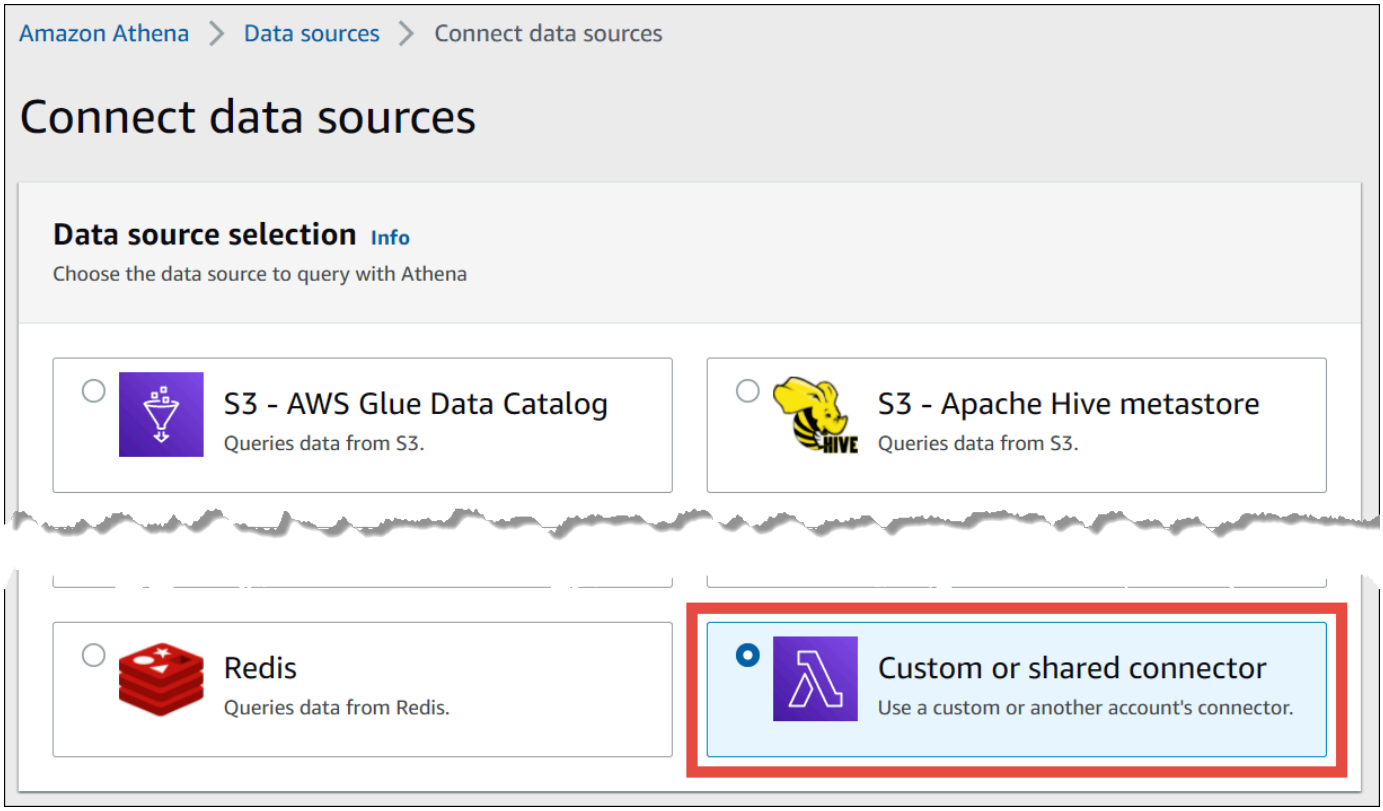
Como analista de dados, você pode receber, de um administrador de dados, o ARN de um conector para adicionar à sua conta. Você pode usar a página Data sources (Origem dos dados) do console do Athena para adicionar o ARN do Lambda fornecido pelo administrador à sua conta.

Para adicionar o Lambda ARN de um conector de dados compartilhado à sua conta

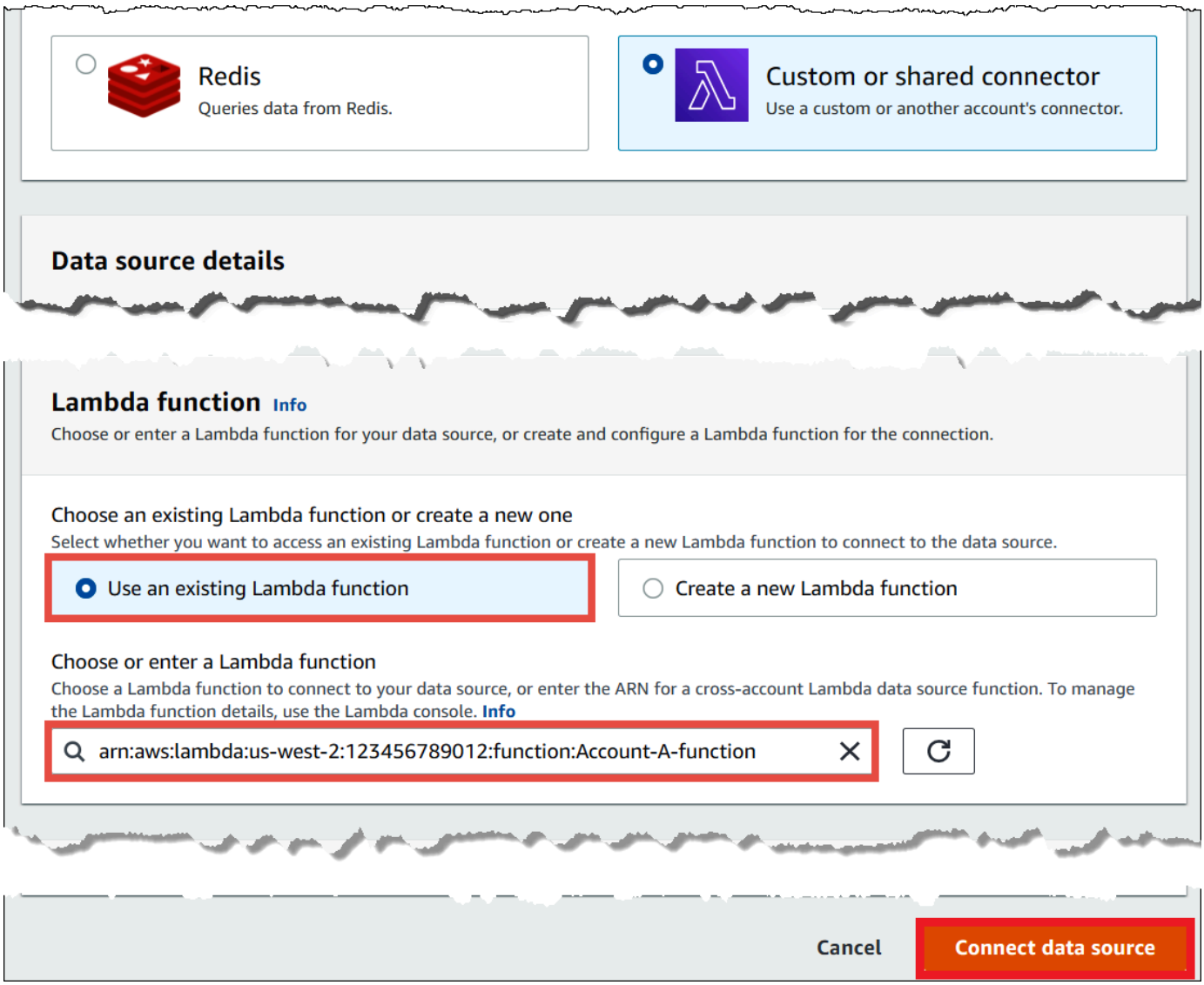
1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se você estiver usando a nova experiência de console e o painel de navegação não estiver visível, escolha o menu de expansão à esquerda.
3. Escolha Data sources (Origens de dados).
4. Na página Data sources (Origens de dados), escolha Connect data source (Conectar origem dos dados).



5. Selecione Custom or shared connector (Conector personalizado ou compartilhado).



6. Na seção Lambda function (Função do Lambda), certifique-se de que a opção Use an existing Lambda function (Usar uma função do Lambda existente) esteja selecionada.



7. Para Choose or enter a Lambda function (Escolher ou inserir uma função do Lambda) insira o ARN do Lambda da Conta A.
8. Escolha Connect data source (Conectar fonte de dados).

Solução de problemas

Se você receber uma mensagem de erro informando que a Conta A não tem permissões para assumir uma função na Conta B, certifique-se de que o nome da função criada na Conta B esteja escrito corretamente e ela que tenha a política adequada anexada.

Atualizar um conector de fonte de dados

O Athena recomenda atualizar regularmente os conectores da fonte de dados que você usa para a versão mais recente para aproveitar os novos recursos e aprimoramentos. Para começar, é necessário localizar o número da versão mais recente.

Localizar a versão mais recente do Athena Query Federation

O número da versão mais recente dos conectores de fonte de dados do Athena corresponde à versão mais recente do Athena Query Federation. Em certos casos, as versões do GitHub podem ser um pouco mais recentes do que as disponíveis no AWS Serverless Application Repository (SAR).

Para localizar o número da versão mais recente do Athena Query Federation

1. Acesse o URL do GitHub https://github.com/awslabs/aws-athena-query-federation/releases/latest.
2. Observe o número da versão no título da página principal no seguinte formato:

   Versão v *year.week_of_year.iteration_of_week* do Athena Query Federation

   Por exemplo, o número da versão v2023.8.3 do Athena Query Federation é 2023.8.3.

Localizar e anotar nomes de recursos

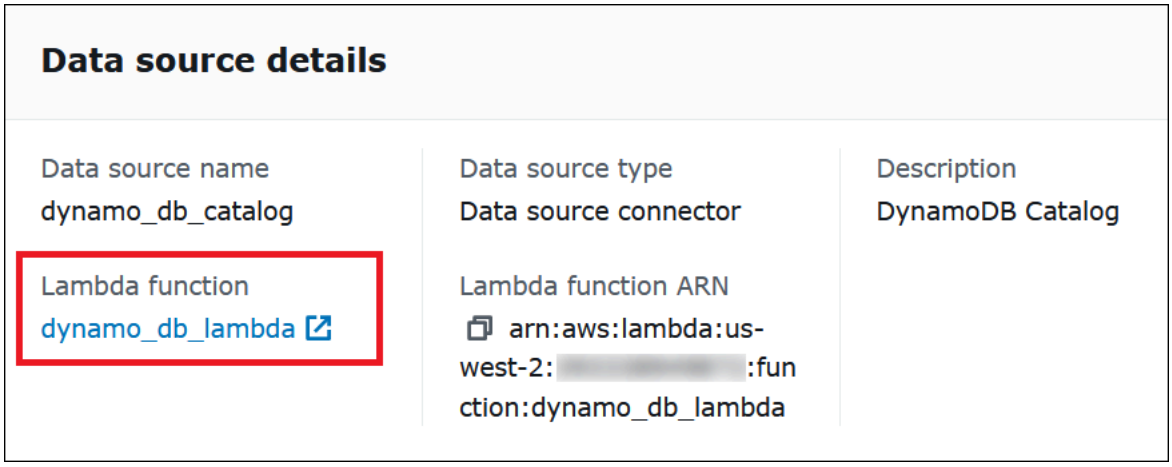
Na preparação para o upgrade, é necessário localizar e anotar as seguintes informações:

1. O nome da função do Lambda para o conector.
2. As variáveis de ambiente da função do Lambda.
3. O nome da aplicação do Lambda, que gerencia a função do Lambda para o conector.

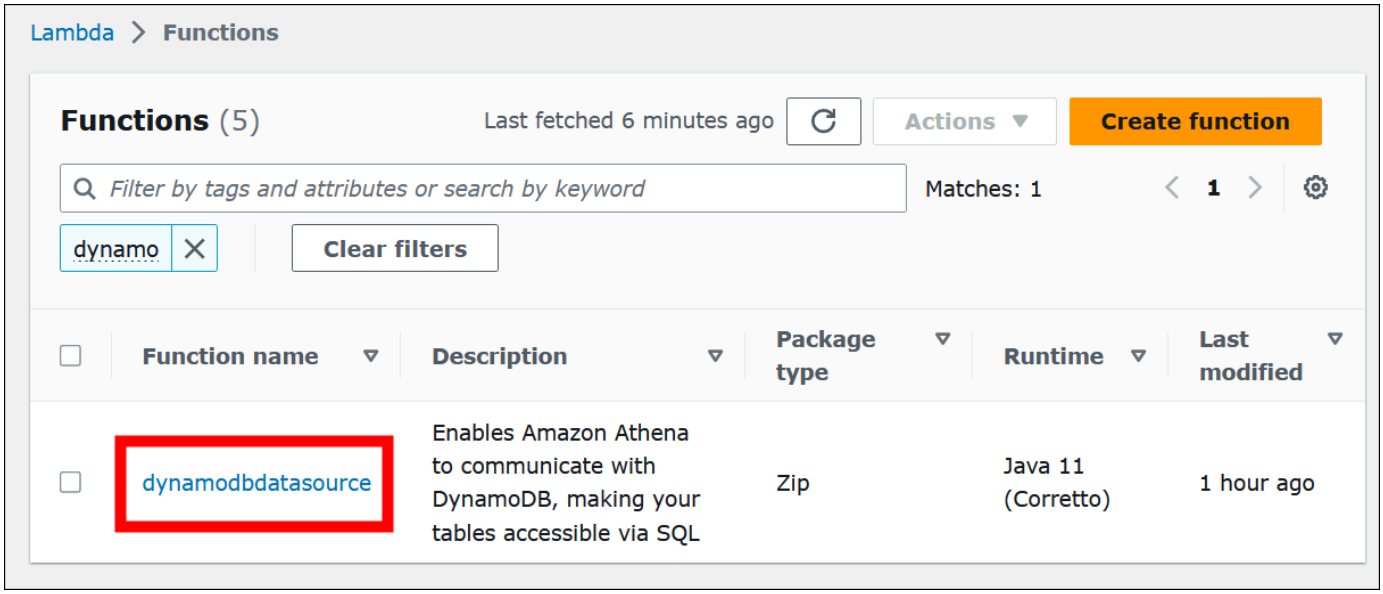
Para localizar nomes de recursos no console do Athena

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.

3. No painel de navegação, escolha Origens dos dados.
4. Na coluna Nome da fonte de dados, escolha o link para a fonte de dados do conector.
5. Na seção Detalhes da fonte de dados, em Função do Lambda, escolha o link para a função do Lambda.



6. Na página Funções, na coluna Nome da função, anote o nome da função do conector.



7. Escolha o link do nome da função.
8. Na seção Visão geral da função, escolha a guia Configuração.
9. No painel à esquerda, escolha Variáveis de ambiente.
10. Na seção Variáveis de ambiente, anote as chaves e os valores correspondentes.
11. Mova a barra de rolagem até o topo da página.
12. Na mensagem Esta função pertence a uma aplicação. Clique aqui para gerenciá-la, escolha o link Clique aqui.
13. Na página serverlessrepo-***your_application_name***, anote o nome da aplicação sem serverlessrepo. Por exemplo, se o nome da aplicação for serverlessrepo-DynamoDbTestApp, o nome da aplicação será DynamoDbTestApp.
14. Permaneça na página do console do Lambda de sua aplicação e continue com as etapas descritas em Localizar a versão do conector que você está usando.

Localizar a versão do conector que você está usando

Siga estas etapas para localizar a versão do conector que você está usando.

Para localizar a versão do conector que você está usando

1. Na página do console do Lambda de sua aplicação do Lambda, escolha a guia Implantações.
2. Na guia Implantações, expanda o Modelo do SAM.
3. Pesquise CodeUri.

4. No campo Chave, em CodeUri, localize a seguinte string:

```
applications-connector_name-
versions-year.week_of_year.iteration_of_week/hash_number
```

   O exemplo a seguir mostra uma string do conector do CloudWatch:

```
applications-AthenaCloudwatchConnector-versions-2021.42.1/15151159...
```

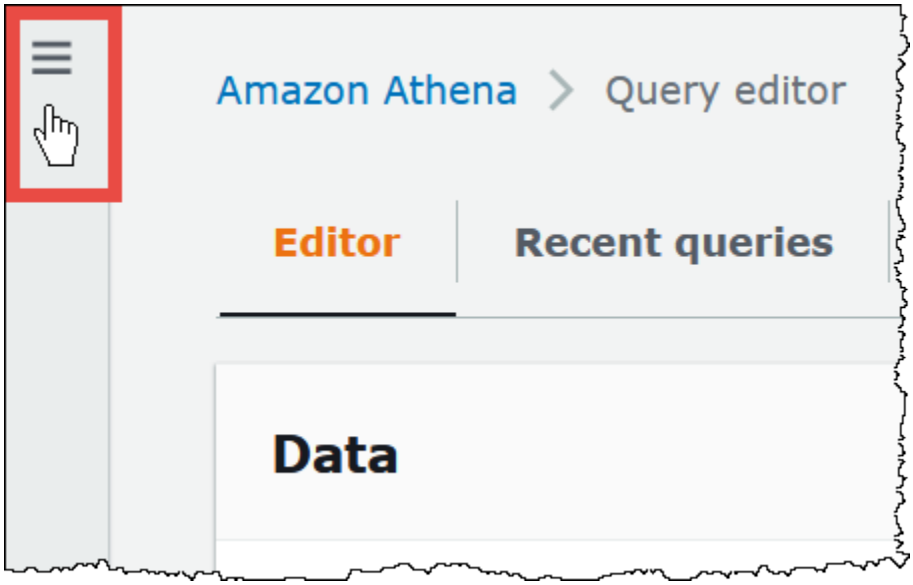
5. Registre o valor de *year.week_of_year.iteration_of_week* (por exemplo, 2021.42.1). Esta é a versão de seu conector.

Implantar a nova versão de seu conector

Siga as etapas a seguir para implantar uma nova versão do conector.

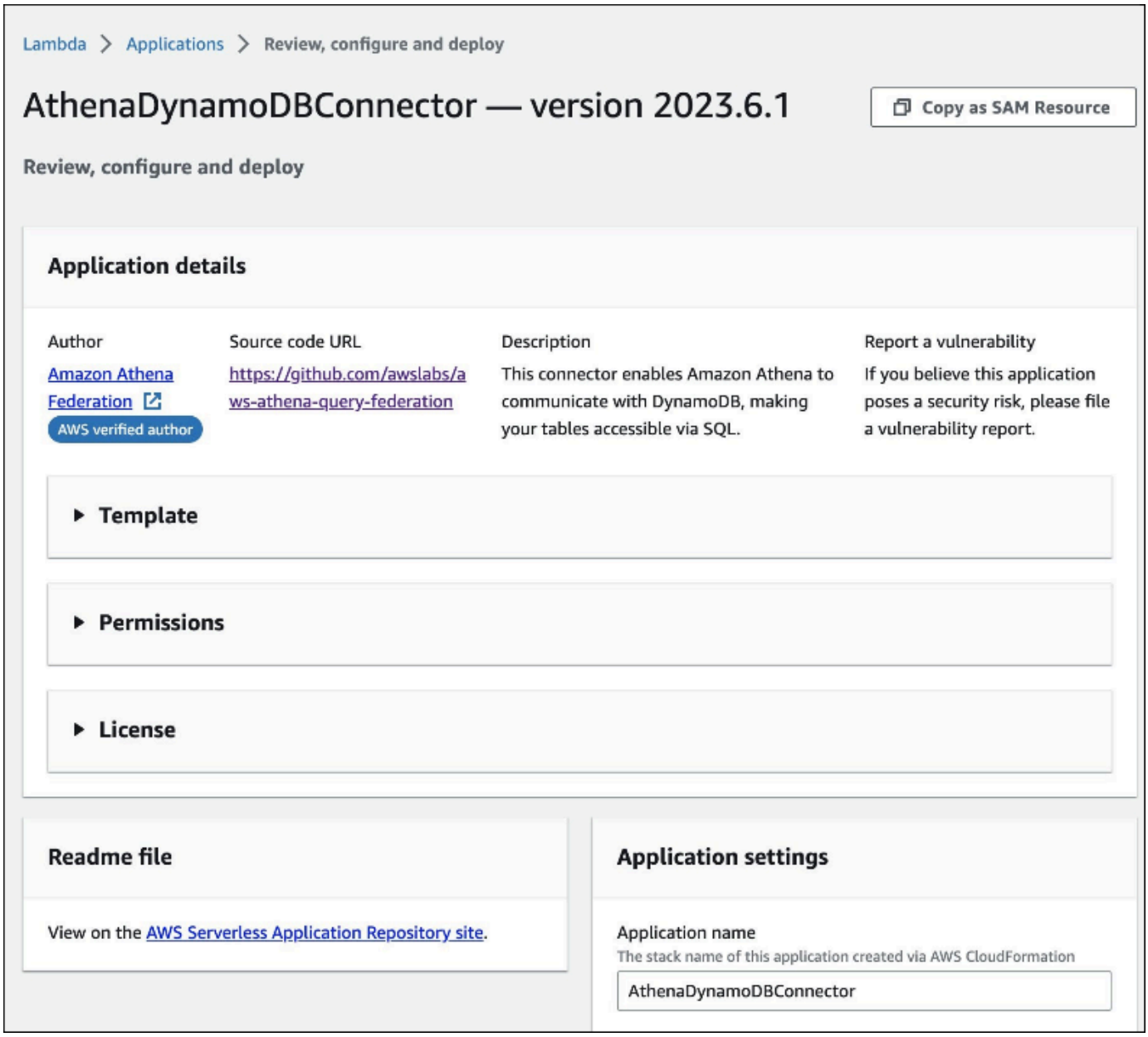
Para implantar uma nova versão de seu conector

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.



3. No painel de navegação, escolha Origens dos dados.
4. Na página Data sources (Origens de dados), escolha Create data source (Criar origem dos dados).
5. Escolha a fonte de dados que deseja atualizar e escolha Próximo.

6. Na seção Detalhes da conexão, escolha Criar função do Lambda. Isso abre o console do Lambda, no qual você poderá implantar a aplicação atualizada.



7. Como você não está de fato criando uma nova fonte de dados, pode fechar a guia do console do Athena.
8. Na página do console do Lambda do conector, execute as seguintes etapas:

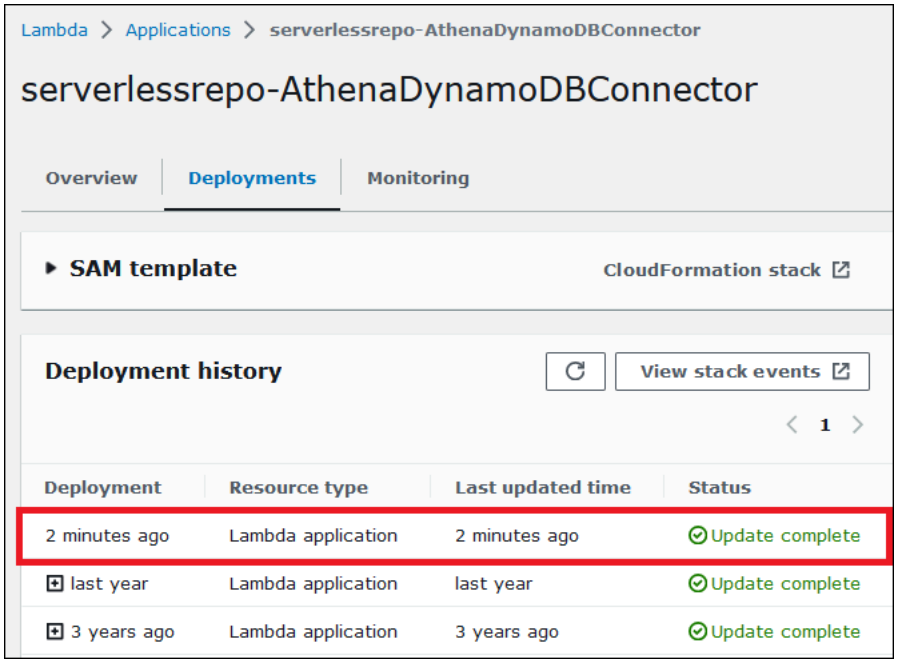
   a. Verifique se removeu o prefixo serverlessrepo- do nome da aplicação e copie o nome da aplicação no campo Nome da aplicação.

   b. Copie o nome da função do Lambda no campo AthenaCatalogName. Alguns conectores chamam esse campo de LambdaFunctionName.

c. Copie as variáveis de ambiente que você registrou nos campos correspondentes.

9. Selecione a opção Eu reconheço que esta aplicação cria perfis do IAM personalizados e políticas de recursos e escolha Implantar.
10. Para verificar se a aplicação foi atualizada, escolha a guia Implantações.

    A seção Histórico de implantações mostra que sua atualização foi concluída.



11. Para confirmar o novo número da versão, você pode expandir Modelo do SAM como antes, localizar o CodeUri e verificar o número da versão do conector no campo Chave.

Agora é possível usar seu conector atualizado para criar consultas federadas do Athena.

Executar consultas federadas

Depois de configurar um ou mais conectores de dados e implantá-los em sua conta, você poderá usá-los nas consultas do Athena.

Consultar uma única origem de dados

Os exemplos nesta seção pressupõem que você tenha configurado e implantado Conector do Amazon Athena para o CloudWatch em sua conta. Use a mesma abordagem para consultar quando usar outros conectores.

Para criar uma consulta do Athena que usa o conector do CloudWatch

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. No editor de consultas do Athena, crie uma consulta SQL que use a sintaxe a seguir na cláusula `FROM`.

```
MyCloudwatchCatalog.database_name.table_name
```

Exemplos

O exemplo a seguir usa o conector do CloudWatch no Athena para se conectar à visualização `all_log_streams` no grupo de logs `/var/ecommerce-engine/order-processor` do CloudWatch Logs. A visualização `all_log_streams` é uma visualização de todos os fluxos de log no grupo de logs. A consulta de exemplo limita o número de linhas retornadas a 100.

```
SELECT *
FROM "MyCloudwatchCatalog"."/var/ecommerce-engine/order-processor".all_log_streams
LIMIT 100;
```

O exemplo a seguir analisa informações da mesma visualização que o exemplo anterior. O exemplo extrai o ID da ordem e o nível de log e filtra qualquer mensagem que tenha o nível `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'
```

Consultar várias origens de dados

Usando um exemplo mais complexo, imagine uma empresa de comércio eletrônico que usa as seguintes fontes de dados para armazenar dados relacionados às compras dos clientes:

- Amazon RDS para MySQL para armazenar dados do catálogo de produtos
- Amazon DocumentDB para armazenar dados de conta de clientes, como endereços de e-mail e endereços de envio

- Amazon DynamoDB para armazenar dados de envio e rastreamento de pedidos

Imagine que um analista de dados dessa aplicação de comércio eletrônico saiba que o tempo de entrega em algumas regiões foi afetado pelas condições climáticas locais. O analista quer saber quantos pedidos estão atrasados, onde os clientes afetados estão localizados e quais foram os produtos mais afetados. Em vez de investigar as fontes de informação separadamente, o analista usa o Athena para unir dados em uma única consulta federada.

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
```

Consultar visualizações federadas

Ao consultar fontes federadas, é possível usar visualizações para ofuscar as fontes de dados subjacentes ou ocultar uniões complexas de outros analistas que consultam os dados.

Considerações e limitações

- As visualizações federadas necessitam do mecanismo Athena versão 3.
- As visualizações federadas são armazenadas no AWS Glue, não na fonte de dados subjacente.
- As visualizações criadas com catálogos federados devem usar uma sintaxe de nome totalmente qualificada, como no seguinte exemplo:

  ```
  "ddbcatalog"."default"."customers"
  ```

- Os usuários que executam consultas em fontes federadas devem ter permissão para consultar fontes federadas.

- A permissão `athena:GetDataCatalog` é necessária para visualizações federadas. Para ter mais informações, consulte Exemplo de políticas de permissões do IAM para permitir consulta federada do Athena.

Exemplos

O exemplo a seguir cria uma visualização denominada `customers` nos dados armazenados em uma fonte de dados federada.

Example

```
CREATE VIEW customers AS
SELECT *
FROM my_federated_source.default.table
```

O exemplo de consulta a seguir mostra uma consulta que faz referência à visualização `customers` em vez da fonte de dados federada subjacente.

Example

```
SELECT id, SUM(order_amount)
FROM customers
GROUP by 1
ORDER by 2 DESC
LIMIT 50
```

O exemplo a seguir cria uma visualização denominada `order_summary` que combina dados de uma fonte de dados federada e de uma fonte de dados do Amazon S3. Da fonte federada, que já foi criada no Athena, a visualização usa as tabelas `person` e `profile`. No Amazon S3, a visualização usa as tabelas `purchase` e `payment`. Para se referir ao Amazon S3, a instrução usa a palavra-chave `awsdatacatalog`. A fonte de dados federada usa a sintaxe de nome totalmente qualificada *`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
```

Recursos adicionais do

- Para ver um exemplo de uma exibição federada que está desacoplada de sua fonte original e está disponível para análise sob demanda em um modelo multiusuário, consulte Estenda seu data mesh com o Amazon Athena e visualizações federadas no AWSBlog Big Data.
- Para obter mais informações sobre como trabalhar com visualizações no Athena, consulte Trabalhar com visualizações.

Como realizar consultas de passagem federadas

No Athena, você pode realizar consultas em fontes de dados federadas usando a linguagem de consulta da própria fonte de dados e enviar a consulta completa para execução na fonte de dados. Essas consultas são chamadas de consultas de passagem. Para executar consultas de passagem, você usa uma função de tabela na sua consulta do Athena. Você inclui em um dos argumentos da função de tabela a consulta de passagem que será realizada na fonte de dados. As consultas de passagem retornam uma tabela que você pode analisar usando o Athena SQL.

Conectores compatíveis

Os seguintes conectores de fonte de dados do Athena são compatíveis com consultas de passagem.

- Azure Data Lake Storage
- Azure Synapse
- Cloudera Hive
- Cloudera Impala
- CloudWatch
- Db2
- Db2 iSeries
- DocumentDB
- DynamoDB
- HBase
- Google BigQuery
- Hortonworks
- MySQL

- Neptune
- OpenSearch
- Oracle
- PostgreSQL
- Redshift
- SAP HANA
- Snowflake
- SQL Server
- Teradata
- Timestream
- Vertica

Considerações e limitações

Ao usar consultas de passagem no Athena, considere os seguintes pontos:

- A passagem de consultas só é compatível com instruções `SELECT` ou operações de leitura do Athena.
- As consultas de passagem devem ser realizadas dentro do contexto do catálogo da consulta externa (ou seja, a consulta que chama a função de tabela).
- O desempenho da consulta poderá variar de acordo com a configuração da fonte de dados.
- Não há compatibilidade com consultas de passagem para exibições.

Sintaxe

A sintaxe geral de passagem de consultas do Athena é a seguinte.

```
SELECT * FROM TABLE(system.function_name(arg1 => 'arg1Value'[, arg2 =>
 'arg2Value', ...]))
```

Para a maioria das fontes de dados, o primeiro e único argumento é `query`, seguido pelo operador de seta => e pela string de consulta.

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

Para simplificar, você pode omitir o argumento nomeado opcional `query` e o operador de seta `=>`.

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

Se a fonte de dados exigir mais do que a string de consulta, use argumentos nomeados na ordem esperada pela fonte de dados. Por exemplo, a expressão *`arg1`* `=>` `'`*`arg1Value`*`'` contém o primeiro argumento e seu valor. O nome *`arg1`* é específico da fonte de dados e pode diferir entre os conectores.

```
SELECT * FROM TABLE(
        system.query(
            arg1 => 'arg1Value',
            arg2 => 'arg2Value',
            arg3 => 'arg3Value'
        ));
```

Para obter informações sobre a sintaxe exata a ser usada com um conector específico, consulte a página do conector desejado.

Uso de aspas

Os valores dos argumentos, incluindo a string de consulta que você transmite, devem estar entre aspas simples, como no exemplo a seguir.

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

Quando a string de consulta estiver entre aspas duplas, a consulta vai falhar. A consulta a seguir falha com a mensagem de erro COLUMN_NOT_FOUND: line 1:43: Column 'select * from testdb.persons limit 10' cannot be resolved.

```
SELECT * FROM TABLE(system.query(query => "SELECT * FROM testdb.persons LIMIT 10"))
```

Para escapar uma aspas simples, adicione uma aspas simples ao original (p. ex., `terry's_group` para `terry''s_group`).

Exemplos

O exemplo de consulta a seguir envia uma consulta para uma fonte de dados. A consulta seleciona todas as colunas na tabela `customer`, limitando os resultados a 10.

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

A instrução a seguir executa a mesma consulta, mas elimina o argumento nomeado opcional `query` e o operador de seta =>.

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

Qualificadores de nomes do Athena e de tabelas federadas

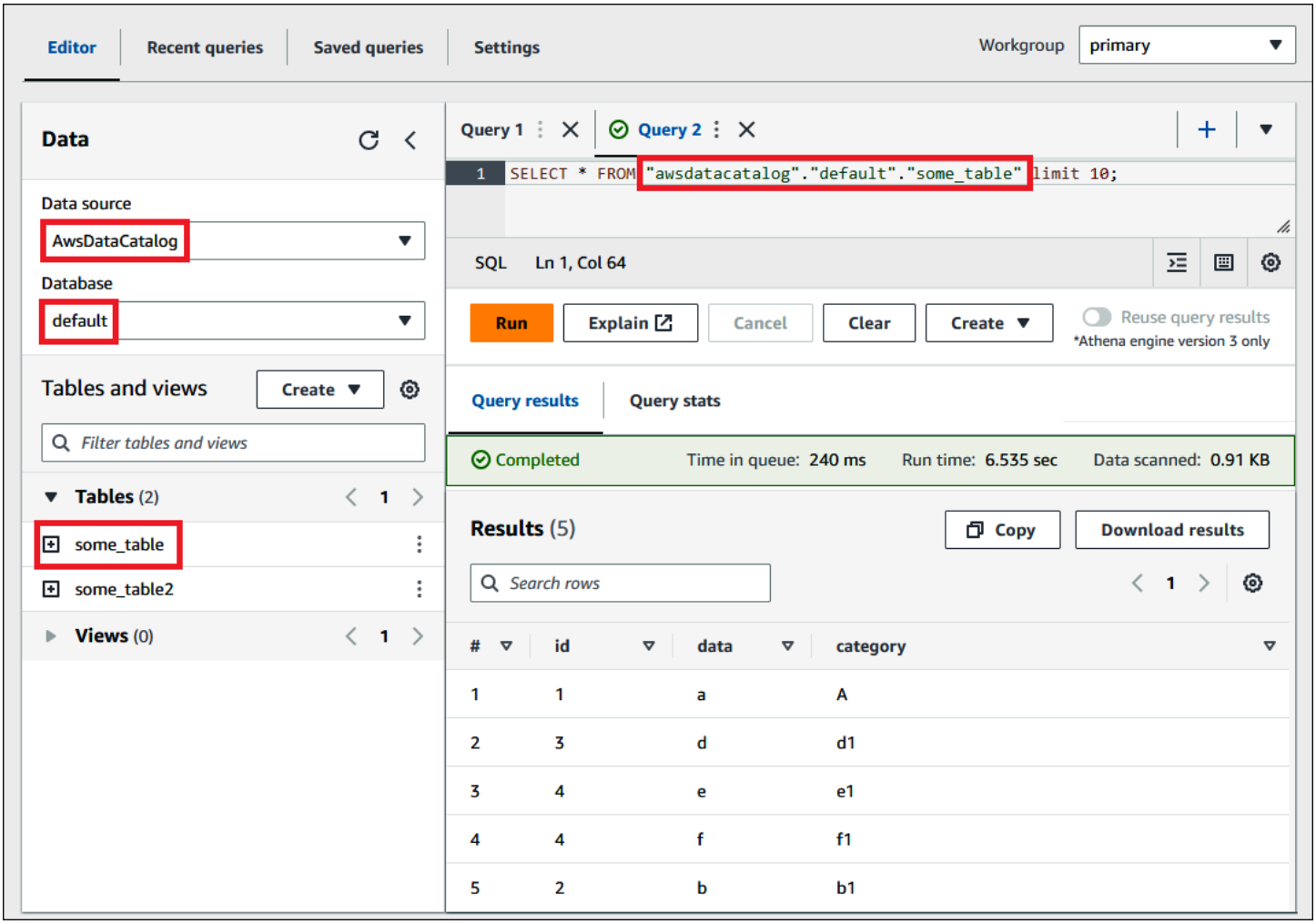
O Athena utiliza os seguintes termos para se referir às hierarquias de objetos de dados:

- Fonte de dados: um grupo de bancos de dados
- Banco de dados: um grupo de tabelas
- Tabela: dados organizados como um grupo de linhas ou colunas

Às vezes, esses objetos também são chamados por nomes alternativos, mas equivalentes, como:

- Às vezes uma fonte de dados é denominada catálogo.
- Às vezes um banco de dados é denominado esquema.

O exemplo de consulta no console do Athena a seguir usa a fonte de dados `awsdatacatalog`, o banco de dados `default` e a tabela `some_table`.



Termos de fontes de dados federadas

Ao consultar fontes de dados federadas, a fonte de dados subjacente pode não usar a mesma terminologia do Athena. Lembre-se dessa distinção ao gravar suas consultas federadas. As seções a seguir descrevem como os termos de objetos de dados do Athena correspondem aos das fontes de dados federadas.

Amazon Redshift

Um banco de dados do Amazon Redshift é um grupo de esquemas do Redshift que contém um grupo de tabelas Redshift.

Athena	Redshift
Fonte de dados do Redshift	Uma função do Lambda do conector Redshift configurada para apontar para um `database` do Redshift.
`data_source.database.table`	`database.schema.table`

Consulta de exemplo

```
SELECT * FROM
Athena_Redshift_connector_data_source.Redshift_schema_name.Redshift_table_name
```

Para obter mais informações sobre o conector, consulte Conector do Amazon Athena para o Redshift.

Cloudera Hive

Um servidor ou cluster do Cloudera Hive é um grupo de bancos de dados do Cloudera Hive que contém um grupo de tabelas Cloudera Hive.

Athena	Hive
Fonte de dados do Cloudera Hive	Função do Lambda do conector do Cloudera Hive configurada para apontar para um `server` Cloudera Hive.
`data_source.database.table`	`server.database.table`

Consulta de exemplo

```
SELECT * FROM
Athena_Cloudera_Hive_connector_data_source.Cloudera_Hive_database_name.Cloudera_Hive_table_name
```

Para obter mais informações sobre o conector, consulte Conector do Amazon Athena para o Cloudera Hive.

Cloudera Impala

Um servidor ou cluster do Impala é um grupo de bancos de dados do Impala que contém um grupo de tabelas Impala.

Athena	Impala
Fonte de dados do Impala	Função do Lambda do conector do Impala configurada para apontar para um `server` Impala.
`data_source.database.table`	`server.database.table`

Consulta de exemplo

```
SELECT * FROM
Athena_Impala_connector_data_source.Impala_database_name.Impala_table_name
```

Para obter mais informações sobre o conector, consulte Conector do Amazon Athena para o Cloudera Impala.

MySQL

Um servidor MySQL é um grupo de bancos de dados MySQL que contém um grupo de tabelas MySQL.

Athena	MySQL
Fonte de dados do MySQL	Função do Lambda do conector do MySQL configurada para apontar para um `server` MySQL.
`data_source.database.table`	`server.database.table`

Consulta de exemplo

```
SELECT * FROM
```

```
Athena_MySQL_connector_data source.MySQL_database_name.MySQL_table_name
```

Para obter mais informações sobre o conector, consulte Conector do Amazon Athena para o MySQL.

Oracle

Um servidor (ou banco de dados) Oracle é um grupo de esquemas do Oracle que contém um grupo de tabelas do Oracle.

Athena	Oracle
Fonte de dados do Oracle	Função do Lambda do conector do Oracle configurada para apontar para um `server` Oracle.
`data_source.database.table`	`server.schema.table`

Consulta de exemplo

```
SELECT * FROM
Athena_Oracle_connector_data_source.Oracle_schema_name.Oracle_table_name
```

Para obter mais informações sobre o conector, consulte Conector do Amazon Athena para Oracle.

Postgres

Um servidor (ou cluster) Postgres é um grupo de bancos de dados do Postgres. Um banco de dados do Postgres é um grupo de esquemas do Postgres que contém um grupo de tabelas Postgres.

Athena	Postgres
Fonte de dados do Postgres	Função do Lambda do conector do Postgres configurada para apontar para um `server` e `database` do Postgres.
`data_source.database.table`	`server.database.schema.table`

Consulta de exemplo

```
SELECT * FROM
Athena_Postgres_connector_data_source.Postgres_schema_name.Postgres_table_name
```

Para obter mais informações sobre o conector, consulte Conector do Amazon Athena para o PostgreSQL.

Desenvolver um conector de fonte de dados usando o SDK do Athena Query Federation

Para escrever os próprios conectores de origem dos dados, você pode usar o Athena Query Federation SDK. O Athena Query Federation SDK define um conjunto de interfaces e protocolos de conexão que você pode usar para permitir que o Athena delegue partes do plano de execução de consultas ao código que você escreve e implanta. O SDK inclui um conjunto de conectores e um conector de exemplo.

Você também pode personalizar os conectores predefinidos do Amazon Athena para uso próprio. Você pode modificar uma cópia do código-fonte do GitHub e usar a ferramenta de publicação do conector para criar seu próprio pacote do AWS Serverless Application Repository. Depois de implantar o conector dessa maneira, você poderá usá-lo em suas consultas do Athena.

Para saber sobre como baixar o SDK e obter instruções detalhadas para gravar seu próprio conector, consulte Example Athena connector (Exemplo de conector do Athena) no GitHub.

Conectores de fonte de dados do Athena para o Apache Spark

Alguns conectores de fonte de dados do Athena estão disponíveis como conectores DSV2 do Spark. Os nomes dos conectores DSV2 do Spark têm um sufixo `-dsv2` (por exemplo, `athena-dynamodb-dsv2`).

A seguir, os conectores DSV2 atualmente disponíveis, seu nome de classe do Spark `.format()` e os links para a documentação sobre consultas federadas do Amazon Athena correspondente:

Conector DSV2	Nome da classe do Spark .format()	Documentação
athena-cloudwatch-dsv2	`com.amazonaws.athena.connectors.dsv2.cloudwatch.CloudwatchTableProvider`	CloudWatch

Conector DSV2	Nome da classe do Spark .format()	Documentação
athena-cloudwatch-metrics-dsv2	`com.amazonaws.athena.connectors.dsv2.cloudwatch.metrics.CloudwatchMetricsTableProvider`	Métricas do CloudWatch
atena-aws-cmdb-dsv2	`com.amazonaws.athena.connectors.dsv2.aws.cmdb.AwsCmdbTableProvider`	CMDB
atena-dynamodb-dsv2	`com.amazonaws.athena.connectors.dsv2.dynamodb.DDBTableProvider`	DynamoDB

Para baixar arquivos `.jar` para os conectores DSV2, visite a página DSV2 da consulta federada do Amazon Athena no GitHub e veja a seção Lançamentos, Lançamento ***<versão>***, Ativos.

Especificar o jar para o Spark

Para usar os conectores DSV2 do Athena com o Spark, você envia o arquivo `.jar` do conector para o ambiente do Spark que está usando. As seções a seguir descrevem casos específicos.

Athena para Spark

Para obter informações sobre como adicionar arquivos `.jar` personalizados e configurações personalizadas ao Amazon Athena para Apache Spark, consulte Adicionar arquivos JAR e configuração personalizada do Spark.

Spark em geral

Para passar arquivo `.jar` do conector para o Spark, use o comando `spark-submit` e especifique o Aarquivo `.jar` na opção `--jars`, como no seguinte exemplo:

```
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
```

Spark no Amazon EMR

Para executar um comando `spark-submit` com o parâmetro `--jars` no Amazon EMR, você deve adicionar uma etapa ao cluster do Spark no Amazon. Para obter detalhes sobre como usar `spark-submit` no Amazon EMR, consulte Adicionar uma etapa do Spark no Guia de lançamento do Amazon EMR.

Spark para ETL do AWS Glue

Para ETL do AWS Glue, você pode passar a URL GitHub.com do arquivo `.jar` para o argumento `--extra-jars` do comando `aws glue start-job-run`. A documentação do AWS Glue descreve o parâmetro `--extra-jars` seguindo um caminho do Amazon S3, mas o parâmetro também pode usar uma URL HTTPS. Para obter mais informações, consulte Referência de parâmetros de trabalho no Guia do desenvolvedor do AWS Glue.

Consultar o conector no Spark

Para enviar o equivalente à sua consulta federada existente do Athena no Apache Spark, use a função `spark.sql()`. Por exemplo, suponhamos que você tenha a consulta do Athena a seguir e deseje usar no Apache Spark.

```
SELECT somecola, somecolb, somecolc
FROM ddb_datasource.some_schema_or_glue_database.some_ddb_or_glue_table
WHERE somecola > 1
```

Para realizar a mesma consulta no Spark usando o conector DSV2 do Amazon Athena para DynamoDB, use o seguinte código:

```
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
''')
```

Especificar parâmetros

As versões DSV2 dos conectores de fonte de dados Athena usam os mesmos parâmetros dos conectores de fonte de dados Athena correspondentes. Para obter informações sobre parâmetros, consulte a documentação do conector de fonte de dados do Athena correspondente.

No código do PySpark, use a sintaxe a seguir para configurar os parâmetros.

```
spark.read.option("athena.connectors.conf.parameter", "value")
```

Por exemplo, o código a seguir define o parâmetro `disable_projection_and_casing` do conector Amazon Athena para o DynamoDB como `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())
```

Políticas do IAM de acesso a catálogos de dados

Para controlar o acesso a catálogos de dados, use as permissões do IAM no nível do recurso ou as políticas do IAM baseadas em identidade.

O procedimento a seguir é específico ao Athena.

Para obter informações específicas do IAM, acesse os links listados no fim desta seção. Para obter informações sobre políticas de catálogo de dados JSON de exemplo, consulte Políticas de catálogo de dados de exemplo.

Para usar o editor visual no console do IAM para criar uma política de catálogo de dados

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Policies (Políticas) e Create policy (Criar política).

3. Na guia Editor visual, selecione Escolher um serviço. Em seguida, escolha o Athena para adicionar à política.
4. Escolha Select actions (Selecionar ações) e defina as ações para adicionar à política. O editor visual mostra as ações disponíveis no Athena. Para obter mais informações, consulte Ações, recursos e chaves de condição do Amazon Athena na Referência de autorização do serviço.
5. Escolha Add actions (Adicionar ações) para digitar uma ação específica ou use caracteres curinga (*) para especificar várias ações.

   Por padrão, a política que você está criando permite as ações que você escolhe. Se você escolher uma ou mais ações compatíveis com as permissões no nível do recurso `datacatalog` no Athena, o editor listará o recurso `datacatalog`.
6. Escolha Recursos para especificar os catálogos de dados específicos para sua política. Para obter políticas de catálogo de dados JSON de exemplo, consulte Políticas de catálogo de dados de exemplo.
7. Especifique o recurso `datacatalog` da seguinte forma:

   ```
   arn:aws:athena:<region>:<user-account>:datacatalog/<datacatalog-name>
   ```
8. Selecione Review Policy (Revisar política), digite um Name (Nome) e uma Description (Descrição) (opcional) para a política que você está criando. Revise o resumo da política para ter certeza de que você concedeu as permissões que pretendia.
9. Escolha Criar política para salvar sua nova política.
10. Anexe essa política baseada em identidade a um usuário, um grupo ou uma função e especifique os recursos do `datacatalog` que eles podem acessar.

Para obter mais informações, consulte os seguintes tópicos na Referência de autorização do serviço e no Manual do usuário do IAM:

- Ações, recursos e chaves de condição do Amazon Athena
- Criar políticas com o editor visual
- Adicionar e remover políticas do IAM
- Controlar o acesso aos recursos

Para obter políticas de catálogo de dados JSON de exemplo, consulte Políticas de catálogo de dados de exemplo.

Para obter informações sobre permissões do AWS Glue e permissões de crawler do AWS Glue, consulte Configurar permissões do IAM para o AWS Glue e Pré-requisitos do crawler no Guia do desenvolvedor do AWS Glue.

Para obter uma lista completa de ações do Amazon Athena, consulte os nomes das ações de API na Referência de API do Amazon Athena.

Políticas de catálogo de dados de exemplo

Esta seção inclui exemplos de políticas que você pode usar para habilitar várias ações nos catálogos de dados.

Um catálogo de dados é um recurso do IAM gerenciado pelo Athena. Portanto, se sua política de catálogo de dados usa ações que usam o `datacatalog` como uma entrada, especifique o ARN do catálogo de dados da seguinte maneira:

```
"Resource": [arn:aws:athena:<region>:<user-account>:datacatalog/<datacatalog-name>]
```

O `<datacatalog-name>` é o nome do seu catálogo de dados. Por exemplo, para um catálogo de dados chamado `test_datacatalog`, especifique-o como um recurso da seguinte forma:

```
"Resource": ["arn:aws:athena:us-east-1:123456789012:datacatalog/test_datacatalog"]
```

Para obter uma lista completa de ações do Amazon Athena, consulte os nomes das ações de API na Referência de API do Amazon Athena. Para obter mais informações sobre as políticas do IAM, consulte Criar políticas com o editor visual no Guia do usuário do IAM. Para obter mais informações sobre como criar políticas do IAM para grupos de trabalho, consulte Políticas do IAM de acesso a catálogos de dados.

- 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 Exemplo de política para acesso total a todos os catálogos de dados

A seguinte política permite acesso total a todos os recursos do catálogo de dados que podem existir na conta. Recomendamos que você use essa política para os usuários da sua conta que devem administrar e gerenciar catálogos de dados para todos os outros usuários.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "athena:*"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

Example Exemplo de política para acesso total a um catálogo de dados especificado

A seguinte política permite acesso total ao único recurso específico do catálogo de dados, denominado `datacatalogA`. Você pode usar essa política para os usuários com controle total sobre um determinado catálogo de dados.

```
{
   "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 Exemplo de política para consulta em um catálogo de dados especificado

Na política a seguir, um usuário tem permissão para executar consultas no `datacatalogA` especificado. O usuário não tem permissão para executar tarefas de gerenciamento para o próprio catálogo de dados, como atualizá-lo ou excluí-lo.

```
{
   "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 Exemplo de política para operações de gerenciamento em um catálogo de dados especificado

Na política a seguir, o usuário tem permissão para criar, excluir, obter detalhes e atualizar um catálogo de dados `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 Exemplo de política para listagem de catálogos de dados

A política a seguir permite que todos os usuários listem todos os catálogos de dados:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListDataCatalogs"
            ],
            "Resource": "*"
        }
    ]
}
```

Example Exemplo de política para operações de metadados em catálogos de dados

A política a seguir permite operações de metadados em catálogos de dados:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:GetDatabase",
                "athena:GetTableMetadata",
                "athena:ListDatabases",
```

```
                "athena:ListTableMetadata"
            ],
            "Resource": "*"
        }
    ]
}
```

Gerenciar origens de dados

Você pode usar a página Data sources (Origens dos dados) do console do Athena para gerenciar as origens dos dados que você cria.

Como exibir uma fonte de dados

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.

3. No painel de navegação, escolha Origens dos dados.
4. Na lista de origens dos dados, escolha o nome da origens dos dados que deseja exibir.

> **Note**
>
> Os itens na coluna Data source name (Nome da fonte de dados) correspondem à saída da ação da API ListDataCatalogs e do comando da CLI list-data-catalogs.

Como editar uma fonte de dados

1. Na página Data sources (Origens dos dados), execute um dos seguintes procedimentos:
   - Selecione o botão ao lado do nome do catálogo e escolha Actions (Ações), Edit (Editar).
   - Escolha o nome da origem dos dados. Em seguida, na página de detalhes, escolha Actions (Ações), Edit (Editar).
2. Na página Edit (Editar), é possível escolher uma função do Lambda diferente para a origem dos dados, alterar a descrição ou adicionar etiquetas personalizadas. Para obter mais informações sobre tags, consulte Etiquetar recursos do Athena.
3. Escolha Salvar.
4. Para editar seu AwsDataCatalog na origem dos dados, escolha o link AwsDataCatalog para abrir a página de detalhes. Em seguida, na página de detalhes, escolha o link para o console do AWS Glue, onde você pode editar seu catálogo.

Para compartilhar uma origem dos dados

Para obter informações sobre como compartilhar origens dos dados, visite os links a seguir.

- Para origens dos dados não baseadas Lambda não Hive, consulte Habilitar consultas federadas entre contas.
- Para AWS Glue Data Catalogs, veja Acesso aos catálogos de dados do AWS Glue entre contas.

Como excluir uma fonte de dados

1. Na página Data sources (Origens dos dados), execute um dos seguintes procedimentos:
   - Selecione o botão ao lado do nome do catálogo e escolha Actions (Ações), Delete (Excluir).
   - Escolha o nome da origem dos dados e, na página de detalhes, escolha Actions (Ações), Delete (Excluir).

> **Note**
>
> O AwsDataCatalog é a origem dos dados padrão da sua conta e não pode ser excluído.

Você é avisado que, ao excluir uma origem dos dados, seu catálogo de dados, tabelas e exibições correspondentes são removidos do editor de consultas. As consultas salvas que usavam a origem dos dados não serão mais executadas no Athena.

2. Para confirmar a exclusão, digite o nome da origem dos dados e escolha Delete (Excluir).

Usar o Amazon DataZone no Athena

Você pode usar o Amazon DataZone para compartilhar, pesquisar e descobrir dados em grande escala além dos limites organizacionais. O DataZone simplifica sua experiência em todos os serviços de análise da AWS, como o Athena, o AWS Glue e o AWS Lake Formation. Por exemplo, se você tiver petabytes de dados em diferentes fontes de dados, poderá usar o Amazon DataZone para criar agrupamentos de pessoas, dados e ferramentas com base nos casos de uso comercial. Para obter mais informações, consulte O que é o Amazon DataZone?.

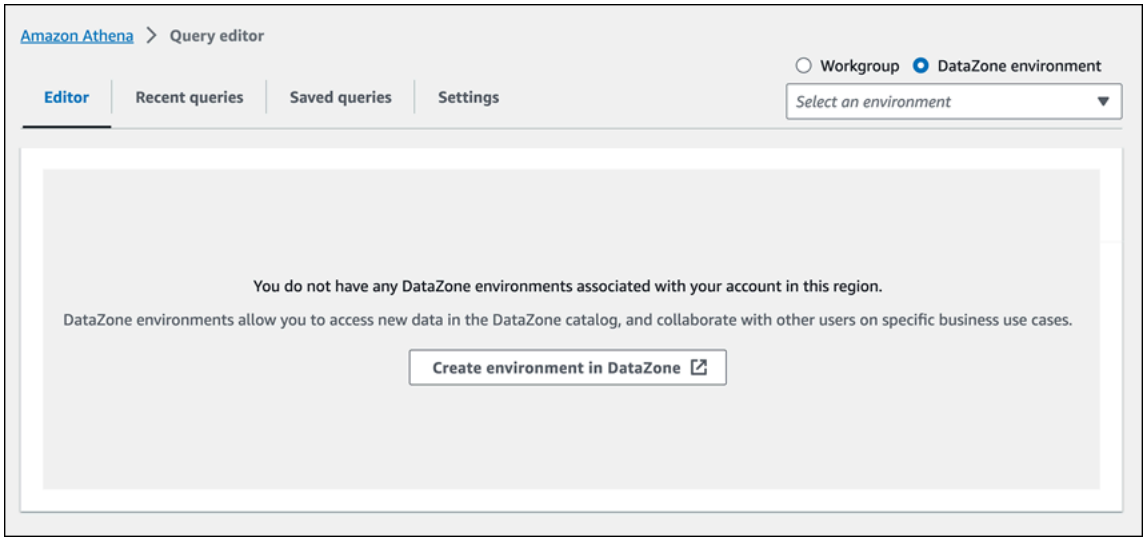
No Athena, você pode usar o editor de consultas para acessar e consultar os ambientes do DataZone. Um ambiente do DataZone especifica uma combinação de projeto e domínio do DataZone. Ao usar um ambiente do DataZone no console do Athena, você assume o perfil do IAM do ambiente do DataZone e só vê os bancos de dados e as tabelas que pertencem a esse ambiente. As permissões são determinadas pelos perfis que você especifica no DataZone.

No Athena, você pode usar o seletor Ambiente do DataZone na página do editor de consultas para escolher um ambiente do DataZone.

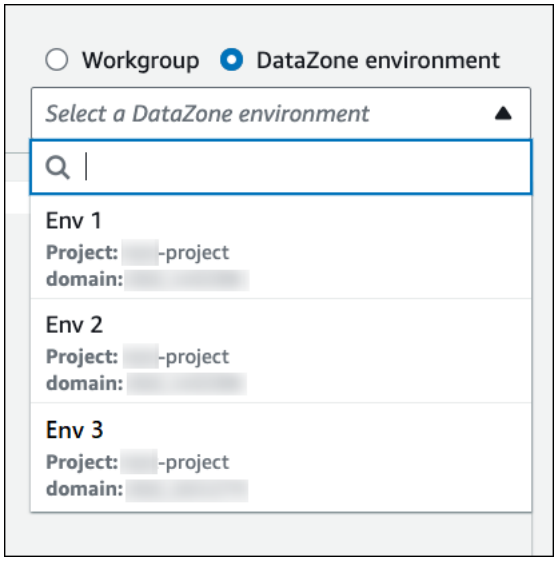
Para abrir um ambiente do DataZone no Athena

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. No canto superior direito do console do Athena, ao lado de Grupo de trabalho, escolha Ambiente do DataZone.

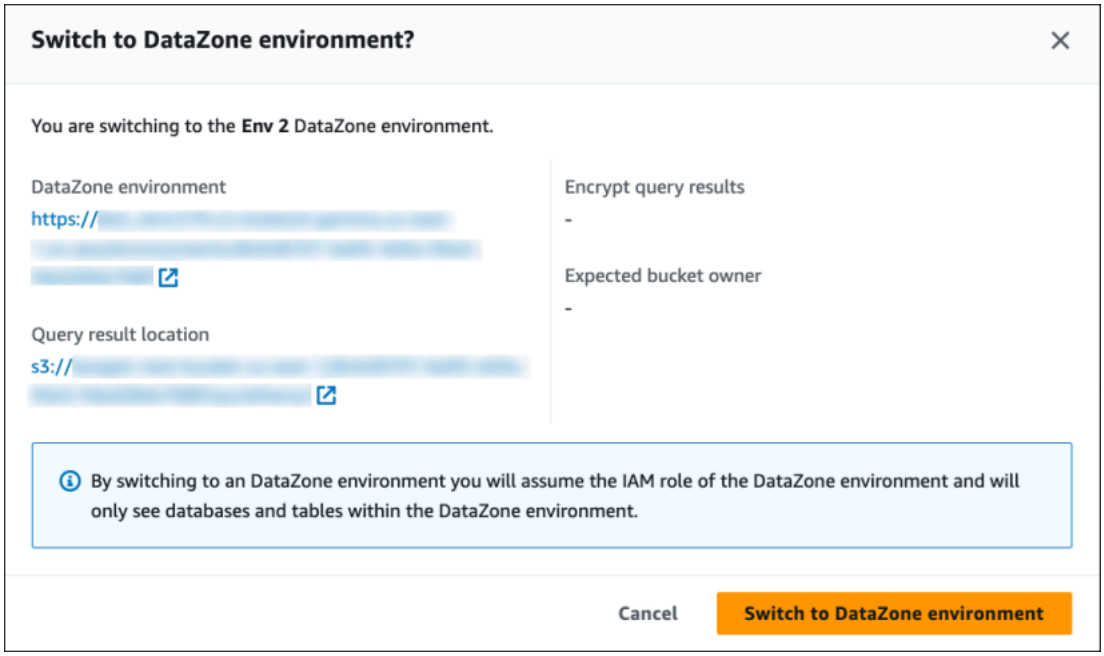
> **Note**
>
> A opção Ambiente do DataZone só está presente quando você tem um ou mais domínios disponíveis no DataZone.



3. Use o seletor Ambiente do DataZone para escolher um ambiente do DataZone.



4. Na caixa de diálogo Alternar para o ambiente do DataZone, verifique se o ambiente é o que você deseja e escolha Alternar para o ambiente do DataZone.



Para obter mais informações sobre como começar a usar o DataZone e o Athena, consulte o tutorial Getting started no Amazon DataZone User Guide.

Conectar-se ao Amazon Athena com drivers ODBC e JDBC

Para explorar e visualizar os dados com as ferramentas de business intelligence, fazer download, instalar e configurar um driver Open Database Connectivity (ODBC) ou Java Database Connectivity (JDBC).

Tópicos

- Conectar-se ao Amazon Athena com JDBC
- Conectar-se ao Amazon Athena com ODBC

Consulte também os seguintes tópicos da Central de Conhecimento da AWS e do blog de big data da AWS:

- How can I use my IAM role credentials or switch to another IAM role when connecting to Athena using the JDBC driver? (Como posso usar minhas credenciais de função do IAM ou alternar para outra função do IAM ao me conectar ao Athena usando o driver JDBC?)
- Como estabelecer a confiança entre o ADFS e a AWS e usar as credenciais do Active Directory para se conectar ao Amazon Athena com o driver ODBC

Conectar-se ao Amazon Athena com JDBC

O Amazon Athena oferece dois drivers JDBC, as versões 2.x e 3.x. O driver Athena JDBC 3.x é o driver de nova geração que oferece melhor desempenho e compatibilidade. O driver JDBC 3.x é compatível com a leitura de resultados de consultas diretamente no Amazon S3, o que melhora o desempenho de aplicações que consomem resultados de consultas de grande porte. O novo driver também tem menos dependências de terceiros, o que facilita a integração com ferramentas de BI e aplicações personalizadas. Na maioria dos casos, você pode usar o novo driver sem alterações ou com o mínimo de alterações na configuração existente.

- Para baixar o driver JDBC 3.x, consulte Driver JDBC 3.x do Athena.
- Para baixar o driver JDBC 2.x, consulte Driver JDBC 2.x do Athena.

Tópicos

- Driver JDBC 3.x do Athena
- Driver JDBC 2.x do Athena

Driver JDBC 3.x do Athena

Você pode usar uma conexão JDBC para se conectar ao Amazon Athena em muitas ferramentas e aplicações de clientes SQL de terceiros.

Requisitos do sistema

- Ambiente de runtime do Java 8 (ou superior)
- Pelo menos 20 MB de espaço em disco disponível

Considerações e limitações

A seguir estão algumas considerações e limitações do driver JDBC 3.x. do Athena.

- Registro em log: o driver 3.x usa o SLF4J, que é uma camada de abstração que permite o uso de qualquer um dos diversos sistemas de registro em log em runtime.
- Criptografia: quando usa o captador do Amazon S3 com a opção de criptografia do `CSE_KMS`, o cliente do Amazon S3 não pode descriptografar os resultados armazenados em um bucket do Amazon S3. Se você precisar de criptografia `CSE_KMS`, poderá continuar a usar o buscador

de streaming. Está nos planos a compatibilidade com a criptografia `CSE_KMS` com o buscador Amazon S3.

Download do driver JDBC 3.x

Esta seção contém informações de download e licença para o driver JDBC 3.x.

> **Important**
>
> Quando você usar o driver JDBC 3.x, preste atenção aos seguintes requisitos:
>
> - Porta 444 aberta: mantenha a porta 444, que o Athena usa para fazer uma transmissão dos resultados das consultas, aberta para o tráfego de saída. Ao usar um endpoint do PrivateLink para se conectar ao Athena, verifique se o grupo de segurança anexado ao endpoint do PrivateLink está aberto para o tráfego de entrada na porta 444.
> - Política athena:GetQueryResultsStream: adicione a ação de política `athena:GetQueryResultsStream` para as entidades principais do IAM que usam o driver JDBC. Essa ação de política não é exposta diretamente com a API. Ela apenas é usada com drivers ODBC e JDBC como parte do suporte a resultados de transmissão. Para visualizar um exemplo de política, consulte Política gerenciada AWS: AWSQuicksightAthenaAccess.

Para baixar o driver JDBC versão 3.x do Amazon Athena, acesse os links a seguir.

Jar com dependências do driver JDBC

O download a seguir empacota o driver e todas as suas dependências no mesmo arquivo `.jar`. Esse download é comumente usado para clientes SQL de terceiros.

3.2.0 uber jar

Jar sem dependências do driver JDBC

O download a seguir é um arquivo `.zip` que contém o `.jar` sem dependências para o driver e arquivos `.jar` separados para as dependências do driver. Esse download é comumente usado para aplicações personalizadas que podem ter dependências conflitantes com as dependências que o driver usa. Esse download é útil se você quiser escolher quais dependências do driver incluir no jar sem dependências e quais excluir se a aplicação personalizada já contiver uma ou mais dependências.

3.2.0 lean jar

Licença

O link a seguir contém o contrato de licença do driver JDBC 3.x.

Licença

Tópicos

- Introdução ao driver JDBC 3.x
- Parâmetros de conexão do JDBC 3.x do Amazon Athena
- Outra configuração do JDBC 3.x
- Notas de versão do JDBC 3.x do Amazon Athena
- Versões anteriores do driver JDBC 3.x driver do Athena

Introdução ao driver JDBC 3.x

Use as informações desta seção para começar a usar o driver JDBC 3.x do Amazon Athena.

Tópicos

- Instruções de instalação
- Executar o driver
- Configurar o driver
- Atualizar o driver Athena JDBC v2

Instruções de instalação

Você pode usar o driver JDBC 3.x em uma aplicação personalizada ou em um cliente SQL de terceiros.

Em uma aplicação personalizada

Baixe o arquivo `.zip` que contém o jar do driver e suas dependências. Cada dependência tem seu próprio arquivo `.jar`. Adicione o jar do driver como uma dependência na sua aplicação personalizada. Adicione seletivamente as dependências do jar do driver, com base em você já ter ou não adicionado essas dependências à aplicação de outra fonte.

Em um cliente SQL de terceiros

Baixe o arquivo jar com dependências do driver e adicione-o ao cliente SQL de terceiros seguindo as instruções para esse cliente.

Executar o driver

Para executar o driver, você pode usar uma aplicação personalizada ou um cliente SQL de terceiros.

Em uma aplicação personalizada

Use a interface JDBC para interagir com o driver JDBC em um programa. O código a seguir mostra um exemplo de aplicação Java personalizada.

```
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
}
```

Em um cliente SQL de terceiros

Siga a documentação do cliente SQL que você está usando. Normalmente, você usa a interface gráfica do usuário do cliente SQL para inserir e enviar a consulta, e os resultados da consulta são exibidos na mesma interface.

Configurar o driver

Você pode usar os parâmetros de conexão para configurar o driver JDBC do Amazon Athena. Para obter os parâmetros de conexão compatíveis, consulte Parâmetros de conexão do JDBC 3.x do Amazon Athena.

Em uma aplicação personalizada

Para definir os parâmetros de conexão do driver JDBC em uma aplicação personalizada, faça um dos seguintes procedimentos:

- Adicione os nomes dos parâmetros e seus valores a um objeto `Properties`. Quando você chamar `Connection#connect`, passe esse objeto junto com a URL. Para obter um exemplo, consulte o exemplo de aplicação Java em Executar o driver.
- Na string de conexão (a URL), use o formato a seguir para adicionar os nomes dos parâmetros e seus valores diretamente após o prefixo do protocolo.

```
<parameterName>=<parameterValue>;
```

  Use um ponto e vírgula no final de cada par nome/valor do parâmetro e não deixe espaço em branco após o ponto e vírgula, como no exemplo a seguir.

```
String url = "jdbc:athena://WorkGroup=primary;Region=us-east-1;...;";AthenaDriver
 driver = new AthenaDriver();Connection connection = driver.connect(url, null);
```

> **Note**
>
> Se um parâmetro for especificado tanto na string de conexão quanto no objeto `Properties`, o valor na string de conexão terá precedência. Não é recomendável especificar o mesmo parâmetro em ambos os lugares.

- Adicione os valores dos parâmetros como argumentos aos métodos de `AthenaDataSource`, como no exemplo a seguir.

```
AthenaDataSource dataSource = new AthenaDataSource();
    dataSource.setWorkGroup("primary");
    dataSource.setRegion("us-east-2");
    ...
    Connection connection = dataSource.getConnection();
    ...
```

Em um cliente SQL de terceiros

Siga as instruções do cliente SQL que você estiver usando. Normalmente, o cliente fornece uma interface gráfica de usuário para inserir os nomes dos parâmetros e seus valores.

Atualizar o driver Athena JDBC v2

A maioria dos parâmetros de conexão do JDBC versão 3 é compatível retroativamente com o driver JDBC versão 2 (Simba). Isso significa que uma string de conexão da versão 2 pode ser reutilizada com a versão 3 do driver. Porém, alguns parâmetros de conexão foram alterados. Essas mudanças estão descritas aqui. Ao atualizar para o driver JDBC versão 3, atualize a configuração existente, se for o caso.

Classe do driver

Algumas ferramentas de BI solicitam que você forneça a classe do driver, encontrada no arquivo `.jar` do driver JDBC. A maioria das ferramentas encontra essa classe automaticamente. O nome totalmente qualificado da classe no driver da versão 3 é `com.amazon.athena.jdbc.AthenaDriver`. No driver da versão 2, a classe era `com.simba.athena.jdbc.Driver`.

String de conexão

O driver da versão 3 usa `jdbc:athena://` como protocolo no início do URL da string de conexão do JDBC. O driver da versão 3 também é compatível com o protocolo `jdbc:awsathena://` da versão 2, mas o uso do protocolo da versão 2 foi descontinuado. Para evitar comportamentos indefinidos, a versão 3 não aceita strings de conexão que comecem com `jdbc:awsathena://` se a versão 2 (ou qualquer outro driver que aceite strings de conexão que comecem com `jdbc:awsathena://`) tiver sido registrada com a classe DriverManager.

Provedores de credenciais

O driver da versão 2 usa nomes totalmente qualificados para identificar os diferentes provedores de credenciais (por exemplo, `com.simba.athena.amazonaws.auth.DefaultAWSCredentialsProviderChain`). O driver da versão 3 usa nomes mais curtos (por exemplo, `DefaultChain`). Os novos nomes são descritos nas seções correspondentes para cada provedor de credenciais.

Os provedores de credenciais personalizadas escritos para o driver da versão 2 precisam ser modificados para que o driver da versão 3 implemente a interface AWSCredentialsProvider a partir

do novo AWS SDK for Java em vez da interface AWSCredentialsProvider a partir do AWS SDK for Java anterior.

`PropertiesFileCredentialsProvider` não é suportado no driver JDBC 3.x. O provedor foi usado no driver JDBC 2.x, mas pertence à versão anterior do AWS SDK para Java, que está chegando ao fim do suporte. Para obter a mesma funcionalidade no driver JDBC 3.x, use o provedor Credenciais do perfil de configuração da AWS.

Nível de log

A tabela a seguir mostra as diferenças nos parâmetros de `LogLevel` dos drivers JDBC versão 2 e versão 3.

Versão do driver JDBC	Nome do parâmetro	Tipo de parâmetro	Valor padrão	Possíveis valores	Exemplo de string de conexão
v2	`LogLevel`	Opcional	0	0-6	`LogLevel=6;`
v3	`LogLevel`	Opcional	TRACE	OFF, ERROR, WARN, INFO, DEBUG, TRACE	`LogLevel=INFO;`

Recuperação da ID da consulta

No driver da versão 2, você desempacota uma instância de `Statement` em `com.interfaces.core.IStatementQueryInfoProvider`, uma interface que tem dois métodos: `#getPReparedQueryId` e `#getQueryId`. Você pode usar esses métodos para obter o ID de execução da consulta que foi executada.

No driver da versão 3, você desempacota as instâncias `Statement`, `PreparedStatement` e `ResultSet` na interface `com.amazon.athena.jdbc.AthenaResultSet`. A interface tem um método: `#getQueryExecutionId`.

Parâmetros de conexão do JDBC 3.x do Amazon Athena

Os parâmetros de conexão compatíveis são divididos aqui em três seções: Parâmetros básicos de conexão, Parâmetros avançados de conexão e Parâmetros de conexão de autenticação. As seções Parâmetros de conexão avançados e Parâmetros de conexão de autenticação têm subseções que agrupam os parâmetros relacionados.

Tópicos

- Parâmetros básicos de conexão
- Parâmetros avançados de conexão
- Parâmetros de conexão de autenticação

Parâmetros básicos de conexão

As seções a seguir descrevem os parâmetros básicos de conexão do driver JDBC 3.x.

Região

A Região da AWS onde as consultas serão executadas. Para conferir a lista de regiões, consulte Amazon Athena endpoints and quotas.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
Região	AwsRegion (obsoleto)	Obrigatório (mas se não for fornecido, será pesquisado usando DefaultAwsRegionProviderChain)	nenhuma

Catálogo

O catálogo que contém os bancos de dados e as tabelas que serão acessados com o driver. Para obter mais informações sobre o catálogo de dados, consulte DataCatalog.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
Catálogo	nenhuma	Opcional	AwsDataCatalog

Banco de dados

A banco de dados em que as consultas serão executadas. As tabelas que não são explicitamente qualificadas com um nome de banco de dados são resolvidas para esse banco de dados. Para obter informações sobre bancos de dados, consulte Banco de dados.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
Banco de dados	Schema	Opcional	padrão

WorkGroup

O grupo de trabalho no qual as consultas serão executadas. Para obter mais informações sobre grupos de trabalho, consulte Grupo de trabalho.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
WorkGroup	nenhuma	Opcional	principal

Local de saída

O local do Amazon S3 em que os resultados de consultas serão armazenados. Para obter informações sobre o local de saída, consulte ResultConfiguration.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
OutputLocation	S3OutputLocation (obsoleto)	Obrigatório (a menos que o grupo de trabalho especifique um local de saída)	nenhuma

Parâmetros avançados de conexão

As seções a seguir descrevem os parâmetros avançados de conexão para o driver JDBC 3.x.

Tópicos

- Parâmetros da criptografia de resultados
- Parâmetros de busca de resultados
- Parâmetros de reutilização dos resultados da consulta
- Parâmetros de sondagem da execução de uma consulta
- Parâmetros de substituição de endpoint
- Parâmetros de configuração de proxy
- Parâmetros de registro em log
- Nome da aplicação
- Teste de conexão
- Número de novas tentativas

Parâmetros da criptografia de resultados

Observe os seguintes pontos:

- A chave do AWS KMS deve ser especificada quando `EncryptionOption` é `SSE_KMS` ou `CSE_KMS`.
- A chave do AWS KMS não pode ser especificada quando a `EncryptionOption` não está especificada ou quando a `EncryptionOption` é `SSE_S3`.

Opção de criptografia

O tipo de criptografia a ser usado nos resultados de consultas quando eles são armazenados no Amazon S3. Para obter mais informações sobre as opções de criptografia, consulte EncryptionConfiguration na Amazon Athena API Reference.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão	Possíveis valores
EncryptionOption	S3OutputEncOption (obsoleto)	Opcional	nenhuma	SSE_S3, SSE_KMS, CSE_KMS

Chave do KMS

O ARN ou ID da chave do KMS, se `SSE_KMS` ou `CSE_KMS` for escolhido como a opção de criptografia. Para obter mais informações, consulte EncryptionConfiguration na Amazon Athena API Reference.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
KmsKey	S3OutputEncKMSKey (obsoleto)	Opcional	nenhuma

Parâmetros de busca de resultados

Buscador de resultados

O buscador que será usado para baixar resultados das consultas.

O buscador de resultados padrão, `S3`, baixa os resultados das consultas diretamente do Amazon S3 sem usar as APIs do Athena. Essa é a opção mais rápida na maioria dos casos. Essa opção não estará disponível se os resultados da consulta forem criptografados com o `CSE_KMS` ou se a política que permitir ao usuário acessar os resultados da consulta, só permitir chamadas do Athena usando o `s3:CalledVia`.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão	Possíveis valores
ResultFetcher	nenhuma	Opcional	S3	S3, GetQueryResults, GetQueryResultsStream

> **Note**
>
> No driver JDBC 2.x, a configuração `UseResultsetStreaming = 1` define o driver para usar a API de streaming do conjunto de resultados. No driver JDBC 3.x, a configuração equivalente é `ResultFetcher=GetQueryResultsStream`.

Tamanho da busca

O valor desse parâmetro é usado como o mínimo para buffers internos e como tamanho das páginas de destino ao buscar resultados. O valor 0 (zero) significa que o driver deve usar seus próprios padrões conforme descrito abaixo. O valor máximo é 1.000.000.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
FetchSize	RowsToFetchPerBlock (obsoleto)	Opcional	0

- O buscador `GetQueryResults` sempre usará um tamanho de página de 1.000, que é o valor máximo permitido para a chamada de API. Quando o tamanho da busca é maior do de 1.000, várias chamadas sucessivas de API são feitas para preencher o buffer acima do mínimo.
- O buscador `GetQueryResultsStream` usa o tamanho de busca configurado como o tamanho de página, ou 10.000 por padrão.
- O buscador `S3` usa o tamanho de busca configurado como o tamanho de página, ou 10.000 por padrão.

Parâmetros de reutilização dos resultados da consulta

Habilitar a reutilização de resultados

Especifica se os resultados anteriores para a mesma consulta poderão reutilizados quando uma consulta for executada. Para obter informações sobre a reutilização de resultados de consulta, consulte ResultReuseByAgeConfiguration.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
EnableResultReuseByAge	nenhuma	Opcional	FALSE

Idade máxima para reutilização dos resultados

Especifica, em minutos, a idade máxima do resultado de uma consulta anterior que o Athena deve considerar para reutilização. Para obter informações sobre a idade máxima para reutilização de resultados, consulte ResultReuseByAgeConfiguration.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
MaxResultReuseAgeInMinutes	nenhuma	Opcional	60

Parâmetros de sondagem da execução de uma consulta

Intervalo mínimo para a sondagem da execução de uma consulta

O tempo mínimo, em milissegundos, a ser aguardado antes de sondar o Athena sobre o status da execução de uma consulta.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
MinQueryExecutionPollingIntervalMillis	MinQueryExecutionPollingInterval (obsoleto)	Opcional	100

Intervalo máximo para sondagem da execução de uma consulta

O tempo máximo, em milissegundos, a ser aguardado antes de sondar o Athena sobre o status da execução de uma consulta.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
MaxQueryExecutionPollingIntervalMillis	MaxQueryExecutionPollingInterval (obsoleto)	Opcional	5000

Multiplicador do intervalo de sondagem da execução de uma consulta

O fator de aumento do período de sondagem. Por padrão, a sondagem começa com o valor de `MinQueryExecutionPollingIntervalMillis` e dobra a cada sondagem até atingir o valor de `MaxQueryExecutionPollingIntervalMillis`.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
QueryExecutionPollingIntervalMultiplier	nenhuma	Opcional	2

Parâmetros de substituição de endpoint

Substituição do endpoint do Athena

O endpoint que o driver usará para fazer chamadas de API para o Athena.

Observe os seguintes pontos:

- Se os protocolos `https://` ou `http://` não forem especificados na URL fornecida, o driver inserirá o prefixo `https://`.
- Se esse parâmetro não for especificado, o driver usará um endpoint padrão.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
AthenaEndpoint	EndpointOverride (obsoleto)	Opcional	nenhuma

Substituição de endpoint de serviço de streaming do Athena

O endpoint que o driver usará para baixar os resultados das consultas quando usar o serviço de streaming do Athena. O serviço de streaming do Athena está disponível na porta 444.

Observe os seguintes pontos:

- Se os protocolos `https://` ou `http://` não forem especificados na URL fornecida, o driver inserirá o prefixo `https://`.
- Se uma porta não for especificada na URL fornecida, o driver inserirá a porta 444 do serviço de streaming.
- Se o parâmetro `AthenaStreamingEndpoint` não for especificado, o driver usará a substituição do `AthenaEndpoint`. Se nem o `AthenaStreamingEndpoint` nem a substituição do `AthenaEndpoint` forem especificados, o driver usará um endpoint de streaming padrão.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
AthenaStreamingEndpoint	StreamingEndpointOverride (obsoleto)	Opcional	nenhuma

Substituição de endpoint do LakeFormation

O endpoint que o driver usará para o serviço Lake Formation quando usar a API AssumeDecorateRoleWithSAML do AWS Lake Formation para recuperar credenciais temporárias. Se esse parâmetro não for especificado, o driver usará um endpoint padrão do Lake Formation.

Observe os seguintes pontos:

- Se os protocolos `https://` ou `http://` não forem especificados na URL fornecida, o driver inserirá o prefixo `https://`.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
LakeFormationEndpoint	LfEndpointOverride (obsoleto)	Opcional	nenhuma

Substituição de endpoint do S3

O endpoint que o driver usará para baixar os resultados das consultas quando usar o buscador do Amazon S3. Se esse parâmetro não for especificado, o driver usará um endpoint padrão do Amazon S3.

Observe os seguintes pontos:

- Se os protocolos `https://` ou `http://` não forem especificados na URL fornecida, o driver inserirá o prefixo `https://`.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
S3Endpoint	Nenhum	Opcional	nenhuma

Substituição de endpoint do STS

O endpoint que o driver usará para o serviço AWS STS quando usar a API AssumeRoleWithSAML do AWS STS para recuperar as credenciais temporárias. Se esse parâmetro não for especificado, o driver usará um endpoint padrão do AWS STS.

Observe os seguintes pontos:

- Se os protocolos `https://` ou `http://` não forem especificados na URL fornecida, o driver inserirá o prefixo `https://`.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
StsEndpoint	StsEndpointOverrid e(obsoleto)	Opcional	nenhuma

Parâmetros de configuração de proxy

Host do proxy

A URL do host proxy. Use esse parâmetro se você precisar que as solicitações do Athena passem por um proxy.

> **Note**
>
> Certifique-se de incluir o protocolo `https://` ou `http://` no início da URL do `ProxyHost`.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
ProxyHost	nenhuma	Opcional	nenhuma

Porta do proxy

A porta a ser usada no host proxy. Use esse parâmetro se você precisar que as solicitações do Athena passem por um proxy.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
ProxyPort	nenhuma	Opcional	nenhuma

Nome de usuário do proxy

O nome de usuário para autenticação no servidor proxy. Use esse parâmetro se você precisar que as solicitações do Athena passem por um proxy.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
ProxyUsername	ProxyUID (obsoleto)	Opcional	nenhuma

Senha do proxy

A senha para a autenticação no servidor proxy. Use esse parâmetro se você precisar que as solicitações do Athena passem por um proxy.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
ProxyPassword	ProxyPWD (obsoleto)	Opcional	nenhuma

Hosts isentos de proxy

Um conjunto dos nomes de host aos quais o driver se conecta sem usar um proxy quando o uso de proxy está habilitado (ou seja, quando os parâmetros de conexão `ProxyHost` e `ProxyPort` estão definidos). Os hosts devem ser separados pelo caractere barra vertical (`|`) (por exemplo, `host1.com|host2.com`).

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
ProxyExemptHosts	NonProxyHost	Opcional	nenhuma

Proxy habilitado para provedores de identidades

Especifica se um proxy deverá ser usado quando o driver se conectar a um provedor de identidades.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
ProxyEnabledForIdP	UseProxyForIdP	Opcional	FALSE

Parâmetros de registro em log

Esta seção descreve os parâmetros relacionados a registro em log.

Nível de log

Especifica o nível para o registro em log do driver. Nada é registrado em log, a menos que o parâmetro `LogPath` também esteja definido.

> **Note**
>
> Recomendamos definir somente o parâmetro `LogPath`, a menos que você tenha requisitos especiais. Definir somente o parâmetro `LogPath` habilita o registro em log e usa o nível de log padrão `TRACE`. O nível do registro em log `TRACE` fornece o registro em log mais detalhado.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão	Possíveis valores
LogLevel	nenhuma	Opcional	TRACE	OFF, ERROR, WARN, INFO, DEBUG, TRACE

Caminho do log.

O caminho para um diretório no computador que executa o driver no qual os logs do driver serão armazenados. Um arquivo de log com um nome exclusivo será criado no diretório especificado. Se definido, habilita o registro em log do driver.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
LogPath	nenhuma	Opcional	nenhuma

Nome da aplicação

O nome da aplicação que usa o driver. Se um valor for especificado para esse parâmetro, esse valor será incluído na string de agente do usuário das chamadas de API que o driver fizer para o Athena.

> **Note**
>
> Você também pode definir o nome da aplicação chamando `setApplicationName` no objeto `DataSource`.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
ApplicationName	nenhuma	Opcional	nenhuma

Teste de conexão

Se definido como `TRUE`, o driver executa um teste de conexão toda vez que uma conexão JDBC é criada, mesmo que uma consulta não seja executada na conexão.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
ConnectionTest	nenhuma	Opcional	VERDADEIRO

> **Note**
>
> Um teste de conexão envia uma consulta `SELECT 1` ao Athena para verificar se a conexão foi configurada corretamente. Isso significa que dois arquivos serão armazenados no Amazon S3 (o conjunto de resultados e os metadados), e tarifas adicionais podem ser aplicadas de acordo com a política de preços do Amazon Athena.

Número de novas tentativas

O número máximo de vezes que o driver deve reenviar uma solicitação recuperável para o Athena.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
NumRetries	MaxErrorRetry (obsoleto)	Opcional	nenhuma

Parâmetros de conexão de autenticação

O driver JDBC 3.x do Athena é compatível com vários métodos de autenticação. Os parâmetros de conexão necessários dependem do método de autenticação usado.

Tópicos

- Credenciais do IAM
- Credenciais padrão
- Credenciais do perfil de configuração da AWS
- Credenciais de perfil de instância
- Credenciais personalizadas
- Credenciais do JWT
- Credenciais do Azure AD
- Credenciais do Okta
- Uso do Ping
- Credenciais do AD FS
- Credenciais do Browser Azure AD
- Credenciais do Browser SAML

Credenciais do IAM

Você pode usar suas credenciais do IAM para se conectar ao Amazon Athena com o driver JDBC definindo os parâmetros de conexão a seguir.

Usuário

Seu ID de chave de acesso da AWS. Para obter informações sobre chaves de acesso, consulte Credenciais de segurança da AWS no Guia do usuário do IAM.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
Usuário	AccessKeyId	Obrigatório	nenhuma

Senha

O ID da chave secreta da AWS. Para obter informações sobre chaves de acesso, consulte Credenciais de segurança da AWS no Guia do usuário do IAM.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
Senha	SecretAccessKey	Opcional	nenhuma

Session token (Token da sessão)

Caso esteja usando credenciais temporárias da AWS, você deve especificar o token da sessão. Para obter informações sobre credenciais temporárias, consulte Credenciais de segurança temporárias no IAM, no Guia do usuário do IAM.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
SessionToken	nenhuma	Opcional	nenhuma

Credenciais padrão

Você pode usar as credenciais padrão que configura no sistema cliente para se conectar ao Amazon Athena definido os parâmetros de conexão a seguir. Para obter informações sobre o uso de credenciais padrão, consulte Usar a cadeia de fornecedores de credenciais padrão no Guia do desenvolvedor do AWS SDK for Java.

Provedor de credenciais

O provedor de credenciais que será usado para autenticar solicitações à AWS. Defina o valor desse parâmetro como `DefaultChain`.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão	Valor a ser usado
CredentialsProvider	AWSCredentialsProviderClass (obsoleto)	Obrigatório	nenhuma	`DefaultChain`

Credenciais do perfil de configuração da AWS

Você pode usar as credenciais armazenadas em um perfil de configuração da AWS definindo os parâmetros de conexão a seguir. Os perfis de configuração da AWS geralmente são armazenados em arquivos no diretório `~/.aws`. Para obter informações sobre os perfis de configuração da AWS, consulte Use profiles no AWS SDK for Java Developer Guide.

Provedor de credenciais

O provedor de credenciais que será usado para autenticar solicitações à AWS. Defina o valor desse parâmetro como `ProfileCredentials`.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão	Valor a ser usado
CredentialsProvider	AWSCredentialsProviderClass (obsoleto)	Obrigatório	nenhuma	`ProfileCredentials`

Profile name

O nome do perfil de configuração da AWS cujas credenciais devem ser usadas para autenticar a solicitação ao Athena.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
ProfileName	nenhuma	Obrigatório	nenhuma

> **Note**
>
> O nome do perfil também pode ser especificado como o valor do parâmetro `CredentialsProviderArguments`, embora esse uso esteja obsoleto.

Credenciais de perfil de instância

Esse tipo de autenticação é usado em instâncias do Amazon EC2. Um perfil de instância é um perfil associado a uma instância do Amazon EC2. O uso de um provedor de credenciais de perfil de instância delega o gerenciamento de credenciais da AWS ao Serviço de metadados de instância do Amazon EC2. Isso elimina a necessidade de desenvolvedores armazenarem credenciais permanentemente na instância do Amazon EC2 ou de se preocuparem com o rodízio ou o gerenciamento de credenciais temporárias.

Provedor de credenciais

O provedor de credenciais que será usado para autenticar solicitações à AWS. Defina o valor desse parâmetro como `InstanceProfile`.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão	Valor a ser usado
CredentialsProvider	AWSCredentialsProviderClass (obsoleto)	Obrigatório	nenhuma	`InstanceProfile`

Credenciais personalizadas

Você pode usar esse tipo de autenticação para fornecer suas próprias credenciais usando uma classe Java que implemente a interface AwsCredentialsProvider.

Provedor de credenciais

O provedor de credenciais que será usado para autenticar solicitações à AWS. Defina o valor desse parâmetro como o nome de classe totalmente qualificado da classe personalizada que implementa a interface AwsCredentialsProvider. No runtime, essa classe deve estar no caminho da classe Java da aplicação que usa o driver JDBC.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão	Valor a ser usado
CredentialsProvider	AWSCredentialsProviderClass (obsoleto)	Obrigatório	nenhuma	O nome da classe totalmente qualificada da implementação personalizada do `AwsCredentialsProvider`

Argumentos do provedor de credenciais

Uma lista separada por vírgulas de argumentos de string para o construtor do provedor de credenciais personalizadas.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
CredentialsProviderArguments	AwsCredentialsProviderArguments (obsoleto)	Opcional	nenhuma

Credenciais do JWT

Com esse tipo de autenticação, você pode usar um token Web JSON (JWT) obtido de um provedor de identidades externo como parâmetro de conexão para autenticação no Athena. O provedor externo de credenciais já deve ser federado com a AWS.

Provedor de credenciais

O provedor de credenciais que será usado para autenticar solicitações à AWS. Defina o valor desse parâmetro como `JWT`.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão	Valor a ser usado
CredentialsProvider	AWSCredentialsProv	Obrigatório	nenhuma	`JWT`

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão	Valor a ser usado
	iderClass (obsoleto)			

Token de identidade Web JWT

O token JWT obtido de um provedor de identidades federado externo. Esse token será usado para autenticação no Athena.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
JwtWebIdentityToken	web_identity_token (obsoleto)	Obrigatório	nenhuma

ARN de perfil do JWT

O nome do recurso da Amazon (ARN) da função a ser assumida. Para obter informações sobre como assumir perfis, consulte AssumeRole na AWS Security Token Service API Reference.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
JwtRoleArn	role_arn (obsoleto)	Obrigatório	nenhuma

Nome da sessão do perfil do JWT

O nome da sessão quando você usa as credenciais do JWT para autenticação. O nome pode ser qualquer nome que você escolher.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
JwtRoleSessionName	role_session_name (obsoleto)	Obrigatório	nenhuma

Credenciais do Azure AD

Um mecanismo de autenticação baseado em SAML que permite a autenticação no Athena usando o provedor de identidades do Azure AD. Esse método pressupõe que uma federação já tenha sido configurada entre o Athena e o Azure AD.

> **Note**
>
> Alguns dos nomes de parâmetros presentes nesta seção têm aliases. Os aliases são equivalentes funcionais dos nomes dos parâmetros e foram fornecidos para disponibilizar a compatibilidade retroativa com o driver JDBC 2.x. Como os nomes dos parâmetros foram aprimorados para seguir uma convenção de nomenclatura mais clara e consistente, recomendamos que você os use em vez dos aliases, que foram descontinuados.

Provedor de credenciais

O provedor de credenciais que será usado para autenticar solicitações à AWS. Defina o valor desse parâmetro como `AzureAD`.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão	Valor a ser usado
CredentialsProvider	AWSCredentialsProviderClass (obsoleto)	Obrigatório	nenhuma	`AzureAD`

Usuário

O endereço de e-mail do usuário do Azure AD a ser usado para autenticação no Azure AD.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
Usuário	UID (obsoleto)	Obrigatório	nenhuma

Senha

A senha do usuário do Azure AD.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
Senha	PWD (obsoleto)	Obrigatório	nenhuma

ID de locatário do Azure AD

O ID de locatário da aplicação do Azure AD.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
AzureAdTenantId	tenant_id (obsoleto)	Obrigatório	nenhuma

ID de cliente do Azure AD

O ID de cliente da aplicação do Azure AD.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
AzureAdClientId	client_id (obsoleto)	Obrigatório	nenhuma

Segredo de cliente do Azure AD

O segredo de cliente da aplicação do Azure AD.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
AzureAdClientSecret	client_secret (obsoleto)	Obrigatório	nenhuma

Perfil preferencial

O nome do recurso da Amazon (ARN) da função a ser assumida. Para obter informações sobre perfis de ARN, consulte AssumeRole na AWS Security Token Service API Reference.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
PreferredRole	preferred_role (obsoleto)	Opcional	nenhuma

Duração da sessão da função

A duração, em segundos, da sessão do perfil. Para obter mais informações, consulte AssumeRole na AWS Security Token Service API Reference.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
RoleSessionDuration	Duration (obsoleto)	Opcional	3600

Lake Formation habilitado

Especifica se a ação da API AssumeDecoratedRoleWithSAML do Lake Formation deverá ser usada para recuperar credenciais temporárias do IAM em vez da ação da API AssumeRoleWithSAML do AWS STS.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
LakeFormationEnabled	nenhuma	Opcional	FALSE

Credenciais do Okta

Um mecanismo de autenticação baseado em SAML que permite a autenticação no Athena usando o provedor de identidades do Okta. Esse método pressupõe que uma federação já tenha sido configurada entre o Athena e o Okta.

Provedor de credenciais

O provedor de credenciais que será usado para autenticar solicitações à AWS. Defina o valor desse parâmetro como `Okta`.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão	Valor a ser usado
CredentialsProvider	AWSCredentialsProviderClass (obsoleto)	Obrigatório	nenhuma	`Okta`

Usuário

O endereço de e-mail do usuário do Okta a ser usado para autenticação no Okta.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
Usuário	UID (obsoleto)	Obrigatório	nenhuma

Senha

A senha para o usuário do Okta.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
Senha	PWD (obsoleto)	Obrigatório	nenhuma

Nome de host do Okta

O URL de sua organização Okta. É possível extrair o parâmetro `idp_host` do URL do link de incorporação em sua aplicação Okta. Para obter as etapas, consulte Recuperar informações de configuração de ODBC do Okta. O primeiro segmento depois de `https://`, até `okta.com`, inclusive, é o host de IdP (por exemplo, `trial-1234567.okta.com` para uma URL que começa com `https://trial-1234567.okta.com`).

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
OktaHostName	IdP_Host (obsoleto)	Obrigatório	nenhuma

ID da aplicação do Okta

O identificador de duas partes da aplicação. Você pode extrair o ID da aplicação da URL do link embutido na aplicação do Okta. Para obter as etapas, consulte Recuperar informações de configuração de ODBC do Okta. O ID da aplicação são os dois últimos segmentos do URL, inclusive a barra no meio. Os segmentos são duas strings de 20 caracteres com uma mistura de números e letras maiúsculas e minúsculas (por exemplo, `Abc1de2fghi3J45kL678/abc1defghij2klmNo3p4`).

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
OktaAppId	App_ID (obsoleto)	Obrigatório	nenhuma

Nome da aplicação do Okta

O nome da aplicação do Okta.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
OktaAppName	App_Name (obsoleto)	Obrigatório	nenhuma

Tipo de MFA do Okta

Se você configurou o Okta para exigir a autenticação multifator (MFA), precisará especificar o tipo de MFA do Okta e os parâmetros adicionais, dependendo do segundo fator que quiser usar.

O tipo de MFA do Okta é o segundo tipo de fator de autenticação (depois da senha) a ser usado para autenticação no Okta. Os segundos fatores compatíveis incluem as notificações por push enviadas pela aplicação Okta Verify e as senhas temporárias de uso único (TOTPs) geradas pelo Okta Verify, pelo Google Authenticator ou enviadas por SMS. As políticas de segurança de cada organização determinam se a MFA é necessária ou não para o login do usuário.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão	Possíveis valores
OktaMfaType	okta_mfa_type (obsoleto)	Obrigatório, se o Okta for	nenhuma	`oktaverif ywithpush`

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão	Possíveis valores
		configurado para exigir MFA		`oktaverifywithtotp`, `googleauthenticator`, `smsauthentication`

Número de telefone do Okta

O número de telefone para o qual o Okta enviará uma senha temporária de uso único por SMS quando o tipo de `smsauthentication` MFA for escolhido. O número de telefone deve ser um número de telefone nos EUA ou no Canadá.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
OktaPhoneNumber	okta_phone_number (obsoleto)	Obrigatório, se OktaMfaType for `smsauthentication`	nenhuma

Tempo de espera da MFA do Okta

A duração, em segundos, de espera até que o usuário confirme o recebimento de uma notificação por push do Okta antes que o driver gere uma exceção de tempo limite.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
OktaMfaWaitTime	okta_mfa_wait_time (obsoleto)	Opcional	60

Perfil preferencial

O nome do recurso da Amazon (ARN) da função a ser assumida. Para obter informações sobre perfis de ARN, consulte AssumeRole na AWS Security Token Service API Reference.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
PreferredRole	preferred_role (obsoleto)	Opcional	nenhuma

Duração da sessão da função

A duração, em segundos, da sessão do perfil. Para obter mais informações, consulte AssumeRole na AWS Security Token Service API Reference.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
RoleSessionDuration	Duration (obsoleto)	Opcional	3600

Lake Formation habilitado

Especifica se a ação da API AssumeDecoratedRoleWithSAML do Lake Formation deverá ser usada para recuperar credenciais temporárias do IAM em vez da ação da API AssumeRoleWithSAML do AWS STS.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
LakeFormationEnabled	nenhuma	Opcional	FALSE

Uso do Ping

Um mecanismo de autenticação baseado em SAML que permite a autenticação no Athena usando o provedor de identidades Ping Federate. Esse método pressupõe que uma federação já tenha sido configurada entre o Athena e o Ping Federate.

Provedor de credenciais

O provedor de credenciais que será usado para autenticar solicitações à AWS. Defina o valor desse parâmetro como `Ping`.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão	Valor a ser usado
CredentialsProvider	AWSCredentialsProviderClass (obsoleto)	Obrigatório	nenhuma	`Ping`

Usuário

O endereço de e-mail do usuário do Ping Federate a ser usado para autenticação no Ping Federate.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
Usuário	UID (obsoleto)	Obrigatório	nenhuma

Senha

A senha do usuário do Ping Federate.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
Senha	PWD (obsoleto)	Obrigatório	nenhuma

PingHostName

O endereço do servidor Ping. Para encontrar seu endereço, acesse o URL a seguir e visualize o campo Endpoint da aplicação de SSO.

```
https://your-pf-host-#:9999/pingfederate/your-pf-app#/spConnections
```

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
PingHostName	IdP_Host (obsoleto)	Obrigatório	nenhuma

PingPortNumber

O número da porta a ser usada para se conectar ao host de IdP.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
PingPortNumber	IdP_Port (obsoleto)	Obrigatório	nenhuma

PingPartnerSpId

O endereço do provedor de serviços. Para encontrar o endereço do provedor de serviços, acesse o URL a seguir e visualize o campo Endpoint da aplicação de SSO.

```
https://your-pf-host-#:9999/pingfederate/your-pf-app#/spConnections
```

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
PingPartnerSpId	Partner_SPID (obsoleto)	Obrigatório	nenhuma

Perfil preferencial

O nome do recurso da Amazon (ARN) da função a ser assumida. Para obter informações sobre perfis de ARN, consulte AssumeRole na AWS Security Token Service API Reference.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
PreferredRole	preferred_role (obsoleto)	Opcional	nenhuma

Duração da sessão da função

A duração, em segundos, da sessão do perfil. Para obter mais informações, consulte AssumeRole na AWS Security Token Service API Reference.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
RoleSessionDuration	Duration (obsoleto)	Opcional	3600

Lake Formation habilitado

Especifica se a ação da API AssumeDecoratedRoleWithSAML do Lake Formation deverá ser usada para recuperar credenciais temporárias do IAM em vez da ação da API AssumeRoleWithSAML do AWS STS.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
LakeFormationEnabled	nenhuma	Opcional	FALSE

Credenciais do AD FS

Um mecanismo de autenticação baseado em SAML que viabiliza a autenticação no Athena usando o Microsoft Active Directory Federation Services (AD FS). Esse método pressupõe que o usuário já configurou uma federação entre o Athena e o AD FS.

Provedor de credenciais

O provedor de credenciais que será usado para autenticar solicitações à AWS. Defina o valor desse parâmetro como `ADFS`.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão	Valor a ser usado
CredentialsProvider	AWSCredentialsProviderClass (obsoleto)	Obrigatório	nenhuma	`ADFS`

Usuário

O endereço de e-mail do usuário do AD FS a ser usado para autenticação no AD FS.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
Usuário	UID (obsoleto)	Necessário para a autenticação baseada em formulário. Opcional para a autentica ção integrada do Windows.	nenhuma

Senha

A senha do usuário do AD FS.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
Senha	PWD (obsoleto)	Necessário para a autenticação baseada em formulário. Opcional para a autentica ção integrada do Windows.	nenhuma

Nome de host do ADFS

O endereço do servidor AD FS.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
AdfsHostName	IdP_Host (obsoleto)	Obrigatório	nenhuma

Número da porta do ADFS

O número da porta a ser usada para se conectar ao servidor do AD FS.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
AdfsPortNumber	IdP_Port (obsoleto)	Obrigatório	nenhuma

Parte confiável do ADFS

A parte confiável. Use esse parâmetro para substituir o URL do endpoint da parte confiável do AD FS.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
AdfsRelyingParty	LoginToRP (obsoleto)	Opcional	`urn:amazon:webservices`

WIA habilitado no ADFS

Booliano. Use esse parâmetro para habilitar a autenticação integrada do Windows (WIA) com o AD FS.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
AdfsWiaEnabled	none	Opcional	FALSE

Perfil preferencial

O nome do recurso da Amazon (ARN) da função a ser assumida. Para obter informações sobre perfis de ARN, consulte `AssumeRole` na Referência do API do AWS Security Token Service.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
PreferredRole	preferred_role (obsoleto)	Opcional	nenhuma

Duração da sessão da função

A duração, em segundos, da sessão do perfil. Para obter mais informações, consulte `AssumeRole` na Referência de APIs do AWS Security Token Service.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
RoleSessionDuration	Duration (obsoleto)	Opcional	`3600`

Lake Formation habilitado

Especifica se o sistema deve usar a ação de API `AssumeDecoratedRoleWithSAML` do Lake Formation para recuperar credenciais temporárias do IAM em vez da ação de API `AssumeRoleWithSAML` do AWS STS.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
LakeFormationEnabled	`none`	Opcional	FALSE

Credenciais do Browser Azure AD

O Browser Azure AD é um mecanismo de autenticação baseado em SAML que funciona com o provedor de identidades do Azure AD e é compatível com autenticação multifator. Ao contrário do mecanismo padrão de autenticação do Azure AD, esse mecanismo não requer nome de usuário, senha ou segredo do cliente nos parâmetros de conexão. Assim como o mecanismo de autenticação padrão do Azure AD, o Browser Azure AD também pressupõe que o usuário já tenha configurado a federação entre o Athena e o Azure AD.

Provedor de credenciais

O provedor de credenciais que será usado para autenticar solicitações à AWS. Defina o valor desse parâmetro como `BrowserAzureAD`.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão	Valor a ser usado
CredentialsProvider	AWSCredentialsProviderClass (obsoleto)	Obrigatório	nenhuma	`BrowserAzureAD`

ID de locatário do Azure AD

O ID de locatário da aplicação do Azure AD

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
AzureAdTenantId	tenant_id (obsoleto)	Obrigatório	nenhuma

ID de cliente do Azure AD

O ID de cliente da aplicação do Azure AD

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
AzureAdClientId	client_id (obsoleto)	Obrigatório	nenhuma

Tempo limite de resposta do provedor de identidades

O período de tempo, em segundos, até o driver parar de esperar a resposta SAML do Azure AD.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
IdpResponseTimeout	idp_response_timeout (obsoleto)	Opcional	120

Perfil preferencial

O nome do recurso da Amazon (ARN) da função a ser assumida. Para obter informações sobre perfis de ARN, consulte AssumeRole na AWS Security Token Service API Reference.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
PreferredRole	preferred_role (obsoleto)	Opcional	nenhuma

Duração da sessão da função

A duração, em segundos, da sessão do perfil. Para obter mais informações, consulte AssumeRole na AWS Security Token Service API Reference.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
RoleSessionDuration	Duration (obsoleto)	Opcional	3600

Lake Formation habilitado

Especifica se a ação da API AssumeDecoratedRoleWithSAML do Lake Formation deverá ser usada para recuperar credenciais temporárias do IAM em vez da ação da API AssumeRoleWithSAML do AWS STS.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
LakeFormationEnabled	nenhuma	Opcional	FALSE

Credenciais do Browser SAML

O Browser SAML é um plug-in de autenticação genérico que pode funcionar com provedores de identidades baseados em SAML e é compatível com autenticação multifator.

Provedor de credenciais

O provedor de credenciais que será usado para autenticar solicitações à AWS. Defina o valor desse parâmetro como `BrowserSaml`.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão	Valor a ser usado
CredentialsProvider	AWSCredentialsProviderClass (obsoleto)	Obrigatório	nenhuma	`BrowserSaml`

URL de login de autenticação única

O URL de autenticação única para a aplicação no provedor de identidades baseado em SAMAL.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
SsoLoginUrl	login_url (obsoleto)	Obrigatório	nenhuma

Escutar porta

O número da porta que é usada para receber a resposta do SAML. Esse valor deve corresponder à URL com a qual você configurou o provedor de identidades baseado em SAML (por exemplo, `http://localhost:7890/athena`).

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
ListenPort	listen_port (obsoleto)	Opcional	`7890`

Tempo limite de resposta do provedor de identidades

O período de tempo, em segundos, até o driver parar de esperar a resposta SAML do Azure AD.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
IdpResponseTimeout	idp_response_timeout (obsoleto)	Opcional	120

Perfil preferencial

O nome do recurso da Amazon (ARN) da função a ser assumida. Para obter informações sobre perfis de ARN, consulte AssumeRole na AWS Security Token Service API Reference.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
PreferredRole	preferred_role (obsoleto)	Opcional	nenhuma

Duração da sessão da função

A duração, em segundos, da sessão do perfil. Para obter mais informações, consulte AssumeRole na AWS Security Token Service API Reference.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
RoleSessionDuration	Duration (obsoleto)	Opcional	3600

Lake Formation habilitado

Especifica se a ação da API AssumeDecoratedRoleWithSAML do Lake Formation deverá ser usada para recuperar credenciais temporárias do IAM em vez da ação da API AssumeRoleWithSAML do AWS STS.

Nome do parâmetro	Alias	Tipo de parâmetro	Valor padrão
LakeFormationEnabled	nenhuma	Opcional	FALSE

Outra configuração do JDBC 3.x

As seções a seguir descrevem algumas configurações adicionais para o driver JDBC 3.x.

Tempo limite de rede

O tempo, em milissegundos, durante o qual o driver aguardará uma resposta quando fizer uma chamada de API para o Athena. Após esse período, o driver gera uma exceção de tempo limite.

O tempo limite da rede não pode ser definido como um parâmetro de conexão. Para configurá-lo, chame o método `setNetworkTimeout` em um objeto `Connection` JDBC. Esse valor pode ser alterado durante o ciclo de vida da conexão JDBC. O valor padrão desse parâmetro é `infinity`.

O exemplo a seguir define o tempo limite de rede como 5.000 milissegundos.

```
...
AthenaDriver driver = new AthenaDriver();
Connection connection = driver.connect(url, connectionParameters);
connection.setNetworkTimeout(null, 5000);
...
```

Tempo limite de consulta

O tempo, em segundos, que o driver aguardará a conclusão de uma consulta no Athena após enviá-la. Após esse período, o driver tenta cancelar a consulta enviada e gera uma exceção de tempo limite.

O tempo limite da consulta não pode ser definido como um parâmetro de conexão. Para configurá-lo, chame o método `setQueryTimeout` em um objeto `Statement` JDBC. Esse valor pode ser alterado durante o ciclo de vida de uma instrução JDBC. O valor padrão desse parâmetro é `0` (zero). Um valor `0` significa que as consultas podem ser executadas até a sua conclusão (respeitados os Service Quotas).

Os exemplos a seguir definem o tempo limite de consulta como 5.000 milissegundos.

```
...
AthenaDriver driver = new AthenaDriver();
Connection connection = driver.connect(url, connectionParameters);
Statement statement = connection.createStatement();
statement.setQueryTimeout(5);
```

```
...
```

Notas de versão do JDBC 3.x do Amazon Athena

Estas notas de versão fornece detalhes de melhorias e correções no driver JDBC 3.x do Amazon Athena.

3.2.0

Lançado em 26/04/2024

Melhorias

- Desempenho da operação de catálogo: o desempenho foi aprimorado para operações de catálogo que não usam caracteres curinga.
- Alteração do intervalo mínimo de pesquisa: o padrão do intervalo mínimo de pesquisa foi modificado para reduzir o número de chamadas de API que o driver faz para o Athena. As conclusões de consultas ainda são detectadas o mais rápido possível.
- Possibilidade de descoberta de ferramentas de BI: o driver ficou mais facilmente detectável para ferramentas de business intelligence.
- Mapeamento de tipos de dados: o mapeamento de tipos de dados para os tipos de dados DDL do Athena `binary`, `array` e `struct`foi aprimorado.
- AWSVersão do SDK: a versão do AWS SDK usada no driver foi atualizada para 2.25.34.

Correções

- Listagens de tabelas de catálogos federados: corrigido um problema que fazia com que catálogos federados retornassem uma lista vazia de tabelas.
- getSchemas: corrigido um problema que fazia com que o método JDBC DatabaseMetaData#getSchemas buscasse bancos de dados somente do catálogo padrão, e não de todos os catálogos.
- getColumns: corrigido um problema que fazia com que um catálogo nulo fosse retornado quando o método JDBC DatabaseMetaData#getSchemas era chamado com um nome de catálogo nulo.

3.1.0

Lançado em 15/02/2024

Melhorias

- Suporte adicionado para autenticação integrada do Windows por meio do Microsoft Active Directory Federation Services (AD FS) e autenticação com base em formulário.
- Para compatibilidade com versões anteriores da versão 2.x, o subprotocolo JDBC `awsathena` agora é aceito, mas produz um aviso de depreciação. Em vez disso, use o subprotocolo JDBC `athena`.
- `AwsDataCatalog` agora é o padrão para o parâmetro do catálogo, e `default` é o padrão para o parâmetro do banco de dados. Essas alterações garantem que os valores corretos do catálogo e do banco de dados atuais sejam retornados em vez de nulos.
- Em conformidade com a especificação JDBC, `IS_AUTOINCREMENT` e `IS_GENERATEDCOLUMN` agora retornam uma string vazia em vez de `NO`.
- O tipo de dados `int` do Athena agora é mapeado para o mesmo tipo de JDBC `integer` do Athena, em vez de para `other`.
- Quando os metadados da coluna do Athena não contêm os campos opcionais `precision` e `scale`, o driver agora retorna zero para os valores correspondentes em uma coluna `ResultSet`.
- A versão do AWS SDK foi atualizada para 2.21.39.

Correções

- Foi corrigido um problema com `GetQueryResultsStream` que causava a ocorrência de uma exceção quando resultados de texto simples do Athena tinham uma contagem de colunas inconsistente com a contagem de colunas nos metadados de resultados do Athena.

3.0.0

Lançado em 16/11/2023

O driver Athena JDBC 3.x é o driver de nova geração que oferece melhor desempenho e compatibilidade. O driver JDBC 3.x é compatível com a leitura de resultados de consultas diretamente no Amazon S3, o que melhora o desempenho de aplicações que consomem resultados de consultas de grande porte. O novo driver também tem menos dependências de terceiros, o que facilita a integração com ferramentas de BI e aplicações personalizadas.

Versões anteriores do driver JDBC 3.x driver do Athena

Recomendamos muito que você use a versão mais recente do driver JDBC 3.x. A versão mais recente do driver contém as melhorias e correções mais recentes. Use uma versão mais antiga somente se sua aplicação apresentar incompatibilidades com a versão mais recente.

Jar com dependências do driver JDBC

O download a seguir empacota o driver e todas as suas dependências no mesmo arquivo `.jar`. Esse download é comumente usado para clientes SQL de terceiros.

- 3.1.0 uber jar
- 3.0.0 uber jar

Jar sem dependências do driver JDBC

O download a seguir é um arquivo `.zip` que contém o `.jar` sem dependências para o driver e arquivos `.jar` separados para as dependências do driver. Esse download é comumente usado para aplicações personalizadas que podem ter dependências conflitantes com as dependências que o driver usa. Esse download é útil se você quiser escolher quais dependências do driver incluir no jar sem dependências e quais excluir se a aplicação personalizada já contiver uma ou mais dependências.

- 3.1.0 lean jar
- 3.0.0 lean jar

Driver JDBC 2.x do Athena

Você pode usar uma conexão JDBC para conectar o Athena às ferramentas de inteligência de negócios e outras aplicações, como o SQL Workbench. Para isso, use os links do Amazon S3 nesta página para baixar, instalar e configurar o driver JDBC 2.x do Athena. Para obter informações sobre como criar o URL da conexão JDBC, consulte o guia de instalação e configuração do driver JDBC, disponível para download. Para obter informações sobre permissões, consulte Acesso por meio de conexões JDBC e ODBC. Para enviar comentários sobre o driver JDBC, envie um e-mail para athena-feedback@amazon.com. A partir da versão 2.0.24, duas versões do driver estão disponíveis: uma que inclui o AWS SDK e uma que não o inclui.

Important

Ao usar o driver JDBC, observe os seguintes requisitos:

- Porta 444 aberta: mantenha a porta 444, que o Athena usa para fazer uma transmissão dos resultados das consultas, aberta para o tráfego de saída. Ao usar um endpoint do PrivateLink para se conectar ao Athena, verifique se o grupo de segurança anexado ao endpoint do PrivateLink está aberto para o tráfego de entrada na porta 444. Se a porta 444 estiver bloqueada, você pode receber a mensagem de erro [Simba][AthenaJDBC](100123) Ocorreu um erro. Exceção durante a inicialização da coluna.
- Política athena:GetQueryResultsStream: adicione a ação de política `athena:GetQueryResultsStream` para as entidades principais do IAM que usam o driver JDBC. Essa ação de política não é exposta diretamente com a API. Ela apenas é usada com drivers ODBC e JDBC como parte do suporte a resultados de transmissão. Para visualizar um exemplo de política, consulte Política gerenciada AWS: AWSQuicksightAthenaAccess.
- Uso do driver JDBC para diversos catálogos de dados: para usar o driver JDBC para diversos catálogos de dados com o Athena (por exemplo, ao usar um metastore do Hive externo ou consultas federadas), inclua `MetadataRetrievalMethod=ProxyAPI` em sua string de conexão JDBC.
- Drivers 4.1: a partir de 2023, a compatibilidade do driver para a versão 4.1 do JDBC será descontinuado. Nenhuma atualização adicional será lançada. Se você estiver usando um driver JDBC 4.1, é altamente recomendado fazer a migração para o driver 4.2.

Driver JDBC 2.x com AWS SDK

A versão 2.1.5 do driver JDBC está em conformidade com o padrão de dados da API 4.2 do JDBC e requer o JDK 8.0 ou versões posteriores. Para obter informações sobre como verificar a versão do Java Runtime Environment (JRE) que você usa, consulte a documentação do Java.

Use o link a seguir para baixar do arquivo `.jar` do driver JDBC 4.2.

- AthenaJDBC42-2.1.5.1000.jar

O download do arquivo `.zip` contém o arquivo `.jar` para o JDBC 4.2 e inclui o SDK da AWS e a documentação, as notas de lançamento, as licenças e os contratos que o acompanham.

- SimbaAthenaJDBC-2.1.5.1000.zip

Driver JDBC 2.x sem AWS SDK

A versão 2.1.5 do driver JDBC está em conformidade com o padrão de dados da API 4.2 do JDBC e requer o JDK 8.0 ou versões posteriores. Para obter informações sobre como verificar a versão do Java Runtime Environment (JRE) que você usa, consulte a documentação do Java.

Use o link a seguir para baixar o arquivo `.jar` do driver JDBC 4.2 sem o SDK da AWS.

- AthenaJDBC42-2.1.5.1001.jar

O download do arquivo `.zip` contém o arquivo `.jar` para o JDBC 4.2 e a documentação, as notas de lançamento, as licenças e os contratos que o acompanham. O AWS SDK não está incluído.

- SimbaAthenaJDBC-2.1.5.1001.zip

Notas da versão, contrato de licença e avisos do driver JDBC 2.x

Depois de fazer download da versão de que você precisa, leia as notas de release e revise o Contrato de licença e as notificações.

- Notas de lançamento
- Contrato de licença
- Notificações
- Licenças de terceiros

Documentação do driver JDBC 2.x

Faça download da seguinte documentação do driver:

- Guia de instalação e configuração do driver JDBC. Use este guia para instalar e configurar o driver.
- Guia de migração do driver JDBC. Use este guia para migrar de versões anteriores para a versão atual.

Conectar-se ao Amazon Athena com ODBC

O Amazon Athena oferece dois drivers ODBC, versões 1.x e 2.x. O driver ODBC 2.x do Athena é uma nova alternativa que é compatível com sistemas Linux, macOS ARM, macOS Intel e Windows 64 bits. O driver Athena 2.x oferece suporte a todos os plug-ins de autenticação compatíveis com o driver ODBC 1.x, e quase todos os parâmetros de conexão são compatíveis com versões anteriores.

- Para baixar o driver ODBC 2.x, consulte Amazon Athena ODBC 2.x.
- Para baixar o driver ODBC 1.x, consulte Driver ODBC 1.x do Athena.

Tópicos

- Amazon Athena ODBC 2.x
- Driver ODBC 1.x do Athena
- Usar o conector do Power BI para Amazon Athena

Amazon Athena ODBC 2.x

Você pode usar uma conexão ODBC para se conectar ao Amazon Athena por muitas ferramentas e aplicações de clientes SQL de terceiros. Configure a conexão ODBC no computador cliente.

Considerações e limitações

- Para obter informações sobre como migrar o driver ODBC 1.x do Athena para o driver ODBC 2.x do Athena, consulte Migrar para o driver ODBC 2.x.
- Ao usar o captador do S3 com a opção de criptografia do `CSE_KMS`, o cliente do Amazon S3 não poderá descriptografar o resultado armazenado no bucket do Amazon S3. Como solução alternativa, use a opção da API de transmissão do Athena para buscar o conjunto de resultados.

Download do driver ODBC 2.x

Para baixar o driver ODBC do Amazon Athena 2.x, acesse os links nesta página.

> **Important**
>
> Ao usar o driver ODBC 2.x, observe os seguintes requisitos:

- Porta 444 aberta: mantenha a porta 444, que o Athena usa para fazer uma transmissão dos resultados das consultas, aberta para o tráfego de saída. Ao usar um endpoint do PrivateLink para se conectar ao Athena, verifique se o grupo de segurança anexado ao endpoint do PrivateLink está aberto para o tráfego de entrada na porta 444.
- Política athena:GetQueryResultsStream: adicione a ação de política `athena:GetQueryResultsStream` às entidades principais do IAM que usam o driver ODBC. Essa ação de política não é exposta diretamente com a API. Ela apenas é usada com drivers ODBC e JDBC como parte do suporte a resultados de transmissão. Para visualizar um exemplo de política, consulte Política gerenciada AWS: AWSQuicksightAthenaAccess.

Linux

Versão do driver	Link para fazer download
ODBC 2.0.3.0 para Linux 64 bits	Driver ODBC 2.0.3.0 para Linux 64 bits

macOS (ARM)

Versão do driver	Link para fazer download
ODBC 2.0.3.0 para macOS 64 bits (ARM)	Driver ODBC 2.0.3.0 para macOS 64 bits (ARM)

macOS (Intel)

Versão do driver	Link para fazer download
ODBC 2.0.3.0 para macOS 64 bits (Intel)	Driver ODBC 2.0.3.0 para macOS 64 bits (Intel)

Windows

Versão do driver	Link para fazer download
ODBC 2.0.3.0 para Windows 64 bits	Driver ODBC 2.0.3.0 para Windows 64 bits

Tópicos

- Conceitos básicos do driver ODBC 2.x
- Parâmetros de conexão do ODBC 2.x do Athena
- Migrar para o driver ODBC 2.x
- Solução de problemas do driver ODBC 2.x
- Notas da versão do ODBC 2.x do Amazon Athena

Conceitos básicos do driver ODBC 2.x

Utilize as informações desta seção para começar a usar o driver ODBC 2.x do Amazon Athena. O driver é compatível com os sistemas operacionais Windows, Linux e macOS.

Tópicos

- Windows
- Linux
- macOS

Windows

Se quiser usar um computador cliente com Windows para acessar o Amazon Athena, é necessário ter o driver ODBC do Amazon Athena.

Requisitos do sistema Windows

Instale o driver ODBC do Amazon Athena em computadores clientes que acessarão os bancos de dados do Amazon Athena diretamente em vez de usar um navegador da Web.

O sistema Windows que você usa deve atender aos seguintes requisitos:

- Você tem direitos de administrador
- Um dos seguintes sistemas operacionais:
  - Windows 11, 10 ou 8.1
  - Windows Server 2019, 2016 ou 2012
- Pelo menos 100 MB de espaço em disco disponível
- O Microsoft Visual C++ redistribuível para Visual Studio para Windows de 64 bits está instalado.

Instalar o driver ODBC do Amazon Athena

Para baixar e instalar o driver ODBC do Amazon Athena para Windows

1. Baixe o arquivo de instalação `AmazonAthenaODBC-2.x.x.x.msi`.
2. Inicie o arquivo de instalação e escolha Próximo.
3. Para aceitar os termos do contrato de licença, marque a caixa de seleção e escolha Próximo.
4. Para alterar o local da instalação, escolha Procurar, navegue até a pasta desejada e escolha OK.
5. Para aceitar o local da instalação, escolha Próximo.
6. Escolha Instalar.
7. Quando a instalação terminar, escolha Concluir.

Modos de definir as opções de configuração do driver

Para controlar o comportamento do driver ODBC do Amazon Athena no Windows, é possível definir as opções de configuração do driver destes modos:

- No programa Administrador de Fonte de Dados ODBC, ao configurar um nome de fonte de dados (DSN).
- Adicionando ou alterando as chaves de registro do Windows no seguinte local:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI\YOUR_DSN_NAME
```

- Definindo as opções do driver na string de conexão quando você se conecta de maneira programática.

Configurar um nome de fonte de dados no Windows

Depois de baixar e instalar o driver ODBC, é necessário adicionar uma entrada de nome de fonte de dados (DSN) ao computador cliente ou instância do Amazon EC2. As ferramentas de cliente SQL usam essa fonte de dados para se conectar ao banco de dados do Amazon Athena.

Para criar uma entrada de DSN do sistema

1. No menu Iniciar do Windows, clique com o botão direito do mouse em Fontes de dados ODBC (64 bits) e escolha Mais, Executar como administrador.
2. No Administrador de fonte de dados ODBC, escolha a guia Drivers.
3. Na coluna Nome, verifique se o ODBC do Amazon Athena (x64) está presente.
4. Execute um destes procedimentos:

   - Para configurar o driver para todos os usuários do computador, escolha a guia DSN do sistema. Como as aplicações que usam uma conta diferente para carregar dados podem não conseguir detectar DSNs de usuários de outra conta, é recomendável usar a opção de configuração de DSN do sistema.

     > **Note**
     >
     > Usar a opção DSN do sistema requer privilégios administrativos.

   - Para configurar o driver somente para sua conta de usuário, escolha a guia DSN do usuário.
5. Escolha Adicionar. A caixa de diálogo Criar nova fonte de dados é exibida.
6. Escolha o ODBC do Amazon Athena (x64) e depois Concluir.
7. Na caixa de diálogo Configuração do ODBC do Amazon Athena, insira as informações a seguir. Para obter informações detalhadas sobre essas opções, consulte Principais parâmetros de conexão do ODBC 2.x.

   - Em Nome da fonte de dados, insira o nome que você deseja usar para identificar a fonte de dados.
   - Em Descrição, insira uma descrição que ajude a identificar a fonte de dados.
   - Em Região, insira o nome da Região da AWS em que você usará o Athena (por exemplo, **`us-west-1`**).
   - Em Catálogo, insira o nome do catálogo do Amazon Athena. O padrão é AWSDataCatalog, que é usado pelo AWS Glue.

- Em Banco de dados, insira o nome do banco de dados do Amazon Athena. O padrão é padrão.
- Em Grupo de trabalho, insira o nome do grupo de trabalho do Amazon Athena. O padrão é primário.
- Em Local de saída do S3, insira o local do Amazon S3 em que os resultados da consulta serão armazenados (por exemplo, **`s3://DOC-EXAMPLE-BUCKET/`**).
- (Opcional) Em Opções de criptografia, escolha uma opção de criptografia. O padrão é `NOT_SET`.
- (Opcional) Em Chave do KMS, escolha uma chave de criptografia do KMS, se necessário.

8. Para especificar as opções de configuração para autenticação do IAM, escolha Opções de autenticação.
9. Insira as seguintes informações:

   - Em Tipo de autenticação, escolha Credenciais do IAM. Esse é o padrão. Para obter mais informações sobre os tipos de autenticação disponíveis, consulte Opções de autenticação.
   - Em Nome de usuário, insira um nome de usuário.
   - Em Senha, insira uma senha.
   - Em Token de sessão, insira um token de sessão se quiser usar credenciais temporárias da AWS. Para obter informações sobre credenciais temporárias, consulte Uso de credenciais temporárias com recursos da AWS no Guia do usuário do IAM.

10. Escolha OK.
11. Na parte inferior da caixa de diálogo Configuração do ODBC do Amazon Athena, escolha Testar. Se o computador cliente se conectar com êxito ao Amazon Athena, a caixa Teste de conexão informará Conexão com êxito. Caso contrário, a caixa informará Falha na conexão com as informações de erro correspondentes.
12. Escolha OK para fechar o teste de conexão. A fonte de dados que você criou já aparece na lista de nomes de fontes de dados.

Usar uma conexão sem DSN no Windows

É possível usar uma conexão sem DSN para se conectar a um banco de dados sem um nome de fonte de dados (DSN). O exemplo a seguir mostra uma string de conexão para o driver ODBC do Amazon Athena (x64) que se conecta ao 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

Se quiser usar um computador cliente com Linux para acessar o Amazon Athena, é necessário ter o driver ODBC do Amazon Athena.

Requisitos do sistema Linux

Cada computador cliente Linux no qual você instale o driver deve atender aos seguintes requisitos.

- Ter acesso root.
- Usar umas das seguintes distribuições do Linux:
  - Red Hat Enterprise Linux (RHEL) 7 ou 8
  - CentOS 7 ou 8.
- Ter 100 MB de espaço em disco disponível.
- Usar a versão 2.3.1 ou superior do unixODBC.
- Usar a versão 2.26 ou posterior da GNU C Library (glibc).

Como instalar o conector de dados ODBC no Linux

Siga o procedimento abaixo para instalar o driver ODBC do Amazon Athena em um sistema operacional Linux.

Para instalar o driver ODBC do Amazon Athena no Linux

1. Insira um dos seguintes comandos:

   ```
   sudo rpm -Uvh AmazonAthenaODBC-2.X.Y.Z.rpm
   ```

   ou

   ```
   sudo yum --nogpgcheck localinstall AmazonAthenaODBC-2.X.Y.Z.rpm
   ```

2. Após a conclusão da instalação, insira um dos seguintes comandos para verificar se o driver está instalado:

- ```
  yum list | grep amazon-athena-odbc-driver
  ```

  Saída:

  ```
  amazon-athena-odbc-driver.x86_64 2.0.2.1-1.amzn2int installed
  ```

- ```
  rpm -qa | grep amazon
  ```

  Saída:

  ```
  amazon-athena-odbc-driver-2.0.2.1-1.amzn2int.x86_64
  ```

Configurar um nome de fonte de dados no Linux

Após a instalação do driver, você poderá encontrar exemplos de arquivos `.odbc.ini` e `.odbcinst.ini` no seguinte local:

- `/opt/athena/odbc/ini/`.

Use os arquivos `.ini` nesse local como exemplos para configurar o driver ODBC e o nome da fonte de dados (DSN) do Amazon Athena.

> **Note**
>
> Por padrão, os gerenciadores de driver ODBC usam os versões ocultas dos arquivos de configuração `.odbc.ini` e `.odbcinst.ini`, localizadas no diretório inicial.

Para especificar o caminho dos arquivos `.odbc.ini` e `.odbcinst.ini` usando o unixODBC, execute as etapas a seguir.

Para especificar localizações de arquivos **`.ini`** do ODBC usando unixODBC

1. Defina `ODBCINI` com o caminho completo e o nome de arquivo do arquivo `odbc.ini`, como no exemplo a seguir.

```
export ODBCINI=/opt/athena/odbc/ini/odbc.ini
```

2. Defina `ODBCSYSINI` com o caminho completo do diretório que contém o arquivo `odbcinst.ini`, como no exemplo a seguir.

```
export ODBCSYSINI=/opt/athena/odbc/ini
```

3. Digite o comando a seguir para verificar se você está usando o gerenciador de drivers UnixODBC e os arquivos `odbc*.ini` corretos:

```
username % odbcinst -j
```

Exemplo de saída

```
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. Se quiser usar um nome da fonte de dados (DSN) para estabelecer conexão com seu armazenamento de dados, configure o arquivo `odbc.ini` para definir nomes de fonte de dados (DSNs). Defina as propriedades no arquivo `odbc.ini` para criar um DSN que especifique as informações de conexão para seu armazenamento de dados, como no exemplo a seguir.

```
[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. Configure o arquivo `odbcinst.ini`, como no exemplo a seguir.

```
[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. Após instalar e configurar o driver ODBC do Amazon Athena, use a ferramenta de linha de comando `isql` do unixODBC para verificar a conexão, como no exemplo a seguir.

```
username % isql -v "athena_odbc_test"
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
SQL>
```

macOS

Se quiser usar um computador cliente com macOS para acessar o Amazon Athena, é necessário ter o driver ODBC do Amazon Athena.

Requisitos do sistema macOS

Cada computador macOS no qual você instale o driver deve atender aos seguintes requisitos.

- Usar o macOS versão 14 ou superior.

- Ter 100 MB de espaço em disco disponível.
- Usar a versão 3.52.16 ou superior do iODBC.

Como instalar o conector de dados ODBC no macOS

Siga o procedimento abaixo para baixar e instalar o driver ODBC do Amazon Athena para sistemas operacionais macOS.

Para baixar e instalar o driver ODBC do Amazon Athena para macOS

1. Baixe o arquivo de pacote `.pkg`.
2. Clique duas vezes no arquivo `.pkg`.
3. Siga as etapas no assistente para instalar o driver.
4. Na página Contrato de licença, pressione Continuar e escolha Concordar.
5. Escolha Instalar.
6. Quando a instalação terminar, escolha Concluir.
7. Insira o seguinte comando para verificar se o driver está instalado:

   ```
   > pkgutil --pkgs | grep athenaodbc
   ```

   Dependendo do seu sistema, a saída poderá ser semelhante à uma das seguintes opções.

   ```
   com.amazon.athenaodbc-x86_64.Config
   com.amazon.athenaodbc-x86_64.Driver
   ```

   ou

   ```
   com.amazon.athenaodbc-arm64.Config
   com.amazon.athenaodbc-arm64.Driver
   ```

Configurar um nome de fonte de dados no macOS

Após a instalação do driver, você poderá encontrar exemplos de arquivos `.odbc.ini` e `.odbcinst.ini` nos seguintes locais:

- Computadores com processador Intel: `/opt/athena/odbc/x86_64/ini/`

- Computadores com processador ARM: `/opt/athena/odbc/arm64/ini/`

Use os arquivos `.ini` nesse local como exemplos para configurar o driver ODBC e o nome da fonte de dados (DSN) do Amazon Athena.

> **Note**
>
> Por padrão, os gerenciadores de driver ODBC usam os versões ocultas dos arquivos de configuração `.odbc.ini` e `.odbcinst.ini`, localizadas no diretório inicial.

Para especificar o caminho dos arquivos `.odbc.ini` e `.odbcinst.ini` usando o gerenciador de drivers iODBC, execute as etapas a seguir.

Para especificar localizações de arquivos **`.ini`** do ODBC usando o gerenciador de drivers iODBC

1. Defina `ODBCINI` para o caminho completo e o nome de arquivo do arquivo `odbc.ini`.
   - Para computadores macOS com processadores Intel, aplique a sintaxe a seguir.

     ```
     export ODBCINI=/opt/athena/odbc/x86_64/ini/odbc.ini
     ```
   - Para computadores macOS com processadores ARM, aplique a sintaxe a seguir.

     ```
     export ODBCINI=/opt/athena/odbc/arm64/ini/odbc.ini
     ```
2. Defina `ODBCSYSINI` para o caminho completo e o nome de arquivo do arquivo `odbcinst.ini`.
   - Para computadores macOS com processadores Intel, aplique a sintaxe a seguir.

     ```
     export ODBCSYSINI=/opt/athena/odbc/x86_64/ini/odbcinst.ini
     ```
   - Para computadores macOS com processadores ARM, aplique a sintaxe a seguir.

     ```
     export ODBCSYSINI=/opt/athena/odbc/arm64/ini/odbcinst.ini
     ```
3. Se quiser usar um nome da fonte de dados (DSN) para estabelecer conexão com seu armazenamento de dados, configure o arquivo `odbc.ini` para definir nomes de fonte de dados (DSNs). Defina as propriedades no arquivo `odbc.ini` para criar um DSN que especifique as informações de conexão para seu armazenamento de dados, como no exemplo a seguir.

```
[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. Configure o arquivo `odbcinst.ini`, como no exemplo a seguir.

```
[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. Após instalar e configurar o driver ODBC do Amazon Athena, use a ferramenta de linha de comando `iodbctest` para verificar a conexão, como no exemplo a seguir.

```
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>
```

Parâmetros de conexão do ODBC 2.x do Athena

As opções da caixa de diálogo de Configuração do ODBC do Amazon Athena incluem Opções de autenticação, Opções avançadas, Opções de registro em log, Substituições de endpoints e Opções de proxy. Para obter informações detalhadas sobre cada um deles, acesse os links correspondentes.

- Principais parâmetros de conexão do ODBC 2.x
- Opções de autenticação
- Opções avançadas
- Opções de registro em log
- Substituições de endpoints
- Opções de proxy

Principais parâmetros de conexão do ODBC 2.x

As seções a seguir descrevem cada um dos principais parâmetros de conexão.

Nome da fonte de dados

Especifica o nome de sua fonte de dados.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
DSN	Opcional para tipos de conexão sem DSN	none	`DSN=AmazonAthenaOdbcUsWest1;`

Descrição

Contém a descrição de sua fonte de dados.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
Descrição	Opcional	none	`Description=Connection to Amazon Athena us-west-1;`

Catálogo

Especifica o nome do catálogo de dados. Para obter mais informações, consulte DataCatalog na Amazon Athena API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
Catálogo	Opcional	`AwsDataCatalog`	`Catalog=AwsDataCatalog;`

Região

Especifica o Região da AWS. Para obter informações sobre Regiões da AWS, consulte Regiões e zonas de disponibilidade.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
AwsRegion	Obrigatório	none	AwsRegion =us-west-1;

Banco de dados

Especifica o nome do banco de dados. Para obter mais informações, consulte Database na Amazon Athena API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
Schema	Opcional	default	Schema=default;

WorkGroup

Especifica o nome do grupo de trabalho. Para obter mais informações sobre grupos de trabalho, consulte WorkGroup na Amazon Athena API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
WorkGroup	Opcional	primary	Workgroup =primary;

Local de saída

Especifica o local do Amazon S3 em que os resultados da consulta são armazenados. Para obter mais informações sobre o local de saída, consulte ResultConfiguration na Amazon Athena API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
S3OutputLocation	Obrigatório	none	`S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/;`

Opções de criptografia

Nome do parâmetro da caixa de diálogo: opções de criptografia

Especifica a opção de criptografia. Para obter mais informações sobre as opções de criptografia, consulte EncryptionConfiguration na Amazon Athena API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Possíveis valores	Exemplo de string de conexão
S3OutputEncOption	Opcional	none	`NOT_SET`, `SSE_S3`, `SSE_KMS`, `CSE_KMS`	`S3OutputEncOption=SSE_S3;`

Chave KMS

Especifica uma chave do KMS para criptografia. Para obter mais informações sobre configuração de criptografia para chaves do KMS, consulte EncryptionConfiguration na Amazon Athena API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
S3OutputEncKMSKey	Opcional	none	`S3OutputEncKMSKey=your_key;`

Teste de conexão

O administrador de fonte de dados ODBC oferece a opção Teste, que você pode usar para testar sua conexão ODBC 2.x com o Amazon Athena. Para obter as etapas, consulte Configurar um nome de fonte de dados no Windows. Quando você testa uma conexão, o driver ODBC chama a ação da API GetWorkGroup do Athena. A chamada usa o tipo de autenticação e o provedor de credenciais correspondente que você especificou para recuperar credenciais. Não há cobrança pelo teste de conexão ao usar o driver ODBC 2.x. O teste não gera resultados de consulta no bucket do Amazon S3.

Opções de autenticação

É possível se conectar ao Amazon Athena usando os tipos de autenticação a seguir. Para todos os tipos, o nome da string de conexão é `AuthenticationType`, o tipo de parâmetro é `Required` e o valor padrão é `IAM Credentials`. Para obter informações sobre os parâmetros de cada tipo de autenticação, acesse o link correspondente. Para obter parâmetros de autenticação comuns, consulte Parâmetros de autenticação comuns.

Tipo de autenticação	Exemplo de string de conexão
Credenciais do IAM	`AuthenticationType=IAM Credentials;`
Perfil do IAM	`AuthenticationType=IAM Profile;`
AD FS	`AuthenticationType=ADFS;`
Azure AD	`AuthenticationType=AzureAD;`
Azure AD do navegador	`AuthenticationType=BrowserAzureAD;`
SAML do navegador	`AuthenticationType=BrowserSAML;`
Browser SSO OIDC	`AuthenticationType=BrowserSSOOIDC;`
Credenciais padrão	`AuthenticationType=Default Credentials;`
Credenciais externas	`AuthenticationType=External Credentials;`

Tipo de autenticação	Exemplo de string de conexão
Perfil de instância	`AuthenticationType=Instance Profile;`
JWT	`AuthenticationType=JWT;`
Okta	`AuthenticationType=Okta;`
Ping	`AuthenticationType=Ping;`

Credenciais do IAM

Você pode usar suas credenciais do IAM para se conectar ao Amazon Athena com o driver ODBC usando os parâmetros da string de conexão descritos nesta seção.

Tipo de autenticação

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
AuthenticationType	Obrigatório	`IAM Credentials`	`AuthenticationType=IAM Credentials;`

ID de usuário

Seu ID de chave de acesso da AWS. Para obter mais informações sobre chaves de acesso, consulte Credenciais de segurança da AWS no Guia do usuário do IAM.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
UID	Obrigatório	none	`UID=AKIAIOSFODNN7EXAMPLE;`

Senha

O ID de sua chave secreta da AWS. Para obter mais informações sobre chaves de acesso, consulte Credenciais de segurança da AWS no Guia do usuário do IAM.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
PWD	Obrigatório	none	`PWD=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKE;`

Session token (Token da sessão)

Caso esteja usando credenciais temporárias da AWS, você deve especificar o token da sessão. Para obter informações sobre credenciais temporárias, consulte Credenciais de segurança temporárias no IAM, no Guia do usuário do IAM.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
SessionToken	Opcional	none	`SessionToken=AQoDYXdzEJr...<remainder of session token>;`

Perfil do IAM

É possível configurar um perfil nomeado para se conectar ao Amazon Athena usando o driver ODBC. Para usar as credenciais disponíveis em seu perfil de instância de hospedagem do Amazon EC2, defina o parâmetro `credential_source` como `Ec2InstanceMetadata`. Para usar um provedor de credenciais personalizado em um perfil nomeado, especifique um valor para o parâmetro `plugin_name` na configuração do perfil.

Tipo de autenticação

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
AuthenticationType	Obrigatório	`IAM Credentials`	`AuthenticationType=IAM Profile;`

Perfil do AWS

O nome do perfil a ser usado com a conexão ODBC. Para obter mais informações sobre perfis, consulte Usar perfis nomeados no Guia do usuário da AWS Command Line Interface.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
Perfil da AWS	Obrigatório	none	`AWSProfile=default;`

Perfil preferencial

O nome do recurso da Amazon (ARN) da função a ser assumida. Utiliza-se o parâmetro de perfil preferencial quando o provedor de credenciais personalizadas é especificado pelo parâmetro `plugin_name` na configuração do perfil. Para obter mais informações sobre perfis de ARN, consulte AssumeRole na AWS Security Token Service API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
preferred_role	Opcional	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Duração da sessão

A duração, em segundos, da sessão do perfil. Para obter mais informações sobre duração da sessão, consulte AssumeRole na AWS Security Token Service API Reference. Utiliza-se o parâmetro de duração da sessão quando o provedor de credenciais personalizadas é especificado pelo parâmetro `plugin_name` na configuração do perfil.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
duration	Opcional	`900`	`duration=900;`

Nome do plug-in

Especifica o nome de um provedor de credenciais personalizadas utilizado em um perfil nomeado. Esse parâmetro pode assumir os mesmos valores do campo Tipo de autenticação do administrador de fonte de dados ODBC, mas é usado somente pela configuração do `AWSProfile`.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
plugin_name	Opcional	`none`	`plugin_name=AzureAD;`

AD FS

O AD FS é um plug-in de autenticação baseado em SAML que funciona com o provedor de identidades do Active Directory Federation Service (AD FS). O plug-in é compatível com a autenticação integrada do Windows e a autenticação baseada em formulários. Ao usar a autenticação integrada do Windows, você pode omitir o nome de usuário e a senha. Para obter informações sobre como configurar o AD FS e o Athena, consulte Configuração do acesso federado ao Amazon Athena para usuários do Microsoft AD FS usando um cliente ODBC.

Tipo de autenticação

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
AuthenticationType	Obrigatório	`IAM Credentials`	`AuthenticationType=ADFS;`

ID de usuário

Seu nome de usuário para se conectar ao servidor do AD FS. Para a autenticação integrada do Windows, você pode omitir o nome de usuário. Se a configuração do AD FS exigir um nome de usuário, será necessário fornecê-lo no parâmetro de conexão.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
UID	Opcional para a autenticação integrada do Windows	`none`	`UID=domain\username;`

Senha

Sua senha para se conectar ao servidor do AD FS. Assim como o campo do nome de usuário, você poderá omitir o nome de usuário ao usar a autenticação integrada do Windows. Se a configuração do AD FS exigir uma senha, será necessário fornecê-la no parâmetro de conexão.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
PWD	Opcional para a autenticação integrada do Windows	`none`	`PWD=password_3EXAMPLE;`

Perfil preferencial

O nome do recurso da Amazon (ARN) da função a ser assumida. Se sua declaração SAML tiver vários perfis, você poderá especificar esse parâmetro para escolher o perfil a ser assumido. Esse perfil deve estar presente na declaração SAML. Para obter mais informações sobre perfis de ARN, consulte AssumeRole na AWS Security Token Service API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
preferred_role	Opcional	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Duração da sessão

A duração, em segundos, da sessão do perfil. Para obter mais informações sobre duração da sessão, consulte AssumeRole na AWS Security Token Service API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
duration	Opcional	`900`	`duration=900;`

Host de IdP

O nome de host do serviço AD FS.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
idp_host	Require	none	`idp_host=<server-name>.<company.com>;`

Porta de IdP

A porta a ser usada para se conectar ao host do AD FS.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
idp_port	Obrigatório	none	`idp_port=443;`

LoginToRP

A parte confiável. Use esse parâmetro para substituir o URL do endpoint da parte confiável do AD FS.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
LoginToRP	Opcional	`urn:amazon:webservices`	`LoginToRP=trustedparty;`

Azure AD

O Azure AD é um plug-in de autenticação baseado em SAML que funciona com o provedor de identidades do Azure AD. Este plugin não é compatível com autenticação multifator (MFA). Caso precise de suporte para MFA, considere usar o plug-in `BrowserAzureAD` em vez disso.

Tipo de autenticação

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
AuthenticationType	Obrigatório	`IAM Credentials`	`AuthenticationType=AzureAD;`

Perfil preferencial

O nome do recurso da Amazon (ARN) da função a ser assumida. Para obter informações sobre perfis de ARN, consulte AssumeRole na AWS Security Token Service API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
preferred_role	Opcional	none	`preferred_role=arn:aws:iam::123456789012:id/user1;`

Duração da sessão

A duração, em segundos, da sessão do perfil. Para obter mais informações, consulte AssumeRole na AWS Security Token Service API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
duration	Opcional	`900`	`duration=900;`

ID do locatário

Especifica o ID do locatário da aplicação.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
idp_tenant	Obrigatório	none	`idp_tenant=123zz112z-z12d-1z1f-11zz-f111aa111234;`

ID de cliente

Especifica o ID do cliente da aplicação.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
client_id	Obrigatório	none	`client_id=9178ac27-a1bc-1a2b-1a2b-a123abcd1234;`

Segredo do cliente

Especifica o segredo do cliente.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
client_secret	Obrigatório	none	`client_secret=zG12q~.xzG1xxxZ1wX1.~ZzXXX1XxkHZizeT1zzZ;`

Azure AD do navegador

O Browser Azure AD é um plug-in de autenticação baseado em SAML que funciona com o provedor de identidades do Azure AD e é compatível com autenticação multifator. Ao contrário do plug-in padrão do Azure AD, esse plug-in não requer nome de usuário, senha ou segredo do cliente nos parâmetros de conexão.

Tipo de autenticação

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
AuthenticationType	Obrigatório	`IAM Credentials`	`AuthenticationType=BrowserAzureAD;`

Perfil preferencial

O nome do recurso da Amazon (ARN) da função a ser assumida. Se sua declaração SAML tiver vários perfis, você poderá especificar esse parâmetro para escolher o perfil a ser assumido. O perfil especificado deve estar presente na declaração SAML. Para obter mais informações sobre perfis de ARN, consulte AssumeRole na AWS Security Token Service API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
preferred_role	Opcional	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Duração da sessão

A duração, em segundos, da sessão do perfil. Para obter mais informações sobre duração da sessão, consulte AssumeRole na AWS Security Token Service API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
duration	Opcional	`900`	`duration=900;`

ID do locatário

Especifica o ID do locatário da aplicação.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
idp_tenant	Obrigatório	none	`idp_tenant=123zz112z-z12d-1z1f-11zz-f111aa111234;`

ID de cliente

Especifica o ID do cliente da aplicação.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
client_id	Obrigatório	none	`client_id=9178ac27-a1bc-1a2b-1a2b-a123abcd1234;`

Timeout (Tempo limite)

A duração, em segundos, até que o plug-in pare de esperar pela resposta SAML do Azure AD.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
timeout	Opcional	`120`	`timeout=90;`

Habilitar cache de arquivos do Azure

Habilita um cache de credenciais temporárias. Esse parâmetro de conexão permite que as credenciais temporárias sejam armazenadas em cache e reutilizadas entre vários processos. Use essa opção para reduzir o número de janelas do navegador abertas ao usar ferramentas de BI, como o Microsoft Power BI.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
browser_azure_cache	Opcional	1	`browser_azure_cache=0;`

SAML do navegador

O Browser SAML é um plug-in de autenticação genérico que pode funcionar com provedores de identidade baseados em SAML e é compatível com autenticação multifator. Para obter informações

detalhadas sobre a configuração, consulte Configurar o Single Sign-On usando ODBC, SAML 2.0 e o provedor de identidade Okta.

Tipo de autenticação

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
AuthenticationType	Obrigatório	`IAM Credentials`	`AuthenticationType=BrowserSAML;`

Perfil preferencial

O nome do recurso da Amazon (ARN) da função a ser assumida. Se sua declaração SAML tiver vários perfis, você poderá especificar esse parâmetro para escolher o perfil a ser assumido. Esse perfil deve estar presente na declaração SAML. Para obter mais informações sobre perfis de ARN, consulte AssumeRole na AWS Security Token Service API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
preferred_role	Opcional	`none`	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Duração da sessão

A duração, em segundos, da sessão do perfil. Para obter mais informações, consulte AssumeRole na AWS Security Token Service API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
duration	Opcional	`900`	`duration=900;`

URL de login

O URL de autenticação única exibido para sua aplicação.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
login_url	Obrigatório	none	`login_url=https://trial-1234567.okta.com/app/trial-1234567_oktabrowsersaml_1/zzz4izzzAzDFBzZz1234/sso/saml;`

Escutar porta

O número da porta que é usada para receber a resposta do SAML. Esse valor deve corresponder ao URL do Centro de Identidade do IAM com o qual você configurou o IdP (por exemplo, `http://localhost:7890/athena`).

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
listen_port	Opcional	`7890`	`listen_port=7890;`

Timeout (Tempo limite)

A duração, em segundos, até que o plug-in pare de esperar pela resposta SAML do provedor de identidades.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
timeout	Opcional	`120`	`timeout=90;`

Browser SSO OIDC

O Browser SSO OIDC é um plug-in de autenticação que funciona com o AWS IAM Identity Center. Para obter informações sobre como habilitar e usar o Centro de Identidade do IAM, consulte Step 1: Enable IAM Identity Center no Guia do usuário do AWS IAM Identity Center.

Tipo de autenticação

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
AuthenticationType	Obrigatório	`IAM Credentials`	`AuthenticationType=BrowserSSOOIDC;`

URL inicial do Centro de Identidade do IAM

O URL do portal de acesso da AWS. A ação da API StartDeviceAuthorization do Centro de Identidade do IAM usa esse valor para o parâmetro `startUrl`.

Para copiar o URL do portal de acesso da AWS

1. Faça login no AWS Management Console e abra o console do AWS IAM Identity Center em https://console.aws.amazon.com/singlesignon/.
2. No painel de navegação, selecione Configurações.
3. Na página Configurações, em Origem de identidade, escolha o ícone da área de transferência para o URL do portal de acesso da AWS.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
sso_oidc_start_url	Obrigatório	none	`sso_oidc_start_url=https://app_id.awsapps.com/start;`

Região do Centro de Identidade do IAM

A Região da AWS em que o SSO está configurado. Os clientes `SSOOIDCClient` e `SSOClient` do AWS SDK usam esse valor para o parâmetro `region`.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
sso_oidc_region	Obrigatório	none	`sso_oidc_region=us-east-1;`

Escopos

A lista de escopos que são definidos pelo cliente. Após a autorização, a lista restringe as permissões quando um token de acesso é concedido. A ação da API RegisterClient do Centro de Identidade do IAM usa esse valor para o parâmetro `scopes`.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
sso_oidc_scopes	Opcional	none	`sso_oidc_scopes=scope1,scope2,scope3;`

ID da conta

O identificador da Conta da AWS que é atribuída ao usuário. A API GetRoleCredentials do Centro de Identidade do IAM usa esse valor para o parâmetro `accountId`.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
sso_oidc_account_id	Obrigatório	none	`sso_oidc_account_id=123456789123;`

Nome do perfil

O nome amigável do perfil atribuído ao usuário. O nome que você especifica para esse conjunto de permissões é exibido no portal de acesso da AWS como um perfil disponível. A ação da API GetRoleCredentials do Centro de Identidade do IAM usa esse valor para o parâmetro `roleName`.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
sso_oidc_role_name	Obrigatório	none	`sso_oidc_role_name=AthenaReadAccess;`

Timeout (Tempo limite)

O número de segundos em que a API de SSO de sondagem deve verificar o token de acesso.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
sso_oidc_timeout	Opcional	`120`	`sso_oidc_timeout=60;`

Habilitar cache de arquivos

Habilita um cache de credenciais temporárias. Esse parâmetro de conexão permite que as credenciais temporárias sejam armazenadas em cache e reutilizadas entre vários processos. Use

essa opção para reduzir o número de janelas do navegador abertas ao usar ferramentas de BI, como o Microsoft Power BI.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
sso_oidc_cache	Opcional	1	`sso_oidc_cache=0;`

Credenciais padrão

É possível usar as credenciais padrão que você configura em seu sistema cliente para se conectar ao Amazon Athena. Para obter informações sobre o uso de credenciais padrão, consulte Usar a cadeia de fornecedores de credenciais padrão no Guia do desenvolvedor do AWS SDK for Java.

Tipo de autenticação

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
AuthenticationType	Obrigatório	`IAM Credentials`	`AuthenticationType=Default Credentials;`

Credenciais externas

As credenciais externas são um plug-in de autenticação genérico que pode ser usado para se conectar a qualquer provedor de identidades externo baseado em SAML. Para usar o plug-in, passe um arquivo executável que retorna uma resposta SAML.

Tipo de autenticação

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
AuthenticationType	Obrigatório	`IAM Credentials`	`AuthenticationType=External Credentials;`

Caminho executável

O caminho para o executável que tem a lógica de seu provedor de credenciais personalizado baseado em SAML. A saída do executável deverá ser a resposta SAML analisada do provedor de identidades.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
ExecutablePath	Obrigatório	`none`	ExecutablePath=C:\Users*user_name* *external_credential.exe*

Lista de argumentos

A lista de argumentos que você quer passar para o executável.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
ArgumentList	Opcional	`none`	`ArgumentList=` *arg1 arg2 arg3*

Perfil de instância

Esse tipo de autenticação é usado em instâncias do EC2 e é fornecido pelo serviço de metadados do Amazon EC2.

Tipo de autenticação

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
AuthenticationType	Obrigatório	`IAM Credentials`	`AuthenticationType=Instance Profile;`

JWT

O plug-in JWT (JSON Web Token) fornece uma interface que usa JSON Web Tokens para assumir um perfil do Amazon IAM. A configuração depende do provedor de identidades. Para obter informações sobre como configurar a federação para o Google Cloud e a AWS, consulte Configurar a federação de identidade da carga de trabalho com a AWS ou o Azure na documentação do Google Cloud.

Tipo de autenticação

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
AuthenticationType	Obrigatório	`IAM Credentials`	`AuthenticationType=JWT;`

Perfil preferencial

O nome do recurso da Amazon (ARN) da função a ser assumida. Para obter mais informações sobre perfis de ARN, consulte AssumeRole na AWS Security Token Service API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
preferred_role	Opcional	`none`	`preferred_role=arn:aws:IAM:`

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
			`:12345678 9012:id/user1;`

Duração da sessão

A duração, em segundos, da sessão do perfil. Para obter mais informações sobre duração da sessão, consulte AssumeRole na AWS Security Token Service API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
duration	Opcional	`900`	`duration=900;`

JSON Web Token

O JSON Web Token que é usado para recuperar credenciais temporárias do IAM usando a ação da API AssumeRoleWithWebIdentity do AWS STS. Para obter informações sobre como gerar JSON Web Tokens para usuários do Google Cloud Platform (GCP), consulte Como usar tokens OAuth do JWT na documentação do Google Cloud.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
`web_identity_token`	Obrigatório	`none`	`web_identity_token=eyJhbGc. ..<remainder of token>;`

Nome da sessão da função

Um nome para a sessão. Uma técnica comum é usar o nome ou o identificador do usuário da aplicação como o nome de sessão do perfil. Isso associa, de maneira conveniente, as credenciais de segurança temporárias que sua aplicação usa ao usuário correspondente.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
role_session_name	Obrigatório	none	`role_session_name=familiarname;`

Okta

O Okta é um plug-in de autenticação baseado em SAML que funciona com o provedor de identidades do Okta. Para obter informações sobre como configurar a federação para o Okta e o Amazon Athena, consulte Configurar o SSO para ODBC usando o plugin Okta e o Okta Identity Provider.

Tipo de autenticação

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
AuthenticationType	Obrigatório	`IAM Credentials`	`AuthenticationType=Okta;`

ID de usuário

Seu nome de usuário do Okta.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
UID	Obrigatório	none	`UID=jane.doe@org.com;`

Senha

A senha de usuário do Okta.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
PWD	Obrigatório	none	`PWD=oktauserpasswordexample;`

Perfil preferencial

O nome do recurso da Amazon (ARN) da função a ser assumida. Para obter mais informações sobre perfis de ARN, consulte AssumeRole na AWS Security Token Service API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
preferred_role	Opcional	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

Duração da sessão

A duração, em segundos, da sessão do perfil. Para obter mais informações, consulte AssumeRole na AWS Security Token Service API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
duration	Opcional	`900`	`duration=900;`

Host de IdP

O URL de sua organização Okta. É possível extrair o parâmetro `idp_host` do URL do link de incorporação em sua aplicação Okta. Para obter as etapas, consulte Recuperar

informações de configuração de ODBC do Okta. O primeiro segmento depois de `https://`, até `okta.com` (inclusive) é seu host de IdP (por exemplo, `http://trial-1234567.okta.com`).

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
idp_host	Obrigatório	`None`	`idp_host=dev-99999999.okta.com;`

Porta de IdP

O número da porta a ser usada para se conectar ao host de IdP.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
idp_port	Obrigatório	`None`	`idp_port=443;`

ID da aplicação do Okta

O identificador de duas partes da aplicação. É possível extrair o parâmetro `app_id` do URL do link de incorporação em sua aplicação Okta. Para obter as etapas, consulte Recuperar informações de configuração de ODBC do Okta. O ID da aplicação são os dois últimos segmentos do URL, inclusive a barra no meio. Os segmentos são duas strings de 20 caracteres com uma mistura de números e letras maiúsculas e minúsculas (por exemplo, `Abc1de2fghi3J45kL678/abc1defghij2klmNo3p4`).

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
app_id	Obrigatório	`None`	`app_id=0oa25kx8ze9A3example/alnexamplea0piaWa0g7;`

Nome da aplicação Okta

O nome da aplicação do Okta.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
app_name	Obrigatório	None	app_name=amazon_aws_redshift;

Tempo de espera do Okta

Especifica a duração, em segundos, para aguardar o código de autenticação multifator (MFA).

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
okta_mfa_wait_time	Opcional	10	okta_mfa_wait_time=20;

Tipo de MFA do Okta

O tipo de fator MFA. Os tipos compatíveis são: Google Authenticator, SMS (Okta), Okta Verify com Push e Okta Verify com TOTP. As políticas de segurança de cada organização determinam se a MFA é necessária ou não para o login do usuário.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Possíveis valores	Exemplo de string de conexão
okta_mfa_type	Optional	None	googleauthenticator, smsauthentication,	okta_mfa_type=oktaverifywithpush;

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Possíveis valores	Exemplo de string de conexão
			`oktaverifywithpush, oktaverifywithtotp`	

Número de telefone do Okta

O número de telefone a ser usado com a autenticação do AWS SMS. Esse parâmetro é necessário somente para registro multifatorial. Se o seu número de celular já estiver registrado ou se a autenticação do AWS SMS não for usada pela política de segurança, você poderá ignorar esse campo.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
okta_mfa_phone_number	Necessário para registro no MFA, opcional em outros casos	`None`	`okta_mfa_phone_number=19991234567;`

Habilitar cache de arquivos do Okta

Habilita um cache de credenciais temporárias. Esse parâmetro de conexão permite que as credenciais temporárias sejam armazenadas em cache e reutilizadas entre vários processos abertos por aplicações de BI. Use essa opção para evitar o controle de utilização da API do Okta.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
okta_cache	Opcional	`0`	`okta_cache=1;`

Ping

O Ping é um plug-in baseado em SAML que funciona com o provedor de identidades PingFederate.

Tipo de autenticação

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
AuthenticationType	Obrigatório	`IAM Credentials`	`AuthenticationType=Ping;`

ID de usuário

O nome de usuário do servidor PingFederate.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
UID	Obrigatório	`none`	`UID=pingusername@domain.com;`

Senha

A senha do servidor PingFederate.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
PWD	Obrigatório	`none`	`PWD=pingpassword;`

Perfil preferencial

O nome do recurso da Amazon (ARN) da função a ser assumida. Se sua declaração SAML tiver vários perfis, você poderá especificar esse parâmetro para escolher o perfil a ser assumido. Esse perfil deve estar presente na declaração SAML. Para obter mais informações sobre perfis de ARN, consulte AssumeRole na AWS Security Token Service API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
preferred_role	Opcional	none	`preferred_role=arn:aws:iam::123456789012:id/user1;`

Duração da sessão

A duração, em segundos, da sessão do perfil. Para obter mais informações sobre duração da sessão, consulte AssumeRole na AWS Security Token Service API Reference.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
duration	Opcional	`900`	`duration=900;`

Host de IdP

O endereço do servidor Ping. Para encontrar seu endereço, acesse o URL a seguir e visualize o campo Endpoint da aplicação de SSO.

```
https://your-pf-host-#:9999/pingfederate/your-pf-app#/spConnections
```

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
idp_host	Obrigatório	none	`idp_host=ec2-1-83-65-12.compute-1.amazonaws.com;`

Porta de IdP

O número da porta a ser usada para se conectar ao host de IdP.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
idp_port	Obrigatório	None	`idp_port=443;`

SPID de parceiro

O endereço do provedor de serviços. Para encontrar o endereço do provedor de serviços, acesse o URL a seguir e visualize o campo Endpoint da aplicação de SSO.

```
https://your-pf-host-#:9999/pingfederate/your-pf-app#/spConnections
```

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
partner_spid	Obrigatório	None	`partner_spid=https://us-east-1.signin.aws.amazon.com/platform/saml/<...>;`

Parâmetro de URI do Ping

Passa um argumento de URI para uma solicitação de autenticação ao Ping. Use esse parâmetro para ignorar a limitação de perfil único do Lake Formation. Configure o Ping para reconhecer o parâmetro passado e verificar se o perfil passado existe na lista de perfis atribuídos ao usuário. Em seguida, envie um único perfil na declaração SAML.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
ping_uri_param	Opcional	None	`ping_uri_param=role=my_iam_role;`

Parâmetros de autenticação comuns

Os parâmetros desta seção são comuns aos tipos de autenticação, conforme observado.

Usar o proxy para IdP

Permite a comunicação entre o driver e o IdP pelo proxy. Essa opção está disponível para os seguintes plug-ins de autenticação:

- AD FS
- Azure AD
- Azure AD do navegador
- Browser SSO OIDC
- JWT
- Okta
- Ping

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
UseProxyForIdP	Opcional	`0`	`UseProxyForIdP=1;`

Usar o Lake Formation

Usa a ação da API AssumeDecoratedRoleWithSAML do Lake Formation para recuperar credenciais temporárias do IAM em vez da ação da API AssumeRoleWithSAML do AWS STS. Essa opção está disponível para os plug-ins de autenticação Azure AD, Browser Azure AD, Browser SAML, Okta, Ping e AD FS.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
LakeformationEnabled	Opcional	`0`	`LakeformationEnabled=1;`

SSL inseguro (IdP)

Desabilita o SSL ao se comunicar com o IdP. Essa opção está disponível para os plug-ins de autenticação Azure AD, Browser Azure AD, Okta, Ping e AD FS.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
SSL_Insecure	Opcional	`0`	`SSL_Insecure=1;`

Substituições de endpoints

Substituição do endpoint do Athena

A classe `endpointOverride ClientConfiguration` usa esse valor para substituir o endpoint HTTP padrão para o cliente Amazon Athena. Para obter mais informações, consulte AWS Client configuration no Guia do desenvolvedor do AWS SDK for C++.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
EndpointOverride	Opcional	`none`	`EndpointOverride=athena.us-west-2.amazonaws.com;`

Substituição do endpoint de transmissão do Athena

O método `ClientConfiguration.endpointOverride` usa esse valor para substituir o endpoint HTTP padrão para o cliente Amazon Athena. Para obter mais informações, consulte AWS Client configuration no Guia do desenvolvedor do AWS SDK for C++. O serviço Athena Streaming está disponível pela porta 444.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
Streaming EndpointOverride	Opcional	`none`	`StreamingEndpointOverride=athena.us-west-1.amazonaws.com:444;`

Substituição de endpoint do AWS STS

O método `ClientConfiguration.endpointOverride` usa esse valor para substituir o endpoint HTTP padrão para o cliente AWS STS. Para obter mais informações, consulte AWS Client configuration no Guia do desenvolvedor do AWS SDK for C++.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
StsEndpointOverride	Opcional	`none`	`StsEndpointOverride=sts.us-west-1.amazonaws.com;`

Substituição de endpoint do Lake Formation

O método `ClientConfiguration.endpointOverride` usa esse valor para substituir o endpoint HTTP padrão para o cliente Lake Formation. Para obter mais informações, consulte AWS Client configuration no Guia do desenvolvedor do AWS SDK for C++.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
LakeFormationEndpointOverride	Opcional	`none`	`LakeFormationEndpointOverride=lakeformation.us-west-1.amazonaws.com;`

Substituição de endpoint de SSO

O método `ClientConfiguration.endpointOverride` usa esse valor para substituir o endpoint HTTP padrão para o cliente de SSO. Para obter mais informações, consulte AWS Client configuration no Guia do desenvolvedor do AWS SDK for C++.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
SSOEndpointOverride	Opcional	`none`	`SSOEndpointOverride=portal.sso.us-east-2.amazonaws.com;`

Substituição de endpoint OIDC de SSO

O método `ClientConfiguration.endpointOverride` usa esse valor para substituir o endpoint HTTP padrão para o cliente OIDC de SSO. Para obter mais informações, consulte AWS Client configuration no Guia do desenvolvedor do AWS SDK for C++.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
SSOOIDCEndpointOverride	Opcional	`none`	`SSOOIDCEndpointOverride=oidc.us-east-2.amazonaws.com`

Opções avançadas

Tamanho da busca

O número máximo de resultados (linhas) a serem retornados nesta solicitação. Para obter informações sobre parâmetros, consulte GetQuery MaxResults. Para a API de transmissão, o valor máximo é 10000000.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
RowsToFetchPerBlock	Opcional	`1000` que não seja transmissão `20000` para transmissão	`RowsToFetchPerBlock=20000;`

Habilitar a reutilização de resultados

Especifica se os resultados da consulta anterior poderão ser reutilizados quando a consulta for executada. Para obter informações sobre parâmetros, consulte ResultReuseByAgeConfiguration.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
EnableResultReuse	Opcional	`0`	`EnableResultReuse=1;`

Idade máxima da reutilização de resultados

Especifica, em minutos, a idade máxima de um resultado de consulta anterior que o Athena deverá considerar para reutilização. Para obter informações sobre parâmetros, consulte ResultReuseByAgeConfiguration.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
ReusedResultMaxAgeInMinutes	Opcional	`60`	`ReusedResultMaxAgeInMinutes=90;`

Habilitar a API de transmissão

Escolhe se deseja usar a API de transmissão do Athena para buscar o conjunto de resultados.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
UseResultsetStreaming	Opcional	`0`	`UseResultsetStreaming=1;`

Habilitar o captador do S3

Busca o conjunto de resultados gerado pelo Athena com base no bucket do Amazon S3 interagindo diretamente com o Amazon S3.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
EnableS3Fetcher	Opcional	1	`EnableS3Fetcher=1;`

Usar vários threads do S3

Busca dados do Amazon S3 usando vários threads. Quando essa opção está habilitada, o arquivo de resultados armazenado no bucket do Amazon S3 é buscado em paralelo usando vários threads.

Habilite essa opção somente se você tiver uma boa largura de banda da rede. Por exemplo, em nossas medições em uma instância do EC2 c5.2xlarge, um cliente S3 de thread único atingiu 1 Gbps, enquanto clientes S3 de vários segmentos atingiram 4 Gbps de throughput de rede.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
UseMultipleS3Threads	Opcional	`0`	`UseMultipleS3Threads=1;`

Usar um único catálogo e esquema

Por padrão, o driver ODBC consulta o Athena para obter a lista de catálogos e esquemas disponíveis. Essa opção força o driver a usar o catálogo e o esquema especificados pela caixa de diálogo de configuração Administrador de fonte de dados ODBC ou pelos parâmetros de conexão.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
UseSingleCatalogAndSchema	Opcional	`0`	`UseSingleCatalogAndSchema=1;`

Usar consulta para listar tabelas

Para catálogos do tipo `LAMBDA`, permite que o driver ODBC envie uma consulta SHOW TABLES para obter uma lista das tabelas disponíveis. Essa é a configuração padrão. Se esse parâmetro for definido como 0, o driver ODBC usará a API ListTableMetadata do Athena para obter uma lista das tabelas disponíveis. Observe que, para catálogos do tipo `LAMBDA`, o uso de `ListTableMetadata` resulta em regressão de desempenho.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
UseQueryToListTables	Opcional	`1`	`UseQueryToListTables=1;`

Usar WCHAR para tipos de strings

Por padrão, o driver ODBC usa `SQL_CHAR` e `SQL_VARCHAR` para os tipos de dados de string `char`, `varchar`, `string`, `array`, `map<>`, `struct<>` e `row` do Athena. Definir esse parâmetro como 1 força o driver a usar `SQL_WCHAR` e `SQL_WVARCHAR` para tipos de dados de string. Os tipos caractere largo e variável larga são usados para garantir que seja possível armazenar e recuperar corretamente caracteres de diferentes linguagens.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
UseWCharForStringTypes	Opcional	`0`	`UseWCharForStringTypes=1;`

Consultar catálogos externos

Especifica se o driver precisa consultar catálogos externos do Athena. Para obter mais informações, consulte Migrar para o driver ODBC 2.x.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
QueryExternalCatalogs	Opcional	`0`	`QueryExternalCatalogs=1;`

Verificar SSL

Controla se os certificados SSL deverão ser verificados ao usar o AWS SDK. Esse valor é transmitido ao parâmetro `ClientConfiguration.verifySSL`. Para obter mais informações, consulte AWS Client configuration no Guia do desenvolvedor do AWS SDK for C++.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
VerifySSL	Opcional	`1`	`VerifySSL=0;`

Tamanho do bloco de resultados do S3

Especifica, em bytes, o tamanho do bloco a ser baixado para uma única solicitação da API GetObject do Amazon S3. O valor padrão é 67108864 (64 MB). Os valores mínimo e máximo permitidos são 10485760 (10 MB) e 2146435072 (cerca de 2 GB).

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
S3ResultBlockSize	Opcional	`67108864`	`S3ResultBlockSize=268435456;`

Comprimento da coluna de string

Especifica o comprimento da coluna para colunas com um tipo de dados `string`. Como o Athena usa o tipo de dados de string do Apache Hive, que não tem precisão definida, o tamanho padrão relatado pelo Athena é 2147483647 (`INT_MAX`). Como as ferramentas de BI geralmente pré-alocam memória para as colunas, isso pode resultar em alto consumo de memória. Para evitar isso, o

driver ODBC do Athena limita a precisão relatada para colunas do tipo de dados `string` e expõe o parâmetro de conexão `StringColumnLength` para que seja possível alterar o valor padrão.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
StringColumnLength	Opcional	255	`StringColumnLength =65535;`

Comprimento de coluna de tipo complexo

Especifica o comprimento da coluna para colunas com tipos de dados complexos, como `map`, `struct` e `array`. Assim como StringColumnLength, o Athena relata 0 de precisão para colunas com tipos de dados complexos. O driver ODBC do Athena define a precisão padrão para colunas com tipos de dados complexos e expõe o parâmetro de conexão `ComplexTypeColumnLength` para que seja possível alterar o valor padrão.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
ComplexTypeColumnLength	Opcional	65535	`ComplexTypeColumnLength=123456;`

Certificado CA confiável

Instrui o cliente HTTP sobre onde encontrar o armazenamento confiável de certificados SSL. Esse valor é transmitido ao parâmetro `ClientConfiguration.caFile`. Para obter mais informações, consulte AWS Client configuration no Guia do desenvolvedor do AWS SDK for C++.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
TrustedCerts	Opcional	`%INSTALL_PATH%/bin`	`TrustedCerts=C:\\Program Files\\Amazon Athena ODBC Driver\\bin\\cacert.pem;`

Período mínimo de sondagem

Especifica o valor mínimo, em milissegundos, a ser aguardado antes de sondar o Athena sobre o status de execução da consulta.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
MinQueryExecutionPollingInterval	Opcional	`100`	`MinQueryExecutionPollingInterval=200;`

Período máximo de sondagem

Especifica o valor máximo, em milissegundos, a ser aguardado antes de sondar o Athena sobre o status de execução da consulta.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
MaxQueryExecutionPollingInterval	Opcional	`60000`	`MaxQueryExecutionPollingInterval=1000;`

Multiplicador de sondagens

Especifica o fator para aumentar o período da sondagem. Por padrão, a sondagem começa com o valor do período mínimo de sondagem e dobra com cada sondagem até atingir o valor do período máximo de sondagem.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
QueryExecutionPollingIntervalMultiplier	Opcional	2	`QueryExecutionPollingIntervalMultiplier=2;`

Duração máxima da sondagem

Especifica o valor máximo, em milissegundos, a ser aguardado para que o driver possa sondar o Athena sobre o status de execução da consulta.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
MaxPollDuration	Opcional	`1800000`	`MaxPollDuration=1800000;`

Tempo limite da conexão

O tempo (em milissegundos) que a conexão HTTP aguardará para estabelecer uma conexão. Esse valor é definido para o cliente Athena `ClientConfiguration.connectTimeoutMs`. Se não for especificado, o valor padrão de curl será usado. Para obter informações sobre parâmetros de conexão, consulte Client Configuration no Guia do desenvolvedor do AWS SDK for Java.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
ConnectionTimeout	Opcional	`0`	`ConnectionTimeout=2000;`

Tempo limite da solicitação

Especifica o tempo limite de leitura do soquete para clientes HTTP. Esse valor é definido para o parâmetro `ClientConfiguration.requestTimeoutMs` do cliente Athena. Para obter informações sobre parâmetros, consulte Client Configuration no Guia do desenvolvedor do AWS SDK for Java.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
RequestTimeout	Opcional	`10000`	`RequestTimeout=30000;`

Opções de proxy

Host do proxy

Se você exigir que os usuários passem por um proxy, use este parâmetro para definir o host do proxy. Esse parâmetro corresponde ao parâmetro `ClientConfiguration.proxyHost` no AWS SDK. Para obter mais informações, consulte AWS Client configuration no Guia do desenvolvedor do AWS SDK for C++.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
ProxyHost	Opcional	none	`ProxyHost =127.0.0.1;`

Porta do proxy

Use este parâmetro para definir a porta do proxy. Esse parâmetro corresponde ao parâmetro `ClientConfiguration.proxyPort` no AWS SDK. Para obter mais informações, consulte AWS Client configuration no Guia do desenvolvedor do AWS SDK for C++.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
ProxyPort	Opcional	none	`ProxyPort=8888;`

Nome de usuário do proxy

Use este parâmetro para definir o nome de usuário do proxy. Esse parâmetro corresponde ao parâmetro `ClientConfiguration.proxyUserName` no AWS SDK. Para obter mais informações, consulte AWS Client configuration no Guia do desenvolvedor do AWS SDK for C++.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
ProxyUID	Opcional	none	`ProxyUID= username;`

Senha do proxy

Use este parâmetro para definir a senha do proxy. Esse parâmetro corresponde ao parâmetro `ClientConfiguration.proxyPassword` no AWS SDK. Para obter mais informações, consulte AWS Client configuration no Guia do desenvolvedor do AWS SDK for C++.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
ProxyPWD	Opcional	`none`	`ProxyPWD=password;`

Host sem proxy

Use este parâmetro opcional para especificar um host ao qual o driver se conecta sem usar proxy. Esse parâmetro corresponde ao parâmetro `ClientConfiguration.nonProxyHosts` no AWS SDK. Para obter mais informações, consulte AWS Client configuration no Guia do desenvolvedor do AWS SDK for C++.

O parâmetro de conexão `NonProxyHost` é transmitido para a opção `CURLOPT_NOPROXY` do curl. Para obter informações sobre o formato `CURLOPT_NOPROXY`, consulte CURLOPT_NOPROXY na documentação do curl.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
NonProxyHost	Opcional	`none`	`NonProxyHost=.amazonaws.com,localhost,.example.net,.example.com;`

Usar proxy

Habilita o tráfego do usuário por meio do proxy especificado.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
UseProxy	Opcional	none	`UseProxy=1;`

Opções de registro em log

É necessário ter direitos de administrador para modificar as configurações descritas aqui. Para fazer as alterações, você pode usar a caixa de diálogo Opções de registro em log do administrador de fonte de dados ODBC ou modificar o registro do Windows diretamente.

Nível de log

Essa opção habilita os logs do driver ODBC. No Windows, você pode usar o registro ou uma caixa de diálogo para habilitar ou desabilitar o registro em log. A opção está localizada no seguinte caminho de registro:

```
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Amazon Athena\ODBC\Driver
```

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
LogLevel	Opcional	`0`	`LogLevel=1;`

Caminho do log.

Especifica o caminho para o arquivo em que os logs do driver ODBC estão armazenados. É possível usar o registro ou uma caixa de diálogo para definir esse valor. A opção está localizada no seguinte caminho de registro:

```
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Amazon Athena\ODBC\Driver
```

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
LogPath	Opcional	none	`LogPath=C:\Users\ username\projects \internal\trunk\;`

Usar o AWS Logger

Especifica se o registro em log do AWS SDK está habilitado. Especifique 1 para habilitar, 0 para desabilitar.

Nome da string de conexão	Tipo de parâmetro	Valor padrão	Exemplo de string de conexão
UseAwsLogger	Opcional	`0`	`UseAwsLogger=1;`

Migrar para o driver ODBC 2.x

Como a maioria dos parâmetros de conexão do ODBC 2.x do Athena é compatível com versões anteriores do driver ODBC 1.x, é possível reutilizar a maior parte da string de conexão existente com o driver ODBC 2.x do Athena. Porém, os parâmetros de conexão a seguir necessitam de modificações.

Nível de log

Embora o driver ODBC atual forneça uma variedade de opções de registro em log disponíveis, começando de `LOG_OFF (0)` a `LOG_TRACE (6)`, o driver ODBC do Amazon Athena tem apenas dois valores: 0 (desabilitado) e 1 (habilitado).

Para obter mais informações sobre o registro em log do driver ODBC 2.x, consulte Opções de registro em log.

	Driver ODBC 1.x	Driver ODBC 2.x
Nome da string de conexão	`LogLevel`	`LogLevel`

	Driver ODBC 1.x	Driver ODBC 2.x
Tipo de parâmetro	Opcional	Opcional
Valor padrão	`0`	`0`
Possíveis valores	`0-6`	`0,1`
Exemplo de string de conexão	`LogLevel=6;`	`LogLevel=1;`

MetadataRetrievalMethod

O driver ODBC atual oferece várias opções para recuperar os metadados do Athena. O driver ODBC do Amazon Athena descontinua o `MetadataRetrievalMethod` e sempre usa a API do Amazon Athena para extrair metadados.

O Athena inclui o sinalizador `QueryExternalCatalogs` para consultar catálogos externos. Para consultar catálogos externos com o driver ODBC atual, defina `MetadataRetrievalMethod` como `ProxyAPI`. Para consultar catálogos externos com o driver ODBC do Athena, defina `QueryExternalCatalogs` como `1`.

	Driver ODBC 1.x	Driver ODBC 2.x
Nome da string de conexão	`MetadataRetrievalMethod`	`QueryExternalCatalogs`
Tipo de parâmetro	Opcional	Opcional
Valor padrão	`Auto`	`0`
Possíveis valores	`Auto`, `AWS Glue`, `ProxyAPI`, `Query`	`0`,`1`
Exemplo de string de conexão	`MetadataRetrievalMethod=ProxyAPI;`	`QueryExternalCatalogs=1;`

Teste de conexão

Quando você testa uma conexão do driver ODBC 1.x, o driver executa uma consulta `SELECT 1` que gera dois arquivos no bucket do Amazon S3: um para o conjunto de resultados e outro para os metadados. A conexão de teste é cobrada conforme a política de preços do Amazon Athena.

Quando você testa uma conexão do driver ODBC 2.x, o driver chama a ação da API GetWorkGroup do Athena. A chamada usa o tipo de autenticação e o provedor de credenciais correspondente que você especificou para recuperar credenciais. Não há cobrança pelo teste de conexão ao usar o driver ODBC 2.x, e o teste não gera resultados de consulta no bucket do Amazon S3.

Solução de problemas do driver ODBC 2.x

Se você encontrar problemas com o driver ODBC do Amazon Athena, entre em contato com o AWS Support (no AWS Management Console, escolha Suporte, Support Center).

Não se esqueça de incluir as informações a seguir e forneça outros detalhes que ajudem a equipe de suporte a entender seu caso de uso.

- Descrição (obrigatório): uma descrição contendo informações detalhadas sobre seu caso de uso e a diferença entre o comportamento esperado e o observado. Inclua todas as informações que possam ajudar os engenheiros de suporte a acompanhar o problema com facilidade. Se o problema for intermitente, especifique as datas, os carimbos de data e hora ou os pontos de intervalo em que o problema ocorreu.
- Informações sobre a versão (obrigatório): informações sobre a versão do driver, o sistema operacional e as aplicações usadas. Por exemplo, “Driver ODBC versão 1.2.3, Windows 10 (x64), Power BI”.
- Arquivos de log (obrigatório): o número mínimo de arquivos de log do driver ODBC necessários para entender o problema. Para obter informações sobre as opções de registro em log para o driver ODBC 2.x, consulte Opções de registro em log.
- String de conexão (obrigatório): sua string de conexão ODBC ou uma captura de tela da caixa de diálogo que mostra os parâmetros de conexão utilizados. Para obter informações sobre parâmetros de conexão, consulte Parâmetros de conexão do ODBC 2.x do Athena.
- Etapas do problema (opcional): se possível, inclua etapas ou um programa independente que possa ajudar a reproduzir o problema.
- Informações de erro de consulta (opcional): em caso de erros que envolvam consultas DML ou DDL, inclua as seguintes informações:

  - Uma versão completa ou simplificada da consulta DML ou DDL com falha.
  - O ID da conta e a Região da AWS usada e o ID de execução da consulta.
- Erros de SAML (opcional): se você tiver um problema relacionado à autenticação com a declaração SAML, inclua as seguintes informações:
  - O provedor de identidades e o plug-in de autenticação que foram usados.
  - Um exemplo com o token SAML.

Notas da versão do ODBC 2.x do Amazon Athena

Essas notas de versão fornecem detalhes de aprimoramentos, atributos, problemas conhecidos e alterações de fluxo de trabalho no driver ODBC 2.x do Amazon Athena.

2.0.3.0

Lançado em 08/04/2024

O driver ODBC v2.0.3.0 do Amazon Athena contém as seguintes melhorias e correções.

Melhorias

- Adição de compatibilidade com MFA para o plug-in de autenticação Okta nas plataformas Linux e Mac.
- Agora, a biblioteca `athena-odbc.dll` e o instalador `AmazonAthenaODBC-2.x.x.x.msi` para Windows estão assinados.
- Atualização do arquivo `cacert.pem` de certificado CA que é instalado com o driver.
- Melhora no tempo necessário para listar tabelas nos catálogos Lambda. Para catálogos do tipo `LAMBDA`, agora o driver ODBC pode enviar uma consulta SHOW TABLES para obter uma lista das tabelas disponíveis. Para obter mais informações, consulte Usar consulta para listar tabelas.
- Introdução do parâmetro de conexão `UseWCharForStringTypes` para relatar tipos de dados de string usando `SQL_WCHAR` e `SQL_WVARCHAR`. Para obter mais informações, consulte Usar WCHAR para tipos de strings.

Correções

- Correção de um aviso sobre corrupção do registro que ocorria quando a ferramenta Get-OdbcDsn do PowerShell era usada.
- Atualização da lógica de análise para processar comentários no início das strings de consulta.

- Agora, os tipos de dados de data e de carimbo de data/hora permitem zero no campo do ano.

Para baixar o driver ODBC v2, consulte Download do driver ODBC 2.x. Para obter informações sobre conexão, consulte Amazon Athena ODBC 2.x.

2.0.2.2

Lançado em 13/02/2024

O driver ODBC v2.0.2.2 do Amazon Athena contém as seguintes melhorias e correções.

Melhorias

- Adição de dois parâmetros de conexão, `StringColumnLength` e `ComplexTypeColumnLength`, que você pode usar para alterar o comprimento padrão da coluna para tipos de dados de string e complexos. Para obter mais informações, consulte Comprimento da coluna de string e Comprimento de coluna de tipo complexo.
- Adição de compatibilidade com os sistemas operacionais Linux e macOS (Intel e ARM). Para obter mais informações, consulte Linux e macOS.
- Atualização de AWS-SDK-CPP para a versão de tag 1.11.245.
- Atualização da biblioteca curl para a versão 8.6.0.

Correções

- Solução de um problema que causava o relato de valores incorretos nos metadados do conjunto de resultados para tipos de dados semelhantes a strings na coluna de precisão.

Para fazer download do driver ODBC v2, consulte Download do driver ODBC 2.x. Para obter informações sobre conexão, consulte Amazon Athena ODBC 2.x.

2.0.2.1

Lançada em 7/12/2023

O driver ODBC v2.0.2.1 do Amazon Athena contém as melhorias e correções apresentadas a seguir.

Melhorias

- Aprimoramento da segurança de thread do driver ODBC para todas as interfaces.

- Quando o registro em log estiver habilitado, os valores de data e de hora passarão a ser registrados com precisão de milissegundos.
- Durante a autenticação com o plug-in Browser SSO OIDC, o terminal será aberto para exibir o código do dispositivo ao usuário.

Correções

- Resolução de um problema de liberação de memória que ocorria ao analisar resultados da API de transmissão.
- Solicitações para as interfaces `SQLTablePrivileges()`, `SQLSpecialColumns()`, `SQLProcedureColumns()` e `SQLProcedures()` retornarão conjuntos de resultados vazios.

Para fazer download do driver ODBC v2, consulte Download do driver ODBC 2.x. Para obter informações sobre conexão, consulte Amazon Athena ODBC 2.x.

2.0.2.0

Lançado em 17/10/2023

O driver ODBC v2.0.2.0 do Amazon Athena contém as seguintes melhorias e correções.

Melhorias

- O atributo de cache de arquivos foi adicionado aos plug-ins de autenticação baseados no navegador Azure AD, no Browser SSO OIDC e no Okta.

  As ferramentas de BI, como o Power BI e plug-ins baseados em navegador, usam várias janelas de navegador. O novo parâmetro de conexão de cache de arquivos permite que credenciais temporárias sejam armazenadas em cache e reutilizadas entre vários processos abertos por aplicações de BI.
- Agora, as aplicações podem consultar informações sobre o conjunto de resultados depois que uma instrução é preparada.
- Os tempos limite padrão de conexão e solicitação foram aumentados para uso com redes de clientes mais lentas. Para obter mais informações, consulte Tempo limite da conexão e Tempo limite da solicitação.
- As substituições de endpoint foram adicionadas para SSO e SSO OIDC. Para obter mais informações, consulte Substituições de endpoints.

- Foi adicionado um parâmetro de conexão para passar um argumento de URI para uma solicitação de autenticação ao Ping. Você pode usar esse parâmetro para ignorar a limitação de perfil único do Lake Formation. Para obter mais informações, consulte Parâmetro de URI do Ping.

Correções

- Corrigido um problema de estouro de números inteiros que ocorria ao usar o mecanismo de vinculação baseado em linha.
- O tempo limite foi removido da lista de parâmetros de conexão exigidos para o plug-in de autenticação do Browser SSO OIDC.
- Foram adicionadas as interfaces que faltavam para `SQLStatistics()`, `SQLPrimaryKeys()`, `SQLForeignKeys()` e `SQLColumnPrivileges()`, e a capacidade de retornar conjuntos de resultados vazios mediante solicitação.

Para baixar o driver ODBC v2, consulte Download do driver ODBC 2.x. Para obter informações sobre conexão, consulte Amazon Athena ODBC 2.x.

2.0.1.1

Lançado em 10/08/2023

O driver Amazon Athena ODBC v2.0.1.1 contém as seguintes melhorias e correções.

Melhorias

- Foi adicionado o registro de URI ao plug-in de autenticação Okta.
- Foi adicionado o parâmetro de função preferencial ao plug-in externo do provedor de credenciais.
- Adicionando tratamento para o prefixo do perfil no nome do perfil de arquivo de configuração AWS.

Correções

- Corrigido um problema Região da AWS de uso que ocorreu ao trabalhar com Lake Formation e clientes AWS STS.
- Restaurou as chaves de partição ausentes na lista de colunas da tabela.
- Adicionou o tipo de autenticação `BrowserSSOOIDC` que faltava para o perfil AWS.

Para baixar o driver ODBC v2, consulte Download do driver ODBC 2.x.

2.0.1.0

Lançado em 29/06/2023

O Amazon Athena lança o driver ODBC v2.0.1.0.

O Athena lançou um novo driver ODBC que melhora a experiência de conexão, consulta e visualização de dados de aplicações compatíveis de desenvolvimento SQL e business intelligence. A versão mais recente do driver ODBC do Athena é compatível com recursos do driver existente e é fácil de atualizar. A nova versão tem suporte para autenticação de usuários pelo AWS IAM Identity Center. Também oferece a opção de ler os resultados da consulta do Amazon S3, podendo disponibilizar os resultados da consulta para você mais cedo.

Para obter mais informações, consulte Amazon Athena ODBC 2.x.

Driver ODBC 1.x do Athena

Use os links desta página para baixar o contrato de licença para o driver ODBC 1.x do Amazon Athena, os drivers ODBC e a documentação do ODBC. Para obter informações sobre a string de conexão ODBC, consulte o arquivo PDF do Guia de Instalação e Configuração do Driver ODBC, disponível para download nesta página. Para obter informações sobre permissões, consulte Acesso por meio de conexões JDBC e ODBC.

> **Important**
>
> Ao usar o driver ODBC 1.x, observe os seguintes requisitos:
>
> - Porta 444 aberta: mantenha a porta 444, que o Athena usa para fazer uma transmissão dos resultados das consultas, aberta para o tráfego de saída. Ao usar um endpoint do PrivateLink para se conectar ao Athena, verifique se o grupo de segurança anexado ao endpoint do PrivateLink está aberto para o tráfego de entrada na porta 444.
> - Política athena:GetQueryResultsStream: adicione a ação de política `athena:GetQueryResultsStream` às entidades principais do IAM que usam o driver ODBC. Essa ação de política não é exposta diretamente com a API. Ela apenas é usada com drivers ODBC e JDBC como parte do suporte a resultados de transmissão. Para visualizar um exemplo de política, consulte Política gerenciada AWS: AWSQuicksightAthenaAccess.

Windows

Versão do driver	Link para fazer download
ODBC 1.2.3.1000 para Windows 32 bits	Driver ODBC 1.2.3.1000 para Windows 32 bits
ODBC 1.2.3.1000 para Windows 64 bits	Driver ODBC 1.2.3.1000 para Windows 64 bits

Linux

Versão do driver	Link para fazer download
ODBC 1.2.3.1000 para Linux 32 bits	Driver ODBC 1.2.3.1000 para Linux 32 bits
ODBC 1.2.3.1000 para Linux 64 bits	Driver ODBC 1.2.3.1000 para Linux 64 bits

OSX

Versão do driver	Link para fazer download
ODBC 1.2.3.1000 para OSX	Driver ODBC 1.2.3.1000 para OSX

Documentação

Conteúdo	Link da documentação
Contrato de licença do driver ODBC para Amazon Athena	Contrato de licença
Documentação para o ODBC 1.2.3.1000	Guia de instalação e de configuração do driver ODBC versão 1.2.3.1000
Notas de versão para o ODBC 1.2.3.1000	Notas de versão do driver ODBC versão 1.2.3.1000

Notas do driver ODBC

Conexão sem usar proxy

Se você quiser especificar determinados hosts aos quais o driver se conecta sem usar um proxy, poderá usar a propriedade `NonProxyHost` opcional na string de conexão do ODBC.

A propriedade `NonProxyHost` especifica uma lista separada por vírgulas de hosts que o conector pode acessar sem passar pelo servidor de proxy quando uma conexão por proxy está habilitada, como no seguinte exemplo:

```
.amazonaws.com,localhost,.example.net,.example.com
```

O parâmetro de conexão `NonProxyHost` é transmitido para a opção `CURLOPT_NOPROXY` do curl. Para obter informações sobre o formato `CURLOPT_NOPROXY`, consulte CURLOPT_NOPROXY na documentação do curl.

Configuração do acesso federado ao Amazon Athena para usuários do Microsoft AD FS usando um cliente ODBC

Para configurar o acesso federado ao Amazon Athena para usuários do Microsoft Active Directory Federation Services (AD FS) usando um cliente ODBC, primeiro estabeleça a confiança entre o AD FS e sua conta da AWS. Com essa confiança estabelecida, os usuários do AD podem se federar na AWS usando as credenciais do AD e assumir permissões de um perfil do AWS Identity and Access Management (IAM) para acessar recursos da AWS, como a API do Athena.

Para criar essa confiança, você adiciona o AD FS como um provedor SAML à sua Conta da AWS e cria um perfil do IAM que os usuários federados podem assumir. No lado do AD FS, você adiciona a AWS como parte confiável e grava regras de declaração SAML para enviar os atributos de usuário adequados à AWS para autorização (especificamente, do Athena e do Amazon S3).

A configuração do acesso do AD FS ao Athena envolve as seguintes etapas principais:

1. Configuração de um provedor e de um perfil SAML do IAM

2. Configuração do AD FS

3. Criação de usuários e grupos do Active Directory

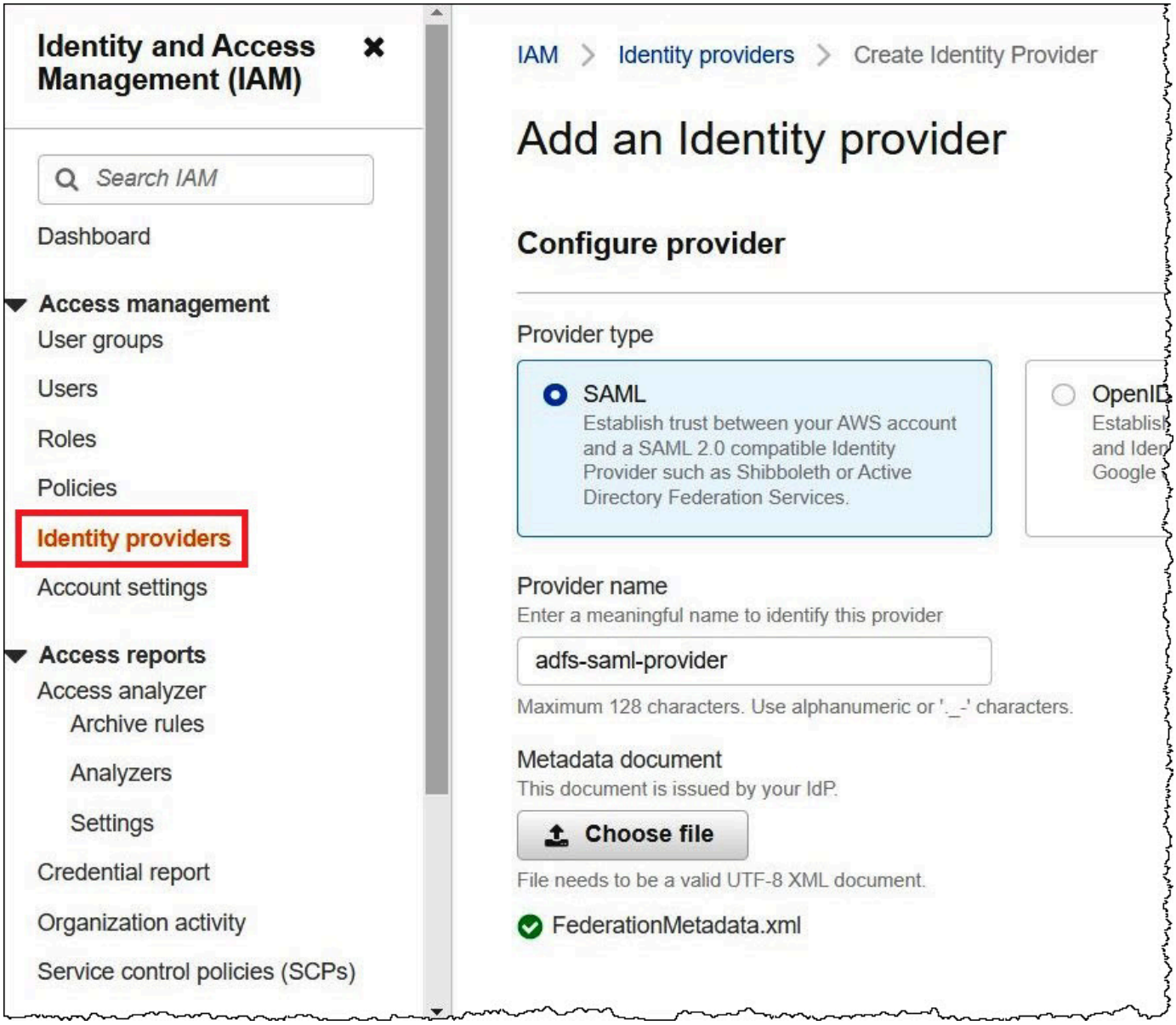
4. Configuração da conexão ODBC do AD FS para o Athena

1. Configuração de um provedor e de um perfil SAML do IAM

Nesta seção, você adicionará o AD FS como um provedor SAML à sua conta da AWS e criará um perfil do IAM que seus usuários federados poderão assumir.

Para configurar um provedor SAML

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Identity providers (Provedores de identidade).
3. Escolha Add provider (Adicionar provedor).
4. Em Provider type (Tipo de provedor), escolha SAML.



5. Em Provider name (Nome do provedor), insira **`adfs-saml-provider`**.
6. Em um navegador, insira o endereço a seguir para baixar do arquivo XML de federação para seu servidor do AD FS. Para executar essa etapa, o navegador deve ter acesso ao servidor do AD FS.

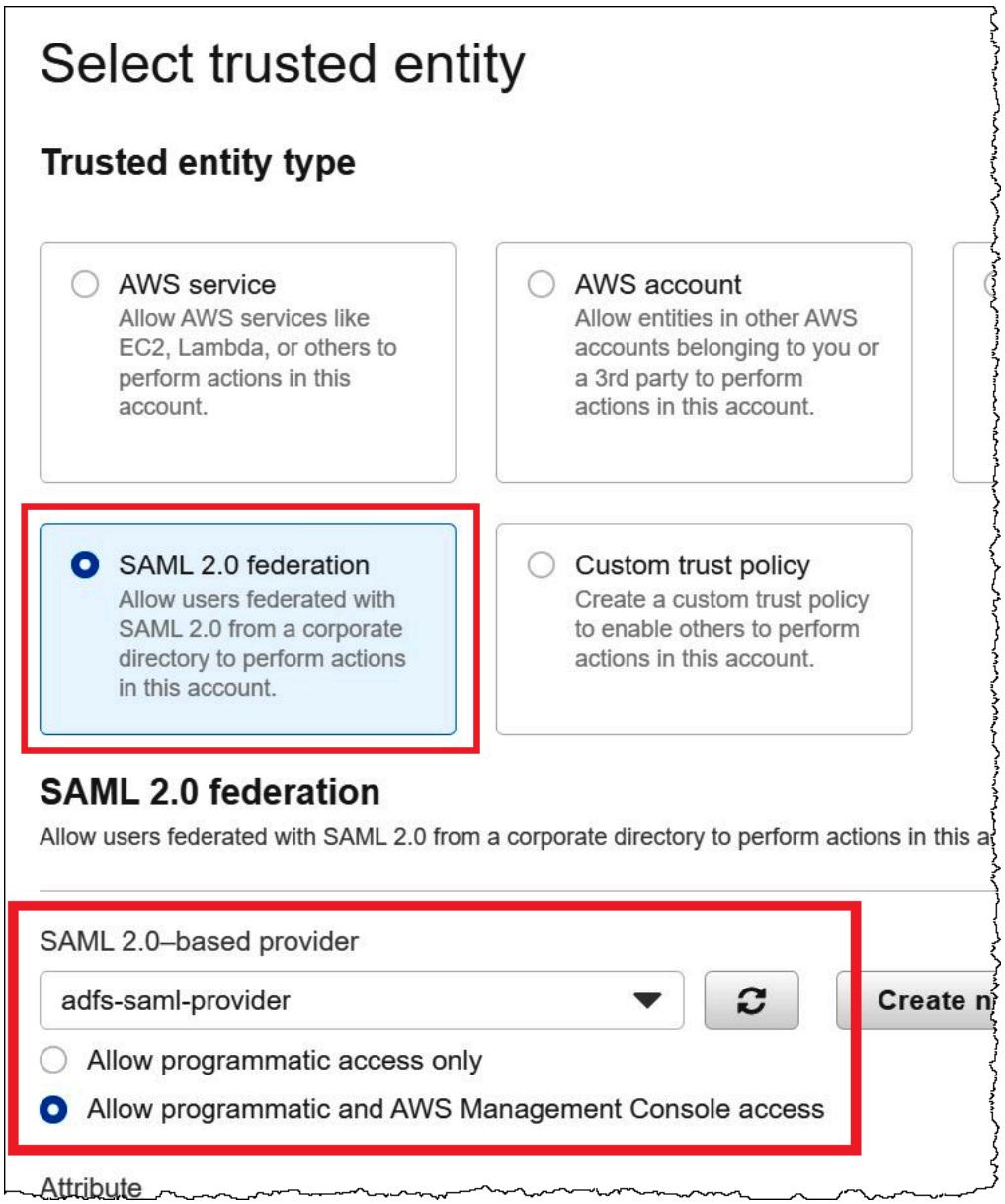
   ```
   https://adfs-server-name/federationmetadata/2007-06/federationmetadata.xml
   ```

7. No console do IAM, em Metadata document (Documentos de metadados), escolha Choose file (Escolher arquivo) e, em seguida, realize o upload do arquivo de metadados da federação para a AWS.
8. Para finalizar, escolha Add provider (Adicionar provedor).

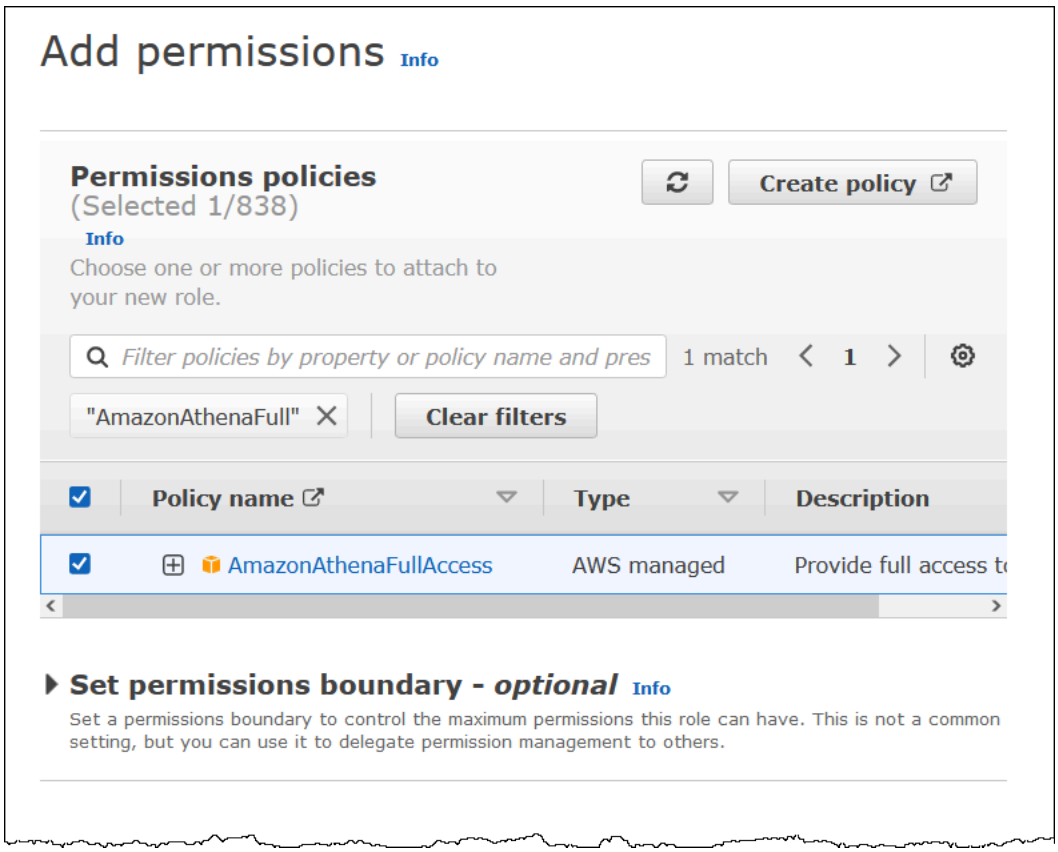
Em seguida, você criará o perfil do IAM que seus usuários federados poderão assumir.

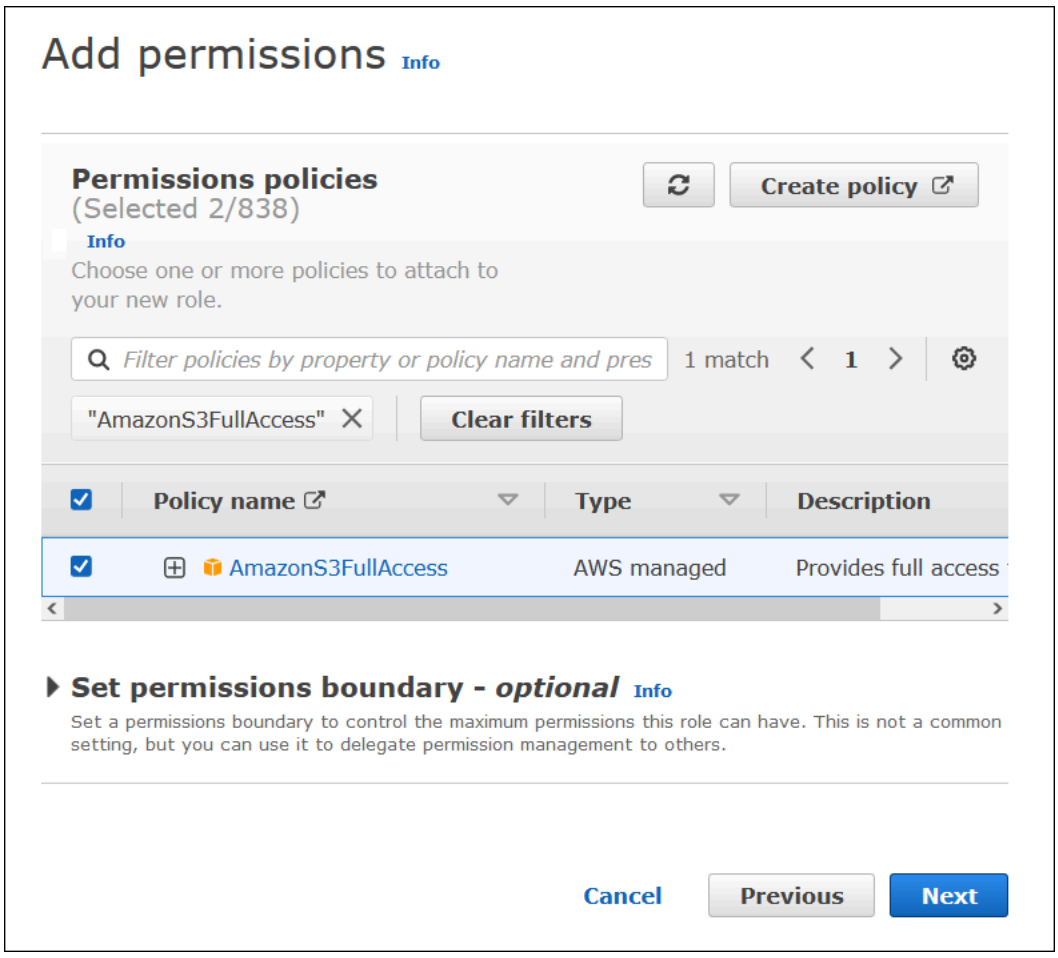
Para criar um perfil do IAM para usuários federados

1. No painel de navegação do console do IAM, escolha Roles (Perfis).
2. Selecione Criar função.
3. Em Trusted entity type (Tipo de entidade confiável), escolha SAML 2.0 federation (Federação SAML 2.0).
4. Em SAML 2.0-based provider (Provedor baseado em SAML 2.0), escolha o provedor adfs-saml-provider que você criou.
5. Escolha Permitir acesso programático e ao Console de Gerenciamento da AWS e escolha Próximo.



6. Na página Add permissions (Adicionar permissões), filtre as políticas de permissões do IAM necessárias para o perfil e, em seguida, marque as caixas de seleção correspondentes. Este tutorial anexa as políticas `AmazonAthenaFullAccess` e `AmazonS3FullAccess`.


Add permissions Info
Permissions policies
(Selected 1/838)
Info
Choose one or more policies to attach to your new role.
Create policy
Filter policies by property or policy name and pres
1 match
1
"AmazonAthenaFull"
Clear filters
Policy name
Type
Description
AmazonAthenaFullAccess
AWS managed
Provide full access t
Set permissions boundary - optional Info
Set a permissions boundary to control the maximum permissions this role can have. This is not a common setting, but you can use it to delegate permission management to others.

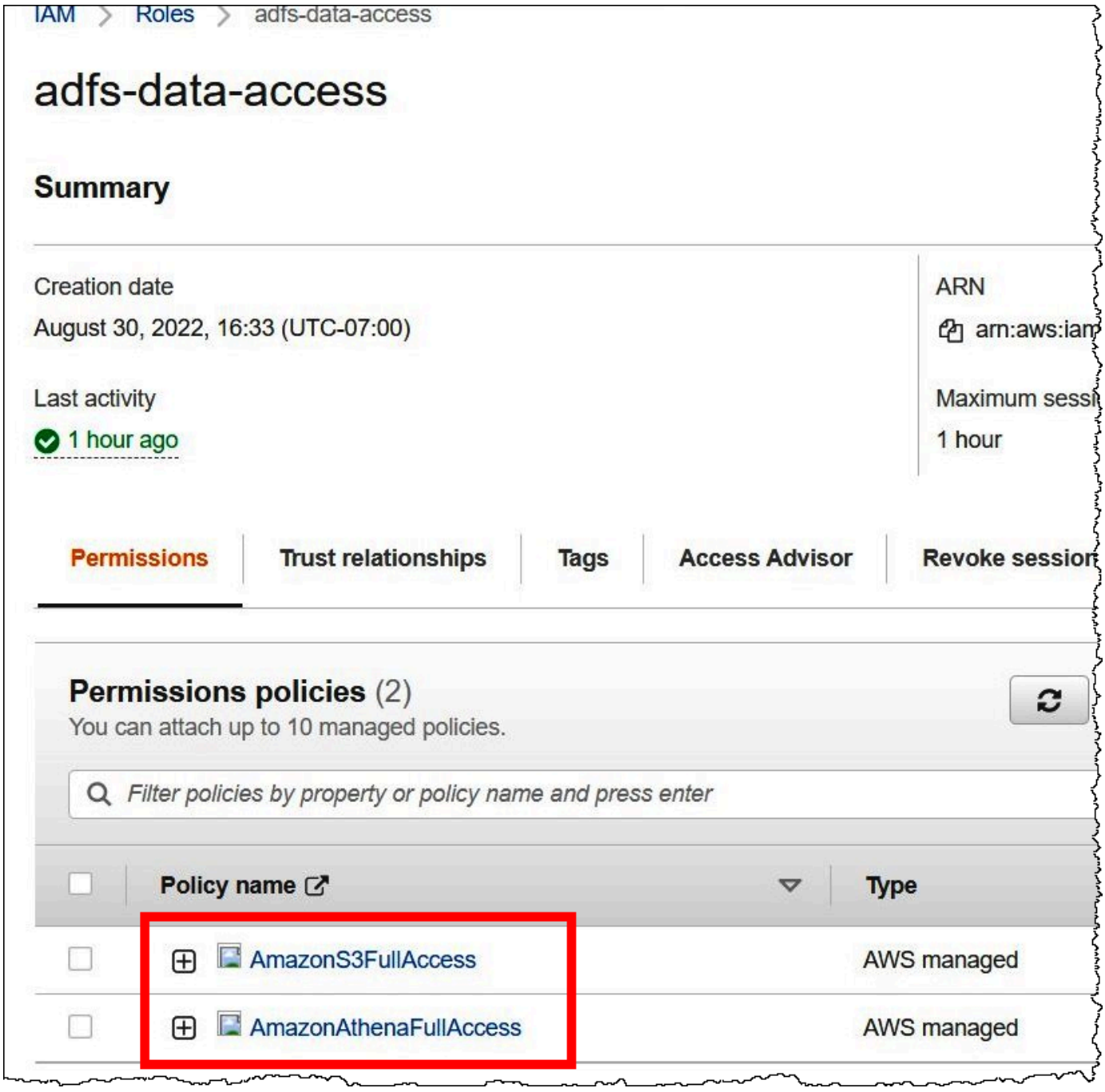


7. Escolha Próximo.
8. Na página Name, review, and create (Nomear, analisar e criar), em para Role name (Nome do perfil), insira um nome para o perfil. Este tutorial usa o nome adfs-data-access.

   Em Step 1: Select trusted entities (Etapa 1: selecionar entidades confiáveis), o campo Principal (Entidade principal) deve ser preenchido automaticamente com `"Federated:" "arn:aws:iam::account_id:saml-provider/adfs-saml-provider"`. O campo `Condition` deve conter `"SAML:aud"` e `"https://signin.aws.amazon.com/saml"`.

Em Step 2: Add permissions (Etapa 2: adicionar permissões) é apresentado as políticas vinculadas ao perfil.

9. Selecione Criar função. Uma mensagem da barra de notificação confirma a criação do perfil.
10. Na página Roles (Perfis), escolha o nome do perfil que você acabou de criar. A página de resumo do perfil mostra as políticas que foram vinculadas.



2. Configuração do AD FS

Agora está tudo pronto para você adicionar a AWS como parte confiável e gravar regras de declaração SAML para poder enviar os atributos de usuário adequados à AWS para autorização.

A federação baseada em SAML tem duas partes participantes: o IdP (Active Directory) e a parte confiável (AWS), que é o serviço ou aplicação que usará a autenticação do IdP.

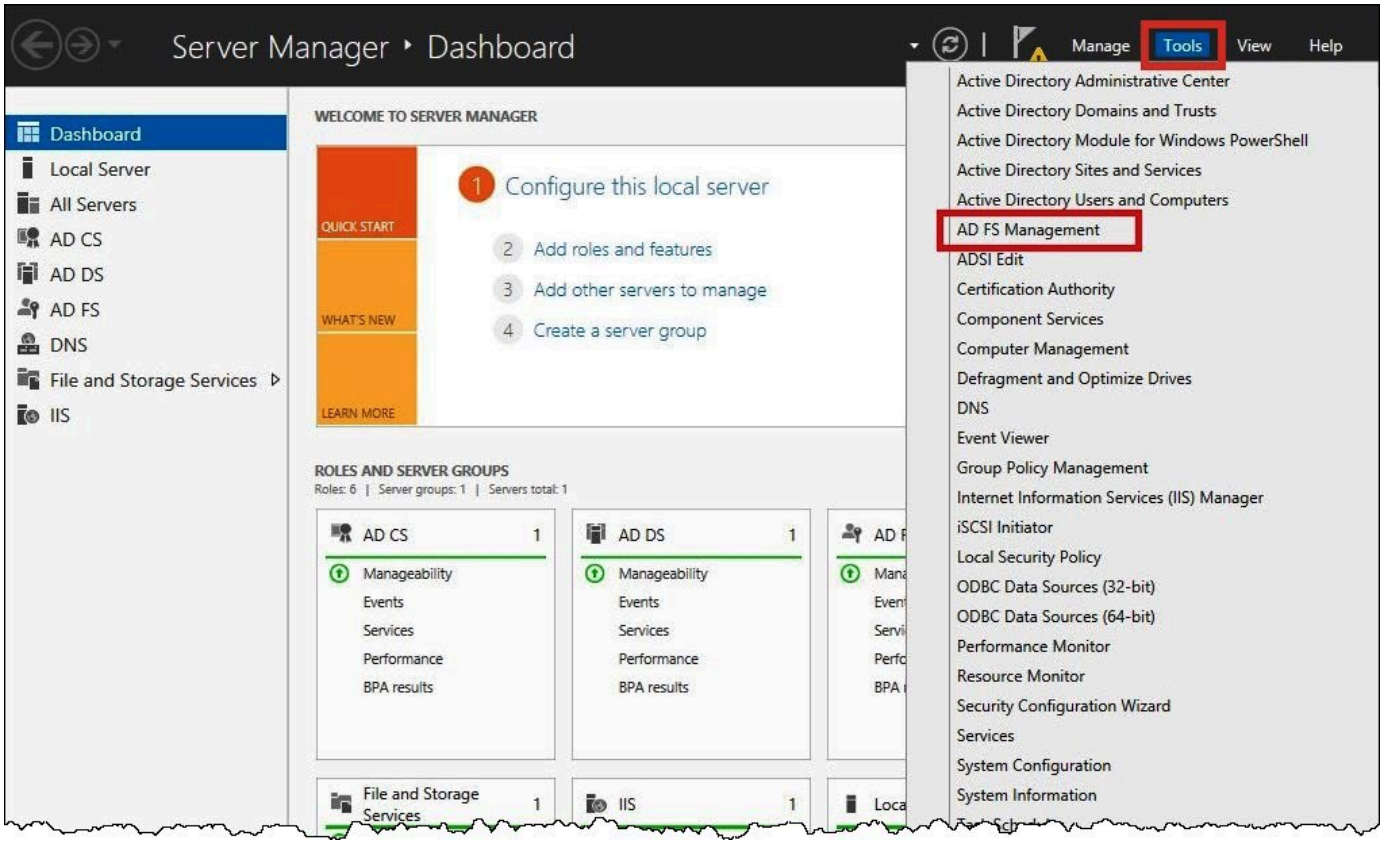
Para configurar o AD FS, primeiro é necessário adicionar um objeto de confiança de terceira parte confiável e, em seguida, configurar as regras de declaração SAML para a parte confiável. O AD FS usa regras de declaração para formar uma declaração SAML que é enviada para uma parte confiável. A asserção SAML afirma que as informações sobre o usuário do AD são verdadeiras e que o usuário foi autenticado.

Como adicionar um objeto de confiança de terceira parte confiável

Para adicionar um objeto de confiança de terceira parte confiável no AD FS, use o gerenciador de servidores do AD FS.

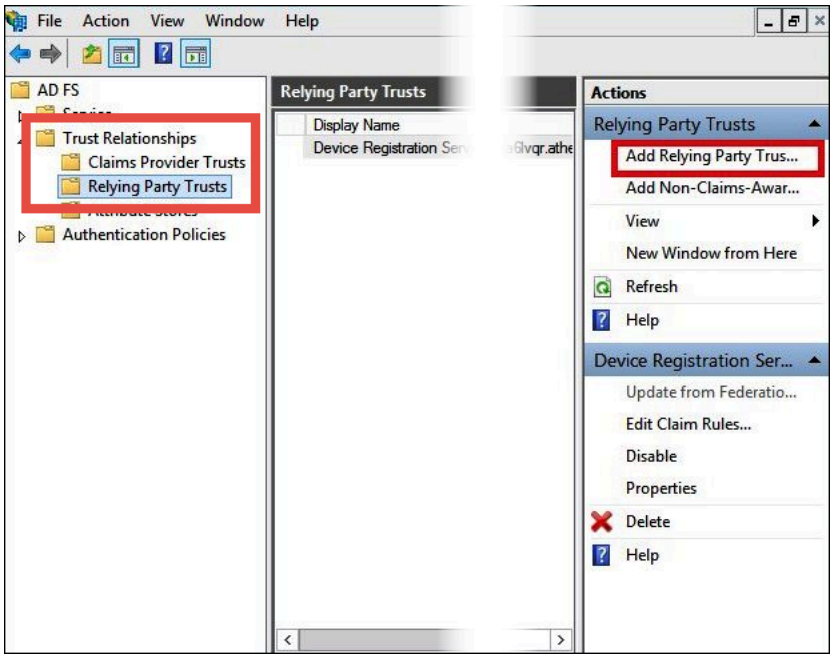
Para adicionar um objeto de confiança de terceira parte confiável no AD FS

1. Entre no servidor do AD FS.
2. No menu Start (Iniciar), abra Server Manager (Gerenciador do Servidor).
3. Escolha Tools (Ferramentas) e, em seguida, selecione AD FS Management (Gerenciamento do AD FS).

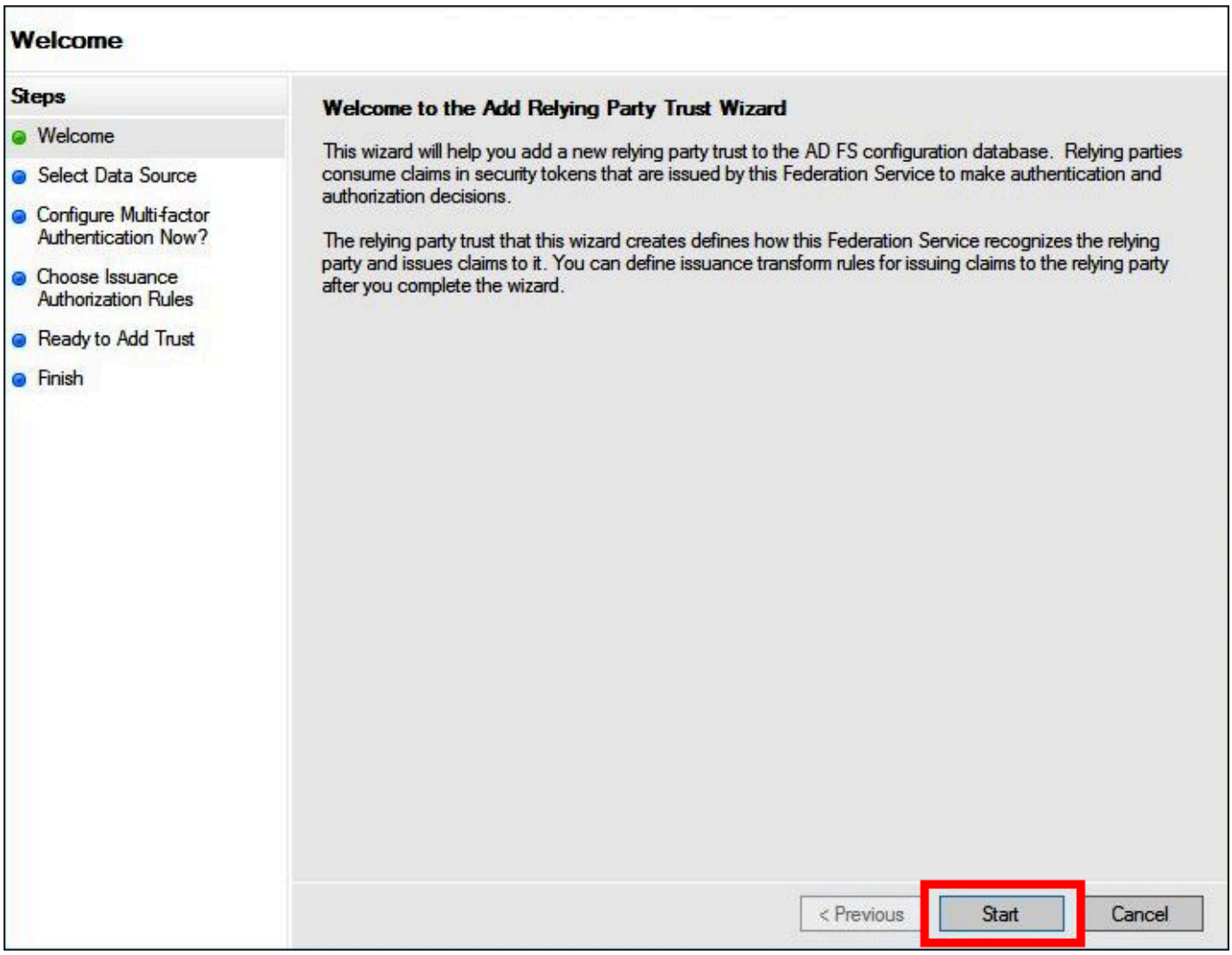


4. No painel de navegação, em Trust Relationships (Relações de confiança), escolha Relying Party Trusts (Objeto de confiança de terceira parte confiável).

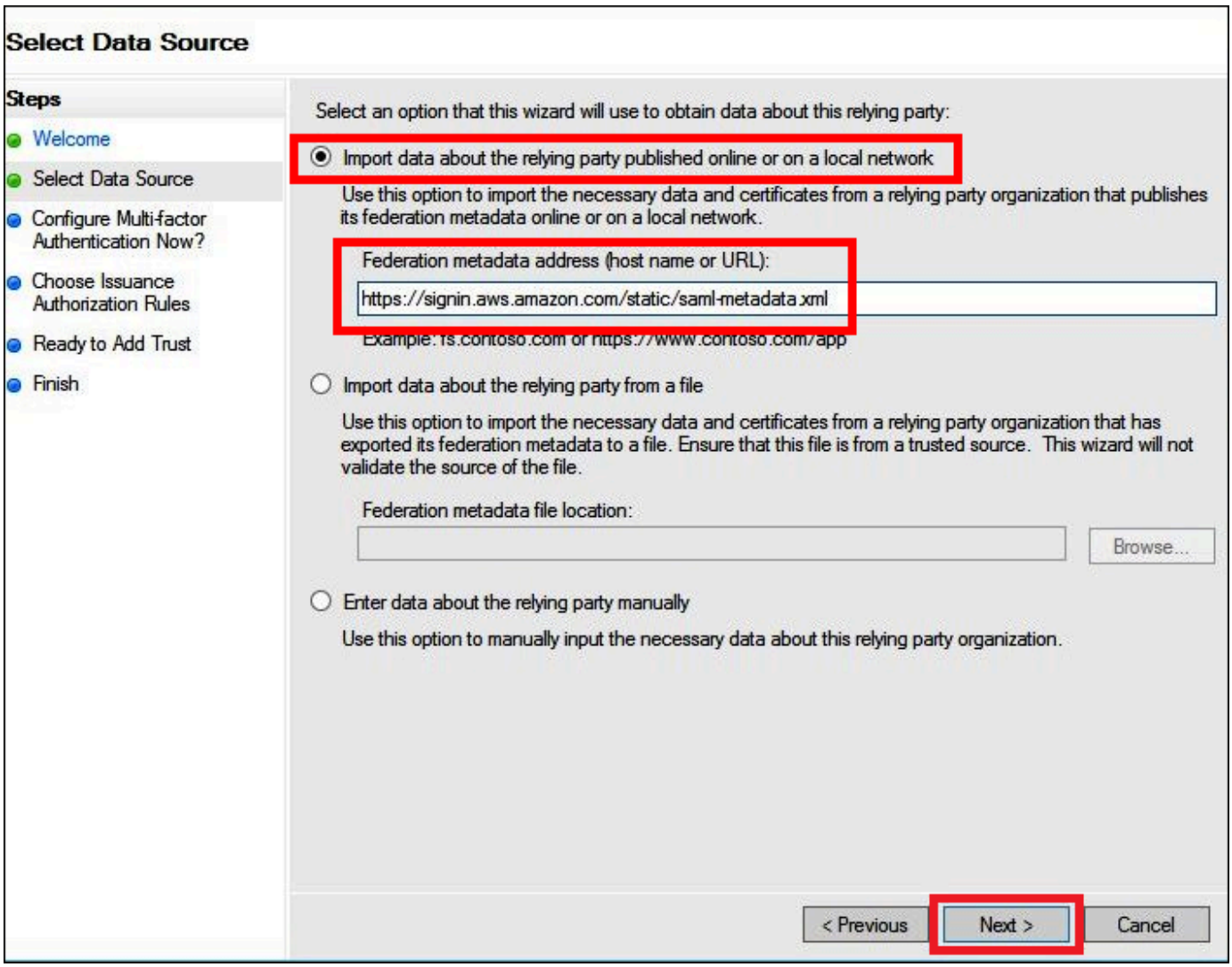
5. Em Actions (Ações), escolha Add Relying Party Trust (Adicionar objeto de confiança de terceira parte confiável).



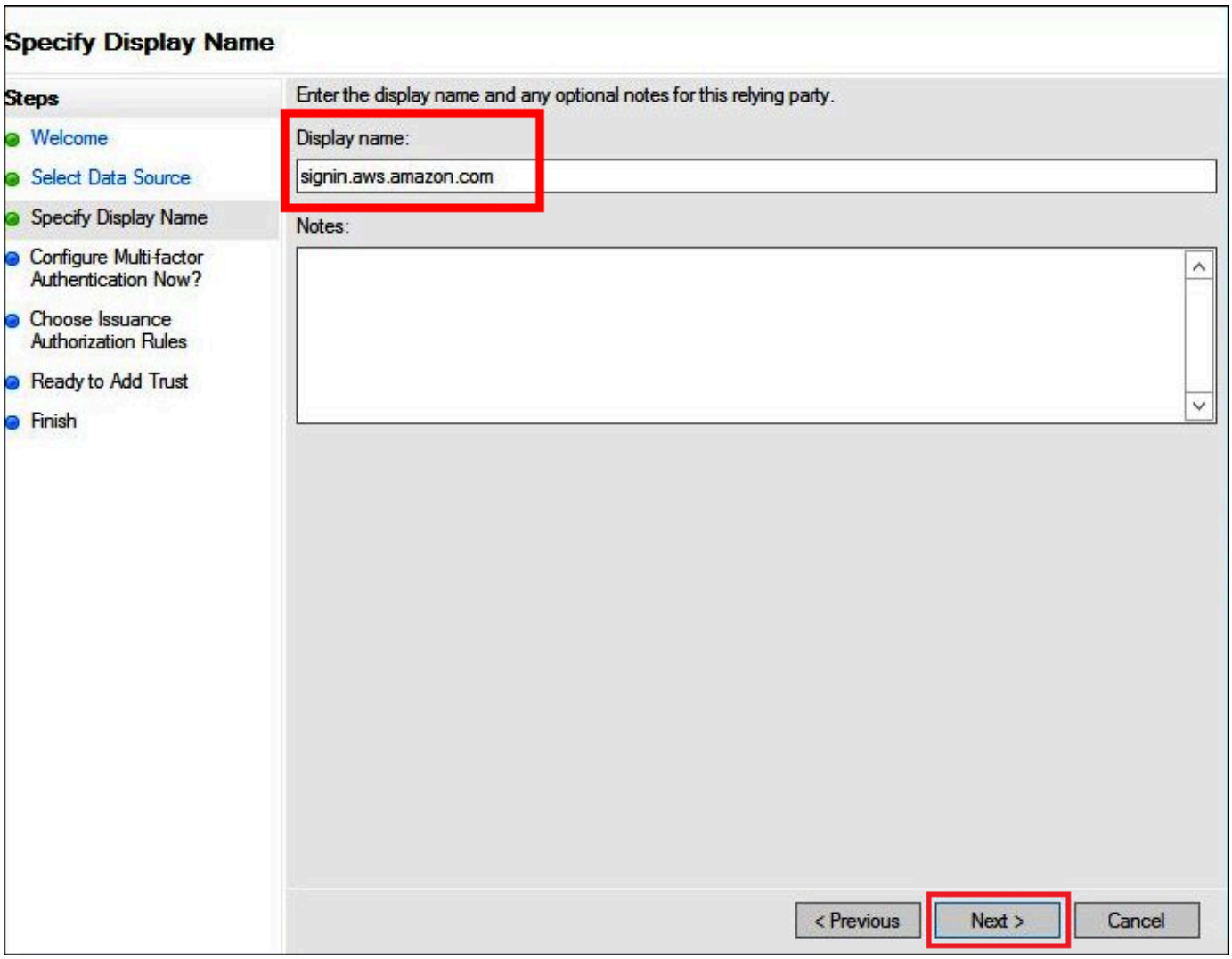
6. Na página Add Relying Party Trust Wizard (Assistente para adicionar confiança da parte dependente) escolha Start (Iniciar).



7. Na tela Select Data Source (Selecionar fonte de dados), selecione a opção Import data about the relying party published online or on a local network (Importar dados sobre a parte confiável publicados online ou em uma rede local).
8. Em Federation metadata address (host name or URL) (Endereço de metadados de federação [nome do host ou URL]), insira o URL **`https://signin.aws.amazon.com/static/saml-metadata.xml`**.
9. Escolha Next (próximo).

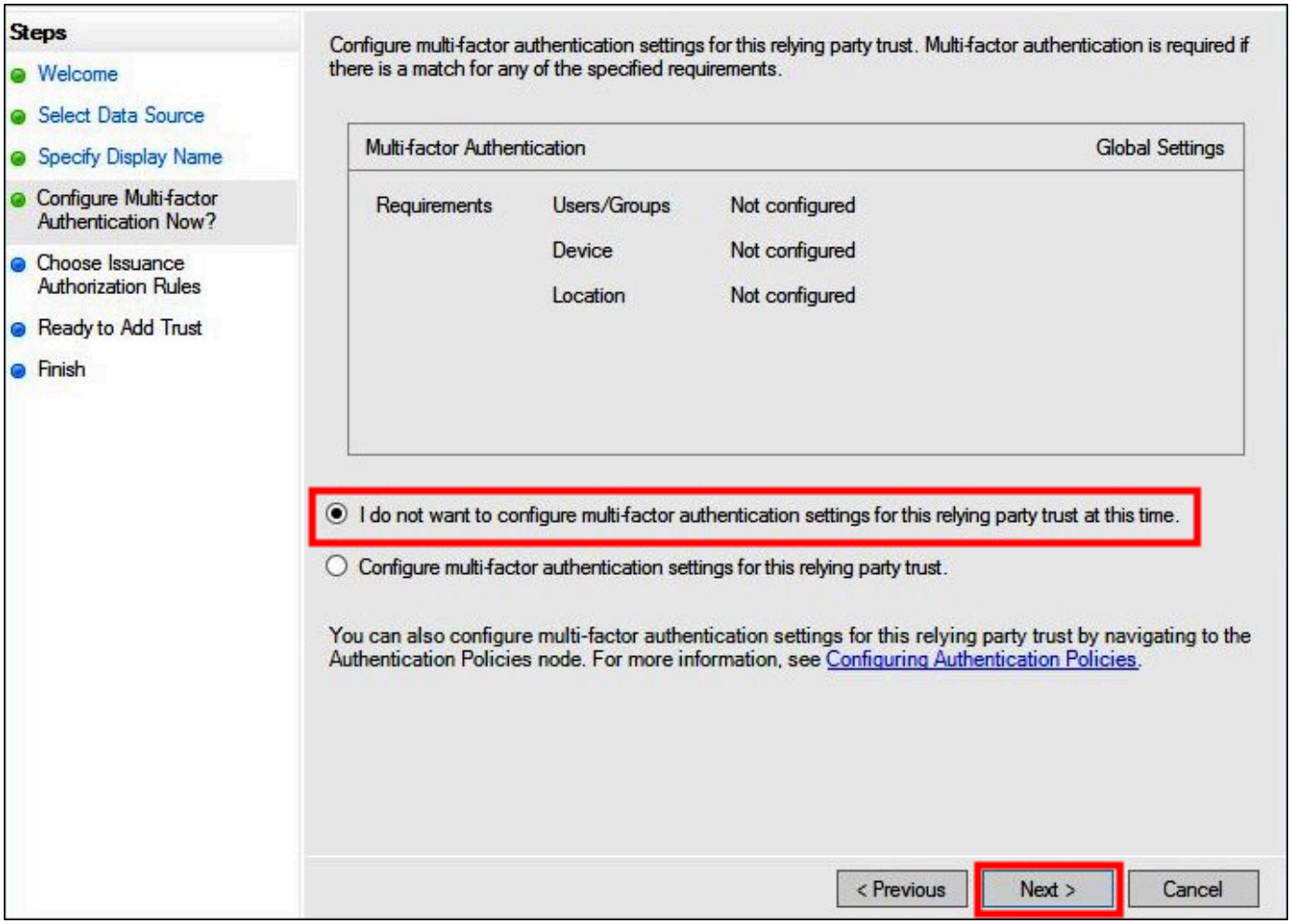


10. Na página Specify Display Name (Especificar nome de exibição), em Display name (Nome de exibição), insira um nome de exibição para a parte confiável e, em seguida, escolha Next (Avançar).



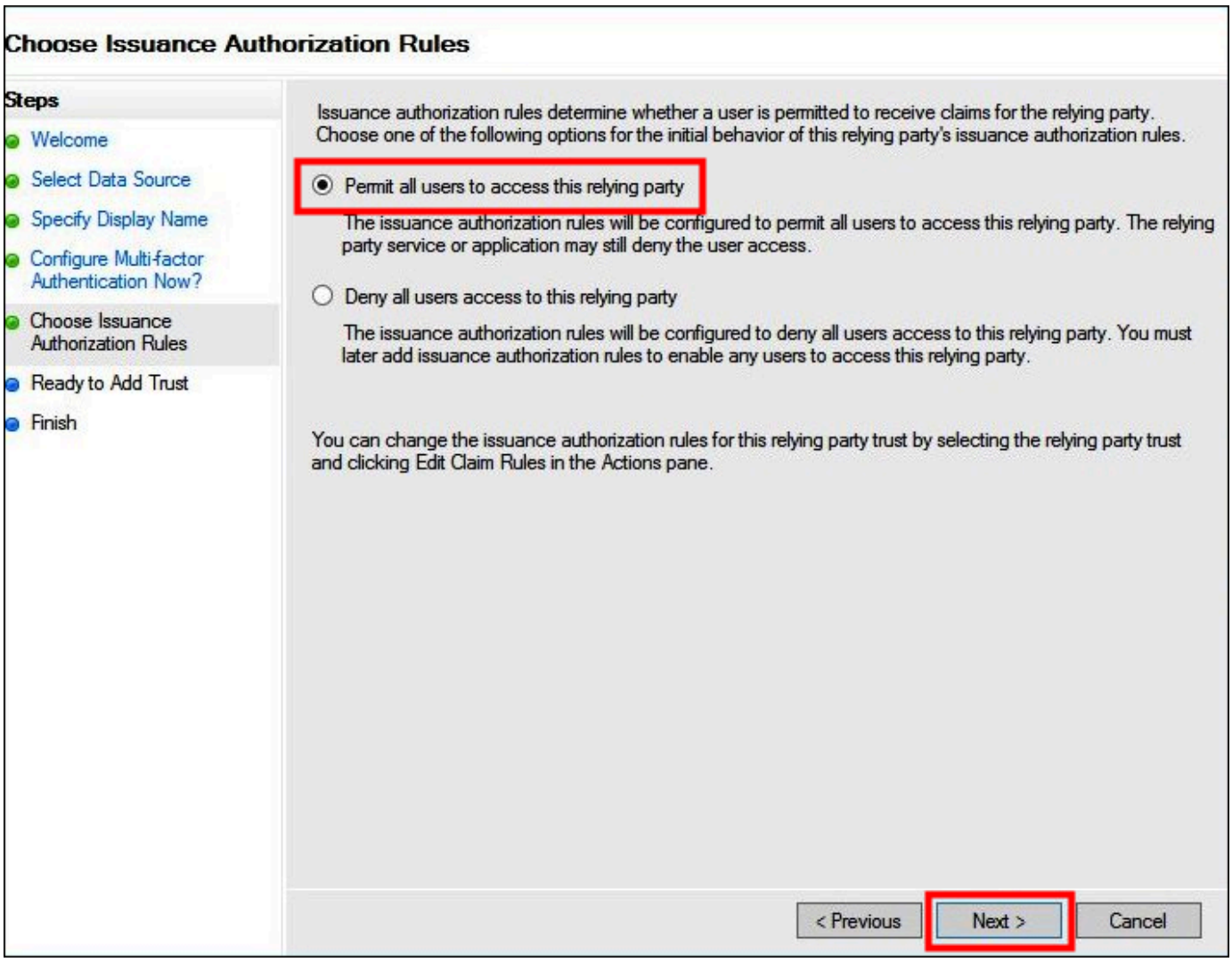
11. Na página Configure Multi-factor Authentication Now (Configurar autenticação multifator agora), este tutorial seleciona I do not want to configure multi-factor authentication for this relying party trust at this time (Não desejo configurar a autenticação multifator para esse objeto de confiança de terceira parte confiável no momento).

    Para obter mais segurança, recomendamos que você configure a autenticação multifator para ajudar a proteger seus recursos da AWS. Como é usado um conjunto de dados de amostra, este tutorial não habilita a autenticação multifator.

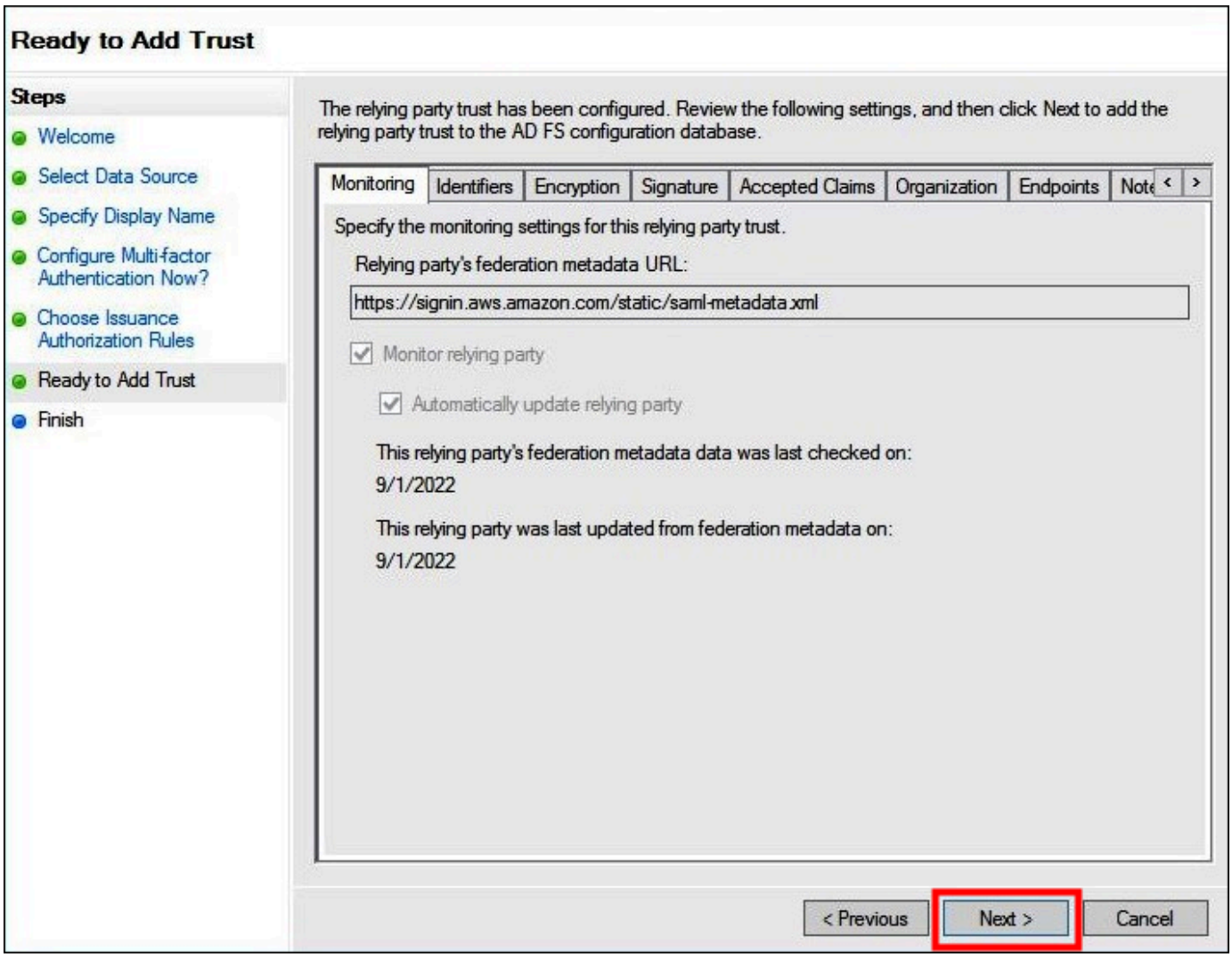


12. Escolha Próximo.
13. Na página Choose Issuance Authorization Rules (Escolher regras de autorização de emissão), selecione Permit all users to access this relying party (Permitir que todos os usuários acessem esta parte confiável).

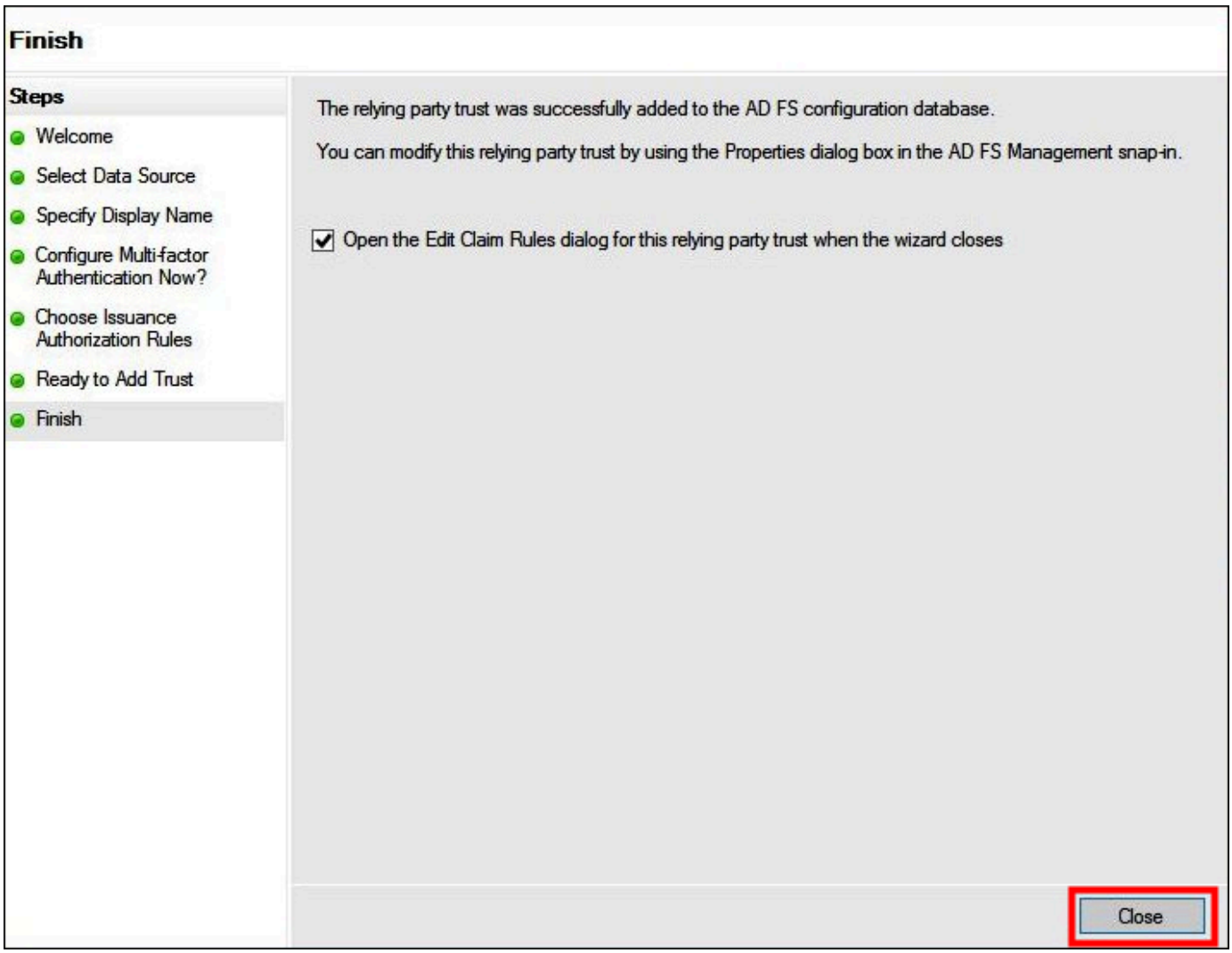
    Essa opção permite que todos os usuários no Active Directory usem o AD FS com a AWS como parte confiável. Você deve considerar seus requisitos de segurança e ajustar essa configuração em conformidade.



14. Escolha Próximo.
15. Na página Ready to Add Trust (Pronto para adicionar confiança), escolha Next (Avançar) para adicionar o objeto de confiança de terceira parte confiável ao banco de dados de configuração do AD FS.



16. Na página Finish (Concluir), escolha Close (Fechar).



Configuração de regras de declaração SAML para a parte confiável

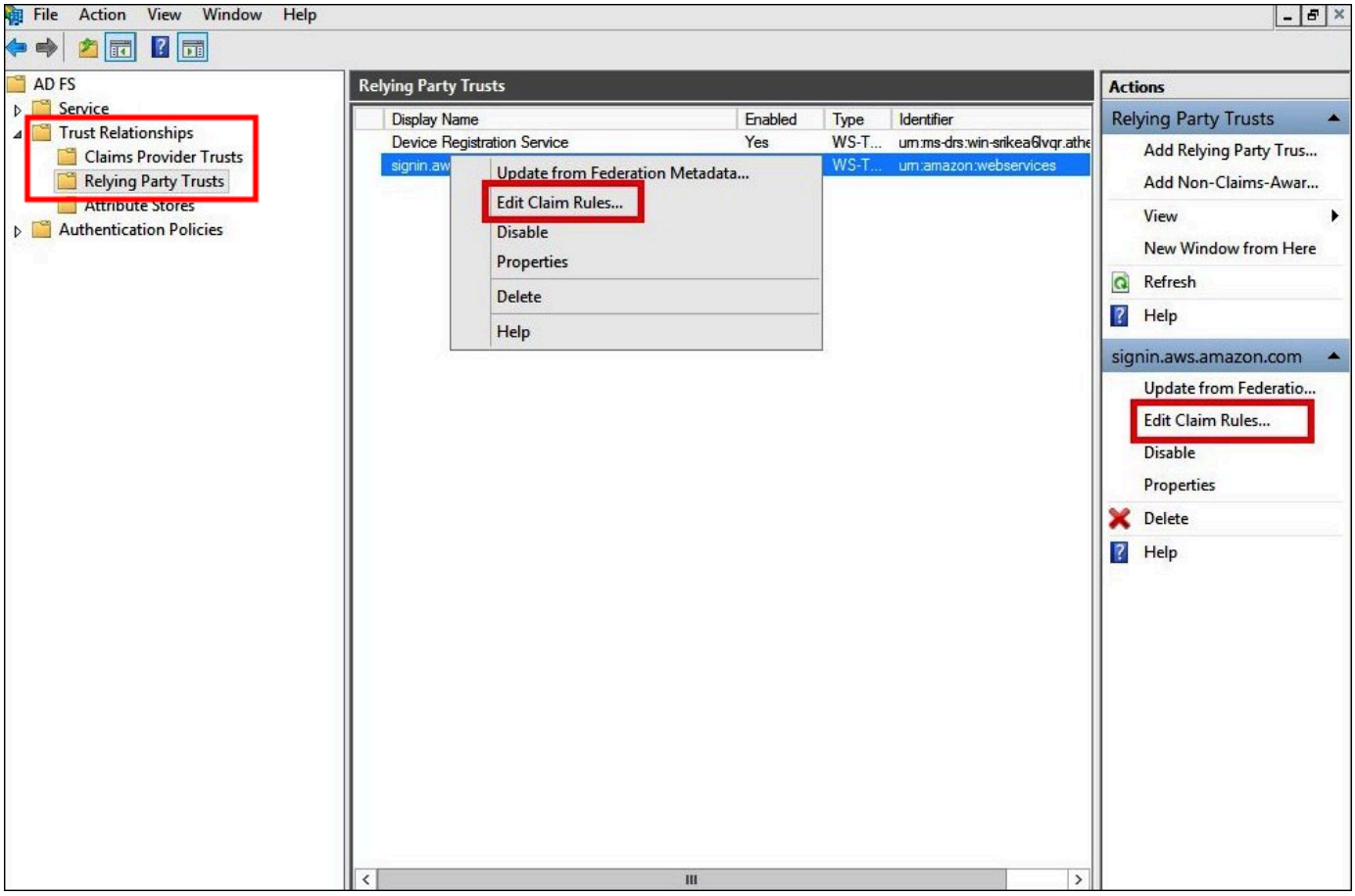
Nesta tarefa, você criará dois conjuntos de regras de declaração.

O primeiro conjunto, as regras de 1 a 4, contém regras de declaração do AD FS que são necessárias para assumir um perfil do IAM com base na associação ao grupo do AD. Essas são regras semelhantes as que você cria se desejar estabelecer acesso federado ao AWS Management Console.

O segundo conjunto, as regras 5 e 6, são regras de declaração necessárias para o controle de acesso do Athena.

Para criar regras de declaração do AD FS

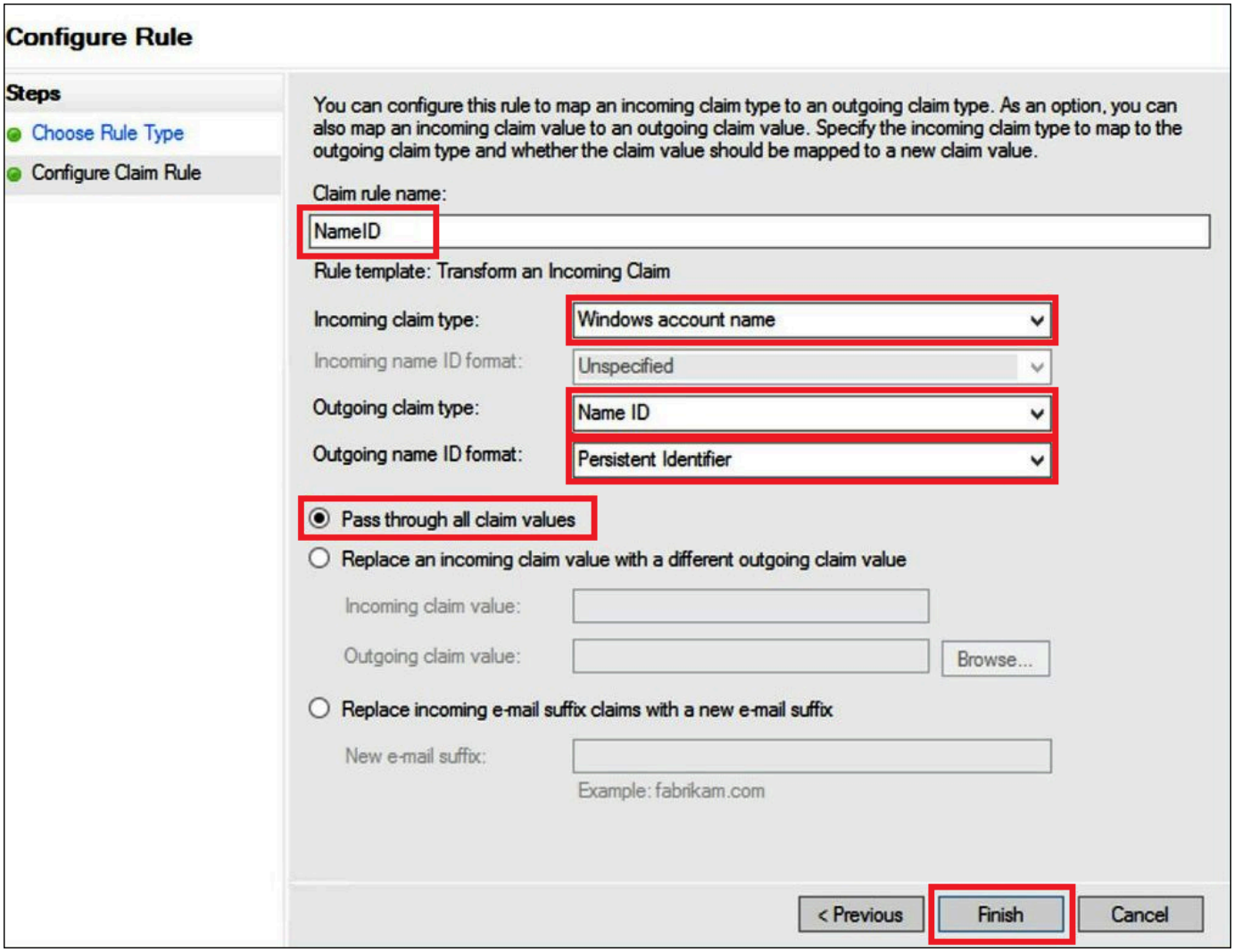
1. No painel de navegação do console AD FS Management, escolha Trust Relationships (Relações de confiança) e Relying Party Trusts (Objetos de confiança de terceira parte confiável).
2. Localize a parte confiável criada na seção anterior.
3. Clique com o botão direito do mouse na parte confiável e escolha Edit Claim Rules (Editar regras de declaração) ou escolha Edit Claim Rules (Editar regras de declaração) no menu Actions (Ações).



4. Escolha Add Rule.
5. Na página Configure Rule (Configurar regra) do Assistente para adição de uma regra de declaração de transformação, insira as informações a seguir para criar a regra de declaração 1 e, em seguida, escolha Finish (Concluir).

   - Em Claim rule name (Nome da regra de declaração), insira **`NameID`**.
   - Em Rule template (Modelo de regra), use Transform an Incoming Claim (Transformar uma declaração de entrada).

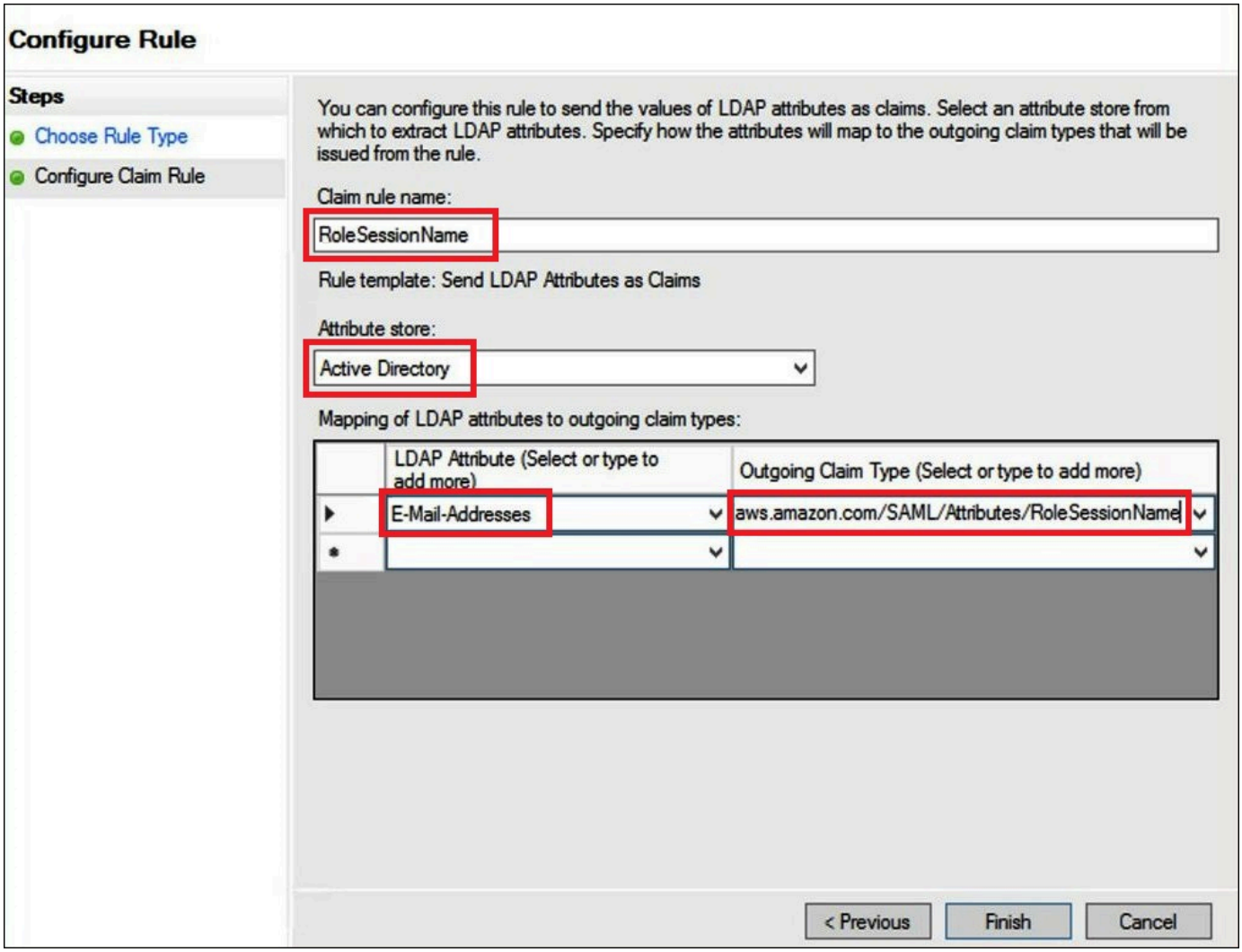
- Em Incoming claim type (Tipo da declaração de entrada), escolha Windows Account Name (Nome da conta do Windows).
- Em Outgoing claim type (Tipo da declaração de saída), escolha Name ID (ID do nome).
- Em Outgoing name ID format (Formato de ID de nome de saída), escolha Persistent Identifier (Identificador persistente).
- Selecione Pass through all claim values (Transmitir todos os valores de declaração).



6. Escolha Add Rule (Adicionar regra), insira as informações a seguir para criar a regra de declaração 2 e, em seguida, escolha Finish (Concluir).

   - Em Claim rule name (Nome da regra de declaração), insira **`RoleSessionName`**.
   - Em Rule template (Modelo de regra), use Send LDAP Attribute as Claims (Enviar atributo LDAP como declarações).

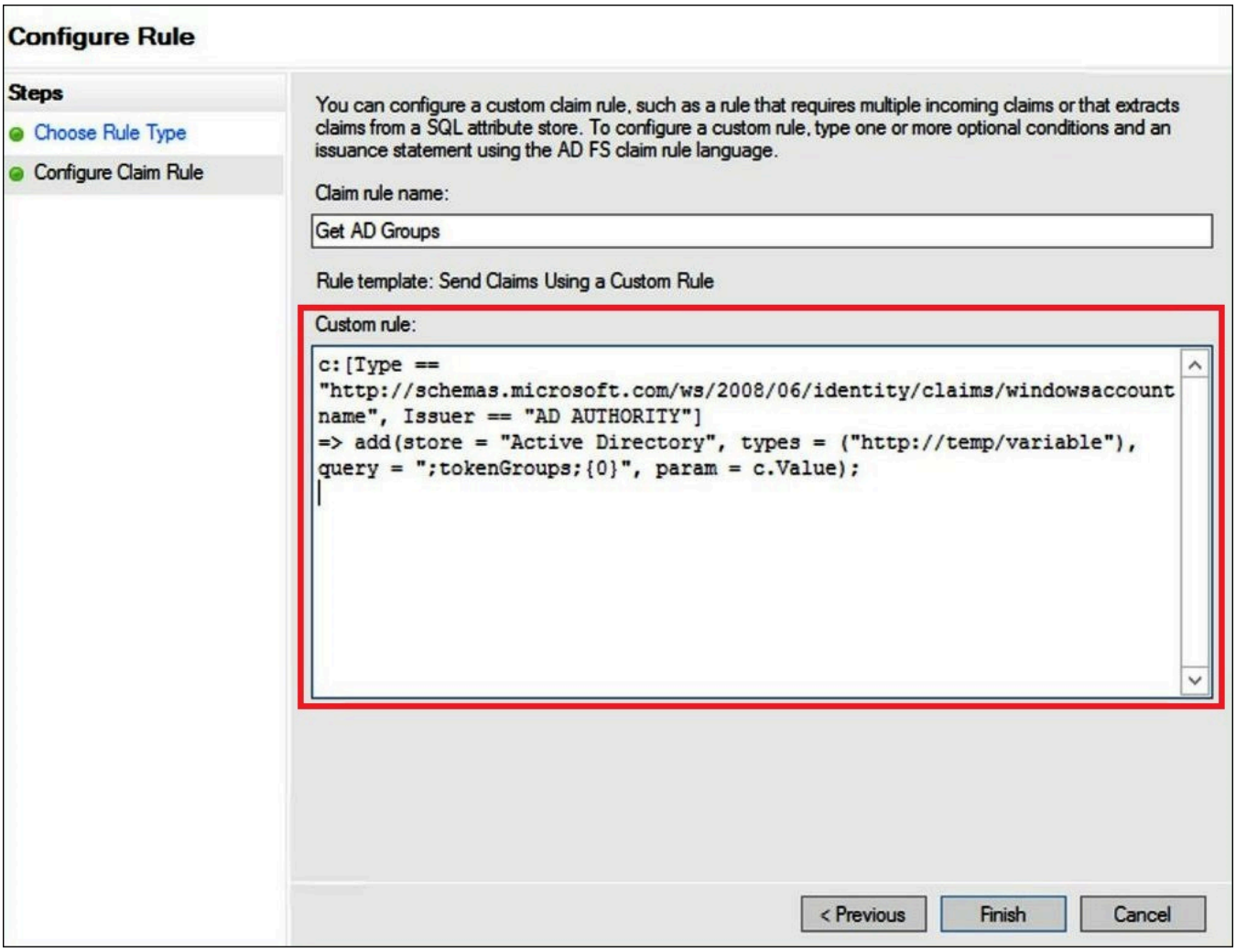
- Em Attribute store (Armazenamento de atributos), escolha Active Directory.
- Em Mapping of LDAP attributes to outgoing claim types (Mapeamento de atributos LDAP para tipos de declaração de saída), adicione o atributo **`E-Mail-Addresses`**. Em Outgoing Claim Type (Tipo da declaração de saída), insira **`https://aws.amazon.com/SAML/Attributes/RoleSessionName`**.



7. Escolha Add Rule (Adicionar regra), insira as informações a seguir para criar a regra de declaração 3 e, em seguida, escolha Finish (Concluir).

   - Em Claim rule name (Nome da regra de declaração), insira **`Get AD Groups`**.
   - Em Rule template (Modelo de regra), use Send Claims Using a Custom Rule (Enviar declarações usando uma regra personalizada).
   - Em Custom rule (Regra personalizada), insira o código a seguir:

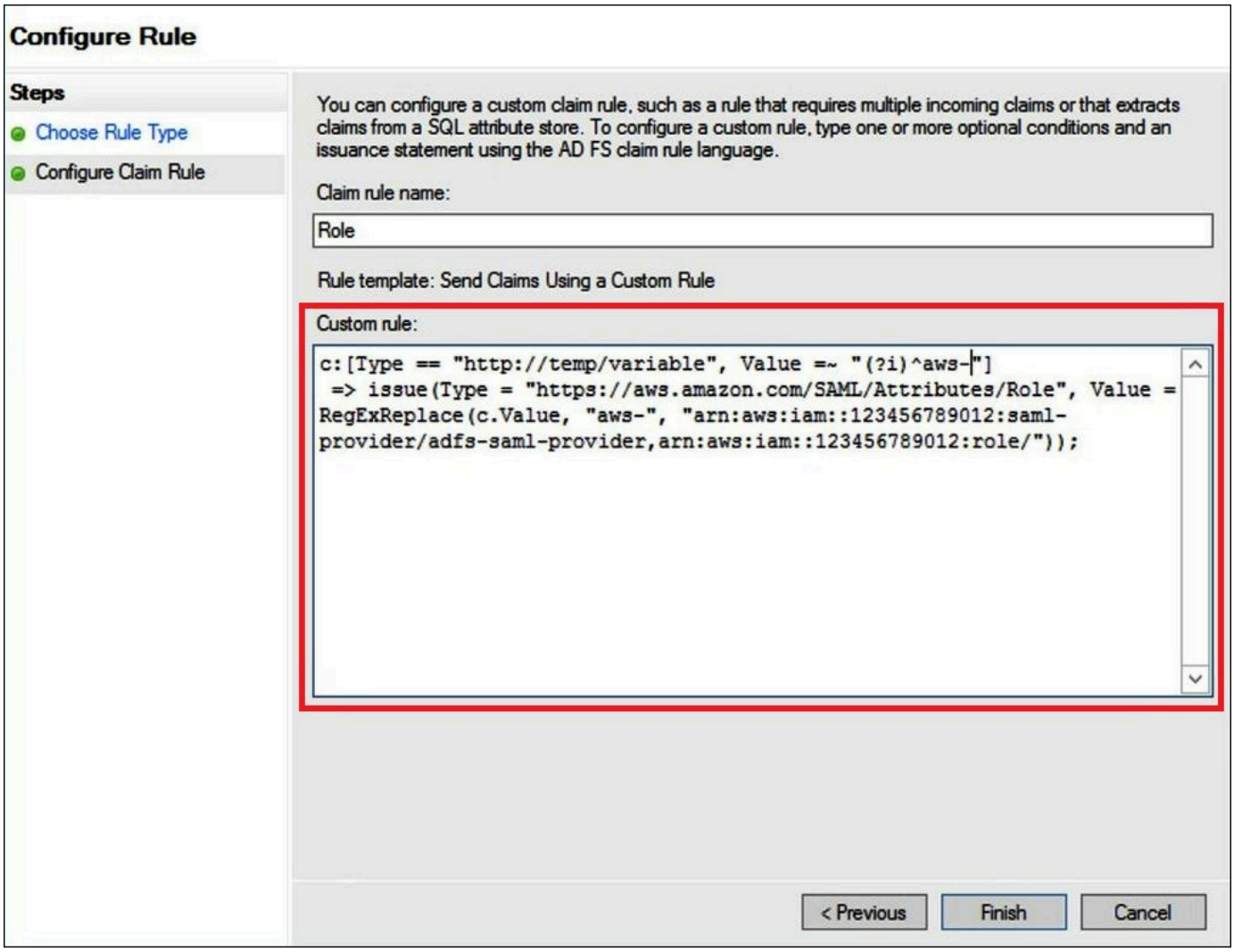
```
c:[Type == "http://schemas.microsoft.com/ws/2008/06/identity/claims/
windowsaccountname",
 Issuer == "AD AUTHORITY"]=> add(store = "Active Directory", types = ("http://
temp/variable"),
 query = ";tokenGroups;{0}", param = c.Value);
```



8. Escolha Add Rule. Insira as informações a seguir para criar a regra de declaração 4 e escolha Finish (Concluir).

   - Em Claim rule name (Nome da regra de declaração), insira **`Role`**.
   - Em Rule template (Modelo de regra), use Send Claims Using a Custom Rule (Enviar declarações usando uma regra personalizada).
   - Em Custom rule (Regra personalizada), insira o código a seguir com o número da sua conta e o nome do provedor SAML que você criou anteriormente:

```
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/"));
```

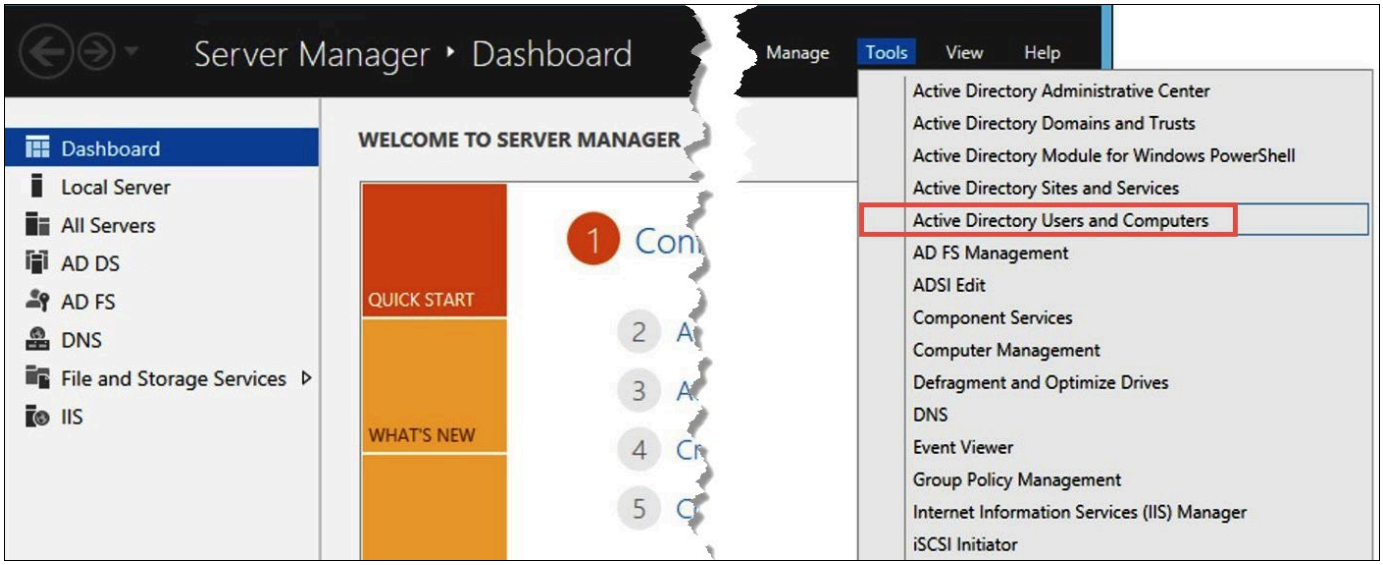


3. Criação de usuários e grupos do Active Directory

Agora está tudo pronto para que você crie usuários do AD que acessarão o Athena e grupos do AD para posicioná-los para que você possa controlar os níveis de acesso por grupo. Após a criação de grupos do AD que categorizam padrões de acesso a dados, você adicionará seus usuários a esses grupos.

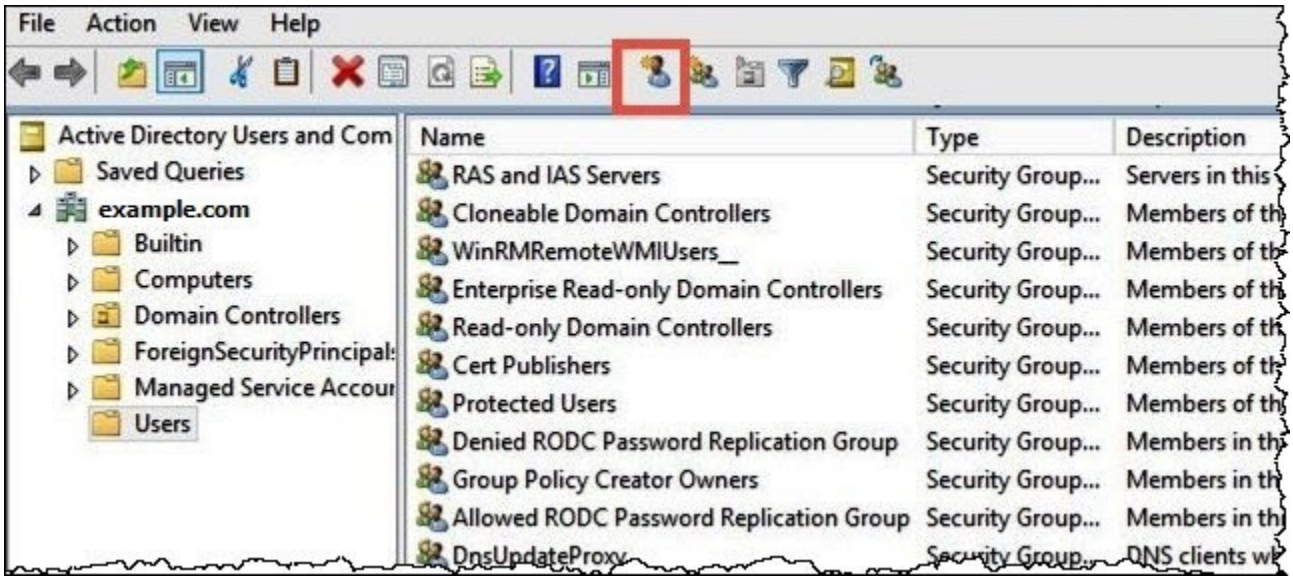
Para criar usuários do AD que acessem o Athena

1. No painel Gerenciador do servidor, escolha Tools (Ferramentas) e, em seguida, selecione Active Directory Users and Computers (Usuários e computadores do Active Directory).

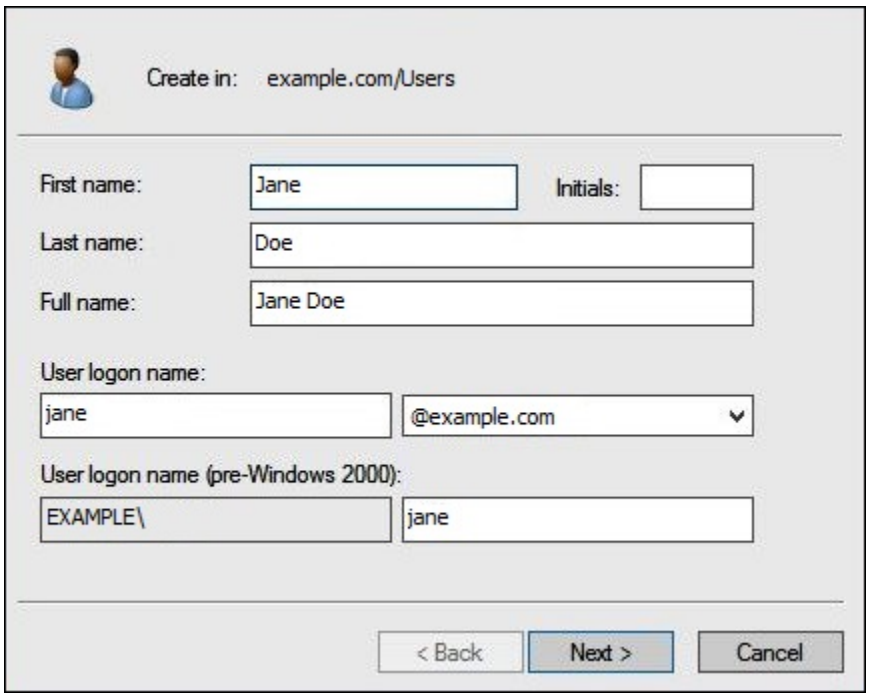


2. No painel de navegação, escolha Users.

3. Na barra de ferramentas Active Directory Users and Computers (Usuários e computadores do Active Directory), escolha a opção Create user (Criar usuário).

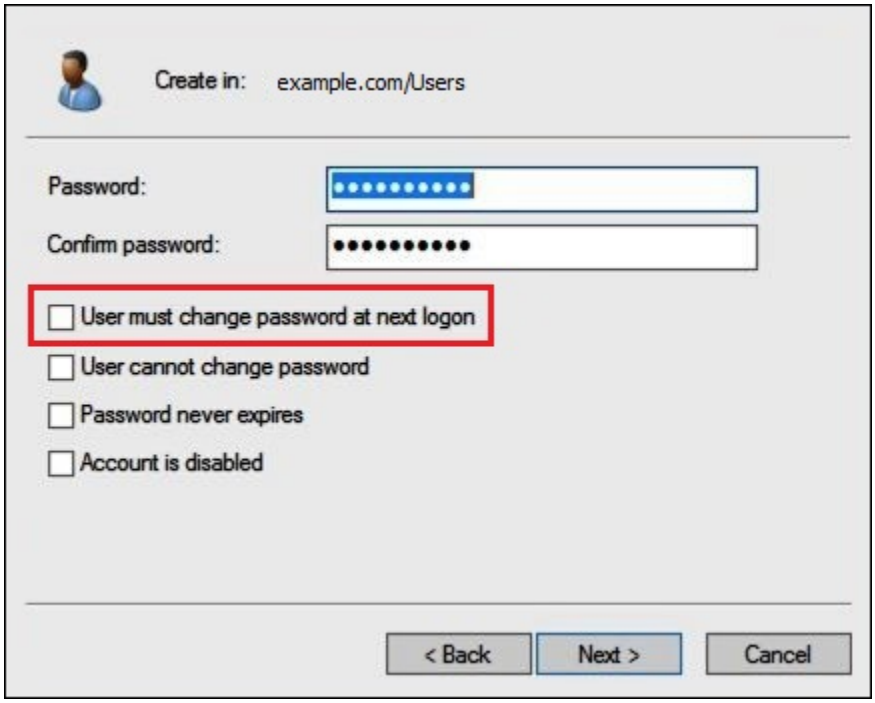


4. Na caixa de diálogo New Object – User (Novo objeto: usuário), em First name (Nome), Last name (Sobrenome) e Full name (Nome completo), insira um nome. Este tutorial usa **`Jane Doe`**.

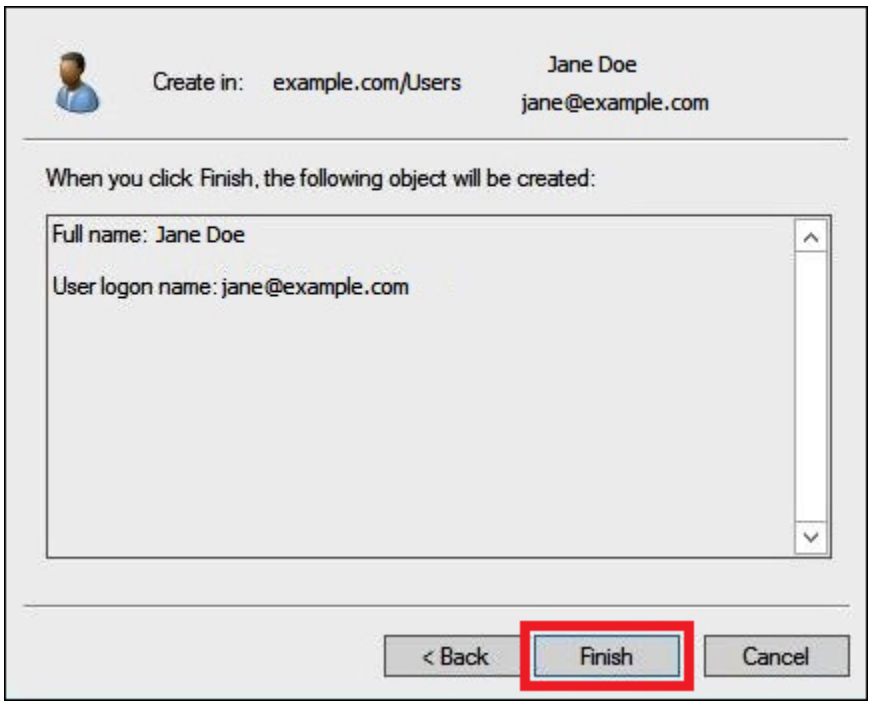


5. Escolha Próximo.
6. Em Password (Senha), insira uma senha e digite-a novamente para confirmar.

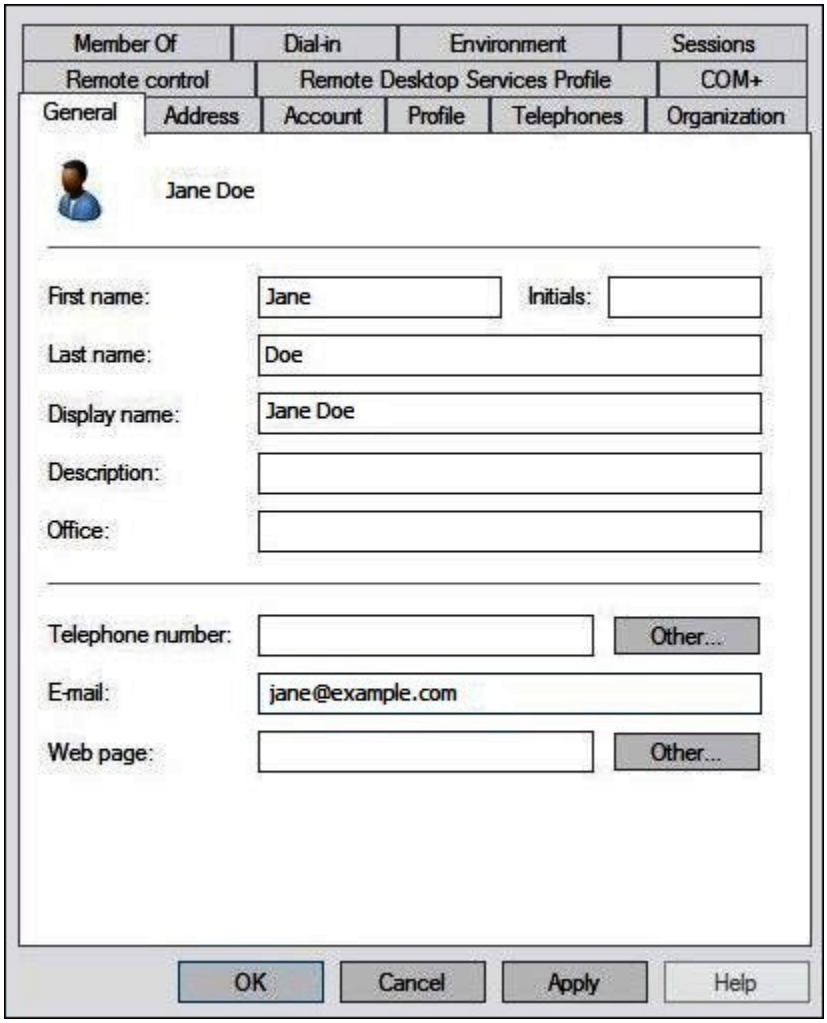
   Para simplificar, este tutorial desmarca User must change password at next sign on (O usuário deve alterar a senha no próximo login). Em cenários reais, você deve requerer que os usuários recém-criados alterem suas senhas.



7. Escolha Próximo.
8. Escolha Terminar.



9. Em Active Directory Users and Computers (Usuários e computadores do Active Directory), escolha o nome de usuário.
10. Na caixa de diálogo Properties (Propriedades) para o usuário, em E-mail, insira um endereço de e-mail. Este tutorial usa **`jane@example.com`**.



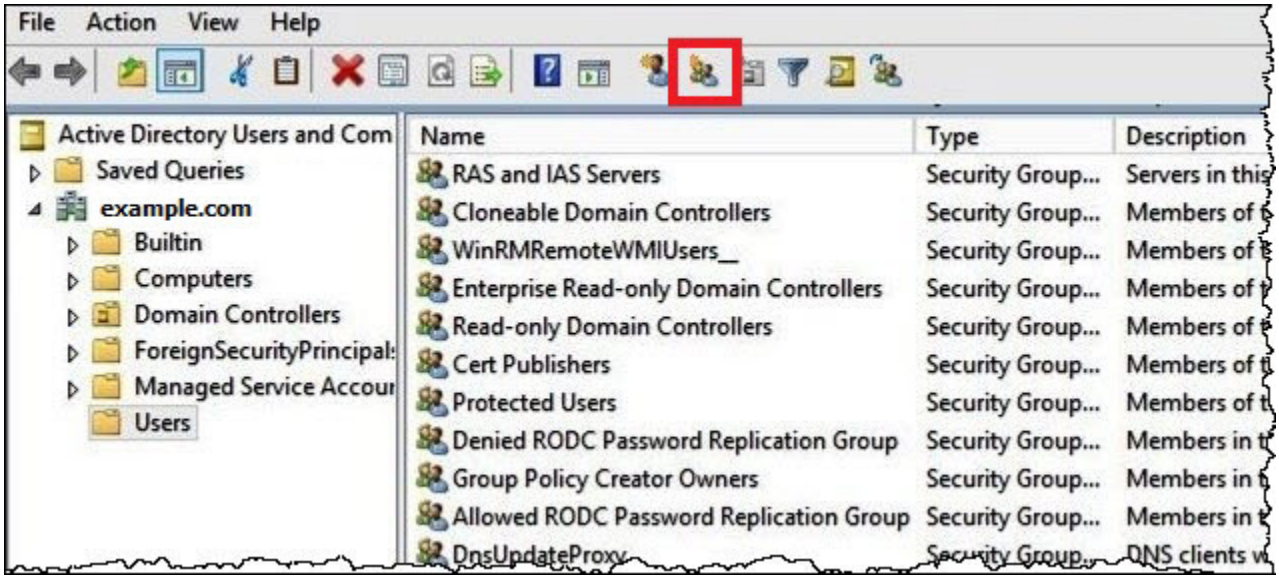
11. Escolha OK.

Criação de grupos do AD para representar padrões de acesso a dados

É possível criar grupos do AD cujos membros assumem o perfil do IAM `adfs-data-access` ao fazerem login na AWS. O exemplo a seguir criará um grupo AD chamado aws-adfs-data-access.

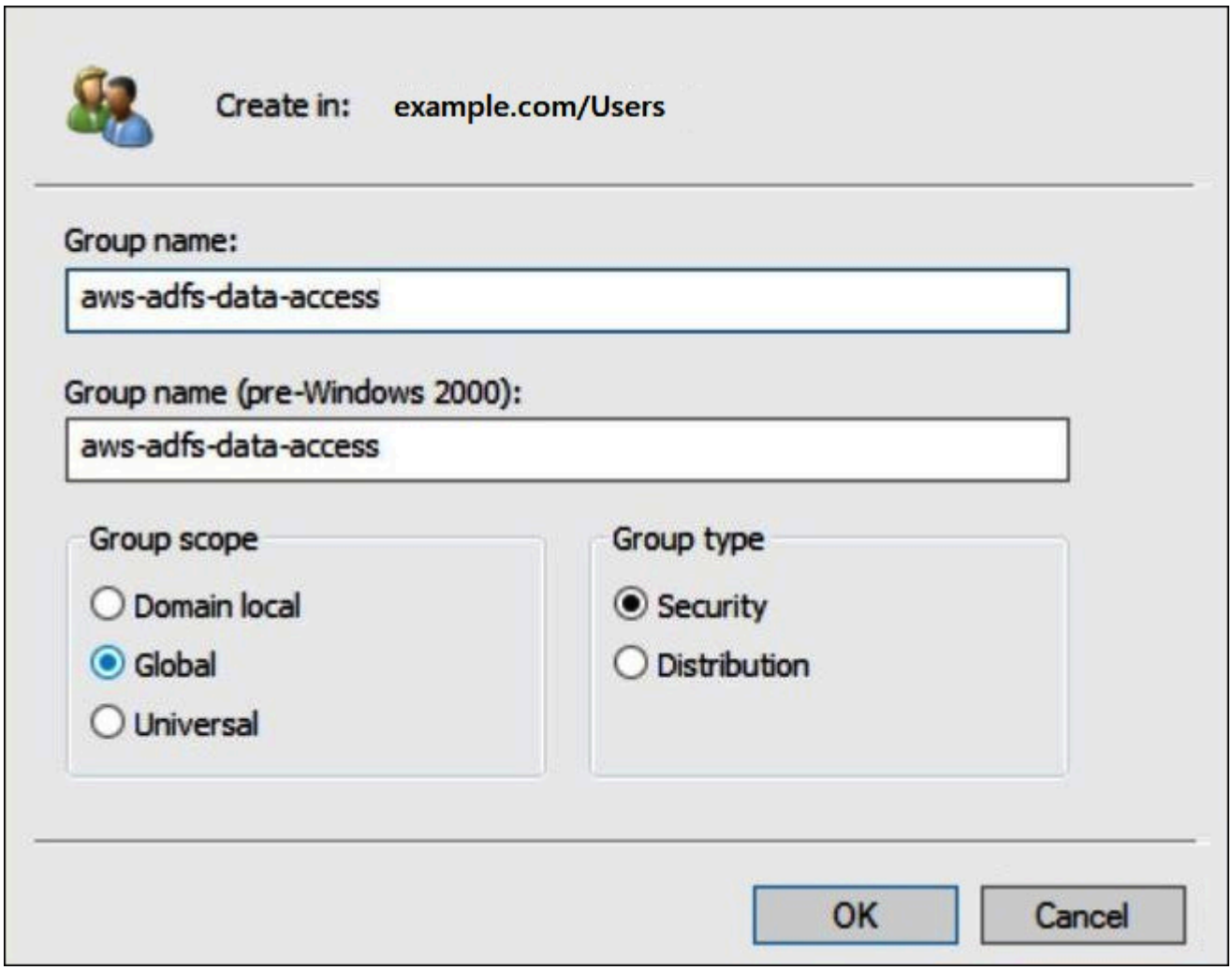
Para criar um grupo do AD

1. No Painel do gerenciador do servidor, no menu Tools (Ferramentas), escolha Active Directory Users and Computers (Usuários e computadores do Active Directory).
2. Na barra de ferramentas, escolha a opção Create new group (Criar novo grupo).



3. Na caixa de diálogo New Object - Group (Novo objeto: grupo), insira as informações a seguir:

   - Em Group name (Nome do grupo), digite **`aws-adfs-data-access`**.
   - Em Group scope (Escopo do grupo), selecione Global (Global).
   - Em Group type (Tipo de grupo), selecione Security (Segurança).



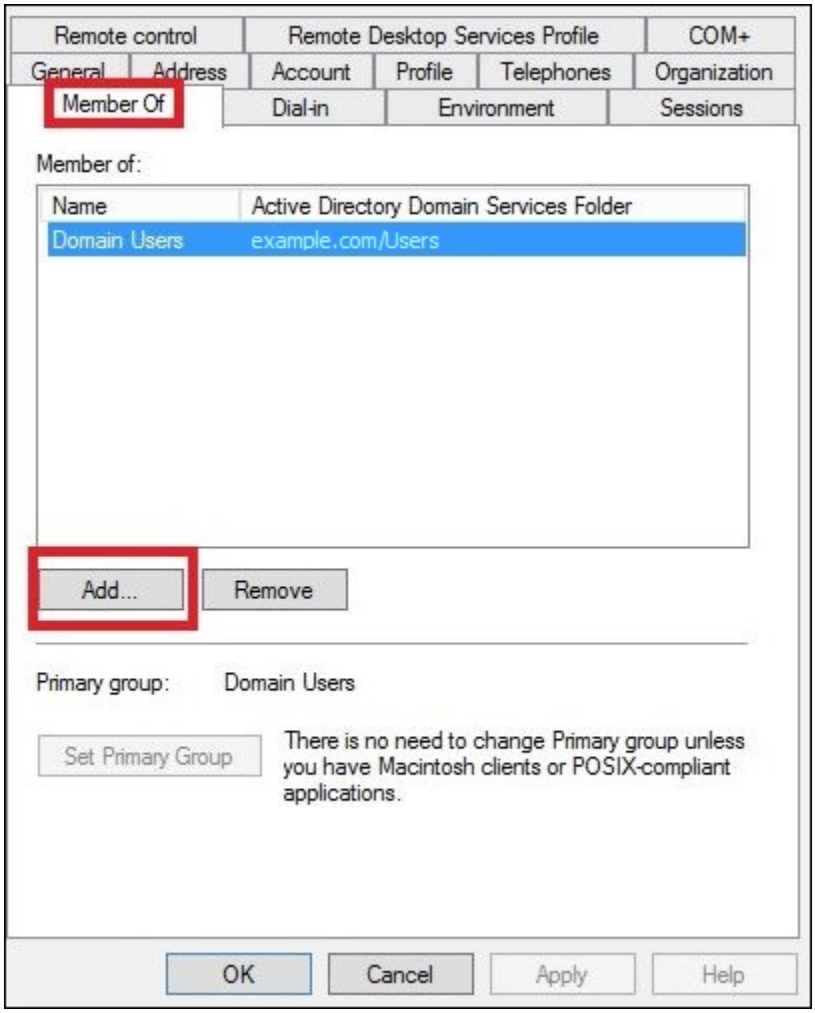
4. Escolha OK.

Adição de usuários do AD aos grupos apropriados

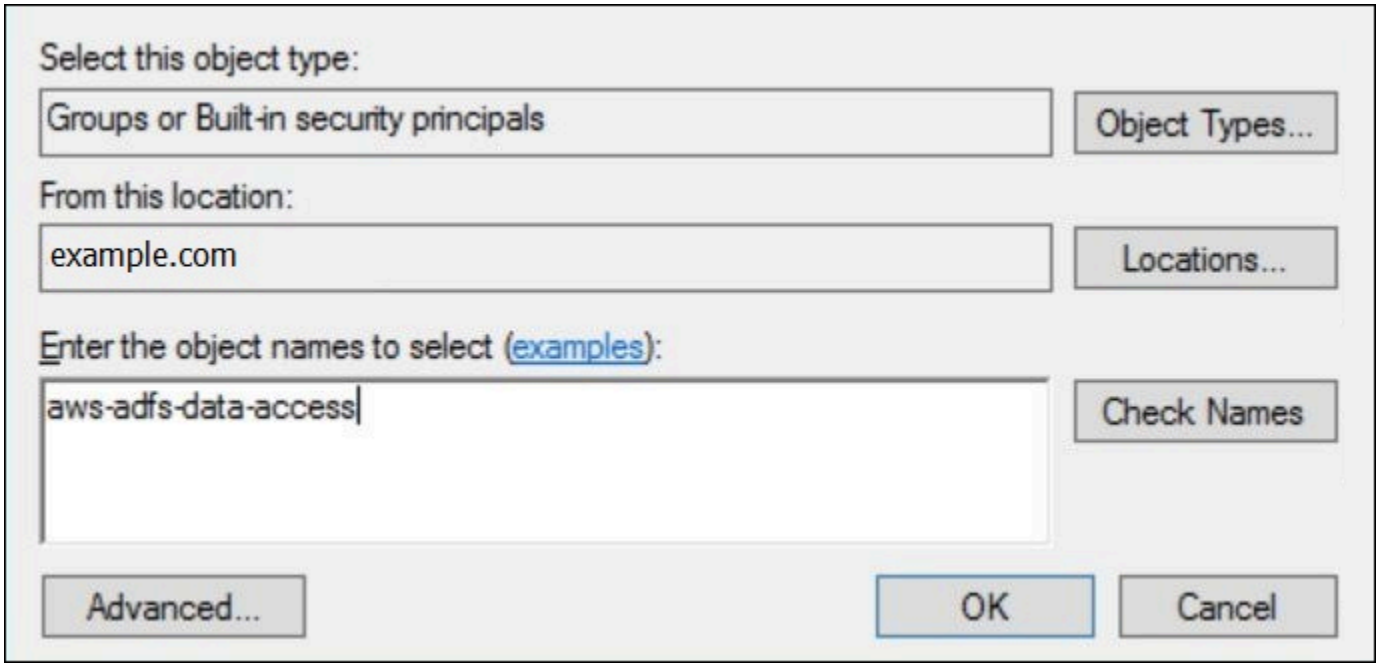
Agora que você criou um usuário AD e um grupo do AD, poderá adicionar o usuário ao grupo.

Para adicionar um usuário do AD a um grupo do AD

1. No Painel do gerenciador do servidor, no menu Tools (Ferramentas), escolha Active Directory Users and Computers (Usuários e computadores do Active Directory).
2. Em First name (Nome) e Last name (Sobrenome), escolha um usuário (por exemplo, Jane Doe).
3. Na caixa de diálogo Properties (Propriedades) para o usuário, na guia Member Of (Membro de), escolha Add (Adicionar).

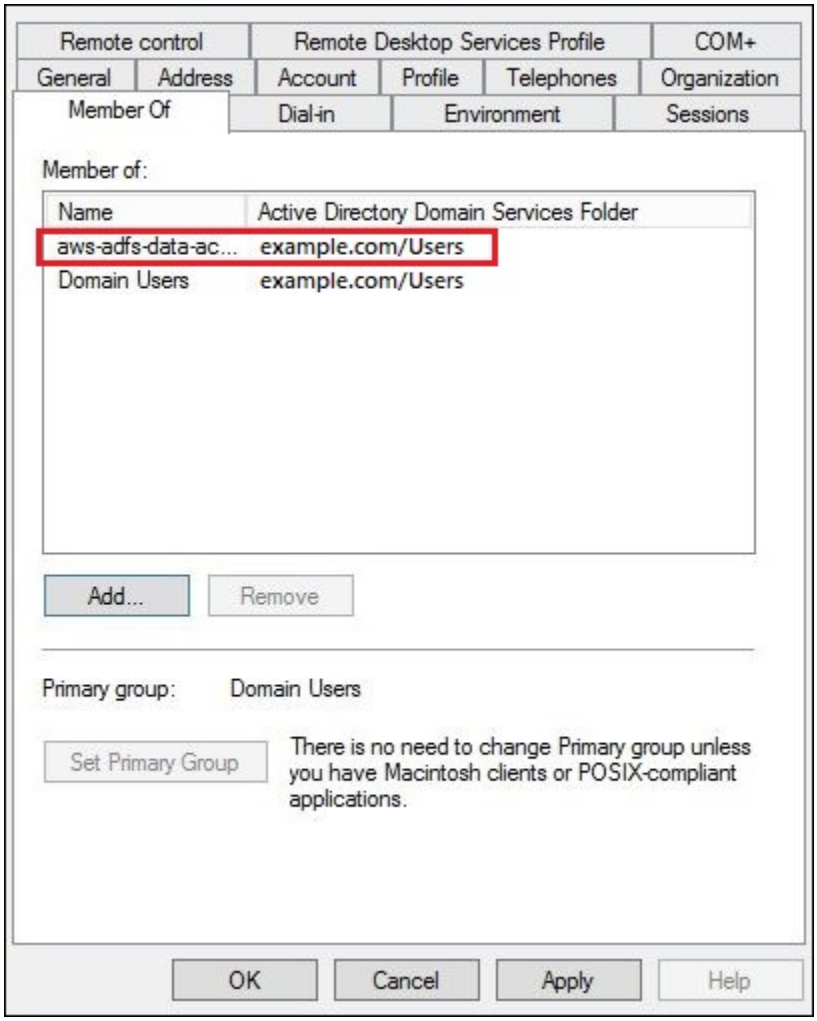


4. Adicione um ou mais grupos do AD FS de acordo com os seus requisitos. Este tutorial adiciona o grupo aws-adfs-data-access.
5. Na caixa de diálogo Select Groups (Selecionar grupos), em Enter the object names to select (Digitar os nomes dos objetos a serem selecionados), digite o nome do grupo AD FS que você criou (por exemplo, **`aws-adfs-data-access`**) e, em seguida, escolha Check Names (Verificar nomes).



6. Escolha OK.

   Na caixa de diálogo Properties (Propriedades) para o usuário, o nome do grupo do AD aparecerá na lista Member of (Membro de).



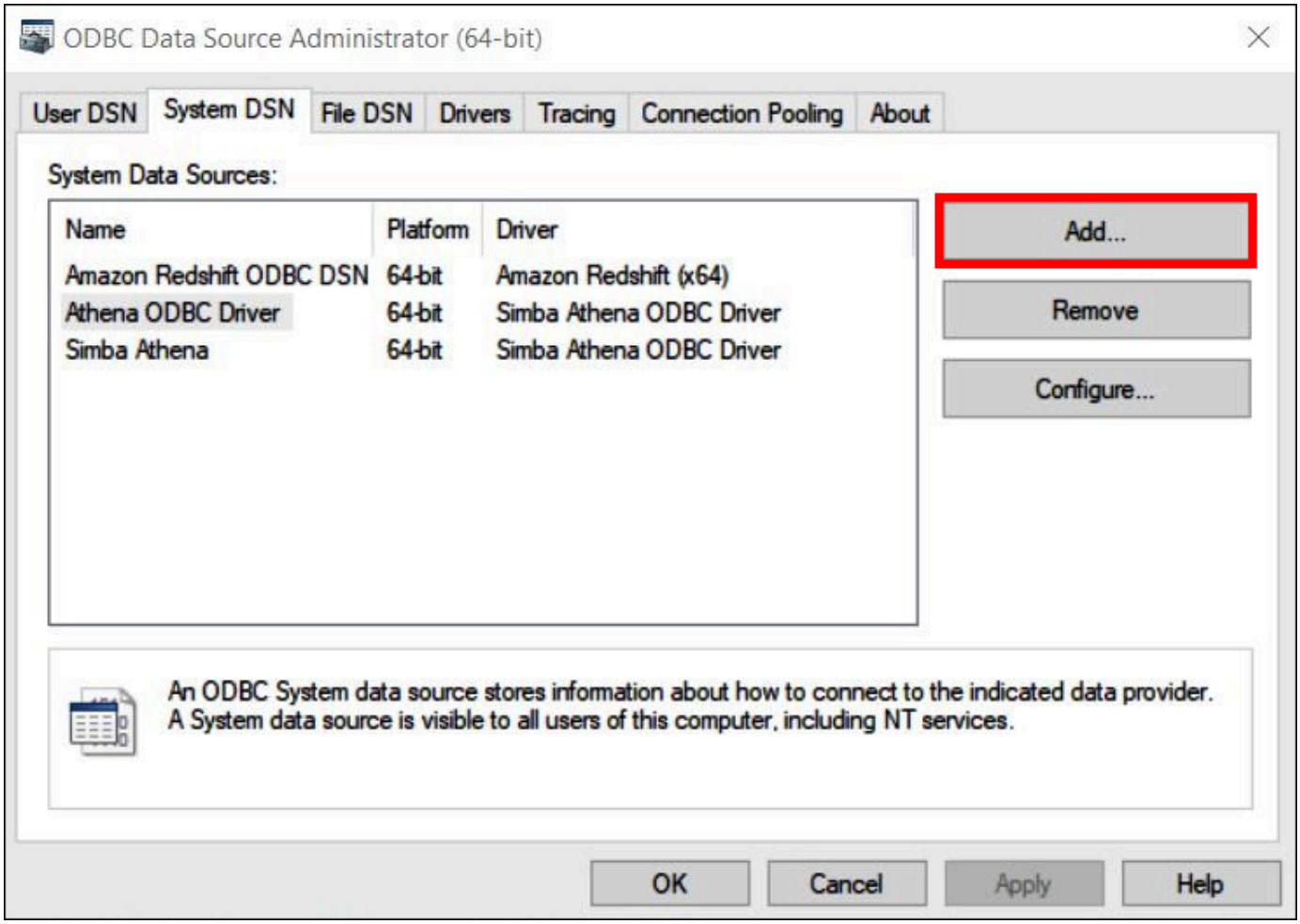
7. Escolha Apply (Aplicar) e, em seguida, escolha OK.

4. Configuração da conexão ODBC do AD FS para o Athena

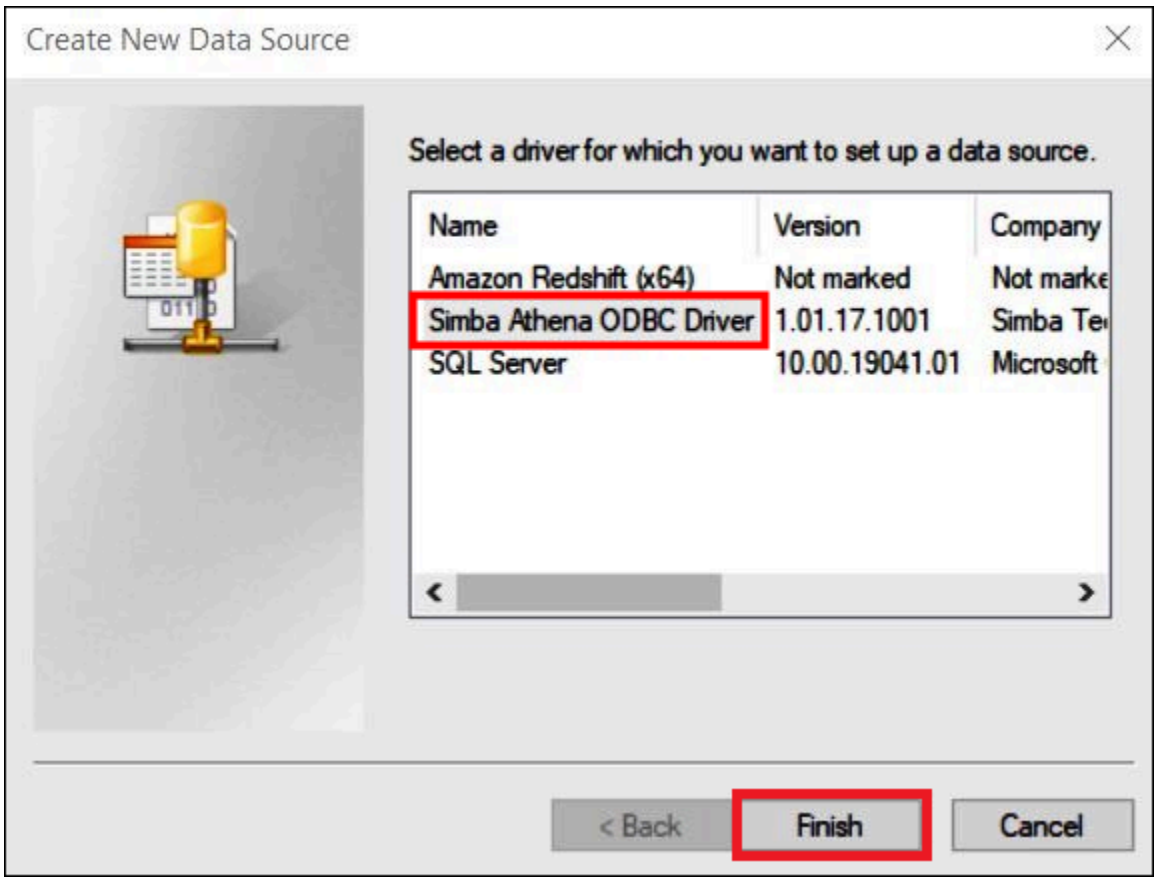
Após a criação dos usuários e grupos do AD, está tudo pronto para você usar o programa de fontes de dados do ODBC no Windows para configurar sua conexão ODBC do Athena para o AD FS.

Para configurar a conexão ODBC do AD FS para o Athena

1. Instale o driver ODBC para Athena. Para obter os links para baixar, consulte Conectar-se ao Amazon Athena com ODBC.
2. No Windows, escolha Start (Iniciar) e ODBC Data Sources (Fontes de dados do ODBC).
3. No programa ODBC Data Source Administrator, escolhaAdd (Adicionar).

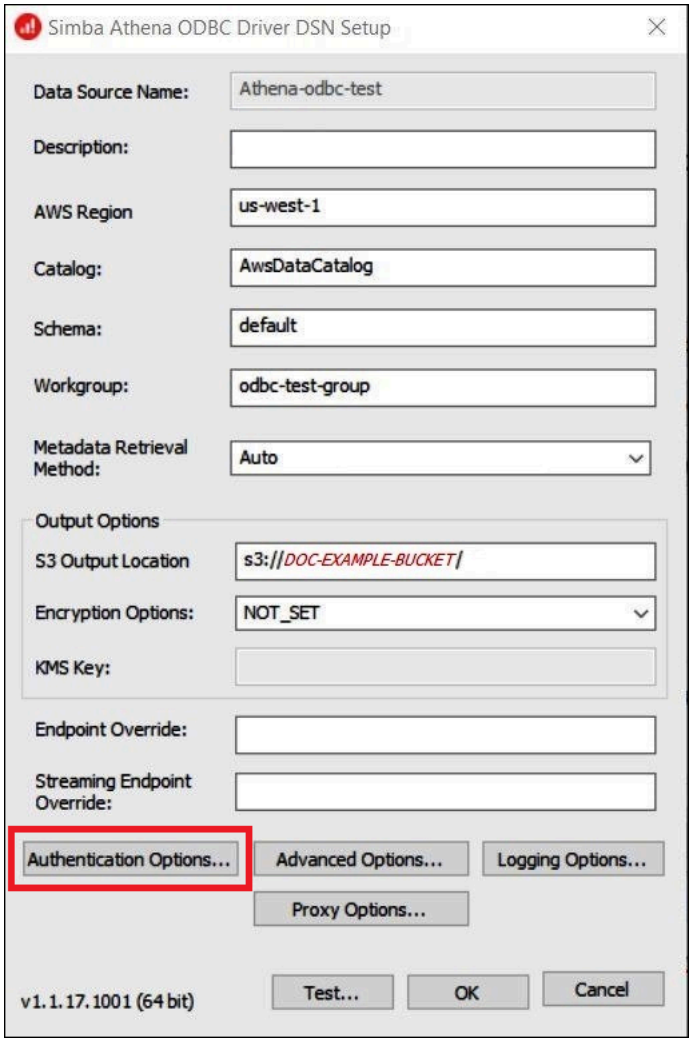


4. Na caixa de diálogo Create New Data Source (Criar nova fonte de dados), escolha Simba Athena ODBC Driver (Driver ODBC do Simba Athena) e, em seguida, selecione Finish (Concluir).



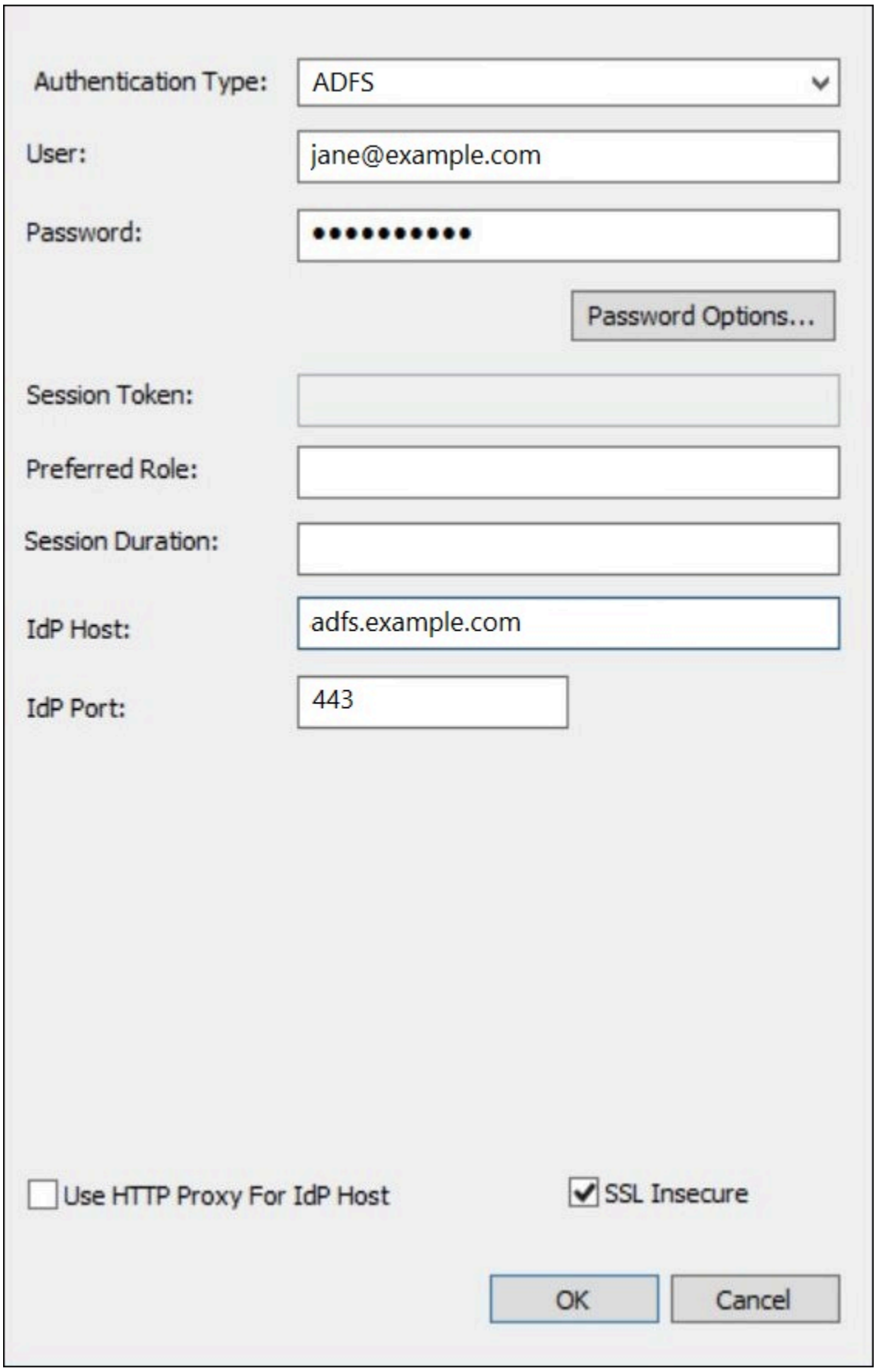
5. Na caixa de diálogo Simba Athena ODBC Driver DSN Setup (Configuração de DNS do driver ODBC do Simba Athena), insira os valores a seguir:

   - Para Data Source Name (Nome da fonte de dados), insira um nome para a fonte de dados (por exemplo, **`Athena-odbc-test`**).
   - Em Description (Descrição), insira uma descrição para a origem dos dados.
   - Para Região da AWS, insira a Região da AWS que você está usando (por exemplo, **`us-west-1`**).
   - Para S3 Output Location, (Local de saída do S3), insira o caminho do Amazon S3 onde deseja que sua saída seja armazenada.



6. Escolha Authentication Options (Opções de autenticação).
7. Na caixa de diálogo Authentication Options (Opções de autenticação), especifique os valores a seguir:

   - Em Authentication Type (Tipo de autenticação), escolha ADFS.
   - Em User (Usuário), insira o endereço de e-mail do usuário (por exemplo, **`jane@example.com`**).
   - Em Password (Senha), insira a senha do ADFS do usuário.
   - Em IdP Host (Host do IdP), insira o nome do servidor do AD FS (por exemplo,**`adfs.example.com`**).
   - Em IdP Port (Porta do IdP), use o valor padrão 443.
   - Selecione a opção SSL Insecure.


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. Escolha OK para fechar as Authentication Options (Opções de autenticação).
9. Selecione Test (Testar) para testar a conexão ou OK para finalizar.

Configurar o SSO para ODBC usando o plugin Okta e o Okta Identity Provider

Esta página descreve como configurar o driver ODBC do Amazon Athena e o plugin Okta para adicionar o recurso de logon único (SSO) usando o provedor de identidade Okta.

Pré-requisitos

A conclusão das etapas neste tutorial requer o seguinte:

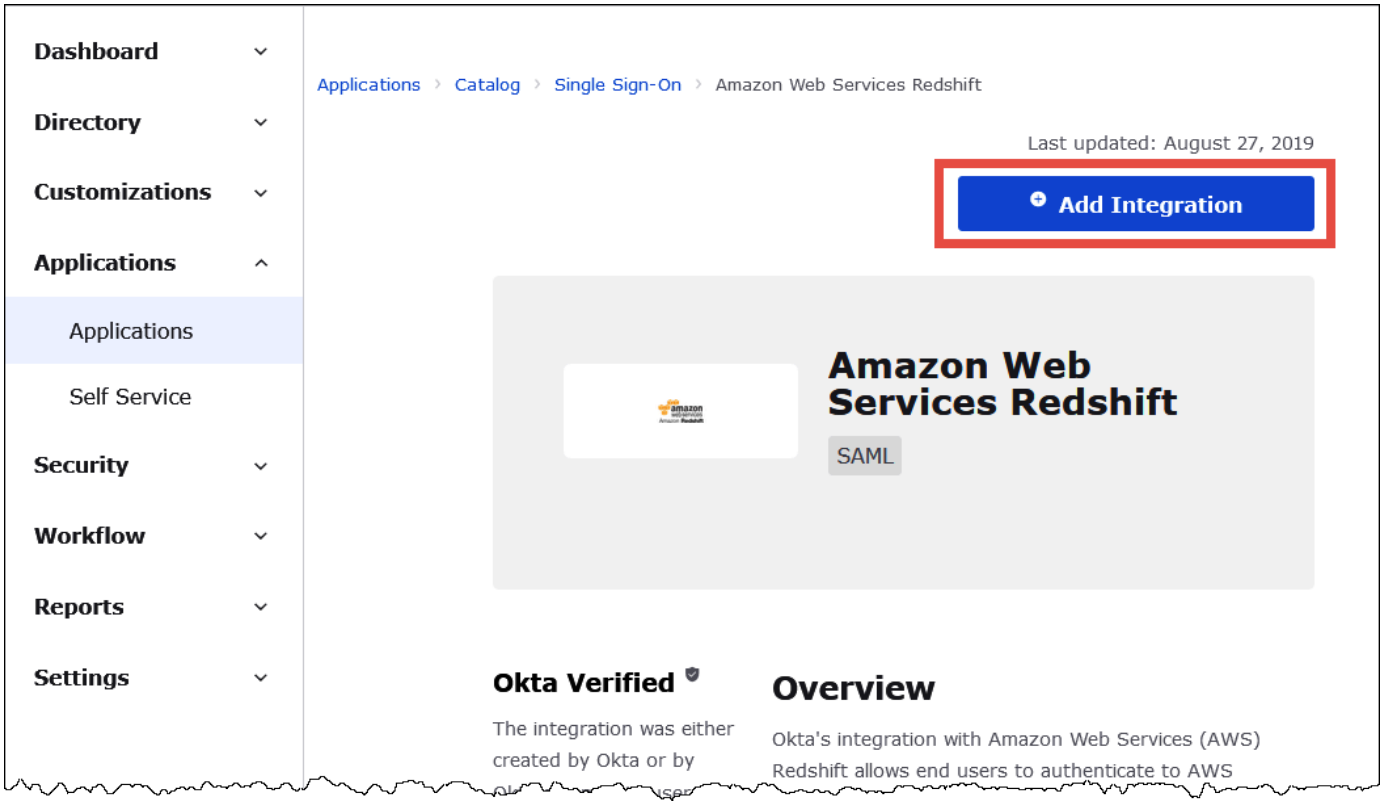
- Driver ODBC do Amazon Athena. Para obter os links para baixar, consulte Conectar-se ao Amazon Athena com ODBC.
- Uma função do IAM que você deseje usar com o SAML. Para obter mais informações, consulte Criar uma função para a federação SAML 2.0 no Guia do usuário do IAM.
- Uma conta do Okta. Para obter informações, acesse Okta.com.

Criar uma integração de aplicações no Okta

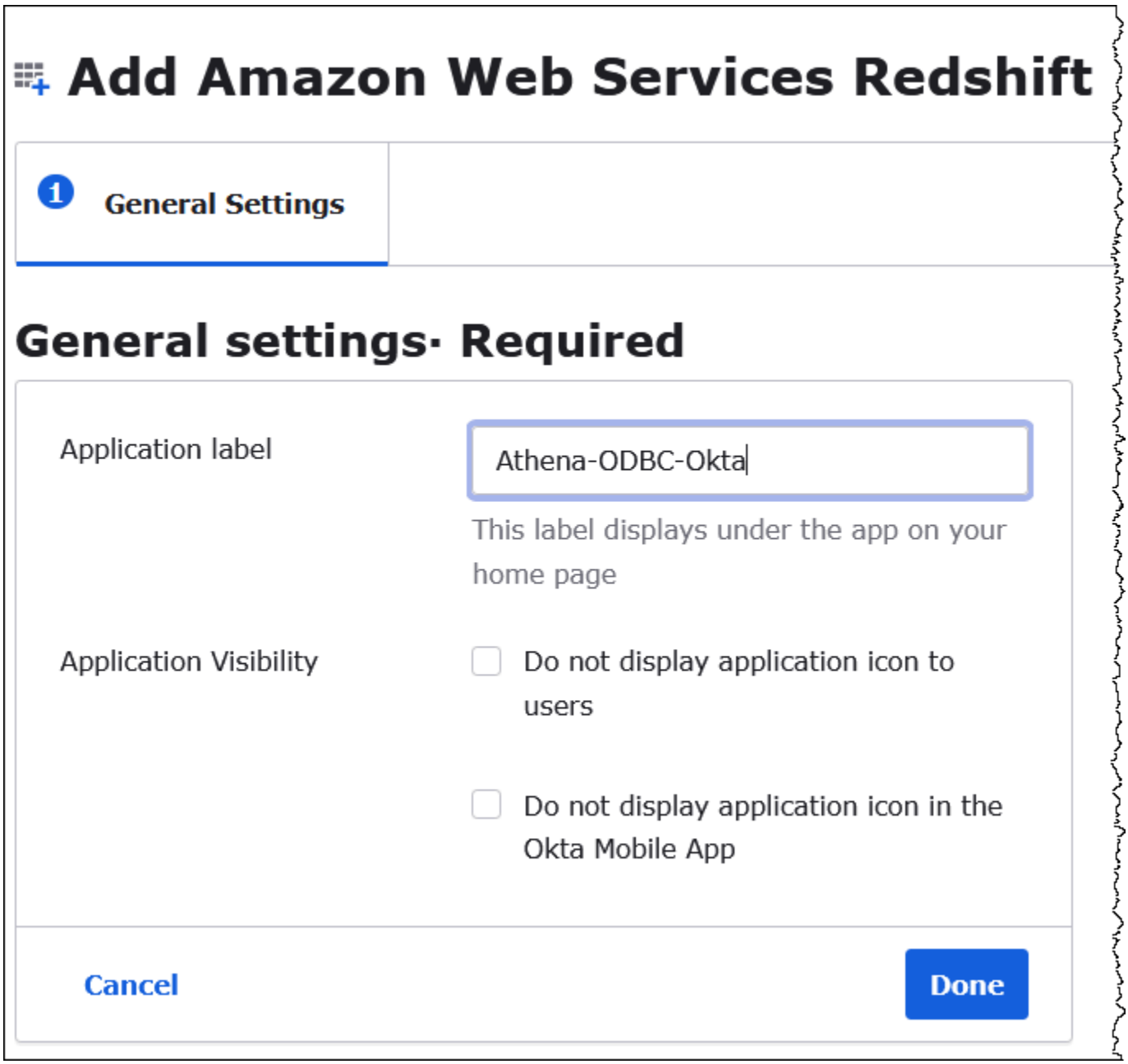
Primeiro, use o painel do Okta para criar e configurar uma aplicação SAML 2.0 para autenticação única no Athena. Você pode usar uma aplicação Redshift existente no Okta para configurar o acesso ao Athena.

Criar uma integração de aplicações no Okta

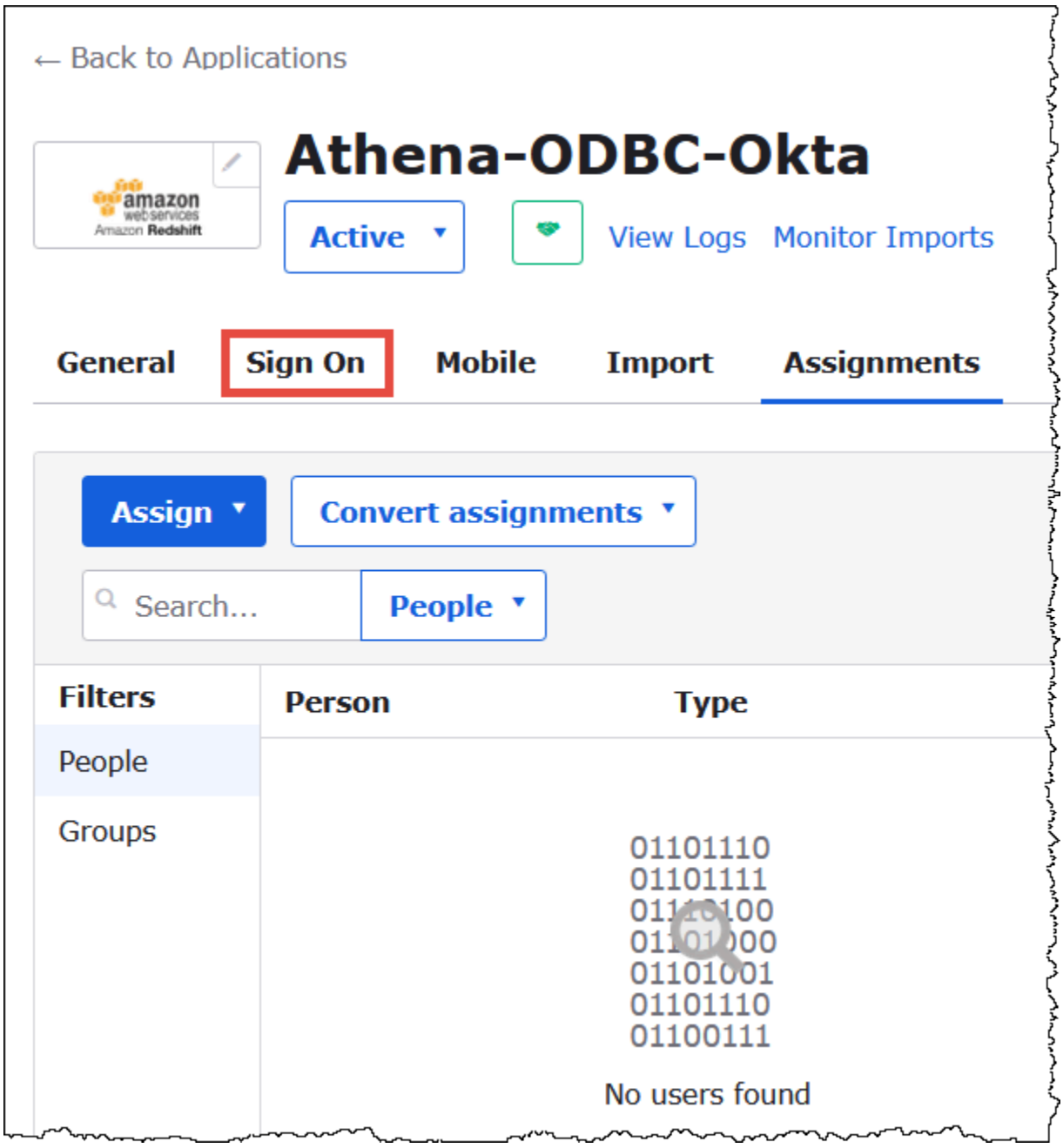
1. Faça login na página de administração da conta em Okta.com.
2. No painel de navegação, escolha Applications (Aplicações), Applications (Aplicações).
3. Na página Applications (Aplicações), escolha Browse App Catalog (Navegar pelo App Catalog).
4. Na página Browse App Integration Catalog (Navegar pelo catálogo de integração de aplicações), na seção Use Case (Caso de uso), escolha All Integrations (Todas as integrações).
5. Na caixa de pesquisa, insira Amazon Web Services Redshift e escolha Amazon Web Services Redshift SAML.
6. Escolha Add Integration (Adicionar integração).



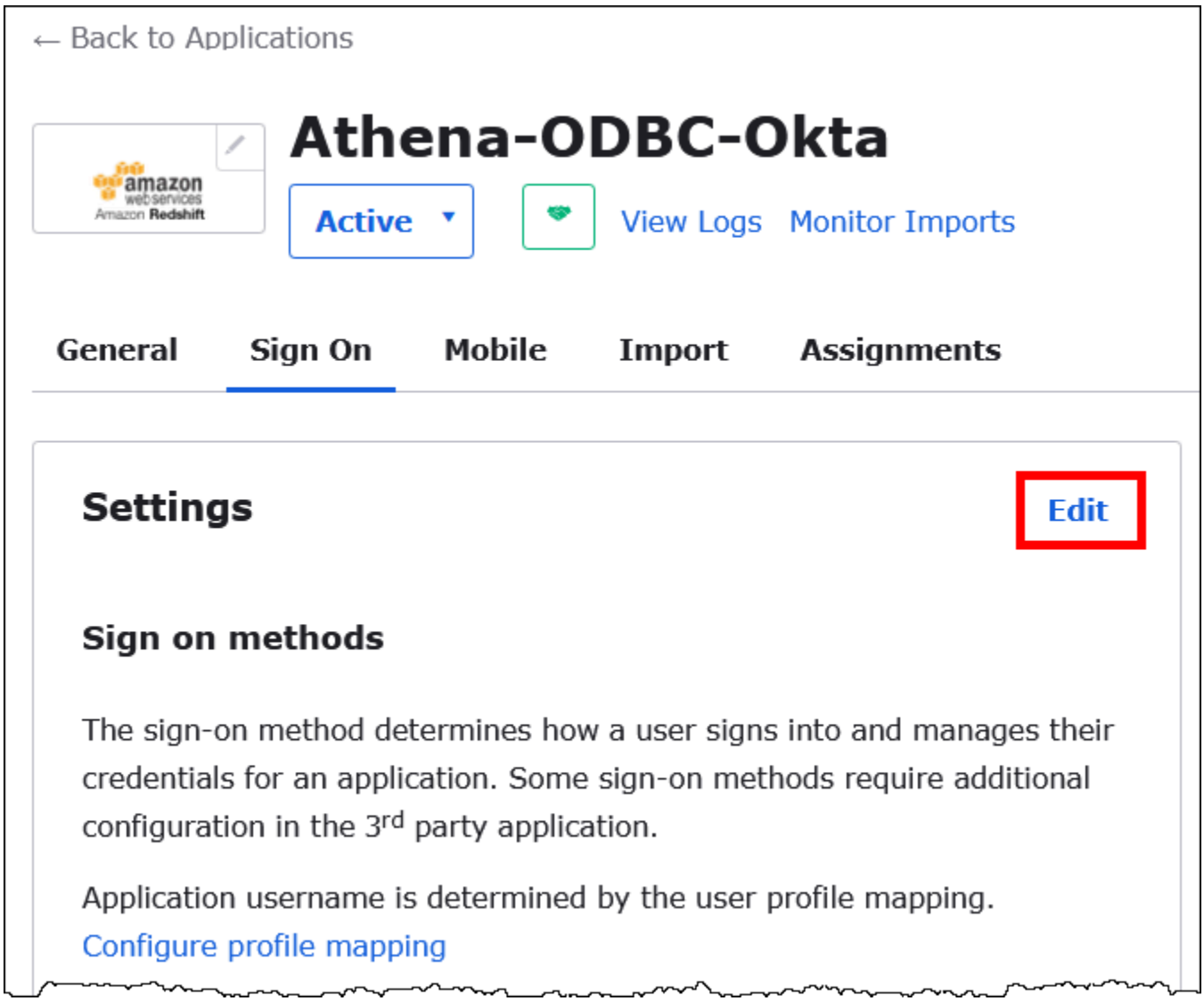
7. Na seção General Settings Required (Configurações gerais obrigatórias), em Application label (Rótulo da aplicação), insira um nome para a aplicação. Este tutorial usa o nome Athena-ODBC-Okta.



8. Selecione Done (Concluído).
9. Na página da aplicação no Okta (por exemplo, Atena-ODBC-Okta), escolha Sign On (Fazer logon).



10. Na seção Settings (Configurações), escolha Edit (Editar).



11. Na seção Advanced Sign-on Settings (Configurações avançadas de logon), configure os valores a seguir.

    - Em IdP ARN and Role ARN (ARN do IdP e ARN do perfil), insira o ARN do IDP da AWS e o ARN do perfil como valores separados por vírgulas. Para obter mais informações sobre o formato da função do IAM, consulte Configurar asserções SAML para a resposta de autenticação no Guia do usuário do IAM.
    - Em Session Duration (Duração da sessão), insira um valor entre 900 e 43.200 segundos. Este tutorial usa o padrão de 3.600 (uma hora).

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

O Athena não utiliza as configurações DbUser Format (Formato DbUser), AutoCreate e Allowed DBGroups (DBGroups permitidos). Não é necessário configurá-las.

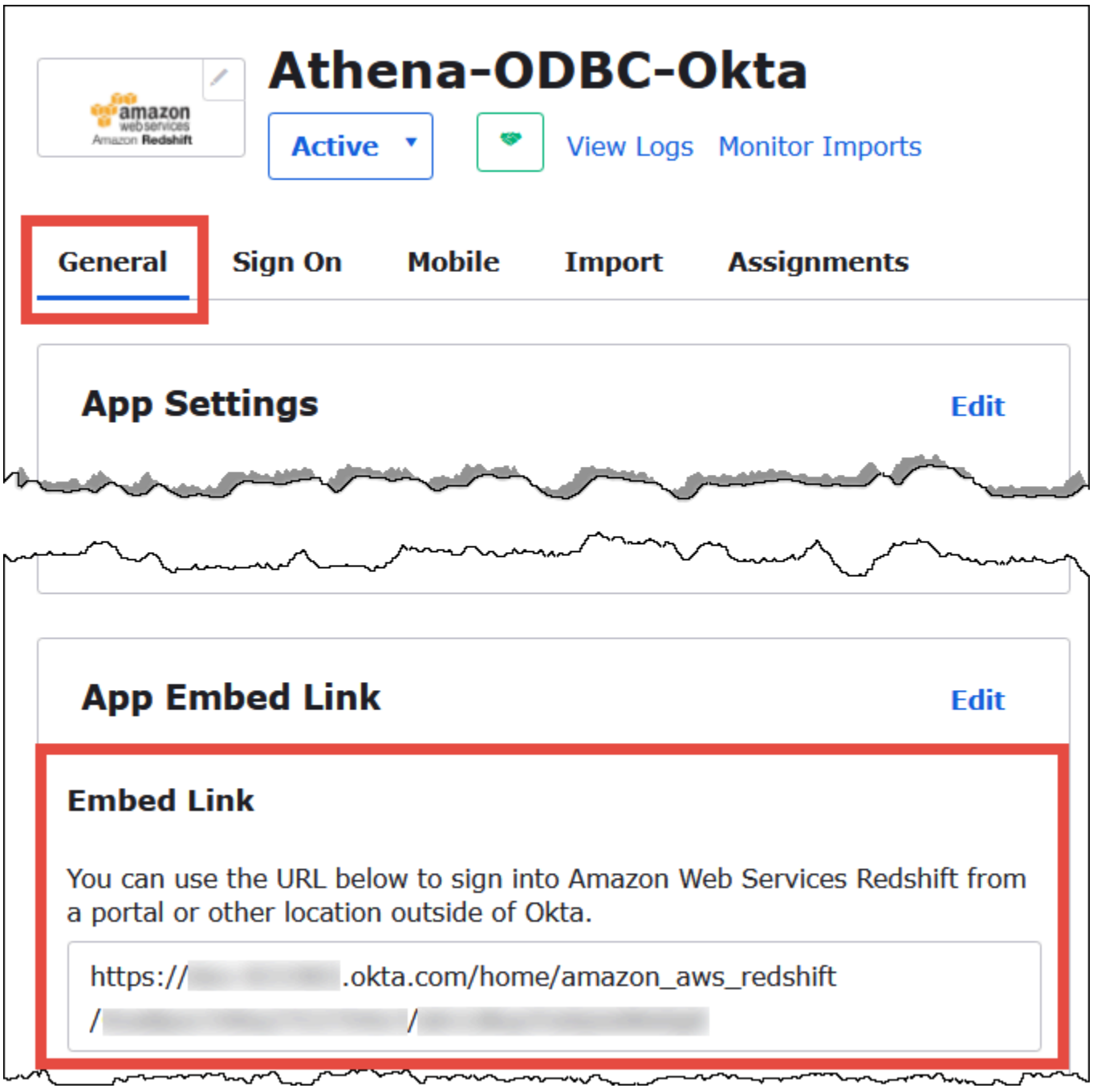
12. Escolha Salvar.

Recuperar informações de configuração de ODBC do Okta

Agora que você criou a aplicação no Okta, já pode recuperar o ID da aplicação e o URL do host do IdP. Eles serão necessários posteriormente para configurar o ODBC para conexão com o Athena.

Recuperar informações de configuração de ODBC do Okta

1. Escolha a guia General (Geral) da aplicação no Okta e role para baixo até a seção App Embed Link (Link de incorporação da aplicação).



O URL do link de incorporação está no seguinte formato:

```
https://trial-1234567.okta.com/home/amazon_aws_redshift/Abc1de2fghi3J45kL678/
abc1defghij2klmNo3p4
```

2. No URL do link de incorporação, extraia e salve as seguintes partes:

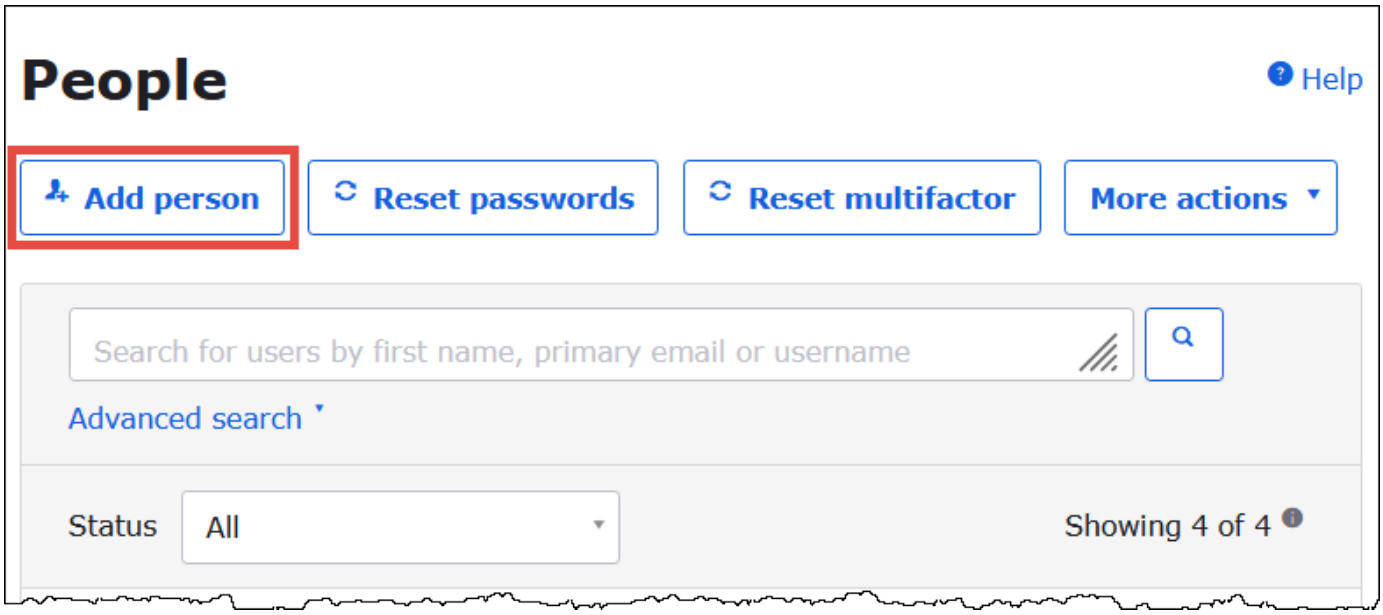
- O primeiro segmento depois de `https://`, até e incluindo `okta.com` (por exemplo, trial-1234567.okta.com). Esse é seu host de IdP.
- Os dois últimos segmentos do URL, inclusive a barra no meio. Os segmentos são duas cadeias de 20 caracteres com uma mistura de números e letras maiúsculas e minúsculas (por exemplo, Abc1de2fghi3J45kL678/abc1defghij2klmNo3p4). Esse é o ID da aplicação.

Adicionar um usuário à aplicação no Okta

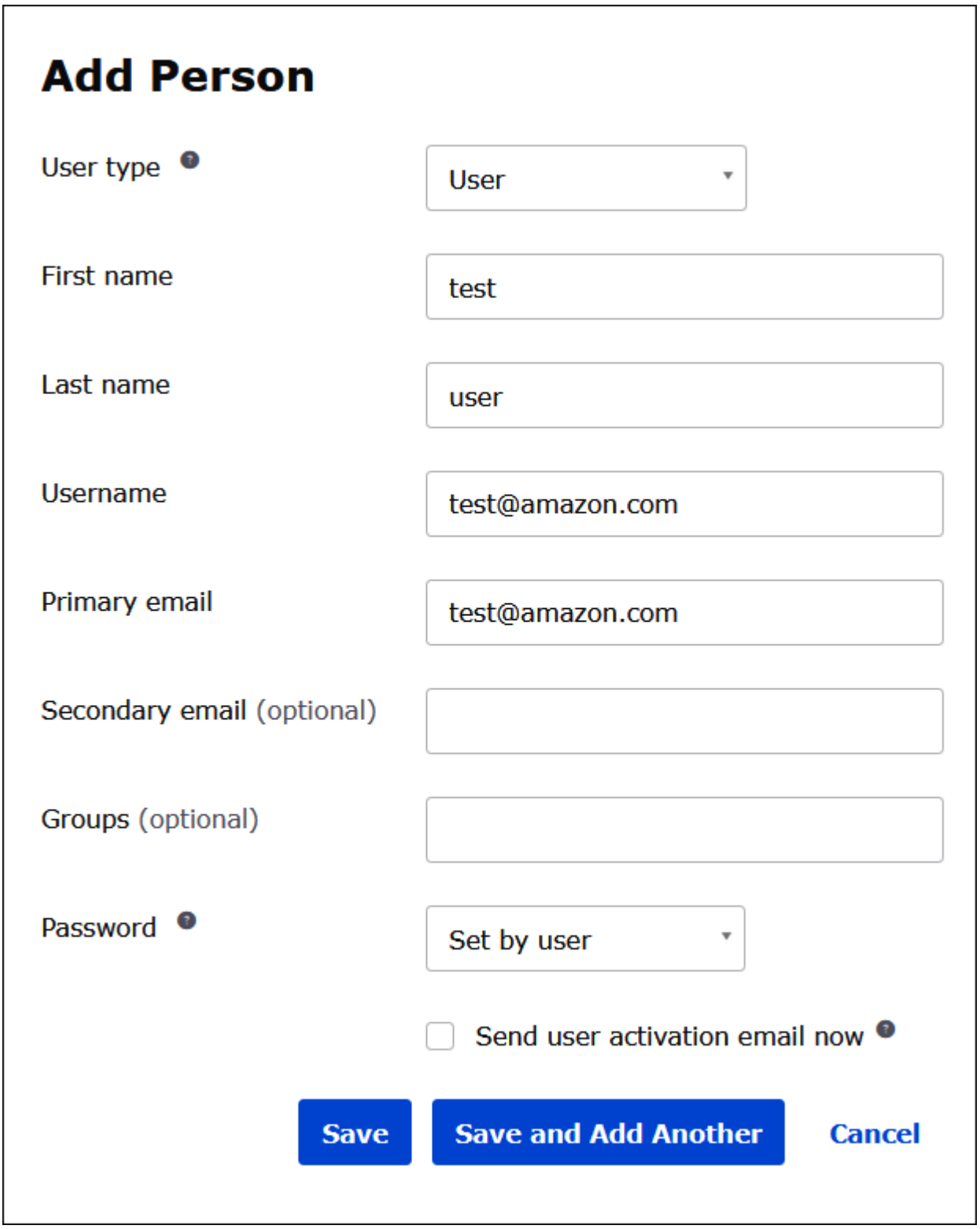
Agora você já pode adicionar um usuário à aplicação no Okta.

Adicionar um usuário à aplicação no Okta

1. No painel de navegação à esquerda, escolha Directory (Diretório) e People (Pessoas).
2. Escolha Add person (Adicionar pessoa).



3. Na caixa de diálogo Add Person (Adicionar pessoa), insira as informações a seguir.
   - Insira os valores em First name (Nome) e Last name (Sobrenome). Este tutorial usa **`test user`**.
   - Insira os valores de Username (Nome de usuário) e Primary email (E-mail principal). Este tutorial usa **`test@amazon.com`** para ambos. Os requisitos de segurança para senhas podem variar.

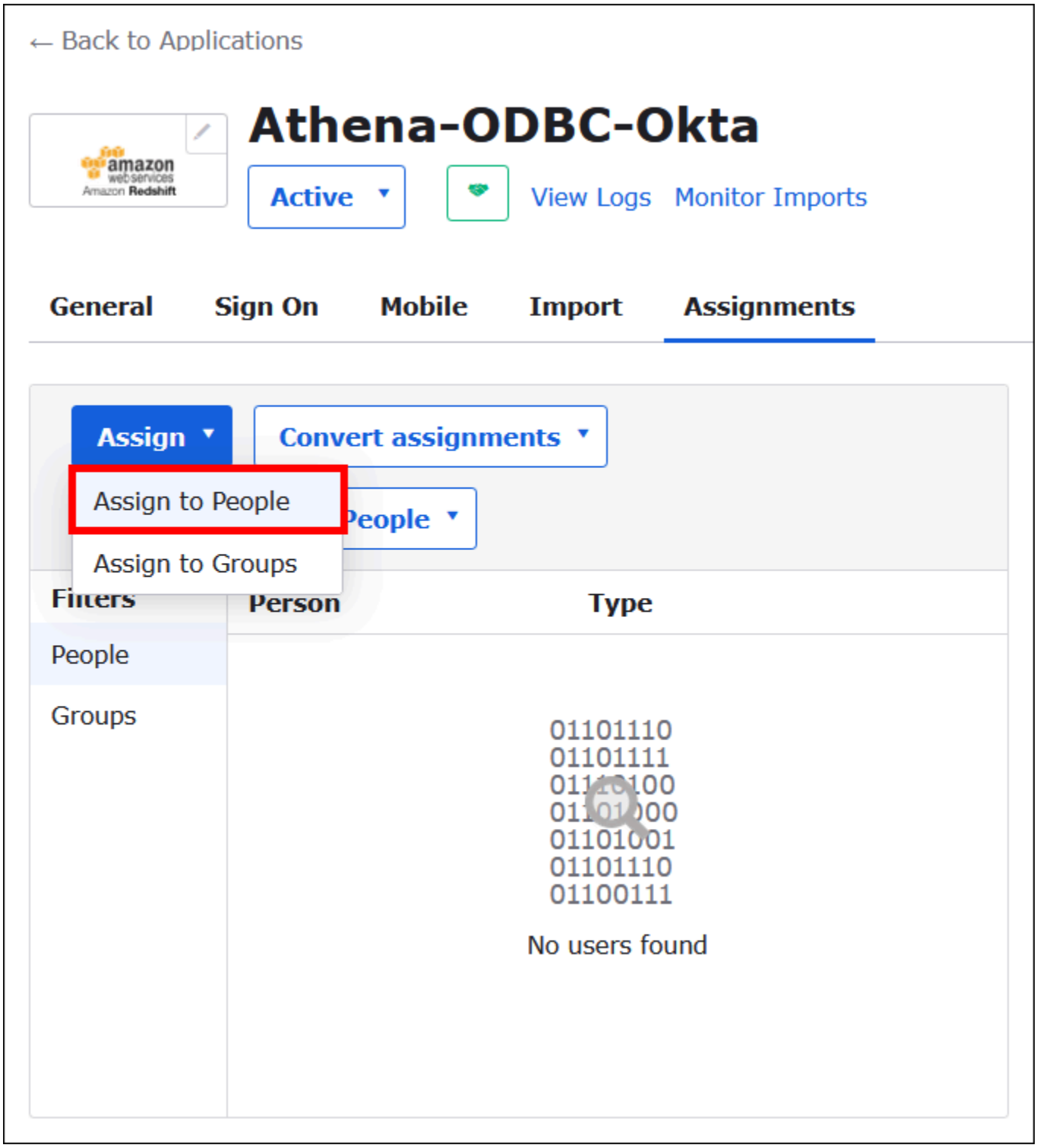


4. Escolha Salvar.

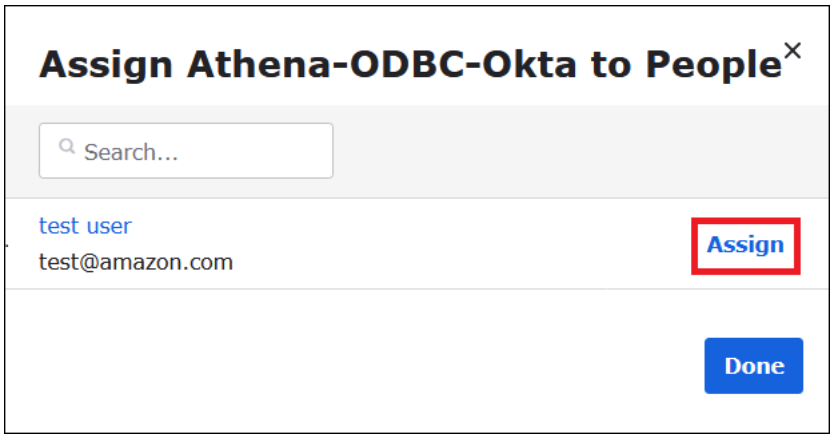
Agora você já pode atribuir o usuário criado à aplicação.

Atribuir um usuário à aplicação:

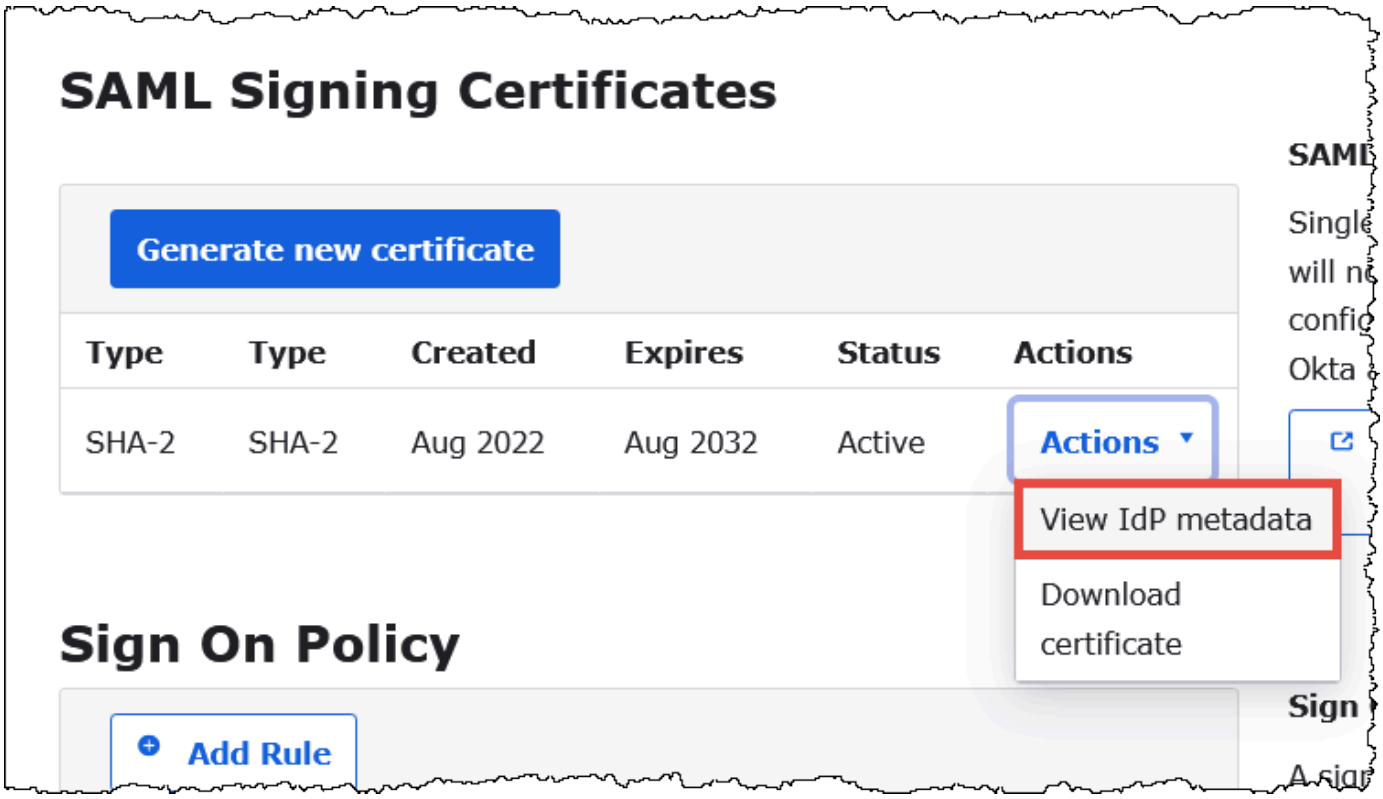
1. No painel de navegação, escolha Applications (Aplicações), Applications (Aplicações) e escolha o nome da aplicação (por exemplo, Atena-ODBC-Okta).
2. Escolha Assign (Atribuir) e depois escolha Assign to People (Atribuir a pessoas).



3. Escolha a opção Assign (Atribuir) para o usuário e escolha Done (Concluído).



4. No prompt, escolha Save and Go Back (Salvar e voltar). A caixa de diálogo exibe o status do usuário como Assigned (Atribuído).
5. Selecione Done (Concluído).
6. Escolha a guia Sign On (Fazer logon).
7. Role para baixo até a seção SAML Signing Certificates (Certificados de assinatura SAML).
8. Escolha Ações.
9. Abra o menu de contexto (clique com o botão direito) em View IdP metadata (Visualizar metadados do IdP) e escolha a opção do navegador para salvar o arquivo.
10. Salve o arquivo com uma extensão `.xml`.

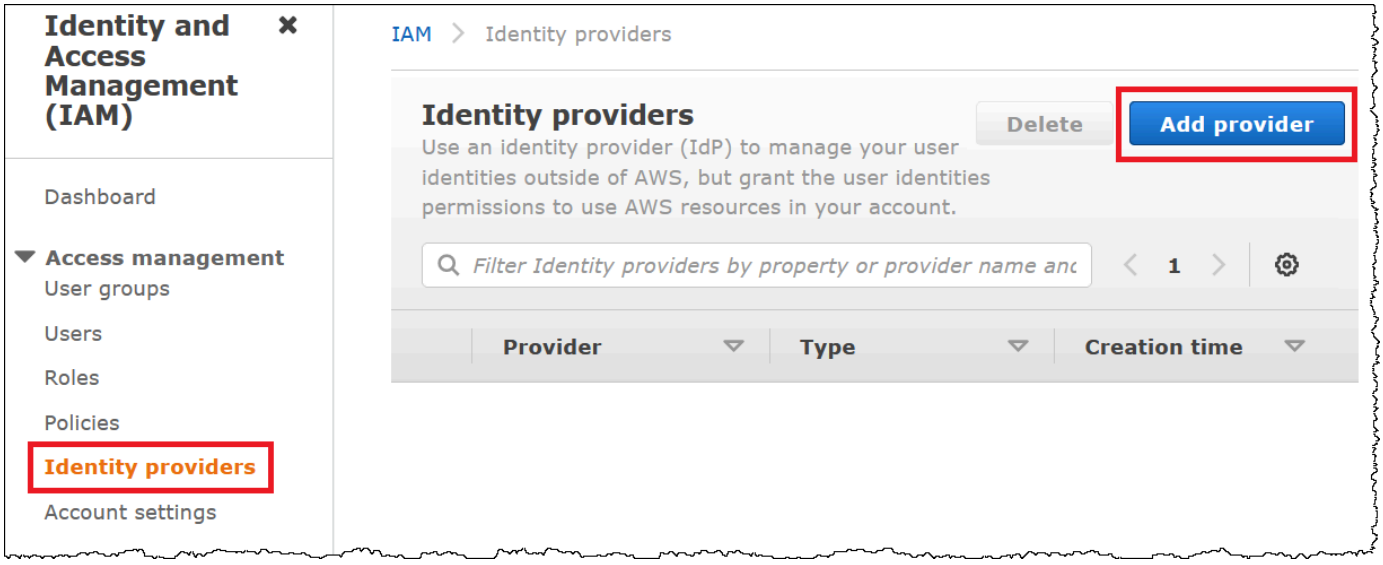


Criar um perfil e um provedor de identidade SAML da AWS

Agora você já pode carregar o arquivo XML de metadados no console do IAM na AWS. Esse arquivo será usado para criar um perfil e um provedor de identidade do AWS SAML. Use uma conta de administrador de produtos da AWS para executar essas etapas.

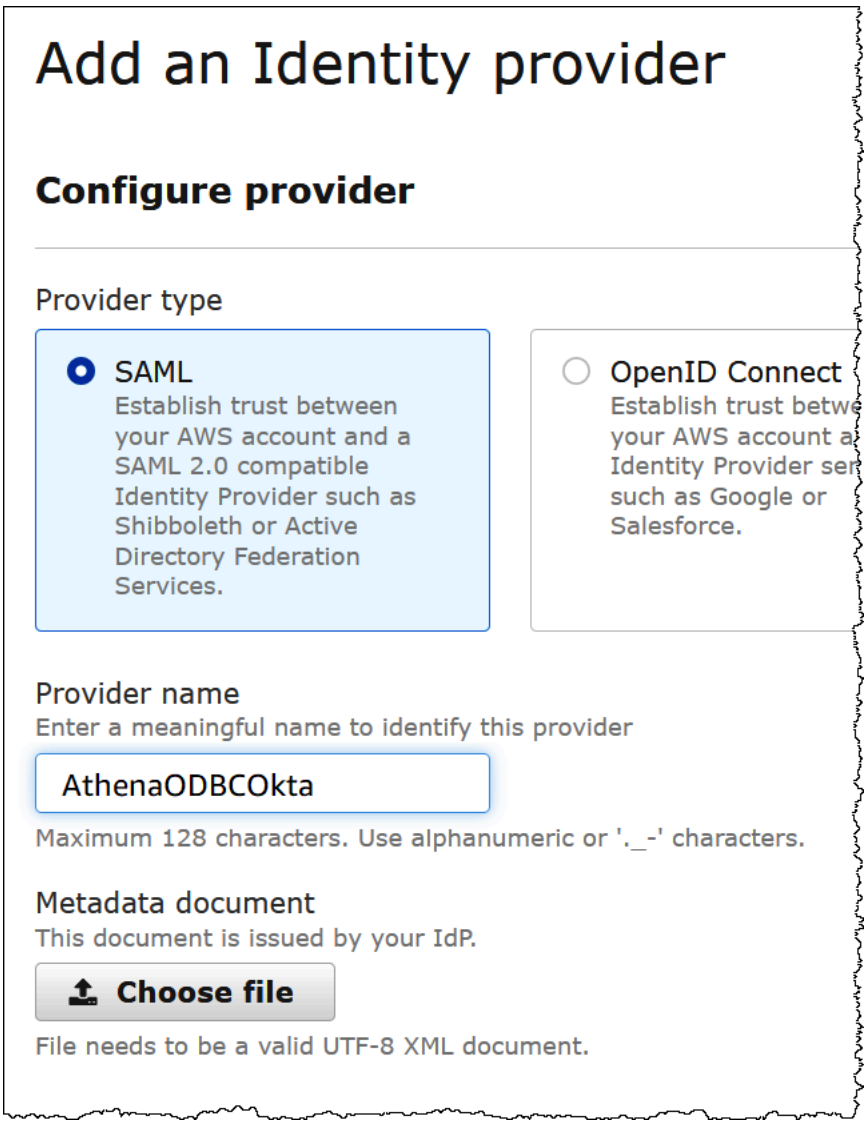
Criar um perfil e um provedor de identidade SAML na AWS

1. Faça login em AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/IAM/.
2. No painel de navegação, escolha Identity providers (Provedores de identidade) e, em seguida Add provider (Adicionar provedor).



3. Na página Add an Identity provider (Adicionar um provedor de identidade), em Configure provider (Configurar provedor), insira as informações a seguir.

   - Em Provider type (Tipo de provedor), escolha SAML.
   - Em Provider name (Nome do provedor), insira um nome para o provedor (por exemplo, **`AthenaODBCOkta`**).
   - Em Metadata document (Documento de metadados), use a opção Choose file (Escolher arquivo) para carregar o arquivo XML de metadados do provedor de identidade (IdP) que você baixou.



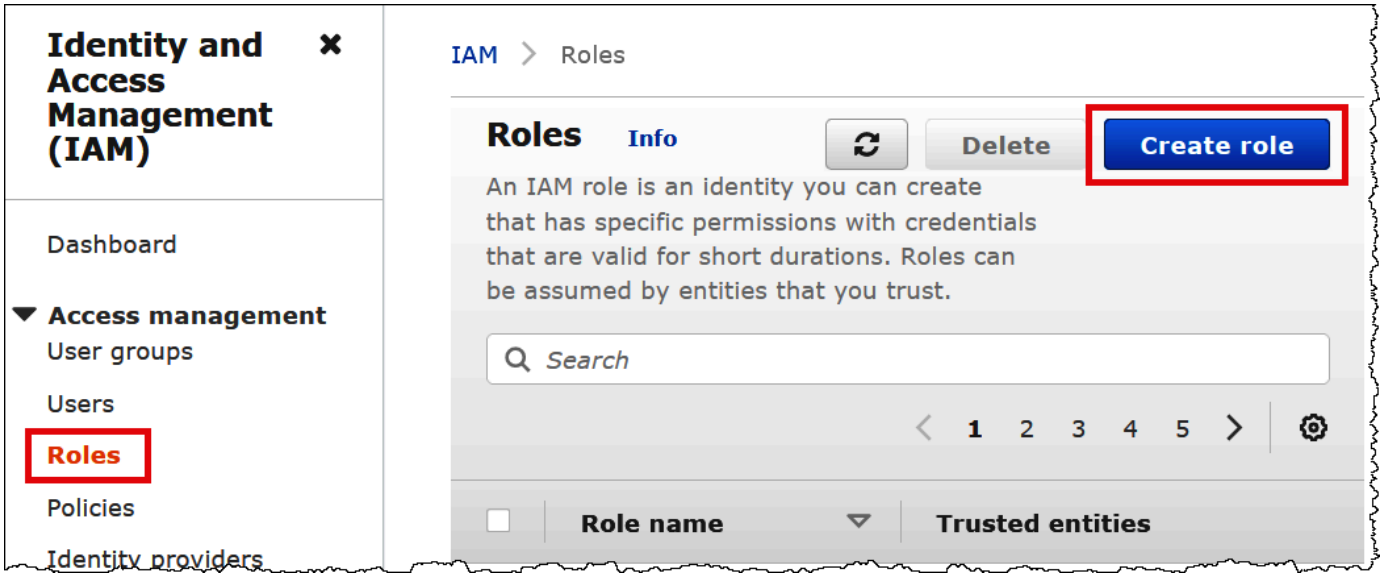
4. Escolha Add provider (Adicionar provedor).

Criar um perfil do IAM para acessar o Athena e o Amazon S3

Agora você já pode criar um perfil do IAM para acessar o Athena e o Amazon S3. Você atribuirá esse perfil ao usuário. Dessa forma, você poderá fornecer ao usuário acesso de logon único ao Athena.

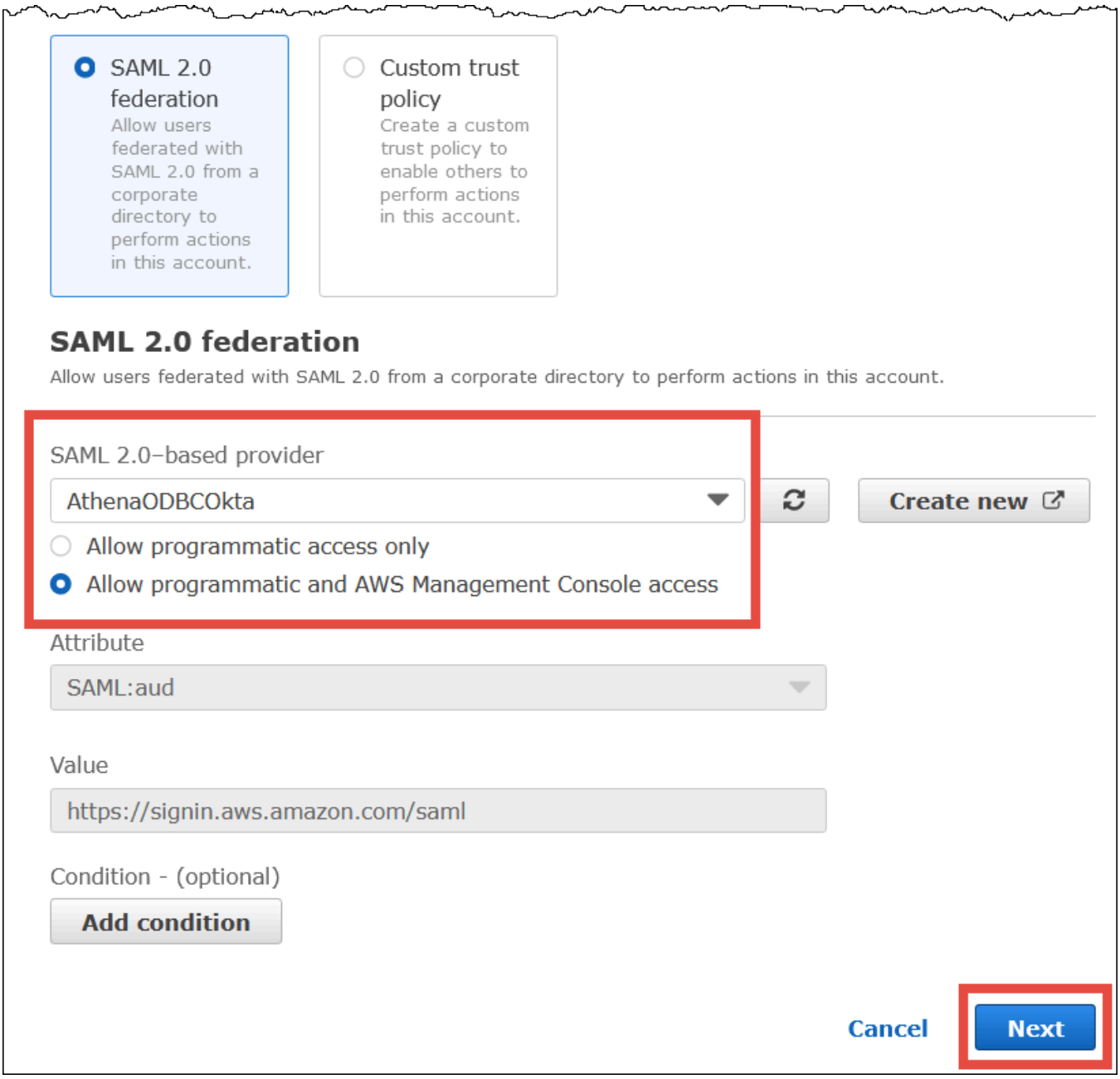
Criar uma perfil do IAM para as tarefas

1. No painel de navegação do console do IAM, escolha Roles (Funções) e Create role (Criar função).

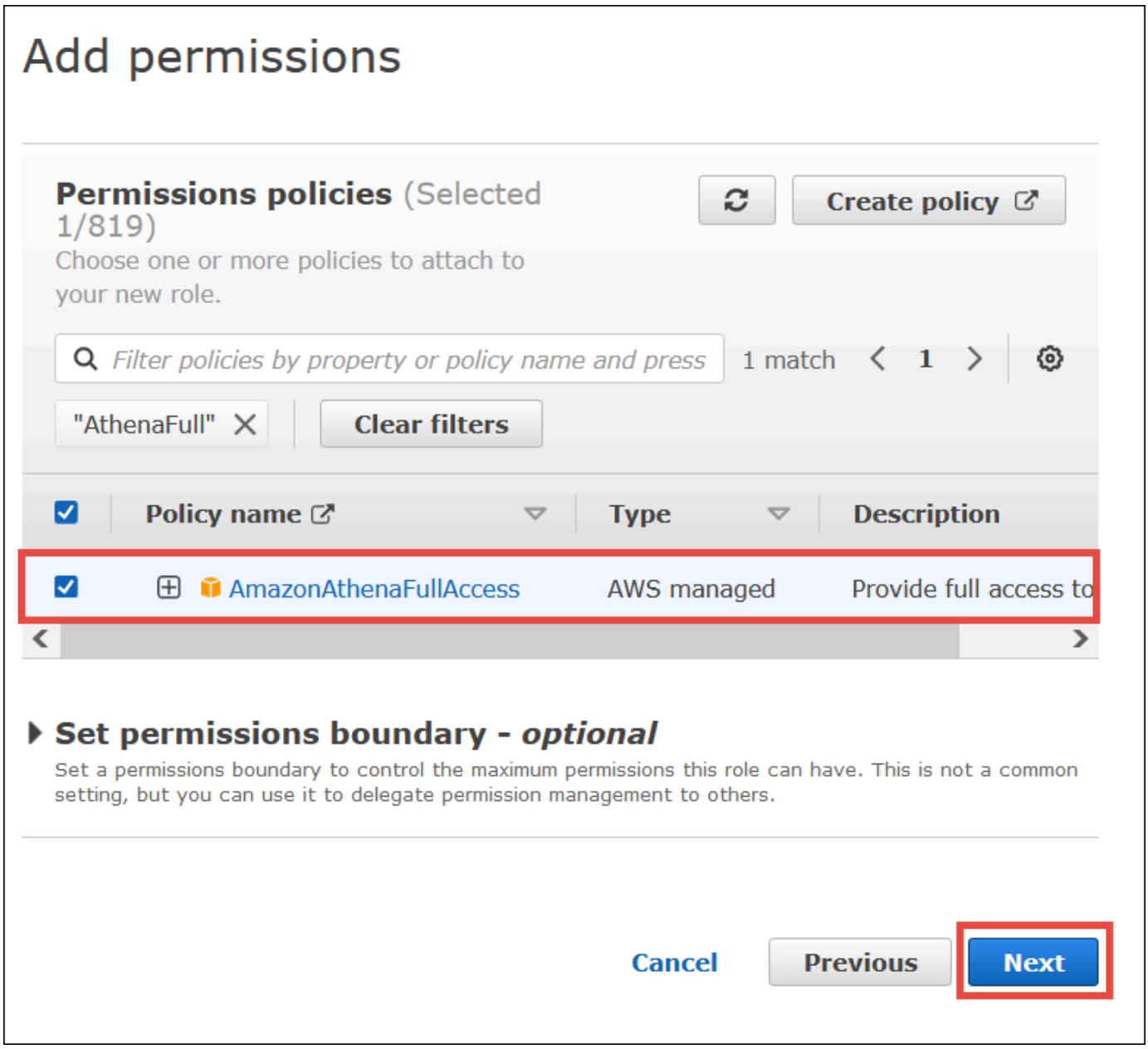


2. Na página Create role (Criar perfil), escolha as seguintes opções:

   - Em Select type of trusted entity (Selecionar tipo de entidade confiável), escolha SAML 2.0 Federation.
   - Em SAML 2.0–based provider (Provedor baseado no SAML 2.0), escolha o provedor de identidade SAML que você criou (por exemplo, AthenaODBCOkta).
   - Selecione Permitir acesso programático e pelo AWS Management Console.



3. Escolha Próximo.
4. Na página Add Permissions (Adicionar permissões), em Filter policies (Filtrar políticas), insira **`AthenaFull`** e pressione ENTER.
5. Selecione a política gerenciada `AmazonAthenaFullAccess` e escolha Next (Próximo).



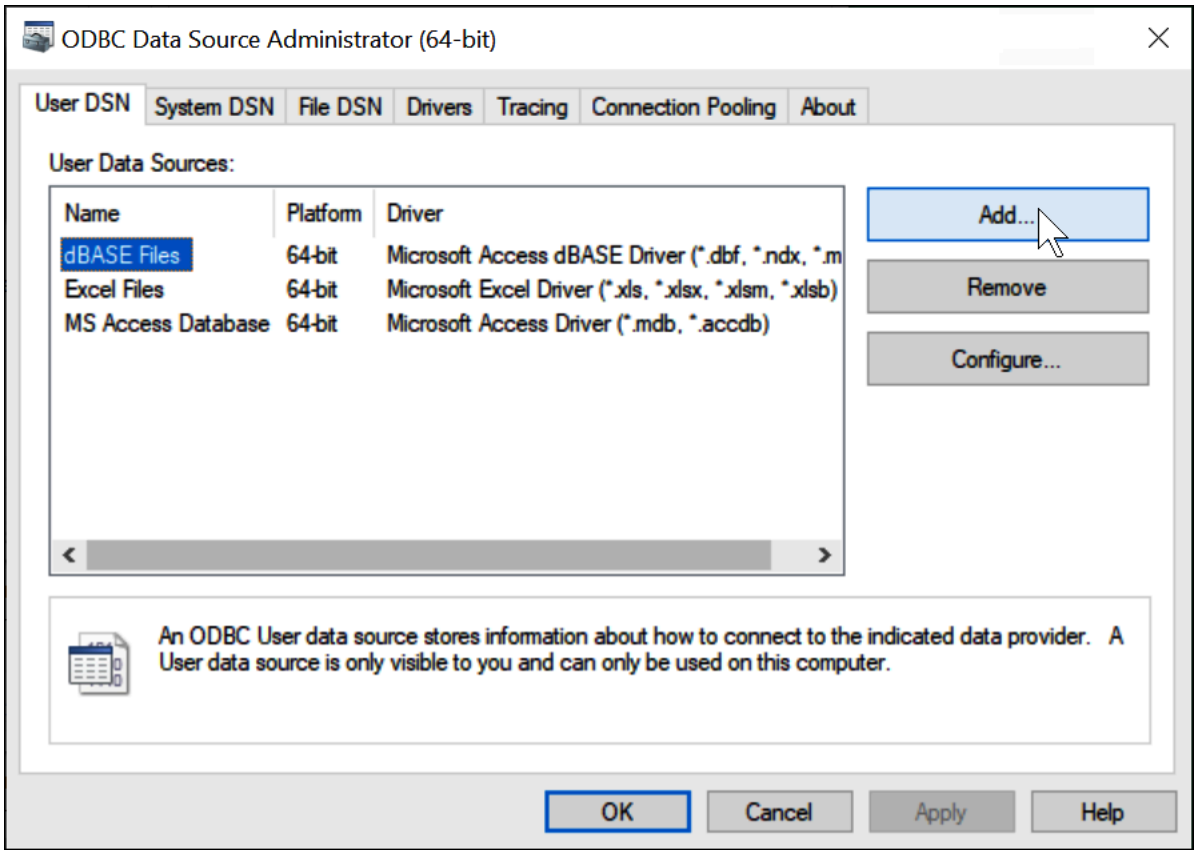
6. Na página Name, review, and create (Nomear, revisar e criar), em Role name (Nome do perfil), insira um nome para o perfil (por exemplo, **`Athena-ODBC-OktaRole`**) e escolha Create role (Criar perfil).

Configurar a conexão ODBC do Okta com o Athena

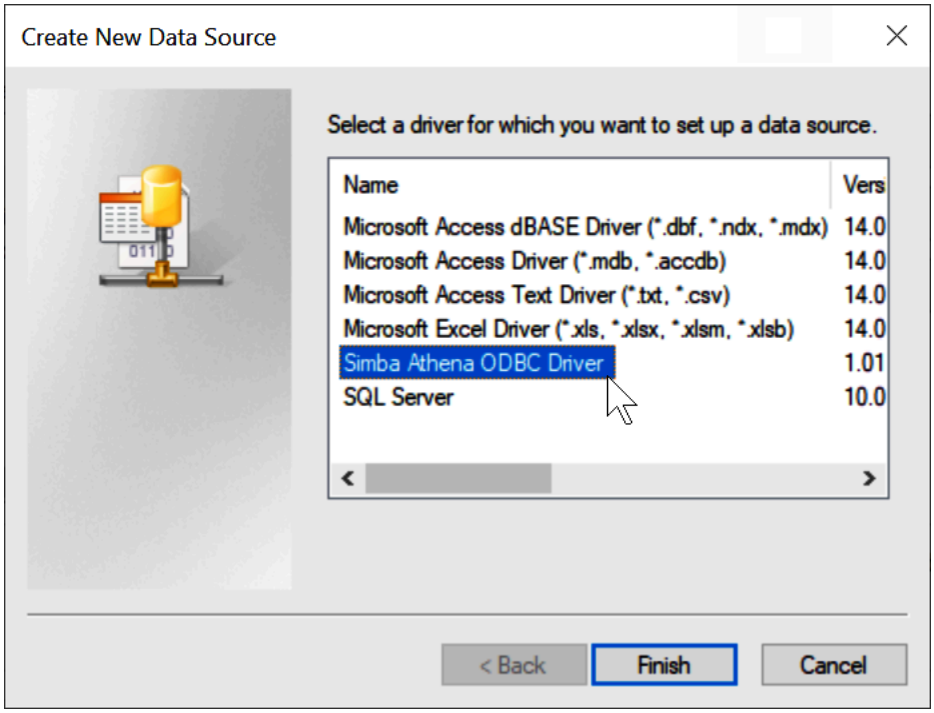
Agora você já pode configurar a conexão ODBC do Okta com o Athena usando o programa ODBC Data Sources no Windows.

Configurar sua conexão ODBC do Okta com o Athena

1. No Windows, inicie o programa ODBC Data Sources .
2. No programa ODBC Data Source Administrator, escolhaAdd (Adicionar).

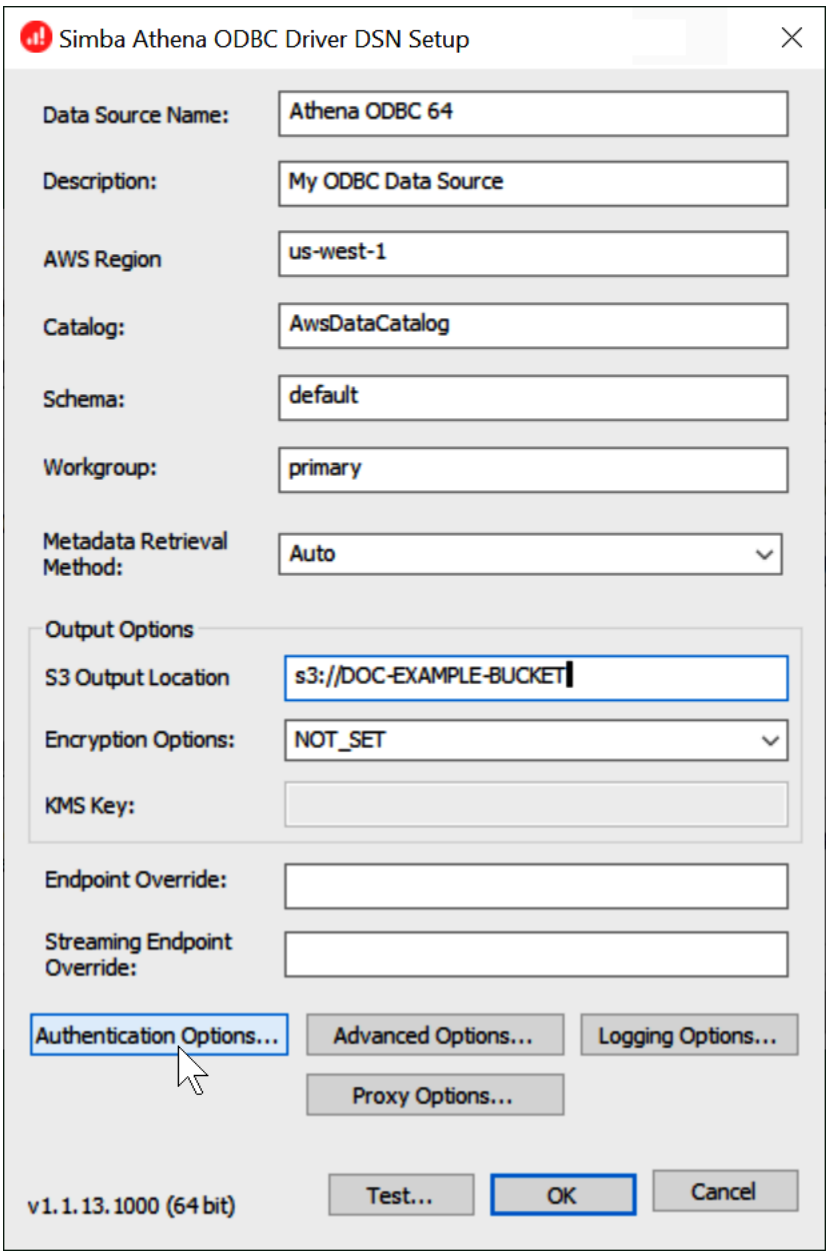


3. Escolha Simba Athena ODBC Driver (Driver ODBC do Simba Athena) e depois escolha Finish (Finalizar).



4. Na caixa de diálogo Simba Athena ODBC Driver DSN Setup (Configuração do driver DSN do Simba Athena), insira os valores descritos.

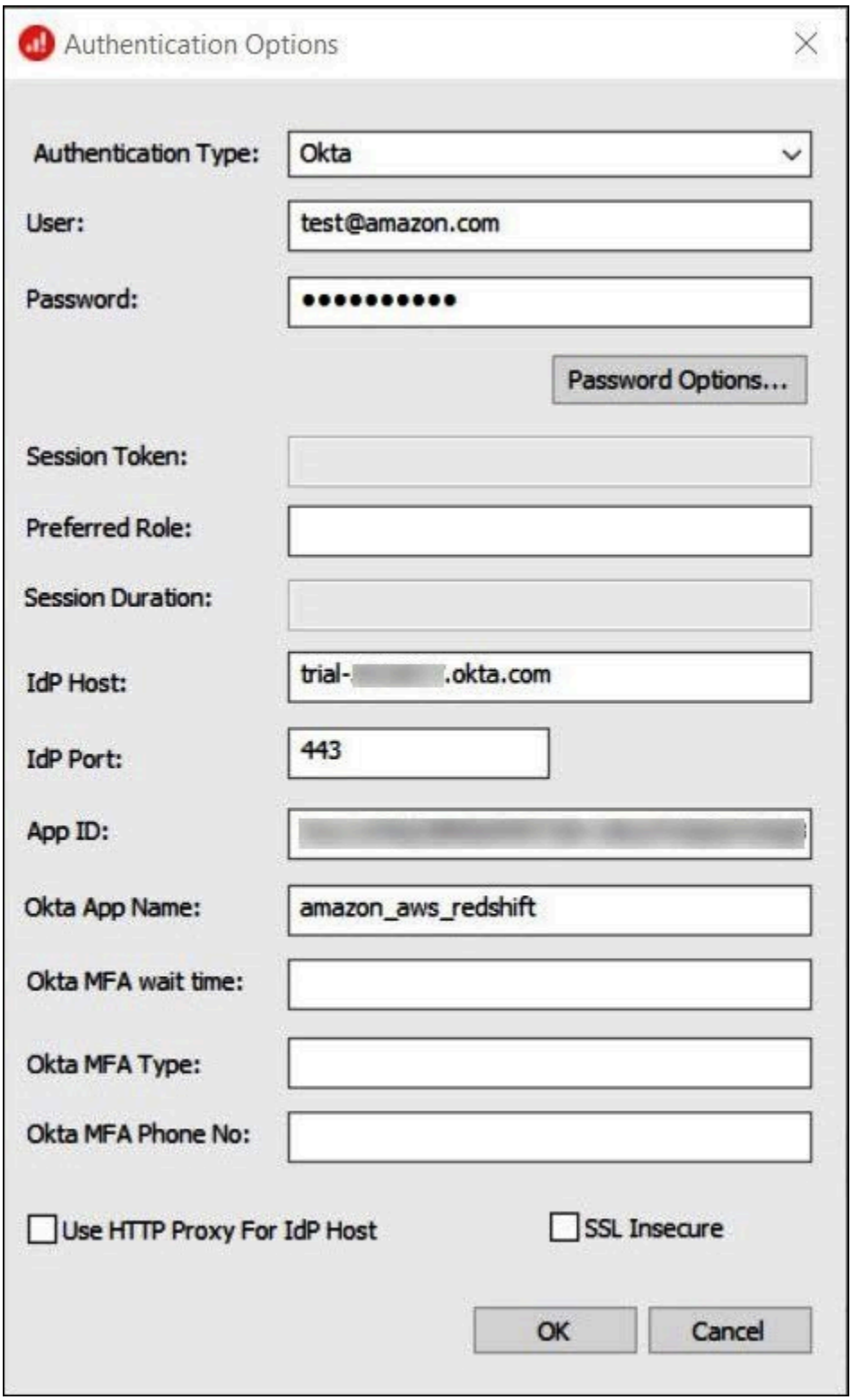
   - Para Data Source Name (Nome da fonte de dados), insira um nome para a fonte de dados (por exemplo, **`Athena ODBC 64`**).
   - Em Description (Descrição), insira uma descrição para a origem dos dados.
   - Em Região da AWS, insira a Região da AWS que você está usando (por exemplo, **`us-west-1`**).
   - Para S3 Output Location, (Local de saída do S3), insira o caminho do Amazon S3 onde deseja que sua saída seja armazenada.



5. Escolha Authentication Options (Opções de autenticação).
6. Na caixa de diálogo Authentication Options (Opções de autenticação, escolha ou insira os valores a seguir.

   - Em Authentication Type (Tipo de autenticação), escolha Okta.
   - Em User (Usuário), insira o nome de usuário do Okta.
   - Em Password (Senha), insira sua senha do Okta.
   - Em IdP Host (Host de IdP), insira o valor registrado anteriormente (por exemplo, **`trial-1234567.okta.com`**).

- Em IdP Port (Porta IdP), insira **`443`**.
- Em App ID (ID da aplicação), insira o valor registrado anteriormente (os dois últimos segmentos do link de incorporação do Okta).
- Em Okta App Name (Nome da aplicação no Okta), insira **`amazon_aws_redshift`**.



7. Escolha OK.
8. Escolha Test (Testar) para testar a conexão ou OK para concluir.

Configurar o Single Sign-On usando ODBC, SAML 2.0 e o provedor de identidade Okta

Para se conectar a origens de dados, você pode usar o Amazon Athena com provedores de identidade (IdPs) como PingOne, Okta, OneLogin e outros. A partir do driver Athena ODBC versão 1.1.13 e do driver Athena JDBC versão 2.0.25, está incluído um plugin SAML de navegador que você pode configurar para trabalhar com qualquer provedor de SAML 2.0. Este tópico mostra como configurar o driver ODBC do Amazon Athena e o plug-in SAML baseado em navegador para adicionar o recurso de autenticação única (SSO) usando o provedor de identidade Okta.

Pré-requisitos

A conclusão das etapas neste tutorial requer o seguinte:

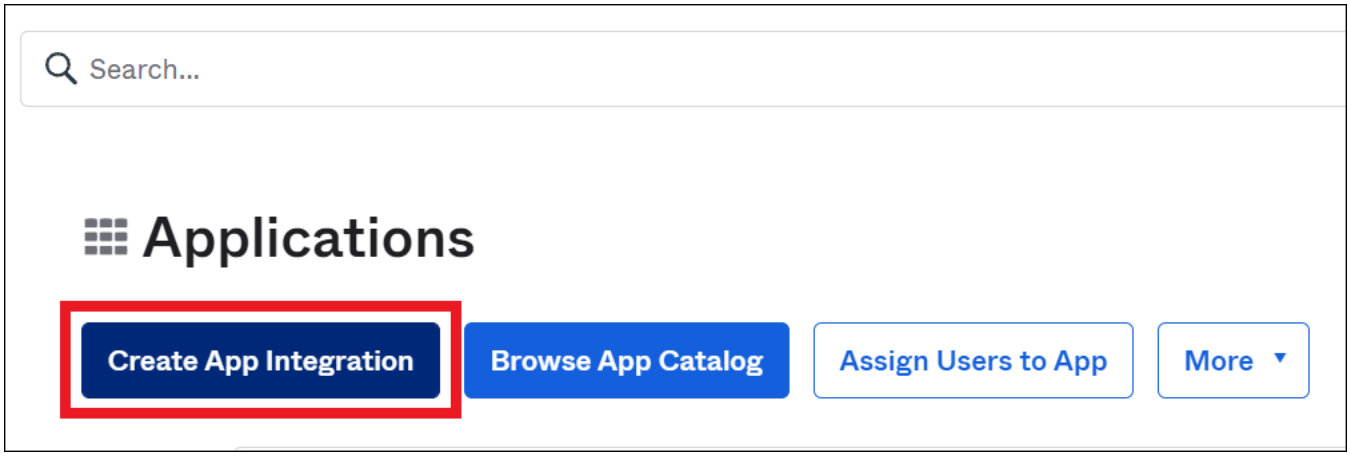
- Driver Athena ODBC versão 1.1.13 ou posterior. As versões 1.1.13 e posteriores incluem suporte a SAML no navegador. Para obter os links de download, consulte Conectar-se ao Amazon Athena com ODBC.
- Uma função do IAM que você deseje usar com o SAML. Para obter mais informações, consulte Criar uma função para a federação SAML 2.0 no Guia do usuário do IAM.
- Uma conta do Okta. Para obter informações, acesse okta.com.

Criar uma integração de aplicações no Okta

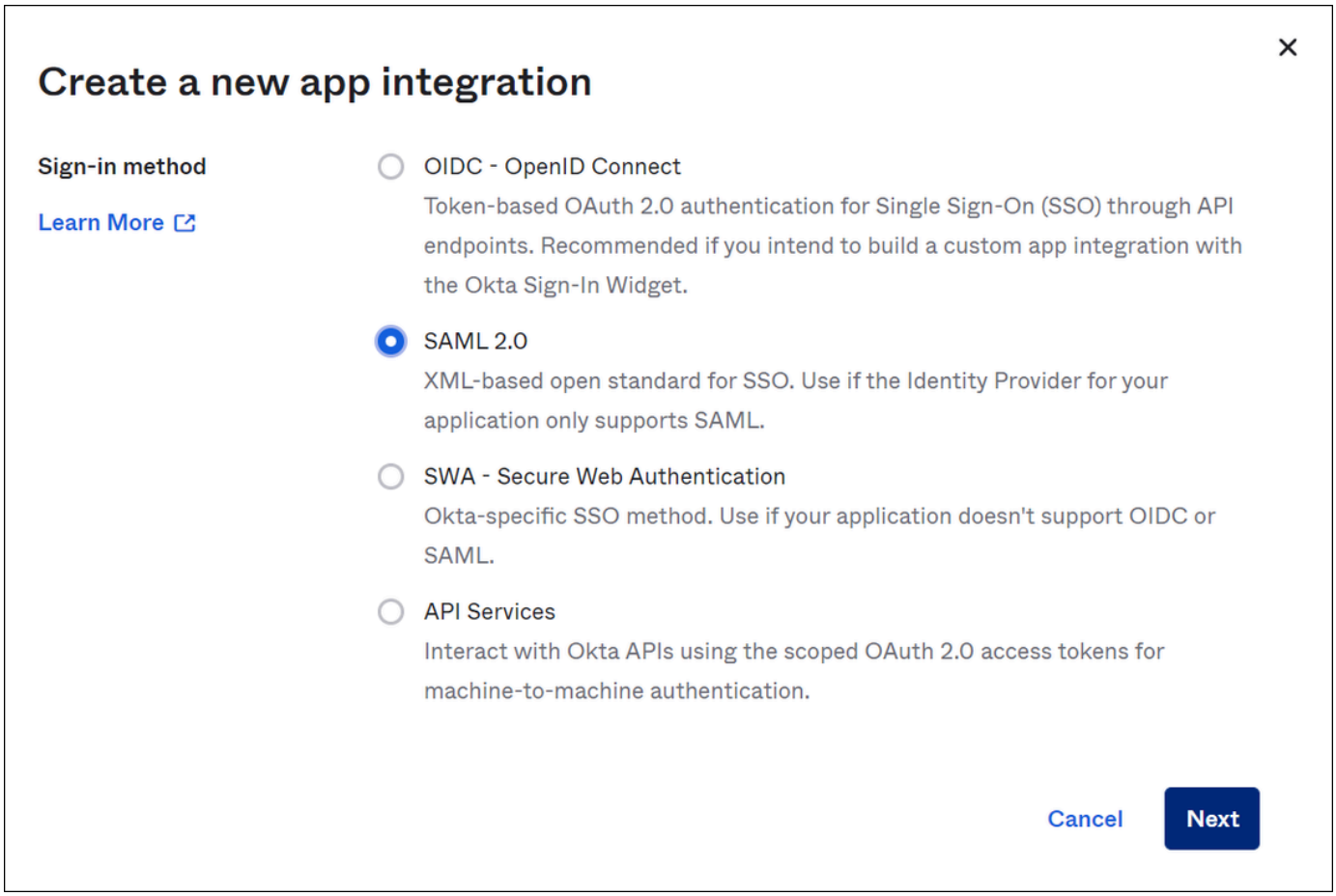
Primeiro, use o painel do Okta para criar e configurar uma aplicação SAML 2.0 para autenticação única no Athena.

Para usar o painel do Okta para configurar a autenticação única para o Athena

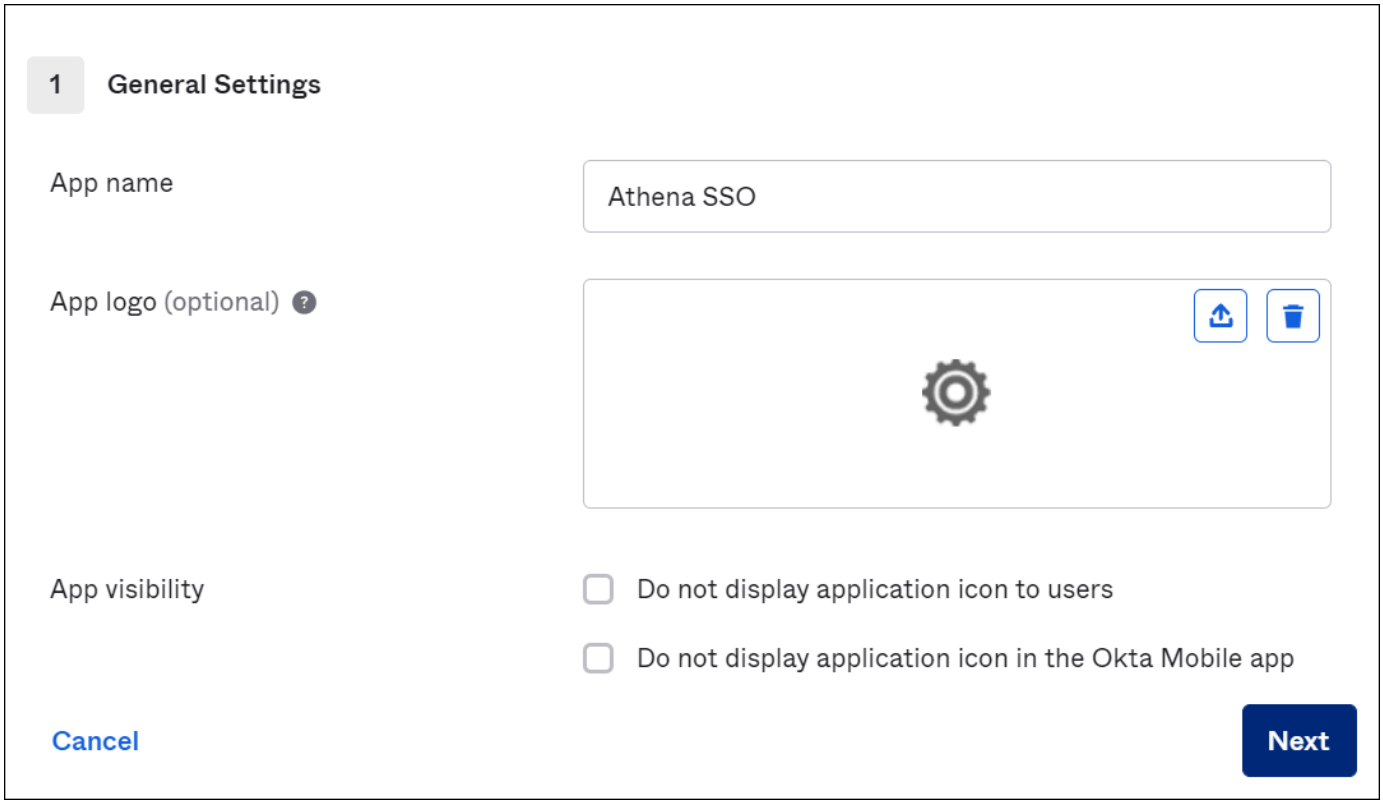
1. Faça login na página de administração do Okta em `okta.com`.
2. No painel de navegação, escolha Applications (Aplicações), Applications (Aplicações).
3. Na página Applications (Aplicações), escolha Create App Integration (Criar integração de aplicação).



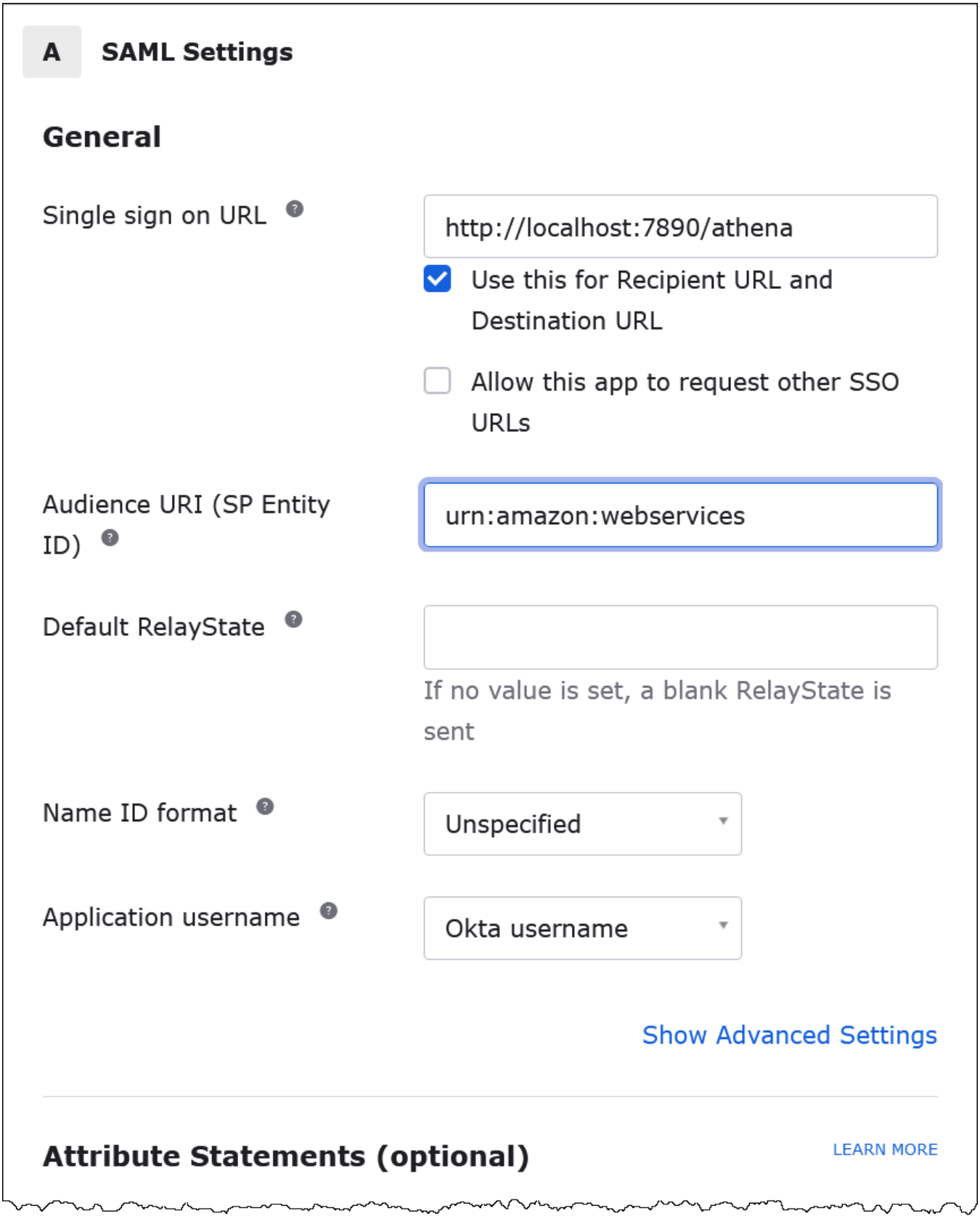
4. Na caixa de diálogo Create a new app integration (Criar uma nova integração de aplicações) para Sign-in method (Método de login), selecioneSAML 2.0 e, depois, escolha Next (Próximo).



5. Na página Create SAML Integration (Criar integração SAML), na seção General Settings (Configurações gerais) insira um nome para a aplicação. Este tutorial usa o nome Athena SSO.



6. Escolha Próximo.
7. Na página Configure SAML (Configurar o SAML), na seção SAML Settings (Configurações do SAML), insira os seguintes valores:
   - Para Single sign on URL, (URL de autenticação única), insira **`http://localhost:7890/athena`**
   - Para Audience URI, (URI do público), insira **`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. Para Attribute Statements (optional) (Instruções de atributo (opcional)), insira os dois pares nome/valor a seguir. Esses são atributos de mapeamento obrigatórios.

   - Para Name (Nome), insira o seguinte URL:

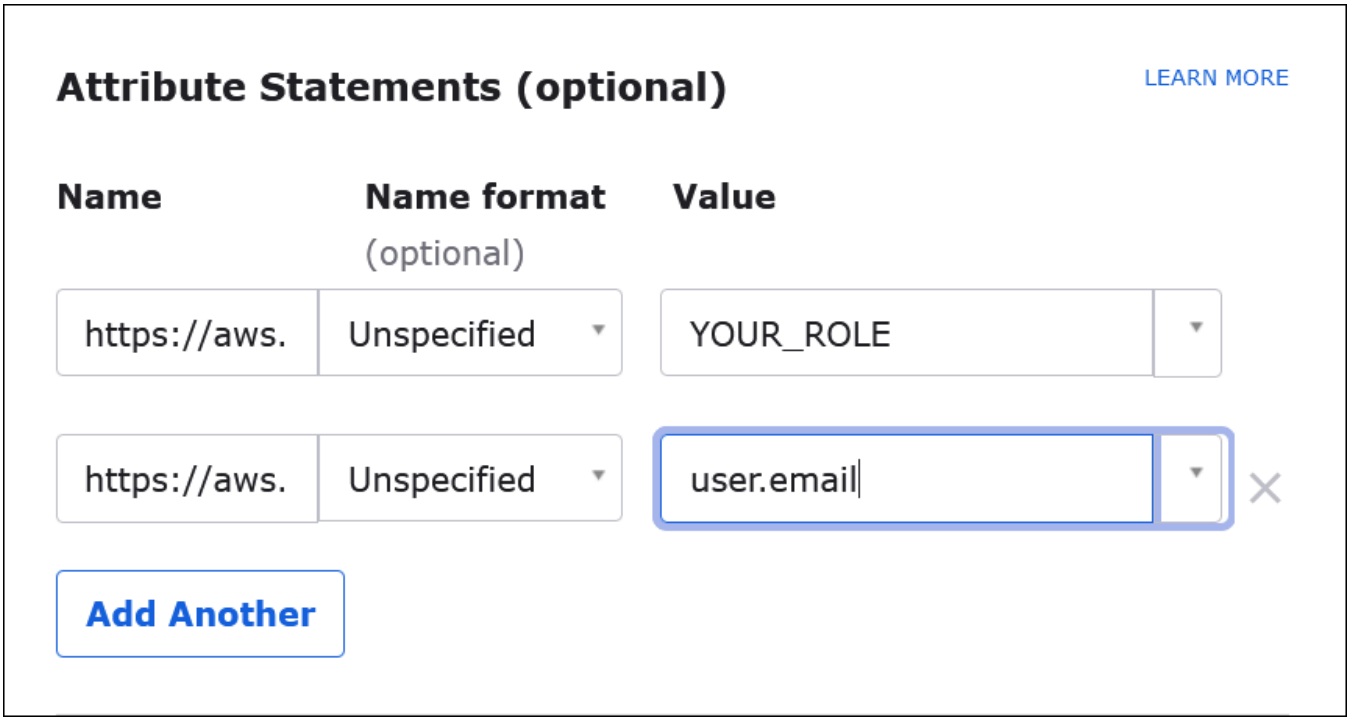
     `https://aws.amazon.com/SAML/Attributes/Role`

     Para Value (Valor) insira o nome da função do IAM. Para obter mais informações sobre o formato da função do IAM, consulte Configurar asserções SAML para a resposta de autenticação no Guia do usuário do IAM.

   - Para Name (Nome), insira o seguinte URL:

     `https://aws.amazon.com/SAML/Attributes/RoleSessionName`

     Em Valor, insira **`user.email`**.



9. Escolha Next (Próximo) e, em seguida, escolha Finish (Concluir).

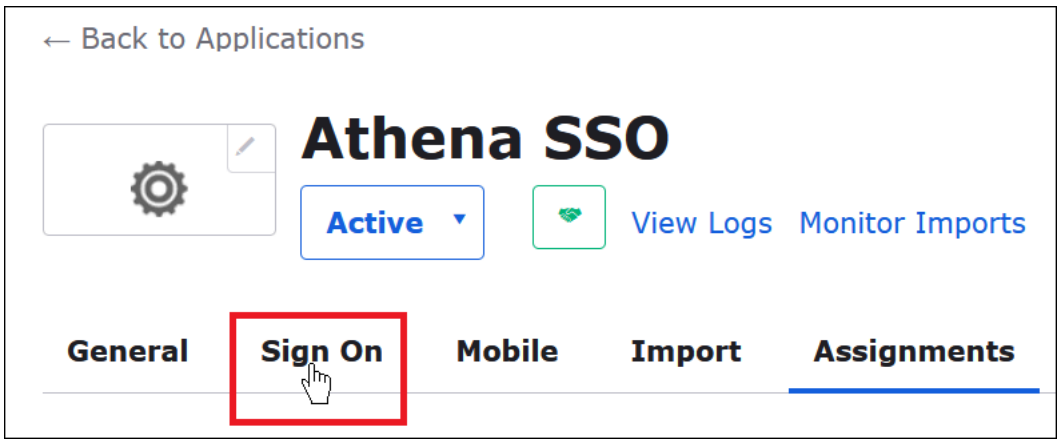
   Quando o Okta cria a aplicação, ele também cria o URL de login que você recuperará em seguida.

Obter o URL de login no painel do Okta

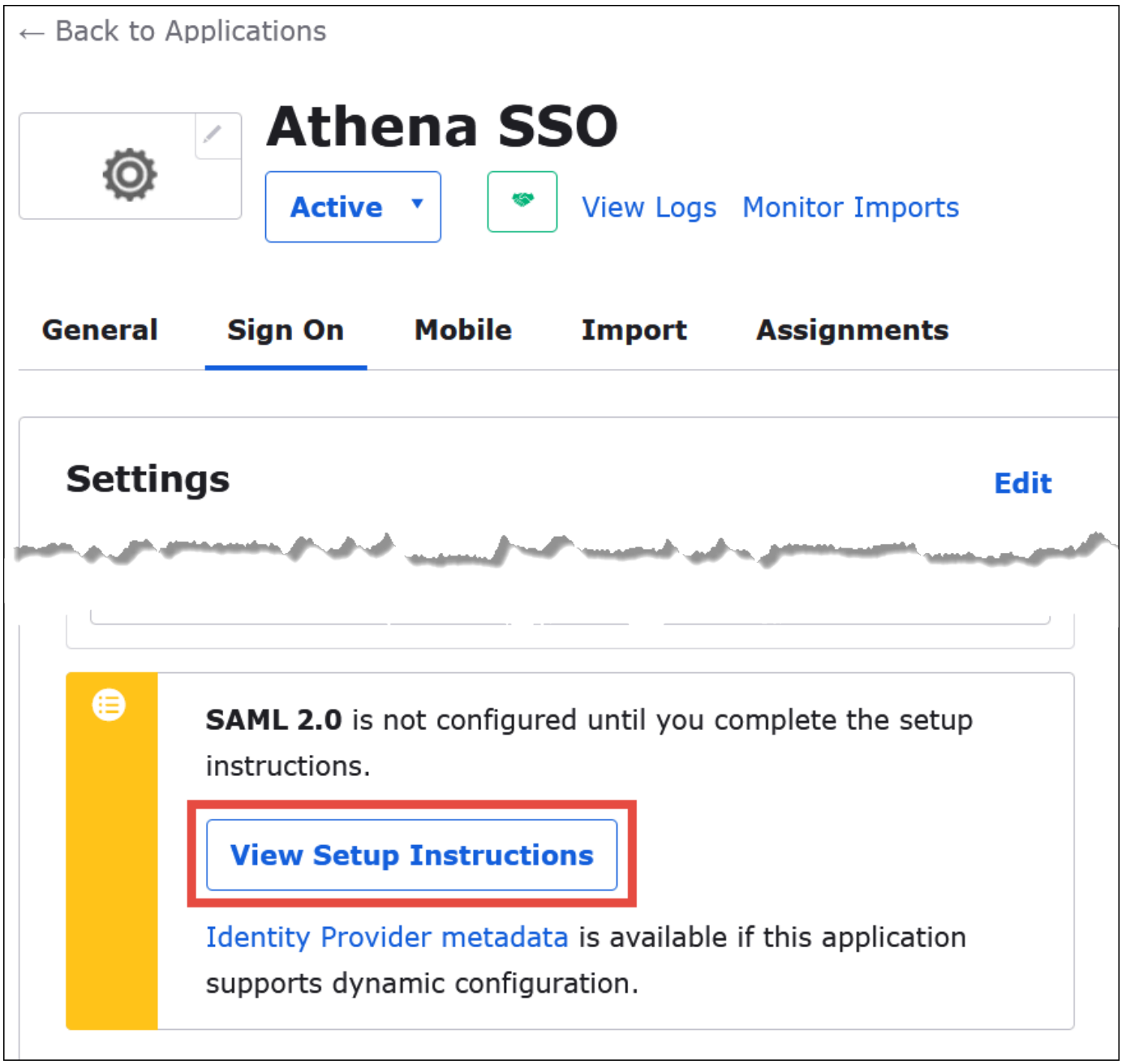
Agora que a aplicação foi criada, você pode obter o URL de login e outros metadados no painel do Okta.

Obter o URL de login no painel do Okta

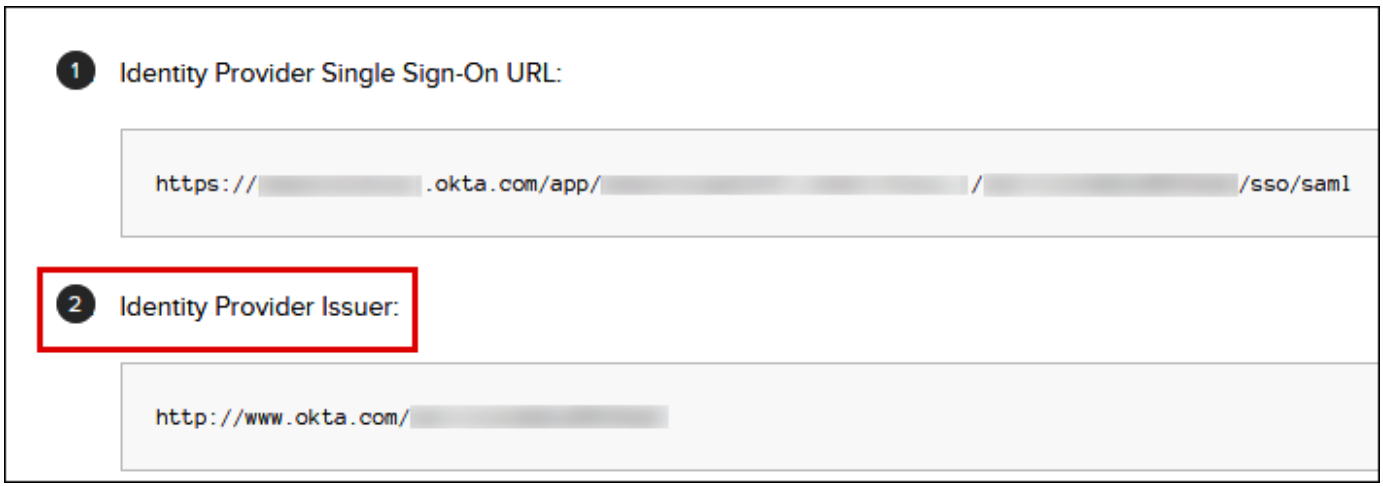
1. No painel de navegação do Okta, escolha Applications (Aplicações), Applications (Aplicações).
2. Escolha a aplicação para a qual deseja encontrar o URL de login (por exemplo, AthenaSSO).
3. Na página da aplicação, selecione Sign On (Logon).



4. Escolha View Setup Instructions (Exibir instruções de configuração).



5. Na página How to Configure SAML 2.0 for Athena SSO (Como configurar o SAML 2.0 para o Athena SSO), localize o URL para Identity Provider Issuer (Emissor do provedor de identidade). Em algumas lugares no painel do Okta, esse URL é referenciado como SAML issuer ID (ID do emissor do SAML).



6. Copie ou armazene o valor de Identity Provider Single Sign-On URL (URL de logon único do provedor de identidade).

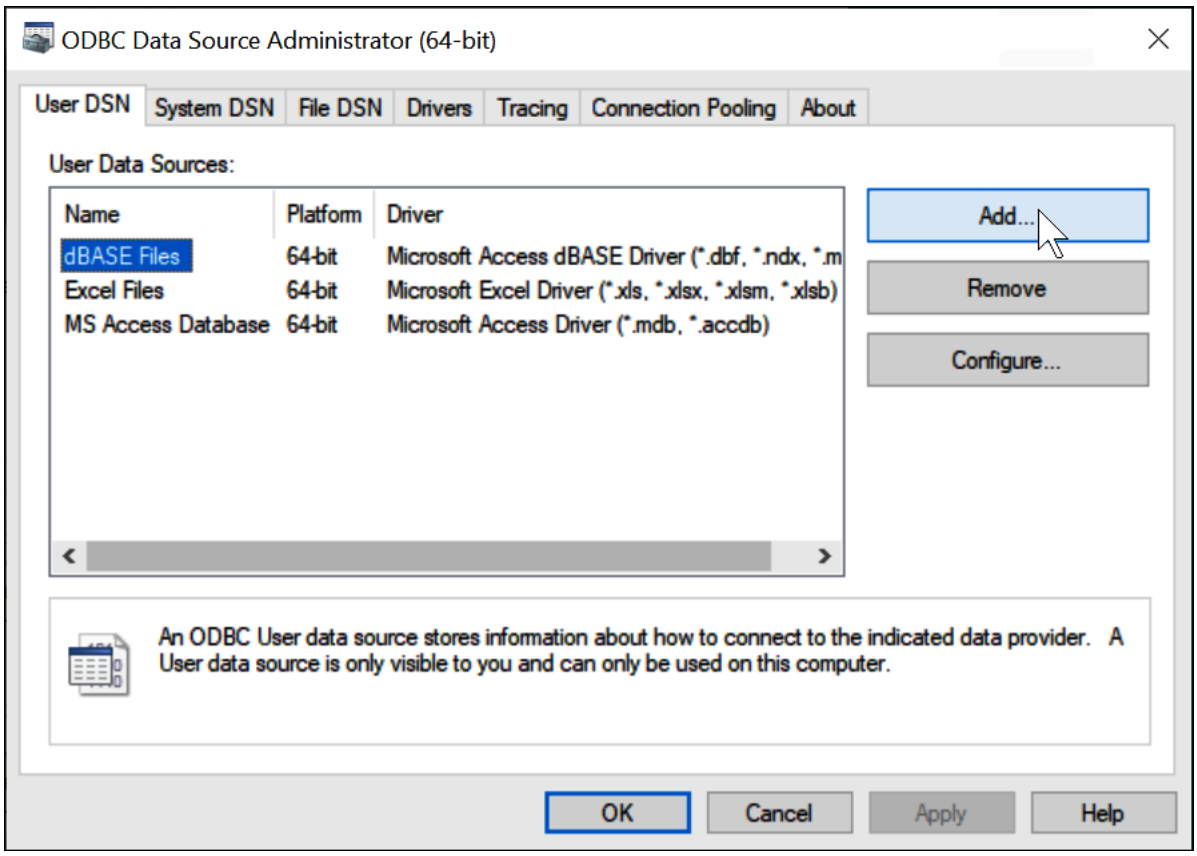
   Na próxima seção, quando configurar a conexão ODBC, você fornecerá esse valor como o parâmetro de conexão Login URL (URL de login) para o plug-in SAML do navegador.

Configurar a conexão ODBC SAML do navegador com o Athena

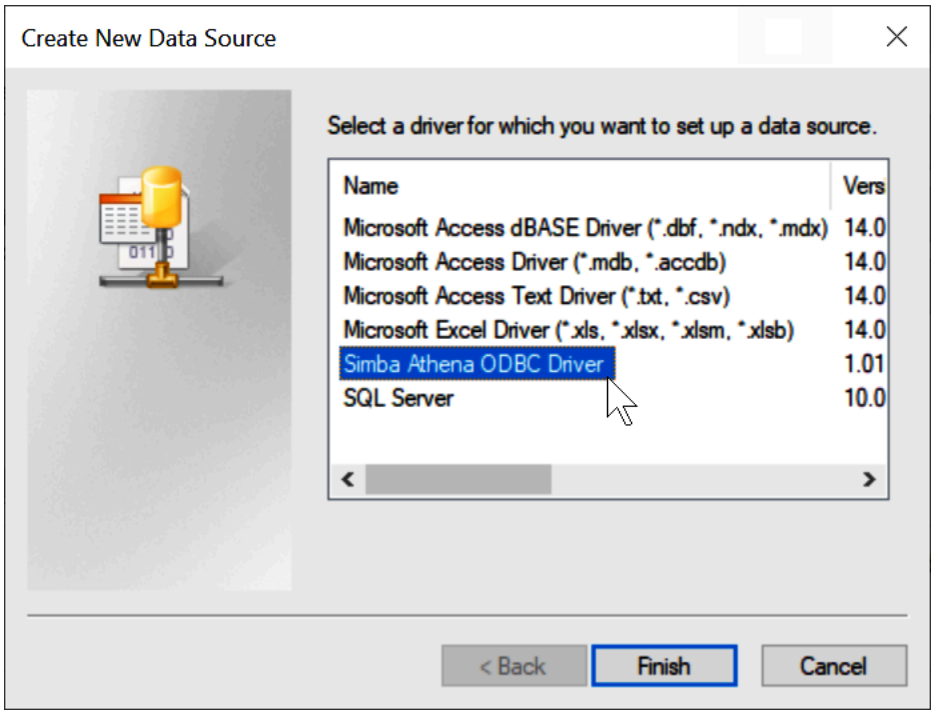
Agora você está pronto para configurar a conexão SAML do navegador com o Athena usando o programa ODBC Data Sources no Windows.

Para configurar a conexão ODBC SAML do navegador com o Athena

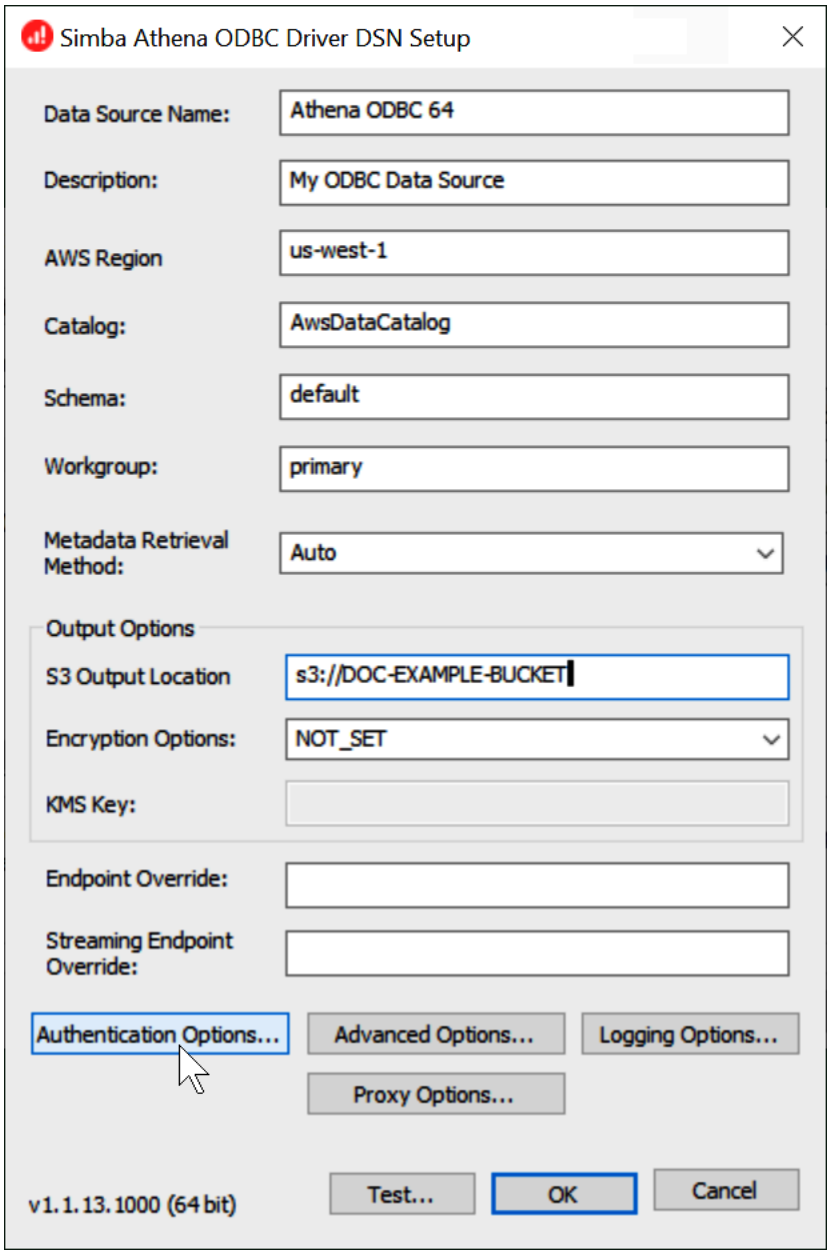
1. No Windows, inicie o programa ODBC Data Sources .
2. No programa ODBC Data Source Administrator, escolhaAdd (Adicionar).



3. Escolha Simba Athena ODBC Driver (Driver ODBC do Simba Athena) e depois escolha Finish (Finalizar).

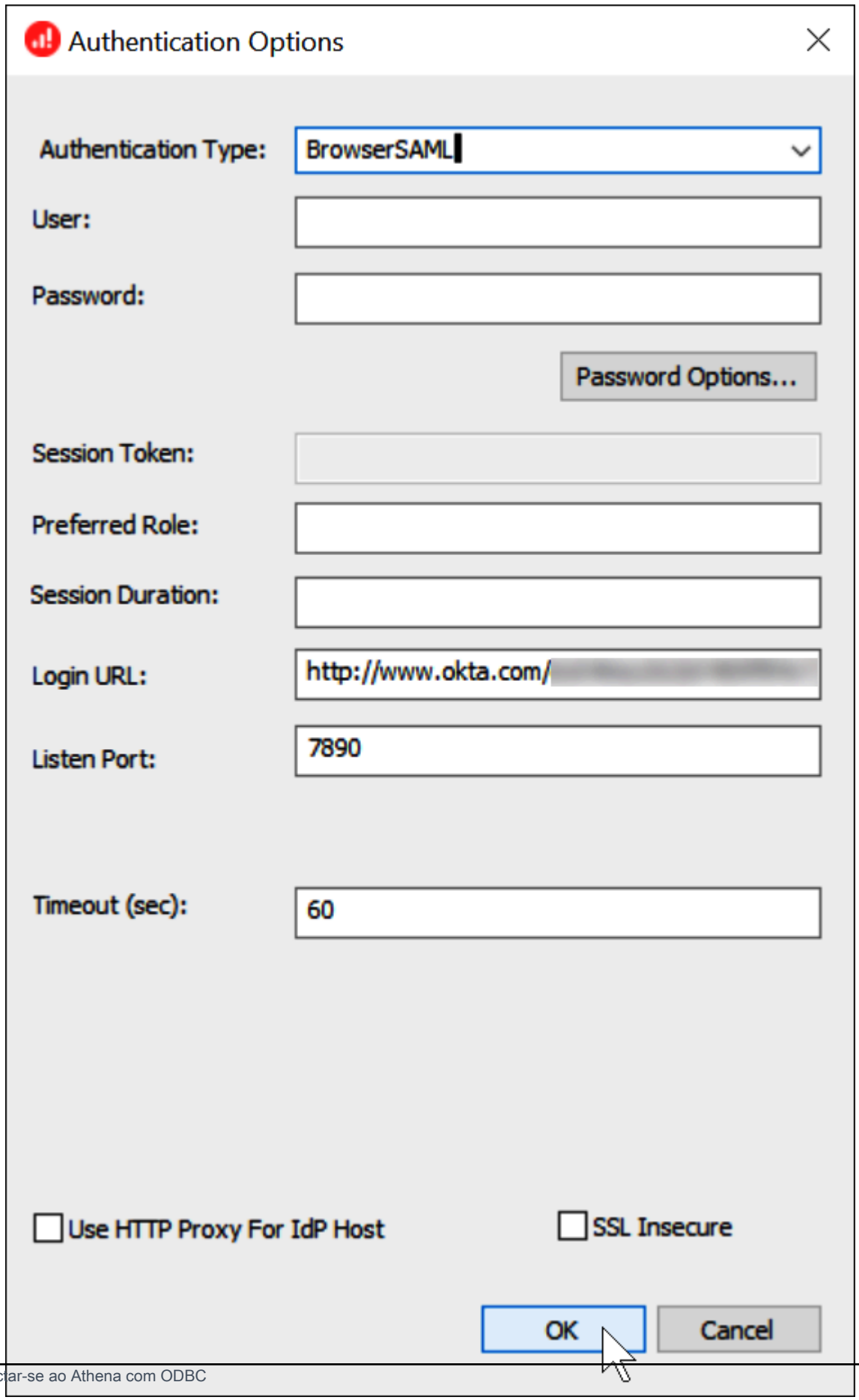


4. Na caixa de diálogo Simba Athena ODBC Driver DSN Setup (Configuração do driver DSN do Simba Athena), insira os valores descritos.



- Para Data Source Name (Nome da origem de dados), insira um nome para a origem dos dados (por exemplo, Athena ODBC 64).
- Em Description (Descrição), insira uma descrição para a origem dos dados.
- Para Região da AWS, insira a Região da AWS que você está usando (por exemplo, **`us-west-1`**).
- Para S3 Output Location, (Local de saída do S3), insira o caminho do Amazon S3 onde deseja que sua saída seja armazenada.

5. Escolha Authentication Options (Opções de autenticação).
6. Na caixa de diálogo Authentication Options (Opções de autenticação, escolha ou insira os valores a seguir.


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

- Para Authentication Type (Tipo de autenticação), escolha BrowserSAML.
- Para Login URL (URL de login), insira o Identity Provider Single Sign-On URL (URL de autenticação única do provedor de identidade) que você obteve no painel do Okta.
- Em Listen Port (Porta de escuta), insira 7890.
- Para Timeout (sec) (Tempo limite (s)), insira um valor de tempo limite de conexão em segundos.

7. Escolha OK para fechar as Authentication Options (Opções de autenticação).
8. Selecione Test (Testar) para testar a conexão ou OK para finalizar.

Usar o conector do Power BI para Amazon Athena

Nos sistemas operacionais Windows, você pode usar o conector do Microsoft Power BI para Amazon Athena para analisar os dados do Amazon Athena no Microsoft Power BI Desktop. Para obter informações sobre o Power BI, consulte Microsoft Power BI. Depois de publicar o conteúdo no serviço do Power BI, você pode usar a versão de julho de 2021 ou mais recente do gateway do Power BI para manter o conteúdo atualizado em operações sob demanda ou agendadas.

Pré-requisitos

Antes de começar, verifique se o seu ambiente atende aos requisitos a seguir. O driver ODBC do Amazon Athena é obrigatório.

- Conta da AWS
- Permissões para usar o Athena
- Driver ODBC para Amazon Athena
- Power BI Desktop

Recursos permitidos

- Importação: as tabelas e colunas selecionadas são importadas para o Power BI Desktop para consulta.
- DirectQuery: os dados não são importados nem copiados para o Power BI Desktop. O Power BI Desktop consulta diretamente a origem dos dados subjacente.

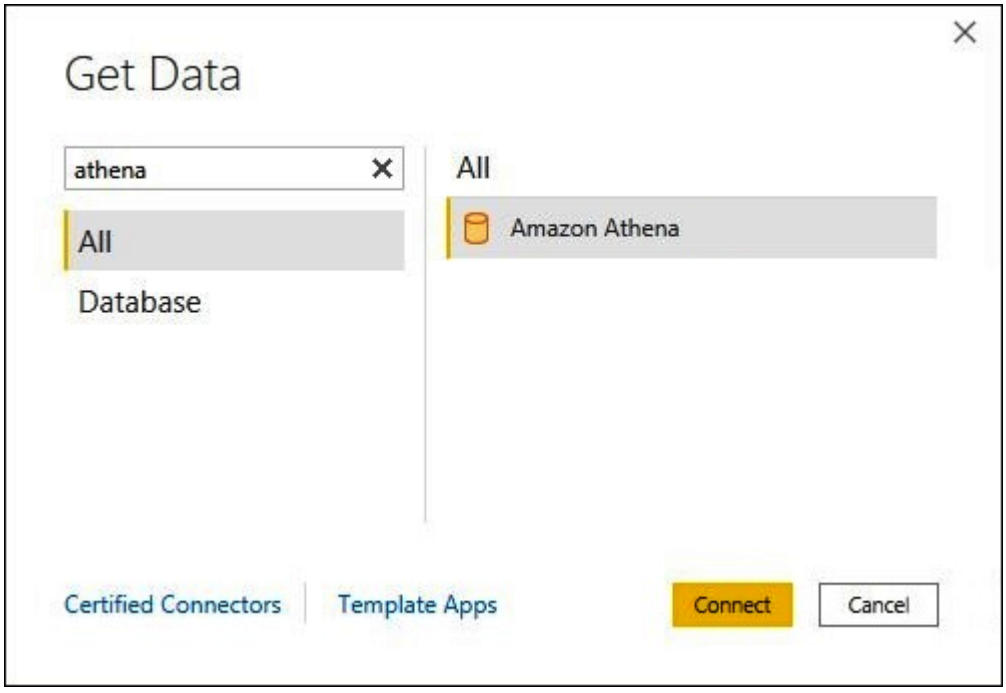
- Gateway do Power BI: um gateway de dados on-premises na Conta da AWS que funciona como uma ponte entre o serviço do Microsoft Power BI e o Athena. O gateway é necessário para ver seus dados no Serviço do Microsoft Power BI.

Conectar ao Amazon Athena

Para conectar o Power BI Desktop aos dados do Amazon Athena, siga as etapas abaixo.

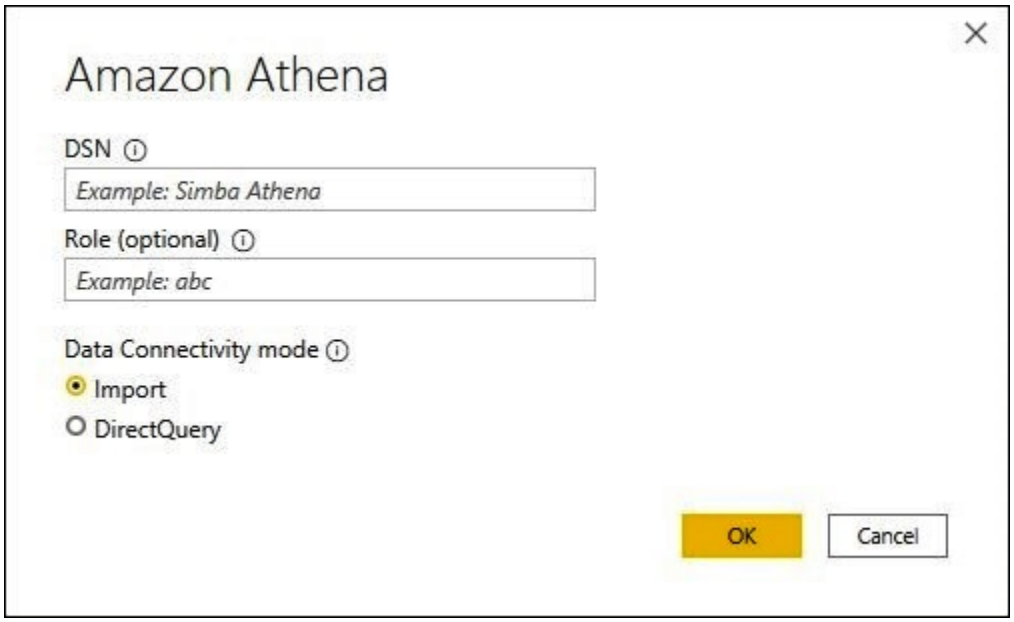
Para se conectar aos dados do Athena pelo Power BI Desktop

1. Inicie o Power BI Desktop.
2. Execute um destes procedimentos:
   - Selecione File (Arquivo), Get Data (Obter dados)
   - Na faixa de opções Home (Início), escolha Get Data (Obter dados).
3. Na caixa de pesquisa, digite Athena.
4. Selecione Amazon Athena e, depois, Connect (Conectar).

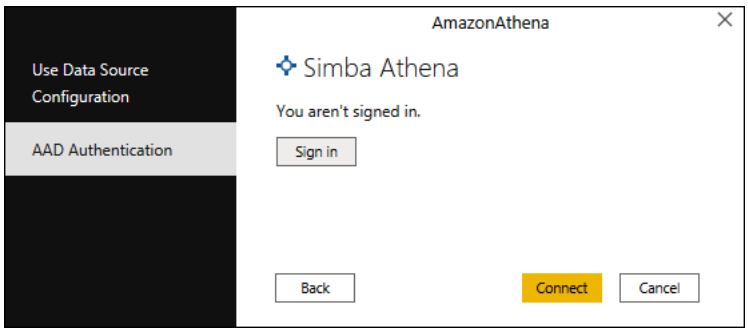


5. Na página de conexão do Amazon Athena, insira as informações a seguir.
   - Em DSN, digite o nome do DSN do ODBC que deseja usar. Para obter instruções sobre como configurar o DSN, consulte a Documentação do driver ODBC.

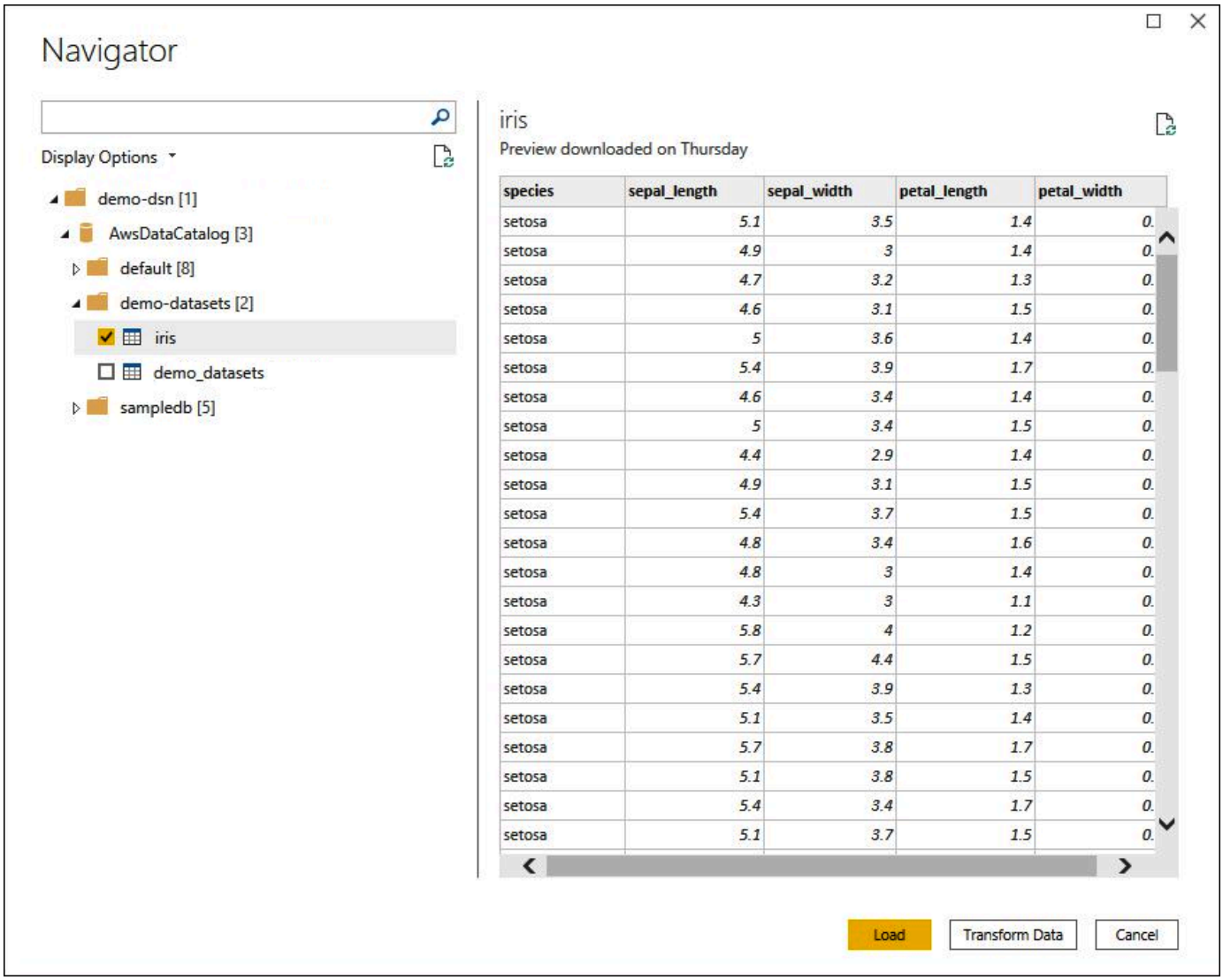
- Em Data Connectivity mode (Modo de conectividade de dados), escolha um modo apropriado ao seu caso de uso, seguindo estas diretrizes gerais:
  - Para conjuntos de dados menores, escolha Import (Importar). Ao usar o modo de importação, o Power BI trabalha com o Athena para importar o conteúdo de todo o conjunto de dados para uso nas suas visualizações.
  - Para conjuntos de dados maiores, escolha DirectQuery. No modo DirectQuery, os dados não são baixados na sua estação de trabalho. Enquanto você cria ou interage com uma visualização, o Microsoft Power BI trabalha com o Athena para consultar dinamicamente a origem dos dados subjacente de modo que você sempre veja os dados atuais. Para obter mais informações sobre o DirectQuery, consulte Use DirectQuery in Power BI desktop (Usar o DirectQuery no Power BI Desktop) na documentação da Microsoft.



6. Escolha OK.
7. No prompt para configurar a autenticação da origem dos dados, escolha Use Data Source Configuration (Usar configuração da origem dos dados) ou AAD Authentication (Configuração AAD) e, depois, Connect (Conectar).

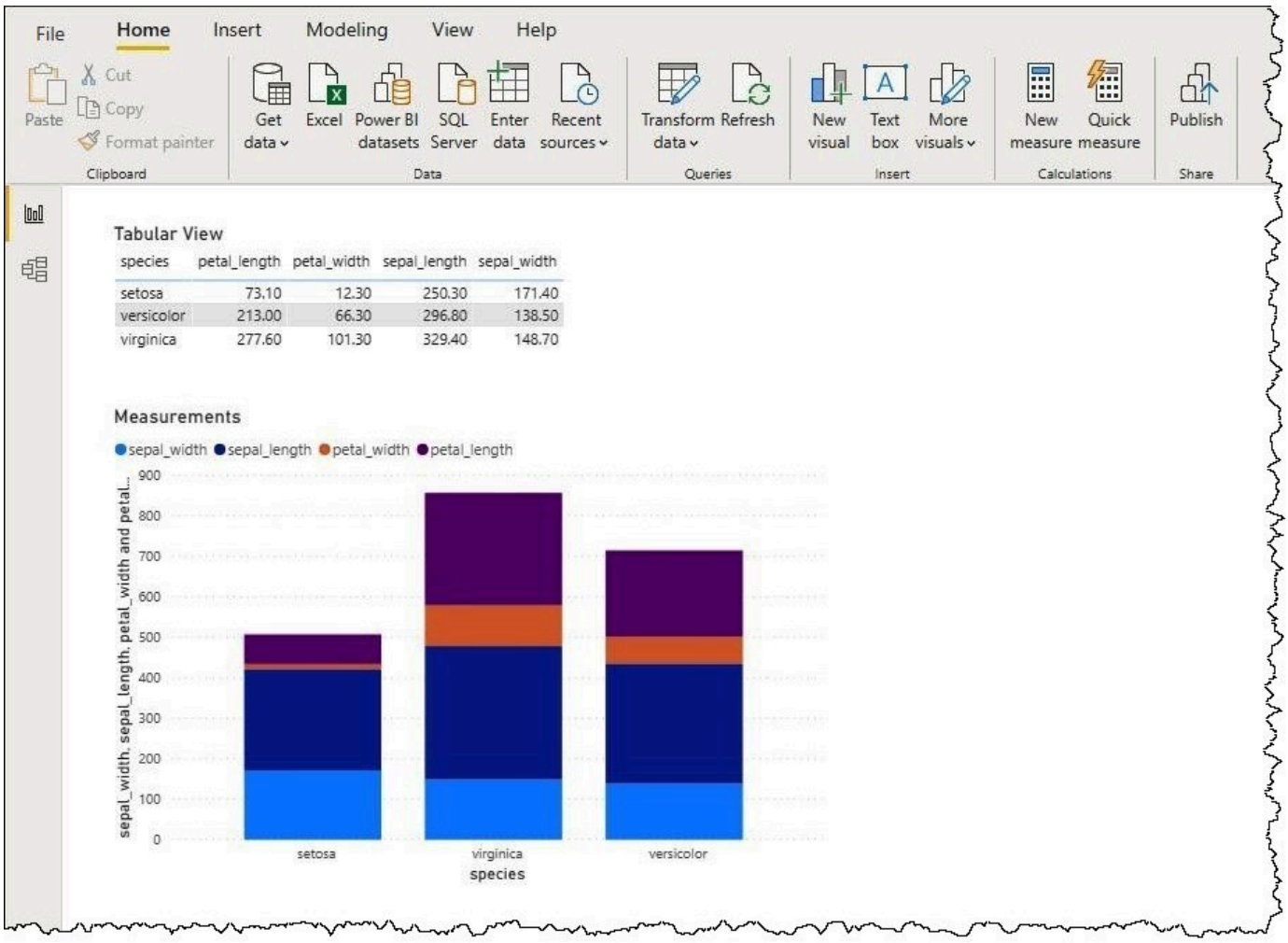


Seu catálogo de dados, seus bancos de dados e suas tabelas aparecem na caixa de diálogo Navigator (Navegador).



8. No painel Display Options (Opções de exibição), marque a caixa de seleção do conjunto de dados que deseja usar.

9. Se você quiser transformar o conjunto de dados antes de importá-lo, vá para a parte inferior da caixa de diálogo e escolha Transform Data (Transformar dados). Esse procedimento abre o Editor do Power Query para você filtrar e refinar o conjunto de dados que deseja usar.
10. Escolha Load. Após a conclusão do carregamento, você poderá criar visualizações conforme mostrado na imagem a seguir. Se você selecionou DirectQuery como o modo de importação, o Power BI emite uma consulta ao Athena para a visualização solicitada.



Configurar um gateway on-premises

Você pode publicar painéis e conjuntos de dados no serviço do Power BI para que outros usuários possam interagir com eles por meio de aplicações Web, móveis e incorporadas. Para ver seus dados no Serviço do Microsoft Power BI, instale o gateway de dados on-premises do Microsoft Power BI na sua Conta da AWS. O gateway funciona como uma ponte entre o Serviço do Microsoft Power BI e o Athena.

Para baixar, instalar e testar um gateway de dados on-premises

1. Acesse a página de download do gateway do Microsoft Power BI e escolha o modo pessoal ou padrão. O modo pessoal é útil para testar localmente o conector do Athena. O modo padrão é ideal em uma configuração de produção multiusuário.
2. Para instalar um gateway on-premises (modo pessoal ou padrão), consulte Instalar um gateway de dados local na documentação da Microsoft.
3. Para testar o gateway, siga as etapas em Usar conectores de dados personalizados com o gateway de dados local na documentação da Microsoft.

Para obter mais informações sobre gateways de dados on-premises, consulte os recursos da Microsoft a seguir.

- O que é um gateway de dados on-premises?
- Diretrizes para implantar um gateway de dados para o Power BI

Para ver um exemplo de configuração do gateway do Power BI para uso com o Athena, consulte o artigo do blog sobre big data da AWS Creating dashboards quickly on Microsoft Power BI using Amazon Athena (Criar painéis rapidamente no Microsoft Power BI usando o Amazon Athena).

Criar bancos de dados e tabelas

O Amazon Athena aceita um subconjunto de instruções Data Definition Language (DDL – Linguagem de definição de dados) e funções e operadores ANSI SQL para definir e consultar tabelas externas em que os dados residem no Amazon Simple Storage Service.

Ao criar um banco de dados e uma tabela no Athena, você descreve o esquema e o local dos dados, o que os prepara na tabela para consulta em tempo real.

Para melhorar a performance das consultas e reduzir os custos, recomendamos particionar os dados e usar formatos de coluna de código aberto para armazenamento no Amazon S3, como Apache Parquet ou ORC.

Tópicos

- Criar bancos de dados no Athena
- Criar tabelas no Athena

- Nomes de tabelas, bancos de dados e colunas
- Palavras-chave reservadas
- Local da tabela no Amazon S3
- Formatos de armazenamento colunar
- Converter em formatos colunares
- Particionar dados no Athena
- Projeção de partições com o Amazon Athena

Criar bancos de dados no Athena

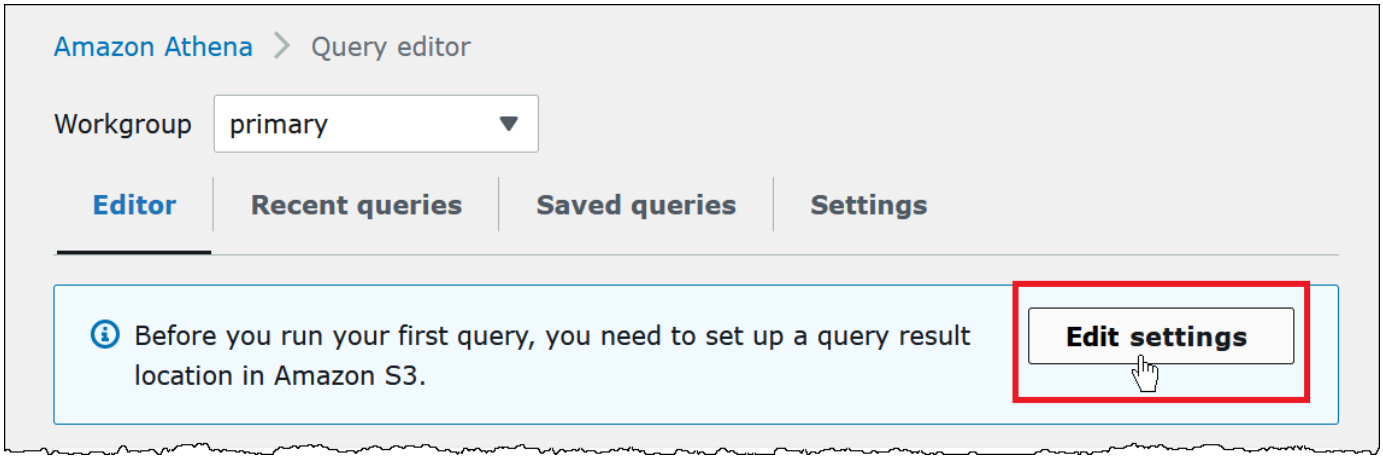
Um banco de dados no Athena é um agrupamento lógico das tabelas que você cria nele.

Pré-requisitos

Se você ainda não tiver um local de saída de consulta configurado no Amazon S3, execute as seguintes etapas de pré-requisito para fazer isso.

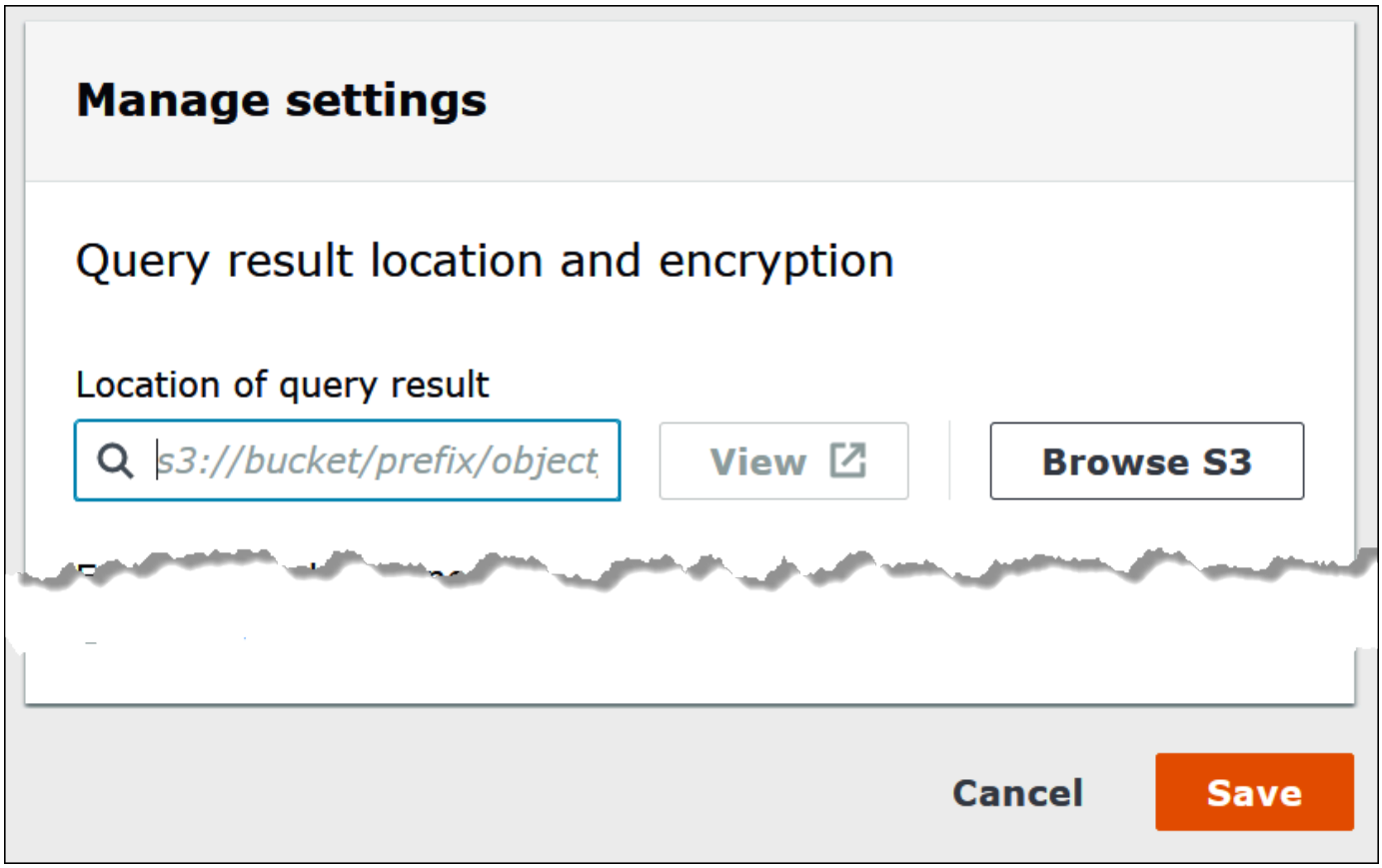
Para criar um local de saída da consulta

1. Usando a mesma Região da AWS e a conta que você usa no Athena, siga as etapas (por exemplo, usando o console Amazon S3) para criar um bucket no Amazon S3 para armazenar os resultados das consultas do Athena. Você configurará esse bucket para ser o local de saída da consulta.
2. Abra o console do Athena em https://console.aws.amazon.com/athena/.
3. Se esta for a primeira vez que você acessa o console do Athena neste Região da AWS, escolha Explore the query editor (Explorar o editor de consultas) para abrir o editor de consultas. Do contrário, o Athena é aberto no editor de consultas.
4. Escolha Edit Settings (Editar configurações) para configurar um local para os resultados de consultas no Amazon S3.

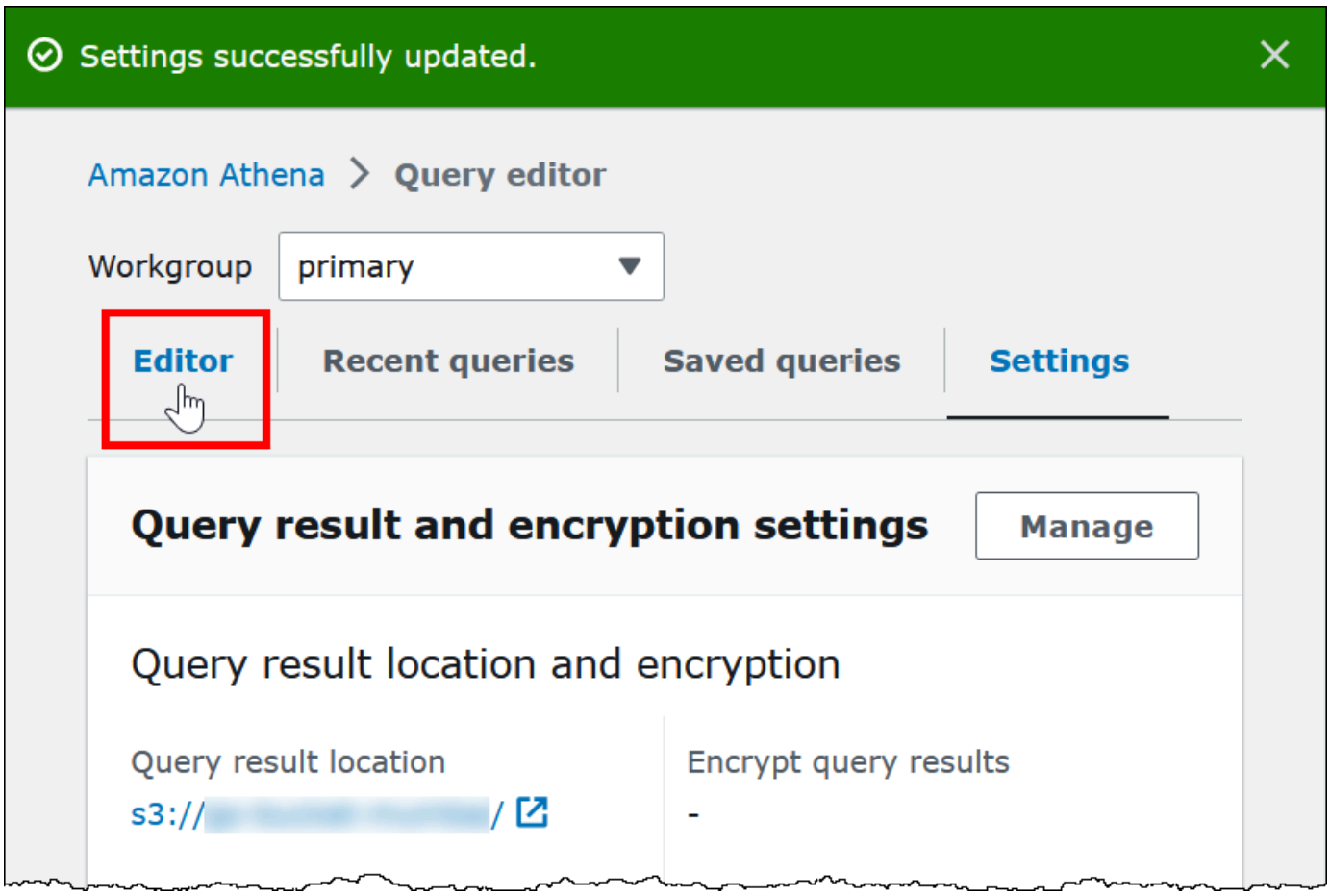


5. Em Manage settings (Gerenciar configurações), faça um dos seguintes procedimentos:

   - Na caixa de texto Query result location (Localização dos resultados da consulta), insira o caminho para o bucket criado no Amazon S3 para resultados de consultas. Adicione o prefixo `s3://` ao caminho.
   - Escolha Browse S3 (Navegar no S3), escolha o bucket do Amazon S3 que você criou na região atual e escolha Choose (Escolher).



6. Escolha Salvar.
7. Selecione Editor para alternar para o editor de consultas.



Criação de um banco de dados

Depois de configurar um local de resultados de consulta, criar um banco de dados no editor de consultas do console Athena é simples.

Para criar um banco de dados usando o editor de consultas do Athena

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Na guia Editor, insira o comando de linguagem de definição de idioma (DDL) `CREATE DATABASE` *`myDataBase`* do Hive. Substitua *`myDatabase`* pelo nome que você deseja usar. Para obter restrições sobre nomes de bancos de dados, consulte Nomes de tabelas, bancos de dados e colunas.
3. Selecione Run (Executar) ou pressione **`Ctrl+ENTER`**.

4. Para fazer com que seu banco de dados seja o banco de dados atual, selecione-o no menu Database (Banco de dados) à esquerda do editor de consultas.

Para obter informações sobre como controlar as permissões para bancos de dados do Athena, consulte Acesso granular a bancos de dados e tabelas no AWS Glue Data Catalog.

Criar tabelas no Athena

É possível executar instruções DDL no console do Athena usando um driver JDBC ou ODBC, ou usando o formulário Create table (Criar tabela) do Athena.

Quando você cria um esquema de tabela no Athena, o Athena armazena o esquema em um catálogo de dados e usa-o para executar consultas.

O Athena usa uma abordagem conhecida como schema-on-read, que significa que um esquema é projetado com base nos dados no momento em que você executa uma consulta. Isso elimina a necessidade de carregamento ou transformação de dados.

O Athena não modifica os dados no Amazon S3.

O Athena usa o Apache Hive para definir tabelas e criar bancos de dados, que são basicamente um namespace lógico de tabelas.

Ao criar um banco de dados e uma tabela no Athena, você apenas descreve o esquema e o local onde os dados da tabela estão localizados no Amazon S3 para consulta em tempo de leitura. Por isso, o banco de dados e a tabela têm um significado um pouco diferente do que o de sistemas de bancos de dados relacionais tradicionais porque os dados não são armazenados com a definição de esquema para o banco de dados e a tabela.

Ao consultar, você consulta a tabela usando SQL padrão e os dados são lidos neste momento. Para encontrar orientações sobre como criar bancos de dados e tabelas, consulte a Documentação do Apache Hive. Veja abaixo as orientações específicas para o Athena.

O tamanho máximo da string de consulta é 256 KB.

O Hive oferece suporte a vários formatos de dados por meio do uso de bibliotecas Serializer-Deserializer (SerDe – Serializador-desserializador). Você também pode definir esquemas complexos usando expressões regulares. Para obter uma lista de bibliotecas SerDe compatíveis, consulte SerDes e formatos de dados compatíveis.

Considerações e limitações

Estas são algumas limitações e considerações importantes sobre as tabelas no Athena.

Requisitos para tabelas no Athena e dados no Amazon S3

Ao criar uma tabela, você especifica o local de um bucket do Amazon S3 para os dados subjacentes usando a cláusula `LOCATION`. Considere o seguinte:

- O Athena pode consultar somente a versão mais recente dos dados em um bucket versionado do Amazon S3, não as versões anteriores.
- Você deve ter as permissões apropriadas para trabalhar com os dados no local do Amazon S3. Para ter mais informações, consulte Acesso ao Amazon S3.
- O Athena permite consultar objetos armazenados com várias classes de armazenamento no mesmo bucket especificado pela cláusula `LOCATION`. Por exemplo, é possível consultar dados em objetos armazenados em classes diferentes de armazenamento (Standard, Standard-IA e Intelligent-Tiering) no Amazon S3.
- O Athena é compatível com buckets de pagamento a cargo do solicitante. Para obter informações sobre como habilitar pagamento a cargo do solicitante para buckets com dados de origem que você pretende consultar no Athena, consulte Criar um grupo de trabalho.
- O Athena não permite consulta de dados nas classes de armazenamento S3 Glacier Flexible Retrieval ou S3 Glacier Deep Archive. Objetos nas classes de armazenamento S3 Glacier Flexible Retrieval e S3 Glacier Deep Archive são ignorados. Como alternativa, você pode usar a classe de armazenamento Amazon S3 Glacier Instant Retrieval, que pode ser consultada pelo Athena. Para obter mais informações, consulte Amazon S3 Glacier Instant Retrieval storage class (Classe de armazenamento Amazon S3 Glacier Instant Retrieval).

  Para obter informações sobre classes de armazenamento, consulte Classes de armazenamento, Alterar a classe de armazenamento de um objeto no Amazon S3, Transição para a classe de armazenamento GLACIER (arquivamento de objetos) e Buckets de pagamento a cargo do solicitante no Guia do usuário do Amazon Simple Storage Service.
- Se você realizar consultas nos buckets do Amazon S3 com um grande número de objetos, e os dados não estiverem particionados, essas consultas poderão afetar os limites de taxa de solicitações Get no Amazon S3 e gerar exceções no Amazon S3. Para evitar erros, particione seus dados. Considere também ajustar suas taxas de solicitações do Amazon S3. Para obter mais informações, consulte Taxas de solicitações e considerações de performance.

- Se você usar a operação de API do AWS Glue CreateTable ou o modelo `AWS::Glue::Table` do AWS CloudFormation para criar uma tabela para uso no Athena sem especificar a propriedade `TableType` e, depois, executar uma consulta DDL, como `SHOW CREATE TABLE` ou `MSCK REPAIR TABLE`, poderá receber a mensagem de erro FALHA: o nome de NullPointerException é nulo.

  Para resolver o erro, especifique um valor para o atributo TableInput `TableType` como parte da chamada de API `CreateTable` do AWS Glue ou do modelo do AWS CloudFormation. Os valores possíveis para `TableType` são `EXTERNAL_TABLE` ou `VIRTUAL_VIEW`.

  Esse requisito é aplicado somente quando você cria uma tabela usando a operação de API do AWS Glue `CreateTable` ou o modelo do `AWS::Glue::Table`. Se você criar uma tabela do Athena usando uma instrução DDL ou um crawler do AWS Glue, a propriedade `TableType` será definida automaticamente para você.

Funções compatíveis

As funções com suporte nas consultas do Athena correspondem às do Trino e do Presto. Para obter mais informações sobre funções individuais consulte a seção de funções e operadores na documentação do Trino ou do Presto.

Transformações de dados transacionais não são compatíveis

O Athena não permite operações baseadas em transações (como as disponíveis no Hive ou no Presto) nos dados de tabelas. Para obter uma lista completa de palavras-chave não compatíveis, consulte DDL incompatível.

Operações que alteram estados de tabela são ACID

Quando você cria, atualiza ou exclui tabelas, essas operações têm compatibilidade com ACID garantida. Por exemplo, se vários usuários ou clientes tentarem criar ou alterar uma tabela existente ao mesmo tempo, somente um será bem-sucedido.

As tabelas são EXTERNAL

Exceto ao criar tabelas do Iceberg, use sempre a palavra-chave `EXTERNAL`. Se você usar `CREATE TABLE` sem a palavra-chave `EXTERNAL` para tabelas que não são do Iceberg, o Athena emitirá um erro. Quando você descarta uma tabela no Athena, apenas os metadados da tabela são removidos, os dados permanecem no Amazon S3.

Criar tabelas usando o AWS Glue ou o console do Athena

É possível criar tabelas no Athena usando o AWS Glue, o formulário para adicionar tabela ou executando uma instrução DDL no editor de consultas do Athena.

Para criar uma tabela usando o crawler do AWS Glue

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. No editor de consultas, ao lado de Tabelas e visualizações, escolha Criar e, em seguida, selecione Crawler do AWS Glue.
3. Siga as etapas contidas na página Add crawler (Adicionar crawler) do console do AWS Glue para adicionar um crawler.

   Para ter mais informações, consulte Usar crawlers do AWS Glue.

Para criar uma tabela usando o formulário para criar tabela do Athena

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. No editor de consultas, ao lado de Tables and views (Tabelas e visualizações), escolha Create (Criar) e, em seguida, escolha S3 bucket data (Dados do bucket do S3).
3. No formulário Create Table From S3 bucket data (Criar tabela com base em dados de bucket do S3), insira as informações para criar sua tabela e escolha Create table (Criar tabela). Para obter mais informações sobre os campos no formulário, consulte Adicionar uma tabela usando um formulário.

Para criar uma tabela usando a DDL do Hive

1. No menu Database (Banco de dados), escolha o banco de dados para o qual deseja criar uma tabela. Se você não especificar um banco de dados na instrução `CREATE TABLE`, a tabela será criada no banco de dados atualmente selecionado no editor de consultas.
2. Insira uma instrução como a seguinte no editor de consultas e, em seguida, escolha Run (Executar) ou pressione **`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/';
```

Mostrar informações da tabela

Depois de criar uma tabela no Athena, o nome dela é exibido na lista Tables (Tabelas) à esquerda. Para exibir informações sobre a tabela e gerenciá-la, escolha os três pontos verticais ao lado do nome da tabela no console do Athena.

- Preview table (Visualizar tabela): mostra as primeiras dez linhas de todas as colunas executando a instrução `SELECT * FROM "database_name"."table_name" LIMIT 10` no editor de consultas do Athena.
- Generate table DDL (Gerar DDL de tabela): gera uma instrução DDL que você pode usar para recriar a tabela executando a instrução `SHOW CREATE TABLE nome_da_tabela` no editor de consultas do Athena.
- Load partitions (Carregar partições): executa a instrução `MSCK REPAIR TABLE table_name` no editor de consultas do Athena. Essa opção só estará disponível se a tabela tiver partições.
- Insert into editor (Inserir no editor): insere o nome da tabela no editor de consultas no local de edição atual.
- Delete table (Excluir tabela): exibe uma caixa de diálogo de confirmação perguntando se você deseja excluir a tabela. Se você concordar, a instrução `DROP TABLE table_name` será executada no editor de consultas do Athena.
- Table properties (Propriedades da tabela): mostra o nome da tabela, o nome do banco de dados, o horário de criação e se a tabela tem dados criptografados.

Nomes de tabelas, bancos de dados e colunas

Use essas dicas para nomear objetos de banco de dados no Athena.

Requisitos para nome de bancos de dados, tabelas e colunas

- Os caracteres aceitáveis para nomes de bancos de dados, nomes de tabelas e nomes de colunas no AWS Glue devem ser strings em UTF-8. A string não pode ter menos do que 1 ou mais de 255 bytes de comprimento. Exceder esse limite gera um erro como O valor em 'name' falhou em satisfazer a restrição: o membro deve ter um comprimento menor ou igual a 255. Os caracteres que podem ser usados incluem espaços e são definidos pelo seguinte padrão de string de linha única:

```
[\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\t]*
```

- Atualmente, o padrão regex AWS Glue permite a adição de espaços iniciais ao início dos nomes. Como pode ser difícil detectar esses espaços iniciais e eles podem causar problemas de usabilidade após a criação, evite criar nomes de objetos que contenham espaços iniciais.
- Se você usar um modelo AWS::Glue::Database do AWS CloudFormation para criar um banco de dados do AWS Glue e não especificar um nome de banco de dados, o AWS Glue gera automaticamente um nome de banco de dados no formato *resource_name–random_string* que não é compatível com o Athena.
- Você pode usar o Gerenciador de Catálogos do AWS Glue para renomear colunas, mas não nomes de tabelas ou nomes de banco de dados. Para resolver essa limitação, é necessário usar uma definição do banco de dados antigo para criar um banco de dados com o novo nome. Em seguida, use as definições das tabelas do banco de dados antigo para recriar as tabelas no novo banco de dados. Para isso, você pode usar a AWS CLI ou o SDK do AWS Glue. Para obter as etapas, consulte Usar a AWS CLI para recriar um banco de dados do AWS Glue e suas tabelas.

Usar letras minúsculas nos nomes de tabela e de coluna da tabela no Athena

O Athena aceita maiúsculas e minúsculas nas consultas DDL e DML, mas usa letras minúsculas nos nomes quando executa a consulta. Por essa razão, evite usar maiúsculas e minúsculas em nomes de tabelas ou colunas, e não confie no uso de letras maiúsculas e minúsculas do Athena para distinguir esses nomes. Por exemplo, se você usar uma instrução DDL para criar uma coluna chamada `Castle`, a coluna criada será mudada para letras minúsculas `castle`. Se você especificar o nome da coluna em uma consulta DML como `Castle` ou `CASTLE`, o Athena mudará o nome para

letras minúsculas para você executar a consulta, mas exibirá o cabeçalho da coluna da forma como você escolheu na consulta.

Os nomes de bancos de dados, tabelas e colunas ter, no máximo, 255 caracteres.

Nomes que começam com um sublinhado

Ao criar tabelas, coloque entre acentos graves os nomes de tabelas, visualizações ou colunas que começam com sublinhado. Por exemplo:

```
CREATE EXTERNAL TABLE IF NOT EXISTS `_myunderscoretable`(
  `_id` string, `_index` string)
LOCATION 's3://DOC-EXAMPLE-BUCKET/'
```

Nomes de tabelas, visualizações ou colunas que começam com números

Ao executar consultas `SELECT`, `CTAS` ou `VIEW`, coloque entre aspas os identificadores, como nomes de tabelas, visualizações ou colunas, que começam com um dígito. Por exemplo:

```
CREATE OR REPLACE VIEW "123view" AS
SELECT "123columnone", "123columntwo"
FROM "234table"
```

Nomes de colunas e tipos complexos

Para tipos complexos, apenas caracteres alfanuméricos, sublinhado (_) e ponto final (`.`) são permitidos nos nomes de colunas. Para criar uma tabela e mapeamentos para chaves que têm caracteres restritos, você pode usar uma instrução DDL personalizada. Para obter mais informações, consulte o artigo Create tables in Amazon Athena from nested JSON and mappings using JSONSerDe (Criar tabelas no Amazon Athena de JSON e mapeamentos aninhados usando JSONSerDe) no blog sobre big data da AWS.

Palavras reservadas

Algumas palavras reservadas no Athena devem ser escapadas. Para inserir um caractere de escape em palavras-chave reservadas em instruções DDL, coloque-as entre acentos graves (`). Para usar escape em palavras-chave reservadas em instruções SQL `SELECT` e em consultas em visualizações, coloque-as entre aspas duplas (").

Para ter mais informações, consulte Palavras-chave reservadas.

Recursos adicionais do

Para obter a sintaxe completa de criação de bancos de dados e tabelas, consulte as páginas a seguir.

- CREATE DATABASE
- CREATE TABLE

Para obter mais informações sobre bancos de dados e tabelas no AWS Glue, consulte Bancos de dados e Tabelas no Guia do desenvolvedor do AWS Glue.

Palavras-chave reservadas

Quando você executa consultas no Athena que incluem palavras-chave reservadas, deve escapá-las entre caracteres especiais. Use as listas deste tópico para verificar quais são as palavras-chave reservadas no Athena.

Para inserir um caractere de escape em palavras-chave reservadas em instruções DDL, coloque-as entre acentos graves (`). Para usar escape em palavras-chave reservadas em instruções SQL `SELECT` e em consultas em visualizações, coloque-as entre aspas duplas (").

- Lista de palavras-chave reservadas em instruções DDL
- Lista de palavras-chave reservadas em instruções SQL SELECT
- Exemplos de consultas com palavras reservadas

Lista de palavras-chave reservadas em instruções DDL

O Athena usa a lista de palavras-chave reservadas a seguir em suas instruções DDL. Se você usá-las sem efetuar escape, o Athena emitirá um erro. Para inserir um caractere de escape nelas, coloque-as entre acentos graves (`).

Você não pode usar palavras-chave reservadas do DDL como nomes de identificadores em instruções DDL sem colocá-las entre acentos graves (`).

```
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
```

Lista de palavras-chave reservadas em instruções SQL SELECT

O Athena usa a lista de palavras-chave reservadas a seguir nas instruções SQL `SELECT` e nas consultas em visualizações.

Se você usar essas palavras-chave como identificadores, deverá colocá-las entre aspas duplas (") em suas instruções de consulta.

```
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
```

Exemplos de consultas com palavras reservadas

A consulta no exemplo a seguir usa acentos graves (`) para efetuar escape das palavras-chave reservadas partition e date relacionadas à DDL que são usadas para um nome de tabela e um dos nomes de coluna:

```
CREATE EXTERNAL TABLE `partition` (
`date` INT,
col2 STRING
)
PARTITIONED BY (year STRING)
STORED AS TEXTFILE
LOCATION 's3://DOC-EXAMPLE-BUCKET/test_examples/';
```

As consultas de exemplo a seguir incluem um nome de coluna que contém as palavras-chave reservadas relacionadas ao DDL em instruções `ALTER TABLE ADD PARTITION` e `ALTER TABLE DROP PARTITION`. As palavras-chave reservadas do DDL são circunscritas com acentos graves (`):

```
ALTER TABLE test_table
ADD PARTITION (`date` = '2018-05-14')
```

```
ALTER TABLE test_table
DROP PARTITION (`partition` = 'test_partition_value')
```

A consulta de exemplo a seguir inclui uma palavra-chave reservada (end) como um identificador em uma instrução `SELECT`. O escape da palavra-chave é efetuado com aspas duplas:

```
SELECT *
FROM TestTable
WHERE "end" != nil;
```

A consulta de exemplo a seguir inclui uma palavra-chave reservada (first) em uma instrução `SELECT`. O escape da palavra-chave é efetuado com aspas duplas:

```
SELECT "itemId"."first"
FROM testTable
LIMIT 10;
```

Local da tabela no Amazon S3

Quando você executa uma consulta `CREATE TABLE` no Athena, ele registra a tabela no Catálogo de Dados do AWS Glue, que é onde o Athena armazena os metadados.

Para especificar o caminho para os dados no Amazon S3, use a propriedade `LOCATION` conforme mostrado no seguinte exemplo:

```
CREATE EXTERNAL TABLE `test_table`(
...
)
ROW FORMAT ...
STORED AS INPUTFORMAT ...
OUTPUTFORMAT ...
LOCATION s3://DOC-EXAMPLE-BUCKET/folder/
```

- Para obter informações sobre a nomenclatura de buckets, consulte Restrições e limitações do bucket no Guia do usuário do Amazon Simple Storage Service.
- Para obter informações sobre o uso de pastas no Amazon S3, consulte Usar pastas no Guia do usuário do Amazon Simple Storage Service.

O `LOCATION` no Amazon S3 especifica todos os arquivos que representam sua tabela.

> **Important**
>
> O Athena lê todos os dados armazenados na pasta do Amazon S3 que você especificar. Se você tem dados que não deseja que o Athena leia, não os armazene na mesma pasta do Amazon S3 que os dados que deseja que o Athena leia. Se você usa particionamento, para garantir que o Athena verifique os dados dentro de uma partição, o filtro `WHERE` deve incluir a partição. Para ter mais informações, consulte Local e partições de tabela.

Ao especificar o `LOCATION` na instrução `CREATE TABLE`, use as seguintes diretrizes:

- Use uma barra no final.
- Você pode usar um caminho para uma pasta do Amazon S3 ou um alias de ponto de acesso do Amazon S3. Para obter informações sobre aliases de ponto de acesso do Amazon S3, consulte Usar um alias em estilo de bucket para seu ponto de acesso no Manual do usuário do Amazon S3.

Use:

```
s3://DOC-EXAMPLE-BUCKET/folder/
```

```
s3://DOC-EXAMPLE-BUCKET-metadata-s3alias/folder/
```

Não use nenhum dos itens a seguir para especificar a `LOCATION` dos dados.

- Não use nomes de arquivo, sublinhado, curingas ou padrões glob para especificar locais de arquivos.
- Não adicione a notação HTTP completa, como `s3.amazon.com` ao caminho do bucket do Amazon S3.
- Não use pastas vazias, como `//`, no caminho, conforme segue: `S3://DOC-EXAMPLE-BUCKET/folder//folder/`. Esse é um caminho válido do Amazon S3, mas o Athena não o permite e o altera para `s3://DOC-EXAMPLE-BUCKET/folder/folder/` removendo o `/` extra.

  Não use:

```
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
```

Local e partições de tabela

Seus dados de origem podem ser agrupados em pastas do Amazon S3 chamadas partições com base em um conjunto de colunas. Por exemplo, essas colunas podem representar o ano, o mês e o dia em que o registro em particular foi criado.

Ao criar uma tabela, você pode optar por torná-la particionada. Quando o Athena executa uma consulta SQL em uma tabela não particionada, ele usa a propriedade `LOCATION` da definição da tabela como o caminho base para listar e verificar todos os arquivos disponíveis. No entanto, para que uma tabela particionada possa ser consultada, você deve atualizar o Catálogo de dados do AWS Glue com informações da partição. Essas informações representam o esquema de arquivos na partição específica e o `LOCATION` dos arquivos no Amazon S3 para a partição.

- Para saber como o crawler do AWS Glue adiciona partições, consulte Como um crawler determina quando criar partições? no Guia do desenvolvedor do AWS Glue.

- Para aprender a configurar o crawler para criar tabelas para dados em partições existentes, consulte Usar várias origens de dados com crawlers.
- Você também pode criar partições em uma tabela diretamente no Athena. Para ter mais informações, consulte Particionar dados no Athena.

Quando o Athena executa uma consulta em uma tabela particionada, ele verifica se as colunas particionadas foram usadas na cláusula `WHERE` da consulta. Em caso afirmativo, o Athena solicita ao Catálogo de dados do AWS Glue para retornar a especificação de partição correspondente às colunas da partição especificada. A especificação de partição inclui a propriedade `LOCATION`, que indica ao Athena qual prefixo do Amazon S3 usar ao ler os dados. Nesse caso, somente os dados armazenados nesse prefixo são verificados. Se você não usar colunas particionadas na cláusula `WHERE`, o Athena verifica todos os arquivos que pertencem às partições da tabela.

Para ver exemplos de como usar o particionamento com o Athena para melhorar o desempenho e reduzir os custos das consultas, leia As dez melhores dicas para ajustar o desempenho do Amazon Athena.

Formatos de armazenamento colunar

Apache Parquet e ORC são formatos de armazenamento colunar otimizados para recuperação rápida de dados usados em aplicações de análises da AWS.

Os formatos de armazenamento colunar têm as seguintes características que os tornam adequados para o uso com o Athena:

- Compactação por coluna, com algoritmo de compactação selecionado para o tipo de dados da coluna para economizar espaço de armazenamento no Amazon S3 e reduzir E/S e espaço no disco durante o processamento de consultas.
- A aplicação de predicados em Parquet e ORC permite que as consultas do Athena busquem somente os blocos de que precisa, melhorando a performance das consultas. Quando uma consulta do Athena obtém valores de coluna específicos dos seus dados, ela usa as estatísticas dos predicados de bloco de dados, como valores máximos e mínimos, para determinar se é para ler ou ignorar o bloco.
- A divisão de dados em Parquet e ORC permite que o Athena divida a leitura de dados entre vários leitores e aumente o paralelismo durante o processamento da consulta.

Para converter seus dados brutos existentes de outros formatos de armazenamento em Parquet ou ORC, é possível executar consultas CREATE TABLE AS SELECT (CTAS) no Athena e especificar um formato de armazenamento de dados como Parquet ou ORC, ou usar o crawler do AWS Glue.

Escolher entre Parquet e ORC

A escolha entre ORC (Optimized Row Columnar) e Parquet depende de seus requisitos específicos de uso.

O Apache Parquet fornece esquemas eficientes de compactação e codificação de dados e é ideal para executar consultas complexas e processar grandes quantidades de dados. O Parquet é otimizado para uso com o Apache Arrow, o que pode ser vantajoso se você usar ferramentas relacionadas ao Arrow.

O ORC fornece uma maneira eficiente de armazenar dados do Hive. Os arquivos ORC geralmente são menores do que os arquivos Parquet, e os índices ORC podem agilizar as consultas. Além disso, o ORC é compatível com tipos complexos, como estruturas, mapas e listas.

Ao escolher entre Parquet e ORC, considere o seguinte:

Desempenho da consulta: o Parquet é compatível com uma variedade maior de tipos de consulta, por isso pode ser a melhor opção se você planeja realizar consultas complexas.

Tipos de dados complexos: se você estiver usando tipos de dados complexos, o ORC pode ser a melhor opção, porque ele é compatível com uma variedade maior de tipos de dados complexos.

Tamanho do arquivo: se o espaço em disco for uma preocupação, o ORC geralmente resulta em arquivos menores, o que pode reduzir os custos de armazenamento.

Compactação: tanto o Parquet quanto o ORC oferecem boa compactação, mas o melhor formato para você pode depender do seu caso específico de uso.

Evolução: tanto o Parquet quanto o ORC são compatíveis com uma evolução do esquema, o que significa que você pode adicionar, remover ou modificar colunas ao longo do tempo.

Tanto o Parquet quanto o ORC são boas opções para aplicações de big data, mas considere os requisitos do seu cenário antes de escolher. Talvez você queira realizar avaliações comparativas em seus dados e consultas para ver qual formato funciona melhor para seu caso de uso.

Converter em formatos colunares

A performance da consulta do Amazon Athena melhora quando os dados são convertidos em formatos colunares de código aberto, como Apache Parquet ou ORC.

As opções para converter facilmente dados de origem, como JSON ou CSV, em um formato colunar, incluem o uso de consultas CREATE TABLE AS ou a execução de trabalhos no AWS Glue.

- Você pode usar consultas (CTAS) `CREATE TABLE AS` para converter dados em Parquet ou ORC em uma única etapa. Para ver um exemplo, consulte Exemplo: gravar resultados da consulta em um formato diferente na página Exemplos de consultas CTAS.
- Para obter informações sobre como executar um trabalho do AWS Glue para transformar dados CSV em Parquet, consulte a seção “Transformar dados CSV em formato Parquet” na publicação Construir a fundação de um data lake com o AWS Glue e o Amazon S3 no blog sobre big data da AWS. O AWS Glue oferece suporte à mesma técnica para converter dados CSV em ORC ou dados JSON em Parquet ou ORC.

Particionar dados no Athena

Ao particionar os dados, você pode restringir a quantidade que cada consulta verifica, o que melhora a performance e reduz o custo. Você pode dividir seus dados em partições usando qualquer chave. Uma prática comum é particionar os dados com base no tempo, normalmente acarretando um esquema de particionamento em vários níveis. Por exemplo, um cliente que tenha dados vindos a cada hora pode optar por particionar por ano, mês, data e hora. Outro cliente, que tem dados oriundos de muitas origens diferentes, mas carregados apenas uma vez por dia, poderia particionar por um identificador de origem dos dados e data.

O Athena pode usar partições no estilo do Apache Hive, cujos caminhos de dados contêm pares de valor-chave conectados por sinais de igual (por exemplo, `country=us/...` ou `year=2021/month=01/day=26/...`). Assim, os caminhos incluem os nomes das chaves de partição e os valores que cada caminho representa. Para carregar novas partições Hive em uma tabela particionada, você pode usar o MSCK REPAIR TABLE, que funciona apenas com partições no estilo Hive.

O Athena também pode usar esquemas de particionamento em estilo não Hive. Por exemplo, os logs do CloudTrail e os fluxos de entrega do Firehose usam componentes de caminho separados para componentes de data, como `data/2021/01/26/us/6fc7845e.json`. Para essas partições

que não seguem o estilo do Hive, use ALTER TABLE ADD PARTITION para adicionar as partições manualmente.

Considerações e limitações

Ao usar o particionamento, lembre-se dos seguintes pontos:

- Se você consultar uma tabela particionada e especificar a partição na cláusula `WHERE`, o Athena verificará somente os dados dessa partição. Para ter mais informações, consulte Local e partições de tabela.
- Se você executar consultas em buckets do Amazon S3 com um grande número de objetos, e os dados não estiverem particionados, essas consultas poderão afetar os limites de taxa de solicitações `GET` no Amazon S3 e gerar exceções no Amazon S3. Para evitar erros, particione seus dados. Considere também ajustar suas taxas de solicitações do Amazon S3. Para obter mais informações, consulte Padrões de design de melhores práticas: otimizar a performance do Amazon S3.
- Os locais das partições que serão usados com o Athena devem aplicar o protocolo do `s3` (por exemplo, `s3://DOC-EXAMPLE-BUCKET/folder/`). No Athena, os locais que usam outros protocolos (por exemplo, `s3a://DOC-EXAMPLE-BUCKET/folder/`) resultam em falhas nas consultas `MSCK REPAIR TABLE` quando elas são executadas nas tabelas que os contêm.
- Verifique se o caminho do Amazon S3 está em letras minúsculas, em vez de maiúsculas e minúsculas concatenadas (por exemplo, `userid` em vez de `userId`). Se o caminho do S3 estiver em maiúsculas e minúsculas concatenadas, o `MSCK REPAIR TABLE` não adicionará as partições ao AWS Glue Data Catalog. Para ter mais informações, consulte MSCK REPAIR TABLE.
- Como `MSCK REPAIR TABLE` verifica uma pasta e as subpastas para encontrar um esquema de partição correspondente, mantenha os dados das tabelas separadas em hierarquias de pastas separadas. Por exemplo, suponha que você tenha dados na tabela 1 em `s3://DOC-EXAMPLE-BUCKET1` e dados na tabela 2 em `s3://DOC-EXAMPLE-BUCKET1/table-2-data`. Se ambas as tabelas forem particionadas por string, `MSCK REPAIR TABLE` adicionará as partições da tabela 2 à tabela 1. Para evitar isso, use estruturas de pastas separadas, como `s3://DOC-EXAMPLE-BUCKET1` e `s3://DOC-EXAMPLE-BUCKET2`. Observe que esse comportamento é consistente com o Amazon EMR e o Apache Hive.
- Se estiver usando o AWS Glue Data Catalog com o Athena, consulte Endpoints e cotas do AWS Glue para obter informações sobre cotas de serviço em partições por conta e por tabela.
  - Embora o Athena ofereça suporte a consulta a tabelas do AWS Glue com 10 milhões de partições, o Athena não pode ler mais de 1 milhão de partições em uma única varredura. Nesses

cenários, a indexação de partições pode ser benéfica. Para obter mais informações, consulte o artigo do blog Big Data da AWS, Melhore o desempenho das consultas do Amazon Athena usando índices de partição do AWS Glue Data Catalog.

- Para solicitar um aumento de cota de partições, se estiver usando o AWS Glue Data Catalog, acesse o console do Service Quotas para o AWS Glue.

Criar e carregar uma tabela com dados particionados

Para criar uma tabela que use partições, use a cláusula `PARTITIONED BY` na sua instrução CREATE TABLE. A cláusula `PARTITIONED BY` define as chaves usadas para particionar dados, como no exemplo a seguir. A cláusula `LOCATION` especifica o local raiz dos dados particionados.

```
CREATE EXTERNAL TABLE users (
first string,
last string,
username string
)
PARTITIONED BY (id string)
STORED AS parquet
LOCATION 's3://DOC-EXAMPLE-BUCKET'
```

Depois de criar a tabela, carregue os dados nas partições para consulta. Para partições no estilo do Hive, você executa MSCK REPAIR TABLE. Para partições que não seguem o estilo do Hive, use ALTER TABLE ADD PARTITION para adicionar as partições manualmente.

Preparar dados em estilo Hive e não Hive para consulta

As seções a seguir mostram como preparar dados de estilo do Hive e de estilo não Hive para consulta no Athena.

Cenário 1: dados armazenados no Amazon S3 no formato Hive

Nesse cenário, as partições são armazenadas em pastas separadas no Amazon S3. Por exemplo, aqui está a listagem parcial para exemplos de impressões de anúncio exibidas pelo `aws s3 ls`, que lista os objetos do S3 sob um prefixo especificado:

```
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/
```

Aqui, os logs são armazenados com o nome da coluna (dt) definido igual a incrementos de data, hora e minuto. Quando especifica o local da pasta pai, o esquema e o nome da coluna particionada em uma DDL, o Athena pode consultar os dados nessas subpastas.

Criar a tabela

Para gerar uma tabela com esses dados, crie uma partição com “dt”, como na seguinte instrução DDL do Athena:

```
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/' ;
```

Esta tabela usa o serializador/desserializador JSON nativo do Hive para ler os dados JSON armazenados no Amazon S3. Para obter mais informações sobre os formatos compatíveis, consulte SerDes e formatos de dados compatíveis.

Executar MSCK REPAIR TABLE

Depois de executar a consulta `CREATE TABLE`, execute o comando `MSCK REPAIR TABLE` no editor de consultas do Athena para carregar as partições, como no exemplo a seguir.

```
MSCK REPAIR TABLE impressions
```

Depois de executar esse comando, os dados estarão prontos para consulta.

Consultar os dados

Consulte os dados da tabela de impressões usando a coluna de partição. Veja um exemplo abaixo:

```
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
```

Esta consulta deve mostrar dados semelhantes aos seguintes:

```
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
```

Cenário 2: Os dados não são particionados no formato Hive

No exemplo a seguir, o comando `aws s3 ls` mostra os logs ELB armazenados no Amazon S3. Observe como o layout de dados não usa pares `key=value` e, portanto, não está no formato Hive. (A opção `--recursive` para o comando `aws s3 ls` especifica que todos os arquivos ou objetos no diretório ou prefixo especificado serão listados.)

```
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_$folder$
```

Executar ALTER TABLE ADD PARTITION

Como os dados não estão no formato Hive, você não pode usar o comando `MSCK REPAIR TABLE` para adicionar as partições à tabela depois de criá-la. Em vez disso, você pode usar o comando ALTER TABLE ADD PARTITION para adicionar cada partição manualmente. Por exemplo, para carregar os dados em s3://athena-examples-*myregion*/elb/plaintext/2015/01/01/, execute a consulta a seguir. Observe que não é necessária uma coluna de partição separada para cada pasta do Amazon S3 e que o valor da chave da partição pode ser diferente da chave do 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/'
```

Se já existir uma partição, você receberá o erro de que a partição já existe. Para evitar esse erro, você pode usar a cláusula `IF NOT EXISTS`. Para ter mais informações, consulte ALTER TABLE ADD PARTITION. Para remover uma partição, use ALTER TABLE DROP PARTITION.

Projeção de partições

Para evitar a necessidade de gerenciar partições, você pode usar a projeção de partição. A projeção de partição é uma opção para tabelas altamente particionadas cuja estrutura é conhecida antecipadamente. Na projeção de partições, os valores e os locais das partições são calculados a partir das propriedades da tabela que foi configurada, não da leitura de um repositório de metadados. Como os cálculos na memória são mais rápidos do que a pesquisa remota, o uso da projeção de partição pode reduzir significativamente os tempos de execução da consulta.

Para ter mais informações, consulte Projeção de partições com o Amazon Athena.

Recursos adicionais do

- Para obter informações sobre opções de particionamento para dados do Firehose, consulte Exemplo do Amazon Data Firehose.
- Também é possível automatizar partições usando o driver JDBC.

- É possível usar CTAS e INSERT INTO para particionar um conjunto de dados. Para ter mais informações, consulte Usar CTAS e INSERT INTO para ETL e análise de dados.

Projeção de partições com o Amazon Athena

É possível usar a projeção de partições no Athena para acelerar o processamento de consultas de tabelas altamente particionadas e automatizar o gerenciamento de partições.

Na projeção de partições, o Athena calcula valores e localizações de partições usando as propriedades da tabela que você configura diretamente em sua tabela no AWS Glue. As propriedades da tabela permitem que o Athena "projete", ou determine, as informações de partição necessárias em vez de fazer mais uma pesquisa de metadados demorada no AWS Glue Data Catalog. Como as operações na memória costumam ser mais rápidas do que as operações remotas, a projeção de partições pode reduzir o runtime de consultas em tabelas altamente particionadas. Dependendo das características específicas da consulta e dos dados subjacentes, a projeção de partições pode reduzir significativamente o runtime para consultas restritas na recuperação de metadados de partição.

Redução e projeção para tabelas extremamente particionadas

A redução de partição coleta metadados e os "reduz" para apenas as partições que se aplicam à sua consulta. Geralmente, isso acelera as consultas. O Athena usa a redução de partição em todas as tabelas com colunas de partição, inclusive aquelas configuradas para projeção de partições.

Normalmente, ao processar consultas, o Athena faz uma chamada `GetPartitions` para o AWS Glue Data Catalog antes de reduzir a partição. Se uma tabela tiver um grande número de partições, o uso de `GetPartitions` poderá prejudicar a performance. Para que isso não aconteça, você pode usar a projeção de partições. A projeção de partições permite que o Athena evite chamar `GetPartitions` porque a configuração de projeção de partições fornece ao Athena todas as informações necessárias para criar as partições propriamente ditas.

Usar a projeção de partições

Para usar a projeção de partições, especifique os intervalos de valores de partição e os tipos de projeção para cada coluna de partição nas propriedades da tabela no AWS Glue Data Catalog ou no metastore do Hive externo. Essas propriedades personalizadas na tabela permitem que o Athena saiba quais padrões de partição esperar ao executar uma consulta na tabela. Durante a execução da consulta, o Athena usa essas informações para projetar os valores de partição, em vez de recuperá-los do AWS Glue Data Catalog ou do metastore do Hive externo. Esse comportamento reduz o

tempo de execução da consulta e também automatiza o gerenciamento de partições, pois acaba com a necessidade de criar manualmente partições no Athena, no AWS Glue ou no metastore do Hive externo.

> **Important**
>
> Habilitar a projeção de partições em uma tabela faz com que o Athena ignore os metadados de partição registrados na tabela no AWS Glue Data Catalog ou no metastore do Hive.

Casos de uso

Os cenários em que a projeção de partições é útil incluem o seguinte:

- As consultas em uma tabela altamente particionada não são concluídas com a rapidez desejada.
- Você adiciona partições regularmente a tabelas à medida que novas partições de data ou hora são criadas em seus dados. Com a projeção de partições, você configura intervalos de datas relativos que podem ser usados à medida que novos dados chegam.
- Você tem dados altamente particionados no Amazon S3. Os dados não são práticos para modelar no AWS Glue Data Catalog ou no metastore do Hive, e suas consultas leem apenas pequenas partes deles.

Estruturas de partições projetáveis

A projeção de partições é mais facilmente configurada quando as partições seguem um padrão previsível como, mas não limitado ao seguinte:

- Inteiros: qualquer sequência contínua de inteiros, como `[1, 2, 3, 4, ..., 1000]` ou `[0500, 0550, 0600, ..., 2500]`.
- Datas: qualquer sequência contínua de datas ou datas e horas, como `[20200101, 20200102, ..., 20201231]` ou `[1-1-2020 00:00:00, 1-1-2020 01:00:00, ..., 12-31-2020 23:00:00]`.
- Valores enumerados: um conjunto finito de valores enumerados, como códigos de aeroporto ou Regiões da AWS.
- Logs de AWS service (Serviço da AWS): normalmente, os logs de AWS service (Serviço da AWS) têm uma estrutura conhecida com um esquema de partição que você pode especificar no AWS Glue e que, portanto, o Athena pode usar na projeção de partições.

Personalizar o modelo de caminho de partição

Por padrão, o Athena cria locais de partição usando o formulário `s3://DOC-EXAMPLE-BUCKET/<table-root>/partition-col-1=<partition-col-1-val>/partition-col-2=<partition-col-2-val>/`. No entanto, se os dados estão organizados de forma diferente, o Athena oferece um mecanismo para personalizar esse modelo de caminho. Para obter as etapas, consulte Especificar locais de armazenamento personalizados do S3.

Considerações e limitações

As seguintes considerações se aplicam:

- A projeção de partições elimina a necessidade de especificar partições manualmente no AWS Glue ou em um metastore do Hive externo.
- Quando você habilita a projeção de partições em uma tabela, o Athena ignora todos os metadados de partição no AWS Glue Data Catalog ou no metastore do Hive externo referentes a essa tabela.
- Se uma partição projetada não existir no Amazon S3, o Athena ainda a projetará. O Athena não gera um erro, mas também não retorna dados. No entanto, se muitas partições estiverem vazias, a performance poderá ser mais lenta em comparação com as partições tradicionais do AWS Glue. Se mais da metade das partições projetadas estiver vazia, é recomendado usar partições tradicionais.
- As consultas de valores que excedem os limites de intervalo definidos para a projeção de partição não retornam erro. Em vez disso, a consulta é executada, mas não retorna nenhuma linha. Por exemplo, se você tem dados relacionados a um cronograma que começa em 2020 e está definidos como `'projection.timestamp.range'='2020/01/01,NOW'`, uma consulta como `SELECT * FROM table-name WHERE timestamp = '2019/02/02'` será concluída com êxito, mas não retornará nenhuma linha.
- A projeção de partições somente pode ser usada quando a tabela é consultada pelo Athena. Se a mesma tabela for lida por meio de outro serviço, como o Amazon Redshift Spectrum, o Athena for Spark ou o Amazon EMR, os metadados de partição padrão serão usados.
- Como a projeção de partições é um recurso somente DML, `SHOW PARTITIONS` não lista as partições projetadas pelo Athena, mas que não estão registradas no catálogo do AWS Glue ou no metastore do Hive externo.
- O Athena não usa as propriedades de tabela de visualizações como configuração para a projeção de partições. Para contornar essa limitação, configure e habilite a projeção de partições nas propriedades das tabelas mencionadas nas visualizações.

- Não é possível usar filtros de dados do Lake Formation com projeção de partições no Athena.

Vídeo

O vídeo a seguir mostra como usar a projeção de partições para melhorar a performance das suas consultas no Athena.

Projeção de partições com o Amazon Athena

Tópicos

- Configurar a projeção de partições
- Tipos compatíveis para projeção de partições
- Particionamento de ID dinâmico
- Exemplo do Amazon Data Firehose

Configurar a projeção de partições

A configuração da projeção de partições nas propriedades de uma tabela é um processo de duas etapas:

1. Especifique os intervalos de dados e os padrões relevantes para cada coluna de partição ou use um modelo personalizado.
2. Habilite a projeção de partições para a tabela.

> **Note**
>
> Antes de adicionar propriedades de projeção de partição para uma tabela existente, a coluna de partição para a qual você está configurando as propriedades de projeção de partição já deve existir no esquema da tabela. Caso a coluna de partição ainda não exista, você deverá adicionar manualmente uma coluna de partição à tabela existente. O AWS Glue não executa esta etapa para você de forma automática.

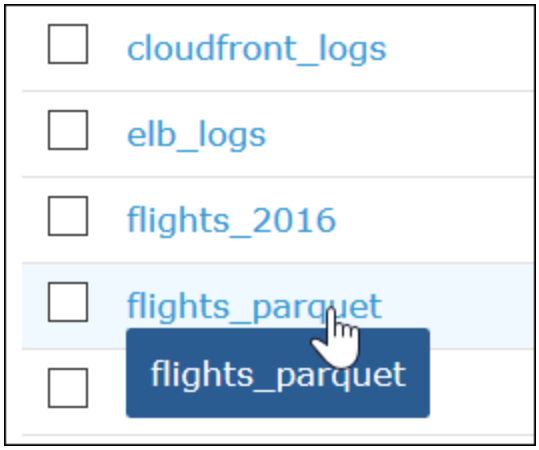
Esta seção mostra como definir as propriedades de tabela para o AWS Glue. Para defini-las, você pode usar o console do AWS Glue, as consultas CREATE TABLE do Athena ou as operações de

AWS Glue API. O procedimento a seguir mostra como definir as propriedades no console do AWS Glue.

Para configurar e habilitar a projeção de partições usando o console do AWS Glue

1. Faça login no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. Escolha a guia Tabelas.

   Na guia Tabelas, você pode editar tabelas existentes ou escolher Adicionar tabelas para criar novas tabelas. Para obter informações sobre como adicionar tabelas manualmente ou com um crawler, consulte Trabalhar com tabelas no console do AWS Glue no Guia do desenvolvedor do AWS Glue.
3. Na lista de tabelas, escolha o link para a tabela que você deseja editar.

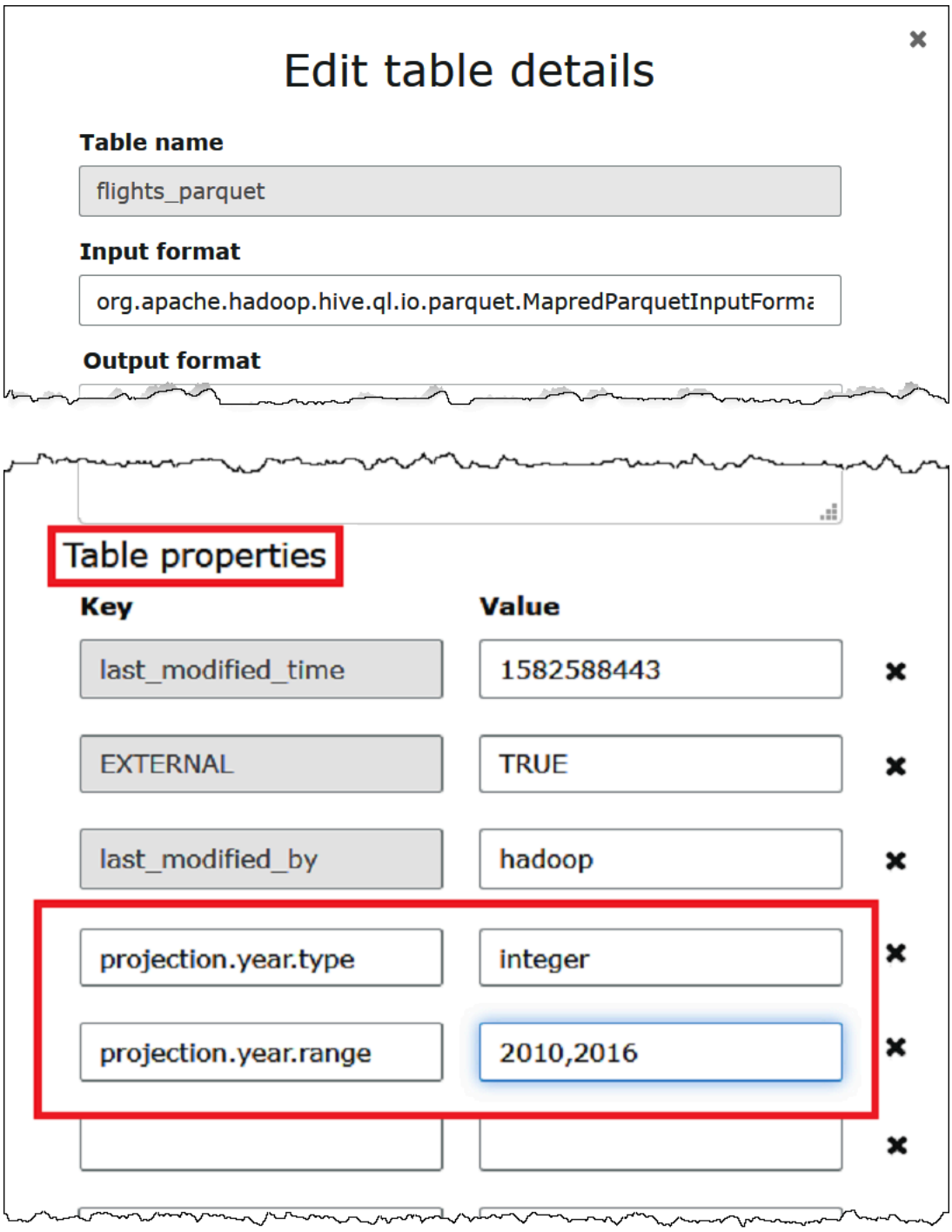


4. Selecione Actions (Ações), Edit (Editar).
5. Na página Edit table (Editar tabela), na seção Table properties (Propriedades da tabela), para cada coluna particionada, adicione o seguinte par de chave-valor:

   a. Em Chave, adicione `projection.columnName.type`.

   b. Em Valor, adicione um dos tipos compatíveis: `enum`, `integer`, `date` ou `injected`. Para ter mais informações, consulte Tipos compatíveis para projeção de partições.
6. Seguindo as orientações em Tipos compatíveis para projeção de partições, adicione outros pares de chave-valor de acordo com seus requisitos de configuração.

   O exemplo de configuração de tabela a seguir configura a coluna `year` para projeção de partições, restringindo os valores que podem ser retornados a um intervalo de 2010 a 2016.

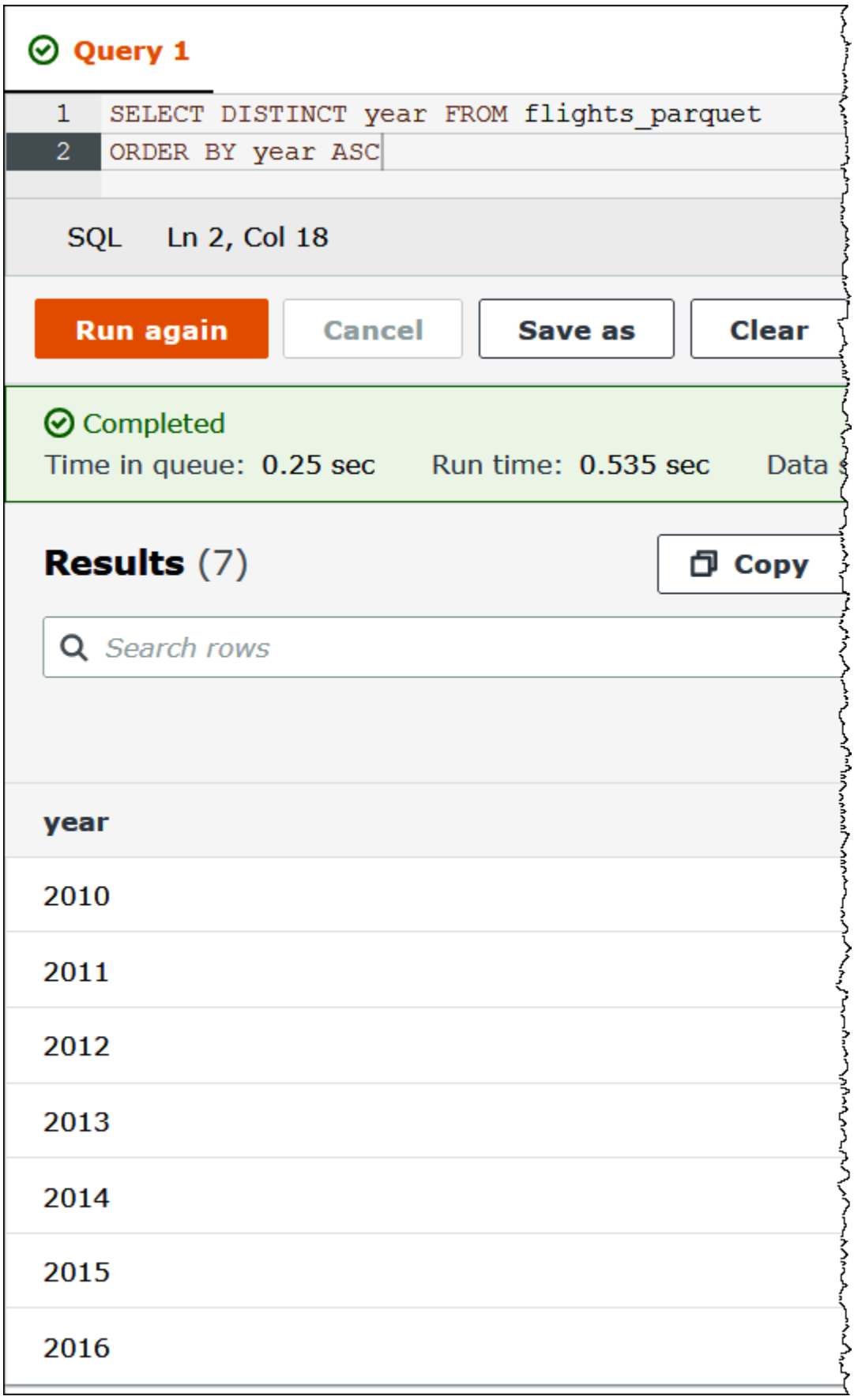


7. Adicione um par de chave-valor para habilitar a projeção de partições. Em Chave, insira `projection.enabled` e, em Valor, insira `true`.

> **Note**
>
> Você pode desativar a projeção de partições nessa tabela a qualquer momento definindo `projection.enabled` como `false`.

8. Quando terminar, escolha Salvar.
9. No editor de consultas do Athena, faça uma consulta de teste nas colunas de tabela que você configurou.

   A consulta de exemplo a seguir usa `SELECT DISTINCT` para retornar os valores exclusivos da coluna `year`. O banco de dados contém dados de 1987 a 2016, mas a propriedade `projection.year.range` restringe os valores retornados para os anos 2010 a 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**
>
> Se você definir `projection.enabled` como `true`, mas não conseguir configurar uma ou mais colunas de partição, receberá uma mensagem de erro como a seguinte: `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).`

Especificar locais de armazenamento personalizados do S3

Ao editar as propriedades de tabela no AWS Glue, você também pode especificar um modelo de caminho do Amazon S3 personalizado para as partições projetadas. Um modelo personalizado permite que o Athena mapeie corretamente os valores de partição para os locais de arquivo personalizados do Amazon S3 que não seguem o padrão `.../column=value/...`.

O uso de um modelo personalizado é opcional. No entanto, se você usar um modelo personalizado, o modelo deverá conter um espaço reservado para cada coluna de partição. Os locais baseados em modelo devem terminar com uma barra para que os arquivos de dados particionados sejam armazenados em uma "pasta" por partição.

Como especificar um modelo de local de partição personalizado

1. Seguindo as etapas para configurar e habilitar a projeção de partições usando o console do AWS Glue, adicione outro par de chave-valor que especifique um modelo personalizado da seguinte forma:

   a. Em Chave, digite `storage.location.template`.

   b. Em Valor, especifique um local que inclua um espaço reservado para cada coluna de partição. Termine cada espaço reservado (e o próprio caminho do S3) com uma barra única.

      Os valores de modelo de exemplo a seguir assumem uma tabela com colunas de partição `a`, `b` e `c`.

      ```
      s3://DOC-EXAMPLE-BUCKET/table_root/a=${a}/${b}/some_static_subdirectory/${c}/
      ```

```
s3://DOC-EXAMPLE-BUCKET/table_root/c=${c}/${b}/some_static_subdirectory/${a}/
${b}/${c}/${c}/
```

Para a mesma tabela, o valor de modelo de exemplo a seguir é inválido porque não contém nenhum espaço reservado para a coluna `c`.

```
s3://DOC-EXAMPLE-BUCKET/table_root/a=${a}/${b}/some_static_subdirectory/
```

2. Escolha Aplicar.

Tipos compatíveis para projeção de partições

Uma tabela pode ter qualquer combinação dos tipos `enum`, `integer`, `date`, ou `injected` de coluna de partição.

Tipo enum

Use o tipo `enum` para colunas de partição cujos valores são membros de um conjunto enumerado (por exemplo, códigos de aeroporto ou Regiões da AWS).

Defina as propriedades da partição na tabela da seguinte forma:

Nome da propriedade	Exemplos de valores	Descrição
`projection.`*`columnName`*`.type`	`enum`	Obrigatório. O tipo de projeção a ser usado para a coluna *`columnName`* . O valor deve ser `enum` (sem diferenciar maiúsculas e minúsculas) para sinalizar o uso do tipo de enumeração. Espaços em branco iniciais e finais são permitidos.
`projection.`*`columnName`*`.values`	`A,B,C,D,E,F,G,Unknown`	Obrigatório. Uma lista separada por vírgulas

Nome da propriedade	Exemplos de valores	Descrição
		de valores de partição enumerados para a coluna *columnName* . Qualquer espaço em branco é considera do parte de um valor de enumeração.

> **Note**
>
> Como prática recomendada, limite o uso de projeções de partições com base em `enum` a no máximo algumas dúzias. Embora não haja um limite específico para projeções `enum`, o tamanho total dos metadados da sua tabela não pode exceder o limite do AWS Glue de cerca de 1 MB quando compactado com gzip. Esse limite é compartilhado entre as partes principais da tabela, como nomes de colunas, local, formato de armazenamento etc. Se você usar mais de algumas dúzias de IDs exclusivos em sua projeção `enum`, considere uma abordagem alternativa, como criar buckets de um número menor de valores exclusivos em um campo substituto. Ao compensar a cardinalidade, você pode controlar o número de valores exclusivos no campo `enum`.

Tipo integer

Use o tipo integer para colunas de partição cujos valores possíveis são interpretáveis como inteiros dentro de um intervalo definido. As colunas integer projetadas estão atualmente limitadas ao intervalo de um Java assinado longo (-2^{63} a $2^{63}-1$, ambos incluídos).

Nome da propriedade	Exemplos de valores	Descrição
`projection.` *columnNam e* `.type`	`integer`	Obrigatório. O tipo de projeção a ser usado para a coluna *columnName* . O valor deve ser `integer` (sem diferenci ar maiúsculas e minúsculas) para sinalizar o uso do tipo de

Nome da propriedade	Exemplos de valores	Descrição
		inteiro. Espaços em branco iniciais e finais são permitidos.

`projection.` *`columnName`* `.range`	`0,10`
	
	`-1,8675309`
	
	`0001,9999`
		Obrigatório. Uma lista separada por vírgulas de dois elementos que fornece os valores de intervalo mínimo e máximo a serem retornados por consultas na coluna *`columnName`* . Observe que os valores devem ser separados por uma vírgula, não por um hífen. Esses valores são inclusivos, podem ser negativos e podem ter zeros à esquerda. Espaços em branco iniciais e finais são permitidos.

`projection.` *`columnName`* `.interval`	`1`
	
	`5`
		Opcional. Um inteiro positivo que especifica o intervalo entre valores de partição sucessivos para a coluna *`columnName`* . Por exemplo, um valor `range` de “1,3" com um valor `interval` de “1" produz os valores 1, 2 e 3. O mesmo valor `range` com um valor `interval` de “2" produz os valores 1 e 3, ignorando 2. Espaços em branco iniciais e finais são permitidos. O padrão é um.

Nome da propriedade	Exemplos de valores	Descrição

`projection.` *`columnName`* `.digits`	`1`
	
	`5`
		Opcional. Um inteiro positivo que especifica o número de dígitos a serem incluídos na representação final do valor da partição para a coluna *`columnName`* . Por exemplo, um valor `range` de “1,3" que tem um valor `digits` de “1" produz os valores 1, 2 e 3. O mesmo valor `range` com um valor `digits` de “2" produz os valores 01, 02 e 03. Espaços em branco iniciais e finais são permitidos. O padrão é nenhum número estático de dígitos e nenhum zero à esquerda.

Tipo date

Use o tipo date para colunas de partição cujos valores são interpretáveis como datas (com horas opcionais) dentro de um intervalo definido.

> **Important**
>
> As colunas date projetadas são geradas no Horário Universal Coordenado (UTC) no tempo de execução da consulta.

Nome da propriedade	Exemplos de valores	Descrição
`projection.` *`columnName`* `.type`	`date`	Obrigatório. O tipo de projeção a ser usado para a coluna *`columnName`* . O valor deve ser

Nome da propriedade	Exemplos de valores	Descrição
		`date` (sem diferenciar maiúsculas e minúsculas) para sinalizar o uso do tipo de data. Espaços em branco iniciais e finais são permitidos.

`projection.` *`columnName`* `.range`	`201701,201812`
	
	`01-01-2010,12-31-2018`
	
	`NOW-3YEARS,NOW`
	
	`201801,NOW+1MONTH`
		Obrigatório. Uma lista separada por vírgulas de dois elementos que fornece os valores `range` mínimo e máximo para a coluna *`columnName`* . Esses valores são inclusivos e podem usar qualquer formato compatível com os tipos de data `java.time.*` em Java. Os valores mínimo e máximo devem usar o mesmo formato. O formato especificado na propriedade `.format` deve ser o formato usado para esses valores.
		
		Essa coluna também pode conter strings de data relativas, formatadas neste padrão de expressão regular:
		
		`\s*NOW\s*(([\+\-])\s*([0-9]+)\s*(YEARS?|MONTHS?|WEEKS?|DAYS?|HOURS?|MINUTES?|SECONDS?)\s*)?`
		
		Espaços em branco são permitidos. No entanto, em literais de data, eles são considerados parte das strings de data em si.

`projection.` *`columnName`* `.format`	`yyyyMM`
	
	`dd-MM-yyyy`
	
	`dd-MM-yyyy-HH-mm-ss`
		Obrigatório. Uma string de formato de data baseada no formato de data Java DateTimeFormatter. Pode ser qualquer tipo `Java.time.*` compatível.

Nome da propriedade	Exemplos de valores	Descrição

`projection.` *`columnNam e`* `.interval`	`1`
	
	`5`
		Um inteiro positivo que especifica o intervalo entre valores de partição sucessivos para a coluna *`columnName`* . Por exemplo, um valor `range` de `2017-01,2018-12` com um valor `interval` de `1` e um valor `interval. unit` de `MONTHS` produz os valores 2017-01, 2017-02, 2017-03 e assim por diante. O mesmo valor `range` com um valor `interval` de `2` e um valor `interval.unit` de `MONTHS` produz os valores 2017-01, 2017-03, 2017-05 e assim por diante. Espaços em branco iniciais e finais são permitidos.
		
		Quando as datas fornecidas são com precisão de um dia ou um mês, o `interval` é opcional e o padrão é 1 dia ou 1 mês, respectivamente. Caso contrário, o `interval` será obrigatório.

`projection.` *`columnNam e`* `.interval.unit`	`YEARS`
	
	`MONTHS`
	
	`WEEKS`
	
	`DAYS`
	
	`HOURS`
	
	`MINUTES`
	
	`SECONDS`
	
	`MILLIS`
		Uma palavra de unidade de tempo que representa a forma serializada de uma ChronoUnit. Os valores possíveis são `YEARS`, `MONTHS`, `WEEKS`, `DAYS`, `HOURS`, `MINUTES`, `SECONDS` ou `MILLIS`. Esses valores diferenci am maiúsculas de minúsculas.
		
		Quando as datas fornecidas são com precisão de um dia ou um mês, o `interval.unit` é opcional e o padrão é 1 dia ou 1 mês, respectiv amente. Caso contrário, `interval.unit` será obrigatório.

Tipo injected

Use o tipo injetado para colunas de partição com valores possíveis que não podem ser gerados processualmente dentro de algum intervalo lógico, mas que são fornecidos na cláusula `WHERE` de uma consulta como um único valor.

É importante ter em mente os seguintes pontos:

- Consultas em colunas injetadas falharão se uma expressão de filtro não for fornecida para cada coluna injetada.
- Consultas com vários valores para uma expressão de filtro em uma coluna injetada só terão êxito se os valores forem disjuntos.
- Somente colunas do tipo `string` são permitidas.

Nome da propriedade	Value	Descrição
`projection.` *`columnNam e`* `.type`	`inje`	Obrigatório. O tipo de projeção a ser usado para a coluna *`columnName`* . Somente o tipo `string` é permitido. O valor especificado deve ser `injected` (sem diferenci ação de maiúsculas e minúsculas). Espaços em branco iniciais e finais são permitidos.

Para ter mais informações, consulte Uso do tipo de projeção `injected`.

Particionamento de ID dinâmico

Quando os dados são particionados por uma propriedade com alta cardinalidade ou quando os valores não podem ser conhecidos antecipadamente, é possível usar o tipo de projeção `injected`. Exemplos dessas propriedades são os nomes de usuário e os IDs de dispositivos ou de produtos. Quando você usa o tipo de projeção `injected` para configurar uma chave de partição, o Athena usa valores da própria consulta para calcular o conjunto de partições que serão lidas.

Para que o Athena possa executar uma consulta em uma tabela que tem uma chave de partição configurada com o tipo de projeção `injected`, o seguinte deve ser verdadeiro:

- A consulta deve incluir, no mínimo, um valor para a chave de partição.

- Os valores devem ser literais ou expressões que possam ser avaliadas sem a necessidade de leitura de dados.

Se algum desses critérios não for atendido, a consulta falhará e apresentará o seguinte erro:

CONSTRAINT_VIOLATION: para a coluna de partição projetada injetada *column_name*, a cláusula WHERE deve conter somente condições de igualdade estática e pelo menos uma dessas condições deve estar presente.

Uso do tipo de projeção **injected**

Imagine que você tem um conjunto de dados que consiste em eventos de dispositivos de IoT, os quais são particionados nos IDs dos dispositivos. Esse conjunto de dados tem as seguintes características:

- Os IDs dos dispositivos são gerados de forma aleatória.
- Novos dispositivos são provisionados com frequência.
- Atualmente, existem centenas de milhares de dispositivos e, no futuro, haverá milhões.

Esse conjunto de dados é complexo de gerenciar usando metastores tradicionais. É difícil manter as partições sincronizadas entre o armazenamento de dados e o metastore, e a filtragem de partições pode ser lenta durante o planejamento da consulta. Entretanto, se você configurar uma tabela para usar a projeção de partição e usar o tipo de projeção `injected`, obterá duas vantagens: não precisará gerenciar partições no metastore e as consultas não precisarão procurar metadados de partições.

O exemplo `CREATE TABLE`, apresentado a seguir, cria uma tabela para o conjunto de dados de eventos do dispositivo que acabamos de descrever. A tabela usa o tipo de projeção “injected”.

```
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}"
)
```

O exemplo de consulta apresentado a seguir pesquisa o número de eventos recebidos de três dispositivos específicos ao longo de 12 horas.

```
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
```

Quando você executa essa consulta, o Athena visualiza os três valores para a chave de partição `device_id` e os usa para calcular os locais das partições. O Athena usa os valores com a propriedade `storage.location.template` para gerar os seguintes locais:

- `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`

Se você omitir a propriedade `storage.location.template` da configuração da projeção de partição, o Athena usará o particionamento no estilo Hive para projetar os locais de partições com base no valor em `LOCATION` (por exemplo, `s3://DOC-EXAMPLE-BUCKET/prefix/device_id=4a770164-0392-4a41-8565-40ed8cec737e`).

Exemplo do Amazon Data Firehose

Quando você usa o Firehose para entregar dados ao Amazon S3, a configuração padrão grava objetos com chaves semelhantes ao seguinte exemplo:

```
s3://DOC-EXAMPLE-BUCKET/prefix/yyyy/MM/dd/HH/file.extension
```

Para criar uma tabela do Athena que encontre as partições automaticamente no momento da consulta, em vez de precisar adicioná-las ao AWS Glue Data Catalog à medida que novos dados chegam, você pode usar a projeção de partições.

O exemplo de `CREATE TABLE` a seguir usa a configuração padrão do Firehose.

```
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}/"
)
```

A cláusula `TBLPROPERTIES` na instrução `CREATE TABLE` diz ao Athena o seguinte:

- Usar projeção de partições ao consultar a tabela
- A chave de partição `datehour` é do tipo `date` (que inclui um horário opcional)
- Como as datas são formatadas
- O intervalo dos períodos de datas. Observe que os valores devem ser separados por uma vírgula, não por um hífen.
- Onde encontrar os dados no Amazon S3.

Quando você consulta a tabela, o Athena calcula os valores para `datehour` e usa o modelo de local de armazenamento para gerar uma lista de locais de partição.

Usar o tipo **`date`**

Quando você usa o tipo `date` para a chave de partição projetada, deve especificar um intervalo. Como não há dados para datas antes da criação do fluxo de entrega do Firehose, você pode usar a data de criação como o início. E como você não tem dados para datas no futuro, pode usar o token especial `NOW` como fim.

No exemplo `CREATE TABLE`, a data de início é especificada como 1º de janeiro de 2021 à meia-noite UTC.

> **Note**
>
> Configure um intervalo que corresponda o mais exatamente possível aos seus dados para que o Athena procure apenas partições existentes.

Quando uma consulta é executada na tabela de amostra, o Athena usa as condições na chave de partição `datehour` em combinação com o intervalo para gerar valores. Considere a seguinte consulta:

```
SELECT *
FROM my_ingested_data
WHERE datehour >= '2020/12/15/00'
AND datehour < '2021/02/03/15'
```

A primeira condição na consulta `SELECT` usa uma data antes do início do intervalo de datas especificado pela instrução `CREATE TABLE`. Como a configuração de projeção de partições não especifica nenhuma partição para datas antes de 1º de janeiro de 2021, o Athena procura dados somente nos locais a seguir e ignora as datas anteriores na consulta.

```
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/
```

Da mesma forma, se a consulta fosse executada em uma data e hora antes de 3 de fevereiro de 2021 às 15:00, a última partição refletiria a data e a hora atuais, não a data e a hora na condição da consulta.

Se você quiser consultar os dados mais recentes, pode aproveitar o fato de que Athena não gera datas futuras e especificar apenas uma `datehour` de início, como no exemplo a seguir.

```
SELECT *
FROM my_ingested_data
WHERE datehour >= '2021/11/09/00'
```

Escolher chaves de partição

Você pode especificar como a projeção de partições mapeia os locais de partição para chaves de partição. No exemplo da seção anterior para `CREATE TABLE`, a data e o horário foram combinados em uma chave de partição chamada “datehour”, mas outros esquemas são possíveis. Por exemplo, você também pode configurar uma tabela com chaves de partição separadas por ano, mês, dia e hora.

No entanto, dividir datas em ano, mês e dia significa que o tipo de projeção de partição `date` não pode ser usado. Uma alternativa é separar a data do horário para ainda aproveitar o tipo de projeção da partição `date`, mas facilitar a leitura das consultas que especificam intervalos de horas.

Com isso em mente, o exemplo `CREATE TABLE` a seguir separa a data da hora. Como `date` é uma palavra reservada em SQL, o exemplo usa `day` como nome da chave de partição que representa a data.

```
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}/"
)
```

No exemplo de instrução `CREATE TABLE`, a hora é uma chave de partição separada, configurada como um inteiro. A configuração para a chave de partição de hora especifica o intervalo de 0 a 23 e que a hora deverá ser formatada com dois dígitos quando o Athena gerar os locais de partições.

Uma consulta para a tabela `my_ingested_data2` poderia ser semelhante a esta:

```
SELECT *
FROM my_ingested_data2
WHERE day = '2021/11/09'
AND hour > 3
```

Tipos de chave de partição e tipos de projeção de partições

Observe que a chave `datehour` no primeiro exemplo de `CREATE TABLE` é configurada como `date` na configuração de projeção de partições, mas o tipo de chave de partição é `string`. O mesmo vale para `day` no segundo exemplo. Os tipos na configuração de projeção de partições apenas dizem ao Athena como formatar os valores quando ele gera os locais de partição. Os tipos especificados não alteram o tipo da chave de partição. Em consultas, `datehour` e `day` são do tipo `string`.

Quando uma consulta inclui uma condição como `day = '2021/11/09'`, o Athena analisa a sequência do lado direito da expressão usando o formato de data especificado na configuração de projeção de partições. Depois que o Athena verifica se a data está dentro do intervalo configurado, ele usa o formato de data novamente para inserir a data como uma sequência no modelo de local de armazenamento.

Da mesma forma, para uma condição de consulta como `day > '2021/11/09'`, o Athena analisa o lado direito e gera uma lista de todas as datas correspondentes dentro do intervalo configurado. Em seguida, ele usa o formato de data para inserir cada data no modelo de local de armazenamento para criar a lista de locais de partição.

Escrever a mesma condição que `day > '2021-11-09'` ou `day > DATE '2021-11-09'` não funciona. No primeiro caso, o formato de data não corresponde (observe os hifens em vez de barras) e, no segundo caso, os tipos de dados não correspondem.

Usar prefixos personalizados e particionamento dinâmico

É possível configurar o Firehose com prefixos personalizados e particionamento dinâmico. Usando esses recursos, você pode configurar as chaves do Amazon S3 e configurar esquemas de particionamento que suportem melhor seu caso de uso. Você também pode usar a projeção de partições com esses esquemas de particionamento e configurá-los de acordo.

Por exemplo, você pode usar o recurso de prefixo personalizado para obter chaves do Amazon S3 com datas em formato ISO em vez do esquema padrão de `yyyy/MM/dd/HH`.

Você também pode combinar prefixos personalizados com particionamento dinâmico para extrair uma propriedade como `customer_id` das mensagens do Firehose, como no exemplo a seguir.

```
prefix/!{timestamp:yyyy}-!{timestamp:MM}-!{timestamp:dd}/!
{partitionKeyFromQuery:customer_id}/
```

Com esse prefixo do Amazon S3, o fluxo de entrega do Firehose gravaria objetos em chaves como `s3://DOC-EXAMPLE-BUCKET/prefix/2021-11-01/customer-1234/file.extension`. Para uma propriedade como `customer_id`, na qual os valores podem não ser conhecidos com antecedência, você pode usar o tipo de projeção de partições `injected` e usar uma instrução `CREATE TABLE` como a seguinte:

```
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}/"
)
```

Quando você consulta uma tabela que tem uma chave de partição do tipo `injected`, sua consulta deve incluir um valor para essa chave de partição. Uma consulta para a tabela `my_ingested_data3` poderia ser semelhante a esta:

```
SELECT *
FROM my_ingested_data3
WHERE day BETWEEN '2021-11-01' AND '2021-11-30'
AND customer_id = 'customer-1234'
```

Datas em formato ISO

Como os valores para chave de partição `day` são em formato ISO, você também pode usar o tipo `DATE`, em vez de `STRING`, para a chave de partição de dia, como no exemplo a seguir:

```
PARTITIONED BY (day DATE, customer_id STRING)
```

Quando você consulta, essa estratégia permite que você use funções de data na chave de partição sem análise ou conversão, como no exemplo a seguir:

```
SELECT *
FROM my_ingested_data3
WHERE day > CURRENT_DATE - INTERVAL '7' DAY
AND customer_id = 'customer-1234'
```

> **Note**
>
> A especificação de uma chave de partição do tipo `DATE` pressupõe que você usou o atributo custom prefix para criar chaves do Amazon S3 que tenham datas no formato ISO. Caso esteja usando o formato padrão de `yyyy/MM/dd/HH` do Firehose, você deverá especificar a chave de partição com o tipo `string`, mesmo que a propriedade da tabela correspondente seja do tipo `date`, como no seguinte exemplo:
>
> ```
> PARTITIONED BY (
>   `mydate` string)
> ```

```
TBLPROPERTIES (
  'projection.enabled'='true',
   ...
  'projection.mydate.type'='date',
  'storage.location.template'='s3://DOC-EXAMPLE-BUCKET/prefix/${mydate}')
```

Criar uma tabela a partir de resultados de consultas (CTAS)

Uma consulta `CREATE TABLE AS SELECT` (CTAS) cria uma tabela no Athena com base nos resultados de uma instrução `SELECT` de outra consulta. O Athena armazena arquivos de dados criados pela instrução CTAS em um local especificado no Amazon S3. Para ver a sintaxe, consulte CREATE TABLE AS.

`CREATE TABLE AS` combina uma instrução DDL `CREATE TABLE` com uma instrução DML `SELECT`, por isso tecnicamente contém tanto DDL quanto DML. No entanto, observe que, para fins de cotas de serviço, as consultas CTAS no Athena são tratadas como DML. Para obter informações sobre as cotas de serviço do Athena, consulte Service Quotas.

Use consultas CTAS para:

- Criar tabelas a partir dos resultados da consulta em uma etapa, sem consultar conjuntos de dados brutos repetidamente. Isso facilita o trabalho com conjuntos de dados brutos.
- Transformar os resultados da consulta e migrar tabelas para outros formatos de tabela, como o Apache Iceberg. Isso melhora a performance da consulta e reduz seus custos no Athena. Para ter mais informações, consulte Criar tabelas Iceberg.
- Transformar os resultados da consulta em formatos de armazenamento, como Parquet e ORC. Isso melhora a performance da consulta e reduz seus custos no Athena. Para ter mais informações, consulte Formatos de armazenamento colunar.
- Crie cópias de tabelas existentes que contêm somente os dados necessários.

Tópicos

- Considerações e limitações de consultas CTAS
- Executar consultas CTAS no console
- Particionamento e bucketing no Athena
- Exemplos de consultas CTAS

- Usar CTAS e INSERT INTO para ETL e análise de dados
- Usar CTAS e INSERT INTO para resolver o limite de 100 partições

Considerações e limitações de consultas CTAS

As seções a seguir descrevem considerações e limitações que você deve ter em mente ao usar consultas (CTAS) `CREATE TABLE AS SELECT` no Athena.

Sintaxe da consulta CTAS

A sintaxe de consulta CTAS é diferente da sintaxe de `CREATE [EXTERNAL] TABLE` usada para criar tabelas. Consulte CREATE TABLE AS.

Visualizações vs. consultas CTAS

As consultas CTAS gravam novos dados em um local especificado no Amazon S3, já as visualizações não gravam dados.

Local dos resultados da consulta CTAS

Se o seu grupo de trabalho substituir a configuração do lado do cliente para o local dos resultados das consultas, o Athena criará a tabela no local `s3://DOC-EXAMPLE-BUCKET/tables/<query-id>/`. Para ver o local dos resultados de consultas especificado para o grupo de trabalho, visualize os detalhes do grupo de trabalho.

Se seu grupo de trabalho não substituir o local dos resultados de consultas, você poderá usar a sintaxe `WITH (external_location ='s3://DOC-EXAMPLE-BUCKET/')` em sua consulta CTAS para especificar onde os resultados da consulta CTAS são armazenados.

> **Note**
>
> A propriedade `external_location` deve especificar um local vazio. Uma consulta CTAS verifica se o local do caminho (prefixo) no bucket está vazio e nunca substitui os dados se o local já tiver dados. Para usar o mesmo local novamente, exclua os dados no local do prefixo de chave no bucket.

Se você omitir a sintaxe de `external_location` e não usar a configuração do grupo de trabalho, o Athena usará sua configuração do lado do cliente para o local dos resultados das consultas e criará

a tabela no local s3://DOC-EXAMPLE-BUCKET/*<Unsaved-or-query-name>*/*<year>*/*<month>*/*<date>*/tables/*<query-id>*/.

Localizar arquivos órfãos

Se uma instrução CTAS ou INSERT INTO falhar, é possível que os dados órfãos permaneçam no local dos dados. Como o Athena, em alguns casos, não realiza a exclusão dos dados ou dos dados parciais do seu bucket, você poderá ler esses dados parciais em consultas subsequentes. Para localizar arquivos órfãos para inspeção ou exclusão, é possível usar o arquivo do manifesto de dados que o Athena oferece para rastrear a lista de arquivos a serem gravados. Para obter mais informações, consulte Identificar os arquivos de saída das consultas e DataManifestLocation.

Cláusulas ORDER BY ignoradas

Em uma consulta CTAS, o Athena ignora as cláusulas ORDER BY na parte SELECT da consulta.

Segundo a especificação SQL (ISO 9075 Parte 2), a ordenação das linhas de uma tabela especificada por uma expressão de consulta é garantida somente para a expressão de consulta que contém imediatamente a cláusula ORDER BY. De qualquer forma, as tabelas no SQL são inerentemente desordenadas, e a implementação das cláusulas na subconsulta ORDER BY faria com que a consulta tivesse uma performance ruim e não resultaria em uma saída ordenada. Assim, nas consultas CTAS do Athena, não há garantia de que a ordem especificada pela cláusula ORDER BY será preservada quando os dados forem gravados.

Formatos para armazenar resultados da consulta

Os resultados de consultas CTAS serão armazenados em Parquet por padrão, se não for especificado um formato de armazenamento de dados. É possível armazenar resultados de CTAS em PARQUET, ORC, AVRO, JSON e TEXTFILE. Não há suporte para delimitadores de vários caracteres no formato CTAS TEXTFILE. As consultas CTAS não exigem a especificação de um SerDe para interpretar transformações de formato. Consulte Example: Writing query results to a different format.

Formatos de compactação

A compactação GZIP é usada para os resultados da consulta CTAS nos formatos JSON e TEXTFILE. Para Parquet, você pode usar GZIP ou SNAPPY, e o padrão é GZIP. Para ORC, você pode usar LZ4, SNAPPY, ZLIB ou ZSTD, e o padrão é ZLIB. Para obter exemplos de CTAS que especificam compactação, consulte Example: Specifying data storage and compression formats. Para obter informações sobre compressão no Athena, consulte Suporte a compactação no Athena.

Limites para partições e buckets

Você pode particionar e armazenar em buckets os dados resultantes de uma consulta CTAS. Para ter mais informações, consulte Particionamento e bucketing no Athena. Ao criar uma tabela particionada usando o CTAS, o Athena tem um limite de gravação de 100 partições.

Inclua predicados de particionamento e bucketing no final da cláusula `WITH` que especifica propriedades da tabela de destino. Para obter mais informações, consulte Example: Creating bucketed and partitioned tables e Particionamento e bucketing no Athena.

Para obter informações sobre como contornar a limitação de 100 partições, consulte Usar CTAS e INSERT INTO para resolver o limite de 100 partições.

Criptografia

É possível criptografar os resultados da consulta CTAS no Amazon S3, semelhante à forma como você criptografa outros resultados de consulta no Athena. Para ter mais informações, consulte Criptografar resultados das consultas do Athena armazenados no Amazon S3.

Expected bucket owner (Proprietário esperado do bucket)

Para instruções CTAS, a configuração do proprietário do bucket esperado não se aplica ao local da tabela de destino no Amazon S3. A configuração esperada do proprietário do bucket se aplica somente ao local de saída do Amazon S3 que você especificar para os resultados da consulta do Athena. Para ter mais informações, consulte Especificar um local para resultados de consultas usando o console do Athena.

Tipos de dados

Os tipos de dados da coluna para uma consulta CTAS são os mesmos que os tipos especificados para a consulta original.

Executar consultas CTAS no console

No console do Athena, você pode criar uma consulta CTAS com base em outra consulta.

Para criar uma consulta CTAS a partir de outra consulta

1. Execute a consulta no editor de consultas do console do Athena.

2. Na parte inferior do editor de consultas, escolha a opção Create (Criar) e, em seguida, escolha Table from query (Tabela a partir de consulta).
3. No formulário Create table as select (Criar tabela conforme seleção), preencha os seguintes campos:

   a. Em Table name (Nome da tabela), especifique o nome para sua nova tabela. Use apenas letras minúsculas e sublinhados, como `my_select_query_parquet`.

   b. Em Database configuration (Configuração de banco de dados), use as opções para escolher um banco de dados existente ou criar um banco de dados.

   c. (Opcional) Em Result configuration (Configuração de resultado), para Location of CTAS query results (Localização dos resultados da consulta CTAS), se a configuração de localização dos resultados da consulta de grupo de trabalho não substituir essa opção, faça o seguinte:

      - Insira o caminho para uma localização existente do S3 na caixa de pesquisa ou escolha Browse S3 (Procurar no S3) para escolher uma localização em uma lista.
      - Escolha View (Visualizar) para abrir a página Buckets do console do Amazon S3, na qual você poderá ver mais informações sobre seus buckets existentes e escolher ou criar um bucket com suas próprias configurações.

      Você deverá especificar um local vazio no Amazon S3 no qual os dados serão produzidos. Se os dados já existirem no local especificado, a consulta apresentará erro.

      Se a configuração de localização dos resultados de consulta do seu grupo de trabalho substituir a configuração de localização, o Athena criará a tabela no local `s3://DOC-EXAMPLE-BUCKET/tables/query_id/`.

   d. Em Data format (Formato de dados), especifique o formato dos seus dados.

      - Table type (Tipo de tabela): o tipo de tabela padrão no Athena é o Apache Hive.
      - File format (Formato de arquivo): escolha entre opções como CSV, TSV, JSON, Parquet ou ORC. Para obter mais informações sobre os formatos Parquet e ORC, consulte Formatos de armazenamento colunar.
      - Write compression (Compactação de gravação): (opcional) escolha um formato de compactação. O Athena suporta uma variedade de formatos de compactação para leitura e gravação de dados, incluindo a leitura de uma tabela que usa vários formatos de compactação. Por exemplo, o Athena pode ler com sucesso os dados de uma tabela que

usa o formato de arquivo Parquet quando alguns arquivos Parquet são compactados com o Snappy e outros arquivos Parquet são compactados com o GZIP. O mesmo princípio se aplica aos formatos de armazenamento ORC, arquivo de texto e JSON. Para ter mais informações, consulte Suporte a compactação no Athena.

- Partitions (Partições): (opcional) selecione as colunas que você deseja particionar. O particionamento dos dados restringe a quantidade que cada consulta verifica, o que melhora a performance e reduz o custo. Você pode dividir seus dados em partições usando qualquer chave. Para ter mais informações, consulte Particionar dados no Athena.
- Buckets (Compartimentos): (opcional) selecione as colunas que você deseja agrupar. O agrupamento é uma técnica que agrupa dados com base em colunas específicas em uma só partição. Essas colunas são conhecidas como bucket keys (chaves de bucket). Ao agrupar dados relacionados em um só bucket (um arquivo dentro de uma partição), você reduz significativamente a quantidade de dados digitalizados pelo Athena, melhorando assim a performance da consulta e reduzindo os custos. Para ter mais informações, consulte Particionamento e bucketing no Athena.

e. Em Preview table query (Visualizar consulta da tabela), analise sua consulta. Para ver a sintaxe de consulta, acesse CREATE TABLE AS.

f. Escolha Create table.

Para criar uma consulta CTAS usando um modelo SQL

Use o modelo `CREATE TABLE AS SELECT` para criar uma consulta CTAS no editor de consultas.

1. No console do Athena, ao lado de Tables and views (Tabelas e visualizações), escolha Create table (Criar tabela) e, em seguida, escolha CREATE TABLE AS SELECT. Isso preenche o editor de consultas com uma consulta CTAS com valores de espaço reservado.
2. No editor de consultas, edite a consulta conforme necessário. Para ver a sintaxe de consulta, acesse CREATE TABLE AS.
3. Escolha Executar.

Para ver exemplos, consulte Exemplos de consultas CTAS.

Particionamento e bucketing no Athena

O particionamento e o bucketing são duas formas de reduzir a quantidade de dados que o Athena deve verificar quando você executa uma consulta. O particionamento e o bucketing são complementares e podem ser usados em conjunto. Reduzir a quantidade de dados verificados gera uma performance melhor a um custo menor. Para obter diretrizes gerais sobre a performance das consultas do Athena, consulte As dez melhores dicas de ajuste de performance para o Amazon Athena.

O que é particionamento?

Particionamento significa organizar dados em diretórios (ou “prefixos”) no Amazon S3 com base em uma propriedade específica dos dados. Essas propriedades são chamadas chaves de partição. Uma chave de partição comum é a data ou alguma outra unidade de tempo, como ano ou mês. Porém, um conjunto de dados pode ser particionado por mais de uma chave. Por exemplo, dados sobre vendas de produtos podem ser divididos por data, categoria do produto e mercado.

Decidir como particionar

Bons candidatos para chaves de partição são as propriedades que sempre ou quase sempre são usadas em consultas e têm baixa cardinalidade. Há um equilíbrio entre ter partições em excesso e não ter partições suficientes. Com partições em excesso, o aumento do número de arquivos cria sobrecarga. Também há sobrecarga na filtragem das partições em si. Com poucas partições, as consultas geralmente precisam verificar mais dados.

Criar uma tabela particionada

Quando um conjunto de dados é particionado, é possível criar uma tabela particionada no Athena. Uma tabela particionada é uma tabela que contém chaves de partição. Ao usar `CREATE TABLE`, você adiciona partições à tabela. Quando você usa `CREATE TABLE AS`, as partições criadas no Amazon S3 são adicionadas automaticamente à tabela.

Em uma instrução `CREATE TABLE`, você especifica as chaves de partição na cláusula `PARTITIONED BY (`*`column_name data_type`*`)`. Em uma instrução `CREATE TABLE AS`, você especifica as chaves de partição em uma cláusula `WITH (partitioned_by = ARRAY['`*`partition_key`*`'])` ou `WITH (partitioning = ARRAY['`*`partition_key`*`'])` para tabelas Iceberg. Por questões de performance, as chaves de partição devem ser sempre do tipo `STRING`. Para ter mais informações, consulte Usar string como o tipo de dados para chaves de partição.

Para obter mais detalhes de sintaxe de `CREATE TABLE` e `CREATE TABLE AS`, consulte CREATE TABLE e Propriedades da tabela CTAS.

Consultar tabelas particionadas

Quando você consulta uma tabela particionada, o Athena usa os predicados da consulta para filtrar a lista de partições. Em seguida, ele utiliza os locais das partições correspondentes para processar os arquivos encontrados. O Athena pode reduzir com eficiência a quantidade de dados verificados simplesmente não lendo dados nas partições que não correspondam aos predicados da consulta.

Exemplos

Suponha que você tenha uma tabela particionada por `sales_date` e `product_category` e queira saber a receita total de uma semana em uma categoria específica. Inclua predicados nas colunas`sales_date` e `product_category` para garantir que o Athena verificará somente a quantidade mínima de dados, como no exemplo a seguir.

```
SELECT SUM(amount) AS total_revenue
FROM sales
WHERE sales_date BETWEEN '2023-02-27' AND '2023-03-05'
AND product_category = 'Toys'
```

Suponha que você tenha um conjunto de dados que seja particionado por data, mas também tenha um carimbo de data e hora refinado.

Com as tabelas Iceberg, é possível declarar que uma chave de partição tem uma relação com uma coluna, mas com as tabelas Hive, o mecanismo de consulta não tem conhecimento das relações entre colunas e chaves de partição. Por esse motivo, é necessário incluir um predicado na coluna e na chave de partição da consulta para garantir que a consulta não verifique mais dados do que o necessário.

Por exemplo, suponha que a tabela `sales` no exemplo anterior também tenha uma coluna `sold_at` do tipo de dados `TIMESTAMP`. Se você quiser somente a receita de um intervalo de tempo específico, escreva a consulta da seguinte forma:

```
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'
```

Para obter mais informações sobre a diferença entre consultar tabelas Hive e Iceberg, consulte Como gravar consultas em campos de carimbo de data e hora que também são particionados por tempo.

O que é bucketing?

Bucketing é uma forma de organizar os registros de um conjunto de dados em categorias chamadas buckets.

Esse significado de bucket e bucketing é diferente dos buckets do Amazon S3 e não devem ser confundidos. No bucketing de dados, os registros que têm o mesmo valor para uma propriedade vão para o mesmo bucket. Os registros são distribuídos da forma mais uniforme possível entre os buckets, de modo que cada bucket tenha aproximadamente a mesma quantidade de dados.

Na prática, os buckets são arquivos, e uma função de hash determina o bucket em que um registro entrará. Um conjunto de dados em bucket terá um ou mais arquivos por bucket por partição. O bucket ao qual um arquivo pertence está codificado no nome do arquivo.

Benefícios do bucketing

O bucketing é útil quando um conjunto de dados é classificado em bucket por determinada propriedade, e convém recuperar registros nos quais essa propriedade tenha determinado valor. Como os dados estão em buckets, o Athena pode usar o valor para determinar quais arquivos examinar. Por exemplo, suponha que um conjunto de dados esteja em buckets `customer_id` e você queira localizar todos os registros de um cliente específico. O Athena determina o bucket que contém os registros e lê somente os arquivos desse bucket.

Bons candidatos para bucketing ocorrem quando há colunas que têm alta cardinalidade (ou seja, com muitos valores distintos), estão uniformemente distribuídas e que você frequentemente consulta buscando valores específicos.

> **Note**
>
> O Athena não é compatível com o uso de `INSERT INTO` para adicionar novos registros a tabelas em bucket.

Tipos de dados compatíveis com a filtragem em colunas em bucket

É possível adicionar filtros em colunas classificadas por buckets com determinados tipos de dados. O Athena permite essa filtragem nas colunas em bucket com os seguintes tipos de dados:

- BOOLEAN
- BYTE
- DATA
- DOUBLE
- FLOAT
- INT
- LONG
- SHORT
- STRING
- VARCHAR

Compatível com Hive e Spark

O mecanismo Athena versão 2 é compatível com conjuntos de dados em buckets usando o algoritmo de bucket do Hive, e o mecanismo Athena versão 3 também é compatível com o algoritmo de bucketing do Apache Spark. O bucketing Hive é o padrão. Se seu conjunto de dados for classificado em buckets usando o algoritmo Spark, use a cláusula `TBLPROPERTIES` para definir o valor da propriedade `bucketing_format` como `spark`.

> **Note**
>
> O Athena tem um limite de 100 partições em uma consulta `CREATE TABLE AS SELECT` (CTAS). Da mesma forma, é possível adicionar apenas, no máximo, 100 partições a uma tabela de destino com uma instrução INSERT INTO. Esse limite de cem partições se aplica somente quando a tabela está em buckets e é particionada.
> Se exceder essa limitação, você poderá receber a mensagem de erro HIVE_TOO_MANY_OPEN_PARTITIONS: Exceeded limit of 100 open writers for partitions/buckets (Limite excedido de 100 gravadores abertos para partições/buckets). Para contornar essa limitação, é possível usar uma instrução CTAS e uma série de instruções `INSERT INTO` que criam ou inserem até 100 partições cada. Para ter mais informações, consulte Usar CTAS e INSERT INTO para resolver o limite de 100 partições.

Exemplo de CREATE TABLE para bucketing

Para criar uma tabela para um conjunto de dados em buckets existente, use a cláusula `CLUSTERED BY (column)` seguida da cláusula `INTO N BUCKETS`. A cláusula `INTO N BUCKETS` especifica o número de buckets nos quais os dados são classificados.

No exemplo de `CREATE TABLE` a seguir, o conjunto de dados `sales` é classificado por `customer_id` em oito buckets usando o algoritmo Spark. A instrução `CREATE TABLE` usa as cláusulas`CLUSTERED BY` e `TBLPROPERTIES` para definir as propriedades adequadamente.

```
CREATE EXTERNAL TABLE sales (...)
...
CLUSTERED BY (`customer_id`) INTO 8 BUCKETS
...
TBLPROPERTIES (
  'bucketing_format' = 'spark'
)
```

Exemplo de CREATE TABLE AS (CTAS) para bucketing

Para especificar o bucketing com `CREATE TABLE AS`, use os parâmetros `bucketed_by` e `bucket_count`, como no exemplo a seguir.

```
CREATE TABLE sales
WITH (
  ...
  bucketed_by = ARRAY['customer_id'],
  bucket_count = 8
)
AS SELECT ...
```

Exemplo de consulta de bucketing

O exemplo de consulta a seguir pesquisa os nomes dos produtos que um cliente específico comprou ao longo de uma semana.

```
SELECT DISTINCT product_name
FROM sales
WHERE sales_date BETWEEN '2023-02-27' AND '2023-03-05'
AND customer_id = 'c123'
```

Se essa tabela for particionada por `sales_date` e classificada em buckets por `customer_id`, o Athena poderá calcular o bucket em que os registros do cliente estão. No máximo, o Athena lê um arquivo por partição.

Recursos adicionais do

Para obter um exemplo de `CREATE TABLE AS` que cria tabelas em bucket e particionadas, consulte Exemplo: criar tabelas em bucket e particionadas.

Exemplos de consultas CTAS

Use os exemplos a seguir para criar consultas CTAS. Para obter informações sobre a sintaxe de CTAS, consulte CREATE TABLE AS.

Nesta seção:

- 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 Exemplo: duplicar tabela selecionando todas as colunas

O exemplo a seguir cria uma tabela copiando todas as colunas de uma tabela:

```
CREATE TABLE new_table AS
SELECT *
FROM old_table;
```

Na seguinte variação do mesmo exemplo, a instrução `SELECT` também inclui uma cláusula `WHERE`. Nesse caso, a consulta seleciona somente as linhas da tabela que satisfazem a cláusula `WHERE`:

```
CREATE TABLE new_table AS
SELECT *
FROM old_table
WHERE condition;
```

Example Exemplo: selecionar colunas específicas de uma ou mais tabelas

O exemplo a seguir cria uma consulta que é executada em um conjunto de colunas de outra tabela:

```
CREATE TABLE new_table AS
SELECT column_1, column_2, ... column_n
FROM old_table;
```

Essa variação do mesmo exemplo cria uma tabela a partir de colunas específicas de várias tabelas:

```
CREATE TABLE new_table AS
SELECT column_1, column_2, ... column_n
FROM old_table_1, old_table_2, ... old_table_n;
```

Example Exemplo: criar uma cópia vazia de uma tabela existente

O exemplo a seguir usa `WITH NO DATA` para criar uma tabela vazia e que tenha o mesmo esquema da tabela original:

```
CREATE TABLE new_table
AS SELECT *
FROM old_table
WITH NO DATA;
```

Example Exemplo: especificar armazenamento de dados e formatos de compactação

Com a CTAS, é possível usar a tabela de origem para criar outra tabela em um formato diferente.

Use a propriedade `format` para especificar `ORC`, `PARQUET`, `AVRO`, `JSON` ou `TEXTFILE` como o formato de armazenamento para a nova tabela.

Para os formatos de armazenamento `PARQUET`, `ORC`, `TEXTFILE` e `JSON`, use a propriedade `write_compression` para especificar o formato de compactação para os dados da nova tabela. Para obter mais informações sobre os formatos de compactação suportados por cada formato de arquivo, consulte Suporte a compactação no Athena.

O exemplo a seguir especifica que os dados na tabela `new_table` são armazenados em formato Parquet e usam compactação Snappy. A compactação padrão para Parquet é `GZIP`.

```
CREATE TABLE new_table
WITH (
      format = 'Parquet',
      write_compression = 'SNAPPY')
AS SELECT *
FROM old_table;
```

O exemplo a seguir especifica que os dados na tabela `new_table` são armazenados em formato ORC e usam compactação Snappy. A compactação padrão para ORC é ZLIB.

```
CREATE TABLE new_table
WITH (format = 'ORC',
      write_compression = 'SNAPPY')
AS SELECT *
FROM old_table ;
```

O exemplo a seguir especifica esses dados na tabela `new_table` são armazenados em formato textfile usando a compactação Snappy. A compactação padrão para os formatos textfile e JSON é GZIP.

```
CREATE TABLE new_table
WITH (format = 'TEXTFILE',
      write_compression = 'SNAPPY')
AS SELECT *
FROM old_table ;
```

Example Exemplo: gravar resultados da consulta em um formato diferente

A seguinte consulta CTAS seleciona todos os registros de `old_table`, que poderiam ser armazenados em CSV ou em outro formato, e cria uma nova tabela com os dados subjacentes salvos no Amazon S3 no formato 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 Exemplo: criar tabelas não particionadas

Os exemplos a seguir criam tabelas que não são particionadas. Os dados da tabela são armazenados em formatos diferentes. Alguns desses exemplos especificam o local externo.

O exemplo a seguir cria uma consulta CTAS que armazena os resultados como um arquivo de texto:

```
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;
```

No exemplo a seguir, os resultados são armazenados em Parquet, e os local padrão dos resultados é usado:

```
CREATE TABLE ctas_parquet_unpartitioned
WITH (format = 'PARQUET')
AS SELECT key1, name1, comment1
FROM table1;
```

No a consulta a seguir, a tabela é armazenada em JSON, e colunas específicas são selecionadas entre os resultados da tabela original:

```
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;
```

No exemplo a seguir, o formato é ORC:

```
CREATE TABLE ctas_orc_unpartitioned
WITH (
     format = 'ORC')
AS SELECT key1, name1, comment1
FROM table1;
```

No exemplo a seguir, o formato é Avro:

```
CREATE TABLE ctas_avro_unpartitioned
WITH (
     format = 'AVRO',
     external_location = 's3://DOC-EXAMPLE-BUCKET/ctas_avro_unpartitioned/')
AS SELECT key1, name1, comment1
FROM table1;
```

Example Exemplo: criar tabelas particionadas

Os exemplos a seguir mostram consultas `CREATE TABLE AS SELECT` para tabelas particionadas em diferentes formatos de armazenamento, usando `partitioned_by` e outras propriedades na cláusula `WITH`. Para ver a sintaxe, consulte Propriedades da tabela CTAS. Para obter mais informações sobre como escolher as colunas para particionamento, consulte Particionamento e bucketing no Athena.

> **Note**
>
> Liste as colunas da partição no final da lista de colunas na instrução `SELECT`. Você pode particionar por mais de uma coluna e ter até 100 combinações únicas de partições e buckets. Por exemplo, você pode ter 100 partições se nenhum bucket for especificado.

```
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 Exemplo: criar tabelas em bucket e particionadas

O exemplo a seguir mostra uma consulta `CREATE TABLE AS SELECT` que usa tanto o particionamento quanto o armazenamento em bucket para armazenar os resultados das consultas no Amazon S3. Os resultados da tabela são particionados e armazenados em bucket por diferentes colunas. O Athena permite no máximo 100 combinações únicas de bucket e partição. Por exemplo, se você criar uma tabela com cinco buckets, é possível ter 20 partições com cinco buckets cada. Para ver a sintaxe, consulte Propriedades da tabela CTAS.

Para obter mais informações sobre como escolher as colunas para armazenamento em bucket, consulte Particionamento e bucketing no 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 Exemplo: criação de uma tabela do Iceberg com dados do Parquet

O exemplo a seguir cria uma tabela do Iceberg com arquivos de dados do Parquet. Os arquivos são particionados por mês usando a coluna `dt` na `table1`. O exemplo atualiza as propriedades de retenção na tabela para que dez snapshots sejam retidos, por padrão, em cada ramificação da tabela. Os snapshots dos últimos 7 dias também são retidos. Para obter mais informações sobre as propriedades de tabelas Iceberg no Athena, consulte Propriedade das tabelas.

```
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 Exemplo: criação de uma tabela do Iceberg com dados do Avro

O exemplo a seguir cria uma tabela do Iceberg com arquivos de dados do Avro particionados por `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;
```

Usar CTAS e INSERT INTO para ETL e análise de dados

Você pode usar as instruções Create Table as Select (CTAS) e INSERT INTO no Athena para extrair, transformar e carregar (ETL) dados no Amazon S3 para processamento de dados. Este tópico mostra como usar essas instruções para particionar e converter um conjunto de dados em um formato de dados colunar para otimizá-lo para análise de dados.

As instruções CTAS usam consultas SELECT padrão para criar novas tabelas. É possível usar uma instrução CTAS para criar um subconjunto de dados para análise. Em uma instrução CTAS, é possível particionar os dados, especificar compactação e converter os dados em um formato colunar, como Apache Parquet ou Apache ORC. Ao executar a consulta CTAS, as tabelas e as partições criadas são adicionadas automaticamente ao AWS Glue Data Catalog. Isso torna as novas tabelas e partições criadas imediatamente disponíveis para consultas subsequentes.

Instruções INSERT INTO inserem novas linhas em uma tabela de destino com base em uma instrução de consulta SELECT que é executada em uma tabela de origem. É possível usar instruções INSERT INTO para transformar e carregar dados da tabela de origem no formato CSV em dados da tabela de destino usando todas as transformações com suporte do CTAS.

Visão geral

No Athena, use uma instrução CTAS para executar uma conversão inicial em lote dos dados. Depois disso, use várias instruções INSERT INTO para fazer atualizações incrementais para a tabela criada pela instrução CTAS.

Etapas

- Etapa 1: Criar uma tabela com base no conjunto de dados original

- Etapa 2: Usar CTAS para particionar, converter e compactar os dados
- Etapa 3: Usar INSERT INTO para adicionar dados
- Etapa 4: Avaliar diferenças de custo e performance

Etapa 1: Criar uma tabela com base no conjunto de dados original

O exemplo neste tópico usa um subconjunto legível pelo Amazon S3 do conjunto de dados Global Historical Climatology Network Daily (GHCNd) da NOAA disponível publicamente. Os dados no Amazon S3 têm as características a seguir.

```
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
```

Os dados originais são armazenados no Amazon S3 sem partições. Os dados estão no formato CSV em arquivos, conforme mostrado a seguir.

```
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
```

Os tamanhos de arquivo neste exemplo são relativamente pequenos. Ao mesclá-los em arquivos maiores, é possível reduzir o número total de arquivos, o que melhora a performance das consultas. É possível usar instruções CTAS e INSERT INTO para melhorar a performance de consulta.

Como criar um banco de dados e uma tabela com base no conjunto de dados de exemplo

1. No console do Athena, escolha a Região da AWS Leste dos EUA (Norte da Virgínia). Execute todas as consultas neste tutorial em `us-east-1`.
2. No editor de consultas do Athena, execute o comando CREATE DATABASE para criar um banco de dados.

   ```
   CREATE DATABASE blogdb
   ```

3. Execute a seguinte instrução para criar uma tabela.

```
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/'
```

Etapa 2: Usar CTAS para particionar, converter e compactar os dados

Depois de criar uma tabela, é possível usar uma única instrução CTAS para converter os dados para o formato Parquet com compactação Snappy e para particionar os dados por ano.

A tabela criada na Etapa 1 tem um campo `date` com a data formatada como `YYYYMMDD` (por exemplo, `20100104`). Como a nova tabela será particionada em `year`, a instrução de exemplo no procedimento a seguir usa a função Presto `substr("date",1,4)` para extrair o valor `year` do campo `date`.

Como converter os dados para o formato Parquet com compactação Snappy, particionando por ano

- Execute a instrução CTAS a seguir, substituindo *`your-bucket`* pelo local do seu bucket do 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**
>
> Neste exemplo, a tabela criada inclui apenas os dados de 2015 a 2019. Na Etapa 3, adicione novos dados a essa tabela usando o comando INSERT INTO.

Quando a consulta for concluída, siga o procedimento abaixo para verificar a saída no local do Amazon S3 especificado na instrução CTAS.

Como ver as partições e os arquivos Parquet criados pela instrução CTAS

1. Para mostrar as partições criadas, execute o seguinte comando da AWS CLI. Certifique-se de incluir a barra final (/).

   ```
   aws s3 ls s3://DOC-EXAMPLE-BUCKET/optimized-data/
   ```

   A saída mostra as partições.

   ```
      PRE year=2015/
      PRE year=2016/
      PRE year=2017/
      PRE year=2018/
      PRE year=2019/
   ```

2. Para ver os arquivos Parquet, execute o seguinte comando. Observe que a opção | head -5, que restringe a saída aos primeiros cinco resultados, não está disponível no Windows.

```
aws s3 ls s3://DOC-EXAMPLE-BUCKET/optimized-data/ --recursive --human-readable |
 head -5
```

A saída será semelhante à seguinte.

```
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
```

Etapa 3: Usar INSERT INTO para adicionar dados

Na Etapa 2, você usou o CTAS para criar uma tabela com partições para os anos de 2015 a 2019. No entanto, o conjunto de dados original também contém dados para os anos de 2010 a 2014. Agora, adicione esses dados usando uma instrução INSERT INTO.

Como adicionar dados à tabela usando uma ou mais instruções INSERT INTO

1. Execute o seguinte comando INSERT INTO, especificando os anos antes de 2015 na cláusula 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. Execute o comando `aws s3 ls` novamente, usando a seguinte sintaxe.

```
aws s3 ls s3://DOC-EXAMPLE-BUCKET/optimized-data/
```

A saída mostra as novas partições.

```
     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. Para ver a redução no tamanho do conjunto de dados obtido por meio do uso da compactação e do armazenamento colunar no formato Parquet, execute o seguinte comando.

```
aws s3 ls s3://DOC-EXAMPLE-BUCKET/optimized-data/ --recursive --human-readable --
summarize
```

Os resultados a seguir mostram que o tamanho do conjunto de dados após o Parquet com compactação Snappy é 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. Se mais dados CSV forem adicionados à tabela original, você pode adicionar esses dados à tabela Parquet usando instruções INSERT INTO. Por exemplo, se você tiver novos dados para o ano de 2020, poderá executar a instrução INSERT INTO a seguir. A instrução adiciona os dados e a partição relevante à tabela `new_parquet`.

```
INSERT INTO new_parquet
```

```
SELECT id,
         date,
         element,
         datavalue,
         mflag,
         qflag,
         sflag,
         obstime,
         substr("date",1,4) AS year
FROM original_csv
WHERE cast(substr("date",1,4) AS bigint) = 2020
```

> **Note**
>
> A instrução INSERT INTO oferece suporte à gravação de, no máximo, 100 partições na tabela de destino. No entanto, para adicionar mais de 100 partições, você pode executar várias instruções INSERT INTO. Para ter mais informações, consulte Usar CTAS e INSERT INTO para resolver o limite de 100 partições.

Etapa 4: Avaliar diferenças de custo e performance

Depois de transformar os dados, é possível avaliar os ganhos de performance e a economia de custos executando as mesmas consultas nas tabelas novas e antigas e comparando os resultados.

> **Note**
>
> Para obter informações de custo por consulta do Athena, consulte Definição de preço do Amazon Athena.

Como avaliar diferenças de custo e ganhos de performance

1. Execute a seguinte consulta na tabela original. A consulta localiza o número de IDs distintos para cada valor do ano.

```
SELECT substr("date",1,4) as year,
       COUNT(DISTINCT id)
FROM original_csv
GROUP BY 1 ORDER BY 1 DESC
```

2. Observe o tempo em que a consulta foi executada e a quantidade de dados verificados.
3. Execute a mesma consulta na nova tabela, observando o tempo de execução da consulta e a quantidade de dados verificados.

```
SELECT year,
  COUNT(DISTINCT id)
FROM new_parquet
GROUP BY 1 ORDER BY 1 DESC
```

4. Compare os resultados e calcule a diferença de performance e custo. Os seguintes resultados de exemplo mostram que a consulta de teste na nova tabela foi mais rápida e mais econômica do que a consulta na tabela antiga.

Tabela	Runtime	Dados examinados
Original	16,88 segundos	11,35 GB
Novo	3,79 segundos	428,05 MB

5. Execute a seguinte consulta de exemplo na tabela original. A consulta calcula a temperatura máxima média (Celsius), a temperatura mínima média (Celsius) e a precipitação média (mm) da Terra em 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. Observe o tempo em que a consulta foi executada e a quantidade de dados verificados.
7. Execute a mesma consulta na nova tabela, observando o tempo de execução da consulta e a quantidade de dados verificados.

```
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. Compare os resultados e calcule a diferença de performance e custo. Os seguintes resultados de exemplo mostram que a consulta de teste na nova tabela foi mais rápida e mais econômica do que a consulta na tabela antiga.

Tabela	Runtime	Dados examinados
Original	18,65 segundos	11,35 GB
Novo	1,92 segundos	68 MB

Resumo

Este tópico mostrou como executar operações ETL usando as instruções CTAS e INSERT INTO no Athena. Você executou o primeiro conjunto de transformações usando uma instrução CTAS que converteu dados para o formato Parquet com compactação Snappy. A instrução CTAS também converteu o conjunto de dados de não particionado em particionado. Isso reduziu seu tamanho e reduziu os custos de execução das consultas. Quando novos dados são disponibilizados, é possível usar uma instrução INSERT INTO para transformar e carregar os dados na tabela criada com a instrução CTAS.

Usar CTAS e INSERT INTO para resolver o limite de 100 partições

Athena tem um limite de 100 partições por consulta `CREATE TABLE AS SELECT` (CTAS). Da mesma forma, é possível adicionar, no máximo, 100 partições a uma tabela de destino com uma instrução INSERT INTO.

Se exceder essa limitação, você poderá receber a mensagem de erro HIVE_TOO_MANY_OPEN_PARTITIONS: Exceeded limit of 100 open writers for partitions/buckets (Limite excedido de 100 gravadores abertos para partições/buckets). Para contornar essa limitação, é possível usar uma instrução CTAS e uma série de instruções `INSERT INTO` que criam ou inserem até 100 partições cada.

O exemplo neste tópico usa um banco de dados chamado `tpch100` com dados que residem no local do bucket do Amazon S3 s3://DOC-EXAMPLE-BUCKET/.

Como usar CTAS e INSERT INTO para criar uma tabela com mais de 100 partições

1. Use uma instrução `CREATE EXTERNAL TABLE` para criar uma tabela particionada no campo desejado.

   A instrução de exemplo a seguir particiona os dados de acordo com a coluna `l_shipdate`. A tabela tem 2.525 partições.

```
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. Execute um comando `SHOW PARTITIONS <table_name>` conforme o seguinte para listar as partições.

```
SHOW PARTITIONS lineitem_parq_partitioned
```

Veja a seguir os resultados parciais de exemplo.

```
/*
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. Execute uma consulta CTAS para criar uma tabela particionada.

   O exemplo a seguir cria uma tabela chamada `my_lineitem_parq_partitioned` e usa a cláusula `WHERE` para restringir `DATE` a um valor anterior a `1992-02-01`. Como o conjunto de dados de exemplo começa com janeiro de 1992, somente partições para janeiro de 1992 são criadas.

```
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. Execute o comando `SHOW PARTITIONS` para verificar se a tabela contém as partições desejadas.

```
SHOW PARTITIONS my_lineitem_parq_partitioned;
```

   As partições no exemplo são de janeiro de 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. Use uma instrução `INSERT INTO` para adicionar partições à tabela.

   O exemplo a seguir adiciona partições para as datas do mês de fevereiro de 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. Execute `SHOW PARTITIONS` novamente.

```
SHOW PARTITIONS my_lineitem_parq_partitioned;
```

A tabela de exemplo agora tem partições de janeiro e fevereiro de 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. Continue a usar instruções `INSERT INTO` que leem e adicionam até 100 partições cada. Continue até atingir o número de partições necessárias.

> **Important**
>
> Ao definir a condição `WHERE`, certifique-se de que as consultas não se sobreponham. Caso contrário, algumas partições podem ter dados duplicados.

Referência de SerDe

O Athena oferece várias bibliotecas SerDe para analisar dados de formatos diferentes, como CSV, JSON, Parquet e ORC. O Athena não aceita SerDes personalizado.

Tópicos

- Usar um SerDe
- SerDes e formatos de dados compatíveis

Usar um SerDe

Um SerDe (serializador/desserializador) é uma maneira na qual o Athena interage com os dados em vários formatos.

É o SerDe especificado por você, e não a DDL, que define o esquema de tabela. Em outras palavras, o SerDe pode substituir a configuração de DDL que você especifica no Athena ao criar a tabela.

Para usar um SerDe em consultas

Para usar um SerDe ao criar uma tabela no Athena, siga um destes métodos:

- Especifique `ROW FORMAT DELIMITED` e use as instruções DDL para determinar os delimitadores de campo, como no exemplo a seguir. Quando você especifica `ROW FORMAT DELIMITED`, por padrão, o Athena usa o LazySimpleSerDe.

```
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
ESCAPED BY '\\'
COLLECTION ITEMS TERMINATED BY '|'
```

```
MAP KEYS TERMINATED BY ':'
```

Para ver exemplos de `ROW FORMAT DELIMITED`, consulte os seguintes tópicos:

LazySimpleSerDe para arquivos CSV, TSV e com delimitação personalizada

Consultar os logs do Amazon CloudFront

Consultar os logs do Amazon EMR

Consultar os logs de fluxo do Amazon VPC

Usar CTAS e INSERT INTO para ETL e análise de dados

- Use `ROW FORMAT SERDE` para especificar claramente o tipo de SerDe que o Athena deve usar ao ler e gravar dados na tabela. O exemplo a seguir especifica o LazySimpleSerDe. Para especificar os delimitadores, use `WITH SERDEPROPERTIES`. As propriedades especificadas por `WITH SERDEPROPERTIES` correspondem às instruções separadas (como `FIELDS TERMINATED BY`) no exemplo de `ROW FORMAT DELIMITED`.

```
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
WITH SERDEPROPERTIES (
'serialization.format' = ',',
'field.delim' = ',',
'collection.delim' = '|',
'mapkey.delim' = ':',
'escape.delim' = '\\'
)
```

Para ver exemplos de `ROW FORMAT SERDE`, consulte os seguintes tópicos:

Avro SerDe

Grok SerDe

Bibliotecas SerDe JSON

OpenCSVSerDe para processar CSV

Regex SerDe

SerDes e formatos de dados compatíveis

O Athena permite a criação de tabelas e a consulta de dados em formatos CSV, TSV, com delimitação personalizada e JSON, dados de formatos relacionados ao Hadoop: ORC, Apache Avro e Parquet, logs do Logstash, logs do AWS CloudTrail e logs do Apache WebServer.

> **Note**
>
> Os formatos listados nesta seção são usados pelo Athena para ler dados. Para obter informações sobre os formatos que o Athena usa para gravar dados ao executar consultas CTAS, veja Criar uma tabela a partir de resultados de consultas (CTAS).

Para criar tabelas e consultar dados nesses formatos no Athena, especifique uma classe de serializador/desserializador (SerDe) para que o Athena saiba qual formato é usado e como analisar os dados.

Essa tabela lista os formatos de dados compatíveis com o Athena e as bibliotecas SerDe correspondentes.

SerDe é uma biblioteca personalizada que informa ao catálogo de dados usado pelo Athena como processar os dados. Você especifica um tipo de SerDe listando-o claramente na parte `ROW FORMAT` da instrução `CREATE TABLE` no Athena. Em alguns casos, você pode omitir o nome do SerDe porque o Athena usa alguns tipos de SerDe por padrão para determinados tipos de formatos de dados.

Formatos de dados e SerDes compatíveis

Formato de dados	Descrição	Tipos de SerDe compatíveis com o Athena
Amazon Ion	O Amazon Ion é um formato de dados autodescritivo e com tipagem rica que é um superconjunto de JSON. Esse formato de código aberto foi desenvolvido pela Amazon.	Use a Amazon Ion Hive SerDe.
Apache Avro	Um formato para armazenar dados no Hadoop que	Use o Avro SerDe.

Formato de dados	Descrição	Tipos de SerDe compatíveis com o Athena
	usa esquemas baseados em JSON para valores de registro.	
Apache Parquet	Um formato para armazenamento colunar de dados no Hadoop.	Use o Parquet SerDe e a compactação SNAPPY.
Logs do Apache WebServer	Um formato para armazenar logs no Apache WebServer.	Use o Grok SerDe ou Regex SerDe.
Logs do CloudTrail	Um formato para armazenar logs no CloudTrail.	• Use a Hive JSON SerDe. Para ter mais informações, consulte Consultar os logs do AWS CloudTrail.
Comma-Separated Values (CSV – Valores separados por vírgula)	Em dados em formato CSV, cada linha representa um registro de dados, e cada registro consiste em um ou mais campos, separados por vírgulas.	• Use o LazySimpleSerDe para arquivos CSV, TSV e com delimitação personalizada caso os dados não incluam valores entre aspas ou usem o formato `java.sql.Timestamp` . • Use o OpenCSVSerDe para processar CSV quando os dados incluírem aspas nos valores ou usarem o formato numérico UNIX para `TIMESTAMP` (por exemplo, `1564610311`).

Formato de dados	Descrição	Tipos de SerDe compatíveis com o Athena
Delimitação personalizada	Em dados nesse formato, cada linha representa um registro de dados, e os registros são separados por um delimitador de caractere único personalizado.	Use o LazySimpleSerDe para arquivos CSV, TSV e com delimitação personalizada e especifique um delimitador de caractere único personalizado.
JSON (JavaScript Object Notation)	Em dados JSON, cada linha representa um registro de dados, e cada registro consiste em pares de atributo-valor e matrizes, separados por vírgulas.	• Use o Hive JSON SerDe. • Use o OpenX JSON SerDe.
Logs do Logstash	Um formato para armazenar logs no Logstash.	Use o Grok SerDe.
Optimized Row Columnar (ORC – Colunar de linha otimizada)	Um formato para armazenam ento colunar otimizado de dados do Hive.	Use o ORC SerDe e a compactação ZLIB.
Tab-Separated Values (TSB – Valores separados por tabulação)	Em dados em formato TSV, cada linha representa um registro de dados, e cada registro consiste em um ou mais campos, separados por tabulações.	Use o LazySimpleSerDe para arquivos CSV, TSV e com delimitação personalizada e especifique o caractere separador como `FIELDS TERMINATED BY '\t'` .

Tópicos

- Amazon Ion Hive SerDe
- Avro SerDe
- Grok SerDe
- Bibliotecas SerDe JSON

- LazySimpleSerDe para arquivos CSV, TSV e com delimitação personalizada
- OpenCSVSerDe para processar CSV
- ORC SerDe
- Parquet SerDe
- Regex SerDe

Amazon Ion Hive SerDe

Você pode usar o Amazon Ion Hive SerDe para consultar dados armazenados no formato Amazon Ion. O Amazon Ion é um formato de dados de código aberto ricamente tipado e autodescritivo. O formato Amazon Ion é usado por serviços como o Amazon Quantum Ledger Database(Amazon QLDB) e na linguagem de consulta SQL de código aberto PartiQL.

O Amazon Ion tem formatos binários e de texto intercambiáveis. Esse recurso combina a facilidade de uso do texto com a eficiência da codificação binária.

Para consultar dados do Amazon Ion no Athena, você pode usar o Amazon Ion Hive SerDe, que serializa e desserializa os dados do Amazon Ion. A desserialização permite que você execute consultas nos dados do Amazon Ion ou leia os dados para gravá-los em um formato diferente, como Parquet ou ORC. A serialização permite gerar dados no formato Amazon Ion usando consultas `CREATE TABLE AS SELECT` (CTAS) ou `INSERT INTO` para copiar dados de tabelas existentes.

> **Note**
>
> Como o Amazon Ion é um superconjunto de JSON, você pode usar o Amazon Ion Hive SerDe para consultar conjuntos de dados JSON em formatos diferentes do Amazon Ion. Ao contrário de outras bibliotecas de SerDe JSON, o Amazon Ion SerDe não espera que cada linha de dados esteja em uma única linha. Esse recurso é útil quando você deseja consultar conjuntos de dados JSON que passaram por reformatação automática ou usar caracteres de nova linha para dividir os campos em uma linha.

Para obter informações adicionais e exemplos de como consultar o Amazon Ion com o Athena, consulte Analise conjuntos de dados do Amazon Ion usando o Amazon Athena.

Nome do SerDe

- com.amazon.ionhiveserde.IonHiveSerDe

Considerações e limitações

- Campos duplicados: as structs do Amazon Ion são ordenadas e oferecem suporte a campos duplicados, o que não acontece com `STRUCT<>` e `MAP<>` do Hive. Assim, quando você desserializa um campo duplicado de uma struct do Amazon Ion, um único valor é escolhido de forma não determinística e os outros são ignorados.
- Tabelas de símbolos externos sem suporte: atualmente, o Athena não oferece suporte a tabelas de símbolos externos ou às seguintes propriedades do Amazon Ion Hive SerDe:
  - `ion.catalog.class`
  - `ion.catalog.file`
  - `ion.catalog.url`
  - `ion.symbol_table_imports`
- Extensões de arquivo: o Amazon Ion usa extensões de arquivo para determinar o codec de compactação a ser usado na desserialização de arquivos do Amazon Ion. Assim, os arquivos compactados devem ter a extensão que corresponde ao algoritmo de compactação usado. Por exemplo, se for usado o ZSTD, os arquivos correspondentes deverão ter a extensão `.zst`.
- Dados homogêneos: o Amazon Ion não tem restrições de tipos de dados que podem ser usados como valores em campos específicos. Por exemplo, dois documentos diferentes do Amazon Ion podem ter um campo com o mesmo nome que tenha tipos de dados diferentes. No entanto, como o Hive usa um esquema, todos os valores extraídos para uma única coluna do Hive devem ter o mesmo tipo de dado.
- Restrições de tipo de chave: ao serializar dados de outro formato para o Amazon Ion, certifique-se de que o tipo de chave do mapa seja `STRING`, `VARCHAR` ou `CHAR`. Embora o Hive permita usar qualquer tipo de dado primitivo como chave do mapa, os símbolos do Amazon Ion devem ser um tipo de string.
- Tipo union: atualmente, o Athena não oferece suporte ao tipo union do Hive.
- Tipo de dados “double”: no momento, o Amazon Ion não oferece suporte ao tipo de dados `double`.

Tópicos

- Usar CREATE TABLE para criar tabelas do Amazon Ion
- Usar CTAS e INSERT INTO para criar tabelas do Amazon Ion
- Usar propriedades do Amazon Ion SerDe
- Usar extratores de caminhos

Usar CREATE TABLE para criar tabelas do Amazon Ion

Para criar uma tabela no Athena com dados armazenados no formato Amazon Ion, você pode usar uma das seguintes técnicas em uma instrução CREATE TABLE:

- Especifique `STORED AS ION`. Neste uso, você não precisa especificar explicitamente o Amazon Ion Hive SerDe. Essa escolha é a opção mais direta.
- Especifique os caminhos de classe do Amazon Ion nos campos `ROW FORMAT SERDE`, `INPUTFORMAT` e `OUTPUTFORMAT`.

Você também pode usar instruções `CREATE TABLE AS SELECT` (CTAS) para criar tabelas do Amazon Ion no Athena. Para ter mais informações, consulte Usar CTAS e INSERT INTO para criar tabelas do Amazon Ion.

Especificar STORED AS ION

O exemplo a seguir de instrução `CREATE TABLE` usa `STORED AS ION` antes da cláusula `LOCATION` para criar uma tabela com base em dados de voo no formato Amazon Ion. A cláusula `LOCATION` especifica o bucket ou a pasta em que estão localizados os arquivos de entrada no formato Ion. Todos os arquivos no local especificado estão verificados.

```
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/'
```

Especificar os caminhos de classe do Amazon Ion

Em vez de usar a sintaxe `STORED AS ION`, você pode especificar explicitamente os valores de caminho de classe do Ion para as cláusulas `ROW FORMAT SERDE`, `INPUTFORMAT` e `OUTPUTFORMAT`, como mostrado a seguir.

Parâmetro	Caminho de classe do Ion
`ROW FORMAT SERDE`	`'com.amazon.ionhiveserde.IonHiveSerDe'`
`STORED AS INPUTFORMAT`	`'com.amazon.ionhiveserde.formats.IonInputFormat'`
`OUTPUTFORMAT`	`'com.amazon.ionhiveserde.formats.IonOutputFormat'`

A consulta DDL a seguir usa essa técnica para criar a mesma tabela externa do exemplo anterior.

```
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/'
```

Para obter informações sobre as propriedades SerDe para instruções `CREATE TABLE` no Athena, consulte Usar propriedades do Amazon Ion SerDe.

Usar CTAS e INSERT INTO para criar tabelas do Amazon Ion

Você pode usar as instruções `CREATE TABLE AS SELECT` (CTAS) e `INSERT INTO` para copiar ou inserir dados de uma tabela em uma nova tabela no formato Amazon Ion no Athena.

Em uma consulta CTAS, especifique `format='ION'` na cláusula `WITH`, como no exemplo a seguir.

```
CREATE TABLE new_table
```

```
WITH (format='ION')
AS SELECT * from existing_table
```

Por padrão, o Athena serializa os resultados do Amazon Ion em formato binário Ion, mas você também pode usar o formato de texto. Para usar o formato de texto, especifique `ion_encoding = 'TEXT'` na cláusula CTAS `WITH`, como no exemplo a seguir.

```
CREATE TABLE new_table
WITH (format='ION', ion_encoding = 'TEXT')
AS SELECT * from existing_table
```

Para obter mais informações sobre as propriedades específicas do Amazon Ion na cláusula CTAS `WITH`, consulte a seção a seguir.

Propriedades do Amazon Ion da cláusula CTAS WITH

Em uma consulta CTAS, você pode usar a cláusula `WITH` para especificar o formato Amazon Ion e, opcionalmente, especificar a codificação do Amazon Ion e/ou o algoritmo de compactação a usar.

format

Você pode especificar a palavra-chave `ION` como a opção de formato na cláusula `WITH` de uma consulta CTAS. Ao fazer isso, a tabela que você cria usa o formato especificado de `IonInputFormat` para leituras e serializa dados no formato especificado de `IonOutputFormat`.

O exemplo a seguir especifica que a consulta CTAS usa o formato Amazon Ion.

```
WITH (format='ION')
```

ion_encoding

Opcional

Padrão: `BINARY`

Valores: `BINARY`, `TEXT`

Especifica se os dados são serializados no formato binário ou no formato de texto do Amazon Ion. O exemplo a seguir especifica o formato de texto do Amazon Ion.

```
WITH (format='ION', ion_encoding='TEXT')
```

write_compression

Opcional

Padrão: `GZIP`

Valores: `GZIP`, `ZSTD`, `BZIP2`, `SNAPPY`, `NONE`

Especifica o algoritmo de compactação a ser usado para compactar os arquivos de saída.

O exemplo a seguir especifica que a consulta CTAS grava a saída no formato Amazon Ion usando o algoritmo de compactação Zstandard.

```
WITH (format='ION', write_compression = 'ZSTD')
```

Para obter mais informações sobre compactação de dados no Athena, consulte Suporte a compactação no Athena.

Para obter informações sobre outras propriedades de CTAS em Athena, consulte Propriedades da tabela CTAS.

Usar propriedades do Amazon Ion SerDe

Este tópico contém informações sobre as propriedades SerDe para instruções `CREATE TABLE` no Athena. Para obter mais informações e exemplos do uso de propriedades do Amazon Ion SerDe, consulte SerDe properties (Propriedades SerDe) na documentação sobre o Amazon Ion Hive SerDe no GitHub.

Especificar propriedades SerDe do Amazon Ion

Para especificar propriedades para o Amazon Ion Hive SerDe na instrução `CREATE TABLE`, use a cláusula `WITH SERDEPROPERTIES`. Como `WITH SERDEPROPERTIES` é um subcampo da cláusula `ROW FORMAT SERDE`, você deve especificar primeiro `ROW FORMAT SERDE` e o caminho da classe do Amazon Ion Hive SerDe, como mostra a sintaxe a seguir.

```
...
ROW FORMAT SERDE
 'com.amazon.ionhiveserde.IonHiveSerDe'
WITH SERDEPROPERTIES (
 'property' = 'value',
 'property' = 'value',
```

```
...
)
```

Observe que, embora a cláusula `ROW FORMAT SERDE` seja obrigatória se você quiser usar `WITH SERDEPROPERTIES`, será possível usar `STORED AS ION` ou a sintaxe mais longa `INPUTFORMAT` e `OUTPUTFORMAT` para especificar o formato Amazon Ion.

Propriedades SerDe do Amazon Ion

As propriedades do Amazon Ion SerDe que podem ser usadas em instruções `CREATE TABLE` no Athena são listadas a seguir.

ion.encoding

Opcional

Padrão: `BINARY`

Valores: `BINARY`, `TEXT`

Esta propriedade declara se novos valores adicionados são serializados como Amazon Ion binário ou no formato de texto do Amazon Ion.

O exemplo de propriedade SerDe a seguir especifica o formato de texto do Amazon Ion.

```
'ion.encoding' = 'TEXT'
```

ion.fail_on_overflow

Opcional

Padrão: `true`

Valores: `true`, `false`

O Amazon Ion permite tipos numéricos arbitrariamente grandes, o que não é permitido pelo Hive. Por padrão, o SerDe falhará se o valor do Amazon Ion não se encaixar na coluna Hive, mas você pode usar a opção de configuração `fail_on_overflow` para permitir que o valor estoure em vez de falhar.

Essa propriedade pode ser definida no nível de tabela ou de coluna. Para defini-la no nível de tabela, especifique `ion.fail_on_overflow` como no exemplo a seguir. Isso define o comportamento padrão para todas as colunas.

```
'ion.fail_on_overflow' = 'true'
```

Para controlar uma coluna específica, defina o nome da coluna entre `ion` e `fail_on_overflow`, delimitado por pontos, como no exemplo a seguir.

```
'ion.<column>.fail_on_overflow' = 'false'
```

ion.path_extractor.case_sensitive

Opcional

Padrão: `false`

Valores: `true`, `false`

Determina se os nomes de campos do Amazon Ion devem diferenciar maiúsculas de minúsculas. Quando for definido como `false`, o SerDe ignorará a análise de maiúsculas e minúsculas nos nomes de campo do Amazon Ion.

Por exemplo, suponha que você tenha um esquema de tabela do Hive que defina um campo `alias` em minúsculas e um documento do Amazon Ion com os campos `alias` e `ALIAS`, como no exemplo a seguir.

```
-- Hive Table Schema
alias: STRING

-- Amazon Ion Document
{ 'ALIAS': 'value1'}
{ 'alias': 'value2'}
```

O exemplo a seguir mostra as propriedades SerDe e a tabela extraída resultante quando a diferenciação de maiúsculas e minúsculas for definida como `false`:

```
-- Serde properties
'ion.alias.path_extractor' = '(alias)'
'ion.path_extractor.case_sensitive' = 'false'

--Extracted Table
| alias    |
|----------|
```

```
| "value1" |
| "value2" |
```

O exemplo a seguir mostra as propriedades SerDe e a tabela extraída resultante quando a diferenciação de maiúsculas e minúsculas for definida como `true`:

```
-- Serde properties
'ion.alias.path_extractor' = '(alias)'
'ion.path_extractor.case_sensitive' = 'true'

--Extracted Table
| alias    |
|----------|
| "value2" |
```

No segundo caso, o `value1` do campo `ALIAS` é ignorado quando a diferenciação de maiúsculas e minúsculas é definida como `true` e o extrator de caminhos é especificado como `alias`.

ion.*<column>*.path_extractor

Opcional

Padrão: ND

Valores: string com caminho de pesquisa

Cria um extrator de caminhos com o caminho de pesquisa especificado para a coluna dada. Os extratores de caminhos mapeiam os campos do Amazon Ion para colunas do Hive. Se nenhum extrator de caminhos for especificado, o Athena criará dinamicamente extratores de caminhos em tempo de execução com base nos nomes das colunas.

O exemplo de extrator de caminhos a seguir mapeia `example_ion_field` para `example_hive_column`.

```
'ion.example_hive_column.path_extractor' = '(example_ion_field)'
```

Para obter mais informações sobre extratores de caminhos e caminhos de pesquisa, consulte Usar extratores de caminhos.

ion.timestamp.serialization_offset

Opcional

Padrão: `'Z'`

Valores: `OFFSET`, em que `OFFSET` é representado como *<signal>*`hh:mm`. Valores de exemplo: `01:00`, `+01:00`, `-09:30`, `Z` (UTC, mesmo que 00:00)

Ao contrário dos carimbos de data/hora do Apache Hive, que não têm fuso horário incorporado e são armazenados como um deslocamento da época UNIX, os carimbos de data/hora do Amazon Ion têm um deslocamento. Use essa propriedade para especificar o deslocamento na serialização para o Amazon Ion.

O exemplo a seguir adiciona um deslocamento de uma hora.

```
'ion.timestamp.serialization_offset' = '+01:00'
```

ion.serialize_null

Opcional

Padrão: `OMIT`

Valores: `OMIT`, `UNTYPED`, `TYPED`

O Amazon Ion SerDe pode ser configurado para serializar ou omitir colunas que tenham valores nulos. Você pode optar por gravar nulos fortemente tipados (`TYPED`) ou nulos não tipados (`UNTYPED`). Nulos fortemente tipados são determinados com base no mapeamento padrão de tipos entre o Amazon Ion e o Hive.

O exemplo a seguir especifica nulos fortemente tipados.

```
'ion.serialize_null'='TYPED'
```

ion.ignore_malformed

Opcional

Padrão: `false`

Valores: `true`, `false`

Quando for definida como `true`, a propriedade ignorará as entradas malformadas ou todo o arquivo se o SerDe não conseguir lê-lo. Para obter mais informações, consulte Ignore malformed (Ignorar entradas malformadas) na documentação do GitHub.

ion.*<column>*.serialize_as

Opcional

Padrão: tipo padrão para a coluna.

Valores: string contendo o tipo do Amazon Ion

Determina o tipo de dado do Amazon Ion no qual um valor é serializado. Como os tipos do Amazon Ion e do Hive nem sempre têm um mapeamento direto, alguns tipos do Hive têm vários tipos de dados válidos para serialização. Para serializar dados como um tipo de dado não padrão, use essa propriedade. Para obter mais informações sobre o mapeamento de tipos, consulte a página Type mapping (Mapeamento de tipos) do Amazon Ion no GitHub.

Por padrão, as colunas binárias do Hive são serializadas como blobs do Amazon Ion, mas também podem ser serializadas como um clob do Amazon Ion (objeto grande de caracteres). O exemplo a seguir serializa a coluna `example_hive_binary_column` como um clob.

```
'ion.example_hive_binary_column.serialize_as' = 'clob'
```

Usar extratores de caminhos

O Amazon Ion é um formato de arquivo em estilo de documento, mas o Apache Hive é um formato de coluna simples. Você pode usar propriedades especiais do Amazon Ion SerDe, chamadas de `path extractors`, para fazer o mapeamento entre os dois formatos. Os extratores de caminhos nivelam o formato hierárquico do Amazon Ion, mapeiam valores do Amazon Ion em relação às colunas do Hive e podem ser usados para renomear campos.

O Athena pode gerar seus extratores, mas você também poderá definir os próprios extratores, se necessário.

Extratores de caminhos gerados

Por padrão, o Athena pesquisa valores de nível superior do Amazon Ion que correspondam aos nomes de coluna do Hive e cria extratores de caminhos em tempo de execução com base nesses valores. Se o formato de dados do Amazon Ion corresponder ao esquema da tabela do Hive, o Athena gerará dinamicamente os extratores e você não precisará adicionar nenhum extrator de caminho adicional. Esses extratores de caminhos padrão não são armazenados nos metadados da tabela.

O exemplo a seguir mostra como o Athena gera extratores com base no nome da coluna.

```
-- 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/'
```

Os extratores no exemplo a seguir são gerados pelo Athena. O primeiro extrai o campo `identification` para a coluna `identification`, e o segundo extrai o campo `alias` para a coluna `alias`.

```
'ion.identification.path_extractor' = '(identification)'
'ion.alias.path_extractor' = '(alias)'
```

O exemplo a seguir mostra a tabela extraída.

```
|                 identification                    |  alias   |
|---------------------------------------------------|----------|
|{["name", "driver_license"],["John Smith", "XXXX"]} | "Johnny" |
```

Especificar os próprios extratores de caminhos

Se os campos do Amazon Ion não forem mapeados perfeitamente para as colunas do Hive, você poderá especificar seus próprios extratores de caminhos. Use a sintaxe a seguir na cláusula `WITH SERDEPROPERTIES` da instrução `CREATE TABLE`.

```
WITH SERDEPROPERTIES (
   "ion.path_extractor.case_sensitive" = "<Boolean>",
```

```
   "ion.<column_name>.path_extractor" = "<path_extractor_expression>"
)
```

> **Note**
>
> Por padrão, os extratores de caminhos não diferenciam maiúsculas de minúsculas. Para substituir essa configuração, defina a propriedade SerDe ion.path_extractor.case_sensitive como `true`.

Usar caminhos de pesquisa em extratores de caminhos

A sintaxe da propriedade SerDe para extratores de caminhos contém uma *<path_extractor_expression>*:

```
"ion.<column_name>.path_extractor" = "<path_extractor_expression>"
```

Você pode usar a *<path_extractor_expression>* para especificar um caminho de pesquisa que analise o documento do Amazon Ion e encontre os dados correspondentes. O caminho de pesquisa é colocado entre parênteses e pode conter um ou mais dos componentes separados por espaços a seguir.

- Curinga: corresponde todos os valores.
- Índice: corresponde o valor com o índice numérico especificado. Os índices são baseados em zero.
- Texto: corresponde todos os valores cujos nomes de campo são equivalentes ao texto especificado.
- Anotações: correspondem valores especificados por um componente de caminho empacotado com as anotações especificadas.

O exemplo a seguir mostra um documento do Amazon Ion e alguns exemplos de caminhos de pesquisa.

```
-- 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"
```

Exemplos de extrator

Nivelar e renomear campos

O exemplo a seguir mostra um conjunto de caminhos de pesquisa que nivelam e renomeiam campos. O exemplo usa caminhos de pesquisa para:

- Mapear a coluna `nickname` ao campo `alias`
- Mapear a coluna `name` ao subcampo `name` localizado no struct `identification`.

O exemplo a seguir mostra o documento do Amazon Ion.

```
-- Example Amazon Ion Document
{
    identification: {
        name: "John Smith",
        driver_license: "XXXX"
    },

    alias: "Johnny"
}
```

O exemplo a seguir mostra a instrução `CREATE TABLE` que define os extratores de caminhos.

```
-- 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/'
```

O exemplo a seguir mostra os dados extraídos.

```
-- Extracted Table
| name         |   nickname   |
|--------------|--------------|
| "John Smith" |  "Johnny"    |
```

Para obter mais informações sobre caminhos de pesquisa e exemplos adicionais, consulte a página Ion Java Path Extraction (Extração de caminhos Java em Ion) no GitHub.

Extrair dados de voo em formato de texto

O exemplo de consulta `CREATE TABLE` a seguir usa `WITH SERDEPROPERTIES` para adicionar extratores de caminhos para extrair dados de voo e especificar a codificação de saída como texto do Amazon Ion. O exemplo usa a sintaxe `STORED AS ION`.

```
CREATE EXTERNAL TABLE flights_ion (
    yr INT,
    quarter INT,
    month INT,
    dayofmonth INT,
    dayofweek INT,
    flightdate STRING,
    uniquecarrier STRING,
    airlineid INT,
)
ROW FORMAT SERDE
 'com.amazon.ionhiveserde.IonHiveSerDe'
WITH SERDEPROPERTIES (
 'ion.encoding' = 'TEXT',
 'ion.yr.path_extractor'='(year)',
 'ion.quarter.path_extractor'='(results quarter)',
 'ion.month.path_extractor'='(date month)')
STORED AS ION
LOCATION 's3://DOC-EXAMPLE-BUCKET/'
```

Avro SerDe

Nome do SerDe

Avro SerDe

Nome da biblioteca

org.apache.hadoop.hive.serde2.avro.AvroSerDe

Exemplos

O Athena não permite o uso de `avro.schema.url` para especificar o esquema de tabela por motivos de segurança. Usar `avro.schema.literal`. Para extrair esquema de dados no formato Avro, você pode usar o Apache `avro-tools-<version>.jar` com o parâmetro `getschema`. Isso retorna um esquema que você pode usar na instrução `WITH SERDEPROPERTIES`. Por exemplo:

```
java -jar avro-tools-1.8.2.jar getschema my_data.avro
```

O arquivo `avro-tools-<version>.jar` está localizado no subdiretório `java` da versão do Avro instalada. Para baixar o Avro, acesse Apache Avro releases (Versões do Apache Avro). Para baixar o Apache Avro Tools diretamente, acesse o repositório do Apache Avro Tools Maven.

Depois de obter o esquema, use uma instrução `CREATE TABLE` para criar uma tabela do Athena com base nos dados subjacentes do Avro armazenados no Amazon S3. Para especificar o Avro SerDe, use `ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'`. Conforme demonstrado no exemplo a seguir, é necessário especificar o esquema usando a cláusula `WITH SERDEPROPERTIES`, além de especificar os nomes das colunas e os tipos de dados correspondentes para a tabela.

> **Note**
>
> Substitua *`myregion`* em `s3://athena-examples-myregion/path/to/data/` pelo identificador da região onde o Athena é executado, por exemplo, `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/';
```

Execute a instrução `MSCK REPAIR TABLE` na tabela para atualizar metadados da partição.

```
MSCK REPAIR TABLE flights_avro_example;
```

Consulte as 10 principais cidades de partida pelo número total de partidas.

```
SELECT origin, count(*) AS total_departures
FROM flights_avro_example
WHERE year >= '2000'
GROUP BY origin
ORDER BY total_departures DESC
LIMIT 10;
```

> **Note**
>
> Os dados da tabela de voos foram extraídos dos Voos fornecidos pelo Departamento de Transportes dos EUA, Bureau of Transportation Statistics (Agência de Estatísticas de Transportes). Saturação do original removida.

Grok SerDe

O Logstash Grok SerDe é uma biblioteca com um conjunto de padrões especializados para desserialização de arquivos de texto desestruturados, normalmente logs. Cada padrão Grok é uma expressão regular nomeada. Você pode identificar e reutilizar esses padrões de desserialização conforme necessário. Isso facilita o uso de Grok, em comparação com o uso de expressões regulares. O Grok oferece um conjunto de padrões predefinidos. Você também pode criar padrões personalizados.

Para especificar o Grok SerDe ao criar uma tabela no Athena, use a cláusula `ROW FORMAT SERDE 'com.amazonaws.glue.serde.GrokSerDe'`, seguida da cláusula `WITH SERDEPROPERTIES` que especifica os padrões de correspondência em seus dados, em que:

- A expressão `input.format` define os padrões para que correspondam aos dados. É necessário.
- A expressão `input.grokCustomPatterns` define um padrão personalizado nomeado, que você poderá usar depois dentro da expressão `input.format`. Ela é opcional. Para incluir várias entradas padrão na expressão `input.grokCustomPatterns`, use o caractere de escape de nova linha (\n) para separá-las, da seguinte forma: `'input.grokCustomPatterns'='INSIDE_QS ([^\"]*)\nINSIDE_BRACKETS ([^\ \]]*)')`.
- As cláusulas `STORED AS INPUTFORMAT` e `OUTPUTFORMAT` são obrigatórias.
- A cláusula `LOCATION` especifica um bucket do Amazon S3, que pode conter vários objetos de dados. Todos os objetos de dados no bucket são desserializados para criar a tabela.

Exemplos

Esses exemplos dependem da lista de padrões Grok predefinidos. Consulte padrões predefinidos.

Exemplo 1

Este exemplo usa os dados de origem das entradas de maillog do Postfix salvas em `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>
```

A instrução a seguir cria uma tabela no Athena chamada `mygroktable` com base nos dados de origem usando um padrão personalizado e os padrões predefinidos que você especificar:

```
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/';
```

Comece com um padrão simples, como `%{NOTSPACE:column}`, para obter as colunas mapeadas primeiro e, em seguida, especifique as colunas, se necessário.

Exemplo 2

No exemplo a seguir, você cria uma consulta para logs Log4j. Os logs de exemplo têm as entradas neste formato:

```
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
```

Para consultar os dados desses logs:

- Adicione o padrão Grok ao `input.format` para cada coluna. Por exemplo, para `timestamp`, adicione `%{TIMESTAMP_ISO8601:timestamp}`. Para `loglevel`, adicione `%{LOGLEVEL:loglevel}`.
- Verifique se o padrão em `input.format` está de acordo com o formato exato do log mapeando os traços (`-`) e as vírgulas que separam as entradas no formato do 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/';
```

Exemplo 3

O exemplo a seguir de consulta de logs do Amazon S3 mostra a expressão `'input.grokCustomPatterns'` que inclui duas entradas de padrão, separadas pelo caractere de escape de nova linha (\n), conforme mostrado neste trecho da consulta de exemplo:

'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'
```

Bibliotecas SerDe JSON

No Athena, é possível usar bibliotecas SerDe para desserializar dados JSON. A desserialização converte os dados JSON para que possam ser serializados (gravados) em um formato diferente, como Parquet ou ORC.

- O nativo Hive JSON SerDe
- A OpenX JSON SerDe
- A Amazon Ion Hive SerDe

> **Note**
>
> As bibliotecas do Hive e do OpenX esperam que os dados em JSON estejam em uma única linha (não formatados), com registros separados por um caractere de nova linha. O Amazon Ion Hive SerDe não tem esse requisito e pode ser usado como alternativa porque o formato de dados Ion é um superconjunto do JSON.

Nomes de biblioteca

Use uma das seguintes opções:

org.apache.hive.hcatalog.data.JsonSerDe

org.openx.data.jsonserde.JsonSerDe

com.amazon.ionhiveserde.IonHiveSerDe

Hive JSON SerDe

Normalmente, o Hive JSON SerDe é usado para processar dados JSON como eventos. Esses eventos são representados como strings em uma só linha codificadas em JSON separadas por uma nova linha. O Hive JSON SerDe não permite chaves duplicadas nos nomes de chaves `map` ou `struct`.

> **Note**
>
> O SerDe espera que cada documento JSON esteja em uma única linha de texto, sem caracteres de terminação de linha separando os campos no registro. Se o texto JSON

estiver formatado para impressão, você poderá receber uma mensagem de erro como HIVE_CURSOR_ERROR: Row is not a valid JSON Object (HIVE_CURSOR_ERROR: a linha não é um objeto JSON válido) ou HIVE_CURSOR_ERROR: JsonParseException: Unexpected end-of-input: expected close marker for OBJECT (HIVE_CURSOR_ERROR: JSONParseException: Fim de entrada inesperado: marcador de fechamento esperado para OBJECT) quando tentar consultar a tabela após criá-la. Para obter mais informações, consulte JSON Data Files na documentação do OpenX SerDe no GitHub.

A instrução DDL de exemplo a seguir usa o Hive JSON SerDe para criar uma tabela com base em dados de publicidade online de exemplo. Na cláusula `LOCATION`, substitua *`myregion`* em `s3://DOC-EXAMPLE-BUCKET.elasticmapreduce/samples/hive-ads/tables/impressions` pela região onde o Athena é executado (por exemplo, `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';
```

Especificar formatos de carimbo de data/hora com o Hive JSON SerDe

Para analisar valores de carimbo de data/hora da string, você pode adicionar o subcampo `WITH SERDEPROPERTIES` à cláusula `ROW FORMAT SERDE` e usá-lo para especificar o parâmetro `timestamp.formats`. No parâmetro, especifique uma lista separada por vírgula de um ou mais padrões de carimbo de data/hora, como no seguinte exemplo:

```
...
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")
...
```

Para obter mais informações, consulte Carimbos de data/hora na documentação do Apache Hive.

Carregar a tabela para consulta

Após criar a tabela, execute MSCK REPAIR TABLE para carregá-la e torná-la consultável no Athena:

```
MSCK REPAIR TABLE impressions
```

Como consultar logs do CloudTrail

É possível usar o Hive JSON SerDe para consultar os logs do CloudTrail. Para obter mais informações e instruções `CREATE TABLE` de exemplo, consulte Consultar os logs do AWS CloudTrail.

OpenX JSON SerDe

Assim como o Hive JSON SerDe, você pode usar o OpenX JSON para processar dados JSON. Os dados também são representados como strings em uma só linha codificadas em JSON separadas por uma nova linha. Assim como ocorre com o Hive JSON SerDe, o OpenX JSON SerDe não permite chaves duplicadas nos nomes de chaves `map` ou `struct`.

> **Note**
>
> O SerDe espera que cada documento JSON esteja em uma única linha de texto, sem caracteres de terminação de linha separando os campos no registro. Se o texto JSON estiver formatado para impressão, você poderá receber uma mensagem de erro como HIVE_CURSOR_ERROR: Row is not a valid JSON Object (HIVE_CURSOR_ERROR: a linha não é um objeto JSON válido) ou HIVE_CURSOR_ERROR: JsonParseException:

Unexpected end-of-input: expected close marker for OBJECT (HIVE_CURSOR_ERROR: JSONParseException: Fim de entrada inesperado: marcador de fechamento esperado para OBJECT) quando tentar consultar a tabela após criá-la. Para obter mais informações, consulte JSON Data Files na documentação do OpenX SerDe no GitHub.

Propriedades opcionais

Ao contrário do Hive JSON SerDe, o OpenX JSON SerDe também tem as propriedades SerDe opcionais a seguir que podem ser úteis para resolver inconsistências nos dados.

ignore.malformed.json

Opcional. Quando definido como `TRUE`, permite ignorar a sintaxe JSON malformada. O padrão é `FALSE`.

dots.in.keys

Opcional. O padrão é `FALSE`. Quando definido como `TRUE`, permite que o SerDe substitua por sublinhados os pontos nos nomes-chave. Por exemplo, se o conjunto de dados JSON tem uma chave chamada `"a.b"`, você pode usar essa propriedade para definir o nome da coluna como `"a_b"` no Athena. Por padrão (sem esse SerDe), o Athena não permite pontos nos nomes de coluna.

case.insensitive

Opcional. O padrão é `TRUE`. Quando definido como `TRUE`, o SerDe converte todas as colunas em maiúsculas para minúsculas.

Para usar nomes de chave que diferenciam maiúsculas e minúsculas em seus dados, use `WITH SERDEPROPERTIES ("case.insensitive"= FALSE;)`. Depois, para cada chave que ainda não esteja totalmente em letras minúsculas, forneça um mapeamento do nome da coluna para o nome da propriedade usando a seguinte sintaxe:

```
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES ("case.insensitive" = "FALSE", "mapping.userid" = "userId")
```

Se você tiver duas chaves como `URL` e `Url` que são iguais quando estão em minúsculas, um erro como o seguinte pode ocorrer:

HIVE_CURSOR_ERROR: a linha não é um objeto JSON válido - JSONException: chave duplicada “url”

Para resolver isso, defina a propriedade `case.insensitive` como `FALSE` e mapeie as chaves para nomes diferentes, como no exemplo a seguir:

```
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES ("case.insensitive" = "FALSE", "mapping.url1" = "URL",
 "mapping.url2" = "Url")
```

mapeamento

Opcional. Mapeia os nomes das colunas para chaves JSON que não são idênticas aos nomes da coluna. O parâmetro `mapping` é útil quando os dados JSON contêm chaves que são palavras-chave. Por exemplo, se você tiver uma chave JSON chamada `timestamp`, use a seguinte sintaxe para mapear a chave para uma coluna chamada `ts`:

```
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES ("mapping.ts" = "timestamp")
```

Mapear nomes de campos aninhados com dois pontos para nomes compatíveis com o Hive

Se você tiver um nome de campo com dois pontos dentro de um `struct`, poderá usar a propriedade `mapping` para mapear o campo para um nome compatível com o Hive. Por exemplo, se suas definições de tipo de coluna contiverem `my:struct:field:string`, você poderá mapear a definição para `my_struct_field:string` incluindo a seguinte entrada em `WITH SERDEPROPERTIES`:

```
("mapping.my_struct_field" = "my:struct:field")
```

O exemplo a seguir mostra a instrução `CREATE TABLE` correspondente.

```
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")
```

Exemplo: dados de publicidade

A instrução DDL de exemplo a seguir usa o OpenX JSON SerDe para criar uma tabela com base nos mesmos dados de publicidade online de exemplo usados no exemplo para o Hive JSON SerDe. Na cláusula `LOCATION`, substitua *`myregion`* pelo identificador da região onde o Athena é executado.

```
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';
```

Exemplo: desserializar JSON aninhado

Você pode usar os JSON SerDes para analisar dados codificados por JSON mais complexos. Isso requer o uso de instruções `CREATE TABLE` que usem elementos `struct` e `array` para representar estruturas aninhadas.

O exemplo a seguir cria uma tabela do Athena com base nos dados JSON com estruturas aninhadas. Para analisar os dados codificados em JSON no Athena, certifique-se de que cada documento JSON esteja em sua própria linha, separado por uma nova linha.

Este exemplo pressupõe dados codificados em JSON que tenham a seguinte estrutura:

```
{
"DocId": "AWS",
"User": {
        "Id": 1234,
        "Username": "bob1234",
        "Name": "Bob",
"ShippingAddress": {
```

```
"Address1": "123 Main St.",
"Address2": null,
"City": "Seattle",
"State": "WA"
   },
"Orders": [
   {
     "ItemId": 6789,
     "OrderDate": "11/11/2017"
   },
   {
     "ItemId": 4352,
     "OrderDate": "12/12/2017"
   }
  ]
 }
}
```

A instrução `CREATE TABLE` a seguir usa o Openx-JsonSerDe com os tipos de dados de coleção `array` e `struct` para estabelecer grupos de objetos. Cada documento JSON é listado em sua própria linha, separado por uma nova linha. Para evitar erros, os dados que estão sendo consultados não incluem chaves duplicadas em `struct` ou nomes de chaves de mapa.

```
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/';
```

Recursos adicionais do

Para obter mais informações sobre como trabalhar com JSON e JSON aninhado no Athena, consulte os seguintes recursos:

- Create tables in Amazon Athena from nested JSON and mappings using JSONSerDe (Criar tabelas no Amazon Athena de mapeamentos e JSON aninhado usando JSONSerDe) (blog sobre big data da AWS)
- Recebo mensagens de erro ao tentar ler dados JSON no Amazon Athena (artigo do Centro de conhecimento da AWS)
- hive-json-schema (GitHub): ferramenta escrita em Java que gera instruções `CREATE TABLE` de documentos JSON de exemplo. As instruções `CREATE TABLE` geradas usam o OpenX JSON Serde.

LazySimpleSerDe para arquivos CSV, TSV e com delimitação personalizada

Especificar esse SerDe é opcional. Este é o SerDe para dados em formatos CSV, TSV e com delimitação personalizada que o Athena usa por padrão. Este SerDe será usado se você não especificar qualquer SerDe e especificar somente `ROW FORMAT DELIMITED`. Use esse SerDe caso os dados não tenham valores entre aspas.

Para documentação de referência sobre o LazySimpleSerDe, consulte a seção Hive SerDe do Guia do desenvolvedor do Apache Hive.

Nome da biblioteca

O nome da biblioteca de classes de LazySimpleSerDe é `org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe`. Para obter informações sobre a classe LazySimpleSerDe, consulte LazySimpleSerDe.java em GitHub.com.

Ignorar cabeçalhos

Para ignorar os cabeçalhos nos dados ao definir a tabela, você pode usar a propriedade de tabela `skip.header.line.count`, conforme mostrado no exemplo a seguir.

```
TBLPROPERTIES ("skip.header.line.count"="1")
```

Para ver exemplos, consulte as instruções `CREATE TABLE` em Consultar os logs de fluxo do Amazon VPC e Consultar os logs do Amazon CloudFront.

Exemplo de CSV

O exemplo a seguir mostra como usar `LazySimpleSerDe` para criar uma tabela no Athena com base em dados CSV. Para desserializar arquivos com delimitação personalizada usando esse SerDe, siga o padrão no exemplo, mas use a cláusula `FIELDS TERMINATED BY` para especificar um delimitador de caractere único diferente. LazySimpleSerDe não é compatível com delimitadores de vários caracteres.

> **Note**
>
> Substitua *myregion* em `s3://athena-examples-myregion/path/to/data/` pelo identificador da região onde o Athena é executado, por exemplo, `s3://athena-examples-us-west-1/path/to/data/`.

Use a instrução `CREATE TABLE` para criar uma tabela do Athena usando os dados subjacentes em CSV armazenados no 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/';
```

Execute a `MSCK REPAIR TABLE` para atualizar metadados da partição sempre que uma nova partição for adicionada a essa tabela:

```
MSCK REPAIR TABLE flight_delays_csv;
```

Consulte as 10 rotas principais atrasadas há mais de 1 hora:

```
SELECT origin, dest, count(*) as delays
FROM flight_delays_csv
WHERE depdelayminutes > 60
GROUP BY origin, dest
ORDER BY 3 DESC
LIMIT 10;
```

> **Note**
>
> Os dados da tabela de voos foram extraídos dos Voos fornecidos pelo Departamento de Transportes dos EUA, Bureau of Transportation Statistics (Agência de Estatísticas de Transportes). Saturação do original removida.

Exemplo de TSV

Para criar uma tabela do Athena com base em dados TSV armazenados no Amazon S3, use `ROW FORMAT DELIMITED` e especifique `\t` como delimitador do campo de tabulação, `\n` como separador de linhas e `\` como o caractere de escape. O trecho a seguir mostra essa sintaxe. Nenhum exemplo de dados de voo do TSV está disponível no local de `athena-examples`, mas, como na tabela CSV, você executaria `MSCK REPAIR TABLE` para atualizar os metadados da partição sempre que uma nova partição fosse adicionada.

```
...
```

```
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
ESCAPED BY '\\'
LINES TERMINATED BY '\n'
...
```

OpenCSVSerDe para processar CSV

Ao criar uma tabela do Athena de dados CSV, determine o SerDe a ser usado com base nos tipos de valores que os dados contêm:

- Se os valores nos dados estiverem entre aspas duplas ("), você poderá usar o OpenCSV SerDe para desserializar os valores no Athena. Se os valores nos dados não estiverem entre aspas duplas ("), você poderá omiti-los especificando qualquer SerDe. Neste caso, o Athena usa o `LazySimpleSerDe` padrão. Para ter mais informações, consulte LazySimpleSerDe para arquivos CSV, TSV e com delimitação personalizada.
- Se os dados tiverem valores UNIX numéricos de `TIMESTAMP` (por exemplo, `1579059880000`), use o OpenCSVSerDe. Se os dados usam o formato `java.sql.Timestamp`, use o LazySimpleSerDe.

CSV SerDe (OpenCSVSerDe)

O OpenCSV SerDe tem as seguintes características de dados de string:

- Usa aspas duplas (") como o caractere de aspas padrão e permite que você especifique separadores, aspas e caracteres de escape, como:

  ```
  WITH SERDEPROPERTIES ("separatorChar" = ",", "quoteChar" = "`", "escapeChar" = "\\" )
  ```

- Não é possível inserir um caractere de escape em `\t` ou `\n` diretamente. Para inserir um caractere de escape neles, use `"escapeChar" = "\\"`. Consulte o exemplo neste tópico.
- Não oferece suporte a quebras de linha incorporadas em arquivos CSV.

Para tipos de dados diferentes de `STRING`, o OpenCSVSerDe apresenta o seguinte comportamento:

- Reconhece os tipos de dados `BOOLEAN`, `BIGINT`, `INT` e `DOUBLE`.
- Não reconhece valores vazios ou nulos nas colunas definidas como um tipo de dados numérico, deixando-os como `string`. Uma solução alternativa é criar a coluna com os valores nulos como

`string` e usar `CAST` para converter o campo de uma consulta em um tipo de dados numérico, especificando um valor padrão de `0` para nulos. Para obter mais informações, consulte Quando consulto dados CSV no Athena, aparece o erro HIVE_BAD_DATA: erro ao analisar valor do campo (em inglês) na Central de Conhecimento da AWS.

- Para colunas especificadas com o tipo de dados `timestamp` na instrução `CREATE TABLE`, reconhece os dados de `TIMESTAMP` caso sejam especificados no formato numérico UNIX em milissegundos, como `1579059880000`.
  - O OpenCSVSerDe não permite `TIMESTAMP` no formato `java.sql.Timestamp` compatível com JDBC, como `"YYYY-MM-DD HH:MM:SS.fffffffff"` (precisão de 9 casas decimais).
- Para colunas especificadas com o tipo de dados `DATE` na instrução `CREATE TABLE`, reconhece os valores como datas se os valores representam o número de dias decorridos desde 1º de janeiro de 1970. Por exemplo, o valor `18276` em uma coluna com o tipo de dados `date` é processado como `2020-01-15` quando consultado. Nesse formato UNIX, é considerado que cada dia tenha 86.400 segundos.
  - O OpenCSVSerDe não aceita `DATE` em nenhum outro formato diretamente. Para processar os dados de carimbo de data/hora em outros formatos, você pode definir a coluna como `string` e usar as funções de conversão de tempo para retornar os resultados desejados em sua consulta `SELECT`. Para obter mais informações, consulte o artigo When I query a table in Amazon Athena, the TIMESTAMP result is empty (Quando consulto uma tabela no Amazon Athena, o resultado de TIMESTAMP está vazio) na Central de Conhecimento da AWS.
- Para converter mais colunas para o tipo desejado em uma tabela, crie uma visualização sobre a tabela e use `CAST` para converter para o tipo desejado.

Example Exemplo: usar o tipo TIMESTAMP e o tipo DATE especificados no formato numérico UNIX.

Considere as três colunas de dados separados por vírgula a seguir. Os valores em cada coluna são colocados entre aspas duplas.

```
"unixvalue creationdate 18276 creationdatetime 1579059880000","18276","1579059880000"
```

A instrução a seguir cria uma tabela no Athena com base no local do bucket do Amazon S3 especificado.

```
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'
```

Em seguida, execute a consulta a seguir:

```
SELECT * FROM testtimestamp1
```

A consulta retorna o seguinte resultado, exibindo os dados de data e hora:

```
profile_id                                                          creationdate
 creationdatetime
unixvalue creationdate 18276 creationdatetime 1579146280000        2020-01-15
 2020-01-15 03:44:40.000
```

Example Exemplo: como inserir caracteres de escape em **\t** ou **\n**

Considere os seguintes dados de teste:

```
" \\t\\t\\n 123 \\t\\t\\n ",abc
" 456 ",xyz
```

A instrução a seguir cria uma tabela no Athena que especifica `"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/'
```

Em seguida, execute a consulta a seguir:

```
SELECT * FROM test1;
```

Ele retorna esse resultado, inserindo corretamente caracteres de escape em `\t` ou `\n`:

```
f1            s2
\t\t\n 123 \t\t\n            abc
```

```
456                             xyz
```

Nome do SerDe

CSV SerDe

Nome da biblioteca

Para usar esse SerDe, especifique o nome totalmente qualificado da classe após `ROW FORMAT SERDE`. Especifique também os delimitadores dentro de `SERDEPROPERTIES`, da seguinte forma:

```
...
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
  "separatorChar" = ",",
  "quoteChar"     = "`",
  "escapeChar"    = "\\"
)
```

Ignorar cabeçalhos

Para ignorar os cabeçalhos nos dados ao definir a tabela, você pode usar a propriedade de tabela `skip.header.line.count`, conforme mostrado no exemplo a seguir.

```
TBLPROPERTIES ("skip.header.line.count"="1")
```

Para ver exemplos, consulte as instruções `CREATE TABLE` em Consultar os logs de fluxo do Amazon VPC e Consultar os logs do Amazon CloudFront.

Exemplo

Este exemplo pressupõe dados CSV salvos em `s3://DOC-EXAMPLE-BUCKET/mycsv/` com o seguinte conteúdo:

```
"a1","a2","a3","a4"
"1","2","abc","def"
"a","a1","abc3","ab4"
```

Use uma instrução `CREATE TABLE` para criar uma tabela do Athena com base nos dados. Referencie a classe do OpenCSVSerDe após `ROW FORMAT SERDE` e especifique o separador de

caractere, o caractere de aspas e o caractere de escape em `WITH SERDEPROPERTIES`, como no exemplo a seguir.

```
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/';
```

Consulte todos os valores na tabela:

```
SELECT * FROM myopencsvtable;
```

A consulta retorna os valores a seguir:

```
col1     col2    col3    col4
-----------------------------
a1       a2      a3      a4
1        2       abc     def
a        a1      abc3    ab4
```

ORC SerDe

Nome do SerDe

OrcSerDe

Nome da biblioteca

Esta biblioteca usa a classe OrcSerde.java do Apache Hive para dados no formato ORC. Ela passa o objeto do ORC para o leitor e do ORC para o gravador.

Exemplos

> **Note**
>
> Substitua *myregion* em `s3://athena-examples-myregion/path/to/data/` pelo identificador da região onde o Athena é executado, por exemplo, `s3://athena-examples-us-west-1/path/to/data/`.

O exemplo a seguir cria uma tabela para dados de voos em atraso no ORC. A tabela inclui partições:

```
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");
```

Execute a instrução `MSCK REPAIR TABLE` na tabela para atualizar metadados da partição:

```
MSCK REPAIR TABLE flight_delays_orc;
```

Use esta consulta para obter as 10 rotas principais atrasadas há mais de 1 hora:

```
SELECT origin, dest, count(*) as delays
FROM flight_delays_orc
WHERE depdelayminutes > 60
GROUP BY origin, dest
ORDER BY 3 DESC
LIMIT 10;
```

Parquet SerDe

Nome do SerDe

ParquetHiveSerDe é usado para dados armazenados no formato Parquet.

> **Note**
>
> Para converter dados em formato Parquet, você pode usar consultas CREATE TABLE AS SELECT (CTAS) . Para obter mais informações, consulte Criar uma tabela a partir de resultados de consultas (CTAS), Exemplos de consultas CTAS e Usar CTAS e INSERT INTO para ETL e análise de dados.

Nome da biblioteca

O Athena usa a seguinte classe quando precisa desserializar dados armazenados no Parquet: `org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe`

Exemplo: consultar um arquivo armazenado em Parquet

> **Note**
>
> Substitua *`myregion`* em `s3://athena-examples-`*`myregion`*`/path/to/data/` pelo identificador da região onde o Athena é executado, por exemplo, `s3://athena-examples-us-west-1/path/to/data/`.

Use a instrução `CREATE TABLE` abaixo para criar uma tabela do Athena com base nos dados subjacentes armazenados em formato Parquet no 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");
```

Execute a instrução `MSCK REPAIR TABLE` na tabela para atualizar metadados da partição:

```
MSCK REPAIR TABLE flight_delays_pq;
```

Consulte as 10 rotas principais atrasadas há mais de 1 hora:

```
SELECT origin, dest, count(*) as delays
FROM flight_delays_pq
WHERE depdelayminutes > 60
GROUP BY origin, dest
ORDER BY 3 DESC
```

```
LIMIT 10;
```

> **Note**
>
> Os dados da tabela de voos foram extraídos dos Voos fornecidos pelo Departamento de Transportes dos EUA, Bureau of Transportation Statistics (Agência de Estatísticas de Transportes). Saturação do original removida.

Ignorando estatísticas do Parquet

Quando você lê dados Parquet, pode receber mensagens de erro como as seguintes:

```
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.
```

Para contornar esse problema, use a instrução CREATE TABLE ou ALTER TABLE SET TBLPROPERTIES para definir a `parquet.ignore.statistics` propriedade Parquet SerDe como `true`, como nos exemplos a seguir.

Exemplo de CREATE TABLE

```
...
ROW FORMAT SERDE
'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
WITH SERDEPROPERTIES (
'parquet.ignore.statistics'='true')
STORED AS PARQUET
...
```

Exemplo de ALTER TABLE

```
ALTER TABLE ... SET TBLPROPERTIES ('parquet.ignore.statistics'='true')
```

Regex SerDe

O Regex SerDe usa uma expressão regular (regex) para desserializar dados extraindo grupos regex em colunas de tabela.

Se uma linha nos dados não corresponder à regex, todas as colunas na linha serão retornadas como `NULL`. Se uma linha corresponder à regex, mas tiver menos grupos do que o esperado, os grupos ausentes serão `NULL`. Se uma linha nos dados corresponder à regex, mas tiver mais colunas do que grupos na regex, as colunas adicionais serão ignoradas.

Para obter mais informações, consulte Classe RegexSerDe na documentação do Apache Hive.

Nome do SerDe

RegexSerDe

Nome da biblioteca

RegexSerDe

Exemplos

O exemplo a seguir cria uma tabela de logs do CloudFront usando o RegExSerDe. Substitua *myregion* em `s3://athena-examples-myregion/cloudfront/plaintext/` pelo identificador da região onde o Athena é executado (por exemplo, `s3://athena-examples-us-west-1/cloudfront/plaintext/`).

```
CREATE EXTERNAL TABLE IF NOT EXISTS cloudfront_logs (
  `Date` DATE,
  Time STRING,
  Location STRING,
  Bytes INT,
  RequestIP STRING,
  Method STRING,
  Host STRING,
  Uri STRING,
  Status INT,
  Referrer STRING,
  os STRING,
  Browser STRING,
  BrowserVersion STRING
```

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

Executar consultas SQL usando o Amazon Athena

É possível executar consultas SQL usando origens de dados no Amazon Athena registradas com o AWS Glue Data Catalog e origens de dados, como metastores do Hive e instâncias do Amazon DocumentDB, às quais você se conecta pelo recurso de consulta federada do Athena. Para obter mais informações sobre como trabalhar com fontes de dados, consulte Conectar a origens de dados. Quando você executa uma consulta Data Definition Language (DDL – Linguagem de definição de dados) que modifica o esquema, o Athena grava os metadados no metastore associado à origem dos dados. Algumas consultas, como `CREATE TABLE AS` e `INSERT INTO`, também podem gravar registros no conjunto de dados, por exemplo, adicionando um registro CSV a um local do Amazon S3. Quando você executa uma consulta, o Athena salva os resultados dela no local que você especificou. Isso permite visualizar o histórico de consultas e fazer download e exibir conjuntos de resultados de consulta.

Esta seção apresenta orientações para executar consultas do Athena em origens e tipos de dados comuns usando uma variedade de instruções SQL. Há também orientações gerais de como trabalhar com estruturas e operadores comuns, por exemplo, arrays, concatenação, filtragem, nivelamento e classificação. Outros exemplos incluem consultas de dados em tabelas com estruturas aninhadas e mapas, tabelas baseadas em conjuntos de dados codificados com JSON e conjuntos de dados associados aos Serviços da AWS, como logs do AWS CloudTrail e do Amazon EMR. A cobertura completa do uso do SQL padrão está fora do escopo desta documentação. Para obter mais informações sobre SQL, consulte as referências das linguagens Trino e Presto.

Tópicos

- Visualizar planos de execução para consultas SQL
- Trabalhar com resultados de consultas, consultas recentes e arquivos de saída
- Reutilização dos resultados da consulta
- Visualizar estatísticas e detalhes de execução para consultas concluídas

- Trabalhar com visualizações
- Usar consultas salvas
- Utilizar consultas parametrizadas
- Usar o otimizador baseado em custos
- Consulta de dados do S3 Express One Zone
- Consultar objetos restaurados do Amazon S3 Glacier
- Tratamento de atualizações do esquema
- Consultar matrizes
- Consultar dados geoespaciais
- Consultar JSON
- Usar Machine Learning (ML) com o Amazon Athena
- Executar consultas com funções definidas pelo usuário
- Realizar consultas entre regiões
- Consulta no AWS Glue Data Catalog
- Consulta de logs do AWS service (Serviço da AWS)
- Consultar logs do servidor Web armazenados no Amazon S3

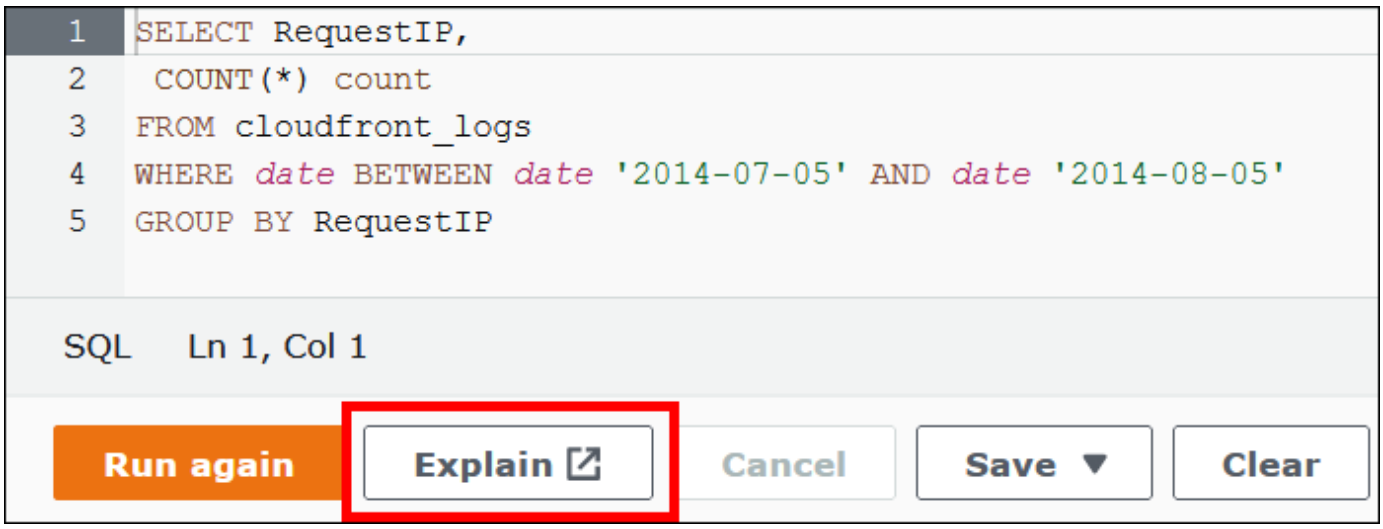
Para ver as considerações e as limitações, consulte Considerações e limitações das consultas SQL no Amazon Athena.

Visualizar planos de execução para consultas SQL

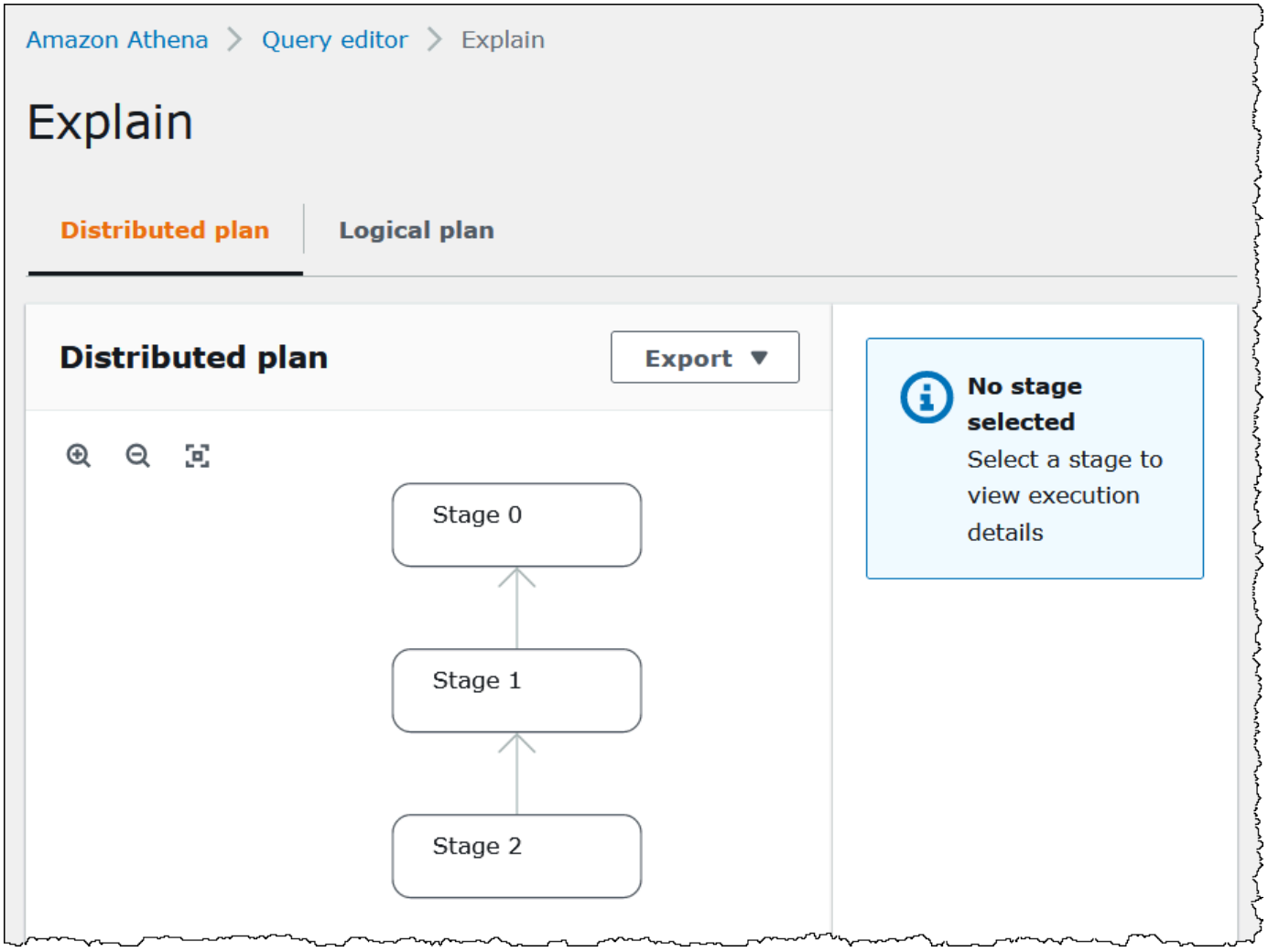
É possível utilizar o editor de consultas do Athena para ver representações gráficas de como uma consulta será executada. Quando você insere uma consulta no editor e escolhe a opção Explain (Explicar), o Athena usa uma Instrução SQL EXPLICAM nessa consulta para criar dois gráficos correspondentes: um plano de execução distribuído e um plano de execução lógico. Esses gráficos podem ser utilizados para analisar, solucionar problemas e melhorar a eficiência das suas consultas.

Para visualizar planos de execução de uma consulta

1. Insira sua consulta no editor de consultas do Athena e escolha Explain (Explicar).



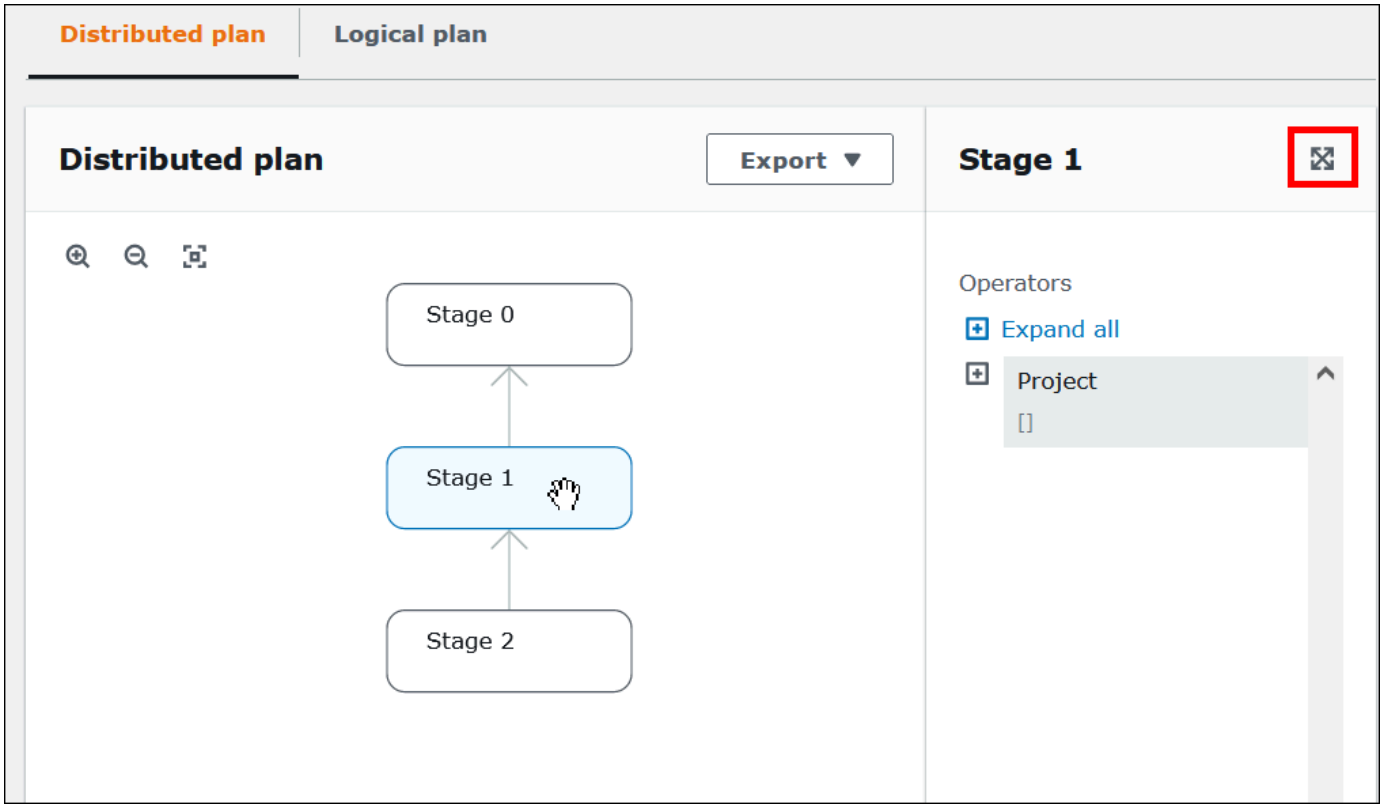
A guia Distributed plan (Plano distribuído) mostra o plano de execução da consulta em um ambiente distribuído. Um plano distribuído tem fragmentos ou estágios de processamento. Cada estágio tem um número de índice baseado em zero e é processado por um ou mais nós. Dados podem ser trocados entre nós.



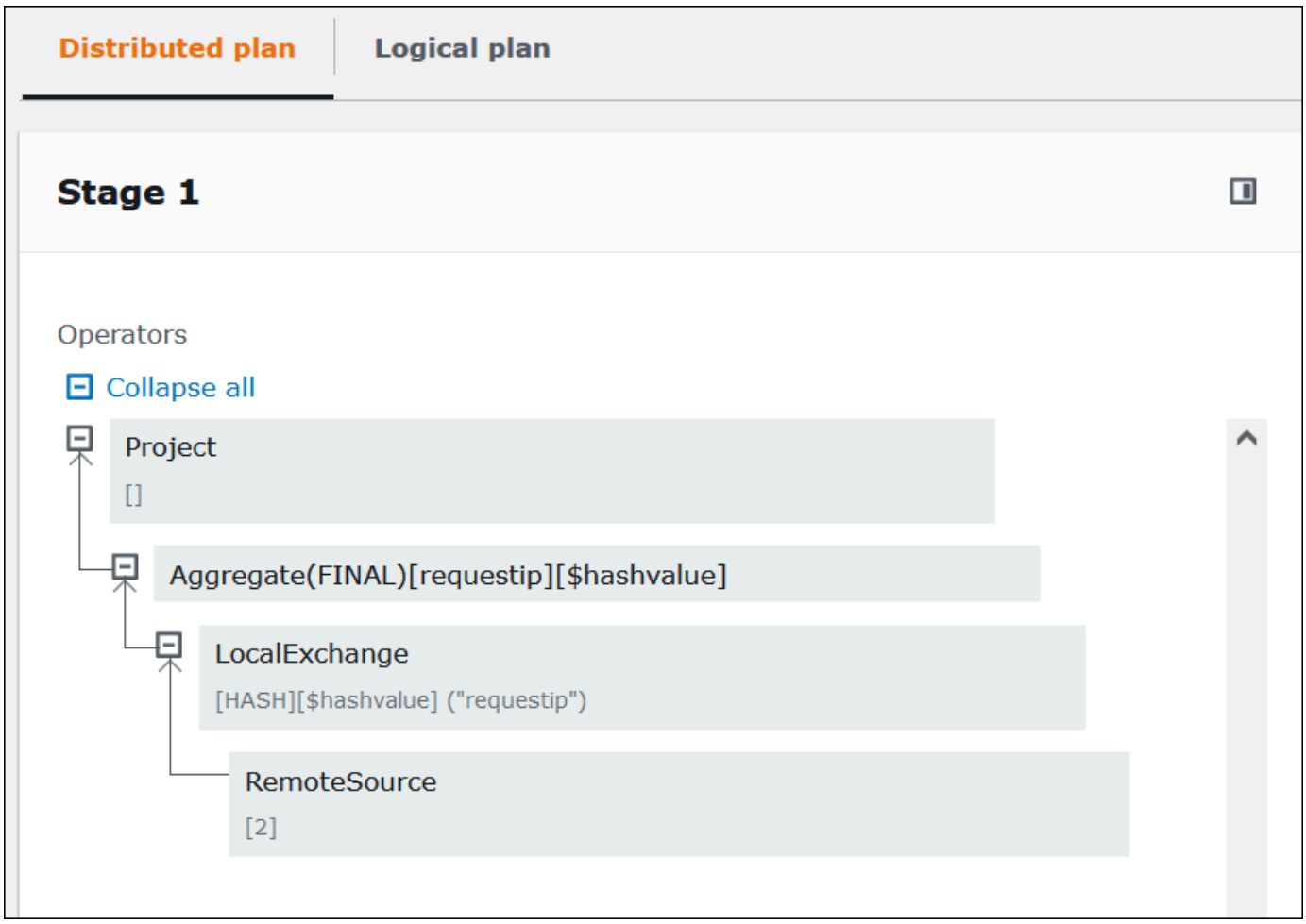
2. Para navegar pelo gráfico, utilize as seguintes opções:

   - Para aumentar ou diminuir o zoom, role com o mouse ou use os ícones de ampliação.
   - Para ajustar o gráfico para que ele caiba na tela, selecione o ícone Zoom to fit (Ampliar para caber).
   - Para mover o gráfico, arraste o ponteiro do mouse.

3. Para ver detalhes de um estágio, escolha o estágio.

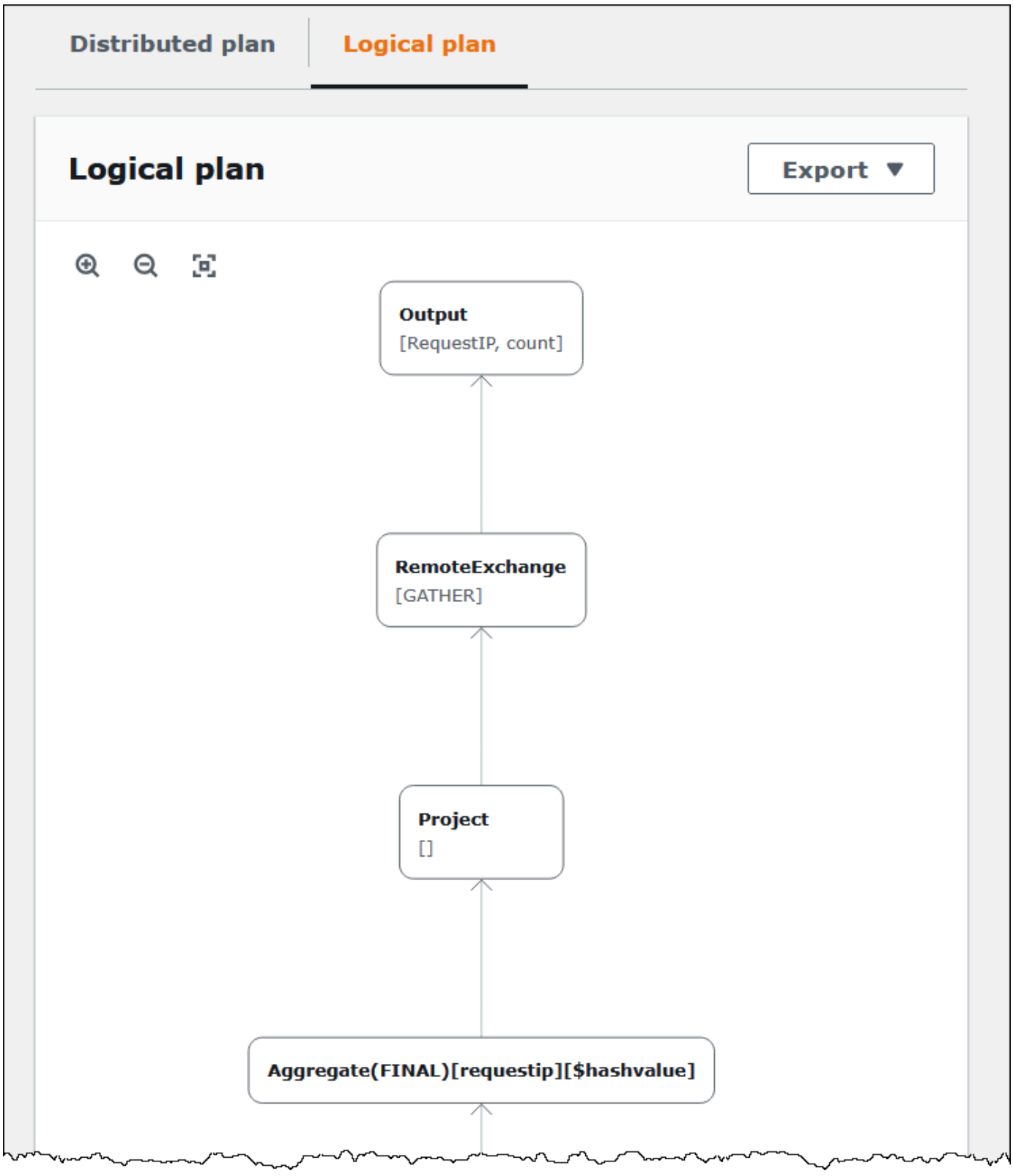


4. Para ver os detalhes do estágio em largura natural, selecione o ícone de expansão na parte superior direita do painel de detalhes.
5. Para ver mais detalhes, expanda um ou mais itens na árvore do operador. Para obter informações sobre fragmentos de planos distribuídos, consulte Tipos de saída da instrução EXPLAIN.



> **Important**
>
> Alguns filtros de partição podem não estar visíveis no gráfico de árvore do operador aninhado, mesmo que o Athena os aplique à sua consulta. Para verificar o efeito desses filtros, execute EXPLAIN ou EXPLAIN ANALYZE na sua consulta e visualize os resultados.

6. Escolha a guia Logical plan (Plano lógico). O gráfico mostra o plano lógico para executar a consulta. Para obter mais informações termos operacionais, consulte Noções básicas sobre os resultados da instrução EXPLAIN do Athena.



7. Para exportar um plano como uma imagem SVG ou PNG, ou como texto JSON, escolha Export (Exportar).

Recursos adicionais do

Para obter mais informações, consulte os recursos a seguir.

Usar EXPLAIN e EXPLAIN ANALYZE no Athena

Noções básicas sobre os resultados da instrução EXPLAIN do Athena

Visualizar estatísticas e detalhes de execução para consultas concluídas

Trabalhar com resultados de consultas, consultas recentes e arquivos de saída

O Amazon Athena armazena automaticamente os resultados das consultas e as informações de metadados de cada consulta executada em um local de resultados de consultas que você pode especificar no Amazon S3. Se necessário, você pode acessar os arquivos nesse local para trabalhar com eles. Também é possível baixar os arquivos de resultados das consultas diretamente do console do Athena.

Para configurar um local de resultados de consultas do Amazon S3 pela primeira vez, veja Especificar um local para resultados de consultas usando o console do Athena.

Os arquivos de saída são salvos automaticamente para cada consulta executada. Para acessar e visualizar arquivos de saída de consultas usando o console do Athena, as entidades principais do IAM (usuários e funções) precisam de permissão para a ação GetObject do Amazon S3 no local de resultados de consultas, além da permissão para a ação GetQueryResults do Athena. O local de resultados da consulta pode ser criptografado. Se o local estiver criptografado, os usuários deverão ter as permissões de chave apropriadas para criptografar e descriptografar o local de resultados da consulta.

> **Important**
>
> Os principais do IAM com permissão para a ação `GetObject` do Amazon S3 no local de resultados de consultas podem recuperar os resultados das consultas do Amazon S3 mesmo que a permissão para a ação `GetQueryResults` do Athena seja negada.

Especificar um local para resultados de consultas

O local de resultados de consultas usado pelo Athena é determinado por uma combinação de configurações de grupo de trabalho e do lado do cliente. As configurações do lado do cliente são baseadas na forma como você executa a consulta.

- Se você executar a consulta usando o console do Athena, o Query result location (Local de resultados da consulta) inserido em Settings (Configurações) na barra de navegação determinará a configuração do lado do cliente.
- Se você executar a consulta usando a API do Athena, o parâmetro `OutputLocation` da ação StartQueryExecution determinará a configuração do lado do cliente.
- Se você usar os drivers ODBC ou JDBC para executar consultas, a propriedade `S3OutputLocation` especificada no URL de conexão determinará a configuração no lado do cliente.

> **Important**
>
> Quando você executa uma consulta usando a API ou usando o driver ODBC ou JDBC, a configuração do console não se aplica.

Cada configuração de grupo de trabalho tem uma opção Override client-side settings (Substituir configurações no lado do cliente) que pode ser habilitada. Quando essa opção está habilitada, as configurações do grupo de trabalho têm precedência sobre as configurações aplicáveis do lado do cliente quando uma entidade principal do IAM associada a esse grupo de trabalho executa a consulta.

Especificar um local para resultados de consultas usando o console do Athena

Antes de executar uma consulta, um local de bucket de resultados de consultas do Amazon S3 precisa ser especificado, ou você deve usar um grupo de trabalho que especificou um bucket e com uma configuração que substitui as configurações do cliente.

Para especificar um local de resultados de consultas na configuração do lado do cliente usando o console do Athena

1. Alterne para o grupo de trabalho para o qual você deseja especificar um local de resultados de consultas. O nome do grupo de trabalho padrão é primary.

2. No painel de navegação, escolha Settings (Configurações).
3. Na barra de navegação, escolha Manage (Gerenciar).
4. Em Manage settings (Gerenciar configurações), faça um dos seguintes procedimentos:

   - Na caixa de texto Query result location (Localização dos resultados da consulta), insira o caminho para o bucket criado no Amazon S3 para resultados de consultas. Adicione o prefixo `s3://` ao caminho.
   - Escolha Browse S3 (Navegar no S3), escolha o bucket do Amazon S3 que você criou na região atual e escolha Choose (Escolher).

   > **Note**
   >
   > Se você estiver usando um grupo de trabalho que especifica um local para os resultados das consultas para todos os usuários do grupo de trabalho, a opção para alterar o local dos resultados das consultas não estará disponível.

5. (Opcional) Escolha View lifecycle configuration (Exibir configuração do ciclo de vida) para visualizar e configurar as regras de ciclo de vida do Amazon S3 em seu bucket de resultados de consulta. As regras de ciclo de vida do Amazon S3 que você cria podem ser regras de expiração ou regras de transição. As regras de expiração excluem os resultados de consultas automaticamente após determinado tempo. As regras de transição os transferem para outro nível de armazenamento do Amazon S3. Para obter mais informações, consulte Definir configuração do ciclo de vida em bucket no Guia do usuário do Amazon Simple Storage Service.
6. (Opcional) Para Expected bucket owner (Proprietário esperado do bucket), insira o ID da Conta da AWS que você espera ser a proprietária do bucket do local de saída. Essa é uma medida de segurança adicional. Se o ID da conta do proprietário do bucket não corresponder ao ID especificado aqui, as tentativas de saída para o bucket falharão. Para obter informações detalhadas, consulte Verificar propriedade do bucket com a condição de proprietário do bucket no Guia do usuário do Amazon S3.

   > **Note**
   >
   > A configuração esperada do proprietário do bucket se aplica somente ao local de saída do Amazon S3 que você especificar para os resultados da consulta do Athena. Ela não se aplica a outros locais do Amazon S3, como locais de origem dos dados em buckets externos do Amazon S3, locais da tabela de destino `CTAS` e `INSERT INTO`, locais de

saída da instrução `UNLOAD`, operações para buckets de vazamento para consultas federadas ou consultas `SELECT` executadas em uma tabela em outra conta.

7. (Opcional) Escolha Encrypt query results (Criptografar resultados da consulta) para criptografar os resultados das consultas que estão armazenados no Amazon S3. Para obter mais informações sobre criptografia no Athena, consulte Criptografia inativa.
8. (Opcional) Escolha Assign bucket owner full control over query results (Atribuir controle total ao proprietário do bucket sobre os resultados das consultas) para que proprietário do bucket tenha acesso de controle total sobre os resultados das consultas quando ACLs estiverem habilitadas para o bucket de resultados de consultas. Por exemplo, se a localização dos resultados das consultas pertencer a outra conta, será possível conceder propriedade e controle total sobre os resultados das consultas à outra conta. Para obter mais informações, consulte Controlar a propriedade de objetos e desabilitar ACLs para seu bucket, no Guia do Usuário do Amazon S3.
9. Escolha Salvar.

Locais padrão já criados

Anteriormente no Athena, se você executasse uma consulta sem especificar um valor em Query result location (Local de resultados de consultas) e a configuração de local de resultados de consultas não fosse substituída por um grupo de trabalho, o Athena criava um local padrão para você. O local padrão era `aws-athena-query-results-`*`MyAcctID`*`-`*`MyRegion`*, em que *`MyAcctID`* era o ID da conta da Amazon Web Services do principal do IAM que executava a consulta, e *`MyRegion`* era a região em que a consulta era executada (por exemplo, `us-west-1`).

Agora, antes de executar uma consulta do Athena em uma região onde sua conta nunca acessou o Athena, você deve especificar um local de resultados de consultas ou usar um grupo de trabalho que substitua a configuração desse local. O Athena não cria mais um local padrão de resultados de consultas para você, mas os locais padrão `aws-athena-query-results-`*`MyAcctID`*`-`*`MyRegion`* que já foram criados permanecem válidos e você pode continuar a usá-los.

Especificar um local para resultados de consultas usando um grupo de trabalho

Especifique o local de resultados de consultas em uma configuração de grupo de trabalho usando o AWS Management Console, a AWS CLI ou a API do Athena.

Ao usar a AWS CLI, especifique a localização do resultado da consulta usando o parâmetro `OutputLocation` da opção `--configuration` ao executar o comando `aws athena create-work-group` ou `aws athena update-work-group`.

Para especificar o local de resultados de consultas para um grupo de trabalho usando o console do Athena

1. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.

2. No painel de navegação, escolha Global networks (Redes globais).
3. Na lista de grupos de trabalho, escolha o link do grupo de trabalho que você deseja editar.
4. Selecione a opção Editar.
5. Em Query result location and encryption (Local do resultado da consulta e criptografia), siga um destes procedimentos:

   - Na caixa de texto Query result location (Local dos resultados de consultas), insira o caminho para o bucket criado no Amazon S3 para resultados de consultas. Adicione o prefixo `s3://` ao caminho.
   - Escolha Browse S3 (Navegar no S3), escolha o bucket do Amazon S3 que você criou na região que desejar usar e escolha Choose (Escolher).

6. (Opcional) Para Expected bucket owner (Proprietário esperado do bucket), insira o ID da Conta da AWS que você espera ser a proprietária do bucket do local de saída. Essa é uma medida de segurança adicional. Se o ID da conta do proprietário do bucket não corresponder ao ID especificado aqui, as tentativas de saída para o bucket falharão. Para obter informações detalhadas, consulte Verificar propriedade do bucket com a condição de proprietário do bucket no Guia do usuário do Amazon S3.

> **Note**
>
> A configuração esperada do proprietário do bucket se aplica somente ao local de saída do Amazon S3 que você especificar para os resultados da consulta do Athena. Ela não se aplica a outros locais do Amazon S3, como locais de origem dos dados em buckets externos do Amazon S3, locais da tabela de destino `CTAS` e `INSERT INTO`, locais de saída da instrução `UNLOAD`, operações para buckets de vazamento para consultas federadas ou consultas `SELECT` executadas em uma tabela em outra conta.

7. (Opcional) Escolha Encrypt query results (Criptografar resultados da consulta) para criptografar os resultados das consultas que estão armazenados no Amazon S3. Para obter mais informações sobre criptografia no Athena, consulte Criptografia inativa.
8. (Opcional) Escolha Assign bucket owner full control over query results (Atribuir controle total ao proprietário do bucket sobre os resultados das consultas) para que proprietário do bucket tenha acesso de controle total sobre os resultados das consultas quando ACLs estiverem habilitadas para o bucket de resultados de consultas. Por exemplo, se a localização dos resultados das consultas pertencer a outra conta, será possível conceder propriedade e controle total sobre os resultados das consultas à outra conta.

   Se a configuração para S3 Object Ownership (Propriedade de objetos do S3) for Bucket owner preferred (Proprietário do bucket preferencial), o proprietário do bucket também possuirá todos os objetos de resultados de consultas gravados a partir deste grupo de trabalho. Por exemplo, quando o grupo de trabalho de uma conta externa habilita essa opção e define a localização dos resultados das consultas como o bucket do Amazon S3 da sua conta, cuja configuração S3 Object Ownership (Propriedade de objetos do S3) é Bucket owner preferred (Proprietário do bucket preferencial), você é o proprietário e tem acesso de controle total sobre os resultados da consulta do grupo de trabalho externo.

   A seleção dessa opção com a configuração configuração S3 Object Ownership (Propriedade de objetos do S3) definida como Bucket owner enforced (Proprietário do bucket imposto) não surte efeito. Para obter mais informações, consulte Controlar a propriedade de objetos e desabilitar ACLs para seu bucket, no Guia do Usuário do Amazon S3.
9. Se quiser exigir que todos os usuários do grupo de trabalho usem o local de resultados de consultas especificado, role para baixo até a seção Settings (Configurações) e selecione Override client-side settings (Substituir configurações no lado do cliente).
10. Escolha Salvar alterações.

Baixar os arquivos de resultados de consultas pelo console do Athena

É possível baixar o arquivo CSV dos resultados da consulta no painel logo após executar uma consulta. Também é possível baixar os resultados da consulta recente na guia Recent queries (Consultas recentes).

> **Note**
>
> Os arquivos de resultados das consultas do Athena são arquivos de dados com informações que podem ser configuradas por usuários específicos. Alguns programas que leem e analisam esses dados podem interpretar alguns deles como comandos (injeção de CSV). Por esse motivo, quando você importa dados CSV de resultados de consultas para um programa de planilha, esse programa pode avisá-lo sobre problemas de segurança. Para manter seu sistema seguro, você deve sempre escolher a opção para desabilitar links ou macros dos resultados de consulta baixados.

Para executar uma consulta e baixar os resultados

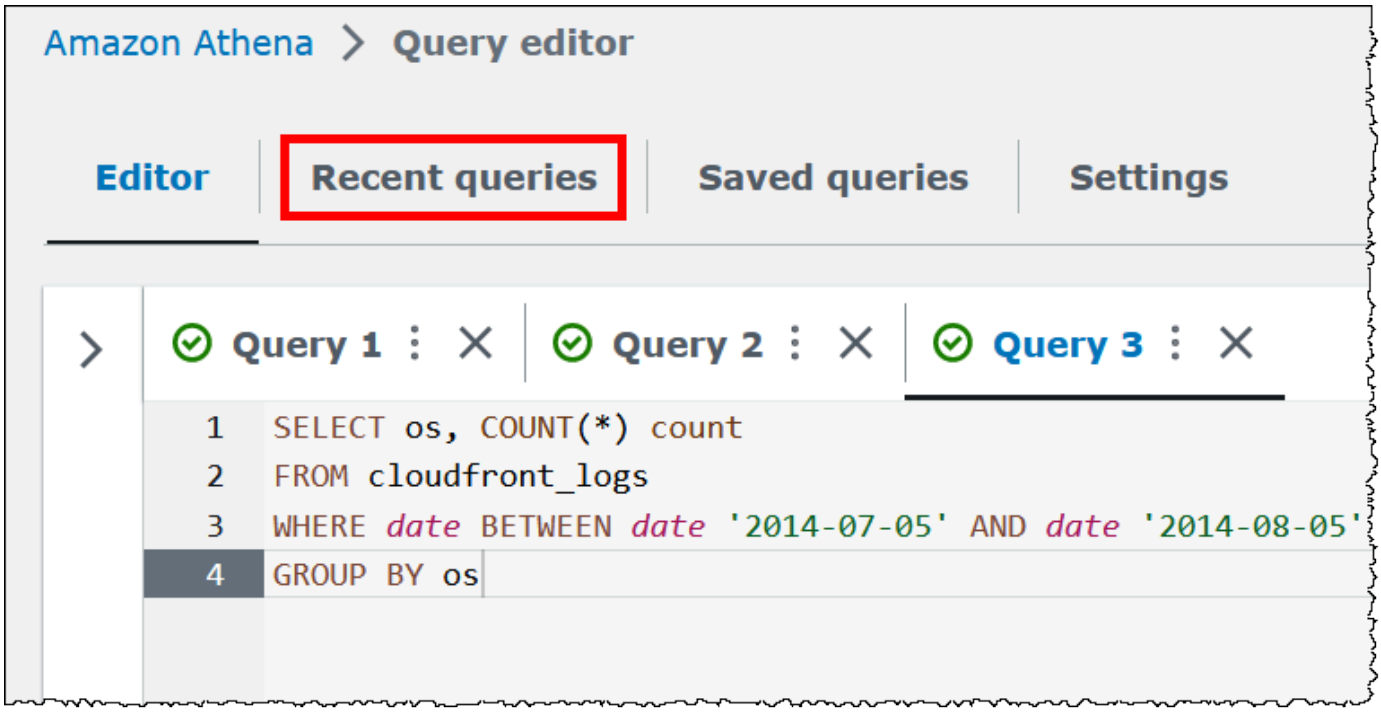
1. Insira sua consulta no editor de consultas e escolha Run (Executar).

   Quando a execução da consulta terminar, o painel Results (Resultados) mostrará os resultados da consulta.

2. Para baixar um arquivo CSV dos resultados da consulta, escolha Download results (Baixar resultados) acima do painel de resultados de consultas. Dependendo da configuração do navegador e do navegador, pode ser necessário confirmar o download.

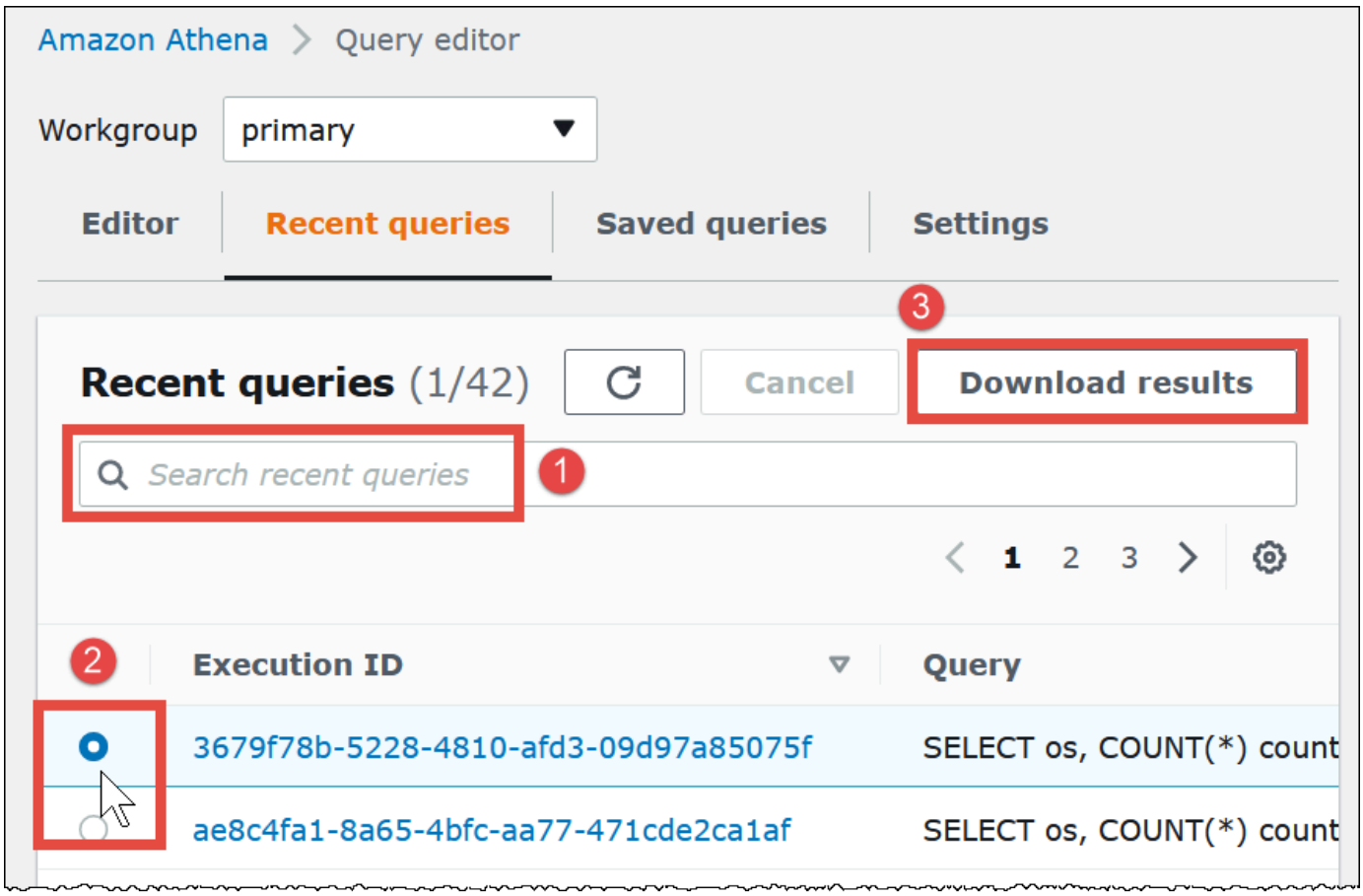
Como fazer download de um arquivo de resultados de uma consulta anterior

1. Escolha Recent querries (Consultas recentes).



2. Use a caixa de pesquisa para localizar a consulta, escolha a consulta e escolha Download results (Baixar resultados).

> **Note**
>
> Não é possível usar a opção Download results (Baixar resultados) para recuperar resultados de consultas que foram excluídos manualmente ou foram excluídos ou movidos para outro local pelas regras de ciclo de vida do Amazon S3.

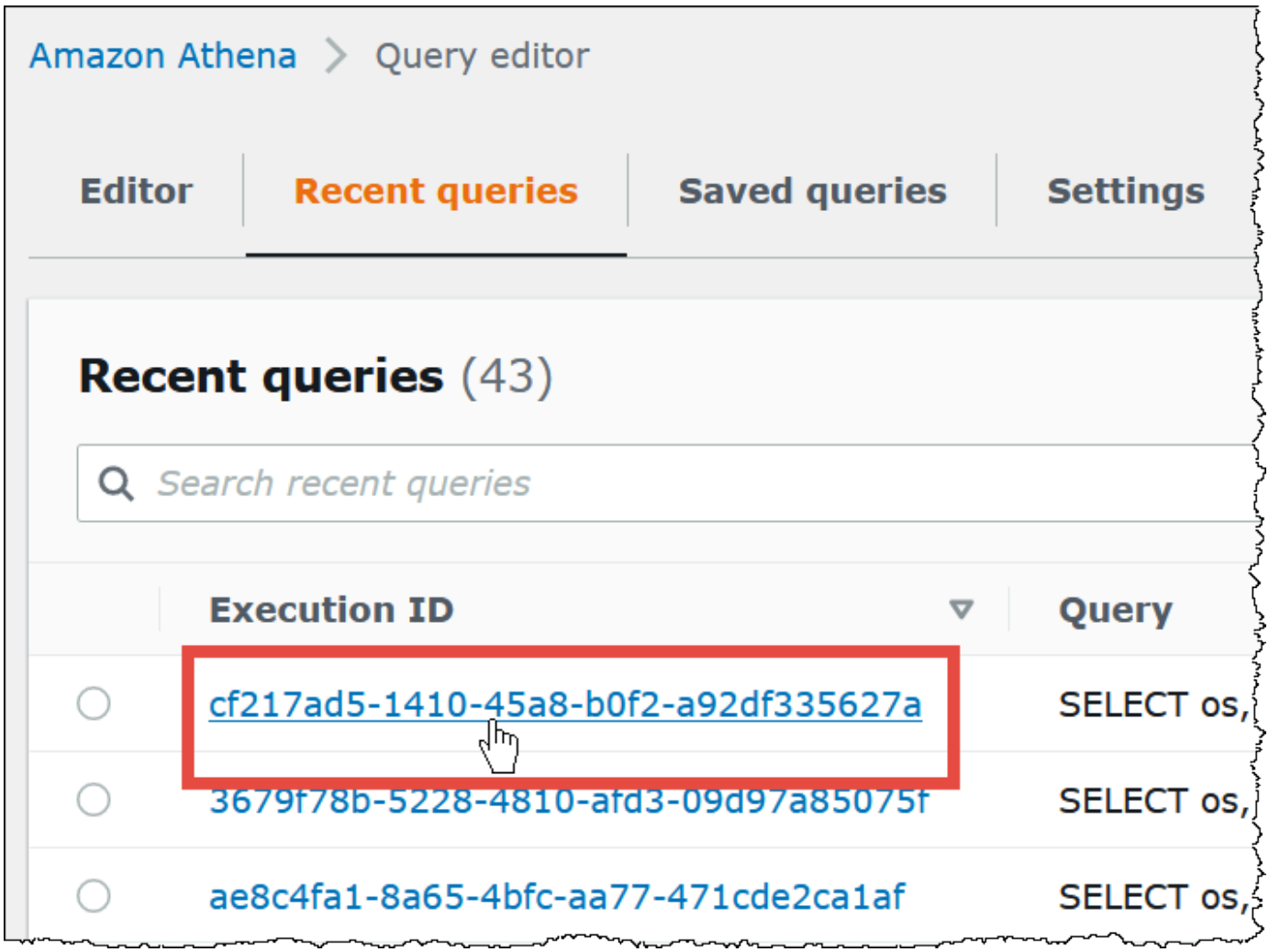


Visualizar consultas recentes

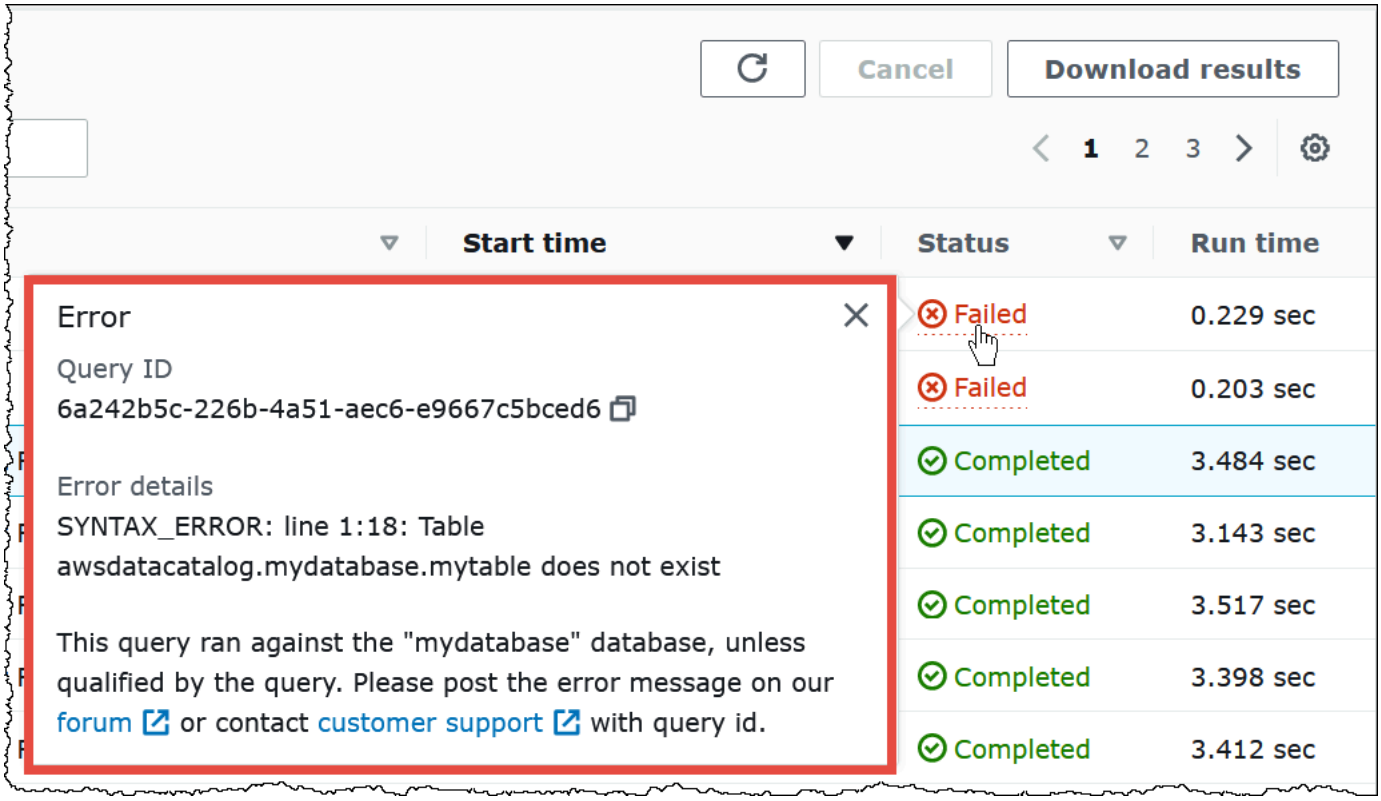
Você pode usar o console do Athena para ver quais consultas foram bem-sucedidas ou falharam, e visualizar os detalhes dos erros para as consultas que falharam. O Athena mantém o histórico de consultas por 45 dias.

Para visualizar consultas recentes no console do Athena

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Escolha Recent queries (Consultas recentes). A guia Recent queries (Consultas recentes) mostra informações sobre cada consulta executada.
3. Para abrir uma instrução de consulta no editor de consultas, escolha o ID de execução da consulta.



4. Para ver os detalhes de uma consulta que não foi bem-sucedida, escolha o link Failed (Com falha) para a consulta.

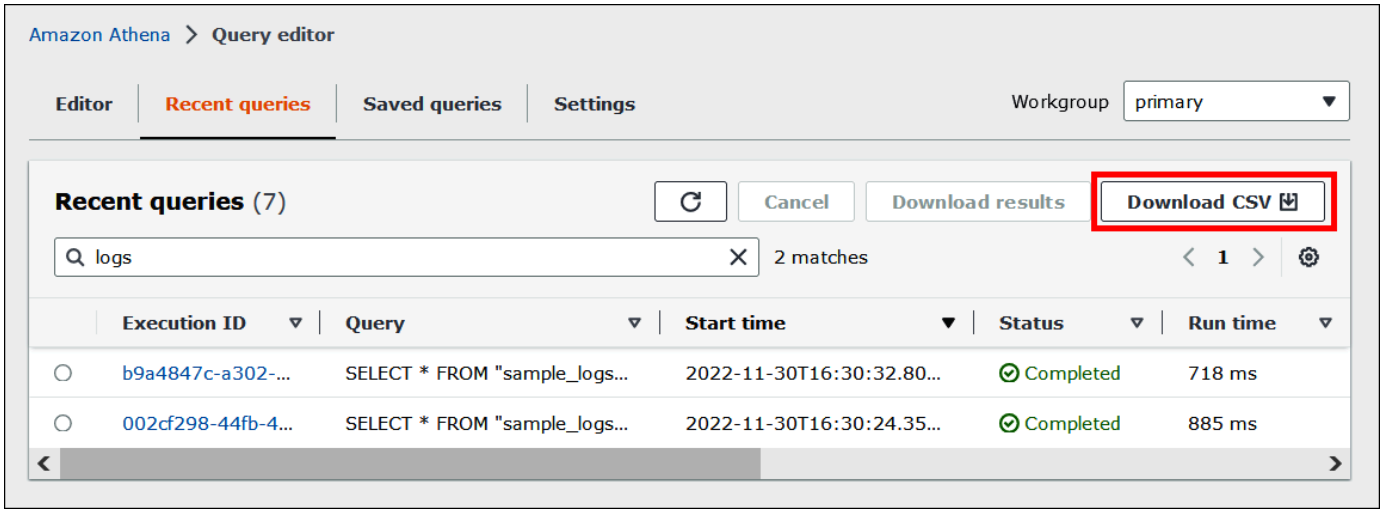


Como baixar diversas consultas recentes para um arquivo CSV

É possível usar a guia Recent queries (Consultas recentes) do console do Athena para exportar uma ou mais consultas recentes para um arquivo CSV com a finalidade de visualizá-las em formato tabular. O arquivo baixado não contém os resultados da consulta, mas a própria string de consulta SQL e outras informações sobre a consulta. Os campos exportados incluem o ID de execução, o conteúdo da string de consulta, o horário de início da consulta, o status, o tempo de execução, a quantidade de dados verificados, a versão usada do mecanismo de consulta e o método de criptografia. É possível exportar, no máximo, 500 consultas recentes ou, no máximo, 500 consultas filtradas usando critérios inseridos na caixa de pesquisa.

Para exportar uma ou mais consultas recentes para um arquivo CSV

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Escolha Recent queries (Consultas recentes).
3. (Opcional) Use a caixa de pesquisa para filtrar as consultas recentes que você deseja baixar
4. Escolha Baixar CSV.



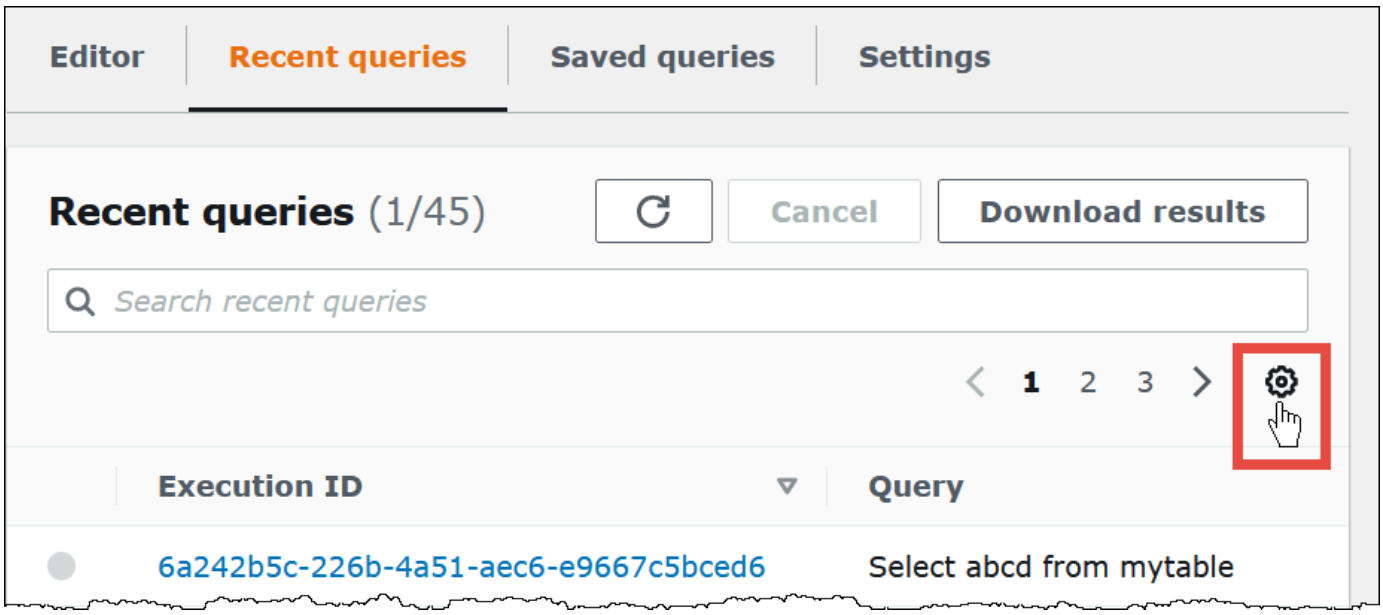
5. Na solicitação de salvamento do arquivo, escolha Save (Salvar). O nome de arquivo padrão é `Recent Queries` seguido por um carimbo de data/hora (por exemplo, `Recent Queries 2022-12-05T16 04 27.352-08 00.csv`).

Como configurar as opções de exibição das consultas recentes

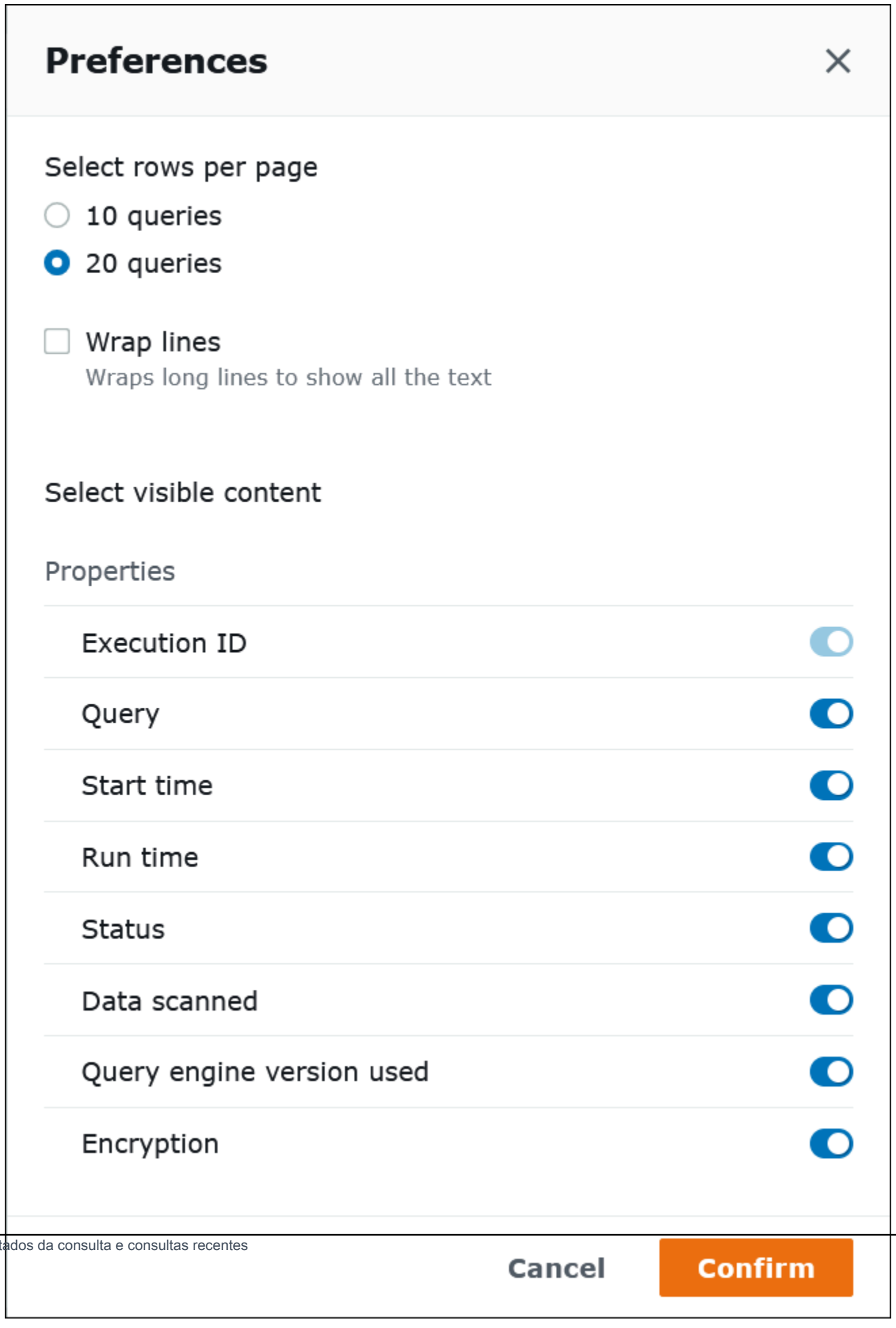
É possível configurar opções para a guia Recent queries (Consultas recentes), como colunas a serem exibidas e quebra de texto.

Para configurar as opções da guia Recent queries (Consultas recentes)

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Escolha Recent queries (Consultas recentes).
3. Escolha o botão de opções (ícone de engrenagem).



4. Na caixa de diálogo Preferences (Preferências), escolha o número de linhas por página, o comportamento de quebra de linha e as colunas a serem exibidas.


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. Selecione a opção Confirmar.

Manter o histórico de consultas por mais de 45 dias

Se você deseja manter o histórico de consultas por mais de 45 dias, é possível recuperá-lo e salvá-lo em um armazenamento de dados, como o Amazon S3. Para automatizar esse processo, é possível usar as ações da API do Athena e do Amazon S3 e os comandos da CLI. O procedimento a seguir resume essas etapas.

Como recuperar e salvar o histórico de consultas programaticamente

1. Use a ação da API ListQueryExecutions do Athena ou o comando da CLI list-query-executions para recuperar os IDs das consultas.
2. Use a ação da API GetQueryExecution do Athena ou o comando da CLI get-query-execution para recuperar as informações sobre cada consulta com base no ID.
3. Use a ação da API PutObject do Amazon S3 ou o comando da CLI put-object para salvar as informações no Amazon S3.

Localizar os arquivos de saída das consultas no Amazon S3

Os arquivos de saída das consultas são armazenados em subpastas no Amazon S3 no padrão de caminho a seguir, a menos que a consulta seja feita em um grupo de trabalho com uma configuração que substitua as configurações no lado do cliente. Quando a configuração do grupo de trabalho substitui as configurações no lado do cliente, a consulta usa o caminho de resultados especificado pelo grupo de trabalho.

```
QueryResultsLocationInS3/[QueryName|Unsaved/yyyy/mm/dd/]
```

- *QueryResultsLocationInS3* é o local de resultados da consulta especificado pelas configurações do grupo de trabalho ou pelo lado do cliente. Para obter mais informações, consulte the section called “Especificar um local para resultados de consultas” adiante neste documento.
- As seguintes subpastas são criadas somente para consultas executadas no console cujo caminho de resultados ainda não foi substituído pela configuração do grupo de trabalho. As consultas executadas pela AWS CLI ou usando a API do Athena são salvas diretamente em *QueryResultsLocationInS3*.
  - *Queryname* é o nome da consulta para a qual os resultados são salvos. Se a consulta foi executada, mas não salva, `Unsaved` será usado.

- *aaaa/mm/dd* é a data em que a consulta foi executada.

Os arquivos associados a uma consulta `CREATE TABLE AS SELECT` são armazenados em uma subpasta `tables` do padrão acima.

Identificar os arquivos de saída das consultas

Os arquivos são salvos no local de resultados de consultas no Amazon S3 com base no nome, no ID e na data de execução da consulta. Os arquivos de cada consulta são nomeados usando *QueryID*, que é um identificador exclusivo que o Athena atribui a cada consulta quando ela é executada.

Os seguintes tipos de arquivo são salvos:

Tipo de arquivo	Padrão de nomenclatura de arquivos	Descrição
Arquivos de resultados da consulta	*QueryID*`.csv` *QueryID*`.txt`	Os arquivos de resultados da consulta DML são salvos no formato CSV (valores separados por vírgulas). Os resultados da consulta DDL são salvos como arquivos de texto sem formatação. Você pode baixar os arquivos de resultados do console do painel Results (Resultados) ao usar o console ou do History (Histórico) da consulta. Para ter mais informações, consulte Baixar os arquivos de resultados de consultas pelo console do Athena.
Arquivos de metadados da consulta	*QueryID*`.csv.metadata`	Os arquivos de metadados de consulta DML e DDL são

Tipo de arquivo	Padrão de nomenclatura de arquivos	Descrição
	QueryID.txt.metadata	salvos no formato binário e não são legíveis por humanos. A extensão do arquivo corresponde ao arquivo relacionado de resultados de consultas. O Athena usa os metadados ao ler os resultados da consulta usando a ação GetQueryResults . Embora esses arquivos possam ser excluídos, não recomendamos porque informações importantes sobre a consulta são perdidas.
Arquivos manifesto de dados	*QueryID*-manifest.csv	Os arquivos manifesto de dados são gerados para monitorar os arquivos que o Athena cria em locais de origens de dados do Amazon S3 quando uma consulta INSERT INTO é executada . Se uma consulta falhar, o manifesto também rastreará os arquivos que a consulta pretendia gravar. O manifesto é útil para identificar arquivos órfãos resultantes de uma consulta com falha.

Usar a AWS CLI para identificar o local e os arquivos de saída das consultas

Para usar a AWS CLI para identificar o local de saída e os arquivos de resultados das consultas, execute o comando `aws athena get-query-execution` conforme o exemplo a seguir. Substitua *abc1234d-5efg-67hi-jklm-89n0op12qr34* pelo ID da consulta.

```
aws athena get-query-execution --query-execution-id abc1234d-5efg-67hi-
jklm-89n0op12qr34
```

Esse comando retorna uma saída semelhante à seguinte. Para ver as descrições de cada parâmetro de saída, consulte get-query-execution na Referência de comandos da AWS CLI.

```
{
    "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"
    }
}
```

Reutilização dos resultados da consulta

Ao executar novamente uma consulta no Athena, é possível optar por reutilizar o último resultado de consulta armazenado, se desejar. Essa opção pode aumentar a performance e reduzir os custos, em termos de números, para os bytes verificados. A reutilização dos resultados da consulta é útil se, por exemplo, você souber que os resultados não serão alterados em um determinado período de tempo. É possível especificar um período máximo para a reutilização dos resultados da consulta. O Athena usará o resultado armazenado desde que não seja mais antigo do que o período especificado. Para obter mais informações, consulte Reduzir custo e melhorar o desempenho de consultas com Amazon Athena no AWSBlog Big Data.

> **Note**
>
> O recurso de reutilização de resultados de consulta exige o mecanismo do Athena versão 3. Para obter informações sobre as mudanças nas versões do mecanismo, consulte Alterar versões do mecanismo do Athena.

Principais atributos

- A reutilização dos resultados da consulta é um recurso opcional por consulta. É possível habilitar a reutilização dos resultados da consulta por consulta.
- O período máximo para a reutilização dos resultados da consulta pode ser especificada em minutos, horas ou dias. O período máximo especificado é equivalente a sete dias, independentemente da unidade de tempo usada. O padrão é 60 minutos.
- Ao habilitar a reutilização de resultados para uma consulta, o Athena irá procurar uma execução anterior da consulta no mesmo grupo de trabalho. Se o Athena encontrar resultados de consulta armazenados correspondentes, ele não executará novamente a consulta, mas direcionará para o local do resultado anterior ou buscará dados nele.

- Para qualquer consulta que habilite a opção de reutilização de resultados, o Athena reutiliza o último resultado da consulta salvo na pasta do grupo de trabalho somente quando todas as condições a seguir são verdadeiras:
  - A string de consulta é uma correspondência exata.
  - O banco de dados e o nome do catálogo correspondem.

- O resultado anterior não é mais antigo que o período máximo especificado, ou não é mais antigo que 60 minutos, caso um período máximo não tenha sido especificado.
- O Athena reutiliza somente uma execução que tenha exatamente a mesma configuração de resultado da execução atual.
- Você tem acesso a todas as tabelas referenciadas na consulta.
- Você tem acesso ao local do arquivo S3 no qual o resultado anterior está armazenado.

Se alguma dessas condições não for atendida, o Athena executará a consulta sem usar os resultados armazenados em cache.

Considerações e limitações

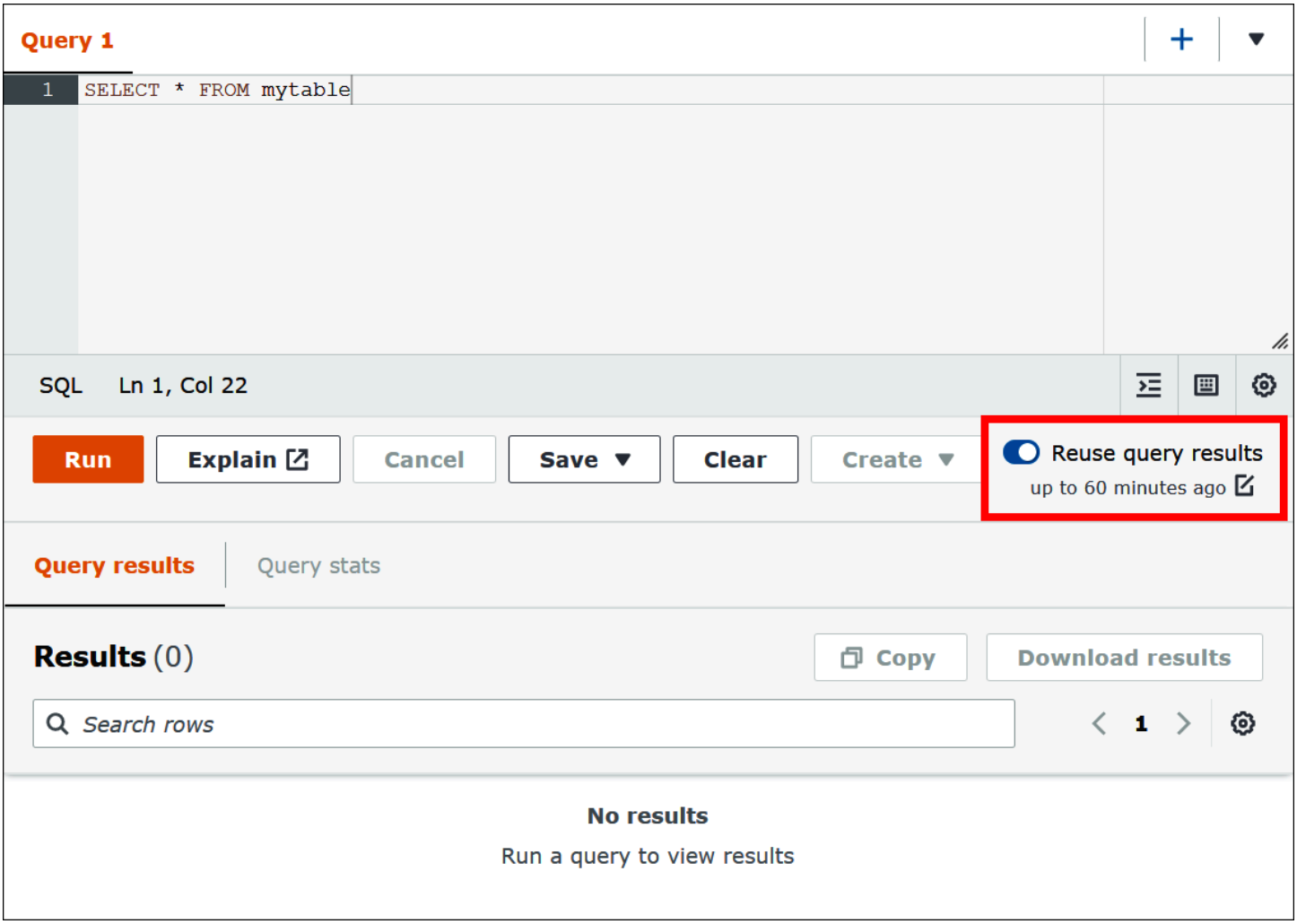
Ao usar o recurso de reutilização dos resultados da consulta, lembre-se dos seguintes pontos:

- O Athena reutiliza os resultados da consulta somente dentro do mesmo grupo de trabalho.
- O recurso de reutilização dos resultados da consulta respeita as configurações do grupo de trabalho. Se você substituir a configuração de resultado de uma consulta, o recurso será desativado.
- Tabelas do Apache Hive, Apache Hudi, Apache Iceberg e Linux Foundation Delta Lake registradas com AWS Glue são suportados. Metastores do Hive externos não são compatíveis.
- Não há suporte para as consultas que fazem referência a catálogos federados ou a um metastore Hive externo.
- Não há suporte para a reutilização dos resultados da consulta para tabelas governadas do Lake Formation.
- A reutilização do resultado da consulta não é compatível quando o local da origem da tabela no Amazon S3 é registrado como um local de dados no Lake Formation.
- Não há suporte para tabelas com permissões de linha e de coluna.
- Tabelas com controle de acesso refinado (por exemplo, filtragem de colunas ou linhas) não são compatíveis.
- Qualquer consulta que referencie uma tabela não compatível não está qualificada para reutilizar os resultados da consulta.
- O Athena requer que você tenha permissões de leitura do Amazon S3 para que o arquivo de saída gerado anteriormente seja reutilizado.

- O recurso de reutilização dos resultados da consulta assume que o conteúdo do resultado anterior não foi modificado. O Athena não verifica a integridade de um resultado anterior antes de usá-lo.
- Se os resultados da consulta da execução anterior foram excluídos ou movidos para um local diferente no Amazon S3, a execução subsequente da mesma consulta não reutilizará os resultados da consulta anterior.
- Há a possibilidade de que resultados potencialmente obsoletos retornem. O Athena não verifica alterações nos dados de origem até que o período máximo de reutilização especificado seja atingido.
- Se diversos resultados estiverem disponíveis para reutilização, o Athena usará o resultado mais recente.
- Consultas que usam operadores ou funções não determinísticas como `rand()` ou `shuffle()` não usam resultados em cache. Por exemplo, `LIMIT` sem `ORDER BY` não é determinístico e não está armazenado em cache, mas `LIMIT` com `ORDER BY` é determinístico e está armazenado em cache.
- Compatibilidade para a reutilização dos resultados da consulta no console do Athena, na API do Athena e no driver JDBC. No momento, a compatibilidade do driver ODBC para reutilização dos resultados da consulta está disponível somente para o Windows.
- Para usar o recurso de reutilização de resultados de consulta com o JDBC, a versão mínima necessária do driver é 2.0.34.1000. Para ODBC, a versão mínima necessária do driver é 1.1.19.1002. Para obter informações sobre download de drivers, consulte Conectar-se ao Amazon Athena com drivers ODBC e JDBC.
- Não há compatibilidade para reutilizar os resultados da consulta em consultas que usem mais de um catálogo de dados.
- Não há compatibilidade para reutilizar os resultados da consulta em consultas que incluam mais de 20 tabelas.

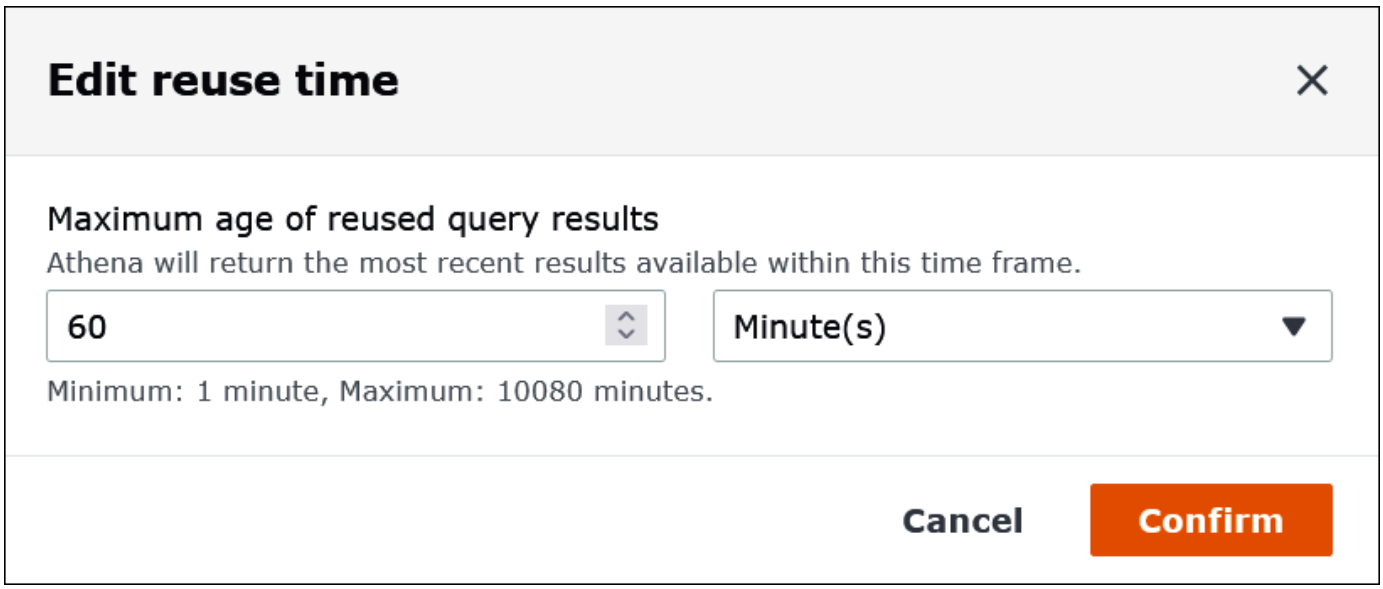
Reutilização dos resultados da consulta no console do Athena

Para usar o recurso, habilite a opção Reuse query results (Reutilizar resultados da consulta) no editor de consultas do Athena.

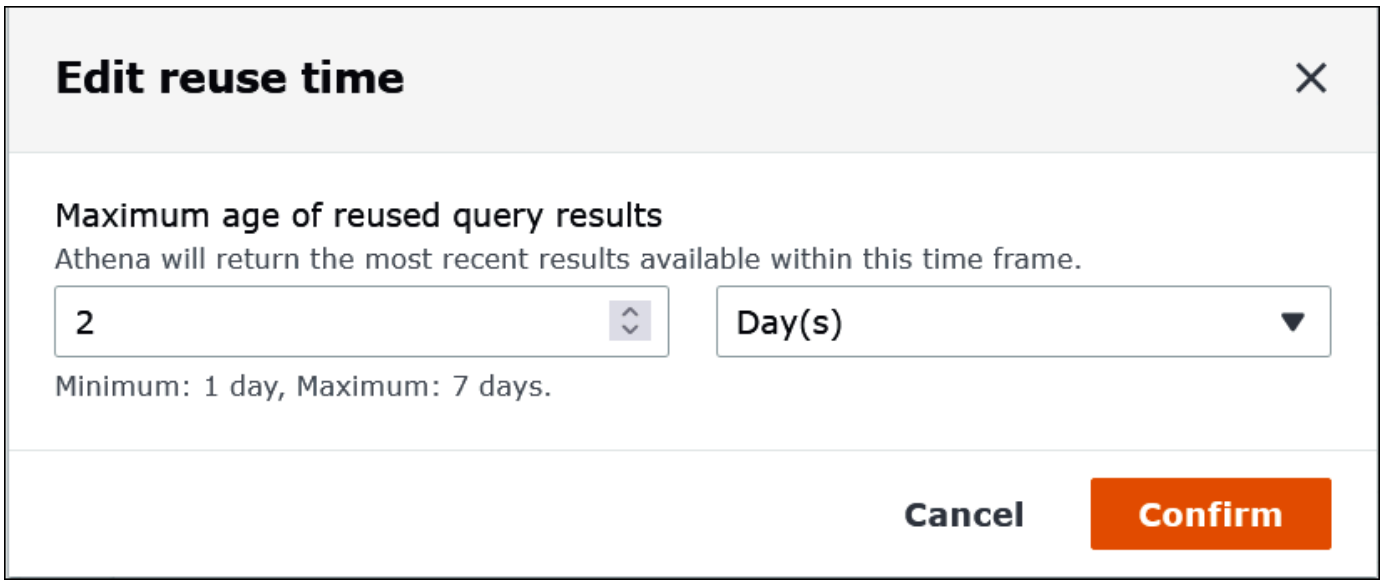


Para configurar o recurso de reutilização dos resultados da consulta

1. No editor de consultas do Athena, na opção Reuse query results (Reutilizar resultados da consulta), escolha o ícone de edição ao lado de up to 60 minutes ago (até 60 minutos atrás).
2. Na caixa de diálogo Edit reuse time (Editar tempo de reutilização), na caixa à direita, escolha uma unidade de tempo (minutos, horas ou dias).
3. Na caixa à esquerda, insira ou escolha o número de unidades de tempo que deseja especificar. O tempo máximo que é possível inserir é o equivalente a sete dias, independentemente da unidade de tempo escolhida.

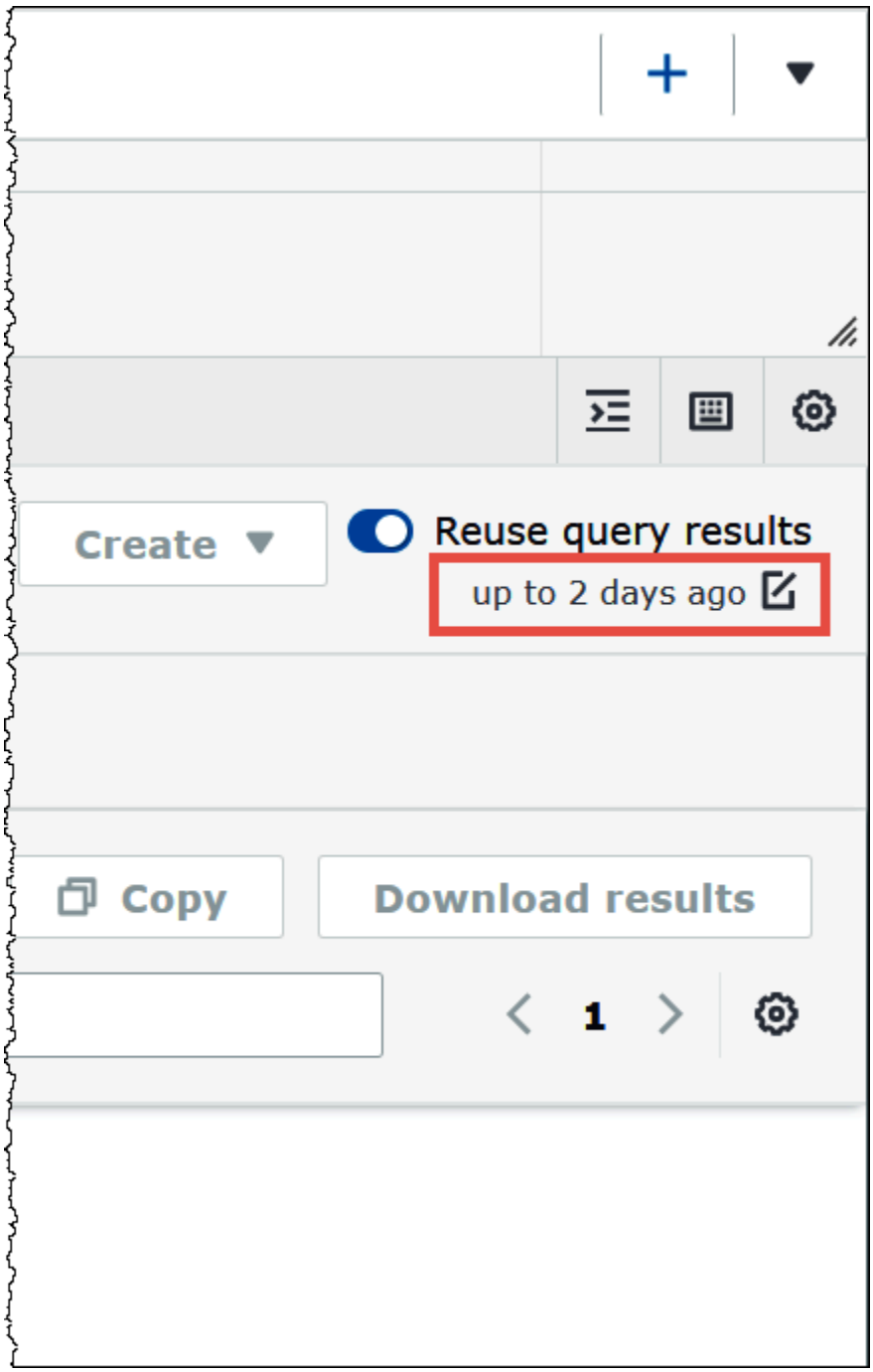


O exemplo a seguir especifica um tempo máximo de reutilização de dois dias.



4. Selecione a opção Confirmar.

   Uma barra de notificação confirmará sua alteração de configuração e a opção Reuse query results (Reutilizar resultados da consulta) exibirá a nova configuração.

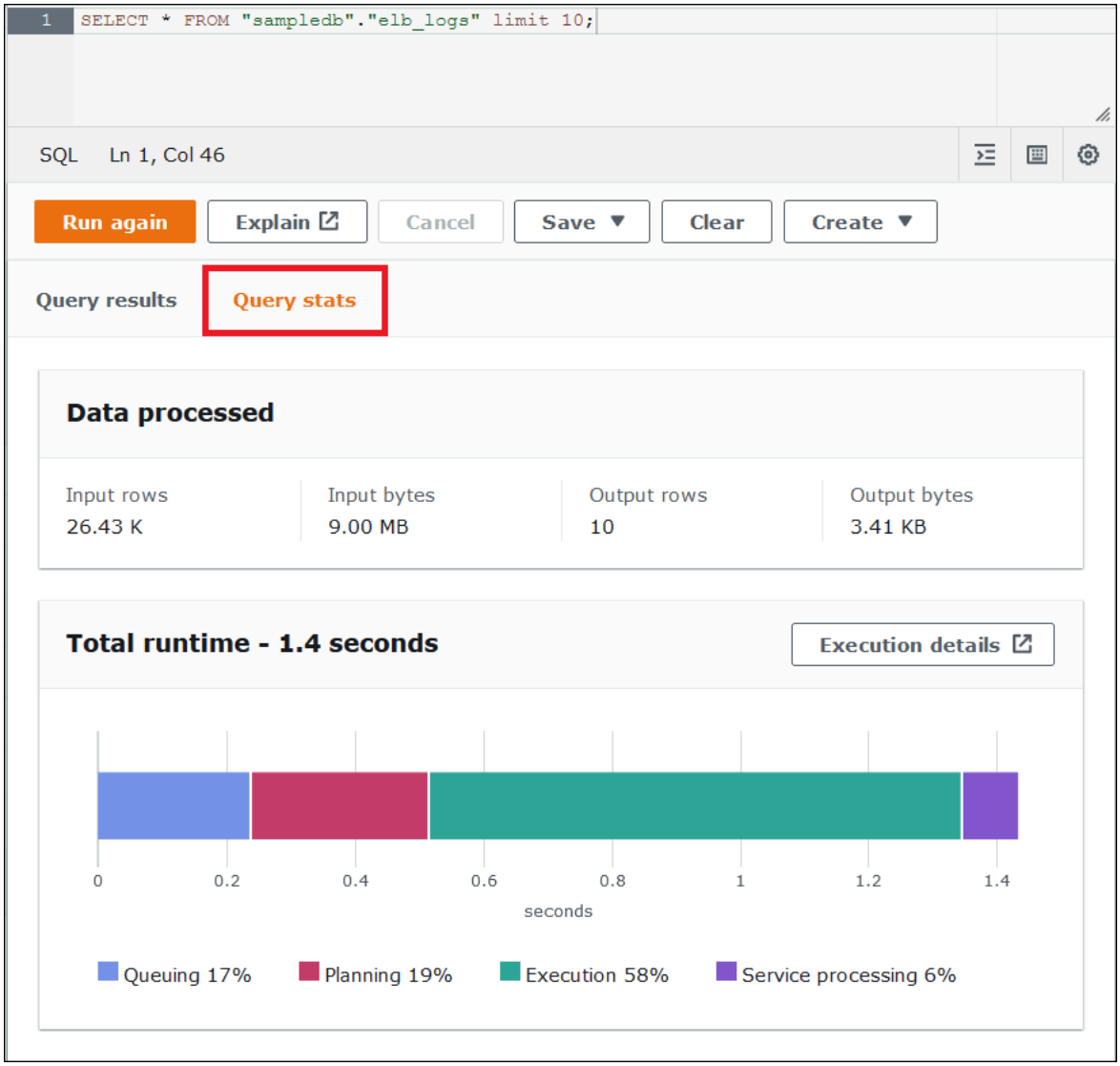


Visualizar estatísticas e detalhes de execução para consultas concluídas

Depois de executar uma consulta, você pode obter estatísticas sobre os dados de entrada e saída processados, ver uma representação gráfica do tempo gasto em cada fase da consulta e explorar os detalhes da execução de maneira interativa.

Para visualizar as estatísticas de uma consulta concluída

1. Depois de executar uma consulta no editor de consultas do Athena, selecione a guia Query stats (Estatísticas da consulta).



A guia Query stats (Estatísticas de consultas) fornece as seguintes informações:

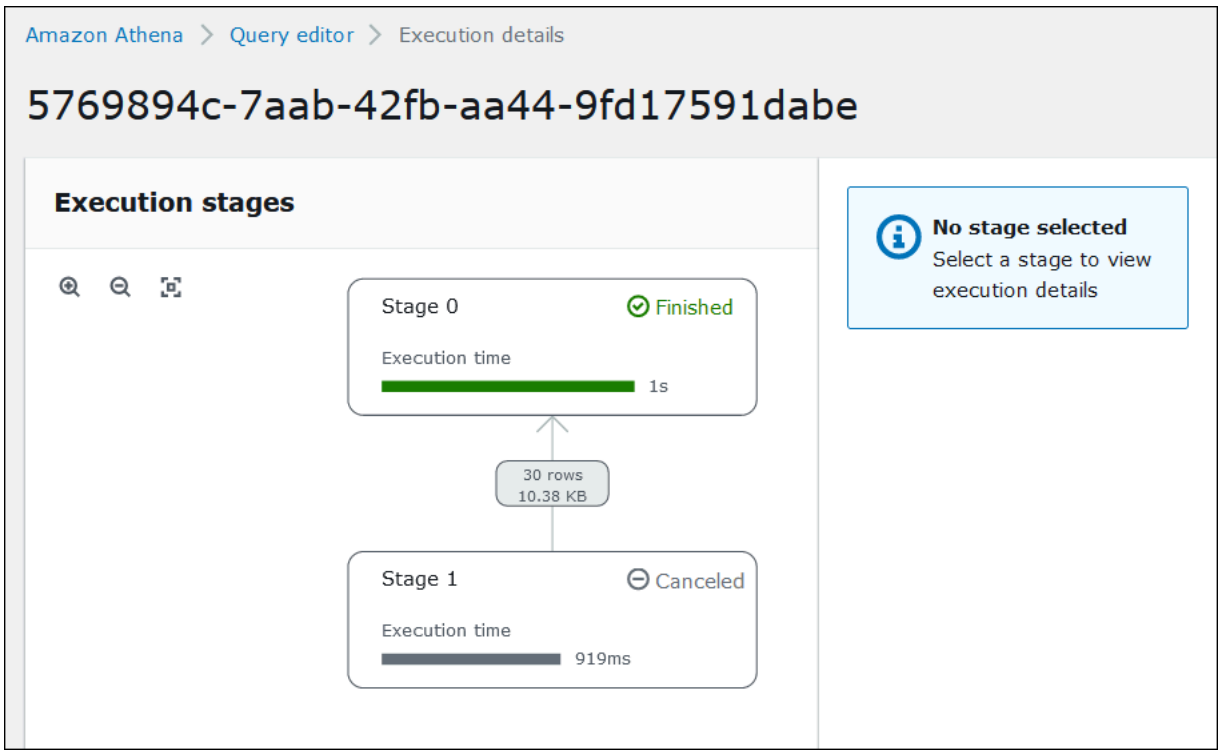
- Data processed (Dados processados): mostra o número de linhas de entrada e bytes processados e o número de linhas e bytes gerados.
- The Total runtime (O tempo de execução total): mostra o tempo total que a consulta levou para ser executada e uma representação gráfica de quanto desse tempo foi gasto em enfileiramento, planejamento, execução e processamento de serviços.

> **Note**
>
> As informações referentes a contagem de linhas e ao tamanho dos dados de entrada e saída em nível de estágio não são mostradas quando uma consulta tem filtros em nível de linha definidos no Lake Formation.

2. Para explorar interativamente as informações sobre como a consulta foi executada, escolha Execution details (Detalhes da execução).

A página Execution details (Detalhes da execução) mostra o ID de execução da consulta e um gráfico dos estágios baseados em zero nessa consulta. As etapas são ordenadas do início ao fim, de baixo para cima. O rótulo de cada estágio mostra quanto tempo o estágio levou para ser executado.

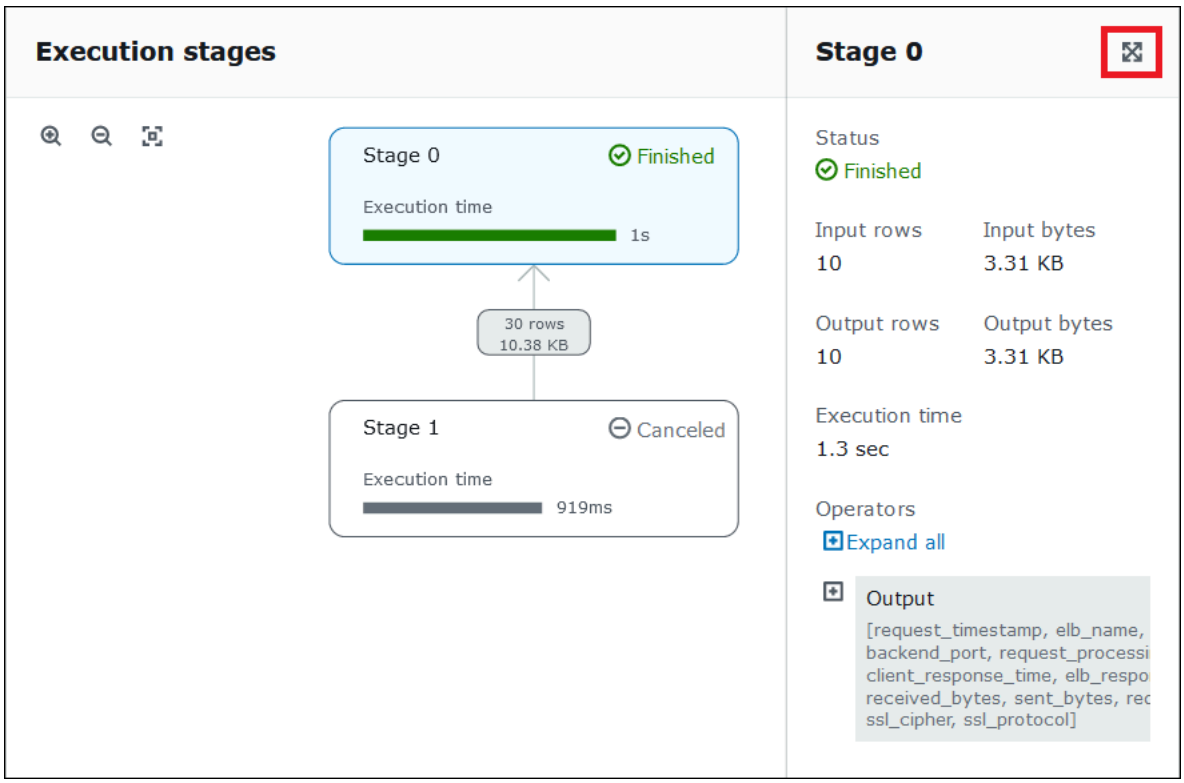
> **Note**
>
> Em geral, o tempo total de execução e o tempo da fase de execução diferem significativamente. Por exemplo, uma consulta com um tempo de execução total em minutos pode mostrar o tempo da fase de execução em horas. Como uma fase é uma unidade lógica de computação executada paralelamente entre várias tarefas, o tempo de execução de uma fase é o tempo de execução agregado de todas as suas tarefas. Apesar dessa discrepância, o tempo de execução da fase pode ser útil como um indicador relativo de qual fase foi computacionalmente mais intensiva em uma consulta.



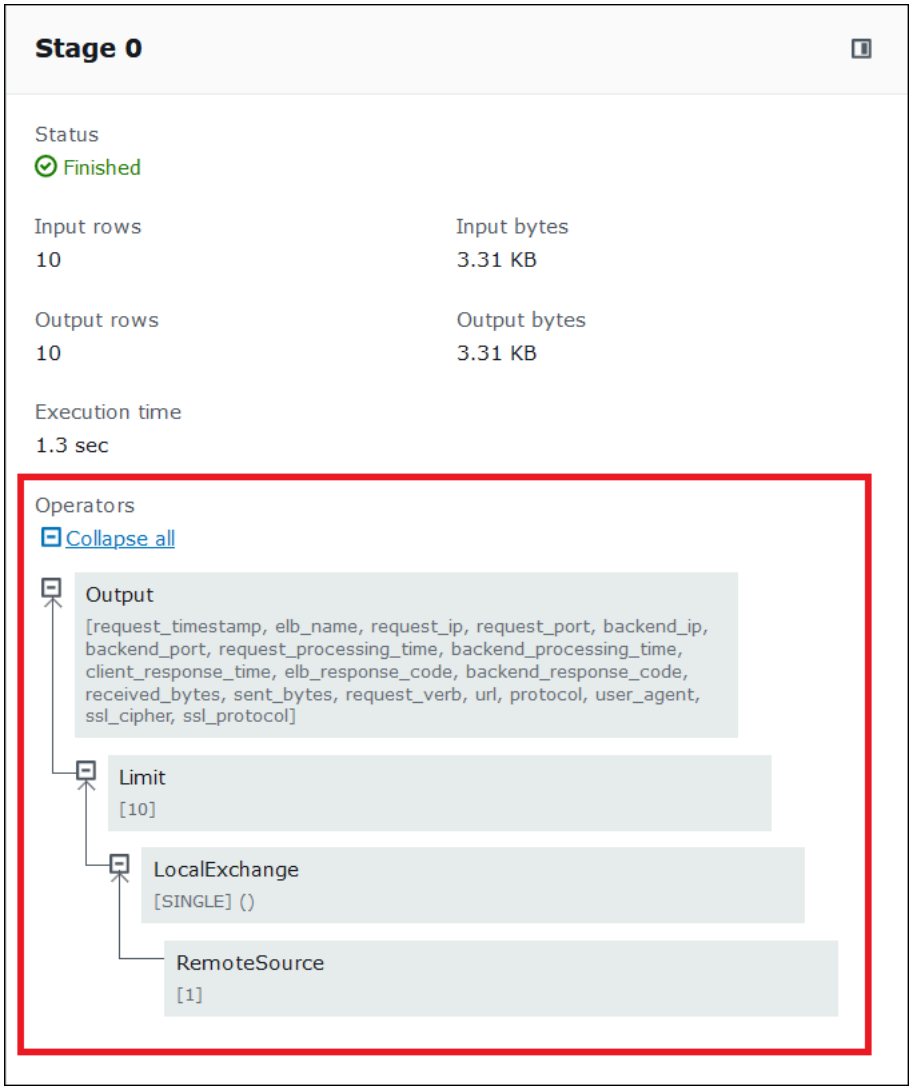
Para navegar pelo gráfico, utilize as seguintes opções:

- Para aumentar ou diminuir o zoom, role com o mouse ou use os ícones de ampliação.
- Para ajustar o gráfico para que ele caiba na tela, selecione o ícone Zoom to fit (Ampliar para caber).
- Para mover o gráfico, arraste o ponteiro do mouse.

3. Para ver mais detalhes de um estágio, escolha esse estágio. O painel de detalhes do estágio à direita mostra o número de linhas e bytes de entrada e saída, bem como uma árvore de operador.



4. Para ver os detalhes do estágio em largura natural, selecione o ícone de expansão na parte superior direita do painel de detalhes.
5. Para obter informações sobre as partes do estágio, expanda um ou mais itens na árvore do operador.



Para mais informações sobre detalhes de execução, consulte Noções básicas sobre os resultados da instrução EXPLAIN do Athena.

Recursos adicionais do

Para obter mais informações, consulte os recursos a seguir.

Visualizar planos de execução para consultas SQL

Usar EXPLAIN e EXPLAIN ANALYZE no Athena

Trabalhar com visualizações

Uma visualização no Amazon Athena é uma tabela lógica, não física. A consulta que define uma exibição é executada sempre que a exibição é referenciada em uma consulta.

Você pode criar uma exibição a partir de uma consulta `SELECT` e, em seguida, fazer referência a essa exibição em futuras consultas. Para ter mais informações, consulte CREATE VIEW.

Tópicos

- Quando usar visualizações?
- Ações com suporte para visualizações no Athena
- Considerações sobre visualizações
- Limitações das visualizações
- Trabalhar com visualizações no console
- Criar visualizações
- Exemplos de visualizações
- Uso das visualizações do AWS Glue Data Catalog

Quando usar visualizações?

Talvez você queira criar visualizações para:

- Consultar um subconjunto de dados. Por exemplo, você pode criar uma visualização com um subconjunto de colunas com base na tabela original para simplificar a consulta de dados.
- Combinar várias tabelas em uma consulta. Se você tem várias tabelas e deseja combiná-las com `UNION ALL`, é possível criar uma visualização com essa expressão para simplificar consultas em tabelas combinadas.
- Ocultar a complexidade de consultas básicas existentes e simplificar as consultas executadas pelos usuários. As consultas básicas geralmente incluem junções entre tabelas, expressões na lista de colunas e outra sintaxe de SQL que dificultam o entendimento e a depuração delas. Você pode criar uma exibição que oculta a complexidade e simplifica consultas.
- Experimente com técnicas de otimização e crie consultas otimizadas. Por exemplo, se você encontrar uma combinação de condições `WHERE`, ordem `JOIN` ou outras expressões que demonstram a melhor performance, você poderá criar uma exibição com essas cláusulas e expressões. Os aplicativos podem realizar consultas relativamente simples contra essa exibição.

Posteriormente, se você encontrar uma melhor maneira de otimizar a consulta original, quando você recriar a exibição, todos os aplicativos aproveitam imediatamente a consulta básica otimizada.

- Oculte a tabela subjacente e os nomes de coluna e minimize os problemas de manutenção se esses nomes forem alterados. Nesse caso, recrie a exibição usando os novos nomes. Todas as consultas que usam a exibição ao invés de tabelas subjacentes continuarão em execução sem alterações.

Ações com suporte para visualizações no Athena

O Athena permite as ações a seguir nas visualizações. Você pode executar esses comandos no editor de consultas.

Statement	Descrição
CREATE VIEW	Cria uma nova exibição a partir de uma consulta `SELECT` especificada. Para ter mais informações, consulte Criar visualizações. A cláusula `OR REPLACE` opcional permite atualizar a exibição existente substituindo-a
DESCRIBE VIEW	Mostra a lista de colunas para a exibição nomeada. Isso permite examinar os atributos de uma exibição complexa.
DROP VIEW	Exclui uma exibição existente. A cláusula `IF EXISTS` opcional suprime o erro se a exibição não existir.
SHOW CREATE VIEW	Mostra a instrução SQL que cria a exibição específica.
SHOW VIEWS	Lista as exibições no banco de dados especificado ou no banco de dados atual se você omitir o nome do banco de dados. Use a cláusula `LIKE` opcional com uma expressão regular para restringir a lista de nomes de exibições. Você também pode ver a lista de exibições no painel esquerdo no console.
SHOW COLUMNS	Liste as colunas no esquema para obter uma visualização.

Considerações sobre visualizações

As seguintes considerações se aplicam à criação e ao uso de visualizações no Athena:

- No Athena, você pode previsualizar e trabalhar com visualizações criadas no console do Athena, no AWS Glue Data Catalog, caso tenha migrado para usá-lo, ou com o Presto em execução no cluster do Amazon EMR conectado ao mesmo catálogo. Você não pode previsualizar nem adicionar ao Athena visualizações que foram criadas de outras maneiras.
- Se você está criando visualizações pelo Catálogo de dados do AWS Glue, deve incluir o parâmetro `PartitionKeys` e definir o valor como uma lista vazia, da seguinte forma: `"PartitionKeys": []`. Caso contrário, sua consulta de visualização falhará no Athena. O seguinte exemplo mostra uma visualização criada no Catálogo de dados com `"PartitionKeys":[]`:

```
aws glue create-table
--database-name mydb
--table-input '{
"Name":"test",
  "TableType": "EXTERNAL_TABLE",
  "Owner": "hadoop",
  "StorageDescriptor":{
     "Columns":[{
           "Name":"a","Type":"string"},{"Name":"b","Type":"string"}],
   "Location":"s3://DOC-EXAMPLE-BUCKET/Oct2018/25Oct2018/",
   "InputFormat":"org.apache.hadoop.mapred.TextInputFormat",
   "OutputFormat": "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat",
   "SerdeInfo":{"SerializationLibrary":"org.apache.hadoop.hive.serde2.OpenCSVSerde",
   "Parameters":{"separatorChar": "|", "serialization.format":
 "1"}}},"PartitionKeys":[]}'
```

- Se você criou visualizações do Athena no catálogo de dados, o catálogo as trata como tabelas. Você pode usar o controle de acesso detalhado no nível da tabela no catálogo de dados para restringir o acesso a essas visualizações.
- O Athena impede que você execute visualizações repetidas e exibe uma mensagem de erro nesses casos. Uma exibição recursiva é uma consulta de exibição que faz referência a si mesmo.
- O Athena exibe uma mensagem de erro quando detecta visualizações obsoletas. Uma exibição obsoleta é relatada quando um dos seguintes itens ocorrer:
  - A exibição faz referência a tabelas ou bancos de dados que não existem.
  - Uma alteração de esquema ou metadados é feita em uma tabela referenciada.

  - Uma tabela referenciada é descartada e recriada com um esquema ou uma configuração diferente.
- Você pode criar e executar exibições aninhadas, desde que a consulta por trás da exibição aninhada seja válida e as tabelas e os bancos de dados existirem.

Limitações das visualizações

- Os nomes das visualizações do Athena não podem conter caracteres especiais, exceto sublinhado (`_`). Para ter mais informações, consulte Nomes de tabelas, bancos de dados e colunas.
- Evite usar palavras-chave reservadas para nomear visualizações. Se você usar palavras-chave reservadas, use aspas duplas para delimitar as palavras-chave reservadas em suas consultas em visualizações. Consulte Palavras-chave reservadas.
- Não é possível usar visualizações criadas no Athena com metastores externos do Hive ou UDFs. Para obter informações sobre como trabalhar com visualizações criadas externamente no Hive, consulte Usar visualizações do Hive.
- Não é possível usar visualizações com funções geoespaciais.
- Não é possível usar as visualizações para gerenciar o controle de acesso aos dados no Amazon S3. Para consultar uma visualização, é necessário ter permissões para acessar os dados armazenados no Amazon S3. Para ter mais informações, consulte Acesso ao Amazon S3.
- Embora seja compatível com consulta de visualizações entre contas tanto no mecanismo Athena versão 2 como no mecanismo Athena versão 3, não é possível criar uma visualização que inclua um AWS Glue Data Catalog entre contas. Para obter informações sobre o acesso a catálogos de dados entre contas, consulte Acesso aos catálogos de dados do AWS Glue entre contas.
- As colunas ocultas de metadados do Hive ou Iceberg `$bucket`, `$file_modified_time`, `$file_size` e `$partition` não são compatíveis para visualizações no Athena. Para obter informações sobre como usar a coluna `$path` de metadados no Athena, consulte Obter os locais de arquivos dos dados de origem no Amazon S3 .

Trabalhar com visualizações no console

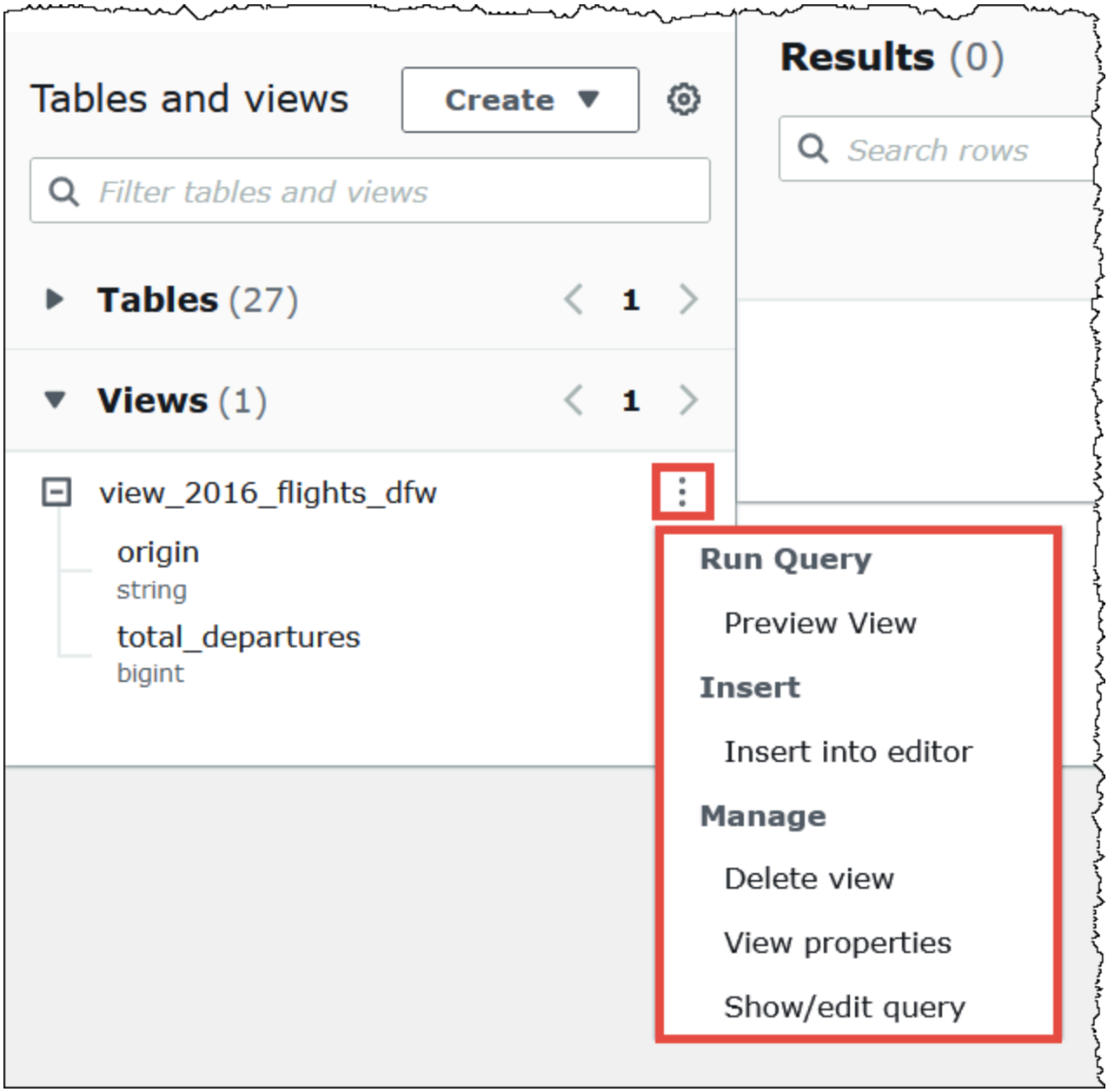
No console do Athena, é possível:

- Localizar todas as visualizações no painel esquerdo, onde as tabelas estão listadas.
- Filtre as exibições.
- Visualize uma exibição, mostre suas propriedades, edite-a ou exclua-a.

Para mostrar as ações de uma exibição

Uma exibição é mostrada no console somente se você já a criou.

1. No console do Athena, escolha Views (Visualizações) e, em seguida, escolha uma visualização para expandi-la e mostrar as colunas na visualização.
2. Escolha os três pontos verticais ao lado da visualização para mostrar uma lista de ações dela.



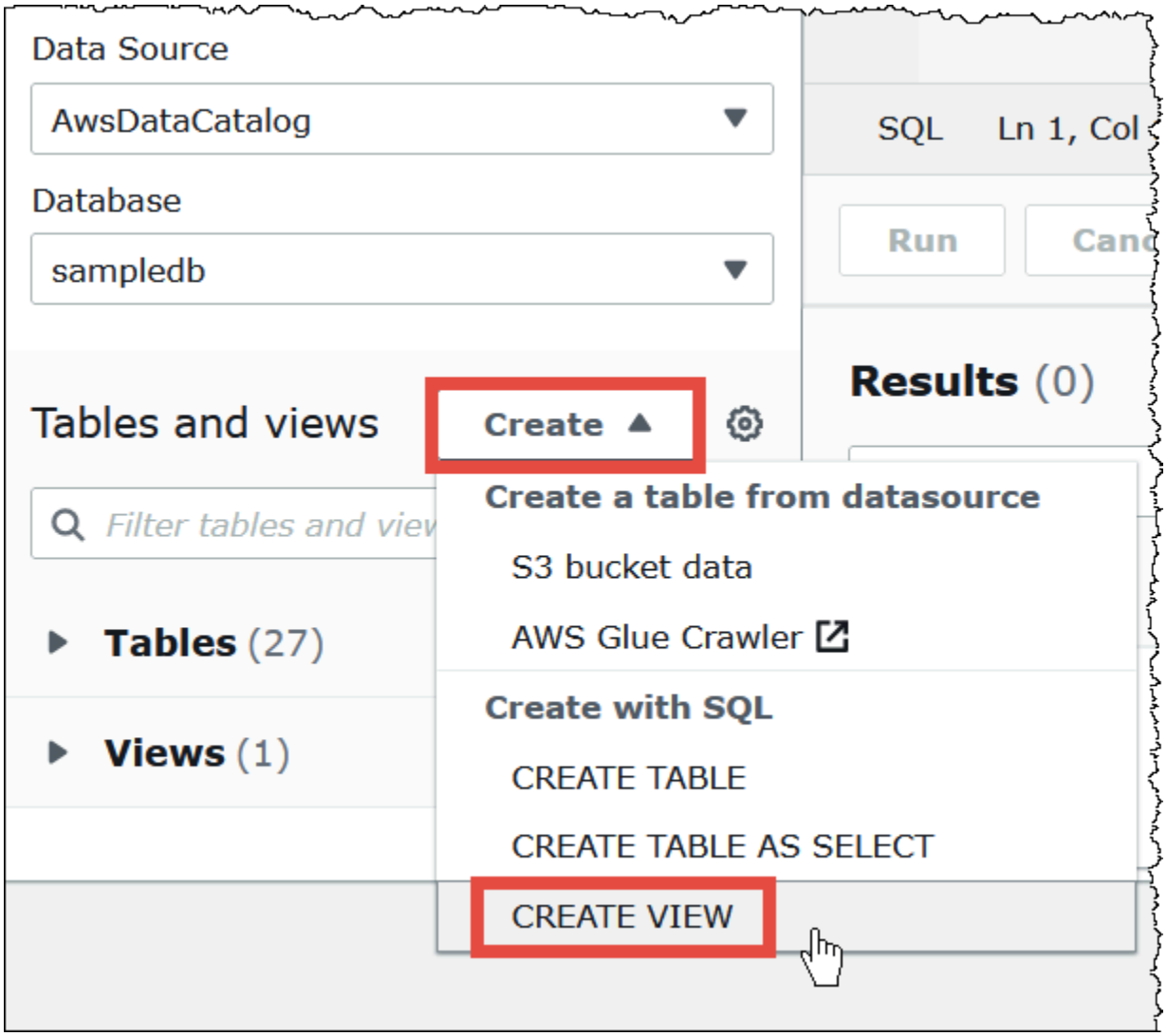
3. Escolha as ações para a visualização: exibir uma prévia, inserir o nome o editor de consultas, excluir, consultar as propriedades ou exibir e editar no editor de consultas.

Criar visualizações

Você pode criar uma visualização no console do Athena usando um modelo ou executando uma consulta existente.

Para usar um modelo para criar uma visualização

1. No console do Athena, ao lado de Tables and views (Tabelas e visualizações), escolha Create (Criar) e, em seguida, escolha Create view (Criar visualização).



   Essa ação coloca um modelo de visualização editável no editor de consultas.

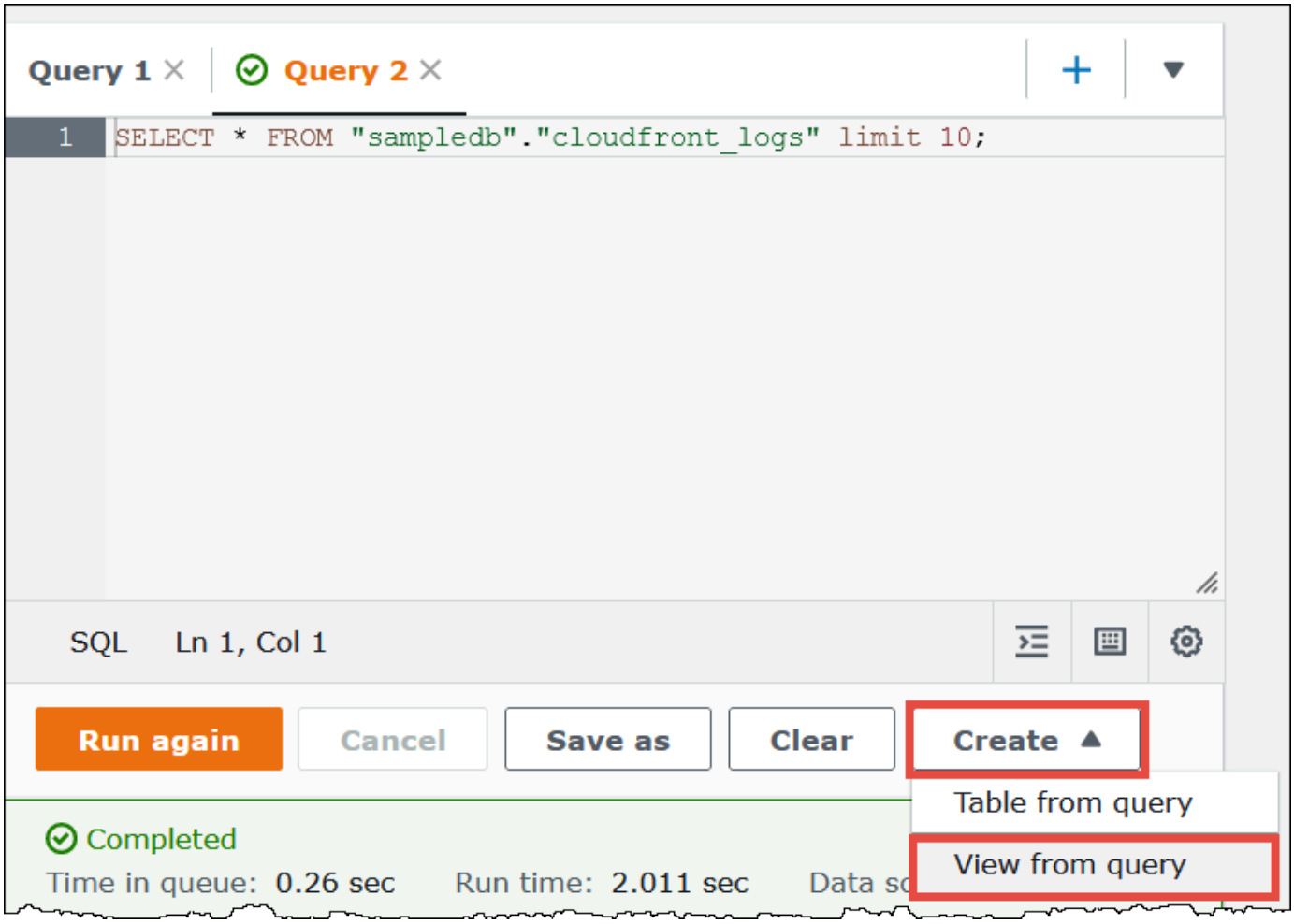
2. Edite o modelo de visualização de acordo com suas necessidades. Quando você digitar um nome para a visualização na instrução, lembre-se de que os nomes das visualizações não podem conter caracteres especiais além do sublinhado (_). Consulte Nomes de tabelas, bancos de dados e colunas. Evite usar Palavras-chave reservadas para nomear visualizações.

Para obter mais informações sobre a criação de visualizações, consulte CREATE VIEW e Exemplos de visualizações.

3. Selecione Run (Executar) para criar a visualização. A visualização aparece na lista de visualizações no console do Athena.

Para criar uma visualização a partir de uma consulta existente

1. Use o editor de consultas do Athena para executar uma consulta existente.
2. Na janela do editor de consultas, escolha Create (Criar) e, em seguida, escolha View from query (Visualização a partir de consulta).



3. Na caixa de diálogo Create View (Criar visualização), insira um nome para a visualização e, em seguida, escolha Create (Criar). Os nomes de visualizações não podem conter caracteres especiais, exceto sublinhado (_). Consulte Nomes de tabelas, bancos de dados e colunas. Evite usar Palavras-chave reservadas para nomear visualizações.

O Athena adiciona a visualização à lista de visualizações no console e exibe a instrução `CREATE VIEW` para a visualização no editor de consulta.

Observações

- Se você excluir uma tabela em que outra tabela se baseia e, depois, tentar executar a visualização, o Athena exibirá uma mensagem de erro.
- Você pode criar uma visualização aninhada, que fica acima de uma visualização existente. O Athena impede que você execute uma visualização repetida com referência a ela mesma.

Exemplos de visualizações

Para mostrar a sintaxe da consulta de exibição, use SHOW CREATE VIEW.

Example Exemplo 1

Considere as duas tabelas a seguir: uma tabela `employees` com duas colunas, `id` e `name` e uma tabela `salariescom` duas colunas, `id` e `salary`.

Neste exemplo, criamos uma exibição chamada `name_salary` como uma consulta `SELECT` que obtém uma lista de IDs mapeados para salários a partir das tabelas `employees` e `salaries`:

```
CREATE VIEW name_salary AS
SELECT
 employees.name,
 salaries.salary
FROM employees, salaries
WHERE employees.id = salaries.id
```

Example Exemplo 2

No exemplo a seguir, criamos uma exibição chamada `view1` que permite que você oculte a sintaxe de consulta mais complexa.

Essa exibição é executada sobre duas tabelas, `table1` e `table2`, em que cada tabela é uma consulta `SELECT` diferente. A visualização seleciona as colunas de `table1` e combina os resultados com `table2`. A junção é baseada na coluna a presente em ambas as tabelas.

```
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;
```

Para obter informações sobre consultar visualizações federadas, consulte Consultar visualizações federadas.

Uso das visualizações do AWS Glue Data Catalog

> Esse recurso está em prévia de release e sujeito a alterações. Para obter mais informações, consulte a seção Beta e pré-visualizações no documento de Termos de serviço da AWS.

Use as visualizações do AWS Glue Data Catalog quando desejar uma perspectiva única e comum entre os Serviços da AWS, como o Amazon Athena e o Amazon Redshift. Nas visualizações do Catálogo de Dados, as permissões de acesso são definidas pelo usuário que criou a visualização, e não pelo usuário que consulta a visualização. Esse método de concessão de permissões é chamado de semântica do programador.

Os casos de uso apresentados a seguir mostram como é possível usar as visualizações do Catálogo de Dados.

- Maior controle de acesso: você cria uma visualização que restringe o acesso aos dados com base no nível de permissões requeridas pelo usuário. Por exemplo, é possível usar as visualizações do Catálogo de Dados para evitar que colaboradores que não trabalham no departamento de recursos humanos (RH) vejam informações de identificação pessoal.

- Garantia de registros completos: ao aplicar determinados filtros na visualização do Catálogo de Dados, você garante que os registros de dados em uma visualização do Catálogo de Dados estejam sempre completos.
- Segurança aprimorada: nas visualizações do Catálogo de Dados, a definição de consulta que cria a visualização deve estar intacta para que a visualização seja criada. Isso torna as visualizações do Catálogo de Dados menos suscetíveis a comandos SQL de agentes mal-intencionados.
- Prevenção do acesso às tabelas subjacentes: a semântica do programador permite que os usuários acessem uma visualização sem disponibilizar a tabela subjacente para eles. Somente o usuário que define a visualização necessita de acesso às tabelas.

As definições de visualização do Catálogo de Dados são armazenadas no AWS Glue Data Catalog. Isso significa que você pode usar o AWS Lake Formation para conceder acesso usando concessões de recursos, concessões de colunas ou controles de acesso baseados em etiquetas. Para obter mais informações sobre como conceder e revogar acesso no Lake Formation, consulte Granting and revoking permissions on Data Catalog resources no Guia do desenvolvedor do AWS Lake Formation.

Permissões

As visualizações do Catálogo de Dados requerem três perfis: `Lake Formation Admin`, `Definer` e `Invoker`.

- **`Lake Formation Admin`**: tem acesso para configurar todas as permissões do Lake Formation.
- **`Definer`**: cria a visualização do Catálogo de Dados. O perfil `Definer` deve ter permissões `SELECT` completas e concedíveis em todas as tabelas subjacentes às quais a definição de visualização faz referência.
- **`Invoker`**: pode consultar a visualização do Catálogo de Dados ou verificar seus metadados.

As relações de confiança do perfil `Definer` devem permitir a ação `sts:AssumeRole` para os principais de serviço do AWS Glue e do Lake Formation, como apresentado no exemplo a seguir.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
```

```
                    "glue.amazonaws.com",
                    "lakeformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

As permissões do IAM para o acesso ao Athena também são necessárias. Para ter mais informações, consulte Politicas gerenciadas pela AWS para o Amazon Athena.

Limitações

- As visualizações do Catálogo de Dados não podem fazer referência a outras visualizações.
- É possível fazer referência a até dez tabelas na definição de visualização.
- As tabelas subjacentes devem estar registradas no Lake Formation.
- A entidade principal `DEFINER` pode ser somente um perfil do IAM.
- O perfil `DEFINER` deve ter permissões `SELECT` completas (concedíveis) nas tabelas subjacentes.
- Não há suporte para as visualizações `UNPROTECTED` do Catálogo de Dados.
- Não há suporte para as funções definidas pelo usuário (UDFs) na definição de visualização.
- As fontes de dados federadas do Athena não podem ser usadas em visualizações do Catálogo de Dados.
- As visualizações do Catálogo de Dados não têm suporte para metastores externos do Hive.
- O Athena exibe uma mensagem de erro quando detecta visualizações obsoletas. Uma exibição obsoleta é relatada quando um dos seguintes itens ocorrer:
  - A exibição faz referência a tabelas ou bancos de dados que não existem.
  - Uma alteração de esquema ou metadados é feita em uma tabela referenciada.
  - Uma tabela referenciada é descartada e recriada com um esquema ou uma configuração diferente.

Criação de uma visualização do Catálogo de Dados

O exemplo de sintaxe apresentado a seguir mostra como um usuário do perfil `Definer` cria a visualização `orders_by_date` do Catálogo de Dados. O exemplo pressupõe que o perfil `Definer` tenha permissões `SELECT` completas na tabela `orders` no banco de dados `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
```

Consulta de uma visualização do Catálogo de Dados

Depois que a visualização for criada, o `Lake Formation Admin` poderá conceder permissões `SELECT` na visualização do Catálogo de Dados para as entidades principais do `Invoker`. Em seguida, as entidades principais do `Invoker` poderão consultar a visualização sem ter acesso às tabelas subjacentes básicas referenciadas pela visualização. Veja a seguir um exemplo de consulta do `Invoker`.

```
SELECT * from orders_by_date where price > 5000
```

Atualização de uma visualização do Catálogo de Dados

O `Lake Formation Admin` ou o `Definer` podem usar a sintaxe `ALTER VIEW UPDATE DIALECT` para atualizar a definição de visualização. O exemplo apresentado a seguir modifica a definição de visualização para selecionar colunas da tabela `returns` em vez de usar a tabela `orders`.

```
ALTER VIEW orders_by_date UPDATE DIALECT
AS
SELECT return_date, sum(totalprice) AS price
FROM returns
WHERE order_city = 'SEATTLE'
GROUP BY orderdate
```

Para obter mais informações sobre a sintaxe para criar e gerenciar visualizações do Catálogo de Dados, consulte Sintaxe da visualização do Catálogo de Dados do Glue.

Sintaxe da visualização do Catálogo de Dados do Glue

Esse recurso está em prévia de release e sujeito a alterações. Para obter mais informações, consulte a seção Beta e pré-visualizações no documento de Termos de serviço da AWS.

Esta seção descreve os comandos da linguagem de definição de dados (DDL) para criar e gerenciar visualizações do AWS Glue Data Catalog.

ALTER VIEW DIALECT

É possível atualizar as visualizações do Catálogo de Dados ao adicionar um dialeto para o mecanismo ou ao atualizar ou descartar um dialeto do mecanismo existente. Somente o `LakeFormationAdmin` e o `Definer` (o usuário que criou a visualização) têm permissões para usar a instrução `ALTER VIEW DIALECT` em uma visualização do Catálogo de Dados.

Sintaxe

```
ALTER VIEW name [ FORCE ] [ ADD|UPDATE ] DIALECT AS query
```

```
ALTER VIEW name [ DROP ] DIALECT
```

FORCE

A palavra-chave `FORCE` faz com que as informações conflitantes do dialeto do mecanismo em uma visualização sejam substituídas pela nova definição. A palavra-chave `FORCE` é útil quando uma atualização em uma visualização do Catálogo de Dados resulta em definições de visualizações conflitantes entre os dialetos dos mecanismos existentes. Suponha que uma visualização do Catálogo de Dados tenha os dialetos do Athena e do Amazon Redshift e que a atualização resulte em um conflito com o Amazon Redshift na definição de visualização. Nesse caso, é possível usar a palavra-chave `FORCE` para permitir que a atualização seja concluída e marcar o dialeto do Amazon Redshift como obsoleto. Quando mecanismos marcados como obsoletos consultam a visualização, a consulta apresenta falhas. Os mecanismos lançam uma exceção para rejeitar resultados obsoletos. Para corrigir isso, atualize os dialetos obsoletos na visualização.

ADD

Adiciona um novo dialeto do mecanismo à visualização do Catálogo de Dados. O mecanismo especificado ainda não deve existir na visualização do Catálogo de Dados.

UPDATE

Atualiza um dialeto do mecanismo que já existe na visualização do Catálogo de Dados.

DROP

Descarta um dialeto do mecanismo existente de uma visualização do Catálogo de Dados. Após descartar um mecanismo de uma visualização do Catálogo de Dados, a visualização do Catálogo de Dados não poderá ser consultada pelo mecanismo que foi descartado. Outros dialetos do mecanismo na visualização ainda podem consultar a visualização.

DIALECT AS

Apresenta uma consulta SQL específica para o mecanismo.

Exemplos

```
ALTER VIEW orders_by_date FORCE ADD DIALECT
AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate
```

```
ALTER VIEW orders_by_date FORCE UPDATE DIALECT
AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate
```

```
ALTER VIEW orders_by_date DROP DIALECT
```

CREATE PROTECTED MULTI DIALECT VIEW

Cria uma visualização do Catálogo de Dados no AWS Glue Data Catalog. Uma visualização do Catálogo de Dados corresponde a um esquema de visualização única que funciona sem complicações no Athena e em outros mecanismos SQL, como o Amazon Redshift e o Amazon EMR.

Sintaxe

```
CREATE [ OR REPLACE ] PROTECTED MULTI DIALECT VIEW view_name
 [ SECURITY DEFINER ]
AS query
```

PROTECTED

Palavra-chave obrigatória. Especifica que a visualização está protegida contra vazamentos de dados. As visualizações do Catálogo de Dados podem ser criadas somente como uma visualização `PROTECTED`.

MULTI DIALECT

Especifica que a visualização oferece suporte aos dialetos SQL de diferentes mecanismos de consulta e, portanto, que ela pode ser lida por esses mecanismos.

SECURITY DEFINER

Especifica que a semântica do programador está em vigor para essa visualização. A semântica do programador significa que as permissões de leitura efetivas nas tabelas subjacentes pertencem à entidade principal ou ao perfil que definiu a visualização, e não à entidade principal que executa a leitura.

OR REPLACE

Uma visualização do Catálogo de Dados não poderá ser substituída se dialetos SQL de outros mecanismos estiverem presentes na visualização. Se o mecanismo de chamada tiver o único dialeto SQL presente na visualização, a visualização poderá ser substituída.

Exemplo

O exemplo apresentado a seguir cria a visualização `orders_by_date` do Catálogo de Dados com base em uma consulta na tabela `orders`.

```
CREATE PROTECTED MULTI DIALECT VIEW orders_by_date
SECURITY DEFINER
AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
WHERE order_city = 'SEATTLE'
GROUP BY orderdate
```

DESCRIBE

Mostra a lista de colunas para a visualização do Catálogo de Dados especificada. A instrução `DESCRIBE` é semelhante à instrução `DESCRIBE` para visualizações do Athena. Ao contrário

das visualizações do Athena, a saída da instrução é controlada pelo controle de acesso do Lake Formation. Dessa forma, a saída dessa consulta não corresponde a todas as colunas da visualização, mas somente às colunas às quais o chamador tem acesso.

Sintaxe

```
DESCRIBE [db_name.]view_name
```

Exemplos

```
DESCRIBE orders
```

DROP VIEW

Descarta uma visualização do Catálogo de Dados somente se o dialeto do mecanismo de chamada está presente na visualização do Catálogo de Dados. Por exemplo, se um usuário chamar `DROP VIEW` usando o Athena, a visualização será descartada somente se o dialeto do Athena existir na visualização. Caso contrário, haverá falha na operação. Somente o administrador do Lake Formation e o programador de visualizações têm permissões para usar a instrução `DROP VIEW` em uma visualização do Catálogo de Dados.

Sintaxe

```
DROP VIEW [ IF EXISTS ] view_name
```

Exemplos

```
DROP VIEW orders_by_date
```

```
DROP FORCE VIEW IF EXISTS orders_by_date
```

A cláusula `IF EXISTS` opcional faz com que o erro seja suprimido se a exibição não existir.

SHOW COLUMNS

Mostra somente os nomes das colunas para uma única visualização do Catálogo de Dados especificada. A instrução `SHOW COLUMNS` é semelhante à instrução `SHOW COLUMNS` para visualizações do Athena. Ao contrário das visualizações do Athena, a saída da instrução é controlada

pelo controle de acesso do Lake Formation. Dessa forma, a saída dessa consulta não corresponde a todas as colunas da visualização, mas somente às colunas às quais o chamador tem acesso.

Sintaxe

```
SHOW COLUMNS {FROM|IN} database_name.view_name
```

```
SHOW COLUMNS {FROM|IN} view_name [{FROM|IN} database_name]
```

MOSTRAR CRIAR EXIBIÇÃO

Mostra a sintaxe SQL que criou a visualização do Catálogo de Dados. O SQL retornado mostra a sintaxe de criação de visualização usada no Athena. Somente as entidades principais do administrador do Lake Formation e do programador de visualizações estão autorizadas a chamar `SHOW CREATE VIEW` em uma visualização do Catálogo de Dados.

Sintaxe

```
SHOW CREATE VIEW view_name
```

Exemplos

```
SHOW CREATE VIEW orders_by_date
```

MOSTRAR EXIBIÇÕES

Lista os nomes de todas as visualizações no banco de dados. Todas as visualizações do Catálogo de Dados no banco de dados que têm o dialeto SQL do mecanismo Athena são listadas. Outras visualizações do Catálogo de Dados que não têm o dialeto do mecanismo Athena presente na visualização são filtradas.

Sintaxe

```
SHOW VIEWS [IN database_name] [LIKE 'regular_expression']
```

Exemplos

```
SHOW VIEWS
```

```
SHOW VIEWS IN marketing_analytics LIKE 'orders*'
```

Usar consultas salvas

É possível usar o console do Athena para salvar, editar, executar, renomear e excluir consultas criadas no editor de consultas.

Considerações e limitações

- É possível atualizar o nome, a descrição e o texto de consultas salvas.
- Apenas é possível atualizar as consultas na sua própria conta.
- Você não pode alterar o grupo de trabalho ou o banco de dados ao qual essa consulta pertence.
- O Athena não mantém um histórico de modificações de consultas. Se quiser manter uma versão específica de uma consulta, salve-a com um nome diferente.

Trabalhar com consultas salvas no console do Athena

Para salvar uma consulta e dar um nome para ela

1. Insira ou execute uma consulta no editor de consultas do console do Athena.
2. Acima da janela do editor de consultas, na guia para a consulta, selecione os três pontos verticais e, em seguida, escolha Save as (Salvar como).
3. Na caixa de diálogo Save query (Salvar consulta), insira um nome para a consulta e uma descrição opcional. Você pode usar a janela expansível Preview SQL query (Visualizar consulta SQL) para verificar o conteúdo da consulta antes de a salvar.
4. Escolha Save query (Salvar consulta).

   No editor de consultas, a guia referente à consulta mostra o nome especificado.

Como executar uma consulta salva

1. No console do Athena, escolha a guia Saved queries (Consultas salvas).
2. Na lista Saved queries (Consultas salvas), escolha o ID da consulta que deseja executar.

   O editor de consultas mostra a consulta escolhida.
3. Escolha Executar.

Para editar uma consulta salva

1. No console do Athena, escolha a guia Saved queries (Consultas salvas).
2. Na lista Saved queries (Consultas salvas), escolha o ID da consulta que deseja editar.
3. Edite consulta no editor de consultas.
4. Execute uma das seguintes etapas:

   - Para executar a consulta, escolha Run (Executar).
   - Para salvar a consulta, selecione os três pontos verticais na guia para a consulta e, em seguida, escolha Save (Salvar).
   - Para salvar a consulta com um nome diferente, selecione os três pontos verticais na guia para a consulta e, em seguida, escolha Save as (Salvar como).

Para renomear ou excluir uma consulta salva já exibida no editor de consultas

1. Selecione os três pontos verticais na guia para a consulta e, em seguida, escolha Rename (Renomear) ou Delete (Excluir).
2. Siga as solicitações para renomear ou excluir a consulta.

Para renomear uma consulta salva não exibida no editor de consultas

1. No console do Athena, escolha a guia Saved queries (Consultas salvas).
2. Marque a caixa de seleção referente à consulta que você deseja renomear.
3. Escolha Rename (Renomear).
4. Na caixa de diálogo Rename query (Renomear consulta), edite o nome da consulta e sua descrição. Você pode usar a janela expansível Preview SQL query (Visualizar consulta SQL) para verificar o conteúdo da consulta antes de a renomear.
5. Escolha Rename query (Renomear consulta).

   A consulta renomeada aparece na lista Saved queries (Consultas salvas).

Para excluir uma consulta salva não exibida no editor de consultas

1. No console do Athena, escolha a guia Saved queries (Consultas salvas).
2. Marque uma ou mais das caixas de seleção referentes às consultas que você deseja excluir.

3. Escolha Excluir.
4. Na solicitação de confirmação, selecione Delete (Excluir).

   Uma ou mais consultas serão removidas da lista Saved queries (Consultas salvas).

Usar a API do Athena para atualizar consultas salvas

Para informações sobre o uso da API do Athena para atualizar uma consulta salva, consulte a ação UpdateNamedQuery na Referência de APIs Athena.

Utilizar consultas parametrizadas

É possível utilizar consultas parametrizadas do Athena para repetir a execução da mesma consulta com valores de parâmetros diferentes no momento da execução e ajudar a evitar ataques de injeção de SQL. No Athena, consultas parametrizadas podem assumir a forma de parâmetros de execução em qualquer consulta DML ou instruções preparadas para SQL.

- Consultas com parâmetros de execução podem ser feitas em uma única etapa e não são específicas para um grupo de trabalho. Coloque pontos de interrogação em qualquer consulta DML para os valores que você deseja parametrizar. Ao executar a consulta, declare os valores do parâmetro de execução sequencialmente. A declaração de parâmetros e a atribuição de valores para esses parâmetros podem ser feitas na mesma consulta, mas de maneira dissociada. Ao contrário de instruções preparadas, é possível selecionar o grupo de trabalho ao enviar uma consulta com parâmetros de execução.
- Instruções preparadas exigem duas instruções SQL separadas: `PREPARE` e `EXECUTE`. Primeiro, você define os parâmetros na instrução `PREPARE`. Em seguida, executa uma instrução `EXECUTE` que fornece os valores dos parâmetros definidos. Instruções preparadas são específicas de um grupo de trabalho, ou seja, não podem ser executadas fora do contexto do grupo de trabalho ao qual pertencem.

Considerações e limitações

- Há compatibilidade com consultas parametrizadas no mecanismo do Athena versão 2 e versões posteriores. Para obter informações sobre as versões do mecanismo do Athena, consulte Versionamento do mecanismo do Athena.
- Atualmente, as consultas parametrizadas são aceitas apenas nas instruções `SELECT`, `INSERT INTO`, `CTAS` e `UNLOAD`.

- Em consultas parametrizadas, parâmetros são posicionais e representados por`?`. Parâmetros recebem valores de acordo com sua ordem na consulta. Não há suporte para parâmetros nomeados.
- No momento, parâmetros `?` podem ser inseridos somente na cláusula `WHERE`. Não há suporte para a sintaxe do tipo `SELECT ? FROM table`.
- Parâmetros de ponto de interrogação não podem ser inseridos entre aspas duplas ou simples (ou seja, `'?'` e `"?"` não são uma sintaxe válida).
- Para que os parâmetros de execução do SQL sejam tratados como strings, eles devem estar entre aspas simples em vez de aspas duplas.
- Se necessário, você pode usar a função `CAST` ao inserir um valor para um termo parametrizado. Por exemplo, se você tiver uma coluna do tipo `date` parametrizada em uma consulta e quiser consultar a data `2014-07-05`, inserir `CAST('2014-07-05' AS DATE)` como valor do parâmetro retornará o resultado.
- Instruções preparadas são específicas de um grupo de trabalho, e seus nomes devem ser exclusivos no grupo de trabalho.
- São necessárias permissões do IAM para as instruções preparadas. Para ter mais informações, consulte Permitir acesso a instruções preparadas.
- Consultas com parâmetros de execução no console do Athena estão limitadas a um máximo de 25 pontos de interrogação.

Consultar com o uso de parâmetros de execução

É possível utilizar espaços reservados de ponto de interrogação em qualquer consulta DML para criar uma consulta parametrizada sem criar uma instrução preparada primeiro. Para executar essas consultas, use o console do Athena, a AWS CLI ou o AWS SDK e declare as variáveis no argumento `execution-parameters`.

Executar consultas com parâmetros de execução no console do Athena

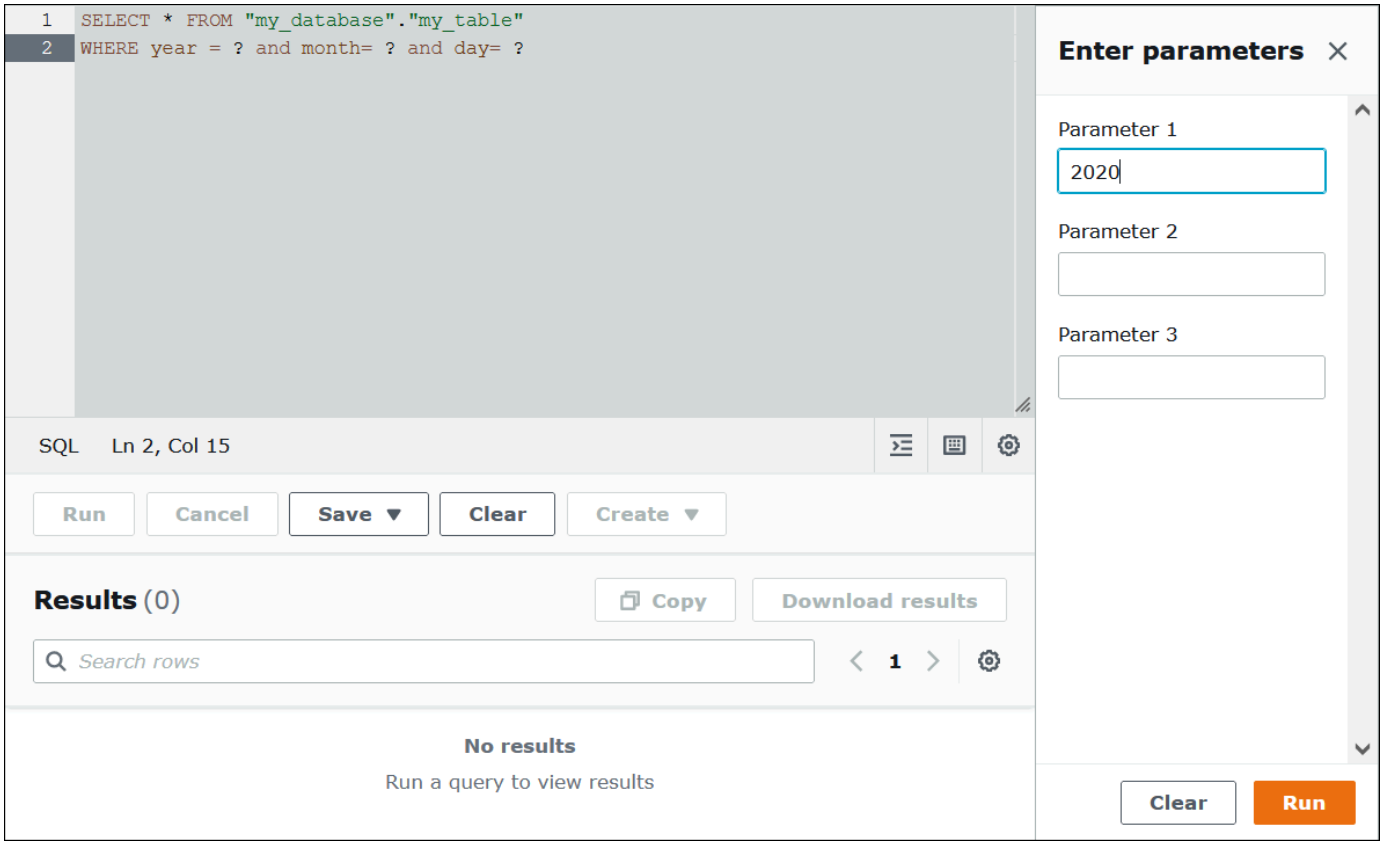
Ao executar uma consulta parametrizada com parâmetros de execução (pontos de interrogação) no console do Athena, você deve informar os valores na ordem em que os pontos de interrogação ocorrem nessa consulta.

Para executar uma consulta com parâmetros de execução

1. Insira uma consulta com espaços reservados de ponto de interrogação no editor do Athena, como no seguinte exemplo.

```
SELECT * FROM "my_database"."my_table"
WHERE year = ? and month= ? and day= ?
```

2. Escolha Executar.
3. Na caixa de diálogo Enter parameters (Inserir parâmetros), insira um valor em ordem para cada um dos pontos de interrogação na consulta.



4. Quando terminar de inserir os parâmetros, escolha Run (Executar). O editor mostra os resultados da consulta para os valores de parâmetro inseridos.

Nesse ponto, você pode realizar uma das seguintes ações:

- Insira valores de parâmetros diferentes para a mesma consulta e escolha Run again (Execute novamente).
- Para limpar todos os valores inseridos de uma só vez, escolha Clear (Limpar).
- Para editar a consulta diretamente (por exemplo, para adicionar ou remover pontos de interrogação), feche primeiro a caixa de diálogo Enter parameters (Inserir parâmetros).

- Para salvar a consulta parametrizada para uso posterior, escolha Save (Salvar) ou Save as (Salvar como) e depois dê um nome para ela. Para obter mais informações sobre o uso de consultas salvas, consulte Usar consultas salvas.

Como conveniência, a caixa de diálogo Enter parameters (Inserir parâmetros) memoriza os valores inseridos anteriormente para a consulta, desde que você utilize a mesma guia no editor de consultas.

Executar consultas com parâmetros de execução com a AWS CLI

Para usar a AWS CLI para executar consultas com parâmetros de execução, use o comando `start-query-execution` e forneça uma consulta parametrizada no argumento `query-string`. Em seguida, no argumento `execution-parameters`, forneça os valores para os parâmetros de execução. O exemplo a seguir ilustra essa técnica.

```
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"
```

Executar consultas com instruções preparadas

Você pode usar uma instrução preparada para a execução repetida da mesma consulta com diferentes parâmetros de consulta. Uma instrução preparada contém espaços reservados de parâmetros dos quais os valores são fornecidos no runtime.

> **Note**
>
> O número máximo de instruções preparadas em um grupo de trabalho é mil.

Instruções SQL

É possível utilizar as instruções SQL `PREPARE`, `EXECUTE` e `DEALLOCATE PREPARE` para executar consultas parametrizadas no editor de consultas do console do Athena.

- Para especificar parâmetros nos quais você costuma usar valores literais, use pontos de interrogação na instrução `PREPARE`.

- Para substituir os parâmetros por valores quando você executar a consulta, use a cláusula `USING` na instrução `EXECUTE`.
- Para remover uma instrução preparada do conjunto de instruções preparadas em um grupo de trabalho, use a instrução `DEALLOCATE PREPARE`.

As seções a seguir apresentam mais detalhes sobre cada uma dessas instruções.

PREPARE

Prepara uma instrução para execução futura. As instruções preparadas são salvas no grupo de trabalho atual com o nome que você especificar. A instrução pode incluir parâmetros no lugar de literais para serem substituídos quando a consulta for executada. Os parâmetros que serão substituídos por valores são indicados por pontos de interrogação.

Sintaxe

```
PREPARE statement_name FROM statement
```

A tabela a seguir descreve esses parâmetros.

Parâmetro	Descrição
statement_name	O nome da instrução que será preparada. O nome deve ser exclusivo no grupo de trabalho.
instrução	Uma consulta `SELECT`, `CTAS` ou `INSERT INTO`.

Exemplos de PREPARE

Os exemplos a seguir mostram o uso da instrução `PREPARE`. Os pontos de interrogação indicam os valores que serão inseridos pela instrução `EXECUTE` quando a consulta for executada.

```
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

Executa uma instrução preparada. Os valores dos parâmetros são especificados na cláusula `USING`.

Sintaxe

```
EXECUTE statement_name [USING value1 [ ,value2, ... ] ]
```

`statement_name` é o nome da instrução preparada. *`value1`* e *`value2`* são os valores que serão especificados para os parâmetros na instrução.

Exemplos de EXECUTE

O exemplo a seguir executa a instrução preparada `my_select1`, que não contém parâmetros.

```
EXECUTE my_select1
```

O exemplo a seguir executa a instrução preparada `my_select2`, que contém um único parâmetro.

```
EXECUTE my_select2 USING 2012
```

O exemplo a seguir executa a instrução preparada `my_select3`, que contém dois parâmetros.

```
EXECUTE my_select3 USING 346078, 12
```

O exemplo a seguir fornece um valor de string para um parâmetro na instrução preparada `my_insert`.

```
EXECUTE my_insert USING 'usa'
```

O exemplo a seguir fornece um valor numérico para o parâmetro `productid` na instrução preparada `my_unload`.

```
EXECUTE my_unload USING 12
```

DEALLOCATE PREPARE

Remove a instrução preparada com o nome especificado da lista de instruções preparadas no grupo de trabalho atual.

Sintaxe

```
DEALLOCATE PREPARE statement_name
```

`statement_name` é o nome da instrução preparada que será removida.

Exemplo

O exemplo a seguir remove a instrução preparada `my_select1` do grupo de trabalho atual.

```
DEALLOCATE PREPARE my_select1
```

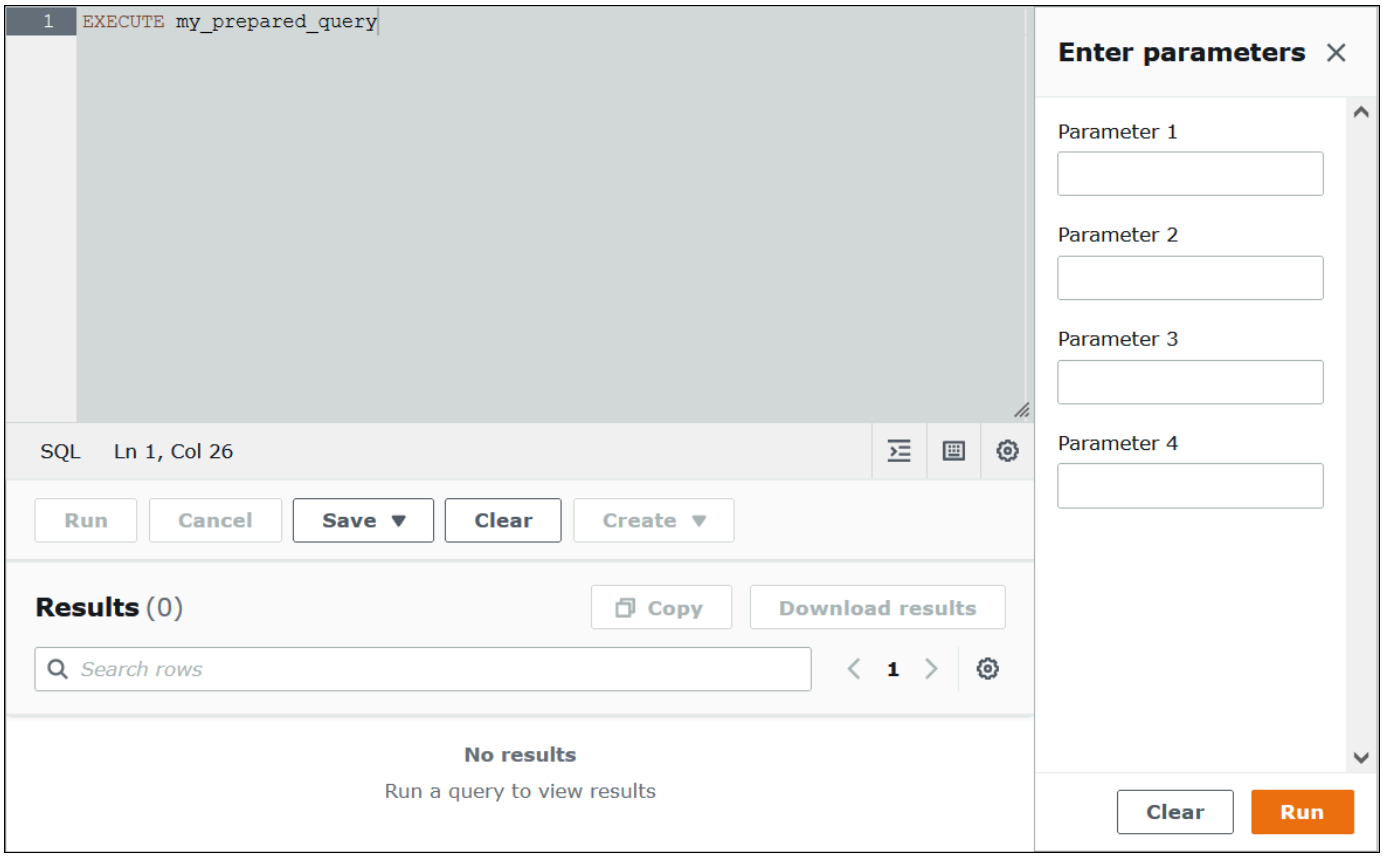
Executar instruções preparadas sem a cláusula USING no console do Athena

Se você executar uma instrução preparada existente com a sintaxe `EXECUTE` *`prepared_statement`* no editor de consultas, o Athena abrirá a caixa de diálogo Enter parameters (Inserir parâmetros), para que seja possível inserir os valores que normalmente entrariam na cláusula `USING` da instrução `EXECUTE ... USING`.

Para executar uma instrução preparada usando a caixa de diálogo Enter parameters (Inserir parâmetros)

1. No editor de consultas, em vez de usar a sintaxe `EXECUTE prepared_statement USING` *`value1`*`,` *`value2`* `...`, use a sintaxe `EXECUTE` *`prepared_statement`*.

2. Escolha Executar. A caixa de diálogo Enter parameters (Inserir parâmetros) é exibida.



3. Insira os valores em ordem na caixa de diálogo Execution parameters (Parâmetros de execução). Como o texto original da consulta não está visível, você deve se lembrar do significado de cada parâmetro posicional ou ter a instrução preparada disponível para referência.
4. Escolha Executar.

Criar instruções preparadas com o uso da AWS CLI

Para usar a AWS CLI para criar uma instrução preparada, é possível utilizar um dos seguintes comandos `athena`:

- Use o comando `create-prepared-statement` e forneça uma instrução de consulta que tenha parâmetros de execução.
- Use o comando `start-query-execution` e forneça uma string de consulta que use a sintaxe `PREPARE`.

Usar create-prepared-statement

Em um comando `create-prepared-statement`, defina o texto da consulta no argumento `query-statement`, como no exemplo a seguir.

```
aws athena create-prepared-statement
--statement-name PreparedStatement1
--query-statement "SELECT * FROM table WHERE x = ?"
--work-group athena-engine-v2
```

Usar start-query-execution e a sintaxe PREPARE

Use o comando `start-query-execution`. Coloque a instrução `PREPARE` no argumento `query-string`, como no exemplo a seguir:

```
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/..."}'
```

Executar instruções preparadas com o uso da AWS CLI

Para executar uma declaração preparada com a AWS CLI, forneça valores para os parâmetros com um dos seguintes métodos:

- Use o argumento `execution-parameters`.
- Use a sintaxe SQL `EXECUTE ... USING` no argumento `query-string`.

Usar o argumento execution-parameters

Nessa abordagem, você usa o comando `start-query-execution` e fornece o nome de uma declaração preparada existente no argumento `query-string`. Em seguida, no argumento `execution-parameters`, forneça os valores para os parâmetros de execução. O exemplo a seguir mostra esse método.

```
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"
```

Usar a sintaxe SQL EXECUTE... USING

Para executar uma instrução preparada existente usando a sintaxe `EXECUTE ... USING`, use o comando `start-query-execution` e coloque o nome da instrução preparada e os valores do parâmetro no argumento `query-string`, como no exemplo a seguir:

```
aws athena start-query-execution
--query-string "EXECUTE PreparedStatement1 USING 1"
--query-execution-context '{"Database": "default"}'
--result-configuration '{"OutputLocation": "s3://DOC-EXAMPLE-BUCKET/..."}'
```

Listar instruções preparadas

Para listar as instruções preparadas para um grupo de trabalho específico, use o comando list-prepared-statements da AWS CLI do Athena ou a ação de API ListPreparedStatements do Athena. O parâmetro `--work-group` é obrigatório.

```
aws athena list-prepared-statements --work-group primary
```

Recursos adicionais do

Veja as seguintes postagens relacionadas no blog de Big Data da AWS.

- Improve reusability and security using Amazon Athena parameterized queries
- Use Amazon Athena parameterized queries to provide data as a service

Usar o otimizador baseado em custos

Você pode usar o atributo otimizador baseado em custos (CBO) no Athena SQL para otimizar as consultas. Opcionalmente, você pode solicitar que o Athena colete estatísticas no nível da tabela ou da coluna para uma das tabelas do AWS Glue. Se todas as tabelas da consulta tiverem estatísticas, o Athena usará as estatísticas para criar um plano de execução que ele determina que terá a melhor performance. O otimizador de consultas calcula planos alternativos com base em um modelo estatístico e depois seleciona o que tem maior probabilidade de executar a consulta em menos tempo.

As estatísticas das tabelas do AWS Glue são coletadas e armazenadas no AWS Glue Data Catalog, e disponibilizadas ao Athena para melhorar o planejamento e a execução das consultas. Essas estatísticas são coletadas no nível da coluna, como número de valores distintos, número de valores

nulos, máximos e mínimos em tipos de arquivo como Parquet, ORC, JSON, ION, CSV e XML. O Amazon Athena usa essas estatísticas para otimizar as consultas aplicando os filtros mais restritivos no processamento das consultas assim que possível. Essa filtragem limita o uso da memória e o número de registros que devem ser lidos para fornecer os resultados das consultas.

Em conjunto com o CBO, o Athena usa um atributo denominado otimizador baseado em regras (RBO). O RBO aplica mecanicamente regras que devem melhorar a performance das consultas. O RBO geralmente é útil porque suas transformações visam simplificar o plano de consulta. Porém, como o RBO não realiza cálculos de custos nem comparações de planos, consultas mais complicadas tornam difícil para o RBO criar um plano ideal.

Por isso, o Athena usa ambos o RBO e o CBO para otimizar as consultas. Depois que o Athena identifica as oportunidades de melhorar a execução das consultas, ele cria um plano ideal. Para obter mais informações sobre detalhes do plano de execução, consulte Visualizar planos de execução para consultas SQL. Para ver uma discussão detalhada sobre como o CBO funciona, consulte a postagem sobre o otimizador baseado em custos no blog AWS Big Data.

Para gerar estatísticas para tabelas do AWS Glue Catalog, você pode usar o console do Athena, o console do AWS Glue ou as APIs do AWS Glue. Como o Athena é integrado ao AWS Glue Catalog, você obtém automaticamente os aprimoramentos de performance das consultas correspondentes quando executa consultas no Amazon Athena.

Considerações e limitações

- Tipos de tabela: atualmente, o atributo CBO no Athena só é compatível com as tabelas do Hive que estão no AWS Glue Data Catalog.
- Athena for Spark: o atributo CBO não está disponível no Athena for Spark.
- Preços: para obter informações sobre preços, visite a página de preços do AWS Glue.

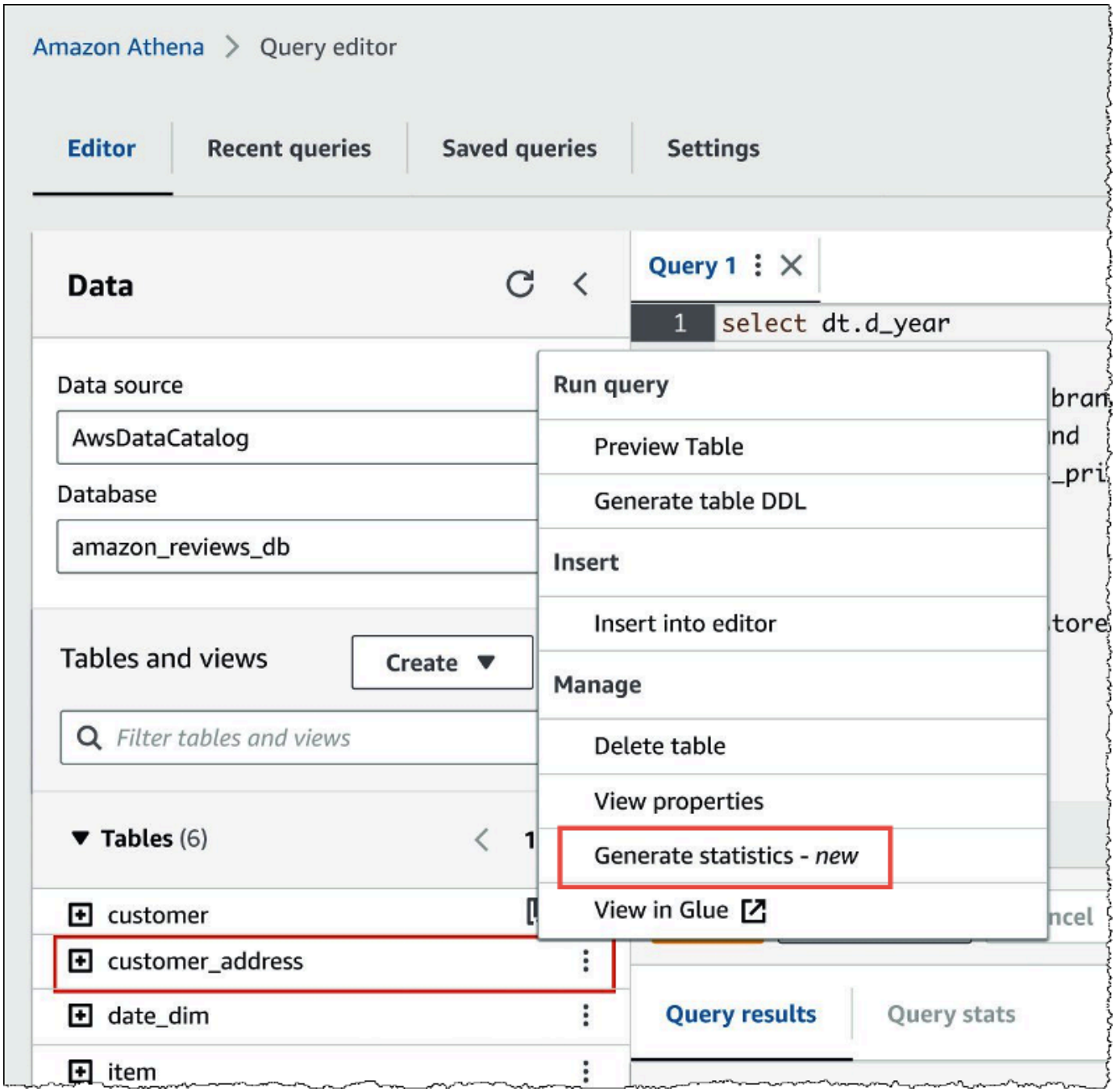
Gerar estatísticas de tabela usando o console do Athena

Esta seção descreve como usar o console do Athena para gerar estatísticas no nível da tabela ou da coluna para uma tabela no AWS Glue. Para obter informações sobre o uso do AWS Glue para gerar estatísticas de tabelas, consulte Working with column statistics no AWS Glue Developer Guide.

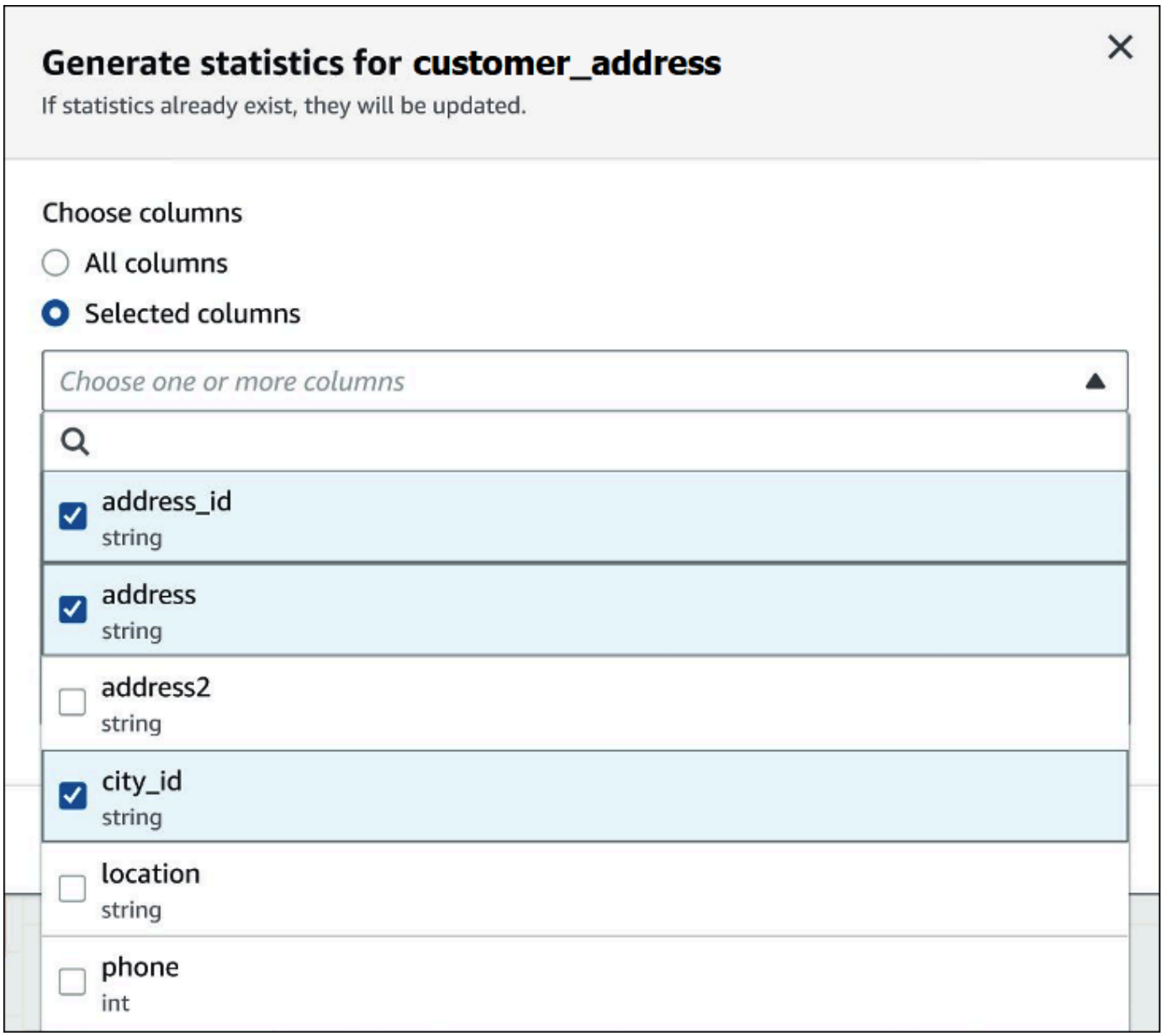
Gerar estatísticas para uma tabela usando o console do Athena

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.

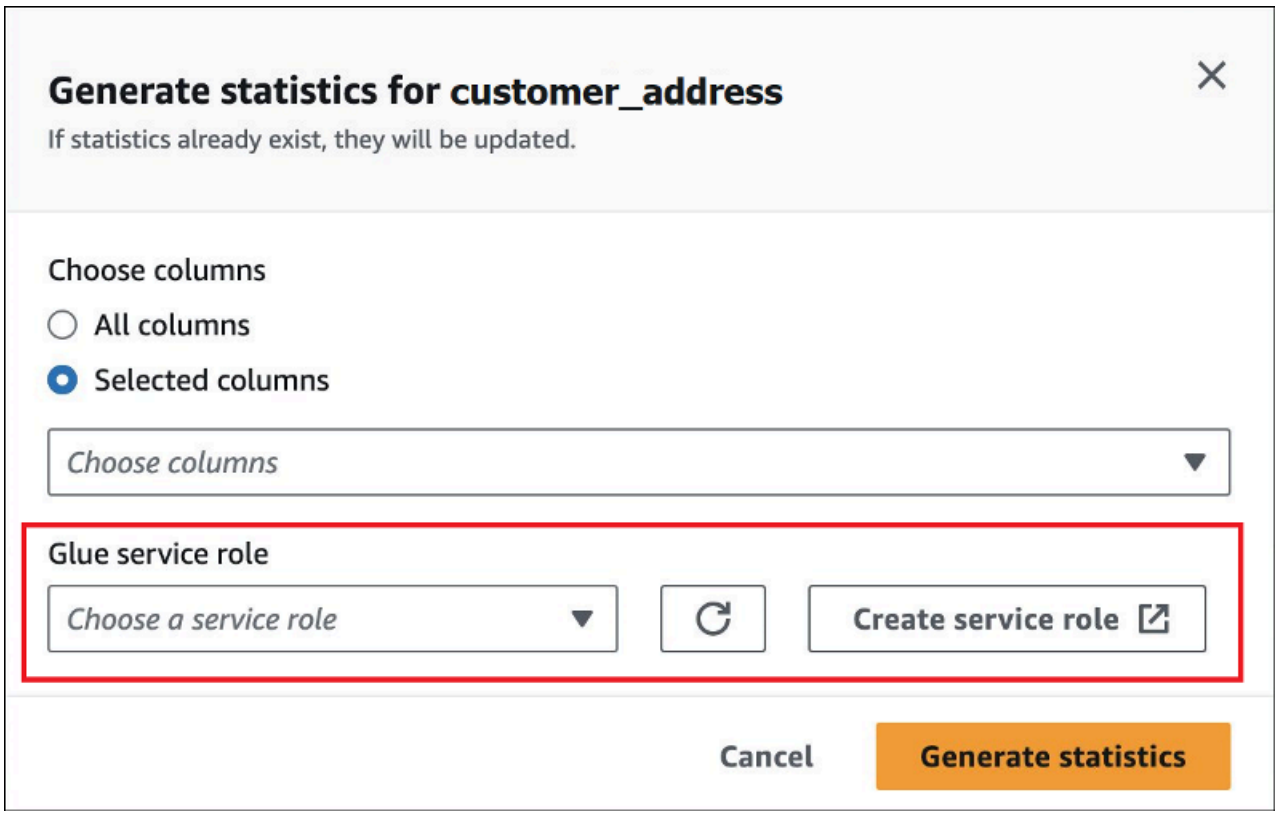
2. Na lista Tabelas do editor de consultas do Athena, escolha os três pontos verticais para a tabela que você deseja e depois escolha Gerar estatísticas.



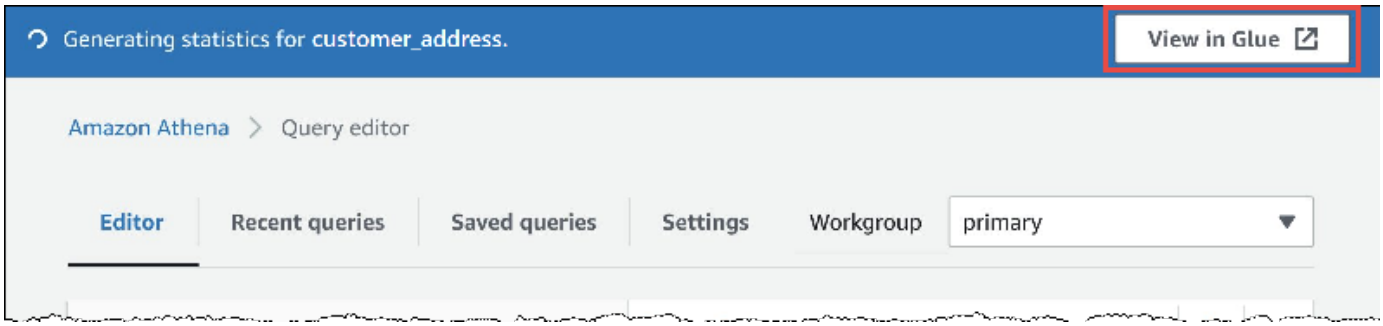
3. Na caixa de diálogo Gerar estatísticas, escolha Todas as colunas para gerar estatísticas para todas as colunas da tabela ou escolha Colunas selecionadas para selecionar colunas específicas. Todas as colunas é a configuração padrão.



4. Para perfil do serviço do AWS Glue, crie ou selecione um perfil de serviço existente para dar permissão ao AWS Glue para gerar estatísticas. O perfil de serviço do AWS Glue também exige permissões de `S3:GetObject` para o bucket do Amazon S3 que contém os dados da tabela.



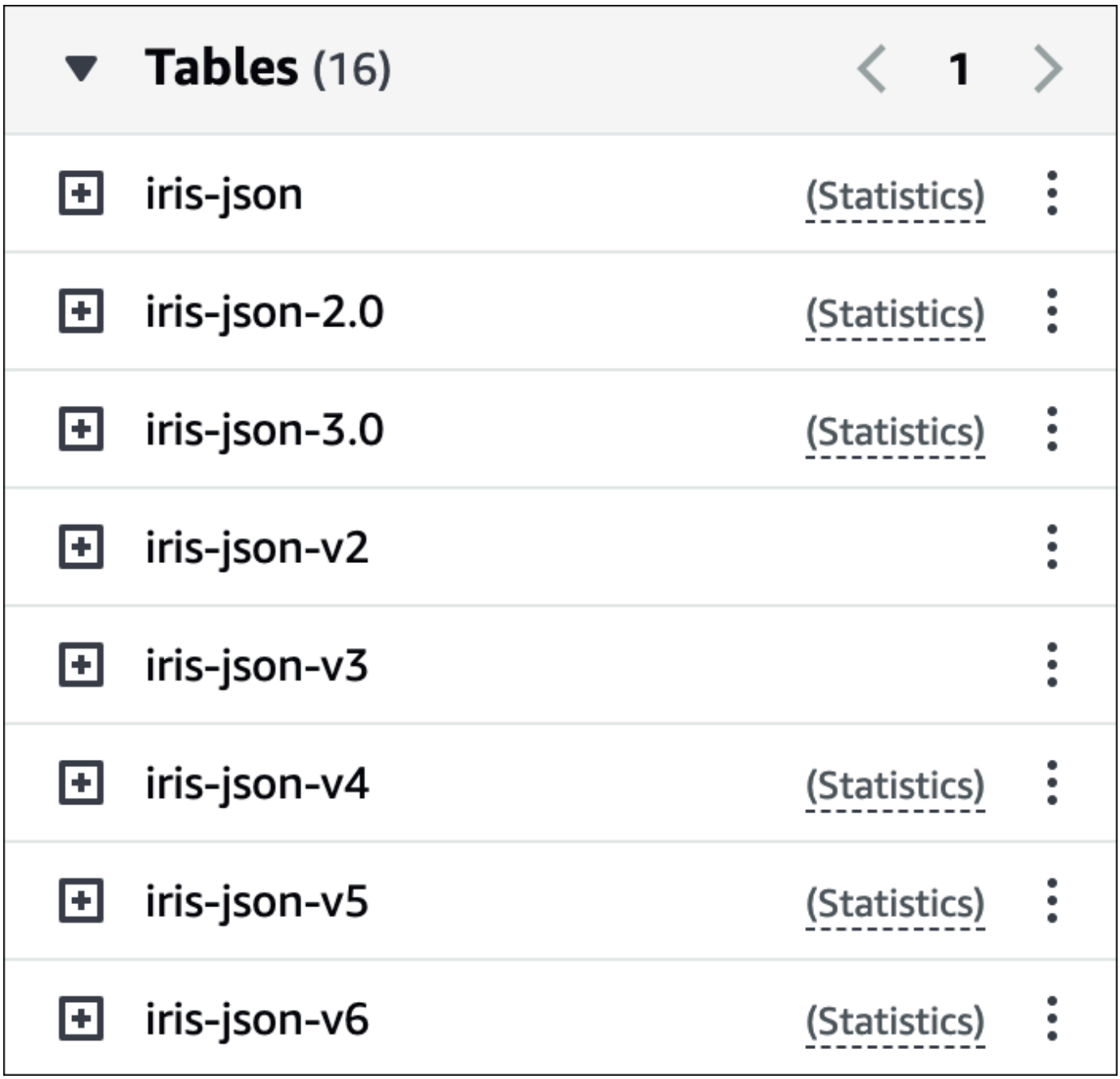
5. Escolha Gerar estatísticas. Um banner de notificação Gerando estatísticas para a ***`table_name`*** mostra o status da tarefa.



6. Para visualizar detalhes no console do AWS Glue, escolha Visualizar no Glue.

   Para obter informações sobre a visualização de estatísticas no console do AWS Glue, consulte Viewing column statistics no AWS Glue Developer Guide.

7. Depois que as estatísticas são geradas, as tabelas e colunas que têm estatísticas trazem a palavra Estatísticas entre parênteses, como na imagem a seguir.



Agora, quando você executar consultas, o Athena realizará a otimização baseada em custos nas tabelas e colunas para as quais as estatísticas foram geradas.

Recursos adicionais do

Para mais informações, consulte o recurso a seguir.

Consulta de dados do S3 Express One Zone

A classe de armazenamento Amazon S3 Express One Zone é uma classe de armazenamento do Amazon S3 com alta performance que fornece tempos de resposta abaixo de dez milissegundos. Dessa forma, essa classe é útil para aplicações que acessam dados frequentemente com centenas de milhares de solicitações por segundo.

A classe S3 Express One Zone replica e armazena dados na mesma zona de disponibilidade para otimizar a velocidade e os custos. Isso difere das classes de armazenamento regionais do Amazon S3, que replicam automaticamente os dados em, no mínimo, três zonas de disponibilidade da AWS em uma Região da AWS.

Para obter mais informações, consulte What is S3 Express One Zone? no Guia do usuário do Amazon S3.

Pré-requisitos

Confirme se as seguintes condições foram atendidas antes de começar a usar:

- Versão 3 do mecanismo Athena: para usar a classe S3 Express One Zone com o Athena SQL, o grupo de trabalho deve estar configurado para usar a versão 3 do mecanismo Athena.
- Permissões do S3 Express One Zone: quando a classe S3 Express One Zone chama uma ação como `GET`, `LIST` ou `PUT` em um objeto do Amazon S3, a classe de armazenamento chama `CreateSession` em seu nome. Por esse motivo, a política do IAM deve permitir a ação `s3express:CreateSession`, que possibilita ao Athena invocar a operação de API correspondente.

Considerações e limitações

Ao consultar a classe S3 Express One Zone com o Athena, considere os pontos apresentados a seguir.

- Os buckets da classe S3 Express One Zone oferecem suporte somente à criptografia `SSE_S3`. Os resultados da consulta do Athena são gravados usando a criptografia `SSE_S3`, independentemente da opção escolhida por você nas configurações do grupo de trabalho para criptografar os resultados da consulta. Essa limitação inclui todos os cenários em que o Athena grava dados em buckets da classe S3 Express One Zone, incluindo instruções `CREATE TABLE AS` (CTAS) e `INSERT INTO`.

- Não há suporte para o crawler do AWS Glue para a criação de tabelas em dados do S3 Express One Zone.
- Não há suporte para a instrução `MSCK REPAIR TABLE`. Como solução alternativa, use ALTER TABLE ADD PARTITION.
- Não há suporte ou o suporte é limitado para os formatos de arquivos e de tabelas apresentados a seguir. Se os formatos não estiverem listados, mas forem compatíveis com o Athena (como Parquet, ORC e JSON), eles também terão suporte para uso com o armazenamento do S3 Express One Zone.

Formato de arquivo ou de tabela	Limitação
Apache Avro	Não suportado
Logs do CloudTrail	Não suportado
Apache Hudi	Não suportado
Amazon Ion	Não suportado
Logs do Logstash	Não suportado
Logs do Apache WebServer	Não suportado
Delta Lake	Não há suporte para DDL. Para obter informações sobre como criar uma tabela do Delta Lake usando um esquema fictício, consulte Sincronização de metadados do Delta Lake. Há suporte para consultas `SELECT` na tabela.

Conceitos básicos

Consultar dados da classe S3 Express One Zone com o Athena é simples. Para começar a usar, siga o procedimento apresentado a seguir.

Como usar o Athena SQL para consultar dados da classe S3 Express One Zone

1. Faça a transição dos seus dados para o armazenamento do S3 Express One Zone. Para obter mais informações, consulte Configurar a classe de armazenamento de um objeto no Guia do usuário do Amazon S3.
2. Use uma instrução CREATE TABLE no Athena para catalogar seus dados no AWS Glue Data Catalog. Para obter informações sobre como criar tabelas no Athena, consulte Criar tabelas no Athena e a instrução CREATE TABLE.
3. (Opcional) Configure a localização do resultado da consulta do grupo de trabalho do Athena para usar um bucket de diretório do Amazon S3. Os buckets de diretório do Amazon S3 têm uma performance aprimorada quando comparados aos buckets gerais e são projetados para workloads ou aplicações críticas à performance que requerem latência consistente abaixo de dez milissegundos. Para obter mais informações, consulte Directory buckets overview no Guia do usuário do Amazon S3.

Consultar objetos restaurados do Amazon S3 Glacier

É possível usar o Athena para consultar objetos restaurados das classes de armazenamento do Amazon S3 S3 Glacier Flexible Retrieval (antigo Glacier) e S3 Glacier Deep Archive. É necessário habilitar esse recurso com base em tabelas. Se você não habilitar o atributo em uma tabela antes de executar a consulta, o Athena ignorará todos os objetos do S3 Glacier Flexible Retrieval e do S3 Glacier Deep Archive da tabela durante a execução da consulta.

Condições e limitações

- Há suporte para a consulta de objetos restaurados do Amazon S3 Glacier somente no mecanismo do Athena versão 3.
- O atributo é compatível somente com tabelas do Apache Hive.
- É necessário restaurar seus objetos antes de consultar os dados; o Athena não restaura objetos para você.

Configurar uma tabela para usar objetos restaurados

Para configurar sua tabela do Athena de modo a incluir objetos restaurados em suas consultas, é necessário definir a propriedade de tabela `read_restored_glacier_objects` como `true`. Para

fazer isso, você pode usar o editor de consultas do Athena ou o console do AWS Glue. Você também pode usar a CLI do AWS Glue, a API do AWS Glue ou o SDK do AWS Glue.

Usar o editor de consultas do Athena

No Athena, você pode usar o comando ALTER TABLE SET TBLPROPERTIES para definir a propriedade da tabela, como no exemplo a seguir.

```
ALTER TABLE table_name SET TBLPROPERTIES ('read_restored_glacier_objects' = 'true')
```

Usar o console de AWS Glue

Edite a tabela no console do AWS Glue e realize as seguintes etapas para adicionar a propriedade de tabela `read_restored_glacier_objects`.

Para configurar as propriedades da tabela no console do AWS Glue

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. Execute um destes procedimentos:
   - Escolha Ir para catálogo de dados.
   - No painel de navegação, escolha Tabelas do catálogo de dados.
3. Na página Tabelas, na lista de tabelas, escolha o link para a tabela que você deseja editar.
4. Selecione Actions (Ações), Edit (Editar).
5. Na página Editar tabela, na seção Propriedades da tabela, adicione o par de chave-valor a seguir.
   - Em Chave, adicione `read_restored_glacier_objects`.
   - Em Valor, insira `true`.
6. Escolha Salvar.

Usando a AWS CLI

Na AWS CLI, você pode usar o comando update-table do AWS Glue e seu argumento `--table-input` para redefinir a tabela e, ao fazer isso, adicionar a propriedade `read_restored_glacier_objects`. No argumento `--table-input`, use a estrutura `Parameters` para especificar a propriedade `read_restored_glacier_objects` e

o valor de `true`. O argumento para `--table-input` não deve ter espaços e deve usar barras invertidas como escape das aspas duplas. No exemplo a seguir, substitua *`my_database`* e *`my_table`* pelos nomes de seu banco de dados e tabela.

```
aws glue update-table \
   --database-name my_database \
   --table-input={\"Name\":\"my_table\",\"Parameters\":{\"read_restored_glacier_objects
\":\"true\"}}
```

> **Important**
>
> O comando `update-table` do AWS Glue funciona no modo de substituição, o que significa que ele substitui a definição da tabela existente pela nova definição especificada pelo parâmetro `table-input`. Por esse motivo, não se esqueça de especificar também todos os campos que deseja que estejam em sua tabela no parâmetro `table-input` ao adicionar a propriedade `read_restored_glacier_objects`.

Tratamento de atualizações do esquema

Esta seção apresenta orientações de como processar as atualizações de esquema nos vários formatos de dados. O Athena é um mecanismo de consulta “schema-on-read”. Isso significa que, quando você cria uma tabela no Athena, ele aplica esquemas durante a leitura dos dados. Ela não altera nem reescrever os dados subjacentes.

Se você antecipar alterações nos esquemas de tabela, considere criá-los em um formato de dados adequado para suas necessidades. Seus objetivos são reutilizar as consultas existentes do Athena nos esquemas em desenvolvimento e evitar erros de incompatibilidade de esquemas ao consultar tabelas com partições.

Para atingir essas metas, escolha um formato de dados da tabela com base na tabela no seguinte tópico.

Tópicos

- Resumo: atualizações e formatos de dados no Athena
- Acesso ao índice em ORC e Parquet
- Tipos de atualizações
- Atualizações em tabelas com partições

Resumo: atualizações e formatos de dados no Athena

A tabela a seguir resume os formatos de armazenamento físico de dados e os manuseios do esquema com suporte. Use esta tabela para ajudar a escolher o formato que permitirá que você continue usando as consultas do Athena mesmo que seus esquemas sejam alterados ao longo do tempo.

Nessa tabela, observe que o Parquet e ORC são formatos de coluna com diferentes métodos de acesso padrão à coluna. Por padrão, o Parquet acessa colunas por nome e o ORC por índice (valor ordinal). Portanto, o Athena oferece uma propriedade SerDe definida no momento da criação de uma tabela para alternar o método de acesso padrão a colunas, o que permite maior flexibilidade com o desenvolvimento do esquema.

Para o Parquet, a propriedade `parquet.column.index.access` pode ser definida como `true`, que define que o método de acesso à coluna usará o número ordinal da coluna. Definir essa propriedade como `false` fará com que o método de acesso à coluna use o nome da coluna. Da mesma forma, para ORC, use a propriedade `orc.column.index.access` para controlar o método de acesso à coluna. Para ter mais informações, consulte Acesso ao índice em ORC e Parquet.

Os formatos CSV e TSV permitem que você faça todos os manuseios do esquema, exceto a reorganização de colunas ou a adição de colunas ao início da tabela. Por exemplo, se a evolução do seu esquema requer apenas a renomeação de colunas, mas não a remoção delas, você pode optar por criar suas tabelas nos formatos CSV ou TSV. Se você precisar remover colunas, não use os formatos CSV ou TSV. Em vez disso, use qualquer um dos outros formatos compatíveis, de preferência um formato de coluna, como Parquet ou ORC.

Atualizações de esquema e formatos de dados no Athena

Tipo esperado de atualização de esquema	Resumo	CSV (com e sem cabeçalhos) e TSV	JSO	AVR	PARQUE leitura por nome (padrão)	PARQUE leitura por índice	ORC: leitura por índice (padrão)	ORC: leitura por nome
Renomear colunas	Armazene seus dados em CSV e TSV, ou em ORC e Parquet se	Y	N	N	N	Y	Y	N

Tipo esperado de atualização de esquema	Resumo	CSV (com e sem cabeçalhos) e TSV	JSO	AVR	PARQUE leitura por nome (padrão)	PARQUE leitura por índice	ORC: leitura por índice (padrão)	ORC: leitura por nome
	eles são lidos por índice.							
Adicionar colunas no início ou no meio da tabela	Armazene seus dados em JSON e AVRO, ou em Parquet e ORC se eles são lidos por nome. Não use CSV e TSV.	N	Y	Y	Y	N	N	Y
Adicionar colunas no final da tabela	Armazene seus dados em CSV ou TSV, JSON, AVRO, ORC ou Parquet.	Y	Y	Y	Y	Y	Y	Y
Remover colunas	Armazene seus dados em JSON e AVRO, ou em Parquet e ORC se eles são lidos por nome. Não use CSV e TSV.	N	Y	Y	Y	N	N	Y
Reclassificar colunas	Armazene seus dados em AVRO, JSON ou em Parquet e ORC se eles são lidos por nome.	N	Y	Y	Y	N	N	Y

Tipo esperado de atualização de esquema	Resumo	CSV (com e sem cabeçalhos) e TSV	JSO	AVR	PARQUE leitura por nome (padrão)	PARQUE leitura por índice	ORC: leitura por índice (padrão)	ORC: leitura por nome
Alterar o tipo de dados de uma coluna	Armazene seus dados em qualquer formato, mas faça o teste da consulta no Athena para garantir que os tipos de dados sejam compatíveis. No Parquet e no ORC, a alteração de um tipo de dados funciona apenas para tabelas particionadas.	Y	Y	Y	Y	Y	Y	Y

Acesso ao índice em ORC e Parquet

PARQUET e ORC são formatos de coluna para armazenamento físico de dados que podem ser lidos por índice ou por nome. O armazenamento dos dados em qualquer um desses formatos permite que você execute todas as operações nos esquemas e execute as consultas do Athena sem erros de incompatibilidade de esquemas.

- Por padrão, o Athena lê o ORC por índice, conforme definido em `SERDEPROPERTIES ( 'orc.column.index.access'='true')`. Para ter mais informações, consulte ORC: leitura por índice.

- O Athena lê Parquet por nome, por padrão, conforme definido em `SERDEPROPERTIES ( 'parquet.column.index.access'='false')`. Para ter mais informações, consulte Parquet: leitura por nome.

Uma vez que essas leituras são padrão, a especificação das propriedades SerDe em suas consultas `CREATE TABLE` é opcional; elas são usadas implicitamente. Quando são usadas, elas permitem que você execute algumas operações de atualização de esquema enquanto impedem outras operações semelhantes. Para habilitar essas operações, execute outra consulta `CREATE TABLE` e altere as configurações SerDe.

> **Note**
>
> As propriedades SerDe não são propagadas automaticamente para cada partição. Use instruções `ALTER TABLE ADD PARTITION` para definir as propriedades SerDe para cada partição. Para automatizar esse processo, escreva um script que execute instruções `ALTER TABLE ADD PARTITION`.

As seções a seguir descrevem esses casos em detalhes.

ORC: leitura por índice

Uma tabela no formato ORC é lida por índice, por padrão. Isso é definido pela seguinte sintaxe:

```
WITH SERDEPROPERTIES (
  'orc.column.index.access'='true')
```

A leitura por índice permite renomear colunas. No entanto, não será mais possível remover colunas nem as adicionar no meio da tabela.

Para fazer com que o ORC seja lido por nome, o que permitirá que você adicione colunas no meio da tabela ou remova as colunas em ORC, defina a propriedade `orc.column.index.access` do SerDe como `false` na instrução `CREATE TABLE`. Com essa configuração, não será mais possível renomear colunas.

> **Note**
>
> No mecanismo Athena versão 2, quando as tabelas ORC são definidas para serem lidas por nome, o Athena exige que todos os nomes de coluna dos arquivos ORC estejam em

letras minúsculas. Como o Apache Spark não usa nomes de campo em letras minúsculas ao gerar arquivos ORC, o Athena talvez não consiga ler os dados que são gerados. A solução alternativa é renomear as colunas usando letras minúsculas ou usar o mecanismo Athena versão 3.

O exemplo a seguir ilustra como alterar o ORC para que ele seja lido por nome:

```
CREATE EXTERNAL TABLE orders_orc_read_by_name (
   `o_comment` string,
   `o_orderkey` int,
   `o_custkey` int,
   `o_orderpriority` string,
   `o_orderstatus` string,
   `o_clerk` string,
   `o_shippriority` int,
   `o_orderdate` string
)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.orc.OrcSerde'
WITH SERDEPROPERTIES (
  'orc.column.index.access'='false')
STORED AS INPUTFORMAT
  'org.apache.hadoop.hive.ql.io.orc.OrcInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat'
LOCATION 's3://DOC-EXAMPLE-BUCKET/orders_orc/';
```

Parquet: leitura por nome

Uma tabela no formato Parquet é lida por nome, por padrão. Isso é definido pela seguinte sintaxe:

```
WITH SERDEPROPERTIES (
  'parquet.column.index.access'='false')
```

A leitura por nome permite que você remova colunas ou as adicione no meio da tabela. No entanto, não será mais possível renomeá-las.

Para fazer com que o Parquet seja lido por índice, o que permitirá que você renomeie colunas, é necessário criar uma tabela com a propriedade `parquet.column.index.access` do SerDe e defini-la como `true`.

Tipos de atualizações

Este tópico descreve algumas alterações que você pode fazer no esquema em instruções `CREATE TABLE` sem alterar os dados de fato. Analisamos cada tipo de atualização de esquema e especificamos os formatos de dados que os permitem no Athena. Para atualizar um esquema, em alguns casos é possível usar um comando `ALTER TABLE`, mas em outros casos você não modifica uma tabela existente. Em vez disso, você cria uma tabela com um novo nome que modifica o esquema usado na instrução `CREATE TABLE` original.

- Adicionar colunas no início ou no meio da tabela
- Adicionar colunas no final da tabela
- Remover colunas
- Renomear colunas
- Reclassificar colunas
- Alterar um tipo de dado da coluna

Dependendo de como você espera que seus esquemas evoluam, para continuar usando as consultas do Athena, escolha um formato de dados compatível.

Considere uma aplicação que lê informações de pedidos em uma tabela de `orders` que existe em dois formatos: CSV e Parquet.

O exemplo a seguir cria uma tabela em 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/';
```

O exemplo a seguir cria a mesma tabela em 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/';
```

Nas seções a seguir, analisamos como as atualizações nessas tabelas afetam as consultas do Athena.

Adicionar colunas no início ou no meio da tabela

A adição de colunas é uma das alterações do esquema mais frequentes. Por exemplo, você pode adicionar uma nova coluna para enriquecer a tabela com novos dados. Ou você poderá adicionar uma nova coluna se a origem para uma coluna existente for alterada e manter a versão anterior deste coluna para ajustar os aplicativos que dependem delas.

Para adicionar colunas no início ou no meio da tabela e continuar executando consultas em tabelas existentes, use os formatos AVRO e JSON, bem como Parquet e ORC, se a propriedade do SerDe estiver definida para leitura por nome. Para ter mais informações, consulte Acesso ao índice em ORC e Parquet.

Não adicione colunas no início ou no meio da tabela em CSV e TSV, pois esses formatos dependem da classificação. Se uma coluna for adicionada em um desses casos, ocorrerá um erro de incompatibilidade de esquema quando o esquema de partições for alterado.

O exemplo a seguir cria uma nova tabela que adiciona uma coluna `o_comment` no meio de uma tabela baseada em dados 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/';
```

Adicionar colunas no final da tabela

Se você criar tabelas em qualquer um dos formatos compatíveis com o Athena, como Parquet, ORC, Avro, JSON, CSV e TSV, poderá usar a instrução `ALTER TABLE ADD COLUMNS` para adicionar colunas após as colunas existentes, mas antes das colunas de partição.

O exemplo a seguir adiciona uma coluna `comment` no final da tabela `orders_parquet` antes de qualquer coluna de partição:

```
ALTER TABLE orders_parquet ADD COLUMNS (comment string)
```

> **Note**
>
> Para ver uma nova coluna de tabela no editor de consultas do Athena depois de executar `ALTER TABLE ADD COLUMNS`, atualize manualmente a lista de tabelas no editor e expanda a tabela outra vez.

Remover colunas

Talvez você precise remover colunas de tabelas se elas não contiverem dados ou restringir o acesso aos dados contidos nelas.

- Você pode remover colunas de tabelas em JSON, Avro e em ORC e Parquet se elas forem lidas por nome. Para ter mais informações, consulte Acesso ao índice em ORC e Parquet.
- Não é recomendável remover colunas das tabelas em CSV e TSV se você deseja reter as tabelas que criou no Athena. A remoção de uma coluna rompe o esquema e requer que você recrie a tabela sem a coluna removida.

Neste exemplo, remova uma coluna `` `totalprice` `` de uma tabela em Parquet e execute uma consulta. No Athena, por padrão o Parquet é lido por nome. É por esse motivo que omitimos a configuração SERDEPROPERTIES que especifica a leitura por nome. Observe que a consulta a seguir é bem-sucedida, mesmo que você altere o esquema:

```
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/';
```

Renomear colunas

Talvez você queira renomear colunas em suas tabelas para corrigir ortografia, tornar os nomes das colunas mais descritivos ou reutilizar uma coluna existente para evitar a reclassificação dela.

Você pode renomear colunas se armazenar seus dados em CSV e TSV, ou em Parquet e ORC, que são configurados para leitura por índice. Para ter mais informações, consulte Acesso ao índice em ORC e Parquet.

O Athena lê os dados em CSV e TSV na ordem das colunas no esquema e os retorna na mesma ordem. Ele não usa os nomes de coluna para mapear os dados para uma coluna. É por esse motivo que você pode renomear as colunas em CSV ou TSV sem interromper as consultas do Athena.

Uma estratégia para renomear colunas é criar uma tabela com base nos mesmos dados subjacentes, mas usando novos nomes de coluna. O exemplo a seguir cria uma tabela `orders_parquet` chamada `orders_parquet_column_renamed`. O exemplo altera o nome da coluna `` `o_totalprice` `` para `` `o_total_price` `` e executa uma consulta no 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/';
```

No caso da tabela do Parquet, a consulta a seguir é executada. No entanto, a coluna renomeada não exibe dados, porque ela estava sendo acessada por nome (um padrão no Parquet) em vez de por índice:

```
SELECT *
FROM orders_parquet_column_renamed;
```

Uma consulta com uma tabela em CSV tem aparência semelhante:

```
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/';
```

No caso da tabela CSV, a consulta a seguir é executada, e os dados são exibidos em todas as colunas, incluindo aquela que foi renomeada:

```
SELECT *
FROM orders_csv_column_renamed;
```

Reclassificar colunas

Você pode reordenar colunas apenas em tabelas com dados em formatos de leitura por nome, como JSON ou Parquet, que leem por nome, por padrão. Você também pode fazer com que o ORC leia por nome, se necessário. Para ter mais informações, consulte Acesso ao índice em ORC e Parquet.

O exemplo a seguir cria uma tabela com as colunas em uma ordem diferente:

```
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/';
```

Alterar um tipo de dado da coluna

Convém usar outro tipo de coluna quando o tipo existente não puder mais conter a quantidade de informações necessárias. Por exemplo, os valores de uma coluna de ID podem exceder o tamanho do tipo de dados `INT` e exigir o uso do tipo de dados `BIGINT`.

Ao planejar usar um tipo de dados diferente para uma coluna, leve em consideração os seguintes pontos:

- Na maioria dos casos, você não pode alterar diretamente o tipo de dados de uma coluna. Em vez disso, você recria a tabela do Athena e define a coluna com o novo tipo de dados.
- Apenas certos tipos de dados podem ser lidos como outros tipos de dados. Consulte a tabela nesta seção para ver os tipos de dados que podem ser tratados dessa forma.
- Para dados em formato Parquet e ORC, você não pode usar um tipo de dados diferente para uma coluna se a tabela não estiver particionada.
- Para tabelas particionadas no Parquet e no ORC, o tipo de coluna de uma partição pode ser diferente do tipo de coluna de outra partição, e o Athena aplicará `CAST` ao tipo desejado, se possível. Para ter mais informações, consulte Evitar erros de não correspondência de esquema para tabelas com partições.
- Para tabelas criadas usando o LazySimpleSerDe apenas, é possível usar a declaração `ALTER TABLE REPLACE COLUMNS` para substituir colunas existentes por um tipo de dados diferente, mas todas as colunas existentes que você deseja manter também devem ser redefinidas na declaração, caso contrário, elas serão excluídas. Para ter mais informações, consulte ALTER TABLE REPLACE COLUMNS.
- Para tabelas Apache Iceberg apenas, você pode usar a declaração ALTER TABLE CHANGE COLUMN para alterar o tipo de dados de uma coluna. A declaração `ALTER TABLE REPLACE`

`COLUMNS` não é suportada para tabelas Iceberg. Para ter mais informações, consulte Esquema de tabela Iceberg em evolução.

> **Important**
>
> É altamente recomendável testar e verificar suas consultas antes de executar as conversões de tipo de dados. Se o Athena não puder usar o tipo de dados de destino, a consulta `CREATE TABLE` poderá falhar.

A tabela a seguir lista os tipos de dados que podem ser tratados como outros tipos de dados:

Tipos de dados compatíveis

Tipos de dados originais	Tipos de dados de destino disponíveis
`STRING`	`BYTE`, `TINYINT`, `SMALLINT`, `INT`, `BIGINT`
`BYTE`	`TINYINT`, `SMALLINT`, `INT`, `BIGINT`
`TINYINT`	`SMALLINT`, `INT`, `BIGINT`
`SMALLINT`	`INT`, `BIGINT`
`INT`	`BIGINT`
`FLOAT`	`DOUBLE`

O exemplo a seguir usa a instrução `CREATE TABLE` da tabela `orders_json` original para criar uma nova tabela chamada `orders_json_bigint`. A nova tabela usa `BIGINT` em vez de `INT` como tipo de dados para a coluna `` `o_shippriority` ``.

```
CREATE EXTERNAL TABLE orders_json_bigint (
   `o_orderkey` int,
   `o_custkey` int,
   `o_orderstatus` string,
   `o_totalprice` double,
   `o_orderdate` string,
   `o_orderpriority` string,
   `o_clerk` string,
```

```
    `o_shippriority` BIGINT
)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 's3://DOC-EXAMPLE-BUCKET/orders_json';
```

A seguinte consulta é executada com êxito, semelhante à consulta SELECT original, antes que o tipo de dados seja alterado:

```
Select * from orders_json
LIMIT 10;
```

Atualizações em tabelas com partições

No Athena, uma tabela e suas partições devem usar os mesmos formatos de dados, mas os esquemas podem ser diferentes. Quando você cria uma nova partição, essa partição geralmente herda o esquema da tabela. Com o passar do tempo, os esquemas podem começar a ser diferentes. Os motivos para isso incluem:

- Se o esquema da tabela é alterado, os esquemas para as partições não são atualizados para permanecer em sincronia com o esquema da tabela.
- O Crawler do AWS Glue permite que você descubra dados em partições com esquemas diferentes. Isso significa que, se você criar uma tabela no Athena com o AWS Glue, depois que o crawler concluir o processamento, os esquemas da tabela e suas partições poderão ser diferentes.
- Se você adicionar partições diretamente usando uma API da AWS.

O Athena processará tabelas com partições com êxito se elas estiverem de acordo com as restrições a seguir. Se essas restrições não forem atendidas, o Athena emitirá um erro HIVE_PARTITION_SCHEMA_MISMATCH.

- Cada esquema de partição é compatível com o esquema da tabela.
- O formato de dados da tabela permite o tipo de atualização que você deseja executar: adicionar, excluir, reclassificar colunas ou alterar um tipo de dados da coluna.

  Por exemplo, para os formatos CSV e TSV, você pode renomear colunas, adicionar novas colunas no final da tabela e alterar o tipo de dados de uma coluna se os tipos forem compatíveis, mas você não pode remover colunas. Para outros formatos, você pode adicionar ou remover colunas ou alterar o tipo de dados de uma coluna para outro se os tipos forem compatíveis. Para obter informações, consulte Resumo: atualizações e formatos de dados no Athena.

Evitar erros de não correspondência de esquema para tabelas com partições

No início da execução da consulta, o Athena verifica o esquema da tabela confirmando se cada tipo de dados da coluna é compatível entre a tabela e a partição.

- Para os tipos de armazenamento de dados ORC e Parquet, o Athena usa os nomes das colunas para a verificação de esquema com base no nome da coluna. Isso elimina erros `HIVE_PARTITION_SCHEMA_MISMATCH` nas tabelas com partições nos tipos Parquet e ORC. (Isso será verdadeiro no ORC se a propriedade SerDe for definida para acessar o índice por nome: `orc.column.index.access=FALSE`. Por padrão, o Parquet lê o índice por nome).
- Para CSV, JSON e AVRO, o Athena usa uma verificação de esquema com base no índice. Isso significa que, se você encontrar um erro de incompatibilidade de esquema, deverá descartar a partição que está causando essa incompatibilidade e recriá-la, de modo que o Athena possa consultá-la sem falhas.

O Athena compara o esquema da tabela com os esquemas da partição. Se você criar uma tabela em CSV, JSON e AVRO no Athena com o crawler do AWS Glue, depois que o crawler concluir o processamento, os esquemas da tabela e suas partições poderão ser diferentes. Se houver incompatibilidade entre os esquemas da tabela e da partição, haverá falha nas consultas no Athena devido a um erro de verificação de esquema semelhante a este: 'crawler_test.click_avro' is declared as type 'string', but partition 'partition_0=2017-01-17' declared column 'col68' as type 'double'.

Uma solução comum para esses erros é eliminar a partição que está causando o erro e recriá-la. Para ter mais informações, consulte ALTER TABLE DROP PARTITION e ALTER TABLE ADD PARTITION.

Consultar matrizes

O Amazon Athena permite criar arrays, concatená-los, convertê-los em tipos de dados diferentes e filtrá-los, nivelá-los e classificá-los.

Tópicos

- Criar matrizes
- Concatenar strings e matrizes
- Converter tipos de dados de matriz
- Encontrar tamanhos
- Acessar elementos da matriz

- Nivelar matrizes aninhadas
- Criar matrizes com base em subconsultas
- Filtrar matrizes
- Classificar matrizes
- Usar funções de agregação com matrizes
- Converter matrizes em strings
- Usar matrizes para criar mapas
- Consultar matrizes de tipos complexos e com estruturas aninhadas

Criar matrizes

Para criar um literal de array no Athena, use a palavra-chave ARRAY seguida por colchetes `[ ]` e inclua os elementos do array separados por vírgulas.

Exemplos

Essa consulta cria uma matriz com quatro elementos.

```
SELECT ARRAY [1,2,3,4] AS items
```

Ela retorna:

```
+-----------+
| items     |
+-----------+
| [1,2,3,4] |
+-----------+
```

Essa consulta cria duas matrizes.

```
SELECT ARRAY[ ARRAY[1,2], ARRAY[3,4] ] AS items
```

Ela retorna:

```
+--------------------+
| items              |
+--------------------+
| [[1, 2], [3, 4]]   |
```

```
+--------------------+
```

Para criar uma matriz com base em colunas selecionadas de tipos compatíveis, use uma consulta, como neste exemplo:

```
WITH
dataset AS (
  SELECT 1 AS x, 2 AS y, 3 AS z
)
SELECT ARRAY [x,y,z] AS items FROM dataset
```

Essa consulta retorna:

```
+-----------+
| items     |
+-----------+
| [1,2,3]   |
+-----------+
```

No exemplo a seguir, duas matrizes são selecionadas e retornadas como uma mensagem de boas-vindas.

```
WITH
dataset AS (
  SELECT
    ARRAY ['hello', 'amazon', 'athena'] AS words,
    ARRAY ['hi', 'alexa'] AS alexa
)
SELECT ARRAY[words, alexa] AS welcome_msg
FROM dataset
```

Essa consulta retorna:

```
+----------------------------------------+
| welcome_msg                            |
+----------------------------------------+
| [[hello, amazon, athena], [hi, alexa]] |
+----------------------------------------+
```

Para criar uma matriz de pares chave/valor, use o operador `MAP` que utiliza uma matriz de chaves seguida de uma matriz de valores, como neste exemplo:

```
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
```

Essa consulta retorna:

```
+-----------------------------------------------------------------------------------------------------------
+
| people
               |
+-----------------------------------------------------------------------------------------------------------
+
| [{last=Smith, first=Bob, age=40}, {last=Doe, first=Jane, age=30}, {last=Smith,
 first=Billy, age=8}] |
+-----------------------------------------------------------------------------------------------------------
+
```

Concatenar strings e matrizes

Concatenar strings

Para concatenar duas strings, você pode usar o operador de barra dupla **||**, como no exemplo a seguir.

```
SELECT 'This' || ' is' || ' a' || ' test.' AS Concatenated_String
```

Essa consulta retorna:

#	Concatenated_String
1	This is a test.

Você pode usar a função `concat()` para obter o mesmo resultado.

```
SELECT concat('This', ' is', ' a', ' test.') AS Concatenated_String
```

Essa consulta retorna:

#	Concatenated_String
1	This is a test.

Você pode usar a função `concat_ws()` para concatenar strings com o separador especificado no primeiro argumento.

```
SELECT concat_ws(' ', 'This', 'is', 'a', 'test.') as Concatenated_String
```

Essa consulta retorna:

#	Concatenated_String
1	This is a test.

Para concatenar duas colunas do tipo de dados string usando um ponto, referencie as duas colunas usando aspas duplas e coloque o ponto entre aspas simples como uma string com codificação rígida. Se uma coluna não for do tipo de dados string, você poderá usar `CAST("column_name" as VARCHAR)` para converter a coluna primeiro.

```
SELECT "col1" || '.' || "col2" as Concatenated_String
FROM my_table
```

Essa consulta retorna:

#	Concatenated_String
1	*col1_string_value* .*col2_string_value*

Concatenar matrizes

Você pode usar as mesmas técnicas para concatenar arrays.

Para concatenar vários arrays, use o operador de barra dupla `||`.

```
SELECT ARRAY [4,5] || ARRAY[ ARRAY[1,2], ARRAY[3,4] ] AS items
```

Essa consulta retorna:

#	itens
1	[[4, 5], [1, 2], [3, 4]]

Para combinar vários arrays em um só, use o operador de barra dupla ou a função `concat()`.

```
WITH
dataset AS (
  SELECT
    ARRAY ['Hello', 'Amazon', 'Athena'] AS words,
    ARRAY ['Hi', 'Alexa'] AS alexa
)
SELECT concat(words, alexa) AS welcome_msg
FROM dataset
```

Essa consulta retorna:

#	welcome_msg
1	[Hello, Amazon, Athena, Hi, Alexa]

Para obter mais informações sobre o uso de `concat()` em outras funções de string, consulte String functions and operators (Funções e operadores de string) na documentação do Trino.

Converter tipos de dados de matriz

Para converter os dados dos arrays em tipos de dados compatíveis, use o operador `CAST`, como `CAST(value AS type)`. O Athena aceita todos os tipos de dados nativos do Presto.

```
SELECT
   ARRAY [CAST(4 AS VARCHAR), CAST(5 AS VARCHAR)]
AS items
```

Essa consulta retorna:

```
+-------+
```

```
| items |
+-------+
| [4,5] |
+-------+
```

Crie duas matrizes com elementos de par chave/valor, converta-as em JSON e concatene, como neste exemplo:

```
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
```

Essa consulta retorna:

```
+-------------------------------------------------+
| items                                           |
+-------------------------------------------------+
| [{"a1":1,"a2":2,"a3":3}, {"b1":4,"b2":5,"b3":6}] |
+-------------------------------------------------+
```

Encontrar tamanhos

A função `cardinality` retorna o tamanho de uma matriz, como neste exemplo:

```
SELECT cardinality(ARRAY[1,2,3,4]) AS item_count
```

Essa consulta retorna:

```
+------------+
| item_count |
+------------+
| 4          |
+------------+
```

Acessar elementos da matriz

Para acessar elementos de matriz, use o operador `[]`, com 1 especificando o primeiro elemento, 2 especificando o segundo elemento e assim por diante, como neste exemplo:

```
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
```

Essa consulta retorna:

```
+------------------------+
| item                   |
+------------------------+
| {"a1":1,"a2":2,"a3":3} |
+------------------------+
```

Para acessar os elementos de uma matriz em uma determinada posição (conhecida como a posição de índice), use a função `element_at()` e especifique o nome da matriz e a posição de índice:

- Se o índice for maior que 0, `element_at()` retornará o elemento especificado por você, contando do início ao fim da matriz. Ele se comporta como o operador `[]`.
- Se o índice for menor que 0, `element_at()` retornará o elemento, contando do fim ao início da matriz.

A consulta a seguir cria uma matriz `words` e seleciona o primeiro elemento `hello` dela como o `first_word`, o segundo elemento `amazon` (contagem a partir do final da matriz) como o `middle_word` e o terceiro elemento `athena` como o `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
```

Essa consulta retorna:

```
+----------------------------------------+
```

```
| first_word  | middle_word | last_word  |
+---------------------------------------+
| hello       | amazon      | athena     |
+---------------------------------------+
```

Nivelar matrizes aninhadas

Ao trabalhar com matrizes aninhadas, você normalmente precisa expandir elementos de matriz aninhados para uma única matriz ou expandir a matriz para várias linhas.

Exemplos

Para nivelar elementos de uma matriz aninhada em uma única matriz de valores, use a função `flatten`. Essa consulta retorna uma linha para cada elemento na matriz.

```
SELECT flatten(ARRAY[ ARRAY[1,2], ARRAY[3,4] ]) AS items
```

Essa consulta retorna:

```
+-----------+
| items     |
+-----------+
| [1,2,3,4] |
+-----------+
```

Para nivelar uma matriz em várias linhas, use `CROSS JOIN` com o operador `UNNEST`, como neste exemplo:

```
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)
```

Essa consulta retorna:

```
+----------------------+
| department  | names  |
```

```
+----------------------+
| engineering | Sharon |
+----------------------|
| engineering | John   |
+----------------------|
| engineering | Bob    |
+----------------------|
| engineering | Sally  |
+----------------------+
```

Para nivelar uma matriz de pares chave/valor, transpor chaves selecionadas para colunas, como neste exemplo:

```
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)
```

Essa consulta retorna:

```
+-------------------------------------+
| first_name | last_name | department  |
+-------------------------------------+
| Bob        | Smith     | engineering |
| Jane       | Doe       | engineering |
| Billy      | Smith     | engineering |
+-------------------------------------+
```

Em uma lista de funcionários, selecione o funcionário com a maior pontuação combinada. `UNNEST` pode ser usado na cláusula `FROM` sem um `CROSS JOIN` anterior pois ele é o operador de junção padrão e, portanto, implícito.

```
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
```

Essa consulta retorna:

```
+---------------------------------+
| name | department | total_score |
+---------------------------------+
| Amy  | devops     | 54          |
+---------------------------------+
```

Em uma lista de funcionários, selecione o funcionário com a pontuação individual mais alta.

```
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
```

Essa consulta retorna:

```
+--------------+
| name | score |
+--------------+
| Amy  | 15    |
+--------------+
```

Considerações e limitações

Se a função UNNEST for usada em uma ou mais matrizes na consulta e uma das matrizes for NULL, a consulta não retornará nenhuma linha. Se a função UNNEST for usada em uma matriz que é uma string vazia, a string vazia será retornada.

Por exemplo, na consulta apresentada a seguir, como a segunda matriz é nula, a consulta não retorna nenhuma linha.

```
SELECT
    col1,
    col2
FROM UNNEST (ARRAY ['apples','oranges','lemons']) AS t(col1)
CROSS JOIN UNNEST (ARRAY []) AS t(col2)
```

No próximo exemplo, a segunda matriz foi modificada para conter uma string vazia. Para cada linha, a consulta retorna o valor em `col1` e uma string vazia para o valor em `col2`. A string vazia na segunda matriz é necessária para que os valores na primeira matriz sejam retornados.

```
SELECT
```

```
    col1,
    col2
FROM UNNEST (ARRAY ['apples','oranges','lemons']) AS t(col1)
CROSS JOIN UNNEST (ARRAY ['']) AS t(col2)
```

Criar matrizes com base em subconsultas

Crie uma matriz com base em uma coleção de filas.

```
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)
```

Essa consulta retorna:

```
+-----------------+
| array_items     |
+-----------------+
| [1, 2, 3, 4, 5] |
+-----------------+
```

Para criar um conjunto de valores exclusivos com base em um conjunto de linhas, use a palavra-chave `distinct`.

```
WITH
dataset AS (
  SELECT ARRAY [1,2,2,3,3,4,5] AS items
)
SELECT array_agg(distinct i) AS array_items
FROM dataset
CROSS JOIN UNNEST(items) AS t(i)
```

Essa consulta retorna o resultado a seguir. Observe que a ordem não é garantida.

```
+-----------------+
| array_items     |
```

```
+-----------------+
| [1, 2, 3, 4, 5] |
+-----------------+
```

Para obter mais informações sobre como usar a função `array_agg`, consulte Funções aggregate na documentação do Trino.

Filtrar matrizes

Crie uma matriz com base em uma coleção de filas caso elas correspondam aos critérios de filtro.

```
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
```

Essa consulta retorna:

```
+-------------+
| array_items |
+-------------+
| [4, 5]      |
+-------------+
```

Filtrar uma matriz com base em se um dos elementos contêm um valor específico, como 2, como neste exemplo:

```
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)
```

Essa consulta retorna:

```
+--------------+
| array_items  |
+--------------+
| [1, 2, 3, 4] |
+--------------+
```

A função do **`filter`**

```
 filter(ARRAY [list_of_values], boolean_function)
```

É possível utilizar a função `filter` em uma expressão ARRAY para criar um novo array que é o subconjunto dos itens na *`list_of_values`* cuja *`boolean_function`* é true. A função `filter` pode ser útil em casos nos quais não é possível usar a função *`UNNEST`*.

O exemplo a seguir filtra valores maiores que zero no array `[1,0,5,-1]`.

```
SELECT filter(ARRAY [1,0,5,-1], x -> x>0)
```

Resultados

```
[1,5]
```

O exemplo a seguir filtra valores não nulos no array `[-1, NULL, 10, NULL]`.

```
SELECT filter(ARRAY [-1, NULL, 10, NULL], q -> q IS NOT NULL)
```

Resultados

```
[-1,10]
```

Classificar matrizes

Para criar um array classificado de valores exclusivos com base em um conjunto de linhas, você pode usar a função array_sort, como no exemplo a seguir.

```
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)
```

Essa consulta retorna:

```
+--------------------+
| array_items        |
+--------------------+
| [1, 2, 3, 4, 5, 6] |
+--------------------+
```

Para obter informações sobre como expandir uma matriz em várias linhas, consulte Nivelar matrizes aninhadas.

Usar funções de agregação com matrizes

- Para adicionar valores dentro de uma matriz, use SUM, como no exemplo a seguir.
- Para agregar várias linhas dentro de uma matriz, use `array_agg`. Para ter mais informações, consulte Criar matrizes com base em subconsultas.

> **Note**
>
> Desde o mecanismo do Athena versão 2, `ORDER BY` é permitido em funções de agregação.

```
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;
```

Na última instrução do SELECT, em vez de usar `sum()` e UNNEST, você pode usar `reduce()` para reduzir o tempo de processamento e a transferência de dados, como no exemplo a seguir.

```
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;
```

As consultas retornam os seguintes resultados. A ordem de resultados obtidos não é garantida.

```
+----------------------+
| array_items  | total |
+----------------------+
| [1, 2, 3, 4] | 10    |
| [5, 6, 7, 8] | 26    |
| [9, 0]       | 9     |
+----------------------+
```

Converter matrizes em strings

Para converter uma matriz em uma string única, use a função `array_join`. O exemplo independente a seguir cria uma tabela chamada `dataset` que contém um array de alias chamado

`words`. A consulta usa `array_join` para unir os elementos do array em `words`, separá-los com espaços e retornar a string resultante em uma coluna de alias chamada `welcome_msg`.

```
WITH
dataset AS (
  SELECT ARRAY ['hello', 'amazon', 'athena'] AS words
)
SELECT array_join(words, ' ') AS welcome_msg
FROM dataset
```

Essa consulta retorna:

```
+---------------------+
| welcome_msg         |
+---------------------+
| hello amazon athena |
+---------------------+
```

Usar matrizes para criar mapas

Os mapas são pares de chave-valor que consistem nos tipos de dados disponíveis no Athena. Para criar mapas, use o operador `MAP` e passe duas matrizes: a primeira é de nomes de coluna (chave) e a segunda é de valores. Todos os valores nas matrizes devem ser do mesmo tipo. Se qualquer um dos elementos de matriz de valor de mapa precisar ser de tipos diferentes, você poderá convertê-los depois.

Exemplos

Este exemplo seleciona um usuário em um conjunto de dados. Ele usa o operador `MAP` e passa duas matrizes. A primeira matriz inclui valores para nomes de coluna, como "primeiro", "último" e "idade". A segunda matriz consiste em valores para cada uma dessas colunas, como "Bob", "Smith", "35".

```
WITH dataset AS (
  SELECT MAP(
    ARRAY['first', 'last', 'age'],
    ARRAY['Bob', 'Smith', '35']
  ) AS user
)
SELECT user FROM dataset
```

Essa consulta retorna:

```
+---------------------------------+
| user                            |
+---------------------------------+
| {last=Smith, first=Bob, age=35} |
+---------------------------------+
```

Você pode recuperar valores `Map` selecionando o nome do campo seguido de `[key_name]`, como neste exemplo:

```
WITH dataset AS (
 SELECT MAP(
   ARRAY['first', 'last', 'age'],
   ARRAY['Bob', 'Smith', '35']
 ) AS user
)
SELECT user['first'] AS first_name FROM dataset
```

Essa consulta retorna:

```
+------------+
| first_name |
+------------+
| Bob        |
+------------+
```

Consultar matrizes de tipos complexos e com estruturas aninhadas

Os dados de origem normalmente contêm matrizes com tipos de dados complexos e estruturas aninhadas. Os exemplos nesta seção mostram como alterar o tipo de dados do elemento, localizar elementos em arrays e encontrar palavras-chave usando as consultas do Athena.

- Criação de um `ROW`
- Alterar nomes de campo em matrizes usando `CAST`
- Filtrar matrizes usando a notação `.`
- Filtrar matrizes com valores aninhados
- Filtrar matrizes usando `UNNEST`
- Localizar palavras-chave em matrizes usando `regexp_like`

Criação de um **ROW**

> **Note**
>
> Os exemplos neste seção usam `ROW` como um meio para criar dados de exemplo com os quais trabalhar. Ao consultar tabelas no Athena, você não precisa criar tipos de dados `ROW` porque eles já foram criados da sua origem dos dados. Quando você usa `CREATE_TABLE`, o Athena define um `STRUCT` nele, preenche-o com dados e cria o tipo de dados `ROW` para cada linha no conjunto de dados. O tipo de dados `ROW` subjacente consiste em campos nomeados de todos os tipos de dados SQL compatíveis.

```
WITH dataset AS (
 SELECT
   ROW('Bob', 38) AS users
 )
SELECT * FROM dataset
```

Essa consulta retorna:

```
+-------------------------+
| users                   |
+-------------------------+
| {field0=Bob, field1=38} |
+-------------------------+
```

Alterar nomes de campo em matrizes usando **CAST**

Para alterar o nome de campo em uma matriz que contenha valores `ROW`, você pode `CAST` a declaração `ROW`:

```
WITH dataset AS (
  SELECT
    CAST(
      ROW('Bob', 38) AS ROW(name VARCHAR, age INTEGER)
    ) AS users
)
SELECT * FROM dataset
```

Essa consulta retorna:

```
+--------------------+
| users              |
+--------------------+
| {NAME=Bob, AGE=38} |
+--------------------+
```

> **Note**
>
> No exemplo acima, você declara `name` como um `VARCHAR` , porque esse é o tipo no Presto. Se você declarar esse `STRUCT` dentro de uma instrução `CREATE TABLE`, use o tipo `String` porque o Hive define esse tipo de dados como `String`.

Filtrar matrizes usando a notação `.`

No exemplo a seguir, selecione o campo `accountId` na coluna `userIdentity` de uma tabela de logs do AWS CloudTrail usando a notação `.`. Para obter mais informações, consulte Consultar logs do AWS CloudTrail.

```
SELECT
  CAST(useridentity.accountid AS bigint) as newid
FROM cloudtrail_logs
LIMIT 2;
```

Essa consulta retorna:

```
+--------------+
| newid        |
+--------------+
| 112233445566 |
+--------------+
| 998877665544 |
+--------------+
```

Para consultar um conjunto de valores, execute esta consulta:

```
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
```

Ela retorna este resultado:

```
+-----------------------------------------------------------------+
| users                                                           |
+-----------------------------------------------------------------+
| [{NAME=Bob, AGE=38}, {NAME=Alice, AGE=35}, {NAME=Jane, AGE=27}] |
+-----------------------------------------------------------------+
```

Filtrar matrizes com valores aninhados

Matrizes grandes normalmente contêm estruturas aninhadas, e você precisa ser capaz de filtrar ou pesquisar valores dentro delas.

Para definir um conjunto de dados para uma matriz de valores que inclui um valor `BOOLEAN` aninhado, execute esta consulta:

```
WITH dataset AS (
  SELECT
    CAST(
      ROW('aws.amazon.com', ROW(true)) AS ROW(hostname VARCHAR, flaggedActivity
 ROW(isNew BOOLEAN))
    ) AS sites
)
SELECT * FROM dataset
```

Ela retorna este resultado:

```
+-------------------------------------------------------+
| sites                                                 |
+-------------------------------------------------------+
| {HOSTNAME=aws.amazon.com, FLAGGEDACTIVITY={ISNEW=true}}  |
+-------------------------------------------------------+
```

Em seguida, para filtrar e acessar o valor `BOOLEAN` desse elemento, continue usando a notação `.`

```
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
```

Essa consulta seleciona os campos aninhados e retorna este resultado:

```
+------------------------+
| hostname       | isnew |
+------------------------+
| aws.amazon.com | true  |
+------------------------+
```

Filtrar matrizes usando **UNNEST**

Para filtrar uma matriz que inclua uma estrutura aninhada por um dos elementos filho, emita uma consulta com um operador UNNEST. Para obter mais informações sobre UNNEST, consulte Nivelar matrizes aninhadas.

Por exemplo, esta consulta encontra nomes de host de sites no conjunto de dados.

```
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
```

Ela retorna:

```
+------------------------+
| hostname       | isnew |
+------------------------+
| aws.amazon.com | true  |
+------------------------+
```

Localizar palavras-chave em matrizes usando **`regexp_like`**

Os exemplos a seguir ilustram como pesquisar uma palavra-chave em um conjunto de dados em um elemento dentro de uma matriz usando a função regexp_like. Ele usa como entrada um padrão de expressão regular para avaliar ou uma lista de termos separados por uma barra vertical (|), avalia o padrão e determina se a string especificada a contém.

O padrão da expressão regular precisa estar contido na string e não precisa corresponder a ela. Para corresponder à string inteira, coloque o padrão com ^ no início e $ no final, como `'^pattern$'`.

Considere uma matriz de sites contendo os respectivos nomes de host e um elemento `flaggedActivity`. Esse elemento inclui um `ARRAY`, contendo vários elementos `MAP`, cada um listando palavras-chave conhecidas diferentes e a contagem de popularidade. Suponhamos que você encontre uma palavra-chave dentro de um `MAP` nesta matriz.

Para pesquisar esse conjunto de dados para sites com uma palavra-chave específica, usamos `regexp_like` em vez do operador SQL `LIKE` semelhante, porque a pesquisa de um grande número de palavras-chave é mais eficiente com `regexp_like`.

Example Exemplo 1: uso do **`regexp_like`**

A consulta neste exemplo usa a função `regexp_like` para pesquisar os termos `'politics|bigdata'` encontrados em valores em matrizes:

```
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)
```

Essa consulta retorna dois sites:

```
+----------------+
| hostname       |
+----------------+
| aws.amazon.com |
```

```
+----------------+
| news.cnn.com   |
+----------------+
```

Example Exemplo 2: uso do **`regexp_like`**

A consulta no exemplo a seguir agrega ao total de pontuações de popularidade dos sites correspondentes aos termos de pesquisa com a função `regexp_like` e, em seguida, ordena da mais alta para a mais baixa.

```
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
```

Essa consulta retorna dois sites:

```
+-------------------------------------+
| hostname       | terms    | total  |
+----------------+------------------+
| news.cnn.com   | politics |  241   |
+----------------+------------------+
| aws.amazon.com | bigdata |  10    |
+----------------+------------------+
```

Consultar dados geoespaciais

Os dados geoespaciais contêm identificadores que especificam uma posição geográfica para um objeto. Entre os exemplos desse tipo de dados estão previsões do tempo, rotas em mapa, tweets com posições geográficas, locais de lojas e rotas aéreas. Os dados geoespaciais têm uma função importante na análise comercial, na geração de relatórios e na previsão.

Os identificadores geoespaciais, como latitude e longitude, permitem converter qualquer endereço postal em um conjunto de coordenadas geográficas.

Tópicos

- O que é uma consulta geoespacial?
- Formatos de dados de entrada e tipos de dados de geometria
- Funções geoespaciais aceitas
- Exemplos: consultas geoespaciais

O que é uma consulta geoespacial?

As consultas geoespaciais são tipos especializados de consultas SQL disponíveis no Athena. Elas diferem de consultas SQL não espaciais das seguintes maneiras:

- Usando os seguintes tipos de dados de geometria especializados: `point`, `line`, `multiline`, `polygone multipolygon`.
- Expressando relacionamentos entre tipos de dados, como geometria `distance`, `equals`, `crosses`, `touches`, `overlaps`, `disjoint` e outros.

Com as consultas geoespaciais no Athena, você pode executar estas e outras operações semelhantes:

- Encontrar a distância entre dois pontos.
- Verificar se uma área (polígono) contém outra.
- Verifique se uma linha cruza ou toca outra linha ou polígono.

Por exemplo, para obter um tipo de dados de geometria `point` de valores do tipo `double` para as coordenadas geográficas do Monte Rainier no Athena, use a função geoespacial `ST_Point (longitude, latitude)`, como no exemplo a seguir.

```
ST_Point(-121.7602, 46.8527)
```

Formatos de dados de entrada e tipos de dados de geometria

Para usar as funções geoespaciais no Athena, insira os dados no formato WKT ou use o SerDe JSON do Hive. Você também pode usar os tipos de dados de geometria disponíveis no Athena.

Formatos de dados de entrada

Para processar as consultas geoespaciais, o Athena permite a entrada de dados nestes formatos:

- Well-Known Text (WKT – Texto bem conhecido). No Athena, o WKT é representado como um tipo de dados `varchar(x)` ou `string`.
- Dados geoespaciais codificados por JSON. Para analisar arquivos JSON com dados geoespaciais e criar tabelas para eles, o Athena usa o SerDe JSON do Hive. Para obter mais informações sobre como usar esse SerDe no Athena, consulte Bibliotecas SerDe JSON.

Tipos de dados de geometria

Para processar as consultas geoespaciais, o Athena aceita os seguintes tipos de dados de geometria especializados:

- `point`
- `line`
- `polygon`
- `multiline`
- `multipolygon`

Funções geoespaciais aceitas

As funções geoespaciais disponíveis no Athena dependem da versão do mecanismo que você usa.

- Para obter informações sobre as funções geoespaciais no mecanismo Athena versão 3, consulte Geospatial functions (Funções geoespaciais) na documentação do Trino.
- Para obter uma lista das alterações de nomes de funções e das novas funções a partir do mecanismo Athena versão 2, consulte Alterações de nomes de funções geoespaciais e novas funções no mecanismo do Athena versão 2.

Para obter informações sobre o versionamento do mecanismo do Athena, consulte Versionamento do mecanismo do Athena.

Tópicos

- Funções geoespaciais no mecanismo Athena versão 3
- Funções geoespaciais no mecanismo do Athena versão 2

Funções geoespaciais no mecanismo Athena versão 3

Para obter informações sobre as funções geoespaciais no mecanismo Athena versão 3, consulte Geospatial functions (Funções geoespaciais) na documentação do Trino.

Funções geoespaciais no mecanismo do Athena versão 2

Este tópico lista as funções geoespaciais do ESRI compatíveis a partir do mecanismo Athena versão 2. Para obter informações sobre as versões do mecanismo do Athena, consulte Versionamento do mecanismo do Athena.

Alterações no mecanismo do Athena versão 2

- Os tipos de entrada e saída de algumas funções foram alterados. Particularmente, o tipo `VARBINARY` não é mais aceito diretamente como entrada. Para ter mais informações, consulte Alterações nas funções geoespaciais.
- Os nomes de algumas funções geoespaciais foram alteradas. Para ter mais informações, consulte Alterações de nomes de funções geoespaciais no mecanismo do Athena versão 2.
- Novas funções foram adicionadas. Para ter mais informações, consulte Novas funções geoespaciais no mecanismo do Athena versão 2.

O Athena permite os seguintes tipos de funções geoespaciais:

- Funções do construtor
- Funções de relacionamentos geoespaciais
- Funções de operação
- Funções do acessor
- Funções de agregação
- Funções de blocos do Bing

Funções do construtor

Use funções do construtor para obter representações binárias dos tipos de dados de geometria `point`, `line` ou `polygon`. Você também pode usar essas funções para converter dados binários em texto e obter valores binários para dados de geometria expressos como Well-Known Text (WKT – Texto bem conhecido).

`ST_AsBinary(geometry)`

Retorna um tipo de dados varbinary que contém a representação WKB da geometria especificada. Exemplo:

```
SELECT ST_AsBinary(ST_Point(-158.54, 61.56))
```

ST_AsText(geometry)

Converte cada um dos tipos de dados de geometria especificados em texto. Retorna um valor em um tipo de dados varchar, que é uma representação WKT do tipo de dados de geometria. Exemplo:

```
SELECT ST_AsText(ST_Point(-158.54, 61.56))
```

ST_GeomAsLegacyBinary(geometry)

Retorna um varbinary herdado da geometria especificada. Exemplo:

```
SELECT ST_GeomAsLegacyBinary(ST_Point(-158.54, 61.56)
```

ST_GeometryFromText(varchar)

Converte um texto no formato WKT em um tipo de dados de geometria. Retorna um valor como um tipo de dados de geometria. Exemplo:

```
SELECT ST_GeometryFromText(ST_AsText(ST_Point(1, 2)))
```

ST_GeomFromBinary(varbinary)

Retorna um objeto de tipo geometria de uma representação WKB. Exemplo:

```
SELECT ST_GeomFromBinary(ST_AsBinary(ST_Point(-158.54, 61.56)))
```

ST_GeomFromLegacyBinary(varbinary)

Retorna um objeto de tipo geometria de um tipo varbinary herdado. Exemplo:

```
SELECT ST_GeomFromLegacyBinary(ST_GeomAsLegacyBinary(ST_Point(-158.54, 61.56)))
```

ST_LineFromText(varchar)

Retorna um valor como um tipo de dados de geometria `line`. Exemplo:

```
SELECT ST_Line('linestring(1 1, 2 2, 3 3)')
```

ST_LineString(array(point))

Retorna um tipo de geometria `LineString` formado de um array de tipos de geometria de ponto. Se houver menos de dois pontos não vazios no array especificado, um `LineString` vazio será retornado. Gera uma exceção quando um elemento no array é nulo, está vazio ou é igual ao anterior. A geometria retornada pode não ser simples. Dependendo da entrada especificada, a geometria retornada pode se intersectar ou conter vértices duplicados. Exemplo:

```
SELECT ST_LineString(ARRAY[ST_Point(-158.54, 61.56), ST_Point(-158.55, 61.56)])
```

ST_MultiPoint(array(point))

Retorna um objeto de geometria `MultiPoint` formado com base nos pontos especificados. Retorna nulo quando o array especificado está vazio. Gera uma exceção quando um elemento no array é nulo ou está vazio. A geometria retornada talvez não seja simples e contenha pontos duplicados se o array especificado tiver duplicatas. Exemplo:

```
SELECT ST_MultiPoint(ARRAY[ST_Point(-158.54, 61.56), ST_Point(-158.55, 61.56)])
```

ST_Point(double, double)

Retorna um objeto `point` de tipo de geometria. Para os valores de dados de entrada para essa função, use valores geométricos, como valores do sistema de coordenadas cartesianas da Universal Transversa de Mercator (UTM) ou unidades de mapas geográficos (latitude e longitude) em graus decimais. Os valores de latitude e longitude usam o sistema geodésico mundial, também conhecido como WGS 1984 ou EPSG:4326. WGS 1984 é o sistema de coordenadas usado pelo sistema GPS.

Por exemplo, na seguinte notação, as coordenadas do mapa são especificadas em latitude e longitude, e o valor `.072284`, que é a distância de folga, é especificada em unidades anguladas na forma de graus decimais:

```
SELECT ST_Buffer(ST_Point(-74.006801, 40.705220), .072284)
```

Sintaxe:

```
SELECT ST_Point(longitude, latitude) FROM earthquakes LIMIT 1
```

O seguinte exemplo usa coordenadas específicas de latitude e longitude:

```
SELECT ST_Point(-158.54, 61.56)
FROM earthquakes
LIMIT 1
```

O próximo exemplo usa coordenadas de longitude e latitude específicas:

```
SELECT ST_Point(-74.006801, 40.705220)
```

O seguinte exemplo usa a função `ST_AsText` para obter a geometria de WKT:

```
SELECT ST_AsText(ST_Point(-74.006801, 40.705220)) AS WKT
```

ST_Polygon(varchar)

Usando a sequência de ordenadas fornecidas no sentido horário, da esquerda para a direita, retorna um tipo de dados de geometria `polygon`. A partir do mecanismo Athena versão 2, apenas os polígonos são aceitos como entradas. Exemplo:

```
SELECT ST_Polygon('polygon ((1 1, 1 4, 4 4, 4 1))')
```

to_geometry(sphericalGeography)

Retorna um objeto de geometria com base no objeto geográfico esférico especificado. Exemplo:

```
SELECT to_geometry(to_spherical_geography(ST_Point(-158.54, 61.56)))
```

to_spherical_geography(geometry)

Retorna um objeto geográfico esférico com base na geometria especificada. Use essa função para converter um objeto de geometria em um objeto geográfico esférico com base na esfera do raio da Terra. É possível usar essa função somente nas geometrias `POINT`, `MULTIPOINT`, `LINESTRING`, `MULTILINESTRING`, `POLYGON` e `MULTIPOLYGON` definidas em espaço 2D ou uma `GEOMETRYCOLLECTION` dessas geometrias. Para cada ponto da geometria especificada, a função verifica se `point.x` está dentro de `[-180.0, 180.0]` e `point.y` está dentro de `[-90.0, 90.0]`. A função usa esses pontos como graus de latitude e longitude para construir a forma do resultado de `sphericalGeography`.

Exemplo:

```
SELECT to_spherical_geography(ST_Point(-158.54, 61.56))
```

Funções de relacionamentos geoespaciais

As funções a seguir expressam os relacionamentos entre duas geometrias diferentes que você especifica como entrada e retornam resultados do tipo `boolean`. A ordem na qual você especifica o par de geometrias importa: o primeiro valor de geometria é chamado de geometria à esquerda, o segundo valor de geometria é chamado de geometria à direita.

Essas funções retornam:

- `TRUE` se e somente se o relacionamento descrito pela função é atendido.
- `FALSE` se e somente se o relacionamento descrito pela função não é atendido.

`ST_Contains(geometry, geometry)`

Retornará `TRUE` se e somente se a geometria à esquerda contiver a geometria à direita. Exemplos:

```
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)`

Retornará `TRUE` se e somente se a geometria à esquerda cruzar a geometria à direita. Exemplo:

```
SELECT ST_Crosses(ST_Line('linestring(1 1, 2 2 )'), ST_Line('linestring(0 1, 2 2)'))
```

`ST_Disjoint(geometry, geometry)`

Retornará `TRUE` se e somente se a interseção da geometria à esquerda e da geometria à direita estiver vazia. Exemplo:

```
SELECT ST_Disjoint(ST_Line('linestring(0 0, 0 1)'), ST_Line('linestring(1 1, 1 0)'))
```

ST_Equals(geometry, geometry)

Retornará TRUE se e somente se a geometria à esquerda for igual à geometria à direita. Exemplo:

```
SELECT ST_Equals(ST_Line('linestring( 0 0, 1 1)'), ST_Line('linestring(1 3, 2 2)'))
```

ST_Intersects(geometry, geometry)

Retornará TRUE se e somente se a geometria à esquerda apresentar intersecção com a geometria à direita. Exemplo:

```
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)

Retornará TRUE se e somente se a geometria à esquerda sobrepuser a geometria à direita. Exemplo:

```
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)

Retornará TRUE somente se a geometria à esquerda tiver o relacionamento Dimensionally Extended nine-Intersection Model (DE-9IM – Modelo de nove interseções estendido dimensionalmente) especificado com a geometria à direita. A terceira entrada (`varchar`) representa o relacionamento. Exemplo:

```
SELECT ST_Relate(ST_Line('linestring(0 0, 3 3)'), ST_Line('linestring(1 1, 4 4)'),
 'T********')
```

ST_Touches(geometry, geometry)

Retornará TRUE se e somente se a geometria à esquerda tocar a geometria à direita.

Exemplo:

```
SELECT ST_Touches(ST_Point(8, 8), ST_Polygon('polygon((1  1, 1  4, 4  4, 4 1))'))
```

ST_Within(geometry, geometry)

Retornará TRUE se e somente se a geometria à esquerda estiver dentro da geometria à direita.

Exemplo:

```
SELECT ST_Within(ST_Point(8, 8), ST_Polygon('polygon((1  1, 1  4, 4  4, 4 1))'))
```

Funções de operação

Use funções de operação para realizar operações nos valores do tipo de dados de geometria. Por exemplo, você pode obter os limites de um único tipo de dados de geometria, interseções entre dois tipos de dados de geometria, diferença entre geometrias à esquerda e à direita, em que cada um é do mesmo tipo de dados de geometria ou um buffer externo ou um anel em torno de um tipo de dados de geometria específico.

geometry_union(array(geometry))

Retorna uma geometria que representa a união do conjunto de pontos das geometrias especificadas. Exemplo:

```
SELECT geometry_union(ARRAY[ST_Point(-158.54, 61.56), ST_Point(-158.55, 61.56)])
```

ST_Boundary(geometry)

Usa como entrada um dos tipos de dados de geometria e retorna o tipo de dados de geometria `boundary`.

Exemplos:

```
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)

Usa como entrada um dos tipos de dados de geometria, como ponto, linha, polígono, multilinha ou multipolígono e uma distância como tipo `double`). Retorna o tipo de dados de geometria armazenado em buffer pela distância especificada (ou raio). Exemplo:

```
SELECT ST_Buffer(ST_Point(1, 2), 2.0)
```

No seguinte exemplo, as coordenadas do mapa são especificadas em latitude e longitude, e o valor `.072284`, que é a distância de folga, é especificado em unidades anguladas na forma de graus decimais:

```
SELECT ST_Buffer(ST_Point(-74.006801, 40.705220), .072284)
```

ST_Difference(geometry, geometry)

Retorna uma geometria da diferença entre a geometria à esquerda e a geometria à direita. Exemplo:

```
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)

Recebe como entrada os tipos de dados de geometria `line`, `polygon`, `multiline` e `multipolygon`. Não oferece suporte ao tipo de dados de geometria `point`. Retorna o envelope como uma geometria, em que o envelope é um retângulo em torno do tipo de dados de geometria especificado. Exemplos:

```
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)

Retorna um array de dois pontos que representam os cantos inferior esquerdo e superior direito do polígono retangular delimitador de uma geometria. Retorna nulo quando a geometria especificada está vazia. Exemplo:

```
SELECT ST_EnvelopeAsPts(ST_Point(-158.54, 61.56))
```

ST_ExteriorRing(geometry)

Retorna a geometria do anel externo do tipo de entrada `polygon`. A partir do mecanismo Athena versão 2, os polígonos são as únicas geometrias aceitas como entradas. Exemplos:

```
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)

Retorna a geometria da interseção da geometria à esquerda e da geometria à direita. Exemplos:

```
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)

Retorna a geometria da diferença geometricamente simétrica entre a geometria à esquerda e a geometria à direita. Exemplo:

```
SELECT ST_AsText(ST_SymDifference(ST_Line('linestring(0 2, 2 2)'),
 ST_Line('linestring(1 2, 3 2)')))
```

ST_Union(geometry, geometry)

Retorna um tipo de dados de geometria que representa a união do conjunto de pontos das geometrias especificadas. Exemplo:

```
SELECT ST_Union(ST_Point(-158.54, 61.56),ST_LineString(array[ST_Point(1,2),
 ST_Point(3,4)]))
```

Funções do acessor

As funções do acessor são úteis para obter valores nos tipos `varchar`, `bigint` ou `double` de tipos de dados `geometry` diferentes, em que `geometry` é qualquer um dos tipos de dados de geometria disponíveis no Athena: `point`, `line`, `polygon`, `multiline` e `multipolygon`. Por exemplo, você pode obter uma área de um tipo de dados de geometria `polygon`, valores X e Y máximos e mínimos

para um tipo de dados de geometria especificado, obter o comprimento de um `line` ou receber o número de pontos em um tipo de dados de geometria especificado.

geometry_invalid_reason(geometry)

Retorna, como tipo de dados varchar, a razão pela qual a geometria especificada não é válida ou não é simples. Se a geometria especificada não é válida nem simples, retorna a razão. Se a geometria especificada é válida e simples, retorna nulo. Exemplo:

```
SELECT geometry_invalid_reason(ST_Point(-158.54, 61.56))
```

great_circle_distance(latitude1, longitude1, latitude2, longitude2)

Retorna, como valor duplo, a distância do círculo máximo entre dois pontos na superfície da Terra em quilômetros. Exemplo:

```
SELECT great_circle_distance(36.12, -86.67, 33.94, -118.40)
```

line_locate_point(lineString, point)

Retorna um valor duplo entre 0 e 1 que representa a localização do ponto mais próximo na string de linha especificada ao ponto especificado como uma fração do comprimento total da linha 2d.

Retorna nulo quando a string de linha ou o ponto especificado está vazio ou é nulo. Exemplo:

```
SELECT line_locate_point(ST_GeometryFromText('LINESTRING (0 0, 0 1)'), ST_Point(0,
 0.2))
```

simplify_geometry(geometry, double)

Usa o algoritmo Ramer-Douglas-Peucker para retornar um tipo de dado de geometria que é uma versão simplificada da geometria especificada. Evita a criação de geometrias derivadas (em particular, polígonos) que são inválidas. Exemplo:

```
SELECT simplify_geometry(ST_GeometryFromText('POLYGON ((1 0, 2 1, 3 1, 3 1, 4 1, 1
 0))'), 1.5)
```

ST_Area(geometry)

Utiliza como entrada um tipo de dado de geometria e retorna uma área no tipo `double`. Exemplo:

```
SELECT ST_Area(ST_Polygon('polygon((1 1, 4 1, 4 4, 1 4))'))
```

ST_Centroid(geometry)

Usa como entrada um tipo de dados de geometria `polygon` e retorna um tipo de dados de geometria `point` que é o centro do envelope do polígono. Exemplos:

```
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)

Retorna um tipo de dados de geometria que é a menor geometria convexa que engloba todas as geometrias na entrada especificada. Exemplo:

```
SELECT ST_ConvexHull(ST_Point(-158.54, 61.56))
```

ST_CoordDim(geometry)

Usa como entrada um dos tipos de dados de geometria aceitos e retorna a contagem de componentes de coordenadas no tipo `tinyint`. Exemplo:

```
SELECT ST_CoordDim(ST_Point(1.5,2.5))
```

ST_Dimension(geometry)

Utiliza como entrada um dos tipos de dados de geometria compatíveis e retorna a dimensão espacial de uma geometria no tipo `tinyint`. Exemplo:

```
SELECT ST_Dimension(ST_Polygon('polygon((1 1, 4 1, 4 4, 1 4))'))
```

ST_Distance(geometry, geometry)

Retorna, com base na referência espacial, um valor duplo que contém a distância cartesiana mínima bidimensional entre duas geometrias nas unidades projetadas. A partir do mecanismo Athena versão 2, a informação retornada será nula se uma das entradas for uma geometria vazia. Exemplo:

```
SELECT ST_Distance(ST_Point(0.0,0.0), ST_Point(3.0,4.0))
```

ST_Distance(sphericalGeography, sphericalGeography)

Retorna, como valor duplo, a distância do círculo máximo entre dois pontos geográficos esféricos em metros. Exemplo:

```
SELECT ST_Distance(to_spherical_geography(ST_Point(61.56,
 -86.67)),to_spherical_geography(ST_Point(61.56, -86.68)))
```

ST_EndPoint(geometry)

Retorna o último ponto de um tipo de dados de geometria `line` como um tipo de dados de geometria `point`. Exemplo:

```
SELECT ST_EndPoint(ST_Line('linestring(0 2, 2 2)'))
```

ST_Geometries(geometry)

Retorna um array de geometrias na coleção especificada. Se a geometria especificada não for múltipla, retornará um array de elemento único. Quando a geometria especificada está vazia, retorna nulo.

Por exemplo, se o objeto for `MultiLineString`, `ST_Geometries` criará um array de objetos `LineString`. Se um objeto for `GeometryCollection`, `ST_Geometries` retornará um array não nivelado dos respectivos integrantes. Exemplo:

```
SELECT ST_Geometries(GEOMETRYCOLLECTION(MULTIPOINT(0 0, 1 1),
 GEOMETRYCOLLECTION(MULTILINESTRING((2 2, 3 3)))))
```

Resultado:

```
array[MULTIPOINT(0 0, 1 1),GEOMETRYCOLLECTION(MULTILINESTRING((2 2, 3 3)))]
```

ST_GeometryN(geometry, index)

Retorna, como tipo de dados de geometria, o elemento de geometria com um índice de número inteiro especificado. Os índices começam em 1. Se a geometria especificada for uma coleção de geometrias (por exemplo, um objeto `GEOMETRYCOLLECTION` ou `MULTI*`), retornará a geometria no índice especificado. Se o índice especificado for menor que 1 ou maior que o número total de elementos na coleção, retornará nulo. Para saber o número total de elementos, use `ST_NumGeometries`. As geometrias singulares (por exemplo, `POINT`, `LINESTRING` ou `POLYGON`)

são tratadas como coleções de um elemento. As geometrias vazias são tratadas como coleções vazias. Exemplo:

```
SELECT ST_GeometryN(ST_Point(-158.54, 61.56),1)
```

ST_GeometryType(geometry)

Retorna, como varchar, o tipo da geometria. Exemplo:

```
SELECT ST_GeometryType(ST_Point(-158.54, 61.56))
```

ST_InteriorRingN(geometry, index)

Retorna o elemento do anel interno no índice especificado (os índices começam em 1). Se o índice especificado for menor que 1 ou maior que o número total de anéis internos na geometria especificada, retornará nulo. Gera um erro quando a geometria especificada não é um polígono. Para saber o número total de elementos, use ST_NumInteriorRing. Exemplo:

```
SELECT ST_InteriorRingN(st_polygon('polygon ((0 0, 1 0, 1 1, 0 1, 0 0))'),1)
```

ST_InteriorRings(geometry)

Retorna um array de geometria de todos os anéis internos encontrados na geometria especificada, ou um array vazio se o polígono não tiver anéis internos. Quando a geometria especificada está vazia, retorna nulo. Quando a geometria especificada não é um polígono, gera um erro. Exemplo:

```
SELECT ST_InteriorRings(st_polygon('polygon ((0 0, 1 0, 1 1, 0 1, 0 0))'))
```

ST_IsClosed(geometry)

Usa como entrada somente os tipos de dados de geometria `line` e `multiline`. Retornará `TRUE` (tipo `boolean`) se e somente se a linha for fechada. Exemplo:

```
SELECT ST_IsClosed(ST_Line('linestring(0 2, 2 2)'))
```

ST_IsEmpty(geometry)

Usa como entrada somente os tipos de dados de geometria `line` e `multiline`. Retornará `TRUE` (tipo `boolean`) somente se a geometria especificada estiver vazia, em outras palavras, quando os valores inicial e final de `line` coincidirem. Exemplo:

```
SELECT ST_IsEmpty(ST_Point(1.5, 2.5))
```

ST_IsRing(geometry)

Retornará TRUE (tipo `boolean`) se e somente se o tipo `line` for fechado e simples. Exemplo:

```
SELECT ST_IsRing(ST_Line('linestring(0 2, 2 2)'))
```

ST_IsSimple(geometry)

Retornará true se a geometria especificada não tiver pontos geométricos anômalos (por exemplo, autointerseção ou autotangência). Para determinar por que a geometria não é simples, use `geometry_invalid_reason()`. Exemplo:

```
SELECT ST_IsSimple(ST_LineString(array[ST_Point(1,2), ST_Point(3,4)]))
```

ST_IsValid(geometry)

Retornará true somente se a geometria especificada estiver bem formada. Para determinar por que a geometria não está bem formada, use `geometry_invalid_reason()`. Exemplo:

```
SELECT ST_IsValid(ST_Point(61.56, -86.68))
```

ST_Length(geometry)

Retorna o comprimento de `line` no tipo `double`. Exemplo:

```
SELECT ST_Length(ST_Line('linestring(0 2, 2 2)'))
```

ST_NumGeometries(geometry)

Retorna, como inteiro, o número de geometrias na coleção. Se a geometria for uma coleção de geometrias (por exemplo, um objeto `GEOMETRYCOLLECTION` ou `MULTI*`), retornará o número de geometrias. Geometrias únicas retornam 1, geometrias vazias retornam 0. Uma geometria vazia em um objeto `GEOMETRYCOLLECTION` é contada como uma geometria. Por exemplo, o seguinte exemplo é avaliado como 1:

```
ST_NumGeometries(ST_GeometryFromText('GEOMETRYCOLLECTION(MULTIPOINT EMPTY)'))
```

ST_NumInteriorRing(geometry)

Retorna o número de anéis internos na geometria `polygon` no tipo `bigint`. Exemplo:

```
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)

Retorna o número de pontos na geometria no tipo `bigint`. Exemplo:

```
SELECT ST_NumPoints(ST_Point(1.5, 2.5))
```

ST_PointN(lineString, index)

Retorna, como tipo de dados de geometria de ponto, o vértice da string de linha especificada no índice inteiro especificado. Os índices começam em 1. Se o índice especificado for menor que 1 ou maior que o número total de elementos na coleção, retornará nulo. Para saber o número total de elementos, use `ST_NumPoints`. Exemplo:

```
SELECT ST_PointN(ST_LineString(array[ST_Point(1,2), ST_Point(3,4)]),1)
```

ST_Points(geometry)

Retorna um array de pontos do objeto de geometria de string de linha especificado. Exemplo:

```
SELECT ST_Points(ST_LineString(array[ST_Point(1,2), ST_Point(3,4)]))
```

ST_StartPoint(geometry)

Retorna o primeiro ponto de um tipo de dados de geometria `line` em um tipo de dados de geometria `point`. Exemplo:

```
SELECT ST_StartPoint(ST_Line('linestring(0 2, 2 2)'))
```

ST_X(point)

Retorna a coordenada X de um ponto no tipo `double`. Exemplo:

```
SELECT ST_X(ST_Point(1.5, 2.5))
```

ST_XMax(geometry)

Retorna a coordenada X máxima de uma geometria no tipo `double`. Exemplo:

```
SELECT ST_XMax(ST_Line('linestring(0 2, 2 2)'))
```

ST_XMin(geometry)

Retorna a coordenada X mínima de uma geometria no tipo `double`. Exemplo:

```
SELECT ST_XMin(ST_Line('linestring(0 2, 2 2)'))
```

ST_Y(point)

Retorna a coordenada Y de um ponto no tipo `double`. Exemplo:

```
SELECT ST_Y(ST_Point(1.5, 2.5))
```

ST_YMax(geometry)

Retorna a coordenada Y máxima de uma geometria no tipo `double`. Exemplo:

```
SELECT ST_YMax(ST_Line('linestring(0 2, 2 2)'))
```

ST_YMin(geometry)

Retorna a coordenada Y mínima de uma geometria no tipo `double`. Exemplo:

```
SELECT ST_YMin(ST_Line('linestring(0 2, 2 2)'))
```

Funções de agregação

convex_hull_agg(geometry)

Retorna a geometria convexa mínima que engloba todas as geometrias inseridas como entrada.

geometry_union_agg(geometry)

Retorna uma geometria que representa a união do conjunto de pontos de todas as geometrias inseridas como entrada.

Funções de blocos do Bing

As funções a seguir convertem geometrias e blocos no sistema de blocos do Bing Mapas da Microsoft.

bing_tile(x, y, zoom_level)

Retorna um objeto de bloco do Bing com base nas coordenadas inteiras x e y e no nível de zoom especificado. O nível de zoom deve ser um número inteiro de 1 a 23. Exemplo:

```
SELECT bing_tile(10, 20, 12)
```

bing_tile(quadKey)

Retorna um objeto de bloco do Bing de um quadkey. Exemplo:

```
SELECT bing_tile(bing_tile_quadkey(bing_tile(10, 20, 12)))
```

bing_tile_at(latitude, longitude, zoom_level)

Retorna um objeto de bloco do Bing na latitude, na longitude e no nível de zoom especificados. A latitude deve ser entre -85,05112878 e 85,05112878. A longitude deve ser entre -180 e 180. Os valores `latitude` e `longitude` devem ser `double`, e `zoom_level` deve ser um número inteiro. Exemplo:

```
SELECT bing_tile_at(37.431944, -122.166111, 12)
```

bing_tiles_around(latitude, longitude, zoom_level)

Retorna um array de blocos do Bing que cercam o ponto de latitude e longitude especificado no nível do zoom informado. Exemplo:

```
SELECT bing_tiles_around(47.265511, -122.465691, 14)
```

bing_tiles_around(latitude, longitude, zoom_level, radius_in_km)

Retorna, no nível do zoom especificado, um array de blocos do Bing. O array contém o conjunto mínimo de blocos do Bing que cobre um círculo do raio especificado em quilômetros ao redor da latitude e longitude especificadas. Os valores `latitude`, `longitude` e `radius_in_km` são `double`, o nível do zoom é um `integer`. Exemplo:

```
SELECT bing_tiles_around(37.8475, 112.596667, 10, .5)
```

bing_tile_coordinates(tile)

Retorna as coordenadas x e y do bloco do Bing especificado. Exemplo:

```
SELECT bing_tile_coordinates(bing_tile_at(37.431944, -122.166111, 12))
```

bing_tile_polygon(tile)

Retorna a representação de polígono do bloco do Bing especificado. Exemplo:

```
SELECT bing_tile_polygon(bing_tile_at(47.265511, -122.465691, 4))
```

bing_tile_quadkey(tile)

Retorna o quadkey do bloco do Bing especificado. Exemplo:

```
SELECT bing_tile_quadkey(bing_tile(52, 143, 10))
```

bing_tile_zoom_level(tile)

Retorna o nível do zoom do bloco do Bing especificado como um inteiro. Exemplo:

```
SELECT bing_tile_zoom_level(bing_tile(52, 143, 10))
```

geometry_to_bing_tiles(geometry, zoom_level)

Retorna o conjunto mínimo de blocos do Bing que cobrem totalmente a geometria especificada no nível do zoom especificado. Os níveis de zoom permitidos são de 1 a 23. Exemplo:

```
SELECT geometry_to_bing_tiles(ST_Point(61.56, 58.54), 10)
```

Alterações de nomes de funções geoespaciais e novas funções no mecanismo do Athena versão 2

Esta seção lista as alterações nos nomes de funções geoespaciais e as funções geoespaciais que são novas no mecanismo do Athena versão 2. Para obter mais informações sobre outras alterações a partir do mecanismo Athena versão 2, consulte Mecanismo do Athena versão 2.

Para obter informações sobre o versionamento do mecanismo do Athena, consulte Versionamento do mecanismo do Athena.

Alterações de nomes de funções geoespaciais no mecanismo do Athena versão 2

Os nomes das funções a seguir foram alterados. Em alguns casos, os tipos de entrada e saída também foram alterados. Para obter mais informações, acesse os links correspondentes.

Nome da função anterior	Início de nome de função no mecanismo Athena versão 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
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

Novas funções geoespaciais no mecanismo do Athena versão 2

As seguintes funções geoespaciais são novas a partir do mecanismo do Athena versão 2. Para obter mais informações, acesse os links correspondentes.

Funções do construtor

- ST_AsBinary
- ST_GeomAsLegacyBinary
- ST_GeomFromBinary
- ST_GeomFromLegacyBinary
- ST_LineString
- ST_MultiPoint
- to_geometry
- to_spherical_geography

Funções de operação

- geometry_union
- ST_EnvelopeAsPts
- ST_Union

Funções do acessor

- geometry_invalid_reason
- great_circle_distance
- line_locate_point
- simplify_geometry
- ST_ConvexHull
- ST_Distance (geografia esférica)
- ST_Geometries
- ST_GeometryN
- ST_GeometryType
- ST_InteriorRingN
- ST_InteriorRings
- ST_IsSimple
- ST_IsValid

- ST_NumGeometries
- ST_PointN
- ST_Points

Funções de agregação

- convex_hull_agg
- geometry_union_agg

Funções de blocos do Bing

- bing_tile
- bing_tile (quadkey)
- bing_tile_at
- bing_tiles_around
- bing_tiles_around (raio)
- bing_tile_coordinates
- bing_tile_polygon
- bing_tile_quadkey
- bing_tile_zoom_level
- geometry_to_bing_tiles

Exemplos: consultas geoespaciais

Os exemplos neste tópico criam duas tabelas de dados de exemplo disponíveis no GitHub e consultam as tabelas com base nos dados. Os dados de exemplo, que são apenas para fins ilustrativos e não têm garantia de serem precisos, estão nos seguintes arquivos:

- **`earthquakes.csv`**: lista terremotos ocorridos na Califórnia. A tabela `earthquakes` de exemplo usa campos desses dados.
- **`california-counties.json`**: lista os dados do condado no estado da Califórnia no formato GeoJSON compatível com ESRI. Os dados incluem muitos campos, como `AREA`, `PERIMETER`, `STATE`, `COUNTY` e `NAME`, mas a tabela `counties` de exemplo usa apenas dois: `Name` (string) e `BoundaryShape` (binário).

> **Note**
>
> O Athena usa `com.esri.json.hadoop.EnclosedEsriJsonInputFormat` para converter os dados JSON no formato binário geoespacial.

O exemplo de código a seguir cria uma tabela chamada `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/';
```

O exemplo de código a seguir cria uma tabela chamada `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/';
```

A consulta de exemplo a seguir usa a função `CROSS JOIN` nas tabelas `counties` e `earthquake`. O exemplo usa `ST_CONTAINS` para consultar os condados com limites que incluem locais de

terremoto, que são especificados com `ST_POINT`. A consulta agrupa esses condados por nome, ordena-os por contagem e os retorna em ordem decrescente.

> **Note**
>
> A partir do mecanismo Athena versão 2, funções como `ST_CONTAINS` deixarão de aceitar o tipo `VARBINARY` como entrada. Por esse motivo, o exemplo usa a função `ST_GeomFromLegacyBinary(varbinary)` para converter o valor binário `boundaryshape` em uma geometria. Para obter mais informações, consulte Alterações nas funções geoespaciais na referência Mecanismo do Athena versão 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
```

Essa consulta retorna:

```
+------------------------+
| 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   |
+------------------------+
```

Recursos adicionais do

Para ver exemplos adicionais de consultas geoespaciais, consulte as seguintes postagens de blog:

- Estenda consultas geoespaciais no Amazon Athena com UDFs e AWS Lambda
- Visualize mais de 200 anos de dados climáticos globais usando o Amazon Athena e o Amazon QuickSight.
- Querying OpenStreetMap with Amazon Athena (Consultar OpenStreetMap com o Amazon Athena)

Consultar JSON

O Amazon Athena permite analisar valores codificados em JSON, extrair dados do JSON, pesquisar valores e saber o comprimento e o tamanho dos arrays JSON.

Tópicos

- Práticas recomendadas de leitura de dados JSON
- Extrair dados do JSON
- Pesquisar valores em matrizes JSON
- Obter comprimento e tamanho de matrizes JSON
- Solucionar problemas de consultas JSON

Práticas recomendadas de leitura de dados JSON

JavaScript Object Notation (JSON) é um método comum para codificar estruturas de dados como texto. Muitos aplicativos e ferramentas produzem dados codificados em JSON.

No Amazon Athena, você pode criar tabelas com base em dados externos e incluir dados codificados em JSON nelas. Para esses tipos de dados de origem, use o Athena junto com Bibliotecas SerDe JSON.

Use as seguintes dicas para ler dados codificados por JSON:

- Escolha o SerDe certo: um JSON SerDe nativo, `org.apache.hive.hcatalog.data.JsonSerDe`; ou um OpenX SerDe, `org.openx.data.jsonserde.JsonSerDe`. Para ter mais informações, consulte Bibliotecas SerDe JSON.
- Certifique-se de que cada registro codificado em JSON seja representado em uma linha separada, não formatado para impressão.

> **Note**
>
> O SerDe espera que cada documento JSON esteja em uma única linha de texto, sem caracteres de terminação de linha separando os campos no registro. Se o texto JSON estiver formatado para impressão, você poderá receber uma mensagem de erro como HIVE_CURSOR_ERROR: Row is not a valid JSON Object (HIVE_CURSOR_ERROR: a linha não é um objeto JSON válido) ou HIVE_CURSOR_ERROR: JsonParseException: Unexpected end-of-input: expected close marker for OBJECT (HIVE_CURSOR_ERROR: JSONParseException: Fim de entrada inesperado: marcador de fechamento esperado para OBJECT) quando tentar consultar a tabela após criá-la. Para obter mais informações, consulte JSON Data Files na documentação do OpenX SerDe no GitHub.

- Gere seus dados codificados por JSON em colunas sem distinção entre letras maiúsculas e minúsculas.
- Forneça uma opção para ignorar registros malformadas, como neste exemplo.

```
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/';
```

- Converta os campos nos dados de origem que tenham um esquema indeterminado em strings codificadas em JSON no Athena.

Ao criar tabelas com os dados do JSON, o Athena analisa os dados com base no esquema existente e predefinido. No entanto, nem todos os dados podem ter um esquema predefinido. Para simplificar o gerenciamento de esquemas nesses casos, costuma ser útil converter os campos nos dados de origem que têm um esquema indeterminado em strings JSON no Athena e usar Bibliotecas SerDe JSON.

Por exemplo, considere um aplicativo IoT que publique eventos com campos comuns de sensores diferentes. Um desses campos deve armazenar uma carga útil personalizada que seja exclusiva do sensor que envia o evento. Nesse caso, como você não sabe o esquema, recomendamos armazenar as informações como uma string codificada em JSON. Para isso, converta os dados na tabela do Athena em JSON, como no exemplo a seguir. Você também pode converter os dados codificados em JSON em tipos de dados do Athena.

- Converter tipos de dados do Athena em JSON
- Converter JSON em tipos de dados do Athena

Converter tipos de dados do Athena em JSON

Para converter os tipos de dados do Athena em JSON, use `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
```

Essa consulta retorna:

```
+-----------------------------------------+
| hello_msg      | some_int | some_map      |
```

```
+----------------------------------------+
| "HELLO ATHENA" | 12345    | {"a":1,"b":2} |
+----------------------------------------+
```

Converter JSON em tipos de dados do Athena

Para converter os dados do JSON em tipos de dados do Athena, use `CAST`.

> **Note**
>
> Neste exemplo, para denotar strings como codificadas em JSON, comece com a palavra-chave `JSON` e use aspas simples, como `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
```

Essa consulta retorna:

```
+-------------------------------------+
| hello_msg    | some_int | some_map  |
+-------------------------------------+
| HELLO ATHENA | 12345    | {a:1,b:2} |
+-------------------------------------+
```

Extrair dados do JSON

Você pode ter dados de origem contendo strings codificadas em JSON que não deseja necessariamente desserializar em uma tabela no Athena. Neste caso, você ainda pode executar operações SQL nesses dados usando as funções JSON disponíveis no Presto.

Considere essa string JSON como um conjunto de dados de exemplo.

```
{"name": "Susan Smith",
```

```
"org": "engineering",
"projects":
    [
     {"name":"project1", "completed":false},
     {"name":"project2", "completed":true}
    ]
}
```

Exemplos: extrair propriedades

Para extrair as propriedades `name` e `projects` da string JSON, use a função `json_extract` como no exemplo a seguir. A função `json_extract` utiliza a coluna que contém a string JSON e a pesquisa usando uma expressão como `JSONPath` com a notação `.`

> **Note**
>
> `JSONPath` realiza um transversal de árvore simples. Ele usa o sinal `$` para denotar a raiz do documento JSON, seguido de um ponto final e um elemento aninhado diretamente na raiz, como `$.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
```

O valor retornado é uma string codificada em JSON, e não um tipo de dados nativo do Athena.

```
+-----------------------------------------------------------------------------------------------
+
| name           | projects
         |
+-----------------------------------------------------------------------------------------------
+
```

```
| "Susan Smith"  | [{"name":"project1","completed":false},
{"name":"project2","completed":true}] |
+-----------------------------------------------------------------------------------------------
+
```

Para extrair o valor escalar da string JSON, use a função `json_extract_scalar`. Ele é semelhante a `json_extract`, mas retorna somente valores escalares (Booliano, número ou string).

> **Note**
>
> Não use a função `json_extract_scalar` em matrizes, mapas ou structs.

```
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
```

Essa consulta retorna:

```
+---------------------------+
| name           | projects |
+---------------------------+
| Susan Smith    |          |
+---------------------------+
```

Para obter o primeiro elemento da propriedade `projects` na matriz de exemplo, use a função `json_array_get` e especifique a posição de índice.

```
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
```

Ele retorna o valor na posição de índice especificada na matriz codificada em JSON.

```
+--------------------------------------+
| item                                 |
+--------------------------------------+
| {"name":"project1","completed":false} |
+--------------------------------------+
```

Para retornar um tipo de string do Athena, use o operador `[]` dentro de uma expressão `JSONPath` e use a função `json_extract_scalar`. Para obter mais informações sobre o `[]`, consulte Acessar elementos da matriz.

```
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
```

Ela retorna este resultado:

```
+--------------+
| project_name |
+--------------+
| project1     |
+--------------+
```

Pesquisar valores em matrizes JSON

Para determinar se um valor específico existe dentro de uma matriz codificada em JSON, use a função `json_array_contains`.

A consulta a seguir lista os nomes dos usuários que estão participando de "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')
```

Essa consulta retorna uma lista de usuários.

```
+-------------+
| user        |
+-------------+
| Susan Smith |
+-------------+
| Jane Smith  |
+-------------+
```

O exemplo de consulta a seguir lista os nomes de usuários que concluíram projetos com o número total de projetos realizados. Ele realiza estas ações:

- Usa instruções `SELECT` aninhadas para fins de clareza.
- Extrai a matriz de projetos.
- Converte a matriz em uma matriz nativa de pares de chave/valor usando `CAST`.
- Extrai cada elemento de matriz individual usando o operador `UNNEST`.
- Filtra valores obtidos por projetos concluídos e os conta.

> **Note**
>
> Ao usar `CAST` em `MAP`, você pode especificar o elemento de chave como `VARCHAR` (String nativa no Presto), mas deixar o valor como JSON, porque os valores no `MAP` são de tipos diferentes: string para o primeiro par de chave/valor e Booliano para o segundo.

```
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
```

Esta consulta retorna o seguinte resultado:

```
+----------------------------------+
| name        | completed_projects |
+----------------------------------+
| Susan Smith | 2                  |
+----------------------------------+
| Jane Smith  | 1                  |
+----------------------------------+
```

Obter comprimento e tamanho de matrizes JSON

Exemplo: **`json_array_length`**

Para obter o tamanho de uma matriz codificada em JSON, use a função `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
```

Essa consulta retorna este resultado:

```
+----------------------+
| name        | count |
+----------------------+
| Susan Smith | 2     |
+----------------------+
| Bob Smith   | 1     |
+----------------------+
| Jane Smith  | 1     |
+----------------------+
```

Exemplo: **json_size**

Para obter o tamanho de uma matriz ou de um objeto codificado em JSON, use a função `json_size` e especifique a coluna que contém a string JSON e a expressão `JSONPath` para a matriz ou o objeto.

```
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
```

Essa consulta retorna este resultado:

```
+----------------------+
| name        | count  |
+----------------------+
| Susan Smith | 2      |
+----------------------+
| Bob Smith   | 1      |
+----------------------+
| Jane Smith  | 1      |
+----------------------+
```

Solucionar problemas de consultas JSON

Para obter ajuda sobre como solucionar problemas com consultas relacionadas ao JSON, leia Erros relacionados ao JSON ou acesse os seguintes recursos:

- Recebo mensagens de erro ao tentar ler dados JSON no Amazon Athena
- Como resolver o erro “HIVE_CURSOR_ERROR: A linha não é um objeto JSON válido - JSONException: Chave duplicada” ao ler arquivos do AWS Config no Athena?)
- A consulta SELECT COUNT no Amazon Athena retorna somente um registro, embora o arquivo JSON de entrada tenha vários registros
- Como posso ver o arquivo de origem do Amazon S3 para uma linha em uma tabela do Athena?

Consulte também Considerações e limitações das consultas SQL no Amazon Athena.

Usar Machine Learning (ML) com o Amazon Athena

O Machine Learning (ML) com Amazon Athena permite que você use o Athena para escrever instruções SQL que executam inferências de Machine Learning (ML) por meio do Amazon SageMaker. Esse recurso simplifica o acesso a modelos de ML para análise de dados, eliminando a necessidade de usar métodos de programação complexos para executar inferências.

Para usar ML com Athena, você define uma função ML com Athena usando a cláusula `USING EXTERNAL FUNCTION`. A função aponta para o endpoint do modelo do SageMaker que você deseja usar e especifica os nomes das variáveis e os tipos de dados para transmitir ao modelo. As cláusulas subsequentes na consulta fazem referência à função para passar os valores para o modelo. O modelo executa inferências com base nos valores que a consulta passa e retorna os resultados da inferência. Para obter mais informações sobre o SageMaker e como os endpoints dele funcionam, consulte o Guia do desenvolvedor do Amazon SageMaker.

Para ver um exemplo que usa ML com inferência do Athena e do SageMaker para detectar um valor anômalo em um conjunto de resultados, consulte o artigo no blog de big data da AWS: Detectar valores anômalos invocando a função de inferência de machine learning do Amazon Athena (em inglês).

Considerações e limitações

- Regiões disponíveis: o recurso de ML do Athena é um recurso das Regiões da AWS compatível para a versão 2 ou posteriores do mecanismo do Athena.
- O endpoint do modelo do SageMaker deve aceitar e retornar **`text/csv`**: para obter mais informações sobre formatos de dados, consulte Common data formats for inference (Formatos de dados comuns para inferência) no Guia do desenvolvedor do Amazon SageMaker.
- O Athena não envia cabeçalhos CSV: se seu endpoint do SageMaker for `text/csv`, o manipulador de entrada não deverá presumir que a primeira linha da entrada seja um cabeçalho CSV. Como o Athena não envia cabeçalhos CSV, a saída retornada ao Athena conterá uma linha a menos do que o esperado pelo Athena e causará um erro.
- Escalabilidade do endpoint do SageMaker: garanta que a escala do endpoint do modelo do SageMaker mencionado seja aumentada na vertical o suficiente para chamadas do Athena ao endpoint. Para obter mais informações, consulte Automatically scale SageMaker models (Escalar modelos do SageMaker automaticamente) no Guia do desenvolvedor do Amazon SageMaker e CreateEndpointConfig na Referência de API do Amazon SageMaker.

- Permissões do IAM: para executar uma consulta que especifica uma função ML com Athena, o principal do IAM que executa a consulta deve ter permissão para executar a ação `sagemaker:InvokeEndpoint` no endpoint do modelo do SageMaker mencionado. Para ter mais informações, consulte Permitir acesso para ML com o Athena.
- Não é possível usar as funções ML com Athena diretamente nas cláusulas **`GROUP BY`**

Sintaxe de ML com Athena

A cláusula `USING EXTERNAL FUNCTION` especifica uma função ML com Athena ou várias funções que podem ser referenciadas por uma instrução `SELECT` subsequente na consulta. Você define o nome da função, os nomes das variáveis e os tipos de dados das variáveis e dos valores de retorno.

Resumo

A sintaxe a seguir mostra uma cláusula `USING EXTERNAL FUNCTION` que especifica uma função ML com Athena.

```
USING EXTERNAL FUNCTION ml_function_name (variable1 data_type[, variable2 data_type]
[,...])
RETURNS data_type
SAGEMAKER 'sagemaker_endpoint'
SELECT ml_function_name()
```

Parâmetros

USING EXTERNAL FUNCTION ***`ml_function_name`*** (***`variable1 data_type`***[, ***`variable2 data_type`***][,...])

`ml_function_name` define o nome da função, que pode ser usada nas cláusulas de consultas subsequentes. Cada *`variable data_type`* especifica uma variável nomeada e o tipo de dados correspondente que o modelo do SageMaker aceita como entrada. O tipo de dados especificado deve ser um permitido pelo Athena.

RETURNS ***`tipo_dados`***

`data_type` especifica o tipo de dados SQL que o *`ml_function_name`* retorna para a consulta como saída do modelo do SageMaker.

SAGEMAKER '***`sagemaker_endpoint`***'

`sagemaker_endpoint` especifica o endpoint do modelo do SageMaker.

SELECT [...] ***ml_function_name***(***expression***) [...]

A consulta SELECT que passa os valores para as variáveis de função e o modelo do SageMaker para retornar um resultado. *ml_function_name* especifica a função que já foi definida na consulta, seguida de uma *expression* que é avaliada para passar os valores. Os valores que são passados e retornados devem coincidir com os tipos de dados correspondentes especificados para a função na cláusula `USING EXTERNAL FUNCTION`.

Exemplo

O exemplo a seguir demonstra uma consulta que usa ML com 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;
```

Exemplos de uso de clientes

Os vídeos a seguir, que usam a versão de previsualização do Machine Learning (ML) com o Amazon Athena, mostram como você pode usar o SageMaker com o Athena.

Prever a rotatividade de clientes

O vídeo a seguir mostra como combinar o Athena com os recursos de machine learning do Amazon SageMaker para prever a rotatividade de clientes.

Predict customer churn using Amazon Athena and Amazon SageMaker (Prever a rotatividade de clientes usando o Amazon Athena e o Amazon SageMaker)

Detectar botnets

O vídeo a seguir mostra como uma empresa usa o Amazon Athena e o Amazon SageMaker para detectar botnets.

Detect botnets using Amazon Athena and Amazon SageMaker (Detectar botnes usando o Amazon Athena e o Amazon SageMaker)

Executar consultas com funções definidas pelo usuário

As funções definidas pelo usuário (UDFs) no Amazon Athena permitem criar funções personalizadas para processar registros ou grupos de registros. Uma UDF aceita parâmetros, executa o trabalho e retorna um resultado.

Para usar uma UDF no Athena, escreva uma cláusula `USING EXTERNAL FUNCTION` antes de uma instrução `SELECT` em uma consulta SQL. A instrução `SELECT` faz referência à UDF e define as variáveis que são passadas para a UDF quando a consulta é executada. A consulta SQL invoca uma função do Lambda usando o runtime Java ao chamar a UDF. As UDFs são definidas dentro da função do Lambda como métodos em um pacote de implantação Java. É possível definir várias UDFs no mesmo pacote de implantação Java para uma função do Lambda. Você também especifica o nome da função do Lambda na cláusula `USING EXTERNAL FUNCTION`.

Você tem duas opções para implantar uma função do Lambda para UDFs do Athena. É possível implantar a função diretamente no Lambda ou usar o AWS Serverless Application Repository. Para encontrar as funções do Lambda existentes para UDFs, pesquise no AWS Serverless Application Repository público ou em seu repositório privado e implante-as no Lambda. Você também pode criar ou modificar o código-fonte Java, empacotá-lo em um arquivo JAR e implantá-lo usando o Lambda ou o AWS Serverless Application Repository. Para ver exemplos de código-fonte e pacotes Java para você começar, consulte Criar e implantar uma UDF usando o Lambda. Para obter mais informações sobre o Lambda, consulte o Guia do desenvolvedor do AWS Lambda. Para ter mais informações sobre o AWS Serverless Application Repository, consulte o Guia do desenvolvedor do AWS Serverless Application Repository.

Para ver um exemplo que usa UDFs com o Athena para traduzir e analisar texto, consulte o artigo no blog do Machine Learning da AWS: Traduzir e analisar texto usando funções SQL com Amazon Athena, Amazon Translate e Amazon Comprehend (em inglês) ou assista a video.

Para ver um exemplo de uso de UDFs para estender consultas geoespaciais no Amazon Athena, consulte Estenda consultas geoespaciais no Amazon Athena com UDFs e AWS Lambda no AWSBlog de Big Data.

Considerações e limitações

- Funções integradas do Athena: as funções integradas no Athena são desenvolvidas para ter alta performance. Recomendamos usar as funções integradas em lugar das UDFs quando possível. Para obter mais informações sobre funções integradas, consulte Funções no Amazon Athena.

- Somente UDFs escalares: o Athena aceita somente UDFs escalares, que processam uma linha por vez e retornam um único valor de coluna. O Athena passa um lote de linhas, possivelmente em paralelo, para a UDF sempre que invoca o Lambda. Ao desenvolver UDFs e consultas, esteja ciente do possível impacto desse processamento no tráfego de rede.
- As funções do manipulador UDF usam o formato abreviado: use o formato abreviado (não o formato completo), para suas funções UDF (por exemplo, `package.Class` em vez de `package.Class::method`).
- Os métodos UDF devem estar em letras minúsculas: os métodos UDF devem estar em letras minúsculas. Não é permitida a combinação de maiúsculas e minúsculas.
- Os métodos UDF exigem parâmetros: os métodos UDF devem ter pelo menos um parâmetro de entrada. A tentativa de invocar um UDF definido sem parâmetros de entrada causa uma exceção de runtime. Os UDFs são destinados a executar funções em registros de dados, mas um UDF sem argumentos não recebe dados, então ocorre uma exceção.
- Suporte ao runtime do Java: atualmente, as UDFs do Athena são compatíveis com os runtimes Java 8 e Java 11 para Lambda. Para obter mais informações, consulte Construir funções do Lambda com Java no Guia do desenvolvedor do AWS Lambda.
- Permissões do IAM: para executar e criar instruções de consulta de UDF no Athena, o principal do IAM que executa a consulta deve ter permissão para executar ações além das funções do Athena. Para ter mais informações, consulte Exemplo de políticas de permissões do IAM para permitir funções definidas pelo usuário (UDF) do Amazon Athena.
- Cotas do Lambda: as cotas do Lambda se aplicam às UDFs. Para obter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do AWS Lambda.
- Filtragem em nível de linha – A filtragem em nível de linha do Lake Formation não é compatível com UDFs.
- Visualizações: não é possível usar visualizações com UDFs.
- Problemas conhecidos: para acessar a lista mais recente de problemas conhecidos, consulte Limitations and issues (Limitações e problemas) na seção awslabs/aws-athena-query-federation do GitHub.

Vídeos

Assista aos vídeos a seguir para saber mais como usar as UDFs no Athena.

Vídeo: Apresentação das funções definidas pelo usuário (UDFs) no Amazon Athena

O vídeo a seguir mostra como você pode usar as UDFs no Amazon Athena para editar informações confidenciais.

> **Note**
>
> A sintaxe neste vídeo é pré-lançamento, mas os conceitos são os mesmos. Usar o Athena sem o grupo de trabalho `AmazonAthenaPreviewFunctionality`.

Introducing user defined functions (UDFs) in Amazon Athena (Apresentação de funções definidas pelo usuário (UDFs) no Amazon Athena)

Vídeo: Traduzir, analisar e editar campos de texto usando consultas SQL no Amazon Athena

O vídeo a seguir mostra como você pode usar as UDFs no Amazon Athena junto com outros Serviços da AWS para traduzir e analisar texto.

> **Note**
>
> A sintaxe neste vídeo é pré-lançamento, mas os conceitos são os mesmos. Para obter a sintaxe correta, consulte a publicação relacionada no blog: Traduzir, editar e analisar texto usando funções SQL com Amazon Athena, Amazon Translate e Amazon Comprehend (em inglês) no blog do Machine Learning da AWS.

Traduzir, analisar e editar campos de texto usando consultas SQL no Amazon Athena

Sintaxe de consulta de UDFs

A cláusula `USING EXTERNAL FUNCTION` especifica uma UDF ou várias UDFs que podem ser referenciadas por uma instrução `SELECT` subsequente na consulta. Você precisa do nome do método da UDF e do nome da função do Lambda que hospeda a UDF. No lugar do nome da função do Lambda, você pode usar o ARN do Lambda. Em cenários entre contas, o ARN do Lambda é obrigatório.

Resumo

```
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)] [...]
```

Parâmetros

USING EXTERNAL FUNCTION ***UDF_name***(***variable1 data_type***[, ***variable2 data_type***][,...])

UDF_name especifica o nome da UDF, que deve corresponder a um método Java com a função do Lambda referenciada. Cada *variable data_type* especifica uma variável nomeada e o tipo de dados correspondente que a UDF aceita como entrada. O *data_type* deve ser um dos tipos de dados do Athena permitidos listados na tabela a seguir e mapear para o tipo de dados Java correspondente.

Tipo de dados do Athena	Tipo de dados Java
TIMESTAMP	java.time.LocalDateTime (UTC)
DATA	java.time.LocalDate (UTC)
TINYINT	java.lang.Byte
SMALLINT	java.lang.Short
REAL	java.lang.Float
DOUBLE	java.lang.Double
DECIMAL (veja a nota RETURNS)	java.math.BigDecimal
BIGINT	java.lang.Long
INTEGER	java.lang.Int
VARCHAR	java.lang.String

Tipo de dados do Athena	Tipo de dados Java
VARBINARY	byte[]
BOOLEAN	java.lang.Boolean
ARRAY	java.util.List
ROW	java.util.Map<String, Object>

RETURNS ***tipo_dados***

O `data_type` especifica o tipo de dados SQL que a UDF retorna como saída. Os tipos de dados do Athena listados na tabela acima são permitidos. Para o tipo de dados `DECIMAL`, use a sintaxe `RETURNS DECIMAL(`*`precision`*`, `*`scale`*`)` em que *`precision`* e *`scale`* são inteiros.

LAMBDA '***lambda_function***'

`lambda_function` especifica o nome da função do Lambda que será invocada ao executar a UDF.

SELECT [...] ***UDF_name***(***expression***) [...]

A consulta `SELECT` que passa os valores para a UDF e retorna um resultado. *`UDF_name`* especifica a UDF que será usada, seguida de uma *`expressão`* que é avaliado para passar os valores. Os valores que são passados e retornados devem coincidir com os tipos de dados correspondentes especificados para a UDF na cláusula `USING EXTERNAL FUNCTION`.

Exemplos

Para ver exemplos de consultas baseadas no código AthenaUDFHandler.java no GitHub, consulte a página Amazon Athena UDF Connector (Conector de UDF do Amazon Athena).

Criar e implantar uma UDF usando o Lambda

Para criar uma UDF personalizada, crie uma nova classe Java ao estender a classe `UserDefinedFunctionHandler`. O código-fonte para UserDefinedFunctionHandler.java no SDK está disponível no GitHub no repositório awslabs/aws-athena-query-federation/athena-federation-sdk,

junto com um exemplo de implementações de UDF que você pode analisar e modificar para criar uma UDF personalizada.

As etapas nessa seção demonstram a criação e compilação de um arquivo Jar da UDF usando Apache Maven da linha de comando e uma implantação.

Etapas para criar uma UDF personalizada para Athena usando o Maven

- Clonar o SDK e preparar o ambiente de desenvolvimento
- Criar um projeto Maven
- Adicionar dependências e plugins ao projeto Maven
- Escrever código Java para UDFs
- Compilar o arquivo JAR
- Implantar o JAR no AWS Lambda

Clonar o SDK e preparar o ambiente de desenvolvimento

Antes de começar, certifique-se de que o git esteja instalado no sistema usando `sudo yum install git -y`.

Como instalar o Query Federation SDK da AWS

- Digite o seguinte na linha de comando para clonar o repositório do SDK. Este repositório inclui o SDK, exemplos e um conjunto de conectores de fonte de dados. Para obter mais informações sobre conectores de fonte de dados, consulte Usar a consulta federada do Amazon Athena.

```
git clone https://github.com/awslabs/aws-athena-query-federation.git
```

Como instalar pré-requisitos para este procedimento

Se você trabalha em uma máquina de desenvolvimento que já tem o Apache Maven, a AWS CLI e a ferramenta de compilação do AWS Serverless Application Model instalados, pode ignorar esta etapa.

1. Na raiz do diretório `aws-athena-query-federation` criado ao executar a clonagem, execute o script prepare_dev_env.sh que prepara o ambiente de desenvolvimento.
2. Atualize o shell para fornecer novas variáveis criadas pelo processo de instalação ou reinicie a sessão do terminal.

```
source ~/.profile
```

> **Important**
>
> Se você ignorar essa etapa, receberá erros posteriormente sobre a AWS CLI ou a ferramenta de compilação do AWS SAM não conseguir publicar a função do Lambda.

Criar um projeto Maven

Execute o seguinte comando para criar o projeto Maven. Substitua *groupId* pelo ID exclusivo da organização e substitua *my-athena-udf* pelo nome do aplicativo. Para obter mais informações, consulte Como criar meu primeiro projeto Maven? na documentação Apache Maven.

```
mvn -B archetype:generate \
-DarchetypeGroupId=org.apache.maven.archetypes \
-DgroupId=groupId \
-DartifactId=my-athena-udfs
```

Adicionar dependências e plugins ao projeto Maven

Adicione as configurações a seguir ao arquivo `pom.xml` do projeto Maven. Para ver um exemplo, consulte o arquivo pom.xml no 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>
```

Escrever código Java para UDFs

Crie uma nova classe ao estender UserDefinedFunctionHandler.java. Escreva as UDFs dentro da classe.

No exemplo a seguir, dois métodos Java para UDFs, `compress()` e `decompress()`, são criados dentro da classe `MyUserDefinedFunctions`.

```
*package *com.mycompany.athena.udfs;

public class MyUserDefinedFunctions
        extends UserDefinedFunctionHandler
{
    private static final String SOURCE_TYPE = "MyCompany";

    public MyUserDefinedFunctions()
```

```
    {
        super(SOURCE_TYPE);
    }

    /**
     * Compresses a valid UTF-8 String using the zlib compression library.
     * Encodes bytes with Base64 encoding scheme.
     *
     * @param input the String to be compressed
     * @return the compressed String
     */
    public String compress(String input)
    {
        byte[] inputBytes = input.getBytes(StandardCharsets.UTF_8);

        // create compressor
        Deflater compressor = new Deflater();
        compressor.setInput(inputBytes);
        compressor.finish();

        // compress bytes to output stream
        byte[] buffer = new byte[4096];
        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(inputBytes.length);
        while (!compressor.finished()) {
            int bytes = compressor.deflate(buffer);
            byteArrayOutputStream.write(buffer, 0, bytes);
        }

        try {
            byteArrayOutputStream.close();
        }
        catch (IOException e) {
            throw new RuntimeException("Failed to close ByteArrayOutputStream", e);
        }

        // return encoded string
        byte[] compressedBytes = byteArrayOutputStream.toByteArray();
        return Base64.getEncoder().encodeToString(compressedBytes);
    }

    /**
     * Decompresses a valid String that has been compressed using the zlib compression
library.
```

```
     * Decodes bytes with Base64 decoding scheme.
     *
     * @param input the String to be decompressed
     * @return the decompressed String
     */
    public String decompress(String input)
    {
        byte[] inputBytes = Base64.getDecoder().decode((input));

        // create decompressor
        Inflater decompressor = new Inflater();
        decompressor.setInput(inputBytes, 0, inputBytes.length);

        // decompress bytes to output stream
        byte[] buffer = new byte[4096];
        ByteArrayOutputStream byteArrayOutputStream = new
 ByteArrayOutputStream(inputBytes.length);
        try {
            while (!decompressor.finished()) {
                int bytes = decompressor.inflate(buffer);
                if (bytes == 0 && decompressor.needsInput()) {
                    throw new DataFormatException("Input is truncated");
                }
                byteArrayOutputStream.write(buffer, 0, bytes);
            }
        }
        catch (DataFormatException e) {
            throw new RuntimeException("Failed to decompress string", e);
        }

        try {
            byteArrayOutputStream.close();
        }
        catch (IOException e) {
            throw new RuntimeException("Failed to close ByteArrayOutputStream", e);
        }

        // return decoded string
        byte[] decompressedBytes = byteArrayOutputStream.toByteArray();
        return new String(decompressedBytes, StandardCharsets.UTF_8);
    }
}
```

Compilar o arquivo JAR

Execute `mvn clean install` para compilar o projeto. Após a compilação com êxito, um arquivo JAR é criado na pasta `target` do projeto nomeado como *`artifactId`*`-`*`version`*`.jar`, no qual *`artifactId`* é o nome fornecido por você no projeto Maven, por exemplo, `my-athena-udfs`.

Implantar o JAR no AWS Lambda

Você tem duas opções para implantar o código no Lambda:

- Implantar usando AWS Serverless Application Repository (recomendado)
- Criar uma função do Lambda do arquivo JAR

Opção 1: implantar no AWS Serverless Application Repository

Ao implantar o arquivo JAR no AWS Serverless Application Repository, você cria um arquivo YAML de modelo do AWS SAM que representa a arquitetura do aplicativo. Em seguida, você especifica esse arquivo YAML e um bucket do Amazon S3 no qual os artefatos da aplicação são carregados e disponibilizados para o AWS Serverless Application Repository. O procedimento abaixo usa o script publish.sh localizado no diretório `athena-query-federation/tools` do Athena Query Federation SDK clonado anteriormente.

Para obter mais informações e conhecer os requisitos, consulte Publicar aplicações no Guia do desenvolvedor do AWS Serverless Application Repository, Conceitos de modelo do AWS SAM no Guia do desenvolvedor do AWS Serverless Application Model e Publicar aplicações com tecnologia sem servidor usando a CLI do AWS SAM.

O exemplo a seguir demonstra parâmetros em um arquivo YAML. Adicione parâmetros similares ao arquivo YAML e salve-o no diretório de projetos. Consulte athena-udf.yaml no GitHub para ver o exemplo completo.

```
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
```

Copie o script `publish.sh` para o diretório de projetos no qual você salvou o arquivo YAML e execute o comando:

```
./publish.sh MyS3Location MyYamlFile
```

Por exemplo, se o local do bucket for `s3://DOC-EXAMPLE-BUCKET/mysarapps/athenaudf` e o arquivo YAML for salvo como `my-athena-udfs.yaml`:

```
./publish.sh DOC-EXAMPLE-BUCKET/mysarapps/athenaudf my-athena-udfs
```

Criar uma função do Lambda

1. Abra o console do Lambda em https://console.aws.amazon.com/lambda/, escolha Create function (Criar função) e selecione Browse serverless app repository (Procurar repositório de aplicações sem servidor)

2. Selecione Private applications (Aplicativos privados), localize o aplicativo na lista ou procure usando palavras-chave e selecione-o.
3. Verifique, forneça detalhes do aplicativo e selecione Deploy (Implantar).

   Agora você pode usar os nomes dos métodos definidos no arquivo JAR da função do Lambda como UDFs no Athena.

Opção 2: criar uma função do Lambda diretamente

Você também pode criar uma função do Lambda diretamente usando o console ou a AWS CLI. O exemplo a seguir demonstra o uso do comando da CLI `create-function` do Lambda.

```
aws lambda create-function \
 --function-name MyLambdaFunctionName \
 --runtime java8 \
 --role arn:aws:iam::1234567890123:role/my_lambda_role \
 --handler com.mycompany.athena.udfs.MyUserDefinedFunctions \
 --timeout 900 \
 --zip-file fileb://./target/my-athena-udfs-1.0-SNAPSHOT.jar
```

Realizar consultas entre regiões

O Athena oferece suporte à consulta de dados do Amazon S3 em uma Região da AWS diferente da região na qual você está usando o Athena. A consulta entre regiões pode ser uma boa opção quando não é prático ou permitido mover dados, ou quando você deseja consultar dados em várias regiões. Mesmo que o Athena não esteja disponível em uma determinada região, será possível consultar os dados nela a partir de outra região na qual o Athena esteja disponível.

Para consultar dados em uma região, sua conta deve estar habilitada nessa região, mesmo que os dados do Amazon S3 não pertençam à sua conta. Para algumas regiões, como Leste dos EUA (Ohio), seu acesso à região é habilitado automaticamente quando sua conta é criada. Outras regiões exigem a opção de inclusão (status opt-in) da conta antes que você possa usá-la. Para obter a lista de regiões que exigem o status opt-in, consulte Regiões disponíveis no Guia do usuário do Amazon EC2 para instâncias do Linux. Para obter instruções específicas sobre como optar por uma região, consulte Gerenciar regiões da AWS no Referência geral da Amazon Web Services.

Considerações e limitações

- Permissões de acesso a dados: para consultar com êxito dados do Amazon S3 no Athena entre regiões, sua conta deve ter permissões para ler os dados. Se os dados que você deseja consultar pertencerem a outra conta, essa outra conta deverá lhe conceder acesso ao local do Amazon S3 que contém os dados.
- Cobranças de transferência de dados: aplicam-se cobranças de transferência de dados do Amazon S3 para consultas entre regiões. A execução de uma consulta pode fazer com que mais dados sejam transferidos do que o tamanho do conjunto de dados. É recomendável que você comece testando suas consultas em um subconjunto de dados e analisando os custos no AWS Cost Explorer.
- AWS Glue: você pode usar o AWS Glue entre regiões. Podem ser aplicadas cobranças adicionais para tráfego do AWS Glue entre regiões. Para obter mais informações, consulte Criar conexões do AWS Glue entre contas e entre regiões no AWS Big Data Blog.
- Opções de criptografia do Amazon S3: há suporte para as opções de criptografia SSE-S3 e SSE-KMS nas consultas entre regiões. Não há suporte para CSE-KMS. Para ter mais informações, consulte Opções de criptografia permitidas do Amazon S3.
- Consultas federadas: não há suporte para o uso de consultas federadas em Regiões da AWS.
- Regiões da China: não há suporte para consultas entre regiões nas regiões da China.

Desde que as condições acima sejam atendidas, você pode criar uma tabela do Athena que aponte para o valor `LOCATION` especificado e consultar os dados de forma transparente. Não é necessária nenhuma sintaxe especial. Para obter mais informações sobre criação de tabelas, consulte Criar tabelas no Athena.

Consulta no AWS Glue Data Catalog

Como o AWS Glue Data Catalog é usado por muitos Serviços da AWS como repositório central de metadados, você pode consultar metadados do catálogo de dados. Para fazer isso, use as consultas SQL no Athena. Você pode usar o Athena para consultar os metadados do catálogo do AWS Glue, como bancos de dados, tabelas, partições e colunas.

Para acessar os metadados do catálogo do AWS Glue, consulte o banco de dados `information_schema` no backend do Athena. As consultas de exemplo neste tópico mostram como usar o Athena para consultar os metadados do catálogo do AWS Glue em casos de uso comuns.

Tópicos

- Considerações e limitações
- Listar bancos de dados e pesquisar em um banco de dados especificado
- Listar tabelas em um banco de dados especificado e pesquisar uma tabela por nome
- Listar partições de uma tabela específica
- Listagem de todas as colunas de todas as tabelas
- Listagem das colunas que tabelas específicas têm em comum
- Listar ou pesquisar colunas de uma tabela ou visualização especificada

Considerações e limitações

- Em vez de consultar o banco de dados `information_schema`, é possível usar comandos DDL individuais do Apache Hive para extrair informações de metadados para bancos de dados, tabelas, exibições, partições e colunas específicos do Athena. No entanto, a saída está em um formato não tabular.
- As consultas de `information_schema` apresentam melhor performance se você tiver uma quantidade pequena a moderada de metadados do AWS Glue. Pode haver erros se você tiver uma grande quantidade de metadados.
- Não é possível usar `CREATE VIEW` para criar uma exibição no banco de dados `information_schema`.

Listar bancos de dados e pesquisar em um banco de dados especificado

Os exemplos nesta seção mostram como listar os bancos de dados em metadados por nome de esquema.

Example – Listar bancos de dados

A consulta de exemplo a seguir lista os bancos de dados da tabela `information_schema.schemata`.

```
SELECT schema_name
FROM   information_schema.schemata
LIMIT  10;
```

A tabela a seguir exibe os resultados do exemplo.

6	alb-databas1
7	alb_original_cust
8	alblogsdatabase
9	athena_db_test
10	athena_ddl_db

Example – Pesquisar em um banco de dados especificado

Na consulta de exemplo a seguir, `rdspostgresql` é um banco de dados de exemplo.

```
SELECT schema_name
FROM   information_schema.schemata
WHERE  schema_name = 'rdspostgresql'
```

A tabela a seguir exibe os resultados do exemplo.

	schema_name
1	rdspostgresql

Listar tabelas em um banco de dados especificado e pesquisar uma tabela por nome

Para listar os metadados de tabelas, você pode consultar um esquema ou nome de tabela.

Example – Listar tabelas por esquema

A consulta a seguir lista tabelas que usam o esquema de tabela `rdspostgresql`.

```
SELECT table_schema,
       table_name,
       table_type
FROM   information_schema.tables
WHERE  table_schema = 'rdspostgresql'
```

A tabela a seguir mostra um exemplo de resultado.

	table_schema	table_name	table_type
1	rdspostgresql	rdspostgresqldb1_public_account	BASE TABLE

Example – Pesquisar uma tabela por nome

A consulta a seguir obtém informações de metadados para a tabela `athena1`.

```
SELECT table_schema,
       table_name,
       table_type
FROM   information_schema.tables
WHERE  table_name = 'athena1'
```

A tabela a seguir mostra um exemplo de resultado.

	table_schema	table_name	table_type
1	padrão	athena1	BASE TABLE

Listar partições de uma tabela específica

Você pode usar `SHOW PARTITIONS` *`table_name`* para listar as partições de uma tabela especificada, como no exemplo a seguir.

```
SHOW PARTITIONS cloudtrail_logs_test2
```

Você também pode usar uma consulta de metadados `$partitions` para listar os números e os valores de partição de uma tabela específica.

Example — Consultar as partições de uma tabela usando a sintaxe $partitions

A consulta de exemplo a seguir lista as partições da tabela `cloudtrail_logs_test2` usando a sintaxe `$partitions`.

```
SELECT * FROM default."cloudtrail_logs_test2$partitions" ORDER BY partition_number
```

A tabela a seguir exibe os resultados do exemplo.

	table_catalog	table_schema	table_name	Ano	Mês	Dia
1	awsdatacatalog	padrão	cloudtrail_logs_test2	2020	08	10
2	awsdatacatalog	padrão	cloudtrail_logs_test2	2020	08	11
3	awsdatacatalog	padrão	cloudtrail_logs_test2	2020	08	12

Listagem de todas as colunas de todas as tabelas

É possível listar todas as colunas de todas as tabelas no `AwsDataCatalog` ou para todas as tabelas em um banco de dados específico no `AwsDataCatalog`.

- Para listar todas as colunas de todos os bancos de dados no `AwsDataCatalog`, use a consulta `SELECT * FROM information_schema.columns`.
- Para restringir os resultados a um banco de dados específico, use `table_schema='`*`database_name`*`'` na cláusula `WHERE`.

Example — Listagem de todas as colunas de todas as tabelas em um banco de dados específico

A consulta de exemplo a seguir lista todas as colunas de todas as tabelas no banco de dados `webdata`.

```
SELECT * FROM information_schema.columns WHERE table_schema = 'webdata'
```

Listagem das colunas que tabelas específicas têm em comum

É possível listar as colunas que tabelas específicas têm em comum em um banco de dados.

- Use a sintaxe `SELECT column_name FROM information_schema.columns`.

- Para a cláusula `WHERE`, use a sintaxe `WHERE table_name IN ('table1', 'table2')`.

Example : listagem de colunas comuns para duas tabelas no mesmo banco de dados

O exemplo de consulta a seguir lista as colunas que as tabelas `table1` e `table2` têm em comum.

```
SELECT column_name
FROM information_schema.columns
WHERE table_name IN ('table1', 'table2')
GROUP BY column_name
HAVING COUNT(*) > 1;
```

Listar ou pesquisar colunas de uma tabela ou visualização especificada

Você pode listar todas as colunas de uma tabela, todas as colunas de uma exibição ou pesquisar uma coluna por nome em um banco de dados e tabela especificados.

Para listar as colunas, use uma consulta `SELECT *`. Na cláusula `FROM`, especifique `information_schema.columns`. Na cláusula `WHERE`, use `table_schema='database_name'` para especificar o banco de dados e `table_name = 'table_name'` para especificar a tabela ou a visualização que tem as colunas que você deseja listar.

Example – Listar todas as colunas de uma tabela especificada

A consulta de exemplo a seguir lista todas as colunas da tabela `rdspostgresqldb1_public_account`.

```
SELECT *
FROM   information_schema.columns
WHERE  table_schema = 'rdspostgresql'
       AND table_name = 'rdspostgresqldb1_public_account'
```

A tabela a seguir exibe os resultados do exemplo.

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_t	comento	extra_info
1	awsdatacatalog	rdspostgresql	rdspostgresqldb1_p	passwor	1		SIM	varcha		

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_type	comment	extra_info
			ublic_account							
2	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	user_id	2		SIM	inteiro		
3	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	created_on	3		SIM	timest		
4	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	last_login	4		SIM	timest		
5	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	email	5		SIM	varcha		
6	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	usernam	6		SIM	varcha		

Example – Listar as colunas de uma visualização especificada

A consulta de exemplo a seguir lista todas as colunas no banco de dados `default` para a exibição `arrayview`.

```
SELECT *
FROM   information_schema.columns
WHERE  table_schema = 'default'
```

```
        AND table_name = 'arrayview'
```

A tabela a seguir exibe os resultados do exemplo.

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_type	comment	extra_info
1	awsdatacatalog	padrão	arrayview	searchdate	1		SIM	varchar		
2	awsdatacatalog	padrão	arrayview	sid	2		SIM	varchar		
3	awsdatacatalog	padrão	arrayview	btid	3		SIM	varchar		
4	awsdatacatalog	padrão	arrayview	p	4		SIM	varchar		
5	awsdatacatalog	padrão	arrayview	infantprice	5		SIM	varchar		
6	awsdatacatalog	padrão	arrayview	sump	6		SIM	varchar		
7	awsdatacatalog	padrão	arrayview	journeymaparray	7		SIM	array(varchar)		

Example – Pesquisar uma coluna por nome em uma tabela e um banco de dados especificados

A consulta de exemplo a seguir procura metadados para a coluna `sid` na exibição `arrayview` do banco de dados `default`.

```
SELECT *
FROM   information_schema.columns
WHERE  table_schema = 'default'
       AND table_name = 'arrayview'
       AND column_name='sid'
```

A tabela a seguir mostra um exemplo de resultado.

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_t	comeo	extra_info
1	awsdatacatalog	padrão	arrayview	sid	2		SIM	varcha		

Consulta de logs do AWS service (Serviço da AWS)

Esta seção inclui vários procedimentos para usar o Amazon Athena para consultar conjuntos de dados conhecidos, como os logs do AWS CloudTrail, Amazon CloudFront, Classic Load Balancer, Application Load Balancer, Network Load Balancer, e os logs de fluxo do Amazon VPC.

As tarefas nesta seção usam o console do Athena, mas você também pode usar outras ferramentas como o driver JDBC do Athena, a AWS CLI ou a referência de API do Amazon Athena.

Para obter informações sobre como usar o AWS CloudFormation para criar automaticamente tabelas de log de serviço, partições e consultas de exemplo da AWS service (Serviço da AWS) no Athena, consulte Automatizar a criação de tabelas de logs de serviço da AWS service (Serviço da AWS) e consultá-las com o Amazon Athena no blog de big data da AWS. Para obter informações sobre como usar uma biblioteca Python para o AWS Glue para criar um framework comum de processamento de logs de AWS service (Serviço da AWS) e consultá-los no Athena, consulte Consulta com facilidade de logs de AWS service (Serviço da AWS) usando o Amazon Athena.

Os tópicos nesta seção pressupõem que você configurou as permissões apropriadas para acessar o Athena e o bucket do Amazon S3 no qual os dados a serem consultados devem residir. Para obter mais informações, consulte Configuração e Conceitos básicos.

Tópicos

- Consultar logs do Application Load Balancer
- Consultar logs do Classic Load Balancer
- Consultar os logs do Amazon CloudFront
- Consultar os logs do AWS CloudTrail
- Consultar os logs do Amazon EMR
- Consultar logs de fluxo do AWS Global Accelerator

- Consultar descobertas do Amazon GuardDuty
- Consultar os logs do AWS Network Firewall
- Consultar os logs do Network Load Balancer
- Consultar os logs do Amazon Route 53 Resolver
- Consultar logs de eventos do Amazon SES
- Consultar os logs de fluxo do Amazon VPC
- Consultar os logs do AWS WAF

Consultar logs do Application Load Balancer

O Application Load Balancer é uma opção de balanceamento de carga do Elastic Load Balancing que habilita a distribuição de tráfego em uma implantação de microsserviços usando contêineres. Consultar logs do Application Load Balancer permite consultar a origem do tráfego, a latência e os bytes transferidos de e para instâncias do Elastic Load Balancing e aplicativos de backend. Para obter mais informações, consulte Logs de acesso para seu Application Load Balancer e Logs de conexão para seu Application Load Balancer no Guia do usuário para Application Load Balancers.

Tópicos

- Pré-requisitos
- Como criar a tabela para logs de acesso do ALB
- Como criar a tabela para logs de acesso do ALB no Athena usando a projeção de partições
- Exemplos de consultas para logs de acesso do ALB
- Como criar a tabela para logs de conexão do ALB
- Como criar a tabela para logs de conexão do ALB no Athena usando a projeção de partições
- Exemplo de consultas para logs de conexão do ALB
- Recursos adicionais do

Pré-requisitos

- Habilite o registro de acesso em log ou registro de conexão em log para permitir que os logs do Application Load Balancer sejam salvos no seu bucket do Amazon S3.
- Um banco de dados para conter a tabela que você criará para o Athena. Para criar um banco de dados, é possível usar o Athena ou o console do AWS Glue. Para obter mais informações,

consulte Criar bancos de dados no Athena neste guia ou Trabalhar com banco de dados no console do AWS Glue, no Guia do desenvolvedor do AWS Glue.

Como criar a tabela para logs de acesso do ALB

1. Copie e cole a seguinte instrução `CREATE TABLE` no editor de consultas do console do Athena. Para obter mais informações sobre os conceitos básicos do Athena, consulte Conceitos básicos. Substitua o caminho na cláusula `LOCATION` pela localização da pasta de log de acesso do Amazon S3. Para obter mais informações sobre o local do arquivo de log de acesso, consulte Arquivos de log de acesso no Guia do usuário para Application Load Balancers. Para obter informações sobre cada campo do arquivo de log, consulte Entradas de log de acesso.

> **Note**
>
> A instrução `CREATE TABLE` a seguir inclui as colunas recém-adicionadas `classification`, `classification_reason` e `traceability_id`. Para criar uma tabela dos logs de acesso do Application Load Balancer que não contenham essas entradas, remova as colunas correspondentes da instrução `CREATE TABLE` e modifique a expressão regex de acordo.

```
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,
            traceability_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. Execute a consulta no console do Athena. Depois que a consulta for concluída, o Athena registrará a tabela `alb_access_logs`, preparando os dados dela para você fazer as consultas.

Como criar a tabela para logs de acesso do ALB no Athena usando a projeção de partições

Como os logs de acesso do ALB têm uma estrutura conhecida com um esquema de partição que você pode especificar antecipadamente, é possível reduzir o tempo de execução das consultas e automatizar o gerenciamento de partições usando o recurso de projeção de partições do Athena. A projeção de partições adiciona automaticamente novas partições à medida que os dados são adicionados. Isso elimina a necessidade de adicionar manualmente as partições usando `ALTER TABLE ADD PARTITION`.

O exemplo de instrução `CREATE TABLE` a seguir usa automaticamente a projeção de partições com base em logs de acesso do ALB a partir de uma data especificada até o dia atual para uma única região da AWS. A instrução é baseada no exemplo da seção anterior, mas adiciona as cláusulas `PARTITIONED BY` e `TBLPROPERTIES` para habilitar a projeção de partições. Nas cláusulas `LOCATION` e `storage.location.template`, substitua os espaços reservados pelos valores que identificam o local do bucket do Amazon S3 dos seus logs de acesso do ABL. Para obter mais informações sobre o local do arquivo de log de acesso, consulte Arquivos de log de acesso no Guia do usuário para Application Load Balancers. Em `projection.day.range`, substitua *2022*/*01*/*01* pela data de início desejada. Depois que você executar a consulta com êxito, poderá consultar a tabela. Você não precisa executar `ALTER TABLE ADD PARTITION` para carregar as partições. Para obter informações sobre cada campo do arquivo de log, consulte Entradas de log de acesso.

```
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,
            traceability_id string
            )
            PARTITIONED BY
            (
             day STRING
            )
            ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
            WITH SERDEPROPERTIES (
            'serialization.format' = '1',
            'input.regex' =
        '([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*):([0-9]*) ([^ ]*)[:-]([0-9]*) ([-.0-9]*)
 ([-.0-9]*) ([-.0-9]*) ([-.0-9]*) (|[-0-9]*) (-|[-0-9]*) ([-0-9]*) ([-0-9]*) \"([^ ]*) (.*) (- |
[^ ]*)\" \"([^\"]*)\" ([A-Z0-9-_]+) ([A-Za-z0-9.-]*) ([^ ]*) \"([^\"]*)\" \"([^\"]*)\"
 \"([^\"]*)\" ([-.0-9]*) ([^ ]*) \"([^\"]*)\" \"([^\"]*)\" \"([^ ]*)\" \"([^\s]+?)\"
 \"([^\s]+)\" \"([^ ]*)\" \"([^ ]*)\" ?([^ ]*)?( .*)?')
            LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/<ACCOUNT-NUMBER>/
elasticloadbalancing/<REGION>/'
            TBLPROPERTIES
            (
             "projection.enabled" = "true",
             "projection.day.type" = "date",
             "projection.day.range" = "2022/01/01,NOW",
             "projection.day.format" = "yyyy/MM/dd",
             "projection.day.interval" = "1",
             "projection.day.interval.unit" = "DAYS",
             "storage.location.template" = "s3://DOC-EXAMPLE-BUCKET/AWSLogs/<ACCOUNT-
NUMBER>/elasticloadbalancing/<REGION>/${day}"
            )
```

Para obter mais informações sobre projeção de partições, consulte Projeção de partições com o Amazon Athena.

Exemplos de consultas para logs de acesso do ALB

A consulta a seguir conta o número de solicitações HTTP GET recebidas pelo load balancer agrupadas pelo endereço IP do cliente:

```
SELECT COUNT(request_verb) AS
 count,
```

```
 request_verb,
 client_ip
FROM alb_logs
GROUP BY request_verb, client_ip
LIMIT 100;
```

Outra consulta mostra os URLs visitados por usuários do navegador Safari:

```
SELECT request_url
FROM alb_logs
WHERE user_agent LIKE '%Safari%'
LIMIT 10;
```

A consulta a seguir mostra registros que têm valores de código de status ELB maiores ou iguais a 500.

```
SELECT * FROM alb_logs
WHERE elb_status_code >= 500
```

O exemplo a seguir mostra como analisar os registros por `datetime`:

```
SELECT client_ip, sum(received_bytes)
FROM alb_logs
WHERE parse_datetime(time,'yyyy-MM-dd''T''HH:mm:ss.SSSSSS''Z')
     BETWEEN parse_datetime('2018-05-30-12:00:00','yyyy-MM-dd-HH:mm:ss')
     AND parse_datetime('2018-05-31-00:00:00','yyyy-MM-dd-HH:mm:ss')
GROUP BY client_ip;
```

O exemplo a seguir consulta a tabela que usa projeção de partição para todos os logs do ALB a partir do dia especificado.

```
SELECT *
FROM alb_logs
WHERE day = '2022/02/12'
```

Como criar a tabela para logs de conexão do ALB

1. Copie e cole a seguinte instrução `CREATE TABLE` no editor de consultas do console do Athena. Para obter mais informações sobre os conceitos básicos do Athena, consulte Conceitos básicos. Substitua o caminho na cláusula `LOCATION` pela localização da pasta de log de conexão do

Amazon S3. Para obter mais informações sobre o local do arquivo de log de conexão, consulte Arquivos de log de conexão no Guia do usuário para Application Load Balancers. Para obter informações sobre cada campo do arquivo de log, consulte Entradas de log de conexão.

```
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,
         traceability_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. Execute a consulta no console do Athena. Depois que a consulta for concluída, o Athena registrará a tabela `alb_connection_logs`, preparando os dados dela para você fazer as consultas.

Como criar a tabela para logs de conexão do ALB no Athena usando a projeção de partições

Como os logs de conexão do ALB têm uma estrutura conhecida com um esquema de partição que você pode especificar antecipadamente, é possível reduzir o tempo de execução das consultas e automatizar o gerenciamento de partições usando o recurso de projeção de partições do Athena. A projeção de partições adiciona automaticamente novas partições à medida que os dados são adicionados. Isso elimina a necessidade de adicionar manualmente as partições usando `ALTER TABLE ADD PARTITION`.

O exemplo de instrução `CREATE TABLE` a seguir usa automaticamente a projeção de partições com base em logs de conexão do ALB a partir de uma data especificada até o dia atual para

uma única região da AWS. A instrução é baseada no exemplo da seção anterior, mas adiciona as cláusulas `PARTITIONED BY` e `TBLPROPERTIES` para habilitar a projeção de partições. Nas cláusulas `LOCATION` e `storage.location.template`, substitua os espaços reservados pelos valores que identificam o local do bucket do Amazon S3 dos seus logs de conexão do ABL. Para obter mais informações sobre o local do arquivo de log de conexão, consulte Arquivos de log de conexão no Guia do usuário para Application Load Balancers. Em `projection.day.range`, substitua *2023*/*01*/*01* pela data de início que deseja usar. Depois que você executar a consulta com êxito, poderá consultar a tabela. Você não precisa executar `ALTER TABLE ADD PARTITION` para carregar as partições. Para obter informações sobre cada campo do arquivo de log, consulte Entradas de log de conexão.

```
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,
         traceability_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}"
            )
```

Para obter mais informações sobre projeção de partições, consulte Projeção de partições com o Amazon Athena.

Exemplo de consultas para logs de conexão do ALB

A consulta a seguir conta as ocorrências nas quais o valor de `tls_verify_status` não foi `'Success'`, agrupadas por endereço IP do cliente:

```
SELECT DISTINCT client_ip, count() AS count FROM alb_connection_logs
WHERE tls_verify_status != 'Success'
GROUP BY client_ip
ORDER BY count() DESC;
```

A consulta a seguir pesquisa ocorrências nas quais o valor de `tls_handshake_latency` foi superior a 2 segundos no intervalo de tempo especificado:

```
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);
```

Recursos adicionais do

- Como analisar os logs de acesso do meu Application Load Balancer usando o Amazon Athena na AWSCentral de Conhecimento.

- Para obter informações sobre códigos de status HTTP do Elastic Load Balancing, consulte Solução de problemas em Application Load Balancers no Guia do usuário para Application Load Balancers.
- Catalogue e analise os logs do Application Load Balancer com mais eficiência com classificadores personalizados AWS Glue e Amazon Athena no AWSBlog Big Data.

Consultar logs do Classic Load Balancer

Consulte os logs do Classic Load Balancer para analisar e saber os padrões do tráfego que entra e sai das instâncias e das aplicações de backend do Elastic Load Balancing. Você pode ver a origem do tráfego, da latência e dos bytes que foram transferidos.

Antes de analisar os logs do Elastic Load Balancing, configure-os para que sejam salvos no bucket de destino do Amazon S3. Para obter mais informações, consulte Habilitar os logs de acesso do seu Classic Load Balancer.

- Criar a tabela de logs do Elastic Load Balancing
- Exemplo de consultas do Elastic Load Balancing

Para criar a tabela de logs do Elastic Load Balancing

1. Copie e cole a instrução DDL a seguir no console do Athena. Verifique a sintaxe dos registros de log do Elastic Load Balancing. Pode ser necessário atualizar a consulta a seguir para incluir as colunas e a sintaxe Regex para a versão mais recente do registro.

```
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. Modifique o bucket `LOCATION` do Amazon S3 para especificar o destino dos logs do Elastic Load Balancing.
3. Execute a consulta no console do Athena. Depois que a consulta for concluída, o Athena registrará a tabela `elb_logs`, preparando os dados dela para as consultas. Para ter mais informações, consulte Exemplo de consultas do Elastic Load Balancing.

Exemplo de consultas do Elastic Load Balancing

Use uma consulta semelhante à consulta do exemplo a seguir. Ele lista os servidores de aplicativos de backend que retornaram um código de resposta de erro `4XX` ou `5XX`. Use o operador `LIMIT` para limitar o número de logs a serem consultados por vez.

```
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;
```

Use uma consulta subsequente para somar o tempo de resposta de todas as transações agrupadas por endereço IP de backend e nome da instância do 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;
```

Para obter mais informações, consulte Analyzing data in S3 using Athena (Analisar dados no S3 usando o Athena).

Consultar os logs do Amazon CloudFront

Você pode configurar o CDN do Amazon CloudFront para exportar os logs de acesso de distribuição da Web para o Amazon Simple Storage Service. Use esses logs para explorar os padrões de navegação dos usuários nas propriedades da Web processadas pelo CloudFront.

Antes de começar a consultar os logs, habilite o log de acesso de distribuições da Web de acordo com a sua distribuição preferida do CloudFront. Para obter informações, consulte Logs de acesso no Guia do desenvolvedor do Amazon CloudFront. Anote o bucket do Amazon S3 no qual salva esses logs.

- Como criar uma tabela para logs padrão do CloudFront
- Como criar uma tabela para logs em tempo real do CloudFront
- Exemplo de consultas para logs padrão do CloudFront

Como criar uma tabela para logs padrão do CloudFront

> **Note**
>
> Esse procedimento funciona para os logs de acesso de distribuição da web no CloudFront. Ele não se aplica a logs de streaming de distribuições RTMP.

Para criar uma tabela para campos de arquivo de log padrão do CloudFront

1. Copie e cole o exemplo de instrução DDL a seguir no editor de consultas no console do Athena. O exemplo de instrução usa os campos do arquivo de log documentados na seção Campos de arquivo de log padrão do Guia do usuário do Amazon CloudFront. Modifique o `LOCATION` para o

bucket do Amazon S3 que armazena seus logs. Para obter informações sobre como usar o editor de consultas, acesse Conceitos básicos.

Essa consulta especifica `ROW FORMAT DELIMITED` e `FIELDS TERMINATED BY '\t'` para indicar que os campos são delimitados por caracteres de tabulação. Para `ROW FORMAT DELIMITED`, o Athena usa o LazySimpleSerDe por padrão. A coluna `date` é escapada com acentos graves (`) porque se trata de uma palavra reservada no Athena. Para ter mais informações, consulte Palavras-chave reservadas.

```
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. Execute a consulta no console do Athena. Depois que a consulta for concluída, o Athena registrará a tabela `cloudfront_standard_logs`, preparando os dados dela para você fazer as consultas.

Como criar uma tabela para logs em tempo real do CloudFront

Para criar uma tabela para campos de arquivo de log em tempo real do CloudFront

1. Copie e cole o exemplo de instrução DDL a seguir no editor de consultas no console do Athena. O exemplo de instrução usa os campos do arquivo de log documentados na seção Logs em tempo real do Guia do usuário do Amazon CloudFront. Modifique o `LOCATION` para o bucket do Amazon S3 que armazena seus logs. Para obter informações sobre como usar o editor de consultas, acesse Conceitos básicos.

   Essa consulta especifica `ROW FORMAT DELIMITED` e `FIELDS TERMINATED BY '\t'` para indicar que os campos são delimitados por caracteres de tabulação. Para `ROW FORMAT DELIMITED`, o Athena usa o LazySimpleSerDe por padrão. A coluna `timestamp` é escapada com acentos graves (`) porque se trata de uma palavra reservada no Athena. Para ter mais informações, consulte Palavras-chave reservadas.

   O exemplo a seguir contém todos os campos disponíveis. Você pode comentar ou remover campos que não sejam necessários.

```
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. Execute a consulta no console do Athena. Depois que a consulta for concluída, o Athena registrará a tabela `cloudfront_real_time_logs`, preparando os dados dela para você fazer as consultas.

Exemplo de consultas para logs padrão do CloudFront

A consulta a seguir adiciona o número de bytes processados pelo CloudFront entre 9 e 11 de junho de 2018. Coloque o nome da coluna de data entre aspas duplas porque se trata de uma palavra reservada.

```
SELECT SUM(bytes) AS total_bytes
FROM cloudfront_standard_logs
WHERE "date" BETWEEN DATE '2018-06-09' AND DATE '2018-06-11'
LIMIT 100;
```

Para eliminar linhas duplicadas (por exemplo, linhas vazias duplicadas) dos resultados da consulta, é possível usar a instrução `SELECT DISTINCT`, conforme o exemplo a seguir.

```
SELECT DISTINCT *
FROM cloudfront_standard_logs
LIMIT 10;
```

Recursos adicionais do

Para obter mais informações sobre como usar o Athena para consultar os logs do CloudFront, leia as publicações do blog sobre big data da AWS a seguir.

Consulte facilmente os logs de AWS service (Serviço da AWS) usando o Amazon Athena de 29 de maio de 2019 (em inglês).

Analise seus logs de acesso do Amazon CloudFront em grande escala (21 de dezembro de 2018).

Construa uma arquitetura serverless para analisar os logs de acesso do Amazon CloudFront usando AWS Lambda, Amazon Athena e o Amazon Managed Service for Apache Flink (26 de maio de 2017).

Consultar os logs do AWS CloudTrail

O AWS CloudTrail é um serviço que registra chamadas de API da AWS e eventos de contas da Amazon Web Services.

Os logs do CloudTrail incluem detalhes sobre todas as chamadas de API feitas para seus Serviços da AWS, incluindo o console. O CloudTrail gera arquivos de log criptografados e os armazena no Amazon S3. Para mais informações, consulte o Guia do usuário do AWS CloudTrail.

> **Note**
>
> Se você quiser realizar consultas SQL nas informações de eventos do CloudTrail entre contas, regiões e datas, considere usar o CloudTrail Lake. O CloudTrail Lake é uma alternativa da AWS à criação de trilhas que agregam informações de uma empresa em um único armazenamento pesquisável de dados de eventos. Em vez de usar o armazenamento de bucket do Amazon S3, ele armazena eventos em um data lake, o que permite consultas mais avançadas e rápidas. Você pode usá-lo para criar consultas SQL que pesquisem eventos em organizações, regiões e em períodos temporais personalizados. Como você executa consultas do CloudTrail Lake diretamente no console do CloudTrail, o uso do CloudTrail Lake não requer o Athena. Para obter mais informações, consulte a Documentação do CloudTrail Lake.

O uso do Athena com os logs do CloudTrail é uma ótima maneira de melhorar a análise das atividades da AWS service (Serviço da AWS). Por exemplo, é possível usar consultas para identificar tendências e isolar ainda mais a atividade por atributos, como endereço IP de origem ou usuário.

Uma aplicação comum é usar os logs do CloudTrail para analisar a segurança e a compatibilidade das atividades operacionais. Para obter um exemplo detalhado, consulte a publicação no blog Big Data da AWS, Análise de segurança, conformidade e atividades operacionais usando o AWS CloudTrail e o Amazon Athena.

Você pode usar o Athena para consultar esses arquivos de log diretamente no Amazon S3 especificando o `LOCATION` dos arquivos de log. É possível fazer isso de duas formas:

- Criando tabelas para arquivos de log do CloudTrail diretamente no console do CloudTrail.
- Criando tabelas manualmente para arquivos de log do CloudTrail no console do Athena.

Tópicos

- Noções básicas sobre logs do CloudTrail e tabelas do Athena
- Usar o console do CloudTrail para criar uma tabela do Athena com logs do CloudTrail
- Criar uma tabela de logs do CloudTrail no Athena usando particionamento manual

- Criar uma tabela para uma trilha de toda a organização usando particionamento manual
- Criar a tabela de logs do CloudTrail no Athena usando a projeção de partições
- Consultar campos aninhados
- Consulta de exemplo
- Dicas para consultar logs do CloudTrail

Noções básicas sobre logs do CloudTrail e tabelas do Athena

Antes de começar a criar tabelas, você deve conhecer melhor o CloudTrail e saber como ele armazena os dados. Isso pode ajudar a criar as tabelas necessárias, tanto pelo console do CloudTrail quanto pelo Athena.

O CloudTrail salva os logs como arquivos de texto JSON no formato gzip compactado (*.json.gzip). O local dos arquivos de log depende de como você configura as trilhas, da Região da AWS ou das regiões onde você está registrando, além de outros fatores.

Para obter mais informações de onde os logs são armazenados, da estrutura JSON e do conteúdo do arquivo de registro, consulte os seguintes tópicos no Manual do usuário do AWS CloudTrail:

- Encontrar os arquivos de log do CloudTrail
- Exemplos de arquivos de log do CloudTrail
- Conteúdo do registro do CloudTrail
- Referência de eventos do CloudTrail

Para coletar e salvar os logs no Amazon S3, habilite o CloudTrail no AWS Management Console. Para obter mais informações, consulte Criar uma trilha no Guia do usuário do AWS CloudTrail.

Anote o bucket de destino do Amazon S3 no qual você salva os logs. Substitua a cláusula `LOCATION` pelo caminho para o local dos logs do CloudTrail e pelo conjunto de objetos com os quais trabalhar. O exemplo usa um valor `LOCATION` de logs para uma determinada conta, mas você pode usar o grau de especificidade adequado ao aplicativo.

Por exemplo:

- Para analisar dados de várias contas, você pode reverter o especificador `LOCATION` para indicar todos os `AWSLogs` usando `LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/'`.

- Para analisar dados de uma data, conta e região específica, use `LOCATION 's3://DOC-EXAMPLE-BUCKET/123456789012/CloudTrail/us-east-1/2016/03/14/'`.

O uso do nível mais alto na hierarquia de objetos proporciona maior flexibilidade ao consultar usando o Athena.

Usar o console do CloudTrail para criar uma tabela do Athena com logs do CloudTrail

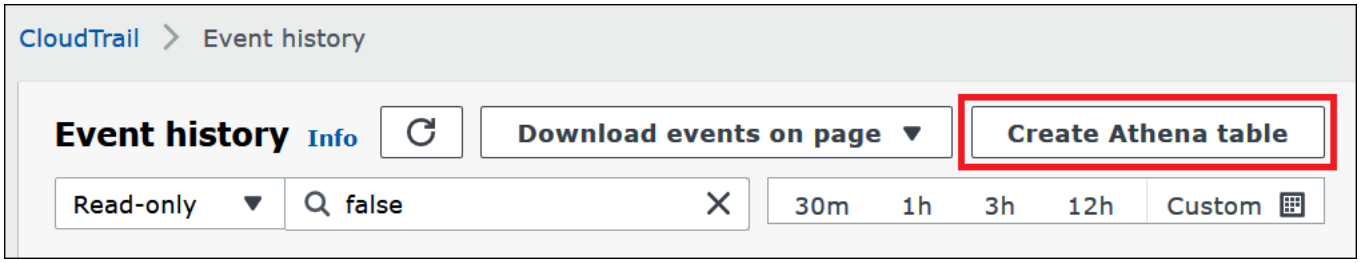
Você pode criar uma tabela do Athena não particionada para consultar logs do CloudTrail diretamente no console do CloudTrail. A criação de uma tabela do Athena pelo console do CloudTrail requer que você esteja conectado com um perfil que tenha permissões suficientes para criar tabelas no Athena.

> **Note**
>
> Não é possível usar o console do CloudTrail para criar uma tabela do Athena para logs de trilha da organização. Em vez disso, crie a tabela manualmente usando o console do Athena para que você possa especificar o local correto de armazenamento. Para obter informações sobre trilhas de organização, consulte Criar uma trilha para uma organização no Manual do usuário do AWS CloudTrail.

- Para obter informações sobre como configurar permissões para o Athena, consulte Configuração.
- Para obter informações sobre como criar uma tabela com partições, consulte Criar uma tabela de logs do CloudTrail no Athena usando particionamento manual.

Para criar uma tabela do Athena para uma trilha do CloudTrail no console do CloudTrail

1. Abra o console do CloudTrail em https://console.aws.amazon.com/cloudfront/.
2. No painel de navegação, selecione Event history (Histórico de eventos).
3. Escolha Create Athena table (Criar tabela do Athena).



4. Em Storage location (Local de armazenamento), use a seta para baixo para selecionar o bucket do Amazon S3 onde os arquivos de log estão armazenados para a trilha que será consultada.

> **Note**
>
> Para localizar o nome do bucket associado a uma trilha, escolha Trails (Trilhas) no painel de navegação do CloudTrail e visualize a coluna S3 bucket (Bucket do S3) da trilha. Para ver o local do bucket do Amazon S3, escolha o link do bucket na coluna S3 bucket (Bucket do S3). O console do Amazon S3 é aberto no local do bucket do CloudTrail.

5. Escolha Create table. A tabela é criada com um nome padrão que inclui o nome do bucket do Amazon S3.

Criar uma tabela de logs do CloudTrail no Athena usando particionamento manual

É possível criar manualmente tabelas de arquivos de log do CloudTrail no console do Athena e executar consultas no Athena.

Para criar uma tabela do Athena para uma trilha do CloudTrail no console do Athena

1. Copie e cole a instrução DDL a seguir no editor de consultas do console do 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**
>
> Sugerimos usar o `org.apache.hive.hcatalog.data.JsonSerDe` exibido no exemplo. Embora exista um `com.amazon.emr.hive.serde.CloudTrailSerde`, no momento ele não processa alguns dos campos mais recentes do CloudTrail.

2. (Opcional) Remova quaisquer campos não obrigatórios para a tabela. Se for necessário ler somente um determinado conjunto de colunas, sua definição de tabela poderá excluir as outras colunas.
3. Modifique `s3://DOC-EXAMPLE-BUCKET/AWSLogs/Account_ID/CloudTrail/` para apontar para o bucket do Amazon S3 que contém os dados de log.
4. Verifique se os campos estão listados corretamente. Para obter mais informações sobre a lista completa de campos em um registro do CloudTrail, consulte Conteúdo do registro do CloudTrail.

   O exemplo a seguir usa a Hive JSON SerDe. Neste exemplo, os campos `requestparameters`, `responseelements` e `additionaleventdata` são listados como tipo `STRING` na consulta, mas são tipos de dados `STRUCT` usados em JSON. Portanto, para obter dados desses campos, use funções `JSON_EXTRACT`. Para ter mais informações, consulte the section called “Extrair dados do JSON”. Para melhorias de performance, o exemplo particiona os dados por Região da AWS, ano, mês e dia.
5. Execute a instrução `CREATE TABLE` no console do Athena.
6. Use o comando ALTER TABLE ADD PARTITION a fim de carregar as partições para que você consiga consultá-las, conforme o exemplo a seguir.

```
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/'
```

Criar uma tabela para uma trilha de toda a organização usando particionamento manual

Para criar uma tabela para arquivos de log do CloudTrail em toda a organização no Athena, siga as etapas listadas em Criar uma tabela de logs do CloudTrail no Athena usando particionamento manual, mas faça as modificações observadas no procedimento a seguir.

Criar uma tabela do Athena para logs do CloudTrail em toda a organização

1. Na instrução `CREATE TABLE`, modifique a cláusula `LOCATION` para incluir o ID da organização, como no exemplo a seguir:

```
LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/organization_id/Account_ID/CloudTrail/'
```

2. Na cláusula `PARTITIONED BY`, adicione uma entrada para o ID da conta como cadeia de caracteres, como no exemplo a seguir:

```
PARTITIONED BY (account string, region string, year string, month string, day
 string)
```

   O exemplo a seguir mostra o resultado combinado:

```
...

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. Na instrução `ALTER TABLE`, na cláusula `ADD PARTITION`, inclua o ID da organização, como no exemplo a seguir:

```
ALTER TABLE table_name ADD
PARTITION (account='111122223333',
region='us-east-1',
year='2022',
month='08',
day='08')
```

4. Na instrução `ALTER TABLE`, na cláusula `LOCATION`, inclua o ID da organização, o ID da conta e a partição que você deseja adicionar, como no exemplo a seguir:

```
LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/organization_id/Account_ID/CloudTrail/us-
east-1/2022/08/08/'
```

No exemplo a seguir, a instrução `ALTER TABLE` mostra o resultado combinado:

```
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/'
```

Criar a tabela de logs do CloudTrail no Athena usando a projeção de partições

Como os logs do CloudTrail têm uma estrutura conhecida com um esquema de partição que você pode especificar antecipadamente, é possível reduzir o tempo de execução das consultas e automatizar o gerenciamento de partições usando o recurso de projeção de partições do Athena. A projeção de partições adiciona automaticamente novas partições à medida que os dados são adicionados. Isso elimina a necessidade de adicionar manualmente as partições usando `ALTER TABLE ADD PARTITION`.

A instrução de exemplo `CREATE TABLE` a seguir usa automaticamente a projeção de partições com base nos logs do CloudTrail a partir de uma data especificada até o dia de hoje para uma única Região da AWS. Nas cláusulas `LOCATION` e `storage.location.template`, substitua os espaços reservados *`bucket`*, *`account-id`* e *`aws-region`* pelos valores idênticos correspondentes. Em `projection.timestamp.range`, substitua *`2020`*/*`01`*/*`01`* pela data de início desejada. Depois que você executar a consulta com êxito, poderá consultar a tabela. Você não precisa executar `ALTER TABLE ADD PARTITION` para carregar as partições.

```
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}')
```

Para obter mais informações sobre projeção de partições, consulte Projeção de partições com o Amazon Athena.

Consultar campos aninhados

Como os campos `userIdentity` e `resources` são tipos de dados aninhados, é necessário um tratamento especial para consultá-los.

O objeto `userIdentity` consiste em tipos STRUCT aninhados. É possível consultá-los usando um ponto para separar os campos, como no seguinte exemplo:

```
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
```

O campo `resources` é um array de objetos `STRUCT`. Para esses arrays, use `CROSS JOIN UNNEST` para remover o aninhamento do array de modo que você possa consultar seus objetos.

O exemplo a seguir retorna todas as linhas em que o ARN do recurso termina com `example/datafile.txt`. Para legibilidade, a função replace remove a substring inicial `arn:aws:s3:::` do 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
```

Veja a seguir exemplos de consultas de eventos `DeleteBucket`. A consulta extrai do objeto `resources` o nome do bucket e o ID da conta à qual o bucket pertence.

```
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
```

Para obter mais informações sobre como remover o aninhamento, consulte Filtrar matrizes.

Consulta de exemplo

O exemplo a seguir mostra a parte de uma consulta que retorna todas as solicitações anônimas (não assinadas) da tabela criada para os logs de eventos do CloudTrail. Essa consulta seleciona

as solicitações em que `useridentity.accountid` são anônimos e `useridentity.arn` não é especificado:

```
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 ]%';
```

Para obter mais informações, consulte a publicação no blog Big Data da AWS, Análise de segurança, conformidade e atividades operacionais usando o AWS CloudTrail e o Amazon Athena.

Dicas para consultar logs do CloudTrail

Para explorar os dados dos logs do CloudTrail, siga estas dicas:

- Antes de consultar os logs, verifique se a tabela de logs tem a mesma aparência da tabela em the section called “Criar uma tabela de logs do CloudTrail no Athena usando particionamento manual”. Se não for a primeira tabela, exclua a tabela existente usando o seguinte comando: `DROP TABLE cloudtrail_logs`.
- Depois de ignorar a tabela existente, recrie-a. Para ter mais informações, consulte Criar uma tabela de logs do CloudTrail no Athena usando particionamento manual.

  Verifique se os campos na consulta do Athena estão listados corretamente. Para obter informações sobre a lista completa de campos em um registro do CloudTrail, consulte Conteúdo do registro do CloudTrail.

  Se a consulta incluir campos em formatos JSON, como `STRUCT`, extraia dados de JSON. Para ter mais informações, consulte Extrair dados do JSON.

  Algumas sugestões para realizar consultas em sua tabela do CloudTrail:
- Comece observando quais usuários do chamaram quais operações da API e de quais endereços IP de origem.
- Use a consulta SQL básica a seguir como o modelo. Cole a consulta no console do Athena e execute-a.

```
SELECT
```

```
 useridentity.arn,
 eventname,
 sourceipaddress,
 eventtime
FROM cloudtrail_logs
LIMIT 100;
```

- Modifique a consulta para explorar ainda mais seus dados.
- Para melhorar a performance, inclua a cláusula `LIMIT` para retornar um subconjunto especificado de linhas.

Consultar os logs do Amazon EMR

As aplicações do Amazon EMR e de big data executadas no Amazon EMR geram arquivos de log. Os arquivos de log são gravados no nó principal, e você também pode configurar o Amazon EMR para armazená-los automaticamente no Amazon S3. Você pode usar o Amazon Athena para consultar esses logs e identificar eventos e tendências de aplicações e clusters. Para obter mais informações sobre os tipos de arquivos de log no Amazon EMR e como salvá-los no Amazon S3, consulte View log files (Exibir arquivos de log) no Guia de gerenciamento do Amazon EMR.

Criar e consultar uma tabela básica estabelecida nos arquivos de log do Amazon EMR

O exemplo a seguir cria uma tabela básica, `myemrlogs`, com base em arquivos de log salvos em `s3://aws-logs-`*`123456789012`*`-`*`us-west-2`*`/elasticmapreduce/`*`j-2ABCDE34F5GH6`*`/elasticmapreduce/`. O local do Amazon S3 usado nos exemplos abaixo reflete o padrão do local predefinido dos logs de um cluster do EMR criado pela conta da Amazon Web Services *`123456789012`* na região *`us-west-2`*. Se você usar um local personalizado, o padrão será s3://DOC-EXAMPLE-BUCKET/*`ClusterID`*.

Para obter informações sobre a criação de uma tabela particionada para potencialmente aprimorar a performance da consulta e reduzir a transferência de dados, consulte Criar e consultar uma tabela particionada baseada nos logs do Amazon EMR.

```
CREATE EXTERNAL TABLE `myemrlogs`(
  `data` string COMMENT 'from deserializer')
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
LINES TERMINATED BY '\n'
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
```

```
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  's3://aws-logs-123456789012-us-west-2/elasticmapreduce/j-2ABCDE34F5GH6'
```

Os exemplos de consultas a seguir podem ser executados na tabela `myemrlogs` criada pelo exemplo anterior.

Example – Consultar os logs de etapas para ocorrências de ERROR, WARN, INFO, EXCEPTION, FATAL ou DEBUG

```
SELECT data,
        "$PATH"
FROM "default"."myemrlogs"
WHERE regexp_like("$PATH",'s-86URH188Z6B1')
        AND regexp_like(data, 'ERROR|WARN|INFO|EXCEPTION|FATAL|DEBUG') limit 100;
```

Example – Consultar um log de instância específico, i-00b3c0a839ece0a9c, para ERROR, WARN, INFO, EXCEPTION, FATAL ou DEBUG

```
SELECT "data",
        "$PATH" AS filepath
FROM "default"."myemrlogs"
WHERE regexp_like("$PATH",'i-00b3c0a839ece0a9c')
        AND regexp_like("$PATH",'state')
        AND regexp_like(data, 'ERROR|WARN|INFO|EXCEPTION|FATAL|DEBUG') limit 100;
```

Example – Consultar os logs da aplicação Presto para ERROR, WARN, INFO, EXCEPTION, FATAL ou DEBUG

```
SELECT "data",
        "$PATH" AS filepath
FROM "default"."myemrlogs"
WHERE regexp_like("$PATH",'presto')
        AND regexp_like(data, 'ERROR|WARN|INFO|EXCEPTION|FATAL|DEBUG') limit 100;
```

Example – Consultar os logs da aplicação Namenode para ERROR, WARN, INFO, EXCEPTION, FATAL ou DEBUG

```
SELECT "data",
```

```
        "$PATH" AS filepath
FROM "default"."myemrlogs"
WHERE regexp_like("$PATH",'namenode')
        AND regexp_like(data, 'ERROR|WARN|INFO|EXCEPTION|FATAL|DEBUG') limit 100;
```

Example – Consultar todos os logs por data e hora para ERROR, WARN, INFO, EXCEPTION, FATAL ou DEBUG

```
SELECT distinct("$PATH") AS filepath
FROM "default"."myemrlogs"
WHERE regexp_like("$PATH",'2019-07-23-10')
        AND regexp_like(data, 'ERROR|WARN|INFO|EXCEPTION|FATAL|DEBUG') limit 100;
```

Criar e consultar uma tabela particionada baseada nos logs do Amazon EMR

Esses exemplos usam o mesmo local de log para criar uma tabela do Athena, mas a tabela é particionada e uma partição é criada para cada local de log. Para ter mais informações, consulte Particionar dados no Athena.

A consulta a seguir cria a tabela particionada denominada `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'
```

Na sequência, as instruções de consulta a seguir criam as partições de tabela com base nos subdiretórios dos tipos diferentes de log que o Amazon EMR cria no 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/'
```

Depois de criar as partições, você pode executar uma consulta `SHOW PARTITIONS` na tabela para confirmar:

```
SHOW PARTITIONS mypartitionedemrlogs;
```

Os exemplos a seguir demonstram que consultas para entradas de log específicas usam a tabela e as partições criadas pelos exemplos acima.

Example – Consultar os logs da aplicação application_1561661818238_0002 na partição de contêineres para ERROR ou WARN

```
SELECT data,
        "$PATH"
FROM "default"."mypartitionedemrlogs"
WHERE logtype='containers'
        AND regexp_like("$PATH",'application_1561661818238_0002')
        AND regexp_like(data, 'ERROR|WARN') limit 100;
```

Example – Consultar a partição Hadoop-Mapreduce do trabalho job_1561661818238_0004 e as falhas de redução

```
SELECT data,
        "$PATH"
FROM "default"."mypartitionedemrlogs"
WHERE logtype='hadoop-mapreduce'
        AND regexp_like(data,'job_1561661818238_0004|Failed Reduces') limit 100;
```

Example – Consultar os logs do Hive na partição do nó em relação ao ID de consulta 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 – Consultar os logs do Resourcemanager na partição do nó em relação à aplicação 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
```

Consultar logs de fluxo do AWS Global Accelerator

É possível usar o AWS Global Accelerator para criar aceleradoras que direcionam o tráfego de rede para endpoints ideais na rede global da AWS. Para obter mais informações sobre o Global Accelerator, consulte What is AWS Global Accelerator? (O que é o ?).

Os logs de fluxo do Global Accelerator permitem que você capture as informações sobre o tráfego de endereço IP que entra e sai das interfaces de rede em suas aceleradoras. Os dados do log de fluxo são publicados no Amazon S3, de onde é possível recuperá-los e visualizá-los. Para obter mais informações, consulte Flow logs in AWS Global Accelerator (Logs de fluxo do).

É possível usar o Athena para consultar os logs de fluxo do Global Accelerator criando uma tabela que especifica o local deles no Amazon S3.

Para criar a tabela de logs de fluxo do Global Accelerator

1. Copie e cole a instrução DDL a seguir no console do Athena. Esta consulta especifica ROW FORMAT DELIMITED e omite a especificação de um SerDe, o que significa que a consulta usa `LazySimpleSerDe`. Nesta consulta, os campos terminam com um espaço.

```
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. Modifique o valor `LOCATION` para apontar para o bucket do Amazon S3 que contém os dados de log.

```
's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/account_id/globalaccelerator/region_code/'
```

3. Execute a consulta no console do Athena. Depois que a consulta for concluída, o Athena registrará a tabela `aga_flow_logs`, disponibilizando os dados dela para consultas.
4. Crie partições para ler os dados, conforme a consulta de exemplo a seguir. Esta consulta cria uma única partição para uma data especificada. Substitua os espaços reservados por data e local.

```
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';
```

Consultas de exemplo para logs de fluxo do AWS Global Accelerator

Example – Listar as solicitações que passam por um local da borda específico

A consulta de exemplo a seguir lista as solicitações que passaram pelo ponto de presença do LHR. Use o operador `LIMIT` para limitar o número de logs a serem consultados por vez.

```
SELECT
  clientip,
  agaregion,
  protocol,
  action
FROM
  aga_flow_logs
WHERE
  agaregion LIKE 'LHR%'
LIMIT
  100;
```

Example – Listar os endereços IP dos endpoints que recebem a maioria das solicitações HTTPS

Para ver quais endereços IP do endpoint estão recebendo o maior número de solicitações HTTPS, use a seguinte consulta. Esta consulta conta o número de pacotes recebidos na porta HTTPS 443, agrupa-os por endereço IP de destino e retorna os 10 principais endereços IP.

```
SELECT
```

```
  SUM(numpackets) AS packetcount,
  endpointip
FROM
  aga_flow_logs
WHERE
  endpointport = 443
GROUP BY
  endpointip
ORDER BY
  packetcount DESC
LIMIT
  10;
```

Consultar descobertas do Amazon GuardDuty

O Amazon GuardDuty é um serviço de monitoramento de segurança que ajuda a identificar atividades inesperadas e potencialmente mal-intencionadas ou não autorizadas em seu ambiente AWS. Quando detecta atividades inesperadas e potencialmente mal-intencionadas, o GuardDuty gera descobertas de segurança que você pode exportar para o Amazon S3 para armazenamento e análise. Depois de exportar as descobertas para o Amazon S3, você pode usar o Athena para consultá-las. Este artigo mostra como criar uma tabela no Athena para as descobertas do GuardDuty e consultá-las.

Para obter mais informações sobre o Amazon GuardDuty, consulte o Manual do usuário do Amazon GuardDuty.

Pré-requisitos

- Habilite o recurso GuardDuty para exportar as descobertas para o Amazon S3. Para conhecer as etapas, consulte Exporting findings (Exportar descobertas) no Guia do usuário do Amazon GuardDuty.

Criar uma tabela no Athena para as descobertas do GuardDuty

Para consultar as descobertas do GuardDuty no Athena, crie uma tabela para elas.

Para criar uma tabela no Athena para as descobertas do GuardDuty

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.

2. Copie a instrução DDL a seguir no console do Athena. Modifique os valores em `LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/account-id/GuardDuty/'` para apontar para as descobertas do GuardDuty no 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**
>
> O SerDe espera que cada documento JSON esteja em uma única linha de texto, sem caracteres de terminação de linha separando os campos no registro. Se o texto JSON estiver formatado para impressão, você poderá receber uma mensagem de erro como HIVE_CURSOR_ERROR: Row is not a valid JSON Object (HIVE_CURSOR_ERROR: a linha não é um objeto JSON válido) ou HIVE_CURSOR_ERROR: JsonParseException: Unexpected end-of-input: expected close marker for OBJECT (HIVE_CURSOR_ERROR: JSONParseException: Fim de entrada inesperado: marcador de fechamento esperado para OBJECT) quando tentar consultar a tabela após criá-la. Para obter mais informações, consulte JSON Data Files na documentação do OpenX SerDe no GitHub.

3. Execute a consulta no console do Athena para registrar a tabela `gd_logs`. Quando a consulta for concluída, as descobertas estarão prontas para serem consultadas no Athena.

Consultas de exemplo

Os exemplos a seguir mostram como consultar as descobertas do GuardDuty no Athena.

Example – Exfiltração de dados de DNS

A consulta a seguir retorna informações sobre as instâncias do Amazon EC2 que podem estar exfiltrando dados por meio das consultas de 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 – Acesso de usuário do IAM não autorizado

A consulta a seguir retorna todos os tipos de descoberta `UnauthorizedAccess:IAMUser` referentes a um principal do IAM de todas as regiões.

```
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;
```

Dicas para consultar descobertas do GuardDuty

Ao criar a consulta, mantenha os seguintes pontos em mente.

- Para extrair dados de campos JSON aninhados, use as funções `json_extract` ou `json_extract_scalar` do Presto. Para ter mais informações, consulte Extrair dados do JSON.
- Certifique-se de que todos os caracteres nos campos JSON estejam em minúsculas.
- Para obter informações sobre como baixar resultados de consulta, verifique Baixar os arquivos de resultados de consultas pelo console do Athena.

Consultar os logs do AWS Network Firewall

AWS Network Firewall é um serviço gerenciado que você pode usar para implantar proteções de rede essenciais em suas instâncias do Amazon Virtual Private Cloud. O AWS Network Firewall funciona em conjunto com o AWS Firewall Manager para que você possa criar políticas com base nas regras do AWS Network Firewall e aplicá-las a suas VPCs e contas de maneira centralizada. Para obter mais informações sobre o AWS Network Firewall, consulte AWS Network Firewall.

É possível configurar os logs do AWS Network Firewall para o tráfego que você encaminha para o mecanismo de regras com estado do firewall. Os logs apresentam informações detalhadas sobre o tráfego de rede, incluindo a hora em que o mecanismo com estado recebeu um pacote, os detalhes do pacote e qualquer ação de regra com estado realizada no pacote. Os logs são publicados no destino de logs que você configurou, de onde é possível recuperá-los e visualizá-los. Para obter mais informações, consulte Registrar o tráfego de rede do AWS Network Firewall no Guia do desenvolvedor do AWS Network Firewall (em inglês).

Criar uma tabela para logs de alertas

1. Modifique o exemplo de instrução DDL a seguir para se adequar à estrutura do seu log de alertas. Pode ser necessário atualizar a instrução para incluir as colunas da versão mais recente dos logs. Para obter mais informações, consulte Conteúdo de um log de firewall no Guia do desenvolvedor do AWS Network Firewall.

```
CREATE EXTERNAL TABLE network_firewall_alert_logs (
  firewall_name string,
  availability_zone string,
  event_timestamp string,
  event struct<
    timestamp:string,
    flow_id:bigint,
    event_type:string,
    src_ip:string,
    src_port:int,
    dest_ip:string,
    dest_port:int,
    proto:string,
    app_proto:string,
    tls_inspected:boolean,
    alert:struct<
      alert_id:string,
      alert_type:string,
      action:string,
      signature_id:int,
      rev:int,
      signature:string,
      category:string,
      severity:int,
      rule_name:string,
      alert_name:string,
      alert_severity:string,
      alert_description:string,
      file_name:string,
      file_hash:string,
      packet_capture:string,
      reference_links:array<string>
    >,
    src_country:string,
    dest_country:string,
    src_hostname:string,
    dest_hostname:string,
    user_agent:string,
    url:string
   >
)
 ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
```

```
LOCATION 's3://DOC-EXAMPLE-BUCKET/path_to_alert_logs_folder/';
```

2. Modifique a cláusula `LOCATION` para especificar a pasta para seus logs no Amazon S3.
3. Execute sua consulta `CREATE TABLE` no editor de consultas do Athena. Depois que a consulta for concluída, o Athena registrará a tabela `network_firewall_alert_logs`, preparando os dados para os quais ela direciona para as consultas.

Exemplo de consulta do log de alertas

O exemplo de consulta de log de alertas nesta seção filtra os eventos nos quais a inspeção TLS foi realizada e que têm alertas com um nível de severidade de 2 ou superior.

A consulta usa aliases para criar cabeçalhos de coluna de saída que mostram o `struct` ao qual a coluna pertence. Por exemplo, o título da coluna do campo `event.alert.category` é `event_alert_category` em vez de apenas`category`. Para personalizar ainda mais os nomes das colunas, você pode modificar os aliases de acordo com suas preferências. Por exemplo, você pode usar sublinhados ou outros separadores para delimitar os nomes de `struct` e os nomes dos campos.

Lembre-se de modificar os nomes de colunas e as referências de `struct` com base na definição da tabela e nos campos que você deseja no resultado da consulta.

```
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;
```

Criar uma tabela para logs de fluxo de rede

1. Modifique o exemplo de instrução DDL a seguir para se adequar à estrutura do seu log de fluxo de redes. Pode ser necessário atualizar a instrução para incluir as colunas da versão mais recente dos logs. Para obter mais informações, consulte Conteúdo de um log de firewall no Guia do desenvolvedor do AWS Network Firewall.

   ```
   CREATE EXTERNAL TABLE network_firewall_netflow_logs (
     firewall_name string,
     availability_zone string,
     event_timestamp string,
     event struct<
       timestamp:string,
       flow_id:bigint,
       event_type:string,
       src_ip:string,
       src_port:int,
       dest_ip:string,
       dest_port:int,
   ```

```
    proto:string,
    app_proto:string,
    netflow:struct<
      pkts:int,
      bytes:bigint,
      start:string,
      `end`:string,
      age:int,
      min_ttl:int,
      max_ttl:int,
      tcp_flags:struct<
        syn:boolean,
        fin:boolean,
        rst:boolean,
        psh:boolean,
        ack:boolean,
        urg:boolean
      >,
      tls_inspected:boolean
    >
  >
)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 's3://DOC-EXAMPLE-BUCKET/path_to_netflow_logs_folder/';
```

2. Modifique a cláusula `LOCATION` para especificar a pasta para seus logs no Amazon S3.
3. Execute a consulta `CREATE TABLE` no editor de consultas do Athena. Depois que a consulta for concluída, o Athena registrará a tabela `network_firewall_netflow_logs`, preparando os dados para os quais ela direciona para as consultas.

Exemplo de consulta de log do Netflow

O exemplo de consulta de log do Netflow nesta seção filtra os eventos nos quais a inspeção TLS foi realizada.

A consulta usa aliases para criar cabeçalhos de coluna de saída que mostram o `struct` ao qual a coluna pertence. Por exemplo, o título da coluna do campo `event.netflow.bytes` é `event_netflow_bytes` em vez de apenas`bytes`. Para personalizar ainda mais os nomes das colunas, você pode modificar os aliases de acordo com suas preferências. Por exemplo, você pode usar sublinhados ou outros separadores para delimitar os nomes de `struct` e os nomes dos campos.

Lembre-se de modificar os nomes de colunas e as referências de `struct` com base na definição da tabela e nos campos que você deseja no resultado da consulta.

```
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
```

Consultar os logs do Network Load Balancer

Use o Athena para analisar e processar os logs do Network Load Balancer. Esses logs recebem informações detalhadas sobre as solicitações Transport Layer Security (TLS) enviadas ao Network Load Balancer. Você pode usar esses logs de acesso para analisar padrões de tráfego e solucionar problemas.

Antes de analisar os logs de acesso do Network Load Balancer, habilite-os e configure-os para serem salvos no bucket de destino do Amazon S3. Para obter mais informações, além de informações sobre cada entrada de log de acesso do Network Load Balancer, consulte Access logs for your Network Load Balancer.

- Criar a tabela de logs do Network Load Balancer
- Exemplo de consultas do Network Load Balancer

Para criar a tabela de logs do Network Load Balancer

1. Copie e cole a instrução DDL a seguir no console do Athena. Verifique a sintaxe dos registros de log do Network Load Balancer. Pode ser necessário atualizar a consulta a seguir para incluir as colunas e a sintaxe Regex para a versão mais recente do registro.

   ```
   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. Modifique o bucket `LOCATION` do Amazon S3 para especificar o destino dos logs do Network Load Balancer.
3. Execute a consulta no console do Athena. Depois que a consulta for concluída, o Athena registrará a tabela `nlb_tls_logs`, preparando os dados dela para as consultas.

Exemplo de consultas do Network Load Balancer

Para ver quantas vezes um certificado é usado, use uma consulta semelhante a este exemplo:

```
SELECT count(*) AS
```

```
         ct,
         cert_arn
FROM "nlb_tls_logs"
GROUP BY  cert_arn;
```

A consulta a seguir mostra quantos usuários estão usando uma versão do TLS anterior à versão 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;
```

Use a consulta a seguir para identificar conexões que levam muito tempo de handshake do TLS.

```
SELECT *
FROM "nlb_tls_logs"
ORDER BY  tls_handshake_time_ms DESC
LIMIT 10;
```

Use a consulta a seguir para identificar e contar as versões de protocolo TLS e os conjuntos de cifras que foram negociados nos últimos 30 dias.

```
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;
```

Consultar os logs do Amazon Route 53 Resolver

Você pode criar tabelas do Athena para seus logs de consulta do Amazon Route 53 Resolver e consultá-las no Athena.

O log de consulta do Route 53 Resolver é destinado ao registro de consultas de DNS feitas por recursos em uma VPC, recursos on-premises que usam endpoints do Resolver de entrada, consultas

que usam um endpoint do Resolver de saída para resolução de DNS recursiva e consultas que usam regras de firewall de DNS do Route 53 Resolver para bloquear, permitir ou monitorar uma lista de domínios. Para obter mais informações sobre o log de consulta do Resolver, veja Log de consulta do Resolver no Guia do desenvolvedor do Amazon Route 53. Para obter informações sobre cada um dos campos nos logs, consulte Valores que aparecem em logs de consultas do Resolver no Guia do desenvolvedor do Amazon Route 53.

Criar a tabela de logs de consulta do Resolver

Você pode usar o editor de consultas no console do Athena para criar e consultar uma tabela com os logs de consulta do Route 53 Resolver.

Para criar e consultar uma tabela do Athena com os logs de consulta do Route 53 Resolver

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. No editor de consultas do Athena, insira a instrução `CREATE TABLE` a seguir. Substitua os valores da cláusula `LOCATION` pelos correspondentes ao local dos logs do Resolver no 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}/'
```

Como os dados de log de consulta do Resolver estão no formato JSON, a instrução CREATE TABLE usa a Biblioteca SerDe do JSON para analisar os dados.

> **Note**
>
> O SerDe espera que cada documento JSON esteja em uma única linha de texto, sem caracteres de terminação de linha separando os campos no registro. Se o texto JSON estiver formatado para impressão, você poderá receber uma mensagem de erro como HIVE_CURSOR_ERROR: Row is not a valid JSON Object (HIVE_CURSOR_ERROR: a linha não é um objeto JSON válido) ou HIVE_CURSOR_ERROR: JsonParseException: Unexpected end-of-input: expected close marker for OBJECT (HIVE_CURSOR_ERROR: JSONParseException: Fim de entrada inesperado: marcador de fechamento esperado para OBJECT) quando tentar consultar a tabela após criá-la. Para obter mais informações, consulte JSON Data Files na documentação do OpenX SerDe no GitHub.

3. Selecione Executar consulta. A instrução cria uma tabela do Athena chamada `r53_rlogs` com colunas que representam cada um dos campos em seus dados de log do Resolver.
4. No editor de consultas do console do Athena, execute a consulta a seguir para verificar se a tabela foi criada.

```
SELECT * FROM "r53_rlogs" LIMIT 10
```

Exemplo de particionamento

O exemplo a seguir mostra uma declaração `CREATE TABLE` para logs de consulta do Resolver que usa projeção de partição e é particionada por VPC e por data. Para obter mais informações sobre projeção de partições, consulte Projeção de partições com o 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}/'
)
```

Consultas de exemplo

Os exemplos a seguir mostram algumas consultas do Athena que você pode executar em seus logs de consulta do Resolver.

Exemplo 1: logs de consulta em ordem decrescente de query_timestamp

A consulta a seguir exibe os resultados do log em ordem decrescente de `query_timestamp`.

```
SELECT * FROM "r53_rlogs"
ORDER BY query_timestamp DESC
```

Exemplo 2: logs de consulta com horários de início e de término específicos

As consultas a seguir analisam os logs entre meia-noite e 8h no dia 24 de setembro de 2020. Substitua os horários de início e de término de acordo com os seus requisitos.

```
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
```

Exemplo 3: logs de consulta baseados em um padrão de nome especificado de consulta de DNS

A consulta a seguir seleciona os registros com nome de consulta que inclui a 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
```

Exemplo 4: solicitações de log de consulta sem resposta

A consulta a seguir seleciona as entradas de log em que a solicitação não recebeu resposta.

```
SELECT query_timestamp, srcids.instance, srcaddr, srcport, query_name, rcode, answers
FROM "r53_rlogs"
WHERE cardinality(answers) = 0
```

Exemplo 5: logs de consulta com uma resposta específica

A consulta a seguir mostra os logs nos quais o valor `answer.Rdata` tem o endereço IP especificado.

```
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';
```

Consultar logs de eventos do Amazon SES

É possível utilizar o Amazon Athena para consultar logs de eventos do Amazon Simple Email Service (Amazon SES).

O Amazon SES é uma plataforma de e-mail que oferece uma forma conveniente e com bom custo-benefício de enviar e receber e-mails usando seus próprios endereços de e-mail e domínios. Você pode monitorar suas atividades de envio do Amazon SES em nível granular usando eventos, métricas e estatísticas.

Com base nas características que você define, é possível publicar eventos do Amazon SES no Amazon CloudWatch, no Amazon Data Firehose ou no Amazon Simple Notification Service. Depois que as informações forem armazenadas no Amazon S3, você poderá consultá-las no Amazon Athena.

Para obter mais informações sobre como analisar eventos de e-mail do Amazon SES com o Firehose, o Amazon Athena e o Amazon QuickSight, consulte o artigo Analisar dados de eventos do Amazon SES com serviços de análise da AWS no blog de mensagens e segmentação da AWS.

Consultar os logs de fluxo do Amazon VPC

Os logs de fluxo do Amazon Virtual Private Cloud capturam informações do tráfego de IP de entrada e saída das interfaces de rede em uma VPC. Use os logs para investigar padrões de tráfego de rede e identificar ameaças e riscos em toda a rede da VPC.

Para consultar seus logs de fluxo da Amazon VPC, você tem duas opções:

- Console do Amazon VPC: use o recurso de integração do Athena no console do Amazon VPC para gerar um modelo do AWS CloudFormation que cria um banco de dados, um grupo de trabalho e uma tabela de logs de fluxo do Athena com particionamento para você. O modelo também cria um conjunto de consultas de log de fluxo predefinidas que você pode usar para obter insights sobre o tráfego que passa pela VPC.

  Para obter mais informações sobre essa abordagem, consulte Consultar logs de fluxo usando o Amazon Athena no Guia do usuário da Amazon VPC.
- Console do Amazon Athena: crie suas tabelas e consultas diretamente no console do Athena. Para obter mais informações, continue lendo esta página.

Criar e consultar tabelas para logs de fluxo da VPC personalizados

Antes de começar a consultar os logs no Athena, habilite os logs de fluxo da VPC e configure-os para serem salvos em seu bucket do Amazon S3. Depois de criar os logs, deixe-os em execução por alguns minutos para coletar alguns dados. Os logs são criados no formato de compactação GZIP que o Athena permite que você consulte diretamente.

Ao criar um log de fluxo da VPC, você pode usar um formato personalizado quando desejar especificar os campos que devem ser retornados no log de fluxo e a ordem em que devem aparecer. Para obter mais informações sobre registros de logs de fluxo, consulte Registros de log de fluxo no Guia do usuário da Amazon VPC.

Considerações comuns

Ao criar tabelas no Athena para logs de fluxo da Amazon VPC, lembre-se dos seguintes pontos:

- Por padrão, no Athena, o Parquet acessará as colunas por nome. Para ter mais informações, consulte Tratamento de atualizações do esquema.
- Use os nomes nos registros de log de fluxo para os nomes das colunas no Athena. Os nomes das colunas no esquema do Athena devem corresponder exatamente aos nomes dos campos nos logs de fluxo da Amazon VPC, com as seguintes diferenças:
  - Substitua os hifens nos nomes dos campos do log da Amazon VPC por sublinhados nos nomes das colunas do Athena. No Athena, os únicos caracteres aceitáveis em nomes de bancos de dados, nomes de tabelas e nomes de colunas são letras minúsculas, números e o caractere de sublinhado. Para ter mais informações, consulte Nomes de bancos de dados, tabelas e colunas.

- Escape os nomes de registro do log de fluxo que são palavras-chave reservadas no Athena, colocando-os entre caracteres de crase.
- Os logs de fluxos da VPC são específicos da Conta da AWS. Quando você publica seus arquivos de log no Amazon S3, o caminho criado pela Amazon VPC no Amazon S3 inclui o ID daConta da AWS que foi usada para criar o log de fluxo. Para obter mais informações, consulte Publicar logs de fluxo no Amazon S3, no Guia do usuário da Amazon VPC.

Instrução CREATE TABLE para logs de fluxo da Amazon VPC

O procedimento a seguir cria uma tabela da Amazon VPC para os logs de fluxo da Amazon VPC. Ao criar um log de fluxo em um formato personalizado, você cria uma tabela com campos que correspondem aos campos especificados quando criou o log de fluxo, na mesma ordem em que os especificou.

Para criar uma tabela do Athena para logs de fluxo da Amazon VPC

1. Insira uma instrução DDL como a que se segue no editor de consultas do console do Athena, seguindo as diretrizes na seção Considerações comuns. A instrução do exemplo cria uma tabela com as colunas de logs de fluxo da Amazon VPC nas versões 2 a 5, conforme documentado em Registros de log de fluxo. Se você usar outro conjunto ou outra ordem de colunas, modifique a instrução conforme apropriado.

```
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");
```

Observe os seguintes pontos:

- A consulta especifica `ROW FORMAT DELIMITED` e omite a especificação de um SerDe. Isso significa que a consulta usa LazySimpleSerDe para arquivos CSV, TSV e com delimitação personalizada. Nesta consulta, os campos terminam com um espaço.
- A cláusula `PARTITIONED BY` usa o tipo `date`. Isso torna possível usar operadores matemáticos em consultas para selecionar qual é mais antiga ou mais recente em relação a uma determinada data.

> **Note**
>
> Como `date` é uma palavra-chave reservada em instruções DDL, é necessário fazer o escape dela com acentos graves. Para ter mais informações, consulte Palavras-chave reservadas.

- Para um log de fluxo da VPC com um formato personalizado diferente, modifique os campos para que correspondam aos campos especificados ao criar o log de fluxo.

2. Modifique `LOCATION 's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/{account_id}/vpcflowlogs/{region_code}/'` para apontar para o bucket do Amazon S3 que contém os dados de log.

3. Execute a consulta no console do Athena. Depois que a consulta for concluída, o Athena registrará a tabela `vpc_flow_logs`, preparando os dados dela para você fazer as consultas.
4. Crie partições para poder ler os dados, conforme o exemplo de consulta a seguir. Esta consulta cria uma única partição para uma data especificada. Substitua os espaços reservados por data e local, conforme necessário.

> **Note**
>
> Esta consulta cria apenas uma única partição, para uma data especificada por você. Para automatizar o processo, use um script que execute essa consulta e crie partições dessa maneira para `year/month/day` ou use uma instrução `CREATE TABLE` que especifique a projeção das partições.

```
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';
```

Exemplos de consulta para a tabela vpc_flow_logs

Use o editor de consultas no console do Athena para executar instruções SQL na tabela criada. É possível salvar as consultas, visualizar consultas anteriores ou baixar os resultados da consulta no formato CSV. Nos exemplos a seguir, substitua `vpc_flow_logs` pelo nome da tabela. Modifique os valores das colunas e outras variáveis de acordo com os seus requisitos.

A consulta de exemplo a seguir lista um máximo de 100 logs de fluxo para a data especificada.

```
SELECT *
FROM vpc_flow_logs
WHERE date = DATE('2020-05-04')
LIMIT 100;
```

A consulta a seguir lista todas as conexões TCP rejeitadas e usa a coluna de partição de data recém-criada, `date`, para extrair dela o dia da semana em que ocorreram esses eventos.

```
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;
```

Para ver qual dos servidores está recebendo o maior número de solicitações HTTPS, use a consulta a seguir. Ela conta o número de pacotes recebidos na porta HTTPS 443, agrupa-os por endereço IP de destino e retorna os 10 principais da última semana.

```
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;
```

Criar tabelas para logs de fluxo no formato Apache Parquet

O procedimento a seguir cria uma tabela da Amazon VPC para os logs de fluxo da Amazon VPC no formato Apache Parquet.

Para criar uma tabela do Athena para logs de fluxo da Amazon VPC no formato Parquet

1. Insira uma instrução DDL como a que se segue no editor de consultas do console do Athena, seguindo as diretrizes na seção Considerações comuns. A instrução do exemplo cria uma tabela com as colunas de logs de fluxo da Amazon VPC nas versões 2 a 5, conforme documentado em Registros de log de fluxo no formato Parquet, com partição do Hive por hora. Se você não tiver partições por hora, remova `hour` da cláusula `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. Modifique a amostra `LOCATION 's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/'` para apontar para o caminho do Amazon S3 que contém os dados de log.
3. Execute a consulta no console do Athena.
4. Se os dados estiverem em um formato compatível com o Hive, execute o comando a seguir no console do Athena para atualizar e carregar as partições do Hive no metastore. Após a conclusão da consulta, você pode consultar os dados na tabela `vpc_flow_logs_parquet`.

   ```
   MSCK REPAIR TABLE vpc_flow_logs_parquet
   ```

   Se não estiver usando dados compatíveis com o Hive, execute ALTER TABLE ADD PARTITION para carregar as partições.

Para obter mais informações sobre como usar o Athena para consultar logs de fluxo da Amazon VPC no formato Parquet, consulte a publicação Optimize performance and reduce costs for network analytics with VPC Flow Logs in Apache Parquet format (Otimize a performance e reduza os custos de análise de rede com logs de fluxo da VPC no formato Apache Parquet) no blog sobre big data da AWS.

Criar e consultar uma tabela para logs de fluxo da Amazon VPC usando projeção de partições

Use uma instrução `CREATE TABLE` como a que se segue para criar uma tabela, particionar a tabela e preencher as partições automaticamente usando projeção de partições. Substitua o nome da tabela `test_table_vpclogs` no exemplo pelo nome da tabela. Edite a cláusula `LOCATION` para especificar o bucket do Amazon S3 que contém os dados de log da Amazon VPC.

A instrução `CREATE TABLE` a seguir é para logs de fluxo da VPC fornecidos em um formato de particionamento em um estilo diferente do Hive. O exemplo permite a agregação de várias contas. Se você estiver centralizando logs de fluxo de VPC de diversas contas em um bucket do Amazon S3, o ID da conta deverá ser inserido no caminho do 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}"
)
```

Exemplos de consulta para test_table_vpclogs

Os exemplos de consulta a seguir consultam a `test_table_vpclogs` criada pela instrução `CREATE TABLE` anterior. Substitua `test_table_vpclogs` nas consultas pelo nome de sua própria tabela. Modifique os valores das colunas e outras variáveis de acordo com os seus requisitos.

Para retornar as primeiras 100 entradas de log de acesso em ordem cronológica para o período de tempo especificado, execute uma consulta como a que se segue.

```
SELECT *
FROM test_table_vpclogs
WHERE day >= '2021/02/01' AND day < '2021/02/28'
ORDER BY day ASC
LIMIT 100
```

Para ver qual servidor recebe os dez principais pacotes HTTP para um período de tempo especificado, execute uma consulta como a que se segue. A consulta conta o número de pacotes recebidos na porta HTTPS 443, agrupa-os por endereço IP de destino e retorna as 10 principais entradas da semana anterior.

```
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
```

Para retornar os logs que foram criados durante um período de tempo especificado, execute uma consulta como a que se segue.

```
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'
```

Para retornar os logs de acesso de um endereço IP de origem em um determinado intervalo de tempo, execute uma consulta como a que se segue.

```
SELECT *
FROM test_table_vpclogs
WHERE srcaddr = '10.117.1.22'
  AND day >= '2021/02/01'
  AND day < '2021/02/28'
```

Para listas as conexões TCP rejeitadas, execute uma consulta como a que se segue.

```
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
```

Para retornar os logs de acesso para o intervalo de endereços IP que começa com `10.117`, execute uma consulta como a que se segue.

```
SELECT *
FROM test_table_vpclogs
WHERE split_part(srcaddr,'.', 1)='10'
  AND split_part(srcaddr,'.', 2) ='117'
```

Para retornar os logs de acesso para um endereço IP de destino em um determinado intervalo de tempo, execute uma consulta como a que segue.

```
SELECT *
FROM test_table_vpclogs
WHERE dstaddr = '10.0.1.14'
  AND day >= '2021/01/01'
  AND day < '2021/01/31'
```

Criar tabelas para logs de fluxo no formato do Apache Parquet usando projeção de partições

A instrução CREATE TABLE da projeção de partições para logs de fluxo da VPC a seguir está no formato do Apache Parquet, não é compatível com o Hive e é particionada por hora e por data, em vez de por dia. Substitua o nome da tabela `test_table_vpclogs_parquet` no exemplo pelo nome da tabela. Edite a cláusula `LOCATION` para especificar o bucket do Amazon S3 que contém os dados de log da Amazon VPC.

```
CREATE EXTERNAL TABLE IF NOT EXISTS test_table_vpclogs_parquet (
  version int,
  account_id string,
  interface_id string,
  srcaddr string,
  dstaddr string,
  srcport int,
  dstport int,
  protocol bigint,
  packets bigint,
  bytes bigint,
  start bigint,
  `end` bigint,
  action string,
  log_status string,
  vpc_id string,
  subnet_id string,
  instance_id string,
  tcp_flags int,
  type string,
  pkt_srcaddr string,
  pkt_dstaddr string,
  az_id string,
  sublocation_type string,
  sublocation_id string,
  pkt_src_aws_service string,
  pkt_dst_aws_service string,
  flow_direction string,
  traffic_path int
)
PARTITIONED BY (region string, date string, hour string)
ROW FORMAT SERDE
'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'
```

```
OUTPUTFORMAT
'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION 's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/{account_id}/vpcflowlogs/'
TBLPROPERTIES (
"EXTERNAL"="true",
"skip.header.line.count" = "1",
"projection.enabled" = "true",
"projection.region.type" = "enum",
"projection.region.values" = "us-east-1,us-west-2,ap-south-1,eu-west-1",
"projection.date.type" = "date",
"projection.date.range" = "2021/01/01,NOW",
"projection.date.format" = "yyyy/MM/dd",
"projection.hour.type" = "integer",
"projection.hour.range" = "00,23",
"projection.hour.digits" = "2",
"storage.location.template" = "s3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/${account_id}/
vpcflowlogs/${region}/${date}/${hour}"
)
```

Recursos adicionais do

Para obter mais informações sobre como usar o Athena para analisar logs de fluxo de VPC, consulte as postagens do blog de AWS big data a seguir:

- Analyze VPC Flow Logs with point-and-click Amazon Athena integration (Análise de logs de fluxo de VPC com a integração de apontar e clicar do Amazon Athena)
- Analyzing VPC flow logs using Amazon Athena and Amazon QuickSight (Análise de logs de fluxo de VPC usando o Amazon Athena e o Amazon QuickSight)
- Optimize performance and reduce costs for network analytics with VPC Flow Logs in Apache Parquet format (Otimização da performance e redução dos custos de análise de rede com os logs de fluxo de VPC no formato Apache Parquet)

Consultar os logs do AWS WAF

O AWS WAF é um firewall para aplicações Web que permite monitorar e controlar as solicitações HTTP e HTTPS que as aplicações Web protegidas recebem dos clientes. Você define como lidar com as solicitações da Web configurando regras dentro de uma lista de controle de acesso (ACL) da Web do AWS WAF. Depois, você protege uma aplicação Web associando a ela uma ACL da Web. Exemplos de recursos de aplicações Web que você pode proteger com o AWS WAF incluem distribuições do Amazon CloudFront, APIs REST do Amazon API Gateway e Application Load

Balancers. Para obter mais informações sobre o AWS WAF, consulte AWS WAF no AWS WAF Developer Guide.

Os logs do AWS WAF incluem informações sobre o tráfego que é analisado pela sua ACL da Web, como a hora em que o AWS WAF recebeu a solicitação do recurso da AWS, os detalhes da solicitação e a ação para a regra correspondente de cada solicitação.

Você pode configurar uma ACL da Web do AWS WAF para publicar logs em um de vários destinos, onde você pode consultá-los e visualizá-los. Para obter mais informações sobre a configuração de registro em log de ACL da Web e sobre o conteúdo dos logs do AWS WAF, consulte Logging AWS WAF web ACL traffic no AWS WAF developer guide.

Para ver um exemplo de como agregar os logs do AWS WAF em um repositório central de data lake e consultá-los no Athena, leia a publicação no blog de big data da AWS: Analisar os logs do AWS WAF com o OpenSearch Service, Amazon Athena e Amazon QuickSight (em inglês).

Este tópico oferece dois exemplos de instruções `CREATE TABLE`: um que usa particionamento e outro que não usa.

> **Note**
>
> As instruções `CREATE TABLE` neste tópico podem ser usadas para logs do AWS WAF v1 e v2. Na v1, o campo `webaclid` contém um ID. Na v2, o campo `webaclid` contém um ARN completo. As instruções `CREATE TABLE` aqui tratam esse conteúdo de forma agnóstica usando o tipo de dados `string`.

Tópicos

- Criar uma tabela para logs do S3 do AWS WAF no Athena usando projeção de partições
- Criar uma tabela para logs do AWS WAF sem particionamento
- Consultas de exemplo para logs do AWS WAF

Criar uma tabela para logs do S3 do AWS WAF no Athena usando projeção de partições

Como os logs do AWS WAF têm uma estrutura conhecida com um esquema de partição que você pode especificar antecipadamente, é possível reduzir o runtime das consultas e automatizar o gerenciamento de partições usando o recurso de projeção de partições do Athena. A projeção de partições adiciona automaticamente novas partições à medida que os dados são adicionados.

Isso elimina a necessidade de adicionar manualmente as partições usando `ALTER TABLE ADD PARTITION`.

A instrução de exemplo `CREATE TABLE` a seguir usa automaticamente a projeção de partições com base nos logs do AWS WAF desde uma data especificada até o dia de hoje para diferentes regiões da AWS. A cláusula `PARTITION BY` neste exemplo particiona por região e data, mas você pode modificá-la de acordo com seus requisitos. Modifique os campos conforme necessário para corresponder à saída do log. Nas cláusulas `LOCATION` e `storage.location.template`, substitua os espaços reservados *`bucket`* e *`accountID`* por valores que identifiquem o local do bucket do Amazon S3 dos seus logs do AWS WAF. Em `projection.day.range`, substitua *`2021`*/*`01`*/*`01`* pela data de início desejada. Depois que você executar a consulta com êxito, poderá consultar a tabela. Você não precisa executar `ALTER TABLE ADD PARTITION` para carregar as partições.

```
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

O formato do caminho na cláusula `LOCATION` no exemplo corresponde ao padrão, mas pode variar com base na configuração AWS WAF implementada. Por exemplo, o exemplo de caminho de log do AWS WAF a seguir é para uma distribuição do CloudFront:

```
s3://DOC-EXAMPLE-BUCKET/AWSLogs/12345678910/WAFLogs/cloudfront/
cloudfronyt/2022/08/08/17/55/
```

Se você tiver problemas ao criar ou consultar sua tabela de logs do AWS WAF, confirme a localização de seus dados de log ou entre em contato com o AWS Support.

Para obter mais informações sobre projeção de partições, consulte Projeção de partições com o Amazon Athena.

Criar uma tabela para logs do AWS WAF sem particionamento

Esta seção descreve como criar uma tabela para logs do AWS WAF sem particionamento ou projeção de partição.

Note

Por motivos de desempenho e custo, não recomendamos usar o esquema não particionado para consultas. Para obter mais informações, consulte Top 10 Performance Tuning Tips for Amazon Athena (As 10 melhores dicas para ajustar o desempenho do Amazon Athena) no AWS Big Data Blog.

Como criar a tabela de AWS WAF

1. Copie e cole a instrução DDL a seguir no console do Athena. Modifique os campos conforme necessário para corresponder à saída do log. Modifique o valor de `LOCATION` do bucket do Amazon S3 de forma que ele corresponda a uma localização que armazena seus logs.

   Essa consulta usa o OpenX JSON SerDe.

   > **Note**
   >
   > O SerDe espera que cada documento JSON esteja em uma única linha de texto, sem caracteres de terminação de linha separando os campos no registro. Se o texto JSON estiver formatado para impressão, você poderá receber uma mensagem de erro como HIVE_CURSOR_ERROR: Row is not a valid JSON Object (HIVE_CURSOR_ERROR: a linha não é um objeto JSON válido) ou HIVE_CURSOR_ERROR: JsonParseException: Unexpected end-of-input: expected close marker for OBJECT (HIVE_CURSOR_ERROR: JSONParseException: Fim de entrada inesperado: marcador de fechamento esperado para OBJECT) quando tentar consultar a tabela após criá-la. Para obter mais informações, consulte JSON Data Files na documentação do OpenX SerDe no 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. Execute a instrução `CREATE EXTERNAL TABLE` no editor de consultas do console do Athena. Isso registra a tabela `waf_logs` e disponibiliza os dados dela para consultas no Athena.

Consultas de exemplo para logs do AWS WAF

Muitos dos exemplos a seguir consultam a tabela de projeção de partições criada anteriormente neste documento. Modifique o nome da tabela, os valores das colunas e outras variáveis de acordo com os seus requisitos. Para melhorar a performance das suas consultas e reduzir os custos, adicione a coluna de partição à condição de filtro.

- 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

Usar as funções de data e hora

- 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

Usar solicitações e endereços bloqueados

- 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 – Contar o número de indicadores que contêm um termo especificado

A consulta a seguir conta o número de indicadores que contêm o termo “amazon” para o intervalo de datas especificado.

```
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 – Contar todos os endereços IP que corresponderam a regras excluídas nos últimos 10 dias

A consulta a seguir conta o número de vezes nos últimos 10 dias que o endereço IP correspondeu à regra excluída no grupo de regras.

```
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 – Agrupar todas as regras gerenciadas contadas pelo número de vezes de correspondência

Se você definiu as ações de regras do grupo de regras como Count na configuração de ACL da Web antes de 27 de outubro de 2022, o AWS WAF salvou suas substituições no JSON da ACL da Web como `excludedRules`. Agora, a configuração JSON para substituir uma regra para Count está nas configurações `ruleActionOverrides`. Para obter mais informações, consulte Substituições de ações em grupos de regras no Guia do desenvolvedor do AWS WAF. Para extrair regras gerenciadas no modo Count da nova estrutura de logs, consulte `nonTerminatingMatchingRules` na seção `ruleGroupList` em vez do campo `excludedRules`, como no exemplo a seguir.

```
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 – Agrupar todas as regras personalizadas contadas pelo número de vezes de correspondência

A consulta a seguir agrupa todas as regras personalizadas contadas pelo número de vezes de correspondência.

```
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
```

Para obter informações sobre os locais de log de regras personalizadas e grupos de regras gerenciados, consulte Monitoramento e ajuste no Guia do desenvolvedor do AWS WAF.

Usar as funções de data e hora

Example – Retornar o campo de carimbo de data/hora no formato ISO 8601 legível

A consulta a seguir usa as funções `from_unixtime` e `to_iso8601` para retornar o campo `timestamp` no formato ISO 8601 legível (por exemplo, `2019-12-13T23:40:12.000Z` em vez de `1576280412771`). A consulta também retorna o nome, o ID e a solicitação da fonte HTTP.

```
SELECT to_iso8601(from_unixtime(timestamp / 1000)) as time_ISO_8601,
       httpsourcename,
       httpsourceid,
       httprequest
FROM waf_logs
LIMIT 10;
```

Example – Retornar os registros das últimas 24 horas

A consulta a seguir usa um filtro na cláusula `WHERE` para retornar o nome, o ID e os campos de solicitação da fonte HTTP dos registros nas últimas 24 horas.

```
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 – Retornar os registros para um intervalo de datas e endereço IP especificados

A consulta a seguir lista os registros em um intervalo de datas especificado para um endereço IP de cliente especificado.

```
SELECT *
```

```
FROM waf_logs
WHERE httprequest.clientip='53.21.198.66' AND "date" >= '2021/03/01' AND "date" <
 '2021/03/31'
```

Example – Para um intervalo de datas especificado, contar o número de endereços IP em intervalos de cinco minutos

A consulta a seguir conta, durante um determinado intervalo de datas, o número de endereços IP em intervalos de cinco minutos.

```
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 – Contar o número de X-Forwarded-For IP nos últimos 10 dias

A consulta a seguir filtra os cabeçalhos de solicitação e conta o número de X-Forwarded-For IP nos últimos 10 dias.

```
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
```

Para obter mais informações sobre as funções de data e hora, consulte Date and Time Functions and Operators (Funções e operadores de data e hora) na documentação do Trino.

Usar solicitações e endereços bloqueados

Example – Extrair os 100 primeiros endereços IP bloqueados por um tipo de regra especificado

A consulta a seguir extrai e conta os 100 primeiros endereços IP que foram bloqueados pela regra de encerramento RATE_BASED durante o intervalo de datas especificado.

```
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 – Contar o número de vezes que uma solicitação foi bloqueada de um país especificado

A consulta a seguir conta o número de vezes que a solicitação chegou de um endereço IP pertencente à Irlanda (IE) e foi bloqueada pela regra de encerramento RATE_BASED.

```
SELECT
  COUNT(httpRequest.country) as count,
  httpRequest.country
FROM waf_logs
WHERE
  terminatingruletype='RATE_BASED' AND
  httpRequest.country='IE'
GROUP BY httpRequest.country
ORDER BY count
LIMIT 100;
```

Example – Contar o número de vezes que uma solicitação foi bloqueada, agrupando por atributos específicos

A consulta a seguir conta o número de vezes que a solicitação foi bloqueada, com resultados agrupados por WebACL, RuleId, ClientIP e 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 – Contar o número de vezes em que houve correspondência com um ID de regra de encerramento específico

A consulta a seguir conta o número de vezes que um ID de regra de encerramento específico foi correspondido (`WHERE terminatingruleid='e9dd190d-7a43-4c06-bcea-409613d9506e'`). A consulta agrupa os resultados por WebACL, Action, ClientIP e 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 – Recuperar os 100 primeiros endereços IP bloqueados durante um intervalo de datas especificado

A consulta a seguir extrai os 100 primeiros endereços IP que foram bloqueados durante um intervalo de datas especificado. A consulta também lista o número de vezes que os endereços IP foram bloqueados.

```
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
```

Para obter informações sobre a consulta de logs do Amazon S3, veja os seguintes tópicos:

- Como analisar os logs de acesso do servidor do Amazon S3 usando o Athena? na Central de Conhecimento da AWS
- Consultar os logs de acesso do Amazon S3 para solicitações usando o Amazon Athena no Guia do usuário do Amazon Simple Storage Service
- Usar o AWS CloudTrail para identificar solicitações do Amazon S3 no Guia do usuário do Amazon Simple Storage Service

Consultar logs do servidor Web armazenados no Amazon S3

Você pode usar o Athena para consultar logs do servidor Web armazenados no Amazon S3. Os tópicos desta seção mostram como criar tabelas no Athena para consultar logs do servidor Web em uma variedade de formatos.

Tópicos

- Consultar logs do Apache armazenados no Amazon S3
- Consultar logs do Internet Information Server (IIS) armazenados no Amazon S3

Consultar logs do Apache armazenados no Amazon S3

Você pode usar o Amazon Athena para consultar arquivos de log do servidor HTTP do Apache armazenados em sua conta do Amazon S3. Este tópico mostra como criar esquemas de tabela para consultar os arquivos de log de acesso do Apache no formato de log comum.

Os campos no formato de log comum incluem endereço IP do cliente, ID do cliente, ID do usuário, carimbo de data/hora de recebimento da solicitação, texto da solicitação do cliente, código de status do servidor e tamanho do objeto retornado ao cliente.

O exemplo de dados a seguir mostra o formato de log comum do Apache.

```
198.51.100.7 - Li [10/Oct/2019:13:55:36 -0700] "GET /logo.gif HTTP/1.0" 200 232
198.51.100.14 - Jorge [24/Nov/2019:10:49:52 -0700] "GET /index.html HTTP/1.1" 200 2165
198.51.100.22 - Mateo [27/Dec/2019:11:38:12 -0700] "GET /about.html HTTP/1.1" 200 1287
198.51.100.9 - Nikki [11/Jan/2020:11:40:11 -0700] "GET /image.png HTTP/1.1" 404 230
198.51.100.2 - Ana [15/Feb/2019:10:12:22 -0700] "GET /favicon.ico HTTP/1.1" 404 30
198.51.100.13 - Saanvi [14/Mar/2019:11:40:33 -0700] "GET /intro.html HTTP/1.1" 200 1608
198.51.100.11 - Xiulan [22/Apr/2019:10:51:34 -0700] "GET /group/index.html HTTP/1.1"
 200 1344
```

Criar uma tabela no Athena para logs do Apache

Antes de consultar os logs do Apache armazenados no Amazon S3, você deve criar um esquema de tabela para o Athena no qual ele possa ler os dados do log. Para criar uma tabela do Athena de logs do Apache, você pode usar Grok SerDe. Para obter mais informações sobre como usar o SerDe do Grok, consulte Escrever classificadores grok personalizados no Guia do desenvolvedor do AWS Glue.

Para criar uma tabela no Athena para logs do servidor Web do Apache

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Cole a instrução DDL a seguir no editor de consultas do Athena. Modifique os valores em `LOCATION 's3://DOC-EXAMPLE-BUCKET/apache-log-folder/'` para apontar para seus logs do Apache no 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. Execute a consulta no console do Athena para registrar a tabela `apache_logs`. Quando a consulta for concluída, os logs estarão prontos para você consultar no Athena.

Exemplo de consultas select para logs do Apache

Example – Filtragem de erros 404

A consulta de exemplo a seguir seleciona a hora de recebimento da solicitação, o texto da solicitação do cliente e o código de status do servidor da tabela `apache_logs`. A cláusula `WHERE` filtra o código de status HTTP `404` (página não encontrada).

```
SELECT request_received_time, client_request, server_status
FROM apache_logs
WHERE server_status = '404'
```

A imagem a seguir mostra os resultados da consulta no editor de consultas do 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 – Filtragem de solicitações com êxito

A consulta de exemplo a seguir seleciona o ID do usuário, a hora de recebimento da solicitação, o texto da solicitação do cliente e o código de status do servidor da tabela `apache_logs`. A cláusula `WHERE` filtra o código de status HTTP `200` (com êxito).

```
SELECT user_id, request_received_time, client_request, server_status
FROM apache_logs
WHERE server_status = '200'
```

A imagem a seguir mostra os resultados da consulta no editor de consultas do 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 : filtragem por carimbo de data/hora

O exemplo a seguir consulta registros cujo horário de recebimento da solicitação é superior ao horário no carimbo de data/hora especificado.

```
SELECT * FROM apache_logs WHERE request_received_time > 10/Oct/2023:00:00:00
```

Consultar logs do Internet Information Server (IIS) armazenados no Amazon S3

Você pode usar o Amazon Athena para consultar os logs do servidor Web do Microsoft Internet Information Server (IIS) armazenados na sua conta do Amazon S3. O IIS usa uma variedade de formatos de arquivo de log, mas este tópico mostra como criar esquemas de tabela para consultar logs no formato de arquivo de log do IIs e estendido do W3C usando o Athena.

Como os formatos de arquivo de log do IIS e estendido do W3C usam delimitadores de caractere único (espaços e vírgulas, respectivamente) e não têm valores entre aspas, você pode usar o LazySimpleSerDe para criar tabelas do Athena para eles.

Formato de arquivo de log estendido do W3C

O formato de dados de arquivo de log estendido do W3C tem campos separados por espaços. Os campos que aparecem nos logs estendidos do W3C são determinados por um administrador do servidor Web que escolhe quais campos de log serão incluídos. Os dados de log de exemplo a seguir têm os campos `date, time`, `c-ip`, `s-ip`, `cs-method`, `cs-uri-stem`, `sc-status`, `sc-bytes`, `cs-bytes`, `time-taken` e `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
```

Criar uma tabela no Athena para logs estendidos do W3C

Antes de consultar os logs estendidos do W3C, você deve criar um esquema de tabela para que o Athena possa ler os dados do log.

Para criar uma tabela no Athena para logs estendidos do W3C

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Cole uma instrução DDL como a seguinte no console do Athena, observando estes pontos:

   a. Adicione ou remova as colunas no exemplo para corresponder aos campos nos logs que você deseja consultar.

   b. Os nomes de colunas no formato de arquivo de log estendido do W3C contêm hifens (-). No entanto, de acordo com as Convenções de nomenclatura do Athena, a instrução de exemplo `CREATE TABLE` os substitui por sublinhados (_).

   c. Para especificar o delimitador de espaço, use `FIELDS TERMINATED BY ' '`.

   d. Modifique os valores em `LOCATION 's3://DOC-EXAMPLE-BUCKET/w3c-log-folder/'` para apontar para seus logs estendidos do W3C no 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. Execute a consulta no console do Athena para registrar a tabela `iis_w3c_logs`. Quando a consulta for concluída, os logs estarão prontos para você consultar no Athena.

Exemplo de consulta select de log estendido do W3C

A consulta de exemplo a seguir seleciona data, hora, destino da solicitação e tempo gasto na solicitação da tabela `iis_w3c_logs`. A cláusula `WHERE` filtra os casos em que o método HTTP é `GET` e o código de status HTTP é `200` (com êxito).

```
SELECT date_col, time_col, cs_uri_stem, time_taken
FROM iis_w3c_logs
WHERE cs_method = 'GET' AND sc_status = '200'
```

A imagem a seguir mostra os resultados da consulta no editor de consultas do 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

Combinar os campos de data e hora

Os campos `date` e `time` delimitados por espaços são entradas separadas nos dados de origem do log, mas você pode combiná-los em um carimbo de data/hora, se desejar. Use as funções concat() e date_parse() em uma consulta SELECT ou CREATE TABLE AS SELECT para concatenar e converter as colunas de data e hora no formato de carimbo de data/hora. O exemplo a seguir usa a consulta CTAS para criar uma nova tabela com uma coluna `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
```

Depois de criar a tabela, consulte a nova coluna de carimbo de data/hora diretamente, como no exemplo a seguir.

```
SELECT derived_timestamp, cs_uri_stem, time_taken
FROM iis_w3c_logs_w_timestamp
WHERE cs_method = 'GET' AND sc_status = '200'
```

A imagem a seguir mostra os resultados da consulta.

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

Formato de arquivo de log do IIS

Ao contrário do formato estendido do W3C, o Formato de arquivo de log do IIS tem um conjunto fixo de campos e inclui uma vírgula como delimitador. O LazySimpleSerDe trata a vírgula como o delimitador e o espaço após a vírgula como o início do próximo campo.

O exemplo a seguir apresenta dados de amostra no formato de arquivo de log do 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, -,
```

Criar uma tabela no Athena para arquivos de log do IIS

Para consultar os logs no formato de arquivo de log do IIS no Amazon S3, crie primeiro um esquema de tabela para que o Athena possa ler os dados do log.

Para criar uma tabela no Athena para logs no formato de arquivo de log do IIS

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Cole a seguinte instrução DDL no console do Athena, observando estes pontos:
   a. Para especificar a vírgula como delimitador, use `FIELDS TERMINATED BY ','`.
   b. Modifique os valores em LOCATION 's3://DOC-EXAMPLE-BUCKET/*iis-log-file-folder*/' para apontar para seus arquivos de log no formato de log do IIS no 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. Execute a consulta no console do Athena para registrar a tabela `iis_format_logs`. Quando a consulta for concluída, os logs estarão prontos para você consultar no Athena.

Exemplo de consulta select no formato de log do IIS

A consulta de exemplo a seguir seleciona data, hora, destino da solicitação e tempo gasto em milissegundos da tabela `iis_format_logs`. A cláusula `WHERE` filtra os casos em que o tipo de solicitação é `GET` e o código de status HTTP é `200` (com êxito). Na consulta, observe que os espaços à esquerda em `' GET'` e `' 200'` são necessários para que a consulta seja bem-sucedida.

```
SELECT request_date, request_time, target_of_operation, time_taken_millisec
FROM iis_format_logs
WHERE request_type = ' GET' AND service_status_code = ' 200'
```

A imagem a seguir mostra os resultados da consulta dos dados de exemplo.

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

Formato de arquivo de log do NCSA

O IIS também usa o formato de log do NCSA, que tem um número fixo de campos em formato de texto ASCII separados por espaços. A estrutura é semelhante ao formato de log comum usado para logs de acesso do Apache. Os campos no formato de dados de log comum do NCSA incluem endereço IP do cliente, ID do cliente (não costuma ser usado), ID do usuário do domínio, carimbo de data/hora de recebimento da solicitação, texto da solicitação do cliente, código de status do servidor e tamanho do objeto retornado ao cliente.

O exemplo a seguir mostra os dados no formato de log comum do NCSA, conforme documentado para o 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
```

Criar uma tabela no Athena para logs do NCSA do IIS

Na instrução `CREATE TABLE`, você pode usar Grok SerDe e um padrão do grok similar ao usado nos logs do servidor Web do Apache. Ao contrário dos logs do Apache, o padrão do grok usa `%{DATA:user_id}` para o terceiro campo, em vez de `%{USERNAME:user_id}` para considerar a presença da barra invertida em `domain\user_id`. Para obter mais informações sobre como usar o SerDe do Grok, consulte Escrever classificadores grok personalizados no Guia do desenvolvedor do AWS Glue.

Para criar uma tabela no Athena para logs do servidor Web do NCSA do IIS

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Cole a instrução DDL a seguir no editor de consultas do Athena. Modifique os valores em `LOCATION 's3://DOC-EXAMPLE-BUCKET/iis-ncsa-logs/'` para apontar para seus logs do NCSA do IIS no 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. Execute a consulta no console do Athena para registrar a tabela `iis_ncsa_logs`. Quando a consulta for concluída, os logs estarão prontos para você consultar no Athena.

Exemplo de consultas Select para logs do NCSA do IIS

Example – Filtragem de erros 404

A consulta de exemplo a seguir seleciona a hora de recebimento da solicitação, o texto da solicitação do cliente e o código de status do servidor da tabela `iis_ncsa_logs`. A cláusula `WHERE` filtra o código de status HTTP `404` (página não encontrada).

```
SELECT request_received_time, client_request, server_status
FROM iis_ncsa_logs
WHERE server_status = '404'
```

A imagem a seguir mostra os resultados da consulta no editor de consultas do Athena.

Results

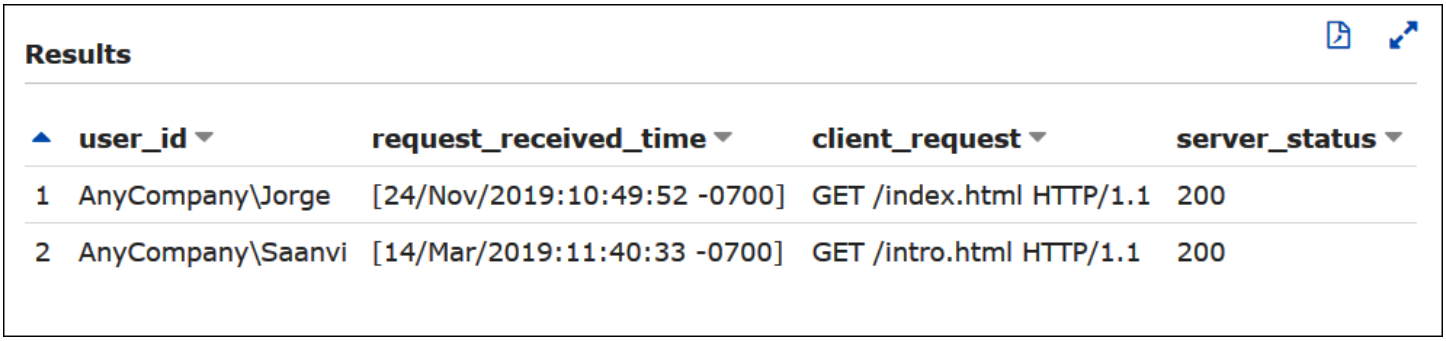
	request_received_time	client_request	server_status
1	[11/Jan/2020:11:40:11 -0700]	GET /image.png HTTP/1.1	404
2	[15/Feb/2019:10:12:22 -0700]	GET /favicon.ico HTTP/1.1	404

Example – Filtragem de solicitações com êxito de um domínio específico

A consulta de exemplo a seguir seleciona o ID do usuário, a hora de recebimento da solicitação, o texto da solicitação do cliente e o código de status do servidor da tabela `iis_ncsa_logs`. A cláusula `WHERE` filtra as solicitações com código de status HTTP `200`(com êxito) de usuários no domínio `AnyCompany`.

```
SELECT user_id, request_received_time, client_request, server_status
FROM iis_ncsa_logs
WHERE server_status = '200' AND user_id LIKE 'AnyCompany%'
```

A imagem a seguir mostra os resultados da consulta no editor de consultas do 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

Usar transações ACID do Athena

O termo “transações ACID” se refere a um conjunto de propriedades (atomicidade, consistência, isolamento e durabilidade) que garantem a integridade dos dados nas transações de banco de dados. As transações ACID permitem que vários usuários adicionem e excluam objetos do Amazon S3 de maneira atômica, simultaneamente e com confiabilidade, ao mesmo tempo que isolam as consultas existentes mantendo a consistência de leitura para consultas no data lake. As transações ACID do Athena adicionam operações de inserção, exclusão, atualização e viagem no tempo à linguagem de manipulação de dados (DML) SQL do Athena. Você e vários usuários podem usar as transações ACID do Athena simultaneamente para fazer modificações confiáveis nos dados do Amazon S3 no nível da linha. As transações do Athena gerenciam automaticamente a semântica e a coordenação dos bloqueios e não requerem uma solução personalizada de bloqueio de registros.

As transações do ACID do Athena e a sintaxe SQL familiar simplificam as atualizações de dados empresariais e regulamentares. Por exemplo, para responder a uma solicitação de eliminação de dados, você pode executar uma operação `DELETE` em SQL. Para fazer correções manuais em registros, você pode usar uma única instrução `UPDATE`. Para recuperar dados que foram excluídos recentemente, você pode emitir consultas de viagem no tempo usando uma instrução `SELECT`.

Como elas são montadas em formatos de tabela compartilhada, as transações ACID do Athena são compatíveis com outros serviços e mecanismos, como o Amazon EMR e o Apache Spark que também suportam formatos de tabela compartilhada.

As transações do Athena estão disponíveis por meio do console do Athena, de operações de API e dos drivers ODBC e JDBC.

Tópicos

- Consultar tabelas do Linux Foundation Delta Lake
- Usar o Athena para consultar conjuntos de dados do Apache Hudi
- Usar tabelas do Apache Iceberg

Consultar tabelas do Linux Foundation Delta Lake

O Linux Foundation Delta Lake é um formato de tabela para análise de big data. É possível usar o Amazon Athena para ler tabelas Delta Lake armazenadas no Amazon S3 de forma direta, sem a necessidade de gerar arquivos de manifesto ou executar a instrução `MSCK REPAIR`.

O formato Delta Lake armazena os valores mínimo e máximo de cada arquivo de dados por coluna. A implementação do Athena usa essas informações para habilitar o salto de arquivos em predicados com a finalidade de eliminar arquivos indesejados da consideração.

Considerações e limitações

O suporte para o Delta Lake no Athena tem as limitações e considerações a seguir:

- Somente tabelas com catálogo do AWS Glue: o suporte nativo do Delta Lake é compatível somente por meio de tabelas registradas no AWS Glue. Se você tiver uma tabela Delta Lake registrada em outro metastore, ainda poderá mantê-la e tratá-la como metastore principal. Como os metadados do Delta Lake são armazenados no sistema de arquivos (por exemplo, no Amazon S3) e não no metastore, o Athena requer somente a propriedade “location” no AWS Glue para ler as tabelas Delta Lake.
- Somente versão 3 do mecanismo: as consultas do Delta Lake tem suporte somente na versão 3 do mecanismo do Athena. Você deve garantir que o grupo de trabalho criado esteja configurado para usar a versão 3 do mecanismo do Athena.
- Versão do leitor Delta Lake — O protocolo do leitor Delta Lake até a versão 3 é suportado.

- Mapeamento de colunas e timestampNtz — O mapeamento de colunas Delta, que permite que as colunas da tabela Delta e as colunas subjacentes do arquivo Parquet usem nomes diferentes, e o timestamp sem fuso horário (timestampNtz) são suportados.
- Nenhum suporte para passagem de tempo: não há suporte para consultas que usam as funcionalidades de passagem de tempo do Delta Lake.
- Somente leitura: não há suporte para gravar instruções DML, como `UPDATE`, `INSERT` ou `DELETE`.
- Compatibilidade com o Lake Formation: a integração com o Lake Formation está disponível para tabelas do Delta Lake com seu esquema sincronizado com o AWS Glue. Para obter mais informações, consulte Usar o AWS Lake Formation com o Amazon Athena e Configurar permissões para uma tabela do Delta Lake no Guia de desenvolvedor do AWS Lake Formation.
- Suporte DDL limitado: as instruções DDL a seguir são compatíveis: `CREATE EXTERNAL TABLE`, `SHOW COLUMNS`, `SHOW TBLPROPERTIES`, `SHOW PARTITIONS`, `SHOW CREATE TABLE` e `DESCRIBE`. Para obter informações sobre como usar a instrução `CREATE EXTERNAL TABLE`, consulte a seção Conceitos básicos.
- Não é possível ignorar objetos do S3 Glacier: se os objetos na tabela do Linux Foundation Delta Lake estiverem em uma classe de armazenamento do Amazon S3 Glacier, definir a propriedade da tabela `read_restored_glacier_objects` como `false` não terá efeito.

  Por exemplo, suponhamos que você emita o seguinte comando:

  ```
  ALTER TABLE table_name SET TBLPROPERTIES ('read_restored_glacier_objects' = 'false')
  ```

  Para tabelas do Iceberg e do Delta Lake, o comando gera o erro Chave de propriedade da tabela não compatível: read_restored_glacier_objects. Para tabelas do Hudi, o comando `ALTER TABLE` não gera um erro, mas os objetos do Amazon S3 Glacier ainda assim não são ignorados. A execução de consultas `SELECT` após o comando `ALTER TABLE` continua a retornar todos os objetos.

Tipos de dados compatíveis para colunas sem partições

Para colunas sem partições, há suporte para todos os tipos de dados compatíveis com o Athena, exceto `CHAR`. Não há suporte para `CHAR` no próprio protocolo do Delta Lake). Os tipos de dados compatíveis incluem:

```
boolean
tinyint
```

```
smallint
integer
bigint
double
float
decimal
varchar
string
binary
date
timestamp
array
map
struct
```

Tipos de dados compatíveis para colunas com partições

Para colunas com partições, o Athena é compatível com tabelas com os tipos de dados a seguir:

```
boolean
integer
smallint
tinyint
bigint
decimal
float
double
date
timestamp
varchar
```

Para obter mais informações sobre os tipos de dados no Athena, consulte Tipos de dados no Amazon Athena.

Conceitos básicos

Para ser consultável, a tabela Delta Lake deve existir no AWS Glue. Se a tabela estiver no Amazon S3, mas não estiver no AWS Glue, execute uma instrução `CREATE EXTERNAL TABLE` usando a sintaxe a seguir. Se a tabela já existe no AWS Glue (por exemplo, porque você está usando o Apache Spark ou outro mecanismo com o AWS Glue), você pode ignorar esta etapa.

```
CREATE EXTERNAL TABLE
```

```
  [db_name.]table_name
  LOCATION 's3://DOC-EXAMPLE-BUCKET/your-folder/'
  TBLPROPERTIES ('table_type' = 'DELTA')
```

Observe a omissão das definições das colunas, da biblioteca SerDe e de outras propriedades da tabela. Ao contrário das tabelas Hive tradicionais, os metadados da tabela Delta Lake são inferidos do log de transações do Delta Lake e sincronizados diretamente com o AWS Glue.

> **Note**
>
> Para tabelas Delta Lake, as instruções `CREATE TABLE` que incluem mais do que as propriedades `LOCATION` e `table_type` não são permitidas.

Leitura de tabelas Delta Lake

Para consultar uma tabela Delta Lake, use a sintaxe SQL padrão `SELECT`:

```
[ 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 ] ]
```

Para obter mais informações sobre a sintaxe `SELECT`, consulte SELECT na documentação do Athena.

O formato Delta Lake armazena os valores mínimo e máximo de cada arquivo de dados por coluna. O Athena usa essas informações para habilitar o salto de arquivo em predicados com a finalidade de eliminar arquivos desnecessários da consideração.

Sincronização de metadados do Delta Lake

O Athena sincroniza os metadados da tabela, incluindo o esquema, as colunas com partições e as propriedades da tabela, para o AWS Glue se você usar o Athena para criar a tabela Delta Lake. Com o passar do tempo, esses metadados podem perder a sincronização com os metadados da

tabela subjacente no log de transações. Para manter a tabela atualizada, escolha uma das seguintes opções:

- Use o crawler do AWS Glue para tabelas Delta Lake. Para obter mais informações, consulte Introdução de compatibilidade com tabelas nativas do Delta Lake com crawlers doAWS Glue no blog sobre AWS Big Data e Agendar um crawler no AWS Glue no Guia do desenvolvedor do AWS Glue.
- Solte e recrie a tabela no Athena.
- Use o SDK, a CLI ou o console do AWS Glue para atualizar manualmente o esquema no AWS Glue.

Observe que os recursos a seguir exigem que seu esquema do AWS Glue sempre tenha o mesmo esquema do log de transações:

- Lake Formation
- Visões
- Filtros de linha e coluna

Se o seu fluxo de trabalho não exigir nenhuma dessas funcionalidades e você preferir não manter essa compatibilidade, pode usar o DDL `CREATE TABLE` no Athena e depois adicionar o caminho do Amazon S3 como um parâmetro SerDe no AWS Glue.

Para criar uma tabela Delta Lake usando o Athena e consoles do AWS Glue

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. No editor de consultas do Athena, use o seguinte DDL para criar sua tabela Delta Lake. Observe que, ao usar esse método, o valor de `TBLPROPERTIES` deve ser `'spark.sql.sources.provider' = 'delta'`, e não `'table_type' = 'delta'`.

   Observe que esse mesmo esquema (com uma única coluna chamada `col` de tipo `array<string>`) é inserido quando você usa o Apache Spark (Athena para Apache Spark) ou a maioria dos outros mecanismos para criar sua tabela.

```
CREATE EXTERNAL TABLE
   [db_name.]table_name(col array<string>)
   LOCATION 's3://DOC-EXAMPLE-BUCKET/your-folder/'
   TBLPROPERTIES ('spark.sql.sources.provider' = 'delta')
```

3. Abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
4. No painel de navegação, escolha Catálogo de dados, Tabelas.
5. Na lista de tabelas, escolha o link da sua tabela.
6. Na página da tabela, escolha Ações, Editar tabela.
7. Na seção Parâmetros do Serde, adicione a chave **`path`** com o valor **`s3://DOC-EXAMPLE-BUCKET/your-folder/`**.
8. Escolha Salvar.

Recursos adicionais do

Para uma discussão sobre o uso de tabelas do Delta Lake com AWS Glue e fazendo consultas com o Athena, consulte Gerencie as operações de dados do UPSERT usando o Delta Lake de código aberto e AWS Glue no AWSBlog Big Data.

Usar o Athena para consultar conjuntos de dados do Apache Hudi

O Apache Hudi é um framework de gerenciamento de dados de código aberto que simplifica o processamento incremental de dados. As ações de inserção, atualização, upsert e exclusão no nível do registro são processadas de forma muito mais granular, reduzindo a sobrecarga. `Upsert` refere-se à capacidade de inserir registros em um conjunto de dados existente, se eles ainda não existirem, ou atualizá-los, se já existirem.

O Hudi processa os eventos de inserção e atualização de dados sem criar muitos arquivos pequenos que podem causar problemas de performance nas análises. O Apache Hudi monitora automaticamente as alterações e mescla os arquivos para que mantenham o dimensionamento ideal. Isso evita a necessidade de criar soluções personalizadas que monitoram e regravam muitos arquivos pequenos em menos arquivos grandes.

Os conjuntos de dados do Hudi são adequados para os seguintes casos de uso:

- Cumprir os regulamentos de privacidade, como o Regulamento geral de proteção de dados (RGPD) e o California Consumer Privacy Act (CCPA), que determinam o direito das pessoas de remover informações pessoais ou alterar o modo como os dados são utilizados.
- Trabalhar com dados de streaming de sensores e outros dispositivos da Internet das Coisas (IoT) que exigem eventos específicos de inserção e atualização de dados.
- Implementar o sistema de Change Data Capture (CDC – Captura de dados de alteração).

Os conjuntos de dados gerenciados pelo Hudi são armazenados no Amazon S3 em formatos de armazenamento abertos. Atualmente, o Athena pode ler conjuntos de dados compactados do Hudi, mas não pode gravar dados do Hudi. O Athena é compatível com o Hudi até a versão 0.8.0 com o mecanismo Athena versão 2 e com o Hudi versão 0.14.0 com o mecanismo Athena versão 3. Isso está sujeito a alteração. O Athena não pode garantir a compatibilidade de leitura com tabelas criadas com versões posteriores do Hudi. Para obter informações sobre o versionamento do mecanismo do Athena, consulte Versionamento do mecanismo do Athena. Para obter mais informações sobre os recursos e o versionamento do Hudi, consulte a documentação do Hudi no site do Apache.

Tipos de tabela dos conjuntos de dados do Hudi

O tipo de conjunto de dados do Hudi pode ser um destes:

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

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

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

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

- Ler consultas otimizadas: para tabelas MoR, as consultas veem os dados mais recentes compactados. Para tabelas CoW, as consultas veem os dados mais recentes confirmados.

A tabela a seguir mostra os possíveis tipos de consulta do Hudi para cada tipo de tabela.

Tipo de tabela	Possíveis tipos de consulta do Hudi
Copiar na gravação	snapshot, incremental
Mesclar na leitura	snapshot, incremental, otimizado para leitura

Atualmente, o Athena permite consultas de snapshot e consultas otimizadas para leitura, mas não permite consultas incrementais. Nas tabelas MoR, todos os dados expostos a consultas otimizadas para leitura são compactados. Isso permite uma boa performance, mas não inclui as confirmações delta mais recentes. As consultas de snapshot contêm os dados mais recentes, mas geram um pouco de sobrecarga computacional, o que reduz a performance dessas consultas.

Para obter mais informações sobre as vantagens e desvantagens dos tipos de tabela e consulta, acesse Table & Queries Types na documentação do Apache Hudi.

Alteração de terminologia do Hudi: as visualizações agora são consultas

A partir da versão 0.5.1, o Apache Hudi alterou uma parte da terminologia. As antigas visualizações agora são chamadas de consultas nas versões mais recentes. A tabela a seguir resume as alterações entre os termos antigos e novos.

Termo antigo	Termo novo
CoW: visualização otimizada para leitura MoR: visualização em tempo real	Consultas de snapshot

Termo antigo	Termo novo
Visualização incremental	Consulta incremental
MoR: visualização otimizada para leitura	Consulta otimizada para leitura

Tabelas de operação de bootstrap

A partir do Apache Hudi versão 0.6.0, o recurso de operação de bootstrap oferece melhor performance com conjuntos de dados do Parquet existentes. Em vez de reescrever o conjunto de dados, uma operação de bootstrap pode gerar apenas os metadados, sem alterar o conjunto de dados.

Você pode usar o Athena para consultar tabelas de uma operação de bootstrap, assim como as outras tabelas baseadas em dados no Amazon S3. Na instrução `CREATE TABLE`, especifique o caminho da tabela do Hudi na cláusula `LOCATION`.

Para obter mais informações sobre como criar tabelas do Hudi usando a operação de bootstrap no Amazon EMR, consulte o artigo New features from Apache Hudi available in Amazon EMR (Novos recursos do Apache Hudi disponíveis no Amazon EMR) no blog da AWS sobre big data.

Listagem de metadados do Hudi

O Apache Hudi tem uma tabela de metadados que contém recursos de indexação para melhorar a performance, como listagem de arquivos, possibilidade de ignorar dados usando estatísticas de coluna e um índice baseado em filtro de Bloom.

Desses recursos, o Athena atualmente só é compatível com o índice de listagem de arquivos. O índice de listagem de arquivos elimina chamadas do sistema de arquivos, como "listar arquivos", ao buscar as informações de um índice que mantém uma partição para mapeamento de arquivos. Isso elimina a necessidade de listar recursivamente todas as partições no caminho da tabela para obter uma visão do sistema de arquivos. Quando você trabalha com grandes conjuntos de dados, essa indexação reduz muito a latência que, de outra forma, ocorreria ao obter arquivos durante gravações e consultas. Também evita gargalos, como controle de utilização de solicitações nas chamadas `LIST` do Amazon S3.

> **Note**
>
> No momento, o Athena não é compatível com a possibilidade de ignorar dados nem com a indexação com filtro de Bloom.

Habilitar tabela de metadados do Hudi

A listagem de arquivos baseada em tabela de metadados é desabilitada por padrão. Para habilitar a tabela de metadados do Hudi e a correspondente funcionalidade de listagem de arquivos, defina a propriedade da tabela `hudi.metadata-listing-enabled` como `TRUE`.

Exemplo

O exemplo de `ALTER TABLE SET TBLPROPERTIES` a seguir habilita a tabela de metadados `partition_cow` no exemplo.

```
ALTER TABLE partition_cow SET TBLPROPERTIES('hudi.metadata-listing-enabled'='TRUE')
```

Considerações e limitações

- O Athena não aceita consultas incrementais.
- O Athena não aceita CTAS nem INSERT INTO em dados do Hudi. Se você quiser ajuda do Athena para escrever conjuntos de dados do Hudi, envie feedback para `<athena-feedback@amazon.com>`.

  Para obter mais informações sobre como escrever dados do Hudi, consulte os seguintes recursos:

  - Working with a Hudi dataset (Trabalhar com um conjunto de dados do Hudi) no Guia de apresentação do Amazon EMR.
  - Writing Data (Gravar dados) na documentação do Apache Hudi.
- Não é permitido usar MSCK REPAIR TABLE em tabelas do Hudi no Athena. Se você precisar carregar uma tabela do Hudi que não foi criada no AWS Glue, use ALTER TABLE ADD PARTITION.
- Não é possível ignorar objetos do S3 Glacier: se os objetos na tabela do Apache Hudi estiverem em uma classe de armazenamento do Amazon S3 Glacier, definir a propriedade da tabela `read_restored_glacier_objects` como `false` não terá efeito.

  Por exemplo, suponhamos que você emita o seguinte comando:

```
ALTER TABLE table_name SET TBLPROPERTIES ('read_restored_glacier_objects' = 'false')
```

Para tabelas do Iceberg e do Delta Lake, o comando gera o erro Chave de propriedade da tabela não compatível: read_restored_glacier_objects. Para tabelas do Hudi, o comando `ALTER TABLE` não gera um erro, mas os objetos do Amazon S3 Glacier ainda assim não são ignorados. A execução de consultas `SELECT` após o comando `ALTER TABLE` continua a retornar todos os objetos.

Vídeo

O vídeo a seguir mostra como você pode usar o Amazon Athena para consultar um conjunto de dados do Apache Hudi otimizado para leitura em seu data lake baseado no Amazon S3.

Query Apache Hudi datasets using Amazon Athena (Usar o Amazon Athena para consultar conjuntos de dados do Apache Hudi)

Criar tabelas do Hudi

Esta seção apresenta exemplos de instruções CREATE TABLE no Athena para tabelas particionadas e não particionadas de dados do Hudi.

Se você já tem tabelas do Hudi criadas no AWS Glue, pode consultá-las diretamente no Athena. Ao criar tabelas do Hudi particionadas no Athena, você deve executar `ALTER TABLE ADD PARTITION` para carregar os dados do Hudi antes de poder consultá-los.

Exemplos de criação de tabelas do tipo Copiar na gravação (CoW)

Tabela CoW não particionada

O exemplo a seguir cria uma tabela CoW não particionada no 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/'
```

Tabela CoW particionada

O exemplo a seguir cria uma tabela CoW particionada no 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/'
```

O exemplo `ALTER TABLE ADD PARTITION` a seguir adiciona duas partições à tabela `partition_cow` de exemplo.

```
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/'
```

Exemplos de criação de tabelas do tipo Mesclar na leitura (MoR)

O Hudi cria duas tabelas no metastore para MoR: uma tabela para consultas de snapshot e uma tabela para consultas otimizadas para leitura. As duas tabelas podem ser consultadas. Nas versões do Hudi anteriores à 0.5.1, a tabela de consultas otimizadas para leitura tinha o nome que você especificava ao criá-la. A partir do Hudi versão 0.5.1, o nome da tabela recebe o sufixo `_ro` por padrão. O nome da tabela de consultas de snapshot é aquele que você especifica com `_rt` acrescentado.

Tabela MoR não particionada

O exemplo a seguir cria uma tabela MoR não particionada no Athena para consultas otimizadas para leitura. Observe que as consultas otimizadas para leitura usam o formato de entrada `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/'
```

O exemplo a seguir cria uma tabela MoR não particionada no Athena para consultas de snapshot. Para consultas de snapshot, use o formato de entrada `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/'
```

Tabela MoR particionada

O exemplo a seguir cria uma tabela MoR particionada no Athena para consultas otimizadas para leitura.

```
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/'
```

O exemplo `ALTER TABLE ADD PARTITION` a seguir adiciona duas partições à tabela `partition_mor` de exemplo.

```
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/'
```

O exemplo a seguir cria uma tabela MoR particionada no Athena para consultas de 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/'
```

Da mesma forma, o exemplo `ALTER TABLE ADD PARTITION` a seguir adiciona duas partições à tabela `partition_mor_rt` de exemplo.

```
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/'
```

Recursos adicionais do

- Para obter informações sobre como usar os conectores personalizados do AWS Glue e trabalhos do AWS Glue 2.0 para criar uma tabela Apache Hudi que você possa consultar com o Athena, consulte Gravar em tabelas do Apache Hudi usando o conector personalizado do AWS Glue no AWS Big Data Blog.
- Para acessar um artigo sobre como usar o Apache Hudi, o AWS Glue e o Amazon Athena para criar uma estrutura de processamento de dados para um data lake, consulte Simplificar o processamento operacional de dados em data lakes usando o AWS Glue e o Apache Hudi no blog AWS Big Data.

Usar tabelas do Apache Iceberg

O Athena oferece suporte a consultas de leitura, viagem no tempo, gravação e DDL para tabelas Apache Iceberg que usam o formato Apache Parquet para dados e o catálogo do AWS Glue para metastore.

Apache Iceberg é um formato de tabela aberta para conjuntos de dados analíticos muito grandes. O Iceberg gerencia grandes coleções de arquivos como tabelas e suporta modernas operações de data lake analítico, como inserção, atualização, exclusão no nível do registro e consultas de viagem no tempo. A especificação do Iceberg permite uma evolução contínua da tabela, como uma evolução de esquema e partição, e é concebida para otimizar o uso no Amazon S3. O Iceberg também ajuda a garantir a correção dos dados em cenários de gravação simultâneos.

Para obter mais informações sobre o Apache Iceberg, consulte https://iceberg.apache.org/.

Considerações e limitações

O suporte do Athena para tabelas Iceberg tem as seguintes considerações e limitações:

- Suporte à versão Iceberg — O Athena é compatível com o Apache Iceberg versão 1.4.2.
- Tabelas somente com catálogo do AWS Glue: apenas tabelas Iceberg criadas com base no catálogo do AWS Glue de acordo com as especificações definidas pela implementação de código aberto do catálogo do Glue são compatíveis com o Athena.

- Suporte de bloqueio de tabela apenas pelo AWS Glue: ao contrário da implementação de código aberto do catálogo do Glue, que oferece suporte ao bloqueio personalizado de plug-in, o Athena oferece suporte apenas ao bloqueio otimista do AWS Glue. Usar o Athena para modificar uma tabela do Iceberg com qualquer outra implementação de bloqueio causará perda de dados e quebra de transações.
- Formatos de arquivo compatíveis: a compatibilidade do formato de arquivo do Iceberg no Athena depende da versão do mecanismo do Athena, conforme mostrado na tabela a seguir.

Versão do mecanismo do Athena	Parquet	ORC	Avro
2	Sim	Não	Não
3	Sim	Sim	Sim

- Tabelas Iceberg v2: o Athena só cria e opera tabelas Iceberg v2. Para saber a diferença entre as tabelas da v1 e v2, consulte Alterações de versão do formato na documentação do Apache Iceberg.
- Exibição de tipos de hora sem fuso horário: a hora e o carimbo de data/hora sem tipos de fuso horário são exibidos no UTC. Se o fuso horário não for especificado em uma expressão de filtro em uma coluna de hora, o UTC será usado.
- Precisão dos dados relacionados ao carimbo de data e hora: embora o Iceberg seja compatível com uma precisão de microssegundos para o tipo de dados de carimbo de data e hora, o Athena é compatível apenas com a precisão de milissegundos para carimbos de data e hora em leituras e gravações. Para dados em colunas relacionadas a horas gravadas durante operações de compactação manual, o Athena retém somente uma precisão de milissegundos.
- Operações sem suporte: as seguintes operações do Athena não são compatíveis com tabelas Iceberg.
  - ALTER TABLE SET LOCATION
- Visualizações: use `CREATE VIEW` para criar visualizações do Athena, conforme descrito em Trabalhar com visualizações. Se você tiver interesse em usar a especificação de visualização do Iceberg para criar visualizações, entre em contato com athena-feedback@amazon.com.
- Comandos de gerenciamento de TTF não compatíveis com o AWS Lake Formation: embora seja possível usar o Lake Formation para gerenciar permissões de acesso de leitura para TransactionTable Formats (TTFs), como Apache Iceberg, Apache Hudi e Linux Foundation Delta Lake, não é possível usar o Lake Formation para gerenciar permissões para operações

como `VACUUM`, `MERGE`, `UPDATE` ou `OPTIMIZE` com esses formatos de tabela. Para obter mais informações sobre a integração do Lake Formation com o Athena, consulte Usar o AWS Lake Formation com o Amazon Athena no Guia do desenvolvedor do AWS Lake Formation.

- Particionar por campos aninhados - A partição por campos aninhados não é suportada. A tentativa de fazer isso produz a mensagem NÃO SUPORTADO: *`particionar por campo aninhado não é suportado: column_name.nested_field_name`*.
- Não é possível ignorar objetos do S3 Glacier: se os objetos na tabela do Apache Iceberg estiverem em uma classe de armazenamento do Amazon S3 Glacier, definir a propriedade da tabela `read_restored_glacier_objects` como `false` não terá efeito.

  Por exemplo, suponhamos que você emita o seguinte comando:

  ```
  ALTER TABLE table_name SET TBLPROPERTIES ('read_restored_glacier_objects' = 'false')
  ```

  Para tabelas do Iceberg e do Delta Lake, o comando gera o erro Chave de propriedade da tabela não compatível: read_restored_glacier_objects. Para tabelas do Hudi, o comando `ALTER TABLE` não gera um erro, mas os objetos do Amazon S3 Glacier ainda assim não são ignorados. A execução de consultas `SELECT` após o comando `ALTER TABLE` continua a retornar todos os objetos.

Se quiser que o Athena ofereça suporte a recurso específico, envie comentários para athena-feedback@amazon.com.

Tópicos

- Criar tabelas Iceberg
- Gerenciar tabelas Iceberg
- Consultar metadados de tabela do Iceberg
- Esquema de tabela Iceberg em evolução
- Consultar dados em tabelas Iceberg e realizar viagens no tempo
- Atualizar dados nas tabelas Iceberg
- Otimizar tabelas Iceberg
- Tipos de dados suportados para tabelas Iceberg no Athena
- Outras operações do Athena em tabelas Iceberg
- Recursos adicionais do

Criar tabelas Iceberg

Para criar uma tabela Iceberg para uso no Athena, você pode usar uma instrução `CREATE TABLE` conforme documentada nesta página, ou pode usar um crawler do AWS Glue.

Usar uma instrução CREATE TABLE

O Athena cria tabelas do Iceberg v2. Para saber a diferença entre as tabelas da v1 e v2, consulte Alterações de versão do formato na documentação do Apache Iceberg.

A cláusula `CREATE TABLE` do Athena cria uma tabela Iceberg sem dados. Você poderá consultar uma tabela diretamente de sistemas externos, como o Apache Spark, se a tabela usar o catálogo do Glue de código aberto do Iceberg. Você não precisa criar uma tabela externa.

> **Warning**
>
> Executar `CREATE EXTERNAL TABLE` resulta na mensagem de erro External keyword not supported for table type ICEBERG (Palavra-chave externa não compatível com o tipo de tabela ICEBERG).

Para criar uma tabela Iceberg no Athena, defina a propriedade `'table_type'` da tabela como `'ICEBERG'` na cláusula `TBLPROPERTIES`, como no resumo da sintaxe a seguir.

```
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] )
```

Para obter informações sobre os tipos de dados que você pode consultar em tabelas Iceberg, consulte Tipos de dados suportados para tabelas Iceberg no Athena.

Particionamento

Para criar tabelas Iceberg com partições, use a sintaxe `PARTITIONED BY`. As colunas usadas para particionamento devem ser especificadas primeiro nas declarações de colunas. O tipo de coluna não deve ser incluído dentro da cláusula `PARTITIONED BY`. Você também pode definir transformações de partição na sintaxe `CREATE TABLE`. Para especificar várias colunas para particionamento, separe as colunas por vírgula (`,`), como no exemplo a seguir.

```
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' )
```

A tabela a seguir mostra as funções de transformação de partição disponíveis.

Função	Descrição	Tipos compatíveis
`year(ts)`	Partição por ano	`date`, `timestamp`
`month(ts)`	Partição por mês	`date`, `timestamp`
`day(ts)`	Partição por dia	`date`, `timestamp`
`hour(ts)`	Partição por hora	`timestamp`
`bucket(N, col)`	Partição por buckets *N* de mod de valor por hash. Esse conceito é semelhante ao bucket por hash para tabelas do Hive.	`int`, `long`, `decimal`, `date`, `timestamp` , `string`, `binary`
`truncate( L, col)`	Partição por valor truncada para *L*	`int`, `long`, `decimal`, `string`

O Athena suporta o particionamento oculto do Iceberg. Para obter mais informações, consulte Particionamento oculto do Iceberg na documentação do Apache Iceberg.

Propriedade das tabelas

Esta seção descreve as propriedades de tabela que você pode especificar como pares de valores-chave na cláusula `TBLPROPERTIES` da instrução `CREATE TABLE`. O Athena permite apenas uma lista predefinida de pares de valores-chave nas propriedades da tabela para criar ou alterar tabelas Iceberg. As tabelas a seguir mostram as propriedades de tabela que você pode especificar. Para

obter mais informações sobre essas opções de compactação, consulte Otimizar tabelas Iceberg mais adiante neste documento. Se você quiser que o Athena suporte uma propriedade específica de configuração de tabela de código aberto, envie comentários para athena-feedback@amazon.com.

format

Descrição	Formato de dados do arquivo
Valores de propriedade permitidos	O formato de arquivo compatível e as combinações de compactaç ão variam conforme a versão do mecanismo Athena. Para ter mais informações, consulte Suporte à compressão de tabelas do Iceberg por formato de arquivo.
Valor padrão	parquet

write_compression

Descrição	Codec de compactação de arquivo
Valores de propriedade permitidos	O formato de arquivo compatível e as combinações de compactaç ão variam conforme a versão do mecanismo Athena. Para ter mais informações, consulte Suporte à compressão de tabelas do Iceberg por formato de arquivo.
Valor padrão	A compactação de gravação padrão varia conforme a versão do mecanismo Athena. Para ter mais informações, consulte Suporte à compressão de tabelas do Iceberg por formato de arquivo.

optimize_rewrite_data_file_threshold

Descrição	Configuração específica de otimização de dados. Se houver menos arquivos de dados que exigem otimização do que o limite fornecido, os arquivos não serão regravados. Isso permite acumular mais arquivos de dados para produzir arquivos mais próximos do tamanho de destino e ignorar computação desnecessária para economizar custos.

Valores de propriedade permitidos	Um número positivo. Deve ser menor que 50.
Valor padrão	5

optimize_rewrite_delete_file_threshold

Descrição	Configuração específica de otimização de dados. Se houver menos arquivos de exclusão associados a um arquivo de dados do que o limite, o arquivo de dados não será regravado. Isso permite acumular mais arquivos de exclusão para cada arquivo de dados para economizar custos.
Valores de propriedade permitidos	Um número positivo. Deve ser menor que 50.
Valor padrão	2

vacuum_min_snapshots_to_keep

Descrição	Número mínimo de snapshots a serem retidos na ramificação principal de uma tabela. Esse valor tem precedência sobre a propriedade `vacuum_max_snapshot_age_seconds`. Se o mínimo restante de snapshots for mais antigo do que a idade especificada por `vacuum_max_snapshot_age_seconds`, os snapshots serão mantidos e o valor de `vacuum_max_snapshot_age_seconds` será ignorado.
Valores de propriedade permitidos	Um número positivo.
Valor padrão	1

vacuum_max_snapshot_age_seconds

Descrição	Período máximo para reter os snapshots na ramificação principal. Esse valor será ignorado se o mínimo restante de snapshots especificado por `vacuum_min_snapshots_to_keep` for maior que a idade especificada. Essa propriedade de comportamento da tabela correspon de à propriedade `history.expire.max-snapshot-age-ms` na configuração do Apache Iceberg.
Valores de proprieda de permitidos	Um número positivo.
Valor padrão	432 mil segundos (cinco dias)

vacuum_max_metadata_files_to_keep

Descrição	O número máximo de arquivos de metadados anteriores a serem retidos na ramificação principal da tabela.
Valores de proprieda de permitidos	Um número positivo.
Valor padrão	100

Exemplo de instrução CREATE TABLE

O exemplo a seguir cria uma tabela do Iceberg com três colunas.

```
CREATE TABLE iceberg_table (
  id int,
  data string,
  category string)
PARTITIONED BY (category, bucket(16,id))
LOCATION 's3://DOC-EXAMPLE-BUCKET/iceberg-folder'
TBLPROPERTIES (
  'table_type'='ICEBERG',
  'format'='parquet',
  'write_compression'='snappy',
  'optimize_rewrite_delete_file_threshold'='10'
)
```

CREATE TABLE AS SELECT (CTAS)

Para obter informações sobre a criação de uma tabela do Iceberg usando a instrução `CREATE TABLE AS`, consulte CREATE TABLE AS, com atenção especial à seção Propriedades da tabela CTAS.

Usar um crawler do AWS Glue

Você pode usar um crawler do AWS Glue para registrar automaticamente suas tabelas Iceberg no AWS Glue Data Catalog. Se quiser migrar de outro catálogo do Iceberg, você pode criar e agendar um crawler do AWS Glue e fornecer os caminhos do Amazon S3 onde as tabelas do Iceberg estão localizadas. Você pode especificar a profundidade máxima dos caminhos do Amazon S3 que o crawler do AWS Glue pode percorrer. Depois de agendar um crawler do AWS Glue, ele extrairá as informações do esquema e atualizará o AWS Glue Data Catalog com as alterações do esquema toda vez que for executado. O crawler do AWS Glue é compatível com a mesclagem de esquemas nos snapshots e atualiza o local mais recente do arquivo de metadados no AWS Glue Data Catalog. Para obter mais informações, consulte Data Catalog and crawlers in AWS Glue.

Gerenciar tabelas Iceberg

O Athena oferece suporte às operações DDL a seguir para tabelas Iceberg.

ALTER TABLE RENAME

Renomeia uma tabela.

Como os metadados de uma tabela Iceberg são armazenados no Amazon S3, você pode atualizar o banco de dados e o nome da tabela de uma tabela Iceberg gerenciada sem afetar as informações de tabela subjacentes.

Resumo

```
ALTER TABLE [db_name.]table_name RENAME TO [new_db_name.]new_table_name
```

Exemplo

```
ALTER TABLE my_db.my_table RENAME TO my_db2.my_table2
```

ALTER TABLE SET PROPERTIES

Adiciona propriedades a uma tabela Iceberg e define os valores atribuídos a elas.

De acordo com as especificações do Iceberg, as propriedades de tabela são armazenadas no arquivo de metadados da tabela Iceberg, e não em AWS Glue. O Athena não aceita propriedades de tabela personalizadas. Consulte a seção Propriedade das tabelas para ver os pares de chave-valor permitidos. Se você quiser que o Athena suporte uma propriedade específica de configuração de tabela de código aberto, envie comentários para athena-feedback@amazon.com.

Resumo

```
ALTER TABLE [db_name.]table_name SET TBLPROPERTIES ('property_name' =
 'property_value' [ , ... ])
```

Exemplo

```
ALTER TABLE iceberg_table SET TBLPROPERTIES (
  'format'='parquet',
  'write_compression'='snappy',
  'optimize_rewrite_delete_file_threshold'='10'
)
```

ALTER TABLE UNSET PROPERTIES

Descarta as propriedades existentes de uma tabela Iceberg.

Resumo

```
ALTER TABLE [db_name.]table_name UNSET TBLPROPERTIES ('property_name' [ , ... ])
```

Exemplo

```
ALTER TABLE iceberg_table UNSET TBLPROPERTIES ('write_compression')
```

DESCRIBE TABLE

Descreve informações da tabela.

Resumo

```
DESCRIBE [FORMATTED] [db_name.]table_name
```

Quando a opção `FORMATTED` é especificada, a saída exibe informações adicionais, como localização e propriedades da tabela.

Exemplo

```
DESCRIBE iceberg_table
```

DESCARTAR TABELA

Descarta uma tabela Iceberg.

> **Warning**
>
> Como as tabelas Iceberg são consideradas tabelas gerenciadas no Athena, descartar uma tabela Iceberg remove todos os dados na tabela.

Resumo

```
DROP TABLE [IF EXISTS] [db_name.]table_name
```

Exemplo

```
DROP TABLE iceberg_table
```

SHOW CREATE TABLE

Exibe uma declaração DDL `CREATE TABLE` que pode ser usada para recriar a tabela Iceberg no Athena. Se o Athena não puder reproduzir a estrutura da tabela (por exemplo, quando a tabela tiver propriedades personalizadas especificadas), um erro UNSUPPORTED (NÃO SUPORTADO) será gerado.

Resumo

```
SHOW CREATE TABLE [db_name.]table_name
```

Exemplo

```
SHOW CREATE TABLE iceberg_table
```

SHOW TABLE PROPERTIES

Mostra uma ou mais propriedades de uma tabela Iceberg. Somente as propriedades de tabela compatíveis com o Athena são mostradas.

Resumo

```
SHOW TBLPROPERTIES [db_name.]table_name [('property_name')]
```

Exemplo

```
SHOW TBLPROPERTIES iceberg_table
```

Consultar metadados de tabela do Iceberg

Em uma consulta SELECT, é possível usar as seguintes propriedades após *table_name* para consultar metadados de tabela do Iceberg:

- $files: mostra os arquivos de dados atuais de uma tabela.
- $manifests: mostra os manifestos do arquivo atual de uma tabela.
- $history: mostra o histórico de uma tabela.
- $partitions: mostra as partições atuais de uma tabela.
- $snapshots: mostra os snapshots de uma tabela.
- $refs: mostra as referências de uma tabela.

Sintaxe

A instrução a seguir lista os arquivos de uma tabela do Iceberg.

```
SELECT * FROM "dbname"."tablename$files"
```

A instrução a seguir lista os manifestos de uma tabela do Iceberg.

```
SELECT * FROM "dbname"."tablename$manifests"
```

A instrução a seguir mostra o histórico de uma tabela do Iceberg.

```
SELECT * FROM "dbname"."tablename$history"
```

O exemplo a seguir mostra as partições de uma tabela do Iceberg.

```
SELECT * FROM "dbname"."tablename$partitions"
```

O exemplo a seguir lista os snapshots de uma tabela do Iceberg.

```
SELECT * FROM "dbname"."tablename$snapshots"
```

O exemplo a seguir mostra as referências de uma tabela do Iceberg.

```
SELECT * FROM "dbname"."tablename$refs"
```

Esquema de tabela Iceberg em evolução

As atualizações de esquema do Iceberg são alterações somente de metadados. Nenhum arquivo de dados é alterado quando você executa uma atualização de esquema.

O formato Iceberg suporta as seguintes alterações na evolução do esquema:

- Adicionar: adiciona uma nova coluna a uma tabela ou a uma `struct` aninhada.
- Descartar: remove uma coluna existente de uma tabela ou `struct` aninhada.
- Renomear: renomeia uma coluna ou campo existente em uma `struct` aninhada.
- Reordenar: altera a ordem das colunas.
- Promoção de tipo: amplia o tipo de uma coluna, um campo `struct`, uma chave `map`, um valor `map` ou um elemento `list`. Atualmente, há suporte para os seguintes casos nas tabelas Iceberg:
  - inteiro para grande inteiro
  - float para double
  - aumento da precisão de um tipo decimal

ALTER TABLE ADD COLUMNS

Adiciona uma ou mais colunas a uma tabela Iceberg existente.

Resumo

```
ALTER TABLE [db_name.]table_name ADD COLUMNS (col_name data_type [,...])
```

Exemplos

O exemplo a seguir adiciona uma coluna `comment` do tipo `string` em uma tabela Iceberg.

```
ALTER TABLE iceberg_table ADD COLUMNS (comment string)
```

O exemplo a seguir adiciona uma coluna `point` do tipo `struct` em uma tabela Iceberg.

```
ALTER TABLE iceberg_table
ADD COLUMNS (point struct<x: double, y: double>)
```

O exemplo a seguir adiciona uma coluna `points`, que é uma matriz de structs , em uma tabela Iceberg.

```
ALTER TABLE iceberg_table
ADD COLUMNS (points array<struct<x: double, y: double>>)
```

ALTER TABLE DROP COLUMN

Descarta uma coluna de uma tabela Iceberg existente.

Resumo

```
ALTER TABLE [db_name.]table_name DROP COLUMN col_name
```

Exemplo

```
ALTER TABLE iceberg_table DROP COLUMN userid
```

ALTER TABLE CHANGE COLUMN

Altera o nome, o tipo, a ordem ou o comentário de uma coluna.

> **Note**
>
> Não há suporte ao `ALTER TABLE REPLACE COLUMNS`. Como `REPLACE COLUMNS` remove todas as colunas e, em seguida, adiciona colunas novas, ele não é compatível no Iceberg. `CHANGE COLUMN` é a sintaxe preferida para a evolução do esquema.

Resumo

```
ALTER TABLE [db_name.]table_name
  CHANGE [COLUMN] col_old_name col_new_name column_type
  [COMMENT col_comment] [FIRST|AFTER column_name]
```

Exemplo

```
ALTER TABLE iceberg_table CHANGE comment blog_comment string AFTER id
```

SHOW COLUMNS

Mostra as colunas em uma tabela.

Resumo

```
SHOW COLUMNS (FROM|IN) [db_name.]table_name
```

Exemplo

```
SHOW COLUMNS FROM iceberg_table
```

Consultar dados em tabelas Iceberg e realizar viagens no tempo

Para consultar um conjunto de dados Iceberg, use a instrução `SELECT` padrão, como a que se segue. As consultas atendem à especificação de formato v2 do Apache Iceberg e realizam a mesclagem na leitura das exclusões de posição e igualdade.

```
SELECT * FROM [db_name.]table_name [WHERE predicate]
```

Para otimizar os tempos de consulta, todos os predicados são enviados para onde os dados residem.

Consultas de viagem no tempo e viagem nas versões

Toda tabela Apache Iceberg mantém um manifesto versionado dos objetos do Amazon S3 que contém. Versões anteriores do manifesto podem ser usadas para consultas de viagem no tempo e de viagem nas versões.

As consultas de viagem no tempo no Athena consultam o Amazon S3 para obter dados históricos usando um snapshot consistente a partir de uma data e hora especificadas. As consultas de viagem nas versões do Athena consultam o Amazon S3 para obter dados históricos a partir de um ID de snapshot especificado.

Consultas de viagem no tempo

Para executar uma consulta de viagem no tempo, use `FOR TIMESTAMP AS OF` *`timestamp`* após o nome da tabela na instrução `SELECT`, como no exemplo a seguir.

```
SELECT * FROM iceberg_table FOR TIMESTAMP AS OF timestamp
```

O horário do sistema a ser especificado para a viagem é um carimbo de data/hora ou uma carimbo de data/hora com um fuso horário. Se não for especificado, o Athena considera o valor como um carimbo de data/hora no horário UTC.

O exemplo a seguir de consultas de viagem no tempo selecionam dados do CloudTrail para a data e hora especificadas.

```
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)
```

Consultas de viagem nas versões

Para executar uma consulta de viagem nas versões (ou seja, exibir um snapshot consistente a partir de uma versão especificada), use `FOR VERSION AS OF` *`version`* após o nome da tabela na instrução `SELECT`, como no exemplo a seguir.

```
SELECT * FROM [db_name.]table_name FOR VERSION AS OF version
```

O parâmetro *`versão`* é o ID do snapshot `bigint` associado a uma versão da tabela Iceberg.

O exemplo de consulta de viagem nas versões a seguir seleciona dados para a versão especificada.

```
SELECT * FROM iceberg_table FOR VERSION AS OF 949530903748831860
```

> **Note**
>
> As cláusulas `FOR SYSTEM_TIME AS OF` e `FOR SYSTEM_VERSION AS OF` na versão 2 do mecanismo do Athena foram substituídas pelas cláusulas `FOR TIMESTAMP AS OF` e `FOR VERSION AS OF` na versão 3 do mecanismo do Athena.

Recuperar o ID do snapshot

Você pode usar a classe Java SnapshotUtil fornecida pelo Iceberg para recuperar o ID do snapshot do Iceberg, como no exemplo a seguir.

```
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);
```

Combinar viagens no tempo e viagens nas versões

Você pode usar a sintaxe de viagem no tempo e viagem nas versões na mesma consulta para especificar condições diferentes de temporização e versionamento, como no exemplo a seguir.

```
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)
```

Criação e consulta de visualizações com tabelas do Iceberg

Para criar e consultar visualizações do Athena em tabelas do Iceberg, use visualizações `CREATE VIEW` conforme descrito em Trabalhar com visualizações.

Exemplo:

```
CREATE VIEW view1 AS SELECT * FROM iceberg_table
```

```
SELECT * FROM view1
```

Se você tiver interesse em usar a especificação de visualização do Iceberg para criar visualizações, entre em contato com athena-feedback@amazon.com.

Como trabalhar com o controle de acesso detalhado do Lake Formation

A versão 3 do mecanismo do Athena é compatível com o controle de acesso detalhado do Lake Formation com tabelas do Iceberg, incluindo o controle de acesso de segurança em nível de coluna e de linha. Esse controle de acesso funciona com consultas de passagem de tempo e com tabelas que realizaram a evolução do esquema. Para ter mais informações, consulte Controle de acesso detalhado do Lake Formation e grupos de trabalho do Athena.

Se a criação da sua tabela do Iceberg ocorreu exterior ao Athena, use o SDK do Apache Iceberg, versão 0.13.0 ou superior, para que as informações da coluna da tabela do Iceberg sejam preenchidas no AWS Glue Data Catalog. Se a tabela do Iceberg não contiver informações de coluna no AWS Glue, você poderá usar a instrução ALTER TABLE SET PROPERTIES do Athena ou o SDK do Iceberg mais recente para corrigir a tabela e atualizar as informações da coluna no AWS Glue.

Atualizar dados nas tabelas Iceberg

Os dados da tabela Iceberg podem ser gerenciados diretamente no Athena usando as consultas `INSERT`, `UPDATE` e `DELETE`. Cada transação de gerenciamento de dados produz um novo snapshot, que pode ser consultado usando a viagem no tempo. As instruções `UPDATE` e `DELETE` seguem

a especificação de exclusão de posição no nível de linha do formato v2 do Iceberg e aplicam o isolamento do snapshot.

Use os comandos a seguir para executar operações de gerenciamento de dados em tabelas Iceberg.

INSERT INTO

Insere dados em uma tabela Iceberg. O `INSERT INTO` do Athena Iceberg, da mesma forma que as consultas `INSERT INTO` atuais em tabelas externas do Hive, é cobrado pela quantidade de dados verificados. Para inserir dados em uma tabela Iceberg, use a sintaxe a seguir, na qual *`query`* pode ser `VALUES (val1, val2, ...)` ou `SELECT (col1, col2, …) FROM [`*`db_name`*`.]`*`table_name`* `WHERE` *`predicate`*. Para obter detalhes de sintaxe e semântica do SQL, consulte INSERT INTO.

```
INSERT INTO [db_name.]table_name [(col1, col2, …)] query
```

Os exemplos a seguir inserem valores na tabela `iceberg_table`.

```
INSERT INTO iceberg_table VALUES (1,'a','c1')
```

```
INSERT INTO iceberg_table (col1, col2, ...) VALUES (val1, val2, ...)
```

```
INSERT INTO iceberg_table SELECT * FROM another_table
```

DELETE

O `DELETE` do Athena Iceberg grava arquivos de exclusão da posição do Iceberg em uma tabela. Essa operação é conhecida como uma exclusão mesclar na leitura. Em contraste com uma exclusão copiar na gravação, a exclusão mesclar na leitura é mais eficiente porque não grava os dados do arquivo novamente. Ao ler dados do Iceberg, o Athena mescla os arquivos de exclusão da posição do Iceberg com arquivos de dados para produzir a visualização mais recente de uma tabela. Para remover esses arquivos de exclusão de posição, você pode executar a ação de compactação REWRITE DATA. As operações `DELETE` são cobradas pela quantidade de dados verificados. Para ver a sintaxe, consulte DELETE.

O exemplo a seguir exclui as linhas de `iceberg_table` que têm `c3` como o valor para `category`.

```
DELETE FROM iceberg_table WHERE category='c3'
```

UPDATE

O UPDATE do Athena Iceberg grava arquivos de exclusão da posição do Iceberg e linhas recém-atualizadas como arquivos de dados na mesma transação. O UPDATE pode ser considerado como uma combinação de `INSERT INTO` e `DELETE`. As operações UPDATE são cobradas pela quantidade de dados verificados. Para ver a sintaxe, consulte UPDATE.

O exemplo a seguir atualiza os valores especificados na tabela `iceberg_table`.

```
UPDATE iceberg_table SET category='c2' WHERE category='c1'
```

MERGE INTO

Atualiza, exclui ou insere linhas de forma condicional em uma tabela do Iceberg. Uma única instrução pode combinar ações de atualização, exclusão e inserção. Para ver a sintaxe, consulte MERGE INTO.

> **Note**
>
> `MERGE INTO` é transacional e é compatível somente com tabelas do Apache Iceberg na versão 3 do mecanismo do Athena.

O exemplo a seguir exclui todos os clientes da tabela `t` que estão na tabela de origem `s`.

```
MERGE INTO accounts t USING monthly_accounts_update s
ON t.customer = s.customer
WHEN MATCHED
THEN DELETE
```

O exemplo a seguir atualiza a tabela de destino `t` com as informações do cliente presentes na tabela de origem `s`. Para linhas de clientes na tabela `t` que têm linhas de clientes correspondentes na tabela `s`, o exemplo incrementa as aquisições na tabela t. Se a tabela `t` não corresponder a uma linha de cliente na tabela `s`, o exemplo irá inserir a linha de cliente da tabela `s` na tabela`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)
```

O exemplo a seguir atualiza condicionalmente a tabela de destino `t` com informações presentes na tabela de origem `s`. O exemplo exclui qualquer linha de destino correspondente cujo endereço de origem seja “Centreville”. Para todas as outras linhas correspondentes, o exemplo adiciona as aquisições da origem e define o endereço de destino como o endereço da origem. Se não houver correspondência na tabela de destino, o exemplo irá inserir a linha da tabela de origem.

```
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)
```

Otimizar tabelas Iceberg

À medida que os dados se acumulam em uma tabela Iceberg, as consultas gradualmente se tornam menos eficientes devido ao aumento do tempo de processamento necessário para abrir arquivos. Custo computacional adicional é incorrido se a tabela contiver delete files. No Iceberg, delete files armazena exclusões no nível de linha e o mecanismo deve aplicar as linhas excluídas aos resultados da consulta.

Para ajudar a otimizar a performance de consultas em tabelas Iceberg, o Athena oferece suporte à compactação manual como um comando de manutenção de tabela. As compactações otimizam o layout estrutural da tabela sem alterar o seu conteúdo.

OPTIMIZE

A ação de compactação `OPTIMIZE table REWRITE DATA` regrava os arquivos de dados em um layout mais otimizado com base no tamanho e no número de delete files associados. Para obter detalhes sobre a sintaxe e as propriedades da tabela, consulte OPTIMIZE.

Exemplo

O exemplo a seguir mescla delete files em arquivos de dados e produz arquivos próximos ao tamanho do arquivo de destino, em que o valor de `category` é `c1`.

```
OPTIMIZE iceberg_table REWRITE DATA USING BIN_PACK
  WHERE category = 'c1'
```

VACUUM

`VACUUM` executa a expiração do snapshot e a remoção de arquivos órfãos. Essas ações reduzem o tamanho dos metadados e removem os arquivos que não estão no estado atual da tabela e que também são mais antigos do que o período de retenção especificado para a tabela. Para obter detalhes sobre a sintaxe, consulte VACUUM.

Exemplo

O exemplo a seguir usa uma propriedade de tabela para configurar a tabela `iceberg_table` para reter os últimos três dias de dados e, em seguida, usa `VACUUM` para expirar os snapshots antigos e remover os arquivos órfãos da tabela.

```
ALTER TABLE iceberg_table SET TBLPROPERTIES (
  'vacuum_max_snapshot_age_seconds'='259200'
)

VACUUM iceberg_table
```

Tipos de dados suportados para tabelas Iceberg no Athena

O Athena pode consultar tabelas Iceberg que contêm os seguintes tipos de dados:

```
binary
boolean
date
decimal
double
float
int
list
long
map
string
struct
timestamp without time zone
```

Para obter mais informações sobre tipos de tabela Iceberg, consulte a página de esquemas para Iceberg na documentação do Apache.

A tabela a seguir mostra a relação entre tipos de dados do Athena e tipos de dados de tabela Iceberg.

Tipo do Iceberg	Tipo do Athena	Observações
`boolean`	`boolean`	
-	`tinyint`	Não suportado para tabelas Iceberg no Athena.
-	`smallint`	Não suportado para tabelas Iceberg no Athena.
`int`	`int`	Nas instruções DML do Athena, esse tipo é `INTEGER`.
`long`	`bigint`	
`double`	`double`	
`float`	`float`	
`decimal(P, S)`	`decimal(P, S)`	`P` é precisão, `S` é escala.
-	`char`	Não suportado para tabelas Iceberg no Athena.
`string`	`string`	Nas instruções DML do Athena, esse tipo é `VARCHAR`.
`binary`	`binary`	
`date`	`date`	
`time`	-	Somente o carimbo de data/hora do Iceberg (sem fuso horário) é suportado para instruções DDL do Athena Iceberg, como `CREATE TABLE`, mas todos os tipos de carimbo de data/hora podem ser consultados por meio do Athena.
`timestamp`	`timestamp`	
`timestamptz`	`timestamptz`	

Tipo do Iceberg	Tipo do Athena	Observações
`list<E>`	`array`	
`map<K,V>`	`map`	
`struct<...>`	`struct`	
`fixed(L)`	-	O tipo `fixed(L)` não é suportado no Athena atualmente.

Para obter mais informações sobre tipos de dados no Athena, consulte Tipos de dados no Amazon Athena.

Outras operações do Athena em tabelas Iceberg

Operações no banco de dados

Quando você usa DROP DATABASE com a opção `CASCADE`, todos os dados da tabela Iceberg também são removidos. As operações DDL a seguir não têm efeito nas tabelas Iceberg.

- CREATE DATABASE
- ALTER DATABASE SET DBPROPERTIES
- SHOW DATABASES
- SHOW TABLES
- SHOW VIEWS

Operações relacionadas à partição

Como as tabelas Iceberg usam o particionamento oculto, você não precisa trabalhar diretamente com partições físicas. Por isso, as tabelas Iceberg no Athena não são compatíveis com as seguintes operações DDL relacionadas à partição:

- SHOW PARTITIONS
- ALTER TABLE ADD PARTITION
- ALTER TABLE DROP PARTITION

- ALTER TABLE RENAME PARTITION

Se quiser ver a evolução da partição no Athena, envie comentários para athena-feedback@amazon.com.

Descarregar tabelas Iceberg

As tabelas Iceberg podem ser descarregadas em arquivos em uma pasta no Amazon S3. Para ter mais informações, consulte UNLOAD.

MSCK REPAIR

Como as tabelas Iceberg acompanham as informações de layout da tabela, executar MSCK REPAIR TABLE, como se faz com tabelas do Hive, não é necessário nem compatível.

Recursos adicionais do

Para artigos detalhados sobre como usar o Athena com tabelas do Apache Iceberg, consulte as seguintes postagens no AWSBlog de Big Data.

- Acelere a engenharia de atributos de ciência de dados em lagos de dados transacionais usando o Amazon Athena com o Apache Iceberg
- Crie um data lake no Apache Iceberg usando o Amazon Athena, o Amazon EMR e AWS Glue
- Execute inserções em um data lake usando o Amazon Athena e o Apache Iceberg
- Crie um data lake transacional usando o Apache Iceberg, AWS Glue e compartilhamentos de dados entre contas usando o AWS Lake Formation e o Amazon Athena
- Use o Apache Iceberg em um data lake para oferecer suporte ao processamento incremental de dados
- Crie um data lake Apache Iceberg em tempo real alinhado ao GDPR
- Automatize a replicação de fontes relacionais em um data lake transacional com o Apache Iceberg e AWS Glue
- Interaja com tabelas do Apache Iceberg usando o Amazon Athena e permissões refinadas entre contas usando AWS Lake Formation
- Crie um data lake transacional na tecnologia sem servidor com o Apache Iceberg, o Amazon EMR Sem Servidor e o Amazon Athena

Segurança do Amazon Athena

A segurança na nuvem na AWS é a nossa maior prioridade. Como cliente da AWS, você contará com um datacenter e uma arquitetura de rede criados para atender aos requisitos das organizações com as maiores exigências de segurança.

A segurança é uma responsabilidade compartilhada entre a AWS e você. O modelo de responsabilidade compartilhada descreve isto como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem: a AWS é responsável pela proteção da infraestrutura que executa Serviços da AWS na Nuvem AWS. A AWS também fornece serviços que podem ser usados com segurança. A eficácia da nossa segurança é regularmente testada e verificada por auditores de terceiros como parte dos Programas de conformidade da AWS. Para saber mais sobre os programas de compatibilidade aplicáveis ao Athena, consulte Serviços da AWS no escopo pelo programa de compatibilidade.
- Segurança na nuvem: sua responsabilidade é determinada pelo AWS service (Serviço da AWS) que você usa. Você também é responsável por outros fatores, inclusive a confidencialidade dos dados, os requisitos da organização, as leis e as regulamentações vigentes.

Essa documentação ajudará você a entender como aplicar o modelo de responsabilidade compartilhada ao usar o Amazon Athena. Os tópicos a seguir mostram como configurar o Athena para atender aos seus objetivos de segurança e compatibilidade. Você também aprenderá a usar outros Serviços da AWS que podem ajudá-lo a monitorar e proteger seus recursos do Athena.

Tópicos

- Proteção de dados no Athena
- Gerenciamento de identidade e acesso no Athena
- Registro e monitoramento no Athena
- Validação de compatibilidade do Amazon Athena
- Resiliência no Athena
- Segurança da infraestrutura no Athena
- Análise de vulnerabilidade e configuração no Athena
- Usar o Athena para consultar dados registrados com o AWS Lake Formation

Proteção de dados no Athena

O AWS modelo de responsabilidade compartilhada é aplicado à proteção de dados no Amazon Athena. Conforme descrito nesse modelo, a AWS é responsável por proteger a infraestrutura global que executa toda a Nuvem AWS. Você é responsável por manter o controle sobre seu conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para ter mais informações sobre a privacidade de dados, consulte as Perguntas frequentes sobre privacidade de dados. Para ter mais informações sobre a proteção de dados na Europa, consulte a AWSpostagem do blog Shared Responsibility Model and GDPR no AWSBlog de segurança da.

Para fins de proteção de dados, recomendamos que você proteja asConta da AWS credenciais da e configure as contas de usuário individuais com o AWS IAM Identity Center ou o AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com os atributos da AWS. Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure o registro em log das atividades da API e do usuário com o .AWS CloudTrail
- Use AWS as soluções de criptografia da , juntamente com todos os controles de segurança padrão dos Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sigilosos armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-2 ao acessar a AWS por meio de uma interface de linha de comandos ou uma API, use um endpoint do FIPS. Para ter mais informações sobre endpoints do FIPS, consulte Federal Information Processing Standard (FIPS) 140-2.

É altamente recomendável que nunca sejam colocadas informações de identificação confidenciais, como endereços de email dos seus clientes, em marcações ou campos de formato livre, como um campo Name (Nome). Isso inclui o trabalho com o Athena ou outros Serviços da AWS por meio do console, da API, da AWS CLI ou dos AWS SDKs. Quaisquer dados inseridos em tags ou campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de

diagnóstico. Se você fornecer um URL para um servidor externo, recomendemos fortemente que não sejam incluídas informações de credenciais no URL para validar a solicitação a esse servidor.

Como uma etapa de segurança adicional, é possível usar chave de contexto da condição global aws:CalledVia para limitar as solicitações somente àquelas feitas pelo Athena. Para ter mais informações, consulte Usar o Athena com chaves de contexto CalledVia.

Proteger vários tipos de dados

Vários tipos de dados estão envolvidos ao usar o Athena para criar bancos de dados e tabelas. Eles incluem dados de origem armazenados na origem no Amazon S3, metadados de bancos de dados e tabelas que você cria ao executar consultas ou o crawler do AWS Glue para descobrir dados, dados de resultados de consultas e histórico de consultas. Esta seção discute cada tipo de dado e fornece orientação sobre a proteção.

- Dados de origem: você armazena os dados de bancos de dados e tabelas no Amazon S3, e o Athena não os modifica. Para obter mais informações, consulte Proteção de dados no Amazon S3 no Guia do usuário do Amazon Simple Storage Service. Você controla o acesso aos seus dados de origem e pode criptografá-los no Amazon S3. Você pode usar o Athena para criar tabelas com base em conjuntos de dados criptografados no Amazon S3.
- Metadados de tabelas e bancos de dados (esquema): o Athena usa a tecnologia schema-on-read (esquema na leitura), o que significa que as definições de tabela são aplicadas aos dados no Amazon S3 quando o Athena executa consultas. Todos os esquemas que você definir serão automaticamente salvos, a não ser que você os exclua explicitamente. No Athena, você pode modificar os metadados do catálogo de dados usando instruções DDL. Você também pode excluir as definições de tabela e o esquema sem afetar os dados subjacentes armazenados no Amazon S3. Os metadados dos bancos de dados e das tabelas que você usa no Athena são armazenados no AWS Glue Data Catalog.

  Você pode definir políticas de acesso granulares para bancos de dados e tabelas registrados no AWS Glue Data Catalog usando o AWS Identity and Access Management (IAM). Você também pode criptografar metadados no AWS Glue Data Catalog. Se você criptografar os metadados, use permissões para metadados criptografados para o acesso.
- Resultados e histórico de consultas, incluindo consultas salvas: os resultados das consultas são armazenados em um local no Amazon S3 que pode ser especificado globalmente ou para cada grupo de trabalho. Se não for especificado, o Athena usará o local padrão em cada caso. Você controla o acesso aos buckets do Amazon S3 nos quais armazena os resultados das consultas e as consultas salvas. Você também pode criptografar os resultados da consulta armazenados no

Amazon S3. Seus usuários devem ter as devidas permissões para acessar os locais do Amazon S3 e descriptografar os arquivos. Para obter mais informações, consulte Criptografar resultados das consultas do Athena armazenados no Amazon S3 neste documento.

O Athena mantém o histórico de consultas por 45 dias. Você pode visualizar o histórico de consultas usando as APIs do Athena, o console e a AWS CLI. Para armazenar as consultas para mais de 45 dias, salve-as. Para proteger o acesso às consultas salvas, use os grupos de trabalho no Athena, restringindo o acesso às consultas salvas somente para os usuários autorizados a visualizá-las.

Tópicos

- Criptografia inativa
- Criptografia em trânsito
- Gerenciamento de chaves
- Privacidade do tráfego entre redes

Criptografia inativa

Você pode executar consultas no Amazon Athena em dados criptografados no Amazon S3 na mesma região e em um número limitado de regiões. Você também pode criptografar os resultados da consulta no Amazon S3 e os dados no Catálogo de dados do AWS Glue.

Você pode criptografar os seguintes ativos no Athena:

- Os resultados de todas as consultas no Amazon S3, que o Athena armazena em um local conhecido como local de resultados do Amazon S3. Você pode criptografar os resultados das consultas armazenados no Amazon S3 sem considerar se o conjunto de dados subjacente está ou não criptografado no Amazon S3. Para ter mais informações, consulte Criptografar resultados das consultas do Athena armazenados no Amazon S3.
- Os dados no catálogo de dados do AWS Glue. Para ter mais informações, consulte Permissões para metadados criptografados no catálogo de dados do AWS Glue.

> **Note**
>
> Quando você usa o Athena para ler uma tabela criptografada, o Athena usa as opções de criptografia especificadas para os dados da tabela, não a opção de criptografia para os

resultados da consulta. Se métodos ou chaves de criptografia diferentes forem configurados para resultados da consulta e dados da tabela, o Athena lê os dados da tabela sem usar a opção de criptografia e a chave usada para criptografar ou descriptografar os resultados da consulta.
No entanto, se você usar o Athena para inserir dados em uma tabela que tenha dados criptografados, o Athena usa a configuração de criptografia especificada para os resultados da consulta para criptografar os dados inseridos. Por exemplo, se você especificar a criptografia `CSE_KMS` para os resultados da consulta, o Athena usa o mesmo ID de chave do AWS KMS que você usou para a criptografia dos resultados da consulta para criptografar os dados da tabela inseridos com o `CSE_KMS`.

Tópicos

- Opções de criptografia permitidas do Amazon S3
- Permissões para dados criptografados no Amazon S3
- Permissões para metadados criptografados no catálogo de dados do AWS Glue
- Criptografar resultados das consultas do Athena armazenados no Amazon S3
- Criar tabelas com base em conjuntos de dados criptografados no Amazon S3

Opções de criptografia permitidas do Amazon S3

O Athena permite as opções de criptografia a seguir para conjuntos de dados e resultados de consulta no Amazon S3.

Tipo de criptografia	Descrição	Suporte entre regiões
SSE-S3	Server Side Encryption (SSE – Criptografia do lado do servidor) com uma chave gerenciada pelo Amazon S3.	Sim
SSE-KMS	Criptografia no lado do servidor (SSE) com uma chave gerenciada pelo cliente do AWS Key Management Service.	Sim

Tipo de criptografia	Descrição	Suporte entre regiões
	Note Com esse tipo de criptografia, o Athena não requer que você indique que os dados sejam criptografados ao criar uma tabela.	
CSE-KMS	Client-Side Encryption (CSE – Criptografia do lado do cliente) com uma chave gerenciada pelo cliente AWS KMS. No Athena, essa opção requer que você use uma instrução `CREATE TABLE` com a cláusula `TBLPROPERTIES` que especifica `'has_encrypted_data'='true'`. Para ter mais informações, consulte Criar tabelas com base em conjuntos de dados criptografados no Amazon S3.	Não

Para obter mais informações sobre a criptografia do AWS KMS com o Amazon S3, consulte O que é o AWS Key Management Service e Como o Amazon Simple Storage Service (Amazon S3) usa o AWS KMS no Guia do desenvolvedor do AWS Key Management Service. Para obter mais informações sobre como usar o SSE-KMS ou o CSE-KMS com o Athena, consulte Launch: Amazon Athena adds support for querying encrypted data (Lançamento: o Amazon Athena adiciona suporte à consulta de dados criptografados) no blog sobre big data da AWS.

Opções não compatíveis

As seguintes opções de criptografia não são compatíveis:

- SSE com chaves fornecidas pelo cliente (SSE-C).
- Criptografia do lado do cliente usando uma chave mestra do lado do cliente.
- Chaves assimétricas.

Para comparar as opções de criptografia do Amazon S3, consulte Proteção de dados usando criptografia no Guia do usuário do Amazon Simple Storage Service.

Ferramentas para criptografia do lado do cliente

Para criptografia do lado do cliente, observe que há duas ferramentas disponíveis:

- Cliente de criptografia do Amazon S3: criptografa os dados somente para o Amazon S3 e é compatível com o Athena.
- AWS Encryption SDK: o SDK pode ser usado para criptografar os dados em qualquer lugar na AWS, mas não é permitido diretamente no Athena.

Essas ferramentas não são compatíveis, e os dados criptografados usando uma ferramenta não podem ser descriptografados pela outra. O Athena apenas é compatível diretamente com o Cliente de criptografia do Amazon S3. Se você usar o SDK para criptografar seus dados, poderá executar consultas do Athena, mas os dados serão retornados como texto criptografado.

Para usar o Athena para consultar dados que foram criptografados com o AWS Encryption SDK, você deve baixar e descriptografar seus dados e, depois, criptografá-los novamente usando o Cliente de criptografia do Amazon S3.

Permissões para dados criptografados no Amazon S3

Dependendo do tipo de criptografia que você usa no Amazon S3, pode ser necessário adicionar permissões, também conhecidas como ações "Permitir", a suas políticas usadas no Athena:

- SSE-S3: se você usar o SSE-S3 para criptografia, os usuários do Athena não exigirão permissões adicionais nas políticas. Basta ter as devidas permissões do Amazon S3 para o local do Amazon S3 apropriado e para as ações do Athena. Para obter mais informações sobre as políticas que concedem permissões adequadas do Athena e do Amazon S3, consulte Politicas gerenciadas pela AWS para o Amazon Athena e Acesso ao Amazon S3.
- AWS KMS: se você usa o AWS KMS para criptografia, os usuários do Athena devem receber permissão para executar determinadas ações do AWS KMS, além das permissões do Athena e do Amazon S3. Você permite essas ações editando a política de chaves das CMKs gerenciadas pelo cliente do AWS KMS usadas para criptografar os dados no Amazon S3. Para adicionar usuários de chaves às políticas do AWS KMS apropriadas, você pode usar o console do AWS KMS em https://console.aws.amazon.com/kms. Para obter informações sobre como adicionar um usuário a uma política de chave do AWS KMS, consulte Permitir que os usuários de chaves usem a CMK no Guia do desenvolvedor do AWS Key Management Service.

> **Note**
>
> Os administradores de políticas de chaves avançadas podem ajustar as políticas de chaves. `kms:Decrypt` é a ação mínima permitida para que um usuário do Athena trabalhe com um conjunto de dados criptografados. Para trabalhar com resultados da consulta criptografados, as ações permitidas mínimas são `kms:GenerateDataKey` e `kms:Decrypt`.

Ao usar o Athena para consultar conjuntos de dados no Amazon S3 com um grande número de objetos criptografados com o AWS KMS, o AWS KMS pode controlar a utilização dos resultados da consulta. Isso é mais provável quando há um grande número de objetos pequenos. O Athena rejeita solicitações de nova tentativa, mas um erro de controle de utilização ainda pode ocorrer. Se estiver trabalhando com vários objetos criptografados e enfrentar esse problema, uma opção é habilitar chaves de bucket do Amazon S3 para reduzir o número de chamadas ao KMS. Para obter mais informações, consulte Redução do custo do SSE-KMS com chaves de bucket do Amazon S3 no Guia do usuário do Amazon Simple Storage Service. Outra opção é aumentar suas cotas de serviço para o AWS KMS. Para obter mais informações, consulte Cotas no Guia do desenvolvedor do AWS Key Management Service.

Para obter informações sobre solução de problemas de permissões ao usar o Amazon S3 com o Athena, consulte a seção Permissões do tópico Solução de problemas no Athena.

Permissões para metadados criptografados no catálogo de dados do AWS Glue

Se você criptografar metadados no AWS Glue Data Catalog, deverá adicionar as ações `"kms:GenerateDataKey"`, `"kms:Decrypt"` e `"kms:Encrypt"` às políticas que usa para acessar o Athena. Para ter mais informações, consulte Acesso a metadados criptografados do Athena no AWS Glue Data Catalog.

Criptografar resultados das consultas do Athena armazenados no Amazon S3

Você pode configurar a criptografia de resultados de consultas usando o console do Athena ou o JDBC ou ODBC. Grupos de trabalhos permitem que você reforce a criptografia dos resultados de consultas.

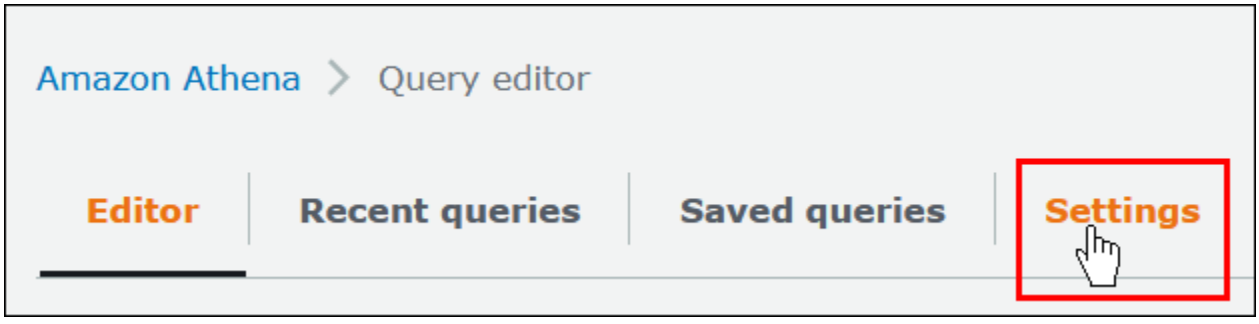
No console, você pode definir a configuração para criptografia dos resultados das consultas de duas formas:

- Configurações do lado do cliente: quando você usa Settings (Configurações) no console ou as operações de API para indicar que deseja criptografar os resultados das consultas, esse procedimento é conhecido como “usar as configurações do lado do cliente”. As configurações no lado do cliente incluem o local dos resultados da consulta e a criptografia. Se você especificá-los, elas serão usadas, a menos que sejam substituídas por configurações do grupo de trabalho.
- Configurações do grupo de trabalho: quando você cria ou edita um grupo de trabalho e seleciona o campo Override client-side settings (Substituir configurações do lado do cliente), todas as consultas executadas nesse grupo de trabalho usam as configurações de local de resultados da consulta do grupo. Para ter mais informações, consulte Configurações do grupo de trabalho sobrepõem as configurações do lado do cliente.

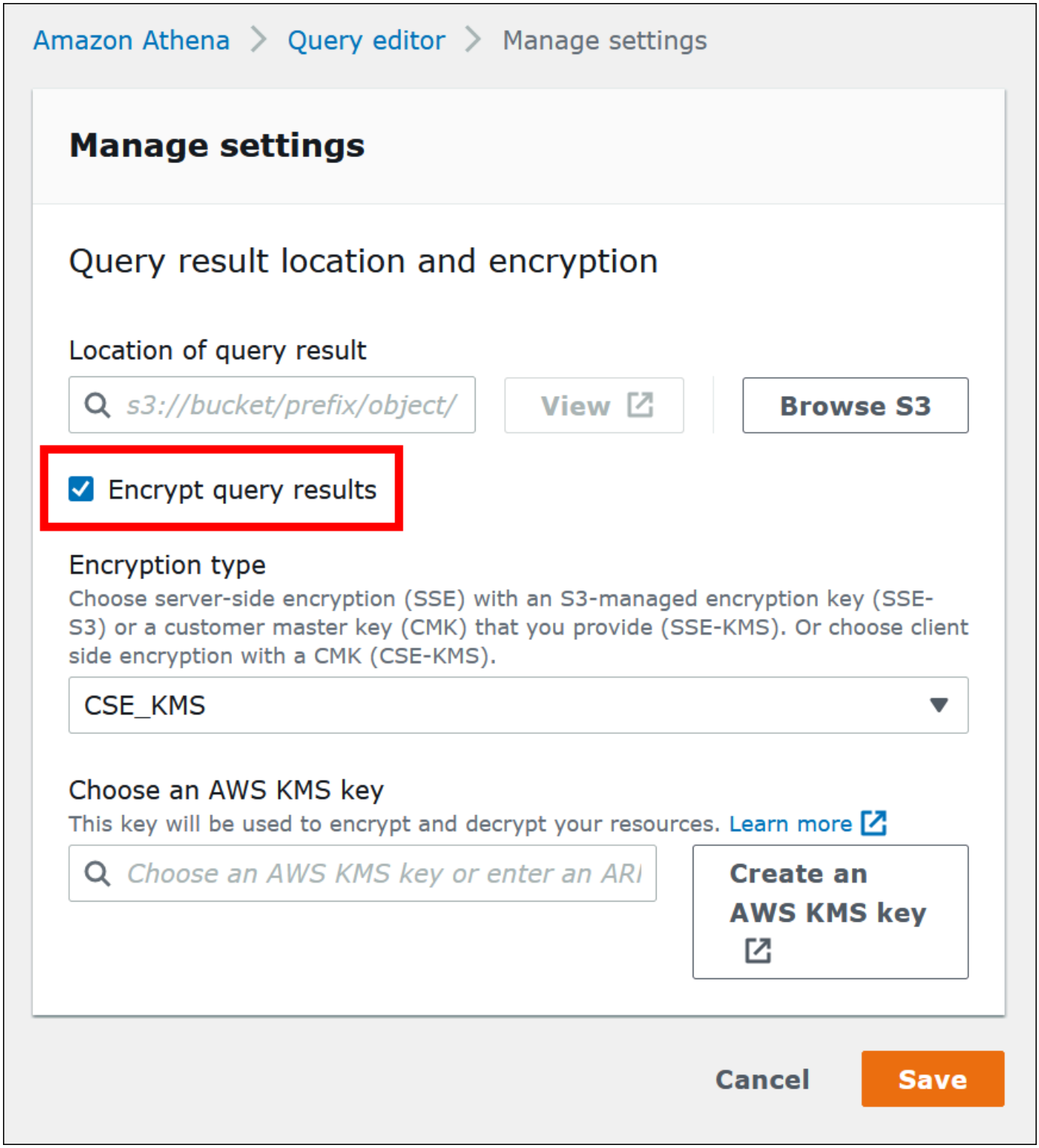
Para criptografar os resultados das consultas armazenados no Amazon S3 usando o console

> **Important**
>
> Se o seu grupo de trabalho estiver com o campo Override client-side settings (Sobrepor configurações no lado do cliente), todas as consultas no grupo de trabalho usarão as configurações do grupo de trabalho. A configuração de criptografia e o local dos resultados da consulta especificados na guia Settings (Configurações) no console do Athena pelas operações de API e pelos drivers JDBC e ODBC não são usados. Para ter mais informações, consulte Configurações do grupo de trabalho sobrepõem as configurações do lado do cliente.

1. No console do Athena, escolha Settings (Configurações).



2. Escolha Gerenciar.
3. Em Location of query result (Local do resultado da consulta), insira ou escolha um caminho do Amazon S3. Esse é o local do Amazon S3 em que os resultados da consulta são armazenados.
4. Escolha Encrypt query results.



5. Para Encryption type, escolha CSE-KMS, SSE-KMS ou SSE-S3. Desses três, o CSE-KMS oferece o nível mais alto de criptografia e o SSE-S3 o mais baixo.
6. Se você escolher SSE-KMS ou CSE-KMS, especifique a chave do AWS KMS.

- Em Escolher uma chave do AWS KMS, se sua conta tiver acesso a uma chave gerenciada pelo cliente (CMK) existente do AWS KMS, escolha o respectivo alias ou insira um ARN de chave do AWS KMS.
- Se sua conta não tiver acesso a uma chave gerenciada pelo cliente (CMK) existente, escolha Criar uma chave do AWS KMS e abra o console do AWS KMS. Para obter mais informações, consulte Criação de chaves Guia do desenvolvedor do AWS Key Management Service.

> **Note**
>
> O Athena permite apenas chaves simétricas para leitura e gravação de dados.

7. Volte para o console do Athena e escolha a chave que você criou por alias ou ARN.
8. Escolha Salvar.

Criptografar os resultados das consultas do Athena ao usar JDBC ou ODBC

Se você se conectar usando o driver JDBC ou ODBC, configure as opções do driver para especificar o tipo de criptografia que será usado e o local do diretório de preparação do Amazon S3. Para configurar o driver JDBC ou ODBC para criptografar os resultados das consultas usando qualquer um dos protocolos de criptografia compatíveis com o Athena, consulte Conectar-se ao Amazon Athena com drivers ODBC e JDBC.

Criar tabelas com base em conjuntos de dados criptografados no Amazon S3

Ao criar uma tabela, indique para o Athena que um conjunto de dados está criptografado no Amazon S3. Isso não é necessário ao usar o SSE-KMS. Para a criptografia do SSE-S3 e do AWS KMS, o Athena determina como descriptografar o conjunto de dados e criar a tabela, portanto, você não precisa inserir informações de chaves.

Os usuários que executam consultas, inclusive o usuário que cria a tabela, devem ter as permissões descritas anteriormente neste tópico.

> **Important**
>
> Se você usa o Amazon EMR com o EMRFS para carregar arquivos Parquet criptografados, deve desabilitar os carregamentos fracionados definindo `fs.s3n.multipart.uploads.enabled` como `false`. Se você não fizer

isso, o Athena não poderá determinar o tamanho do arquivo Parquet, e um erro HIVE_CANNOT_OPEN_SPLIT será emitido. Para obter mais informações, consulte Configure multipart upload for Amazon S3 (Configurar o carregamento fracionado no Amazon S3) no Guia de gerenciamento do Amazon EMR.

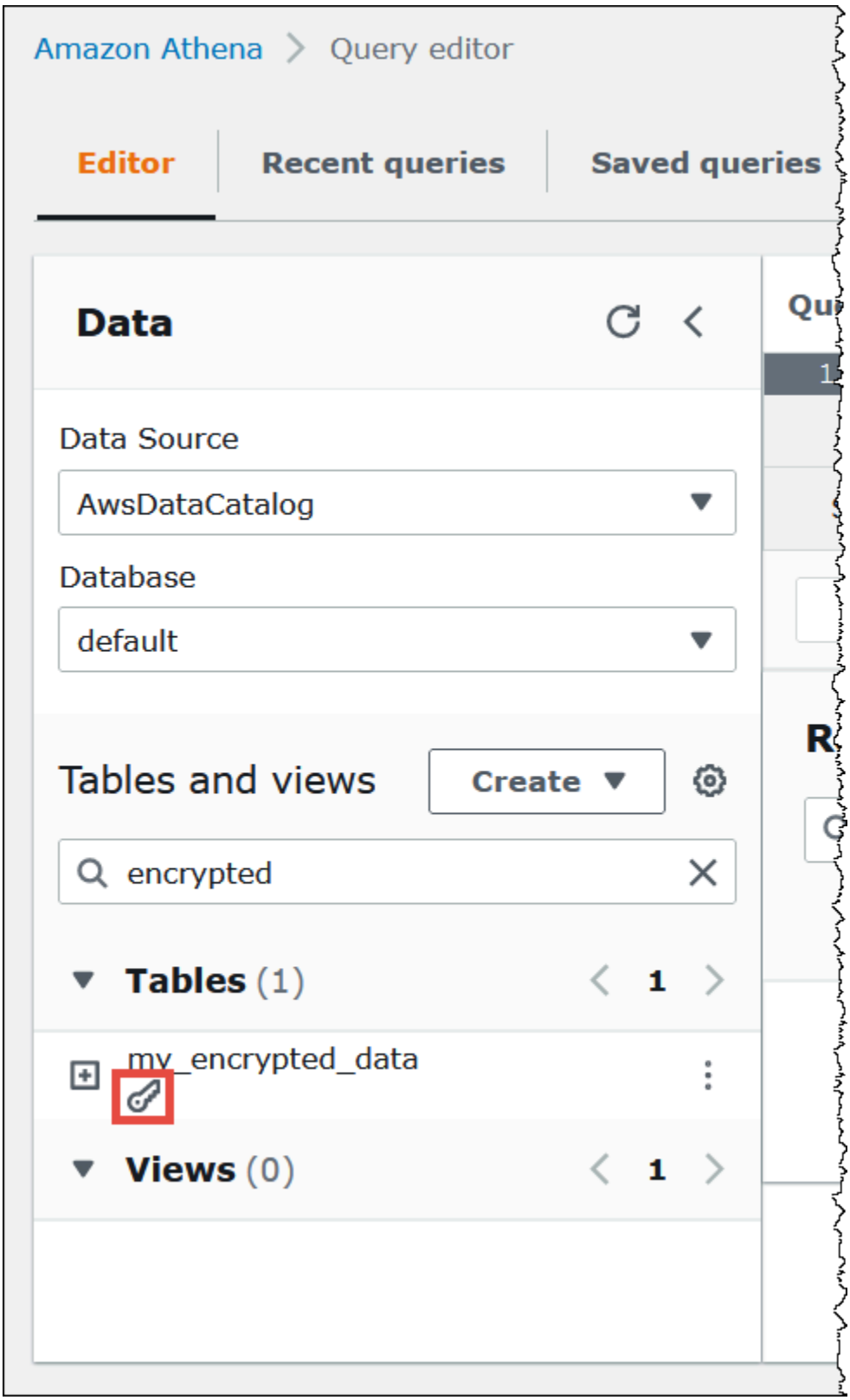
Para indicar que o conjunto de dados está criptografado no Amazon S3, execute uma das etapas a seguir. Essa etapa não será necessária, se o SSE-KMS for usado.

- Em uma instrução CREATE TABLE, use a cláusula `TBLPROPERTIES` que especifica `'has_encrypted_data'='true'`, conforme mostrado no exemplo a seguir.

```
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')
```

- Use o driver JDBC e defina o valor `TBLPROPERTIES`, conforme mostrado no exemplo anterior, ao usar `statement.executeQuery()` para executar a instrução CREATE TABLE.
- Ao usar o console do Athena para criar uma tabela usando um formulário e especificar o local da tabela, selecione a opção Encrypted data set (Conjunto de dados criptografado).

Na lista de tabelas do console do Athena, as tabelas criptografadas exibem um ícone em forma de chave.


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

Criptografia em trânsito

Além de criptografar os dados em repouso no Amazon S3, o Amazon Athena usa a criptografia Transport Layer Security (TLS) para os dados em trânsito entre o Athena e o Amazon S3, e entre o Athena e as aplicações dos clientes que o acessam.

Você deve permitir apenas conexões criptografadas por HTTPS (TLS) usando `aws:SecureTransport condition` nas políticas do IAM no bucket do Amazon S3.

Os resultados de consultas que transmitem para clientes JDBC ou ODBC são criptografados usando TLS. Para obter informações sobre as versões mais recentes dos drivers JDBC e ODBC e as respectivas documentações, consulte Conectar-se ao Amazon Athena com JDBC e Conectar-se ao Amazon Athena com ODBC.

Para conectores de fonte de dados federada do Athena, a compatibilidade com criptografia em trânsito usando TLS depende do conector individual. Para obter informações, consulte a documentação dos conectores de fonte de dados individuais.

Gerenciamento de chaves

O Amazon Athena é compatível com o AWS Key Management Service (AWS KMS) para criptografar bancos de dados no Amazon S3 e resultados das consultas do Athena. O AWS KMS usa as chaves gerenciadas pelo cliente (CMKs) para criptografar os objetos do Amazon S3 e aplica a criptografia de envelope.

No AWS KMS, você pode realizar as seguintes ações:

- Criar chaves
- Importar seu próprio material de chaves para novas CMKs

> **Note**
>
> O Athena permite apenas chaves simétricas para leitura e gravação de dados.

Para obter mais informações, consulte O que é o AWS Key Management Service no Guia do desenvolvedor do AWS Key Management Service e Como o Amazon Simple Storage Service usa o AWS KMS. Para visualizar as chaves na sua conta que a AWS cria e gerencia para você, no painel de navegação, escolha AWS managed keys (Chaves gerenciadas pela AWS).

Se você fizer upload ou acessar objetos criptografados por SSE-KMS, use o Signature versão 4 da AWS para maior segurança. Para obter mais informações, consulte Especificar a versão da assinatura na autenticação de solicitações no Guia do usuário do Amazon Simple Storage Service.

Se suas workloads do Athena criptografam uma grande quantidade de dados, você poderá usar as chaves de bucket do Amazon S3 para reduzir custos. Para obter mais informações, consulte Redução do custo do SSE-KMS com chaves de bucket do Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

Privacidade do tráfego entre redes

O tráfego é protegido entre o Athena e as aplicações on-premises e entre o Athena e o Amazon S3. Por padrão, o tráfego entre o Athena e outros serviços, como AWS Glue e AWS Key Management Service, usa HTTPS.

- Para o tráfego entre o Athena e os clientes e as aplicações on-premises, os resultados das consultas transmitidos para os clientes JDBC ou ODBC são criptografados usando Transport Layer Security (TLS).

  Você pode usar uma das opções de conectividade entre sua rede privada e a AWS:

  - Uma conexão AWS VPN Site-to-Site VPN. Para obter mais informações, consulte O que é o AWS VPN Site-to-Site VPN? no Manual do usuário do AWS Site-to-Site VPN.
  - Uma conexão do AWS Direct Connect. Para obter mais informações, consulte O que é o AWS Direct Connect no Guia do usuário do AWS Direct Connect.
- Para o tráfego entre o Athena e os buckets do Amazon S3, o Transport Layer Security (TLS) criptografa os objetos em trânsito entre o Athena e o Amazon S3, e entre o Athena e as aplicações dos clientes que o acessam. Você deve permitir somente conexões por HTTPS (TLS) usando `aws:SecureTransport condition` nas políticas do IAM de bucket do Amazon S3. Embora o Athena atualmente use o endpoint público para acessar dados em buckets do Amazon S3, isso não significa que os dados atravessem a Internet pública. Todo o tráfego entre o Athena e o Amazon S3 é roteado pelo AWS e é criptografado com TLS.
- Programas de conformidade: o Amazon Athena está em conformidade com vários programas de conformidade da AWS, incluindo SOC, PCI, FedRAMP e outros. Para obter mais informações, consulte Serviços da AWS no escopo por programa de conformidade.

Gerenciamento de identidade e acesso no Athena

O Amazon Athena usa as políticas do AWS Identity and Access Management (IAM) para restringir o acesso às operações do Athena. Para ver a lista completa de permissões do Athena, consulte Ações, recursos e chaves de condição do Amazon Athena na Referência de autorização do serviço.

Sempre que você usar as políticas do IAM, siga as práticas recomendadas do IAM. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM no Manual do usuário do IAM.

Veja abaixo as permissões necessárias para executar as consultas do Athena:

- Locais do Amazon S3 onde os dados subjacentes para consulta estão armazenados. Para obter mais informações, consulte Gerenciamento de identidade e acesso no Amazon S3 no Guia do usuário do Amazon Simple Storage Service.
- Os metadados e recursos armazenados no AWS Glue Data Catalog, como bancos de dados e tabelas, incluindo ações adicionais para metadados criptografados. Para obter mais informações, consulte Configurar permissões do IAM para o AWS Glue e Configurar a criptografia no AWS Glue no Guia do desenvolvedor do AWS Glue.
- Ações de API do Athena. Para obter uma lista das ações de API no Athena, consulte Ações na Referência de API do Amazon Athena.

Os tópicos a seguir apresentam mais informações sobre permissões para áreas específicas do Athena.

Tópicos

- Politicas gerenciadas pela AWS para o Amazon Athena
- Acesso por meio de conexões JDBC e ODBC
- Acesso ao Amazon S3
- Acesso entre contas no Athena aos buckets do Amazon S3
- Acesso granular a bancos de dados e tabelas no AWS Glue Data Catalog
- Acesso aos catálogos de dados do AWS Glue entre contas
- Acesso a metadados criptografados do Athena no AWS Glue Data Catalog
- Acesso a grupos de trabalho e etiquetas
- Permitir acesso a instruções preparadas
- Usar o Athena com chaves de contexto CalledVia

- Permitir acesso a um conector de dados do Athena para metastore externo do Hive
- Permitir acesso da função do Lambda aos metastores externos do Hive
- Exemplo de políticas de permissões do IAM para permitir consulta federada do Athena
- Exemplo de políticas de permissões do IAM para permitir funções definidas pelo usuário (UDF) do Amazon Athena
- Permitir acesso para ML com o Athena
- Permitir acesso federado à API do Athena

Politicas gerenciadas pela AWS para o Amazon Athena

Uma política gerenciada pela AWS é uma política independente criada e administrada pela AWS. As políticas gerenciadas pela AWS são criadas para fornecer permissões a vários casos de uso comuns a fim de que você possa começar a atribuir permissões a usuários, grupos e perfis.

Lembre-se de que as políticas gerenciadas da AWS podem não conceder permissões de menor privilégio para seus casos de uso específicos, pois elas estão disponíveis para todos os clientes da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo cliente específicas para seus casos de uso.

Você não pode alterar as permissões definidas em políticas gerenciadas pela AWS. Se a AWS atualiza as permissões definidas em um política gerenciada pela AWS, a atualização afeta todas as identidades de entidades principais (usuários, grupos e perfis) às quais a política está vinculada. É mais provável que a AWS atualize uma política gerenciada pela AWS quando um novo AWS service (Serviço da AWS) é lançado ou novas operações de API são disponibilizadas para os serviços existentes.

Para mais informações, consulte Políticas gerenciadas pela AWS no Manual do usuário do IAM.

Considerações ao usar políticas gerenciadas com o Athena

As políticas gerenciadas são fáceis de usar e são atualizadas automaticamente com as ações necessárias à medida que o serviço evolui. Ao usar políticas gerenciadas com o Athena, lembre-se do seguinte:

- Para permitir ou negar ações de serviço do Amazon Athena para você mesmo ou outros usuários usando o AWS Identity and Access Management (IAM), anexe as políticas baseadas em identidade aos principais, como usuários ou grupos.

- Cada política baseada em identidade consiste em instruções que definem as ações permitidas ou negadas. Para obter mais informações e instruções detalhadas sobre como anexar uma política a um usuário, consulte Anexar políticas gerenciadas no Guia do usuário do IAM. Para obter uma lista de ações, consulte a Referência de API do Amazon Athena.
- As políticas gerenciadas pelo cliente e baseadas em identidade em linha permitem especificar ações mais detalhadas do Athena em uma política para adaptar o acesso. Recomendamos usar a política `AmazonAthenaFullAccess` como um ponto de partida e permitir ou negar ações específicas listadas na Referência de API do Amazon Athena. Para obter mais informações sobre políticas em linha, consulte Políticas gerenciadas e em linha no Guia do usuário do IAM.
- Se você também tiver entidades principais que se conectam usando JDBC, deverá fornecer as credenciais do driver JDBC para seu aplicativo. Para ter mais informações, consulte Acesso por meio de conexões JDBC e ODBC.
- Se você criptografou o Catálogo de dados do AWS Glue, deve especificar outras ações nas políticas do IAM baseadas em identidade do Athena. Para ter mais informações, consulte Acesso a metadados criptografados do Athena no AWS Glue Data Catalog.
- Se você criar e usar grupos de trabalho, suas políticas deverão incluir o acesso relevante às ações do grupo de trabalho. Para obter informações detalhadas, consulte the section called “ Políticas do IAM para acessar grupos de trabalho” e the section called “Exemplo de políticas de grupo de trabalho”.

Política gerenciada pela AWS: AmazonAthenaFullAccess

A política gerenciada `AmazonAthenaFullAccess` concede acesso completo ao Athena.

Para fornecer o acesso, adicione as permissões aos seus usuários, grupos ou perfis:

- Usuários e grupos no AWS IAM Identity Center:

  Crie um conjunto de permissões. Siga as instruções em Criação de um conjunto de permissões no Guia do usuário do AWS IAM Identity Center.
- Usuários gerenciados no IAM com provedor de identidades:

  Crie um perfil para a federação de identidades. Siga as instruções em Criar um perfil para um provedor de identidades de terceiros (federação) no Guia do usuário do IAM.
- Usuários do IAM:
  - Crie um perfil que seu usuário possa assumir. Siga as instruções em Criação de um perfil para um usuário do IAM no Guia do usuário do IAM.

- (Não recomendado) Vincule uma política diretamente a um usuário ou adicione um usuário a um grupo de usuários. Siga as instruções em Adição de permissões a um usuário (console) no Guia do usuário do IAM.

Agrupamentos de permissões

A política `AmazonAthenaFullAccess` é agrupada nos seguintes conjuntos de permissões.

- **`athena`**: concede aos principais acesso aos recursos do Athena.
- **`glue`**: permite que os principais acessem bancos de dados, tabelas e partições do AWS Glue. Isso é necessário para que o principal possa usar o AWS Glue Data Catalog com o Athena.
- **`s3`**: permite que o principal grave e leia os resultados da consulta do Amazon S3, leia exemplos de dados do Athena disponíveis publicamente que residem no Amazon S3 e liste os buckets. Isso é necessário para que o principal possa usar o Athena para trabalhar com o Amazon S3.
- **`sns`**: permite que os principais listem os tópicos do Amazon SNS e obtenham os atributos dos tópicos. Isso permite que os principais usem os tópicos do Amazon SNS com o Athena para fins de monitoramento e alerta.
- **`cloudwatch`**: permite que os principais criem, leiam e excluam alarmes do CloudWatch. Para ter mais informações, consulte Controlar custos e monitorar consultas com métricas e eventos do CloudWatch.
- **`lakeformation`**: permite que os principais solicitem credenciais temporárias para acessar dados em um local de data lake registrado no Lake Formation. Para obter mais informações, consulte Underlying data access control (Controle de acesso a dados subjacentes) no Guia do desenvolvedor do AWS Lake Formation.
- **`datazone`**: permite que as entidades principais listem projetos, domínios e ambientes do Amazon DataZone. Para obter informações sobre como usar o DataZone no Athena, consulte Usar o Amazon DataZone no Athena.
- **`pricing`**: Fornece acesso total ao AWS Billing and Cost Management. Para obter mais informações, consulte GetProducts na Referência de APIs do AWS Billing and Cost Management.

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

```
            "Effect": "Allow",
            "Action": [
                "athena:*"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseGluePermissions",
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:DeleteDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:CreateTable",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:StartColumnStatisticsTaskRun",
                "glue:GetColumnStatisticsTaskRun",
                "glue:GetColumnStatisticsTaskRuns"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseQueryResultsPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
```

```
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:PutObject",
                "s3:PutBucketPublicAccessBlock"
            ],
            "Resource": [
                "arn:aws:s3:::aws-athena-query-results-*"
            ]
        },
        {
            "Sid": "BaseAthenaExamplesPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::athena-examples*"
            ]
        },
        {
            "Sid": "BaseS3BucketPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:ListAllMyBuckets"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseSNSPermissions",
            "Effect": "Allow",
            "Action": [
                "sns:ListTopics",
                "sns:GetTopicAttributes"
            ],
            "Resource": [
```

```
                "*"
            ]
        },
        {
            "Sid": "BaseCloudWatchPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:GetMetricData"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseLakeFormationPermissions",
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseDataZonePermissions",
            "Effect": "Allow",
            "Action": [
                "datazone:ListDomains",
                "datazone:ListProjects",
                "datazone:ListAccountEnvironments"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BasePricingPermissions",
            "Effect": "Allow",
            "Action": [
                "pricing:GetProducts"
            ],
```

```
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Política gerenciada AWS: AWSQuicksightAthenaAccess

`AWSQuicksightAthenaAccess` concede acesso às ações que o Amazon QuickSight exige para integração com o Athena. É possível anexar a política `AWSQuicksightAthenaAccess` a suas identidades do IAM. Anexe essa política somente aos principais que usam o Amazon QuickSight com o Athena. Essa política inclui algumas ações do Athena que estão defasadas e não foram incluídas na API pública atual, ou que são usadas apenas com os drivers JDBC e ODBC.

Agrupamentos de permissões

A política `AWSQuicksightAthenaAccess` é agrupada nos seguintes conjuntos de permissões.

- **`athena`**: permite que o principal execute consultas em recursos do Athena.
- **`glue`**: permite que os principais acessem bancos de dados, tabelas e partições do AWS Glue. Isso é necessário para que o principal possa usar o AWS Glue Data Catalog com o Athena.
- **`s3`**: permite que o principal grave e leia os resultados das consultas no Amazon S3.
- **`lakeformation`**: permite que os principais solicitem credenciais temporárias para acessar dados em um local de data lake registrado no Lake Formation. Para obter mais informações, consulte Underlying data access control (Controle de acesso a dados subjacentes) no Guia do desenvolvedor do 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": [
                "*"
            ]
        }
    ]
}
```

Atualizações do Athena para políticas gerenciadas pela AWS

Visualize detalhes sobre as atualizações nas políticas gerenciadas pela AWS para o Athena desde que esse serviço começou a monitorar essas alterações.

Alteração	Descrição	Data
AmazonAthenaFullAccess: atualização da política existente	As permissões `datazone:ListDomains` , `datazone:ListProjects` e `datazone:ListAccountEnvironments` foram adicionadas para possibilitar que os usuários do Athena trabalhem com	3 de janeiro de 2024

Alteração	Descrição	Data
	domínios, projetos e ambientes do Amazon DataZone. Para ter mais informações, consulte Usar o Amazon DataZone no Athena.	
AmazonAthenaFullAccess: atualização da política existente	As permissões `glue:StartColumnStatisticsTaskRun` , `glue:GetColumnStatisticsTaskRun` e `glue:GetColumnStatisticsTaskRuns` foram adicionadas para fornecer ao Athena o direito de chamar o AWS Glue para recuperar as estatísticas do atributo otimizador baseado em custos. Para ter mais informações, consulte Usar o otimizador baseado em custos.	3 de janeiro de 2024
AmazonAthenaFullAccess: atualização da política existente	O Athena adicionou `pricing:GetProducts` para fornecer acesso ao AWS Billing and Cost Management. Para obter mais informações, consulte GetProducts na Referência de APIs do AWS Billing and Cost Management.	25 de janeiro de 2023
AmazonAthenaFullAccess: atualização da política existente	O Athena adicionou `cloudwatch:GetMetricData` para recuperar valores de métricas do CloudWatch. Para obter mais informações, consulte GetMetricData na Referência de APIs do Amazon CloudWatch.	14 de novembro de 2022

Alteração	Descrição	Data
AmazonAthenaFullAccess e AWSQuicksightAthenaAccess: atualizações às políticas existentes	O Athena adicionou `s3:PutBucketPublicAccessBlock` para habilitar o bloqueio do acesso público aos buckets criados pelo Athena.	7 de julho de 2021
O Athena começou a monitorar as alterações	O Athena começou a monitorar as alterações em suas políticas gerenciadas pela AWS.	7 de julho de 2021

Acesso por meio de conexões JDBC e ODBC

Para acessar os recursos e Serviços da AWS, como o Athena e os buckets do Amazon S3, insira as credenciais do driver JDBC ou ODBC na sua aplicação. Se estiver usando o driver JDBC ou ODBC, verifique se a política de permissões do IAM inclui todas as ações listadas em Política gerenciada AWS: AWSQuicksightAthenaAccess.

Sempre que você usar as políticas do IAM, siga as práticas recomendadas do IAM. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM no Manual do usuário do IAM.

Métodos de autenticação

Os drivers Athena JDBC e ODBC são compatíveis com a autenticação baseada em SAML 2.0, incluindo os seguintes provedores de identidade:

- Active Directory Federation Services (AD FS)
- Azure Active Directory (AD)
- Okta
- PingFederate

Para obter mais informações, consulte os guias de instalação e configuração dos respectivos drivers, que podem ser baixados em formato PDF nas páginas dos drivers JDBC e ODBC. Para obter informações adicionais relacionadas, consulte o seguinte:

- Permitir acesso federado à API do Athena

- Usar o Lake Formation e os drivers JDBC e ODBC do Athena para acesso federado ao Athena
- Configurar o Single Sign-On usando ODBC, SAML 2.0 e o provedor de identidade Okta

Para obter informações sobre as versões mais recentes dos drivers JDBC e ODBC e as respectivas documentações, consulte Conectar-se ao Amazon Athena com JDBC e Conectar-se ao Amazon Athena com ODBC.

Acesso ao Amazon S3

Você pode conceder acesso aos locais do Amazon S3 usando políticas baseadas em identidade, políticas de recursos de bucket, políticas de ponto de acesso ou qualquer combinação das opções acima. Quando os atores interagem com o Athena, suas permissões passam pelo Athena para determinar o que o Athena poderá acessar. Isso significa que os usuários devem ter permissões para acessar os buckets do Amazon S3 com a finalidade de consultá-los com o Athena.

Sempre que você usar as políticas do IAM, siga as práticas recomendadas do IAM. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM no Manual do usuário do IAM.

Quando você configura `aws:SourceIp` suas políticas, o Athena acessa o bucket do Amazon S3 usando o endereço IP que você especifica. Não é possível restringir ou permitir o acesso aos recursos do Amazon S3 com base nas chaves de condição `aws:SourceVpc` ou `aws:SourceVpce`.

> **Note**
>
> Os grupos de trabalho do Athena que usam a autenticação do Centro de Identidade do IAM requerem que as concessões de acesso do S3 sejam configuradas para o uso de identidades de propagação de identidade confiável. Para obter mais informações, consulte S3 Access Grants and directory identities no Guia do usuário do Amazon Simple Storage Service.

Pontos de acesso e aliases de pontos de acesso do Amazon S3

Se você tem um conjunto de dados compartilhado em um bucket do Amazon S3, manter uma única política de bucket que gerencia o acesso para centenas de casos de uso pode ser um desafio.

Os pontos de acesso de bucket do Amazon S3 ajudam a resolver esse problema. Um bucket pode ter vários pontos de acesso, cada um com uma política que controla o acesso ao bucket de uma maneira diferente.

Para cada ponto de acesso que você cria, o Amazon S3 gera um alias que representa esse ponto. Como o alias está no formato de nome de bucket do Amazon S3, você pode usar o alias no cláusula `LOCATION` das instruções `CREATE TABLE` no Athena. O acesso do Athena ao bucket é controlado pela política do ponto de acesso que o alias representa.

Para obter mais informações, consulte Local da tabela no Amazon S3 e Usar pontos de acesso no Manual do usuário do Amazon S3.

Usar chaves de contexto CalledVia

Para maior segurança, é possível usar a chave de contexto de condição global aws:CalledVia. A chave `aws:CalledVia` contém uma lista ordenada de cada serviço na cadeia que fez solicitações em nome do principal. Especificando o nome da entidade principal de serviço do Athena `athena.amazonaws.com` para as chaves de contexto `aws:CalledVia`, é possível limitar as solicitações somente àquelas feitas pelo Athena. Para ter mais informações, consulte Usar o Athena com chaves de contexto CalledVia.

Recursos adicionais do

Para obter informações detalhadas e exemplos de como conceder acesso ao Amazon S3, consulte os seguintes recursos:

- Demonstrações de exemplo: gerenciar acesso no Guia do usuário do Amazon S3.
- Como posso conceder acesso entre contas a objetos que estão em buckets do Amazon S3? na Central de Conhecimento da AWS.
- Acesso entre contas no Athena aos buckets do Amazon S3.

Acesso entre contas no Athena aos buckets do Amazon S3

Um cenário comum do Amazon Athena é conceder acesso a usuários em uma conta diferente do proprietário do bucket de maneira que eles possam realizar as consultas. Neste caso, use uma política de bucket para conceder acesso.

> **Note**
>
> Para obter informações sobre o acesso entre contas aos catálogos de dados do AWS Glue pelo Athena, consulte Acesso aos catálogos de dados do AWS Glue entre contas.

A política de bucket de exemplo a seguir, criada e aplicada ao bucket `s3://DOC-EXAMPLE-BUCKET` pelo proprietário do bucket, concede acesso a todos os usuários na conta `123456789123`, que é uma conta diferente.

```
{
   "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/*"
          ]
       }
    ]
 }
```

Para conceder acesso a um determinado usuário em uma conta, substitua a chave `Principal` por uma chave que especifique o usuário, em vez de `root`. Por exemplo, para o perfil do usuário `Dave`, use `arn:aws:iam::123456789123:user/Dave`.

Acesso entre contas a um bucket criptografado com uma chave personalizada do AWS KMS

Se você tem um bucket do Amazon S3 criptografado com uma chave personalizada do AWS Key Management Service (AWS KMS), talvez seja necessário conceder aos usuários acesso a ele de outra conta da Amazon Web Services.

Conceder acesso a um bucket criptografado de AWS KMS na Conta A a um usuário na Conta B requer as seguintes permissões:

- A política de bucket na Conta A deve conceder acesso ao perfil assumido pela Conta B.
- A política de chaves do AWS KMS na Conta A deve conceder acesso ao perfil assumido pelo usuário na Conta B.
- O perfil do AWS Identity and Access Management (IAM) assumido na Conta B deve conceder acesso ao bucket e à chave na Conta A.

Os procedimentos a seguir descrevem como conceder cada uma dessas permissões.

Para conceder acesso ao bucket na Conta A para o usuário na Conta B

- Na Conta A, revise a política de bucket do S3 e confirme se há uma instrução que permita o acesso a partir do ID da conta da Conta B.

  Por exemplo, a política de bucket a seguir permite o acesso `s3:GetObject` à conta ID `111122223333`:

```
{
  "Id": "ExamplePolicy1",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ExampleStmt1",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
      "Principal": {
        "AWS": [
          "111122223333"
        ]
      }
    }
  ]
}
```

Para conceder acesso ao usuário na Conta B a partir da política de chave do AWS KMS na Conta A

1. Na política de chaves do AWS KMS para a Conta A, conceda permissões ao perfil assumido na Conta B para as ações a seguir:

   - `kms:Encrypt`
   - `kms:Decrypt`
   - `kms:ReEncrypt*`
   - `kms:GenerateDataKey*`
   - `kms:DescribeKey`

   O exemplo a seguir concede acesso de chave a somente um perfil do 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. Na Conta A, revise a política de chave usando a visualização de políticas do AWS Management Console.
3. Na política de chave, verifique se a seguinte declaração lista a Conta B como principal.

```
"Sid": "Allow use of the key"
```

4. Se a instrução `"Sid": "Allow use of the key"` não estiver presente, execute as seguintes etapas:

   a. Alterne para exibir a política de chave usando a exibição padrão do console.

   b. Adicione o ID da conta B como uma conta externa com acesso à chave.

Para conceder acesso ao bucket e à chave na Conta A do perfil do IAM assumido pela Conta B

1. Na Conta B, abra o console do IAM em https://console.aws.amazon.com/iam/.
2. Abra o perfil do IAM associado ao usuário na Conta B.
3. Analise a lista de políticas de permissões aplicadas ao perfil do IAM.
4. Certifique-se de aplicar uma política que conceda acesso ao bucket.

   A instrução exemplificada a seguir concede ao perfil do IAM acesso às operações `s3:GetObject` e `s3:PutObject` no 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. Certifique-se de que seja aplicada uma política que conceda acesso à chave.

> **Note**
>
> Se o perfil do IAM assumido na Conta B já tiver acesso de administrador, não será necessário conceder acesso à chave usando as políticas do IAM do usuário.

A instrução exemplificada a seguir concede ao perfil do IAM acesso para usar a chave `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"
    }
  ]
}
```

Acesso entre contas a objetos de bucket

Os objetos carregados por uma conta (Conta C) que não seja a conta proprietária do bucket (Conta A) podem exigir ACLs explícitas em nível de objeto que concedam acesso de leitura à conta de consulta (Conta B). Para evitar esse requisito, a Conta C deve assumir uma função na Conta A antes de colocar objetos no bucket da Conta A. Para obter mais informações, consulte Como posso fornecer acesso entre contas a objetos que estão em buckets do Amazon S3?.

Acesso granular a bancos de dados e tabelas no AWS Glue Data Catalog

Se você usar o AWS Glue Data Catalog com o Amazon Athena, poderá definir políticas no nível do recurso para os objetos bancos de dados e tabelas do Catálogo de dados, que são usados no Athena.

> **Note**
>
> O termo “controle de acesso refinado” usado aqui se refere à segurança do banco de dados e da tabela. Para obter informações sobre segurança da coluna, da linha e da célula, consulte Data filtering and cell-level security in Lake Formation (Filtragem de dados e segurança da célula no Lake Formation).

Você define as permissões no nível do recurso nas políticas baseadas em identidade do IAM.

> **Important**
>
> Esta seção aborda as permissões no nível do recurso nas políticas baseadas em identidade do IAM. Elas são diferentes das políticas com base em recursos. Para obter mais informações sobre as diferenças, consulte Políticas baseadas em identidade e em recurso no Guia do usuário do IAM.

Consulte os seguintes tópicos para essas tarefas:

Para executar essa tarefa	Consulte o tópico a seguir
Criar uma política do IAM que define acesso granular a recursos	Criação de políticas do IAM no Guia do usuário do IAM.
Saiba mais sobre as políticas baseadas em identidade do IAM usadas no AWS Glue	Políticas baseadas em identidades (políticas do IAM) no Guia do desenvolvedor do AWS Glue.

Nesta seção

- Limitações
- Acesso do AWS Glue ao catálogo e banco de dados por Região da AWS
- Versões e partições de tabela no AWS Glue

- Exemplos de permissões granulares para tabelas e bancos de dados

Limitações

Considere as seguintes limitações ao usar um controle de acesso granular com o AWS Glue Data Catalog e o Athena:

- Os grupos de trabalho do Athena habilitados para o Centro de Identidade do IAM requerem que o Lake Formation seja configurado para usar as identidades do Centro de Identidade do IAM. Para obter mais informações, consulte Integrating IAM Identity Center no Guia do desenvolvedor do AWS Lake Formation.
- Você pode limitar o acesso apenas a bancos de dados e tabelas. Controles de acesso granulares se aplicam no nível de tabela e não é possível limitar o acesso a partições individuais dentro de uma tabela. Para ter mais informações, consulte Versões e partições de tabela no AWS Glue.
- O AWS Glue Data Catalog contém os seguintes recursos: `CATALOG`, `DATABASE`, `TABLE` e `FUNCTION`.

  > **Note**
  >
  > Nessa lista, os recursos em comum entre o Athena e o AWS Glue Data Catalog são `TABLE`, `DATABASE` e `CATALOG` para cada conta. `Function` é específico de AWS Glue. Para as ações de exclusão no Athena, você deve incluir permissões para as ações do AWS Glue. Consulte Exemplos de permissões granulares para tabelas e bancos de dados.

  A hierarquia é a seguinte: `CATALOG` é um ancestral de todos os `DATABASES` em cada conta, e cada `DATABASE` é um ancestral para todas as suas `TABLES` e `FUNCTIONS`. Por exemplo, para uma tabela chamada `table_test` que pertence a um banco de dados db no catálogo em sua conta, seus ancestrais são db e catálogo na sua conta. Para o banco de dados db, seu ancestral é o catálogo em sua conta e seus descendentes são tabelas e funções. Para obter mais informações sobre a estrutura hierárquica de recursos, consulte Lista de ARNs no catálogo de dados no Guia do desenvolvedor do AWS Glue.
- Para qualquer ação do Athena que não seja de exclusão em um recurso, como `CREATE DATABASE`, `CREATE TABLE`, `SHOW DATABASE`, `SHOW TABLE` ou `ALTER TABLE`, você precisa de permissões para chamar essa ação no recurso (tabela ou banco de dados) e em todos os ancestrais do recurso no Catálogo de dados. Por exemplo, para uma tabela, seus ancestrais são o banco de dados ao qual ela pertence e o catálogo da conta. Para um banco de dados, seu

ancestral é o catálogo da conta. Consulte Exemplos de permissões granulares para tabelas e bancos de dados.

- Para uma ação de exclusão no Athena, como `DROP DATABASE` ou `DROP TABLE`, você também precisa de permissões para chamar a ação em todos os ancestrais e descendentes do recurso no catálogo de dados. Por exemplo, para excluir um banco de dados, você precisa de permissões no banco de dados, no catálogo, que é seu ancestral, e em todas as tabelas e funções definidas pelo usuário, que são seus descendentes. Uma tabela não tem descendentes. Para executar `DROP TABLE`, você precisa de permissões para essa ação na tabela, no banco de dados ao qual ela pertence e no catálogo. Consulte Exemplos de permissões granulares para tabelas e bancos de dados.

Acesso do AWS Glue ao catálogo e banco de dados por Região da AWS

Para que o Athena funcione com o AWS Glue, é necessária uma política que conceda acesso ao banco de dados e ao AWS Glue Data Catalog em sua Região da AWS conta. Para criar bancos de dados, a permissão `CreateDatabase` também é necessária. No exemplo de política a seguir, substitua a Região da AWS, o ID da Conta da AWS e o nome do banco de dados por seus próprios dados.

```
{
   "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"
   ]
}
```

Versões e partições de tabela no AWS Glue

No AWS Glue, tabelas podem ter partições e versões. As versões e partições de tabelas não são consideradas recursos independentes no AWS Glue. O acesso a versões e partições de tabela é garantido concedendo acesso na tabela e em recursos ancestrais da tabela.

Para fins de um controle de acesso detalhado, as seguintes permissões de acesso se aplicam:

- Controles de acesso granulares se aplicam no nível de tabela. Você pode limitar o acesso apenas a bancos de dados e tabelas. Por exemplo, se você permitir acesso a uma tabela particionada, esse acesso se aplicará a todas as partições na tabela. Você não pode limitar o acesso a partições individuais em uma tabela.

  > **Important**
  >
  > Para executar as ações do AWS Glue em partições, são necessárias permissões para ações de partição nos níveis do catálogo, do banco de dados e da tabela. Ter acesso às partições em uma tabela não é suficiente. Por exemplo, para executar `GetPartitions` em uma tabela `myTable` no banco de dados `myDB`, você deve conceder permissões de `glue:GetPartitions` ao catálogo, ao banco de dados `myDB` e aos recursos de `myTable`.

- Controles de acesso granulares não se aplicam a versões da tabela. Assim como ocorre com as partições, o acesso a versões anteriores de uma tabela é concedido por meio de acesso às APIs da versão da tabela do AWS Glue na tabela e aos ancestrais da tabela.

Para obter informações sobre permissões para as ações do AWS Glue, consulte Permissões da API do AWS Glue: referência de ações e recursos no Guia do desenvolvedor do AWS Glue.

Exemplos de permissões granulares para tabelas e bancos de dados

A tabela a seguir lista exemplos de políticas baseadas em identidade do IAM que permitem acesso granular a bancos de dados e tabelas no Athena. Recomendamos começar com esses exemplos e, de acordo com as suas necessidades, ajustá-los para permitir ou negar ações específicas a determinados bancos de dados e tabelas.

Esses exemplos incluem acesso a bancos de dados e catálogos para que Athena e o AWS Glue possam funcionar juntos. Para várias regiões da AWS, inclua políticas similares para cada um dos catálogos e bancos de dados, sendo uma linha para cada região.

Nesses exemplos, substitua o banco de dados `example_db` e os nomes das tabelas `test` pelos nomes dos seus próprios bancos de dados e tabelas.

Instrução DDL	Exemplo de uma política de acesso do IAM que concede acesso ao recurso
ALTER DATABASE	Permite que você modifique as propriedades do banco de dados do `example_db` . `{ "Effect": "Allow", "Action": [ "glue:GetDatabase", "glue:UpdateDatabase" ], "Resource": [ "arn:aws:glue: us-east-1 :123456789012 :catalog", "arn:aws:glue: us-east-1 :123456789012 :database / example_db " ] }`
CREATE DATABASE	Permite que você crie o banco de dados chamado `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 " ] }`
CRIAR TABELA	Permite que você crie uma tabela chamada `test` no banco de dados do `example_db` . `{ "Sid": "DatabasePermissions", "Effect": "Allow",`

Instrução DDL	Exemplo de uma política de acesso do IAM que concede acesso ao recurso

```
   "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"
   ]
}
```

Instrução DDL	Exemplo de uma política de acesso do IAM que concede acesso ao recurso
DROP DATABASE	Permite que você elimine o banco de dados do `example_db` incluindo todas as suas tabelas. `{ "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 /*"] }`

Instrução DDL	Exemplo de uma política de acesso do IAM que concede acesso ao recurso
DESCARTAR TABELA	Permite que você elimine uma tabela particionada chamada `test` no banco de dados do `example_db` . Se a tabela não tiver partições, não inclua ações de partição. `{ "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"] }`

Instrução DDL	Exemplo de uma política de acesso do IAM que concede acesso ao recurso
MSCK REPAIR TABLE	Permite que você atualize os metadados do catálogo depois de adicionar partições compatíveis com o Hive à tabela chamada `test` no banco de dados `example_db` . `{ "Effect": "Allow", "Action": ["glue:GetDatabase", "glue:CreateDatabase", "glue:GetTable", "glue:GetPartitions", "glue:GetPartition", "glue:BatchCreatePartition"], "Resource": ["arn:aws:glue: us-east-1 :123456789012 :catalog", "arn:aws:glue: us-east-1 :123456789012 :database / example_db ", "arn:aws:glue: us-east-1 :123456789 012 :table/example_db /test"] }`
SHOW DATABASES	Permite que você liste todos os bancos de dados no 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/*" ] }`

Instrução DDL	Exemplo de uma política de acesso do IAM que concede acesso ao recurso
SHOW TABLES	Permite que você liste todas as tabelas no banco de dados do `example_db` . `{ "Effect": "Allow", "Action": [ "glue:GetDatabase", "glue:GetTables" ], "Resource": [ "arn:aws:glue: us-east-1 :123456789012 :catalog", "arn:aws:glue: us-east-1 :123456789012 :database / example_db ", "arn:aws:glue: us-east-1 :123456789012 :table/example_db /*" ] }`

Acesso aos catálogos de dados do AWS Glue entre contas

Você pode usar o recurso de catálogo do AWS Glue entre contas do Athena para registrar um catálogo do AWS Glue de outra conta sua. Depois de configurar as permissões do IAM necessárias para o AWS Glue e registrar o catálogo como um recurso DataCatalog do Athena, você poderá usar o Athena para executar consultas entre contas. Para obter informações sobre como usar o console do Athena para registrar um catálogo de outra conta, consulte Registrar um AWS Glue Data Catalog de outra conta.

Para obter mais informações sobre o acesso entre contas no AWS Glue, consulte Concessão de acesso entre contas no Guia do desenvolvedor do AWS Glue.

Antes de começar

Como esse recurso usa as APIs existentes do recurso `DataCatalog` e as funcionalidades do Athena para habilitar o acesso entre contas, recomendamos ler os seguintes tópicos antes de começar:

- Conectar a origens de dados: contém tópicos sobre como usar o Athena com as fontes de catálogo de dados AWS Glue, Hive ou Lambda.
- Políticas de catálogo de dados de exemplo: mostra como escrever políticas que controlam o acesso aos catálogos de dados.
- Usar a AWS CLI com metastores do Hive: mostra como usar a AWS CLI com metastores do Hive, mas contém casos de uso aplicáveis a outras origens de dados.

Considerações e limitações

No momento, o acesso entre contas ao catálogo do AWS Glue no Athena tem as seguintes limitações:

- O recurso está disponível apenas nas Regiões da AWS compatíveis com o mecanismo Athena versão 2 ou posterior. Para obter informações sobre as versões do mecanismo do Athena, consulte Versionamento do mecanismo do Athena. Para fazer upgrade da versão do mecanismo para um grupo de trabalho, consulte Alterar versões do mecanismo do Athena.
- Quando você registra o AWS Glue Data Catalog de outra conta na sua conta, cria um recurso `DataCatalog` regional vinculado aos dados da outra conta somente nessa região específica.
- No momento, as instruções `CREATE VIEW` que incluem um catálogo do AWS Glue entre contas não são permitidas.
- Catálogos criptografados usando chaves gerenciadas da AWS não podem ser consultados entre contas. Em vez disso, use chaves gerenciadas pelo cliente (`KMS_CMK`) para os catálogos que você deseja consultar entre contas. Para obter informações sobre as diferenças entre chaves gerenciadas pelo cliente e pela AWS, consulte Chaves do cliente e chaves da AWS no Guia do desenvolvedor do AWS Key Management Service.

Conceitos básicos

No cenário abaixo, a conta do “tomador” (666666666666) deseja executar uma consulta `SELECT` que faz referência ao catálogo do AWS Glue que pertence à conta do “proprietária” (999999999999), como no seguinte exemplo:

```
SELECT * FROM ownerCatalog.tpch1000.customer
```

No procedimento a seguir, as etapas 1a e 1b mostram como conceder à conta do tomador acesso aos recursos do AWS Glue da conta do proprietário, tanto da perspectiva do tomador quanto do

proprietário. O exemplo concede acesso ao banco de dados `tpch1000` e à tabela `customer`. Altere esses nomes de exemplo de acordo com as suas necessidades.

Etapa 1b: criar um perfil de tomador com acesso aos recursos do AWS Glue do proprietário

Para criar um perfil de conta do tomador com uma política para acessar os recursos do AWS Glue da conta do proprietário, é possível usar o console do AWS Identity and Access Management (IAM) ou a API do IAM. O procedimento a seguir usa o console do IAM.

Para criar um perfil de tomador e a política para acessar os recursos do AWS Glue da conta do proprietário

1. Faça login no console do IAM em https://console.aws.amazon.com/iam/ usando a conta do tomador.
2. No painel de navegação, expanda Gerenciamento de acesso e escolha Políticas.
3. Escolha Criar política.
4. Em Editor de políticas, escolha JSON.
5. No editor de política, insira a seguinte política e modifique-a de acordo com os seus requisitos:

```
{
    "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. Escolha Próximo.
7. Na página Revisar e criar, insira um nome para a política em Nome da política (p. ex., **`CrossGluePolicyForBorrowerRole`**).
8. Escolha Criar política.
9. No painel de navegação, escolha Roles.

10. Escolha Criar Perfil.
11. Na página Selecionar entidade confiável, escolha Conta da AWS e, em seguida, selecione Avançar.
12. Na página Adicionar permissões, insira na caixa de pesquisa o nome da política que você criou (p. ex., **`CrossGluePolicyForBorrowerRole`**).
13. Marque a caixa de seleção ao lado do nome da política e escolha Avançar.
14. Na página Name, review, and create (Nomear, revisar e criar), para Role name (Nome da função), digite um nome para a função (por exemplo, **`CrossGlueBorrowerRole`**).
15. Selecione Criar função.

Etapa 1b: criar uma política do proprietário para permitir que o tomador tenha acesso ao AWS Glue

Para conceder acesso ao AWS Glue da conta do proprietário (999999999999) para o perfil do tomador, é possível usar o console do AWS Glue ou a operação de API PutResourcePolicy do AWS Glue. O procedimento a seguir usa o console do AWS Glue.

Para conceder acesso ao AWS Glue para a conta do mutuário do proprietário

1. Faça login no console do AWS Glue em https://console.aws.amazon.com/glue/ usando a conta do proprietário.
2. No painel de navegação, expanda Data Catalog e escolha Configurações do catálogo.
3. Na caixa Permissions (Permissões), insira a política como mostrado a seguir. Em *`rolename`*, insira o perfil que o tomador criou na etapa 1a (p. ex., **`CrossGlueBorrowerRole`**). Para aumentar o escopo da permissão, é possível usar o caractere curinga * para os dois tipos de recurso: banco de dados e tabela.

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

Ao concluir, recomendamos usar a API do AWS Glue para fazer algumas chamadas de teste entre contas a fim de confirmar se as permissões estão configuradas conforme o esperado.

Etapa 2: o mutuário registra o AWS Glue Data Catalog que pertence à conta de proprietário

O procedimento a seguir mostra como usar o console do Athena para configurar um AWS Glue Data Catalog na conta da Amazon Web Services do proprietário da fonte de dados. Para obter informações sobre como usar operações de API em vez do console para registrar o catálogo, consulte Usar a API para registrar um catálogo de dados do Athena que pertença à conta do proprietário.

Para registrar um AWS Glue Data Catalog pertencente a outra conta

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.

3. Expanda Administração e escolha Fontes de dados.

4. No canto superior direito do console, escolha Create data source (Criar fonte de dados).
5. Na página Escolher uma fonte de dados, em Fontes de dados, escolha S3 - AWS Glue Data Catalog e depois Próximo.
6. Na página Inserir detalhes da fonte de dados, na seção AWS Glue Data Catalog, para Escolher um AWS Glue Data Catalog, escolha AWS Glue Data Catalog em outra conta.
7. Em Data source details (Detalhes da origemdos dados), forneça as seguintes informações:
   - Data source name (Nome da origem dos dados): insira o nome que você deseja usar em suas consultas SQL para fazer referência ao catálogo de dados na outra conta.
   - Description (Descrição): insira uma descrição do catálogo de dados na outra conta (opcional).
   - Catalog ID (ID do catálogo): insira o ID de 12 dígitos da conta da Amazon Web Services à qual o catálogo de dados pertence. O ID da conta da Amazon Web Services é o ID do catálogo.
8. (Opcional) Expanda Etiquetas e adicione pares de chave-valor que você queira associar com a fonte de dados. Para obter mais informações sobre tags, consulte Etiquetar recursos do Athena.
9. Escolha Próximo.
10. Na página Review and create (Revisar e criar), analise as informações fornecidas e selecione Create data source (Criar fonte de dados). A página Data source details (Detalhes da fonte de dados) lista os bancos de dados e etiquetas do catálogo de dados que você registrou.
11. Escolha Data sources (Origens de dados). O catálogo de dados que você registrou está listado na coluna Data source name (Nome da fonte de dados).
12. Para visualizar ou editar as informações do catálogo de dados, escolha o catálogo e selecione Actions (Ações), Edit (Editar).
13. Para excluir o novo catálogo de dados, escolha o catálogo e selecione Actions (Ações) Delete (Excluir).

Etapa 3: O mutuário envia uma consulta

O tomador envia uma consulta que faz referência ao catálogo usando a sintaxe *`catálogo.banco de dados.tabela`*, conforme o seguinte exemplo:

```
SELECT * FROM ownerCatalog.tpch1000.customer
```

Em vez de usar a sintaxe totalmente qualificada, o tomador também pode especificar o catálogo contextualmente, passando o valor por QueryExecutionContext.

Permissões adicionais do Amazon S3

- Se a conta do tomador usar uma consulta do Athena para gravar novos dados em uma tabela na conta do proprietário, o proprietário não terá acesso automaticamente a esses dados no Amazon S3, mesmo que a tabela exista na conta do proprietário. Isso ocorre porque, a menos que seja configurado de outra forma, o tomador é o proprietário do objeto das informações no Amazon S3. Para permitir que o proprietário tenha acesso aos dados, defina as permissões nos objetos adequadamente em uma etapa adicional.
- Determinadas operações DDL entre contas, como MSCK REPAIR TABLE, exigem permissões do Amazon S3. Por exemplo, se a conta do tomador estiver executando uma operação `MSCK REPAIR` entre contas em uma tabela na conta do proprietário que tenha os dados em um bucket do S3 da conta do proprietário, esse bucket deverá conceder permissões ao perfil assumido pelo tomador para que a consulta ocorra com êxito.

Para informações sobre como conceder permissões de bucket, consulte Como definir permissões de bucket de ACL? no Guia do usuário do console do Amazon Simple Storage Service.

Usar um catálogo dinamicamente

Em alguns casos, você pode querer executar testes rápidos em um catálogo do AWS Glue entre contas sem a etapa de pré-requisito de registro. Você poderá executar dinamicamente consultas entre contas sem criar o objeto de recurso `DataCatalog` se as permissões necessárias do IAM e do Amazon S3 estiverem configuradas corretamente, conforme já foi descrito neste documento.

Para referenciar claramente um catálogo sem registro, use a sintaxe no seguinte exemplo:

```
SELECT * FROM "glue:arn:aws:glue:us-east-1:999999999999:catalog".tpch1000.customer
```

Use o formato “`glue:<arn>`”, em que *`<arn>`* é o ARN do AWS Glue Data Catalog que você deseja usar. No exemplo, o Athena usa essa sintaxe para apontar dinamicamente para o catálogo de dados do AWS Glue da conta 999999999999 como se você tivesse criado um objeto `DataCatalog` separado para ele.

Observações sobre o uso de catálogos dinâmicos

Ao usar catálogos dinâmicos, lembre-se dos pontos a seguir.

- O uso de um catálogo dinâmico requer as permissões do IAM que você normalmente usa para as operações de API do Catálogo de dados do Athena. A principal diferença é que o nome do recurso Catálogo de dados segue a convenção de nomenclatura `glue:*`.
- O ARN do catálogo deve pertencer à mesma região onde a consulta está sendo executada.
- Ao usar um catálogo dinâmico em uma consulta DML ou visualização, coloque-o entre aspas duplas escapadas (\"). Ao usar um catálogo dinâmico em uma consulta DDL, coloque-o entre caracteres de aceto grave (`).

Usar a API para registrar um catálogo de dados do Athena que pertença à conta do proprietário

Em vez de usar o console do Athena conforme descrito na Etapa 2, é possível usar operações de API para registrar o catálogo de dados pertencente à conta do proprietário.

O criador do recurso DataCatalog do Athena deve ter as permissões necessárias para executar a operação de API CreateDataCatalog do Athena. Dependendo dos seus requisitos, o acesso às operações adicionais de API pode ser necessário. Para ter mais informações, consulte Políticas de catálogo de dados de exemplo.

O corpo da seguinte solicitação `CreateDataCatalog` registra um catálogo do AWS Glue para acesso entre contas:

```
# 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"
}
```

O código de exemplo a seguir usa um cliente Java para criar o objeto `DataCatalog`.

```
# Sample code to create the DataCatalog through Java client
CreateDataCatalogRequest request = new CreateDataCatalogRequest()
    .withName("ownerCatalog")
    .withType(DataCatalogType.GLUE)
    .withParameters(ImmutableMap.of("catalog-id", "999999999999"));

athenaClient.createDataCatalog(request);
```

Após essas etapas, o mutuário deverá ver `ownerCatalog` quando chamar a operação de API ListDataCatalogs.

Recursos adicionais do

- Registrar um AWS Glue Data Catalog de outra conta
- Configure o acesso entre contas para um AWS Glue Data Catalog compartilhado usando o Amazon Athena no guia Padrões de orientação prescritivaAWS.
- Consultar AWS Glue Data Catalog entre contas usando o Amazon Athena no AWSBlog de Big Data
- Conceder acesso entre contas no AWS GlueGuia do desenvolvedor

Acesso a metadados criptografados do Athena no AWS Glue Data Catalog

Se você usa o AWS Glue Data Catalog com o Amazon Athena, pode habilitar a criptografia no AWS Glue Data Catalog usando o console do AWS Glue ou a API. Para obter informações, consulte Como criptografar seu Data Catalog no Guia do desenvolvedor do AWS Glue.

Se o AWS Glue Data Catalog estiver criptografado, você deverá adicionar as seguintes ações a todas as políticas que são usadas para acessar o 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)"
   }
}
```

Sempre que você usar as políticas do IAM, siga as práticas recomendadas do IAM. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM no Manual do usuário do IAM.

Acesso a grupos de trabalho e etiquetas

Um grupo de trabalho é um recurso gerenciado pelo Athena. Portanto, se sua política de grupo de trabalho usar ações que tomam `workgroup` como entrada, será necessário especificar o ARN do grupo de trabalho, onde *`workgroup-name`* é o nome do seu grupo de trabalho, da seguinte forma:

```
"Resource": [arn:aws:athena:region:AWSAcctID:workgroup/workgroup-name]
```

Por exemplo, para um grupo de trabalho chamado `test_workgroup` na região `us-west-2` da conta da Amazon Web Services `123456789012`, especifique o grupo de trabalho como um recurso usando o seguinte ARN:

```
"Resource":["arn:aws:athena:us-east-2:123456789012:workgroup/test_workgroup"]
```

Para acessar grupos de trabalho habilitados para a propagação de identidade confiável (TIP), os usuários do Centro de Identidade do IAM devem ser atribuídos ao `IdentityCenterApplicationArn` que é retornado pela resposta da ação de API GetWorkGroup do Athena.

- Para obter uma lista de políticas de grupo, consulte the section called “Exemplo de políticas de grupo de trabalho”.
- Para obter uma lista de políticas com base em tags para grupos de trabalho, consulte Políticas de controle de acesso do IAM baseadas em etiquetas.
- Para obter mais informações sobre como criar políticas do IAM para grupos de trabalho, consulte Políticas do IAM para acessar grupos de trabalho.
- Para obter uma lista completa de ações do Amazon Athena, consulte os nomes das ações de API na Referência de API do Amazon Athena.
- Para obter mais informações sobre as políticas do IAM, consulte Criar políticas com o editor visual no Guia do usuário do IAM.

Sempre que você usar as políticas do IAM, siga as práticas recomendadas do IAM. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM no Manual do usuário do IAM.

Permitir acesso a instruções preparadas

Este tópico aborda as permissões do IAM para instruções preparadas no Amazon Athena. Sempre que você usar as políticas do IAM, siga as práticas recomendadas do IAM. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM no Manual do usuário do IAM.

Para obter mais informações sobre instruções preparadas, consulte Utilizar consultas parametrizadas.

As permissões do IAM a seguir são necessárias para criar, gerenciar e executar instruções preparadas.

```
athena:CreatePreparedStatement
athena:UpdatePreparedStatement
athena:GetPreparedStatement
athena:ListPreparedStatements
athena:DeletePreparedStatement
```

Use essas permissões conforme mostrado na tabela a seguir.

Para fazer isso	use estas permissões
Executar uma consulta `PREPARE`	`athena:StartQueryExecution` `athena:CreatePreparedStatement`
Executar novamente uma consulta `PREPARE` para atualizar uma instrução preparada existente	`athena:StartQueryExecution` `athena:UpdatePreparedStatement`
Executar uma consulta `EXECUTE`	`athena:StartQueryExecution` `athena:GetPreparedStatement`
Executar uma consulta `DEALLOCATE PREPARE`	`athena:StartQueryExecution` `athena:DeletePreparedStatement`

Exemplo

O exemplo de política do IAM a seguir concede permissões para gerenciar e executar instruções preparadas em um ID de conta e grupo de trabalho especificados.

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

Usar o Athena com chaves de contexto CalledVia

Quando uma entidade principal faz uma solicitação para a AWS, a AWS reúne as informações da solicitação em um contexto de solicitação que avalia e autoriza a solicitação. É possível usar o elemento `Condition` de uma política JSON para comparar chaves no contexto da solicitação com os valores de chave especificados em sua política. As chaves de contexto de condição global são chaves de condição com um prefixo `aws:`.

A chave de contexto aws:CalledVia

Você pode usar a chave de contexto de condição global aws:CalledVia para comparar os serviços na política com os serviços que fazem solicitações em nome do principal do IAM (usuário ou função). Quando uma entidade principal faz uma solicitação a um AWS service (Serviço da AWS), esse serviço pode usar as credenciais do principal para fazer solicitações subsequentes a outros serviços. A chave `aws:CalledVia` contém uma lista ordenada de cada serviço na cadeia que fez solicitações em nome do principal.

Ao especificar um nome de elemento principal de serviço para a chave de contexto `aws:CalledVia`, você pode tornar a chave de contexto específica do AWS service (Serviço da AWS). Por exemplo, você pode usar a chave de condição `aws:CalledVia` para limitar as solicitações somente àquelas feitas pelo Athena. Para usar a chave de condição `aws:CalledVia`

em uma política com o Athena, especifique o nome do principal do serviço do Athena `athena.amazonaws.com`, como mostrado no exemplo a seguir.

```
 ...
    "Condition": {
        "ForAnyValue:StringEquals": {
            "aws:CalledVia": "athena.amazonaws.com"
        }
    }
...
```

Você pode usar a chave de contexto `aws:CalledVia` para garantir que os autores da chamada só tenham acesso a um recurso (como uma função do Lambda) se o chamarem pelo Athena.

> **Note**
>
> A chave de contexto `aws:CalledVia` não é compatível com o recurso de propagação de identidade confiável.

Adicionar uma chave de contexto CalledVia opcional para acesso granular a uma função do Lambda

O Athena exige que o autor da chamada tenha as permissões `lambda:InvokeFunction` para invocar a função do Lambda associada à consulta. A instrução a seguir permite acesso granular a uma função do Lambda para que o usuário utilize somente o Athena para invocar a função do 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"
                }
            }
```

```
        }
    ]
}
```

O exemplo a seguir mostra a adição da instrução anterior a uma política que permite a um usuário executar e ler uma consulta federada. Os principais com permissão para realizar essas ações podem executar consultas que especificam catálogos do Athena associados a uma origem dos dados federada. No entanto, o principal não pode acessar a função do Lambda associada, a menos que ela seja invocada pelo 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"
                }
            }
        }
    ]
}
```

Para obter mais informações sobre as chaves de condição `CalledVia`, consulte Chaves de contexto de condição global AWS no Manual do usuário do IAM.

Permitir acesso a um conector de dados do Athena para metastore externo do Hive

Os exemplos de política de permissão neste tópico demonstram as ações permitidas necessárias e os recursos para os quais são permitidas. Examine essas políticas com atenção e modifique-as de acordo com seus requisitos antes de anexar políticas de permissões semelhantes às identidades do 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 : permitir que uma entidade principal do IAM consulte dados usando o conector de dados do Athena para metastore externo do Hive

A política a seguir é anexada aos principais do IAM, além de Política gerenciada pela AWS: AmazonAthenaFullAccess, que concede acesso completo às ações do 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"
        }
    ]
}
```

Explicação de permissões

Ações permitidas	Explicação
`"s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket", "s3:PutObject", "s3:ListMultipartUploadParts", "s3:AbortMultipartUpload"`	As ações do `s3` permitem ler e gravar no recurso especificado como "`arn:aws:s3::: MyLambdaSpillBucket /MyLambdaSpillLocation `", em que *MyLambdaSpillLocation* identifica o bucket de vazamento que foi definido na configuração da(s) função(ões) do Lambda invocada(s). O identificador do recurso *arn:aws:lambda:*: MyAWSAcctId :layer:MyAthenaLambdaLayer :** é necessário somente se você usa uma camada do Lambda para criar dependências de tempo de execução personalizadas para reduzir o tamanho do artefato da função no momento da implantação. O * na última posição é um curinga para a versão da camada.
`"lambda:GetFunction", "lambda:GetLayerVersion", "lambda:InvokeFunction"`	Permite que as consultas chamem as funções do AWS Lambda especificadas no bloco `Resource`. Por exemplo, `arn:aws:lambda:*: MyAWSAcctId :function : MyAthenaLambdaFunction `, em que *MyAthenaLambdaFunction* especifica o nome da função do Lambda que será invocada. Várias funções podem ser especificadas conforme mostrado no exemplo.

Example : permitir que uma entidade principal do IAM crie um conector de dados do Athena para metastore externo do Hive

A política a seguir é anexada aos principais do IAM, além de Política gerenciada pela AWS: AmazonAthenaFullAccess, que concede acesso completo às ações do 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*"
        }
    ]
}
```

Explicação de permissões

Permite que as consultas chamem as funções do AWS Lambda para as funções do AWS Lambda especificadas no bloco `Resource`. Por exemplo, `arn:aws:lambda:*:`*`MyAWSAcctId`*`:function:`*`MyAthenaLambdaFunction`*, em que *`MyAthenaLambdaFunction`* especifica o nome da função do Lambda que será invocada. Várias funções podem ser especificadas conforme mostrado no exemplo.

Permitir acesso da função do Lambda aos metastores externos do Hive

Para invocar uma função do Lambda em sua conta, você deve criar uma função que tenha as seguintes permissões:

- `AWSLambdaVPCAccessExecutionRole`: uma permissão de função de execução do AWS Lambda para gerenciar interfaces de rede elásticas que conectam a função a uma VPC. Verifique se você tem um número suficiente de interfaces de rede e endereços IP disponíveis.

- `AmazonAthenaFullAccess`: a política gerenciada AmazonAthenaFullAccess concede acesso completo ao Athena.
- Uma política do Amazon S3 para permitir que a função do Lambda grave no S3 e que o Athena leia o S3.

Por exemplo, a política a seguir define a permissão para o local de vazamento `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"
            ]
        }
    ]
}
```

Sempre que você usar as políticas do IAM, siga as práticas recomendadas do IAM. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

Criar funções do Lambda

Para criar uma função do Lambda em sua conta, são necessárias as permissões de desenvolvimento de função ou a função `AWSLambdaFullAccess`. Para obter mais informações, consulte Políticas do IAM baseadas em identidade para o AWS Lambda.

Como o Athena usa o AWS Serverless Application Repository para criar funções do Lambda, o superusuário ou administrador que cria as funções do Lambda também deve ter políticas do IAM para permitir consultas federadas do Athena.

Registro de catálogo e operações de API de metadados

Para acessar a API de registro de catálogo e operações de API de metadados, use a Política gerenciada AmazonAthenaFullAccess . Se você não usar essa política, adicione as seguintes operações de API às suas políticas do Athena:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListDataCatalogs",
                "athena:GetDataCatalog",
                "athena:CreateDataCatalog",
                "athena:UpdateDataCatalog",
                "athena:DeleteDataCatalog",
                "athena:GetDatabase",
                "athena:ListDatabases",
                "athena:GetTableMetadata",
                "athena:ListTableMetadata"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Invocar o Lambda entre regiões

Para invocar uma função do Lambda em uma região diferente da região em que você executa as consultas do Athena, use o ARN completo da função do Lambda. Por padrão, o Athena invoca as funções do Lambda definidas na mesma região. Se você precisar invocar uma função do Lambda para acessar um metastore do Hive em uma região diferente da região em que você executa as consultas do Athena, informe o ARN completo da função do Lambda.

Por exemplo, suponha que você defina o catálogo `ehms` na região Europa (Frankfurt) `eu-central-1` para usar a função do Lambda a seguir na região Leste dos EUA (Norte da Virgínia).

```
arn:aws:lambda:us-east-1:111122223333:function:external-hms-service-new
```

Quando você especificar o ARN completo dessa maneira, o Athena poderá chamar a função do Lambda `external-hms-service-new` em `us-east-1` para buscar os dados do metastore do Hive de `eu-central-1`.

> **Note**
>
> O catálogo `ehms` deve ser registrado na mesma região onde você executa as consultas do Athena.

Invocar o Lambda entre contas

Às vezes, poderá ser necessário ter acesso a um metastore do Hive por meio de outra conta. Por exemplo, para executar um metastore do Hive, você pode iniciar um cluster do EMR de uma conta diferente daquela usada para as consultas do Athena. Grupos ou equipes diferentes podem executar o metastore do Hive com contas diferentes dentro de sua VPC. Ou você pode querer acessar metadados de diferentes metastores do Hive de diferentes grupos ou equipes.

O Athena usa o suporte do AWS Lambda ao acesso entre contas para habilitar o acesso entre contas aos metastores do Hive.

> **Note**
>
> Observe que, normalmente, o acesso entre contas para o Athena implica no acesso entre contas para os metadados e os dados no Amazon S3.

Imagine o seguinte cenário:

- A conta `111122223333` configura a função do Lambda `external-hms-service-new` em us-east-1 no Athena para acessar um metastore do Hive executado em um cluster do EMR.
- A conta `111122223333` deseja permitir que a conta 444455556666 acesse os dados do metastore do Hive.

Para conceder acesso da conta `444455556666` à função do Lambda `external-hms-service-new`, a conta `111122223333` usa o comando da AWS CLI `add-permission` a seguir. O comando foi formatado para legibilidade.

```
$ 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\"}"
}
```

Para verificar a permissão do Lambda, use o comando `get-policy`, como no exemplo a seguir. O comando foi formatado para legibilidade.

```
$ 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\"}]}"
}
```

Depois de adicionar a permissão, você poderá usar um ARN completo da função do Lambda em `us-east-1`, conforme mostrado abaixo, ao definir o catálogo `ehms`:

```
arn:aws:lambda:us-east-1:111122223333:function:external-hms-service-new
```

Para obter informações sobre invocação entre regiões, consulte Invocar o Lambda entre regiões anteriormente neste tópico.

Conceder acesso a dados entre contas

Antes de executar consultas do Athena, você deve conceder acesso entre contas aos dados no Amazon S3. Você pode fazer isso por meio de uma das seguintes maneiras:

- Atualize a política da lista de controle de acesso do bucket do Amazon S3 com um ID de usuário canônico.
- Adicione o acesso entre contas à política de bucket do Amazon S3.

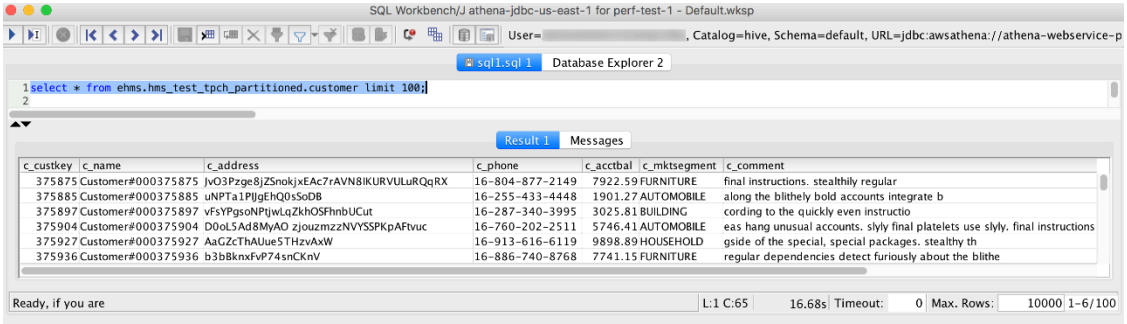
Por exemplo, adicione a política a seguir à política de bucket do Amazon S3 na conta `111122223333` para permitir que a conta `444455556666` leia os dados do local do Amazon S3 especificado.

```
{
    "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**
>
> Talvez seja necessário conceder acesso entre contas ao Amazon S3, não apenas aos dados, mas também ao local de vazamento do Amazon S3. A função do Lambda vaza os dados excedentes para o local de vazamento quando o tamanho do objeto de resposta ultrapassa um determinado limite. Consulte o início deste tópico para obter uma política de exemplo.

No exemplo atual, após o acesso entre contas ser concedido a `444455556666`,, `444455556666` pode usar o catálogo `ehms` em sua própria `account` para consultar tabelas definidas na conta `111122223333`.

No exemplo a seguir, o perfil do SQL Workbench `perf-test-1` é para a conta `444455556666`. A consulta usa o catálogo `ehms` para acessar o metastore do Hive e os dados do Amazon S3 na conta `111122223333`.



Exemplo de políticas de permissões do IAM para permitir consulta federada do Athena

Os exemplos de política de permissão neste tópico demonstram as ações permitidas necessárias e os recursos para os quais são permitidas. Examine essas políticas com atenção e modifique-as de acordo com suas necessidades antes de anexá-las às identidades do IAM.

Para obter informações sobre como anexar políticas a identidades do IAM, consulte Adicionar e remover permissões de identidade do IAM no Guia do usuário do 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 : permitir que uma entidade principal do IAM execute e retorne resultados usando a consulta federada do Athena

A política de permissões baseadas em identidade a seguir permite ações que um usuário ou outro principal do IAM precisa para usar a consulta federada do Athena. Os principais que têm permissão para realizar essas ações podem executar consultas que especificam catálogos do Athena associados a uma origem dos dados federada.

```
{
    "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/*"
            ]
        }
    ]
}
```

Explicação de permissões

Ações permitidas	Explicação
`"athena:GetQueryExecution", "athena:GetQueryResults", "athena:GetWorkGroup", "athena:StartQueryExecution", "athena:StopQueryExecution"`	Permissões do Athena necessárias para executar consultas federadas.
`"athena:GetDataCatalog", "athena:GetQueryExecution," "athena:GetQueryResults", "athena:GetWorkGroup", "athena:StartQueryExecution", "athena:StopQueryExecution"`	Permissões do Athena necessárias para executar consultas de visualizações federadas . A ação `GetDataCatalog` é necessária para exibições.
`"lambda:InvokeFunction"`	Permite que as consultas chamem as funções do AWS Lambda para as funções do AWS Lambda especificadas no bloco `Resource`. Por exemplo, `arn:aws:lambda:*:` *`MyAWSAcctId`* `:function:` *`MyAthenaLambdaFunction`* , em que *`MyAthenaLambdaFunction`* especifica o nome da função do Lambda que será invocada. Como apresentado no exemplo, é possível especificar várias funções.
`"s3:AbortMultipartUpload", "s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket",`	É necessário ter as permissões `s3:ListBucket` e `s3:GetBucketLocation` para acessar o bucket de saída da consulta para

Ações permitidas	Explicação
`"s3:ListMultipartUploadParts", "s3:PutObject"`	entidades principais do IAM que executam `StartQueryExecution` . `s3:PutObject` , `s3:ListMultipartUploadParts` e `s3:AbortMultipartUpload` permitem gravar os resultados de consultas em todas as subpastas do bucket de resultados de consultas, conforme especificado pelo identificador de recurso `arn:aws:s3::: MyQueryResultsBucket /*`, com *MyQueryResultsBucket* indicando o bucket de resultados de consulta do Athena. Para obter mais informações, consulte Trabalhar com resultados de consultas, consultas recentes e arquivos de saída. `s3:GetObject` permite ler os resultados e o histórico de consultas para o recurso especificado como `arn:aws:s3::: MyQueryResultsBucket` , em que *MyQueryResultsBucket* é o bucket de resultados de consultas do Athena. `s3:GetObject` também permite ler o recurso especificado como `"arn:aws:s3::: MyLambdaSpillBucket /MyLambdaSpillPrefix *"`, em que *MyLambdaSpillPrefix* foi especificado na configuração da(s) função(ões) do Lambda invocada(s).

Example : permitir que uma entidade principal do IAM crie um conector de origem de dados

```
{
    "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/*"
        }
```

```
    ]
}
```

Explicação de permissões

Ações permitidas	Explicação
`"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",`	Permita a criação e gerenciamento de funções do Lambda listadas como recursos. No exemplo, um prefixo de nome foi usado no identificador do recurso `arn:aws:lambda:*:` *MyAWSAcctId* `:function:` *MyAthenaLambdaFunctionsPrefix* `*`, em que *MyAthenaLambdaFunctionsPrefix* é um prefixo compartilhado usado no nome do grupo de funções do Lambda de modo que elas não tenham de ser especificadas individualmente como recursos. É possível especificar um ou mais recursos de função do Lambda.
`"s3:GetObject"`	Permite a leitura de um bucket de que o AWS Serverless Application Repository precisa conforme especificado pelo identificador de recurso `arn:aws:s3:::awsserverlessrepo-changesets-` *1iiv3xa62ln3m* `/*`. Esse bucket pode ser específico para sua conta.
`"cloudformation:*"`	Permite a criação e o gerenciamento de pilhas especificadas do AWS CloudFormation pelo recurso *MyCFStackPrefix* . Essas pilhas e conjuntos de pilhas são como o AWS Serverles

Ações permitidas	Explicação
	s Application Repository implanta conectores e UDFs.
`"serverlessrepo:*"`	Permite pesquisar, exibir, publicar e atualizar aplicativos no AWS Serverless Application Repository, especificados pelo identificador de recurso `arn:aws:serverlessrepo:*:*:applications/*` .

Exemplo de políticas de permissões do IAM para permitir funções definidas pelo usuário (UDF) do Amazon Athena

Os exemplos de política de permissão neste tópico demonstram as ações permitidas necessárias e os recursos para os quais são permitidas. Examine essas políticas com atenção e modifique-as de acordo com seus requisitos antes de anexar políticas de permissões semelhantes às identidades do 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 : permitir que uma entidade principal do IAM execute e retorne consultas com uma instrução de UDF do Athena

A seguinte política de permissões baseada em identidade permite ações que um usuário ou outro principal do IAM precisa para executar consultas que usam instruções de UDF do Athena.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "athena:StartQueryExecution",
                "lambda:InvokeFunction",
                "athena:GetQueryResults",
```

```
                "s3:ListMultipartUploadParts",
                "athena:GetWorkGroup",
                "s3:PutObject",
                "s3:GetObject",
                "s3:AbortMultipartUpload",
                "athena:StopQueryExecution",
                "athena:GetQueryExecution",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:athena:*:MyAWSAcctId:workgroup/MyAthenaWorkGroup",
                "arn:aws:s3:::MyQueryResultsBucket/*",
                "arn:aws:lambda:*:MyAWSAcctId:function:OneAthenaLambdaFunction",
                "arn:aws:lambda:*:MyAWSAcctId:function:AnotherAthenaLambdaFunction"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "athena:ListWorkGroups",
            "Resource": "*"
        }
    ]
}
```

Explicação de permissões

Ações permitidas	Explicação
`"athena:StartQueryExecution", "athena:GetQueryResults", "athena:GetWorkGroup", "athena:StopQueryExecution", "athena:GetQueryExecution",`	Permissões do Athena necessárias para executar consultas no grupo de trabalho `MyAthenaWorkGroup` .
`"s3:PutObject", "s3:GetObject", "s3:AbortMultipartUpload"`	`s3:PutObject` e `s3:AbortMultipartUpload` permitem gravar os resultados de consultas em todas as subpastas do bucket de resultados de consultas, conforme especificado pelo identificador do recurso `arn:aws:s3::: MyQueryResultsBucket /*`, em

Ações permitidas	Explicação
	que *MyQueryResultsBucket* é o bucket de resultados de consultas do Athena. Para ter mais informações, consulte Trabalhar com resultados de consultas, consultas recentes e arquivos de saída. `s3:GetObject` permite ler os resultados e o histórico de consultas para o recurso especificado como `arn:aws:s3:::` *MyQueryResultsBucket*, em que *MyQueryResultsBucket* é o bucket de resultados de consultas do Athena. Para ter mais informações, consulte Trabalhar com resultados de consultas, consultas recentes e arquivos de saída. `s3:GetObject` também permite ler o recurso especificado como `"arn:aws:s3::: MyLambdaSpillBucket /MyLambdaSpillPrefix *"`, em que *MyLambdaSpillPrefix* foi especificado na configuração da(s) função(ões) do Lambda invocada(s).
`"lambda:InvokeFunction"`	Permite que as consultas chamem as funções do AWS Lambda especificadas no bloco `Resource`. Por exemplo, `arn:aws:lambda:*: MyAWSAcctId :function : MyAthenaLambdaFunction`, em que *MyAthenaLambdaFunction* especifica o nome da função do Lambda que será invocada. Várias funções podem ser especificadas conforme mostrado no exemplo.

Example : permitir que uma entidade principal do IAM crie uma UDF do 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/*"
        }
    ]
}
```

Explicação de permissões

Ações permitidas	Explicação
`"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",`	Permita a criação e gerenciamento de funções do Lambda listadas como recursos. No exemplo, um prefixo de nome foi usado no identificador do recurso `arn:aws:lambda:*:` *MyAWSAcctId* `:function:` *MyAthenaLambdaFunctionsPrefix* `*`, em que *MyAthenaLambdaFunctionsPrefix* é um prefixo compartilhado usado no nome do grupo de funções do Lambda de modo que elas não tenham de ser especificadas individualmente como recursos. É possível especificar um ou mais recursos de função do Lambda.
`"s3:GetObject"`	Permite a leitura de um bucket de que o AWS Serverless Application Repository precisa conforme especificado pelo identificador de recurso `arn:aws:s3:::awsserverlessrepo-changesets-` *1iiv3xa62ln3m* `/*`.
`"cloudformation:*"`	Permite a criação e o gerenciamento de pilhas especificadas do AWS CloudFormation pelo

Ações permitidas	Explicação
	recurso *MyCFStackPrefix* . Essas pilhas e conjuntos de pilhas são como o AWS Serverless Application Repository implanta conectores e UDFs.
`"serverlessrepo:*"`	Permite pesquisar, exibir, publicar e atualizar aplicativos no AWS Serverless Application Repository, especificados pelo identificador de recurso `arn:aws:serverlessrepo:*:*:applications/*` .

Permitir acesso para ML com o Athena

Os principais do IAM que executam consultas de ML do Athena devem ter permissão para executar a ação `sagemaker:invokeEndpoint` para os endpoints do SageMaker que eles usam. Inclua uma instrução de política semelhante à seguinte em políticas de permissões baseadas em identidade anexadas a identidades de usuário. Além disso, anexe Política gerenciada pela AWS: AmazonAthenaFullAccess, que concede acesso completo às ações do Athena, ou uma política em linha modificada que permita um subconjunto de ações.

Substitua `arn:aws:sagemaker:`*region*`:`*AWSAcctID*`:`*ModelEndpoint* no exemplo pelo ARN ou ARNs de endpoints do modelo a serem usados em consultas. Para obter mais informações, consulte Ações, recursos e chaves de condição do SageMaker na Referência de autorização do serviço.

```
{
            "Effect": "Allow",
            "Action": [
                "sagemaker:invokeEndpoint"
            ],
            "Resource": "arn:aws:sagemaker:us-west-2:123456789012:workteam/public-crowd/default"
}
```

Sempre que você usar as políticas do IAM, siga as práticas recomendadas do IAM. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM no Manual do usuário do IAM.

Permitir acesso federado à API do Athena

Esta seção aborda o acesso federado que permite que um usuário ou aplicação cliente na sua organização chame operações de API do Amazon Athena. Nesse caso, os usuários da sua organização não têm acesso direto ao Athena. Em vez disso, você gerencia as credenciais do usuário fora da AWS no Microsoft Active Directory. O Active Directory é compatível com SAML 2.0 (Security Assertion Markup Language 2.0).

Para autenticar usuários nesse cenário, use o driver JDBC ou ODBC com suporte a SAML.2.0 para acessar o Active Directory Federation Services (ADFS) 3.0 e permitir que uma aplicação cliente chame as operações de API do Athena.

Para obter mais informações sobre o suporte ao SAML 2.0 na AWS, consulte Sobre a federação baseada em SAML 2.0 no Guia do usuário do IAM.

> **Note**
>
> O acesso federado à API do Athena é permitido para um determinado tipo de provedor de identidade (IdP), o Active Directory Federation Service (ADFS 3.0), que faz parte do Windows Server. O acesso federado não é compatível com o recurso de propagação de identidade confiável do Centro de Identidade do IAM. O acesso é estabelecido por meio das versões dos drivers JDBC ou ODBC que oferecem suporte ao SAML 2.0. Para obter informações, consulte Conectar-se ao Amazon Athena com JDBC e Conectar-se ao Amazon Athena com ODBC.

Tópicos

- Antes de começar
- Diagrama de arquitetura
- Procedimento: acesso federado baseado em SAML à API do Athena

Antes de começar

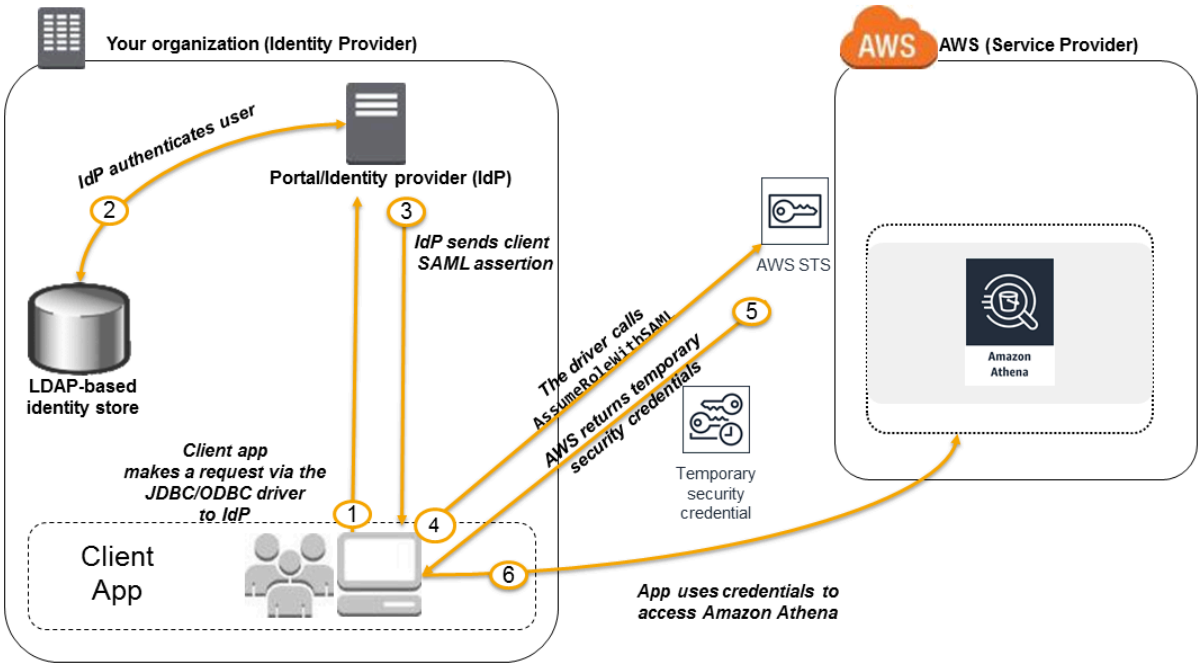
Antes de começar, conclua os pré-requisitos a seguir:

- Dentro da sua organização, instale e configure o ADFS 3.0 como seu IdP.
- Instale e configure as versões mais recentes disponíveis dos drivers JDBC ou ODBC nos clientes usados para acessar o Athena. O driver deve incluir suporte para acesso federado compatível

com o SAML 2.0. Para obter informações, consulte Conectar-se ao Amazon Athena com JDBC e Conectar-se ao Amazon Athena com ODBC.

Diagrama de arquitetura

O diagrama a seguir ilustra esse processo.



1. Um usuário na sua organização usa um aplicativo cliente com o driver JDBC ou ODBC para solicitar a autenticação do IdP da sua organização. O IdP é ADFS 3.0.
2. O IdP autentica o usuário no Active Directory, ou seja, o armazenamento de identidades da sua organização.
3. O IdP cria uma declaração do SAML com informações sobre o usuário e a envia para o aplicativo cliente por meio do driver JDBC ou ODBC.
4. O driver JDBC ODBC chama a operação da API AWS Security Token Service AssumeRoleWithSAML e passa os seguintes parâmetros:
   - O ARN do provedor SAML
   - O ARN da função a ser assumida
   - A declaração do SAML do IdP

Para obter mais informações, consulte AssumeRoleWithSAML na Referência de API do AWS Security Token Service.

5. A resposta da API para o aplicativo cliente por meio do driver JDBC ou ODBC inclui credenciais de segurança temporárias.
6. A aplicação cliente usa as credenciais de segurança temporárias para chamar as operações de API do Athena, permitindo que seus usuários acessem as operações de API do Athena.

Procedimento: acesso federado baseado em SAML à API do Athena

Esse procedimento estabelece uma confiança entre o IdP da sua organização e sua conta da AWS para permitir acesso federado baseado em SAML à operação de API do Amazon Athena.

Para habilitar o acesso federado à API do Athena:

1. Em sua organização, registre a AWS como um provedor de serviços (SP) em seu IdP. Esse processo é conhecido como confiança da parte dependente. Para obter mais informações, consulte Configuração do IdP SAML 2.0 com objeto de confiança de terceira parte confiável no Guia do usuário do IAM. Como parte dessa tarefa, execute estas etapas:
   a. Obtenha o documento de metadados do SAML de amostra deste URL: https://signin.aws.amazon.com/static/saml-metadata.xml.
   b. No IdP da sua organização (ADFS), gere um arquivo XML de metadados equivalente que descreva o seu IdP como um provedor de identidade para a AWS. O arquivo de metadados deve incluir o nome do emissor, uma data de criação, uma data de expiração e chaves que a AWS usa para validar as respostas de autenticação (declarações) da sua organização.
2. No console do IAM, crie uma entidade de provedor de identidade do SAML. Para obter mais informações, consulte Criação de provedores de identidade SAML do IAM no Guia do usuário do IAM. Como parte dessa etapa, realize as seguintes ações:
   a. Abra o console IAM em https://console.aws.amazon.com/iam/.
   b. Faça upload do documento de metadados SAML produzido pelo IdP (ADFS) na Etapa 1 deste procedimento.
3. No console do IAM, crie uma ou mais funções do IAM para seu IdP. Para obter mais informações, consulte Criar uma função para um provedor de identidade de terceiros (federação) no Guia do usuário do IAM. Como parte dessa etapa, realize as seguintes ações:

- Na política de permissões da função, estabeleça o que os usuários da sua organização têm permissão para fazer na AWS.
- Na política de confiança da função, defina como principal a entidade do provedor do SAML que você criou na Etapa 2 deste procedimento.

Isso estabelece uma relação de confiança entre sua organização e a AWS.

4. No IdP da sua organização (ADFS), defina declarações que mapeiam usuários ou grupos na sua organização para as funções do IAM. O mapeamento de usuários e grupos para as funções do IAM também é conhecido como regra de declaração. Observe que usuários e grupos diferentes na sua organização podem ser mapeados para funções do IAM distintas.

   Para obter informações sobre como configurar o mapeamento no ADFS, consulte a publicação do blog: Habilitação de federação na AWS usando Windows Active Directory, ADFS e SAML 2.0.

5. Instale e configure o driver JDBC ou ODBC com o suporte a SAML 2.0. Para obter informações, consulte Conectar-se ao Amazon Athena com JDBC e Conectar-se ao Amazon Athena com ODBC.

6. Especifique a string de conexão do seu aplicativo para o driver JDBC ou ODBC. Para obter informações sobre a string de conexão que sua aplicação deve usar, consulte o tópico "Using the Active Directory Federation Services (ADFS) Credentials Provider" (“Usar o provedor de credenciais do Active Directory Federation Services (ADFS)”) no Guia de instalação e configuração do driver JDBC ou um tópico similar no Guia de instalação e configuração do driver ODBC disponível como downloads de PDF dos tópicos Conectar-se ao Amazon Athena com JDBC e Conectar-se ao Amazon Athena com ODBC.

   A seguir, veja um resumo de alto nível de como configurar a string de conexão para os drivers:

   1. Em `AwsCredentialsProviderClass configuration`, defina o `com.simba.athena.iamsupport.plugin.AdfsCredentialsProvider` para indicar que você deseja usar a autenticação baseada em SAML 2.0 por meio do IdP do ADFS.
   2. Em `idp_host`, forneça o nome de host do servidor do IdP do ADFS.
   3. Em `idp_port`, informe o número da porta que o IdP do ADFS identifica a solicitação de declaração do SAML.
   4. Em `UID` e `PWD`, forneça as credenciais de usuário do domínio do AD. Ao usar o driver no Windows, se `UID` e `PWD` não forem fornecidos, o driver tentará obter as credenciais de usuário do usuário conectado no computador com Windows.

5. Opcionalmente, defina `ssl_insecure` para `true`. Nesse caso, o driver não verifica a autenticidade do certificado SSL para o servidor do IdP do ADFS. Será necessário definir essa opção como `true` se o certificado SSL do IdP do ADFS não tiver sido configurado como confiável pelo driver.
6. Para habilitar o mapeamento de um usuário ou grupo de domínio do Active Directory para uma ou mais funções do IAM (como mencionado na etapa 4 deste procedimento), em `preferred_role`, na conexão do JDBC ou do ODBC, especifique a função do IAM (ARN) a ser assumida. Especificar o `preferred_role` é opcional. Ele é útil se a função não for a primeira a ser listada na regra de declaração.

As seguintes ações ocorrem como resultado desse procedimento:

1. O driver JDBC ou ODBC chama a API AWS STS AssumeRoleWithSAML e transmite as declarações, conforme mostrado na etapa 4 do diagrama de arquitetura.
2. A AWS certifica-se de que a solicitação para assumir a função vem do IdP referenciado na entidade do provedor do SAML.
3. Se a solicitação for bem-sucedida, a operação da API AssumeRoleWithSAML do AWS STS retornará um conjunto de credenciais de segurança temporárias, que a aplicação cliente usa para fazer solicitações assinadas ao Athena.

   Sua aplicação agora tem informações sobre o usuário atual e pode acessar o Athena de forma programática.

Registro e monitoramento no Athena

Para detectar incidentes, receber alertas quando há incidentes e responder a eles, use estas opções com o Amazon Athena:

- Monitorar o Athena com AWS CloudTrail: o AWS CloudTrail oferece um registro das ações executadas por um usuário, uma função ou um AWS service (Serviço da AWS) no Athena. Ele captura as chamadas do console do Athena e as chamadas de código para as operações de API do Athena como eventos. Dessa forma, você pode determinar a solicitação feita ao Athena, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e outros detalhes. Para ter mais informações, consulte Registro de chamadas de API do Amazon Athena com o AWS CloudTrail.

Você também pode usar o Athena para consultar os arquivos de log do CloudTrail para o Athena e também para outros Serviços da AWS. Para ter mais informações, consulte Consultar os logs do AWS CloudTrail.

- Monitorar o uso do Athena com o CloudTrail e o Amazon QuickSight: o Amazon QuickSight é um serviço de business intelligence totalmente gerenciado e com tecnologia de nuvem que permite criar painéis interativos que sua organização pode acessar de qualquer dispositivo. Para obter um exemplo de solução que usa o CloudTrail e o Amazon QuickSight para monitorar o uso do Athena, consulte a postagem no blog de big data da AWS How Realtor.com monitors Amazon Athena usage with AWS CloudTrail and Amazon QuickSight.
- Usar o Amazon EventBridge com o Athena: a Amazon EventBridge oferece uma transmissão quase em tempo real dos eventos do sistema que descrevem as alterações nos recursos da AWS. O EventBridge reconhece essas alterações operacionais logo que acontecem, responde a elas e executa a ação corretiva conforme necessário, enviando mensagens para responder ao ambiente, ativando funções, fazendo alterações e capturando informações de estado. Os eventos são emitidos com base no melhor esforço. Para obter mais informações, consulte Começar a usar o Amazon EventBridge no Manual do usuário do Amazon EventBridge.
- Usar grupos de trabalho para separar usuários, equipes, aplicações ou workloads, definir os limites e controlar os custos de consulta: você pode exibir as métricas relacionadas à consulta no Amazon CloudWatch, controlar os custos de consulta com a configuração de limites para a quantidade de dados verificados, criar limites e acionar ações, como alarmes do Amazon SNS, quando esses limites são violados. Para um procedimento de alto nível, consulte Configurar grupos de trabalho. Use as permissões do IAM no nível do recurso para controlar o acesso a um determinado grupo de trabalho. Para ter mais informações, consulte Usar grupos de trabalho para executar consultas e Controlar custos e monitorar consultas com métricas e eventos do CloudWatch.

Tópicos

- Registro de chamadas de API do Amazon Athena com o AWS CloudTrail

Registro de chamadas de API do Amazon Athena com o AWS CloudTrail

O Athena é integrado ao AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, uma função ou um AWS service (Serviço da AWS) no Athena.

O CloudTrail captura todas as chamadas de API para o Athena como eventos. As chamadas capturadas incluem as chamadas do console do Athena e as chamadas de código para as

operações de API do Athena. Se você criar uma trilha, poderá habilitar a entrega contínua de eventos do CloudTrail para um bucket do Amazon S3, incluindo os eventos do Athena. Se você não configurar uma trilha, ainda poderá visualizar os eventos mais recentes no console do CloudTrail em Histórico de eventos.

Usando as informações coletadas pelo CloudTrail, é possível determinar a solicitação feita para o Athena, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e outros detalhes.

Para saber mais sobre o CloudTrail, consulte o Guia do usuário do AWS CloudTrail.

Você pode usar o Athena para consultar arquivos de log do CloudTrail do próprio Athena e de outros Serviços da AWS. Para obter mais informações, consulte Consultar os logs do AWS CloudTrail, Hive JSON SerDe e a publicação no blog de big data da AWS: Usar as instruções CTAS com o Amazon Athena para reduzir custos e melhorar a performance (em inglês), que usa o CloudTrail para fornecer insights sobre o uso do Athena.

Informações do Athena no CloudTrail

O CloudTrail é habilitado em sua conta da Amazon Web Services quando ela é criada. Quando ocorre uma atividade no Athena, ela é registrada em um evento do CloudTrail com outros eventos de serviços da AWS no Event history (Histórico de eventos). Você pode visualizar, pesquisar e baixar os eventos recentes em sua conta da Amazon Web Services. Para obter mais informações, consulte Viewing events with CloudTrail event history.

Para obter um registro contínuo dos eventos na sua conta da Amazon Web Services, incluindo os eventos do Athena, crie uma trilha. Uma trilha permite que o CloudTrail entregue arquivos de log a um bucket do Amazon S3. Por padrão, quando você cria uma trilha no console, ela é aplicada a todas as Regiões da AWS. A trilha registra em log eventos de todas as regiões na partição da AWS e entrega os arquivos de log para o bucket do Amazon S3 especificado por você. Além disso, você pode configurar outros Serviços da AWS para analisar mais ainda mais e agir com base nos dados de eventos coletados nos logs do CloudTrail. Para obter mais informações, consulte as informações a seguir:

- Visão geral da criação de uma trilha
- Serviços e integrações compatíveis com o CloudTrail
- Configuração notificações do Amazon SNS para o CloudTrail
- Como receber arquivos de log do CloudTrail de várias regiões e Como receber arquivos de log do CloudTrail de várias contas

Todas as ações do Athena são registradas pelo CloudTrail e documentadas na Referência de API do Amazon Athena. Por exemplo, as chamadas às ações StartQueryExecution e GetQueryResults geram entradas nos arquivos de log do CloudTrail.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar:

- Se a solicitação foi feita com credenciais de usuário raiz ou usuário do IAM AWS Identity and Access Management
- Se a solicitação foi feita com credenciais de segurança temporárias de uma função ou de um usuário federado.
- Se a solicitação foi feita por outro AWS service (Serviço da AWS).

Para mais informações, consulte Elemento userIdentity do CloudTrail.

Noções básicas sobre as entradas de arquivos de log do Athena

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log a um bucket do Amazon S3 especificado. Os arquivos de log CloudTrail contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer fonte, e inclui informações sobre a ação solicitada, data e hora da ação, parâmetros de solicitação e assim por diante. Os arquivos de log do CloudTrail não são um rastreamento de pilha ordenada de chamadas de API pública, portanto não são exibidos em uma ordem específica.

> **Note**
>
> Para evitar a divulgação não intencional de informações sensíveis, a entrada `queryString` nos logs `StartQueryExecution` e `CreateNamedQuery` tem um valor de `***OMITTED***`. Isso faz parte do design. Para acessar a string de consulta real, você pode usar a API GetQueryExecution do Athena e transmitir o valor de `responseElements.queryExecutionId` do log do CloudTrail.

Os seguintes exemplos demonstram entradas de log do CloudTrail para:

- StartQueryExecution (bem-sucedido)
- StartQueryExecution (falha)
- CreateNamedQuery

StartQueryExecution (êxito)


```
{
 "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 (falha)


```
{
 "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"
},
```


Validação de compatibilidade do Amazon Athena

Os auditores externos avaliam a segurança e a compatibilidade do Amazon Athena como parte de vários programas de compatibilidade da AWS. Isso inclui SOC, PCI, FedRAMP e outros.

Para obter uma lista dos Serviços da AWS no escopo de programas de conformidade específicos, consulte Serviços da AWS no escopo por programa de conformidade. Para obter informações gerais, consulte Programas de conformidade da AWS.

É possível baixar relatórios de auditoria de terceiros usando o AWS Artifact. Para obter mais informações, consulte Baixar os relatórios no AWS Artifact.

Sua responsabilidade com relação à compatibilidade ao usar o Athena é determinada pela confidencialidade dos dados, pelos objetivos de compatibilidade da empresa e pelos regulamentos e leis aplicáveis. A AWS oferece os seguintes recursos para ajudar na compatibilidade:

- Guias de início rápido de segurança e conformidade: esses guias de implantação abordam as considerações de arquitetura e fornecem etapas para a implantação de ambientes de linha de base concentrados em compatibilidade e segurança na AWS.
- Arquitetura para segurança e conformidade com HIPAA na Amazon Web Services: este whitepaper descreve como as empresas podem usar a AWS para criar aplicações em conformidade com os padrões HIPAA.
- Recursos de compatibilidade da AWS: essa coleção de manuais e guias pode ser aplicável a seu setor e local.
- AWS Config: esse AWS service (Serviço da AWS) avalia até que ponto suas configurações de recursos atendem adequadamente às práticas internas e às diretrizes e regulamentações do setor.

- AWS Security Hub: esse AWS service (Serviço da AWS) fornece uma visão abrangente do estado de sua segurança na AWS que ajuda você a conferir sua conformidade com padrões e práticas recomendadas de segurança do setor.

Resiliência no Athena

A infraestrutura global da AWS se baseia em Regiões da AWS e zonas de disponibilidade. A Regiões da AWS oferece várias zonas de disponibilidade separadas e isoladas fisicamente que são conectadas com baixa latência, altas taxas de throughput e em redes altamente redundantes. Com as Zonas de Disponibilidade, você pode projetar e operar aplicativos e bancos de dados que executam o failover automaticamente entre as Zonas de Disponibilidade sem interrupção. As zonas de disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de datacenter tradicionais.

Para obter mais informações sobre Regiões da AWS e zonas de disponibilidade, consulte Infraestrutura global da AWS.

Além da infraestrutura global da AWS, o Athena conta com vários recursos para ajudar a atender às suas necessidades de resiliência e backup de dados.

O Athena é sem servidor e, portanto, não há infraestrutura para configurar ou gerenciar. O Athena é altamente disponível e executa consultas usando recursos de computação em várias zonas de disponibilidade, encaminhando as consultas de forma automática e adequada se uma zona de disponibilidade específica estiver inacessível. O Athena usa o Amazon S3 como seu armazenamento de dados subjacente, o que torna os dados altamente disponíveis e duráveis. O Amazon S3 dispõe de uma infraestrutura durável para armazenar dados importantes e foi criado para oferecer durabilidade dos objetos de 99,999999999%. Seus dados são armazenados com redundância em várias instalações e diversos dispositivos em cada instalação.

Segurança da infraestrutura no Athena

Como um serviço gerenciado, o Amazon Athena é protegido pela segurança da rede global da AWS. Para obter informações sobre serviços de segurança da AWS e como a AWS protege a infraestrutura, consulte Segurança na Nuvem AWS. Para projetar seu ambiente da AWS usando as práticas recomendadas de segurança de infraestrutura, consulte Proteção de infraestrutura em Security Pillar: AWS Well-Architected Framework.

Você usa as chamadas de API publicadas pela AWS para acessar o Athena por meio da rede. Os clientes devem oferecer suporte para:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Conjuntos de criptografia com sigilo de encaminhamento perfeito (perfect forward secrecy, ou PFS) como DHE (Ephemeral Diffie-Hellman, ou Efêmero Diffie-Hellman) ou ECDHE (Ephemeral Elliptic Curve Diffie-Hellman, ou Curva elíptica efêmera Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, as solicitações devem ser assinadas utilizando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou é possível usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Use as políticas do IAM para restringir o acesso às operações do Athena. Sempre que você usar as políticas do IAM, siga as práticas recomendadas do IAM. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM no Manual do usuário do IAM.

As políticas gerenciadas do Athena são fáceis de usar e atualizadas automaticamente com as ações necessárias de acordo com o desenvolvimento do serviço. As políticas em linha e gerenciadas pelo cliente podem ser ajustadas especificando nelas ações mais granulares do Athena. Conceda o acesso adequado ao local dos dados do Amazon S3. Para obter informações detalhadas e conhecer cenários para a concessão de acesso ao Amazon S3, consulte Demonstrações de exemplo: gerenciar o acesso no Guia do desenvolvedor do Amazon Simple Storage Service. Para obter mais informações e um exemplo das ações do Amazon S3 que devem ser permitidas, consulte o exemplo de política de bucket em Acesso entre contas.

Tópicos

- Conectar-se ao Amazon Athena usando um endpoint da VPC de interface

Conectar-se ao Amazon Athena usando um endpoint da VPC de interface

Você pode melhorar a postura de segurança da sua VPC usando um endpoint da VPC de interface (AWS PrivateLink) e um endpoint da VPC do AWS Glue em sua nuvem privada virtual (VPC). Um endpoint da VPC de interface melhora a segurança ao oferecer controle sobre quais destinos podem ser alcançados de dentro da sua VPC. Cada endpoint da VPC é representado por uma ou mais interfaces de rede elástica (ENIs) com endereços IP privados nas sub-redes da VPC.

O endpoint da VPC de interface conecta sua VPC diretamente ao Athena sem um gateway da Internet, um dispositivo NAT, uma conexão VPN ou uma conexão AWS Direct Connect. Não é

necessário que as instâncias na sua VPC tenham endereços IP públicos para se comunicarem com a API do Athena.

Para usar o Athena por meio da VPC, você deve se conectar de uma instância que esteja dentro da VPC ou conectar sua rede privada à VPC usando o Amazon Virtual Private Network (VPN) ou uma AWS Direct Connect. Para obter informações sobre o Amazon VPN, consulte Conexões VPN no Guia do usuário do Amazon Virtual Private Cloud. Para obter informações sobre o AWS Direct Connect, consulte Creating a connection (Criação de uma conexão) no Guia do usuário do AWS Direct Connect.

O Athena aceita endpoints da VPC em todas as Regiões da AWS onde o Amazon VPC e o Athena estão disponíveis.

É possível criar um endpoint da VPC de interface para se conectar ao Athena usando os comandos do AWS Management Console ou da AWS Command Line Interface (AWS CLI). Para obter mais informações, consulte Creating an interface endpoint (Criação de um endpoint de interface).

Depois de criar um endpoint da VPC de interface, se você habilitar nomes de host DNS privados para o endpoint, o endpoint padrão do Athena (https://athena.*Region*.amazonaws.com) será resolvido para seu endpoint da VPC.

Se você não habilitar nomes de host DNS privados, o Amazon VPC fornecerá um nome de endpoint DNS que poderá ser usado no seguinte formato:

```
VPC_Endpoint_ID.athena.Region.vpce.amazonaws.com
```

Para mais informações, consulte VPC endpoints de interface (AWS PrivateLink) no Guia do usuário da Amazon VPC.

O Athena permite fazer chamadas para todas as ações de API na sua VPC.

Criar uma política de endpoint da VPC para o Athena

Você pode criar uma política de endpoints da Amazon VPC para o Athena para especificar restrições como:

- Entidade principal: a entidade principal que pode executar ações.
- Ações: as ações que podem ser executadas.
- Recursos: os recursos sobre os quais as ações podem ser realizadas.

- Somente identidades confiáveis: use a condição `aws:PrincipalOrgId` para restringir o acesso somente às credenciais que fazem parte de sua organização da AWS. Isso pode ajudar a impedir o acesso de entidades principais não intencionais.
- Somente recursos confiáveis: use a condição `aws:ResourceOrgId` para impedir o acesso a recursos não intencionais.
- Somente identidades e recursos confiáveis: crie uma política combinada para um endpoint da VPC que ajude a impedir o acesso a recursos e entidades principais não intencionais.

Para obter mais informações, consulte Controlar acesso a serviços com endpoints da VPC no Guia do usuário da Amazon VPC e no Apêndice 2: exemplos de políticas de endpoints da VPC no whitepaper da AWS Criar um perímetro de dados na AWS.

Example Política de endpoint da VPC

O exemplo a seguir permite solicitações de identidades de organizações para recursos de organizações, e permite solicitações de entidades principais de serviços da AWS.

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

Sempre que você usar as políticas do IAM, siga as práticas recomendadas do IAM. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM no Manual do usuário do IAM.

Sub-redes compartilhadas

Você não pode criar, descrever, modificar ou excluir endpoints da VPC em sub-redes que são compartilhadas com você. No entanto, você pode usar os endpoints da VPC em sub-redes que são compartilhadas com você. Para obter informações sobre o compartilhamento da VPC, consulte Compartilhar sua VPC com outras contas no Guia do usuário da Amazon VPC.

Análise de vulnerabilidade e configuração no Athena

O Athena é sem servidor, portanto, não há infraestrutura para configurar ou gerenciar. A AWS processa as tarefas básicas de segurança, como aplicação de patches a bancos de dados e sistemas operacionais (SO) convidados, configuração de firewalls e recuperação de desastres. Esses procedimentos foram revisados e certificados por terceiros certificados. Para obter mais detalhes, consulte os seguintes recursos da AWS:

- Modelo de responsabilidade compartilhada
- Práticas recomendadas de segurança, identidade e conformidade

Usar o Athena para consultar dados registrados com o AWS Lake Formation

O AWS Lake Formation permite definir e impor as políticas de acesso no nível do banco de dados, da tabela e da coluna ao usar as consultas do Athena para ler os dados armazenados no Amazon S3. O Lake Formation oferece uma camada de autorização e governança dos dados armazenados no Amazon S3. É possível usar uma hierarquia de permissões no Lake Formation para conceder

ou revogar permissões de leitura de objetos do catálogo de dados, como bancos de dados, tabelas e colunas. O Lake Formation simplifica o gerenciamento de permissões e permite implementar um Fine-Grained Access Control (FGAC – Controle de acesso refinado) para seus dados.

É possível usar o Athena para consultar os dados tanto registrados quanto não registrados no Lake Formation.

As permissões do Lake Formation são aplicadas ao usar o Athena para consultar dados de origens dos locais do Amazon S3 registrados no Lake Formation. As permissões do Lake Formation também são aplicadas ao criar bancos de dados e tabelas que apontam para os locais de dados registrados do Amazon S3. Para usar o Athena com dados registrados no Lake Formation, o Athena deve ser configurado para usar o AWS Glue Data Catalog.

As permissões do Lake Formation não se aplicam à gravação de objetos no Amazon S3 nem à consulta de dados armazenados no Amazon S3 ou de metadados que não estão registrados no Lake Formation. Para dados de origem no Amazon S3 e metadados que não estão registrados no Lake Formation, o acesso é determinado pelas políticas de permissões do IAM para ações do Amazon S3 e do AWS Glue. Os locais de resultados de consulta do Athena no Amazon S3 não podem ser registrados no Lake Formation, e as políticas de permissões do IAM no Amazon S3 controlam o acesso. Além disso, as permissões do Lake Formation não se aplicam ao histórico de consultas do Athena. É possível usar grupos de trabalho do Athena para controlar o acesso ao histórico de consultas.

Para obter mais informações sobre o Lake Formation, consulte Perguntas Frequentes do Lake Formation e o Guia do desenvolvedor do AWS Lake Formation.

Tópicos

- Como o Athena acessa os dados registrados no Lake Formation
- Considerações e limitações ao usar o Athena para consultar dados registrados no Lake Formation
- Gerenciar permissões de usuário do Lake Formation e do Athena
- Aplicar permissões do Lake Formation a bancos de dados e tabelas existentes
- Usar o Lake Formation e os drivers JDBC e ODBC do Athena para acesso federado ao Athena

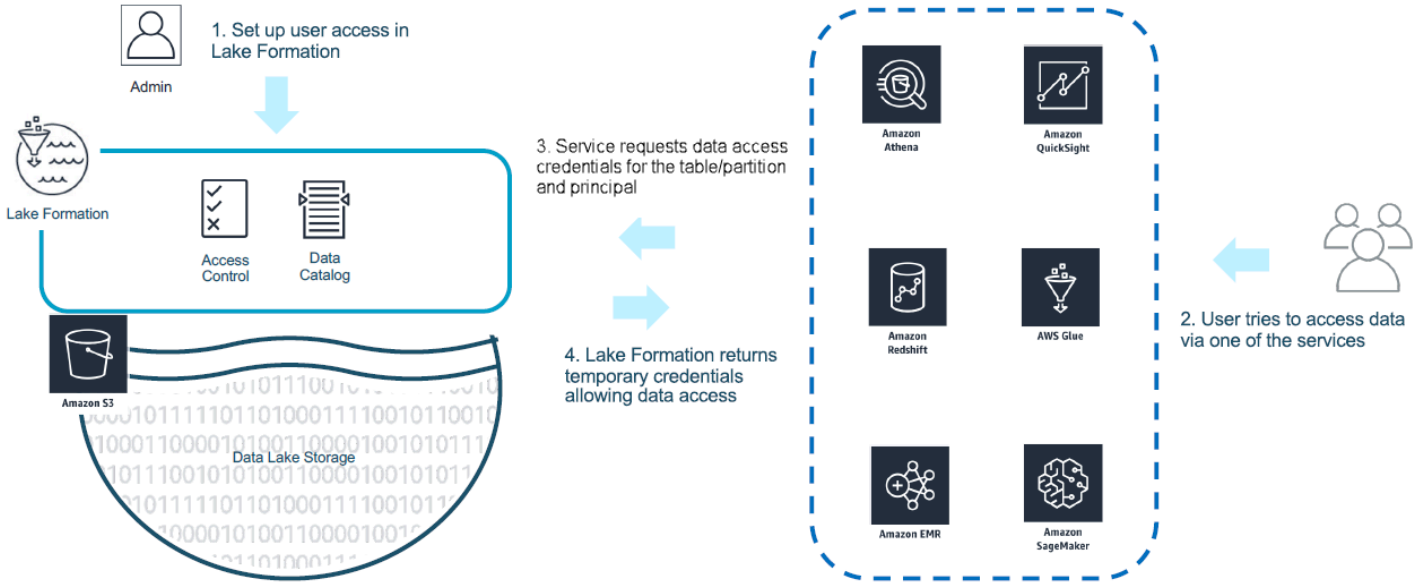
Como o Athena acessa os dados registrados no Lake Formation

O fluxo de trabalho de acesso descrito nesta seção se aplica somente à execução de consultas do Athena em locais e objetos de metadados do Amazon S3 registrados no Lake Formation. Para

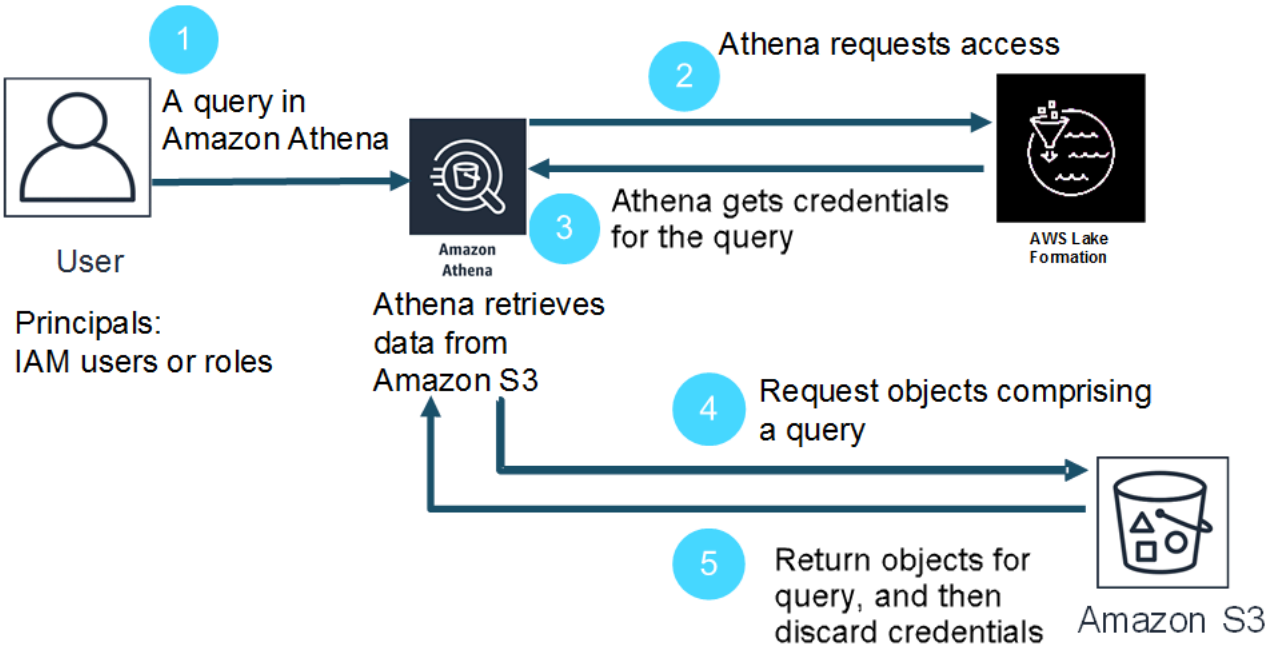
obter mais informações, consulte Registering a data lake (Registrar um data lake) no Guia do desenvolvedor do AWS Lake Formation. Além de registrar os dados, o administrador do Lake Formation aplica as respectivas permissões, que concedem ou revogam o acesso aos metadados no Catálogo de dados e no local dos dados no Amazon S3. Para obter mais informações, consulte Security and access control to metadata and data (Controle de segurança e acesso a metadados e dados) no Guia do desenvolvedor do AWS Lake Formation.

Cada vez que um principal do Athena (usuário, grupo ou função) executa uma consulta nos dados registrados usando o Lake Formation, o Lake Formation verifica se o principal tem as devidas permissões do Lake Formation para o banco de dados, a tabela e o local do Amazon S3, conforme apropriado para a consulta. Se o principal tiver acesso, o Lake Formation venderá credenciais temporárias para o Athena e a consulta será executada.

O diagrama a seguir ilustra o fluxo descrito acima.



O seguinte diagrama mostra como a venda de credenciais funciona no Athena por consulta em um exemplo de consulta SELECT em uma tabela com um local do Amazon S3 registrado no Lake Formation:



1. Um principal executa a consulta SELECT no Athena.
2. O Athena analisa a consulta e verifica as permissões do Lake Formation para ver se o principal recebeu acesso à tabela e às suas colunas.
3. Se o principal tiver acesso, o Athena solicitará as credenciais do Lake Formation. Se o principal não tiver acesso, o Athena emitirá um erro de acesso negado.
4. O Lake Formation emite as credenciais ao Athena para usar na leitura dos dados do Amazon S3 junto com a lista de colunas permitidas.
5. O Athena usa as credenciais temporárias do Lake Formation para consultar os dados do Amazon S3. Após a conclusão da consulta, o Athena descarta as credenciais.

Considerações e limitações ao usar o Athena para consultar dados registrados no Lake Formation

Considere as questões a seguir ao usar o Athena para consultar dados registrados no Lake Formation. Para obter informações adicionais, consulte Problemas conhecidos do AWS Lake Formation no Guia do desenvolvedor do AWS Lake Formation.

Condições e limitações

- Metadados de coluna visíveis para usuários não autorizados em algumas circunstâncias com Avro e SerDe personalizado

- Trabalhar com permissões do Lake Formation para visualizações
- Controle de acesso detalhado do Lake Formation e grupos de trabalho do Athena
- Local dos resultados de consultas do Athena no Amazon S3 não registrado no Lake Formation
- Usar grupos de trabalho do Athena para limitar o acesso ao histórico de consultas
- Acesso ao catálogo de dados entre contas
- Locais do Amazon S3 criptografados por CSE-KMS e registrados no Lake Formation
- Locais de dados particionados registrados no Lake Formation devem estar em subdiretórios de tabela
- Consultas Create Table As Select (CTAS) exigem permissões de gravação do Amazon S3
- A permissão DESCRIBE é necessária no banco de dados padrão

Metadados de coluna visíveis para usuários não autorizados em algumas circunstâncias com Avro e SerDe personalizado

A autorização no nível da coluna do Lake Formation impede que os usuários acessem os dados nas colunas para as quais eles não tenham permissões do Lake Formation. No entanto, em determinadas situações, os usuários podem acessar metadados que descrevem todas as colunas na tabela, incluindo as colunas para as quais eles não têm permissões para os dados.

Isso ocorre quando os metadados da coluna são armazenados nas propriedades das tabelas usando o formato de armazenamento Apache Avro ou um serializador/desserializador (SerDe) personalizado no qual o esquema da tabela é definido nas propriedades da tabela juntamente com a definição do SerDe. Ao usar o Athena com o Lake Formation, recomendamos revisar o conteúdo das propriedades da tabela que você registrou no Lake Formation e, sempre que possível, limitar as informações armazenadas nas propriedades da tabela para impedir que metadados confidenciais fiquem visíveis aos usuários.

Trabalhar com permissões do Lake Formation para visualizações

Para dados registrados no Lake Formation, um usuário do Athena poderá criar um `VIEW` somente se tiver permissões do Lake Formation para as tabelas, as colunas e os locais de dados de origem do Amazon S3 nos quais a `VIEW` se baseia. Depois que uma `VIEW` é criada no Athena, as permissões do Lake Formation podem ser aplicadas à `VIEW`. As permissões no nível da coluna não estão disponíveis para uma `VIEW`. Os usuários com permissões do Lake Formation para uma `VIEW`, mas sem permissões para a tabela e as colunas nas quais a visualização foi baseada, não podem

usar a `VIEW` para consultar dados. No entanto, os usuários com essa combinação de permissões podem usar instruções como `DESCRIBE VIEW`, `SHOW CREATE VIEW` e `SHOW COLUMNS` para ver metadados `VIEW`. Por esse motivo, alinhe as permissões do Lake Formation a cada `VIEW` com as permissões subjacentes da tabela. Filtros de células definidos em uma tabela não se aplicam a um `VIEW` para essa tabela. Os nomes de links de recursos devem ter o mesmo nome que o recurso na conta de origem. Há limitações adicionais ao se trabalhar com visualizações em uma configuração entre contas. Para obter mais informações sobre como configurar permissões para visualizações compartilhadas entre contas, consulte Acesso ao catálogo de dados entre contas.

Controle de acesso detalhado do Lake Formation e grupos de trabalho do Athena

Os usuários do mesmo grupo de trabalho do Athena podem visualizar os dados que o controle de acesso detalhado do Lake Formation configurou para serem acessíveis ao grupo de trabalho. Para obter mais informações sobre como usar o controle de acesso detalhado no Lake Formation, consulte Gerenciar o controle de acesso detalhado usando o AWS Lake Formation no AWS Big Data Blog.

Local dos resultados de consultas do Athena no Amazon S3 não registrado no Lake Formation

O local dos resultados de consultas do Athena no Amazon S3 não pode ser registrado no Lake Formation. As permissões do Lake Formation não limitam o acesso a esses locais. A menos que você limite o acesso, os usuários do Athena podem acessar arquivos de resultados de consulta e metadados mesmo sem permissões do Lake Formation para os dados. Para evitar isso, recomendamos que você use grupos de trabalho para especificar o local dos resultados das consultas e alinhe a associação do grupo de trabalho com as permissões do Lake Formation. Depois disso, você pode usar as políticas de permissões do IAM para limitar o acesso aos locais de resultados de consultas. Para obter mais informações sobre resultados de consultas, veja Trabalhar com resultados de consultas, consultas recentes e arquivos de saída.

Usar grupos de trabalho do Athena para limitar o acesso ao histórico de consultas

O histórico de consultas do Athena expõe uma lista de consultas salvas e strings de consulta completas. A menos que você use grupos de trabalho para separar o acesso a históricos de consultas, os usuários do Athena que não estão autorizados a consultar os dados no Lake Formation poderão visualizar as strings de consultas executadas nesses dados, incluindo nomes de coluna, critérios de seleção e etc. Recomendamos que você use grupos de trabalho para separar históricos de consultas e alinhar a associação do grupo de trabalho do Athena com as permissões do Lake Formation para limitar o acesso. Para ter mais informações, consulte Usar grupos de trabalho para controlar o acesso a consultas e os custos.

Acesso ao catálogo de dados entre contas

Para acessar um catálogo de dados em outra conta, você pode usar o recurso do AWS Glue entre contas do Athena ou configurar o acesso entre contas no Lake Formation.

Acesso ao catálogo de dados entre contas do Athena

Você pode usar o recurso de catálogo do AWS Glue entre contas do Athena para registrar o catálogo em sua conta. Esse recurso está disponível somente no mecanismo Athena versão 2 e posteriores e está limitado ao uso entre contas na mesma região. Para ter mais informações, consulte Registrar um AWS Glue Data Catalog de outra conta.

Se o catálogo de dados a ser compartilhado tiver uma política de recursos configurada no AWS Glue, ele deverá ser atualizado para permitir acesso ao AWS Resource Access Manager e conceder permissões à conta B para usar o catálogo de dados da conta A, como no exemplo a seguir.

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

```
}
```

Para ter mais informações, consulte Acesso aos catálogos de dados do AWS Glue entre contas.

Configurar o acesso entre contas no Lake Formation

O AWS Lake Formation permite que você use uma única conta para gerenciar um catálogo de dados central. Você pode usar esse recurso para implementar o acesso entre contas a metadados do catálogo de dados e aos dados subjacentes. Por exemplo, uma conta de proprietário pode conceder à outra conta (destinatário) a permissão `SELECT` em uma tabela.

Para que um banco de dados ou uma tabela compartilhada apareça no editor de consultas do Athena, crie um link de recurso no Lake Formation para o banco de dados ou a tabela compartilhada. Quando a conta de destinatário no Lake Formation consulta a tabela do proprietário, o CloudTrail adiciona o evento de acesso a dados aos logs das contas tanto de destinatário quanto de proprietário.

Para visualizações compartilhadas, lembre-se dos seguintes pontos:

- As consultas são executadas nos links dos recursos de destino, não na visualização ou tabela de origem. Em seguida, a saída é compartilhada com a conta de destino.
- Não basta compartilhar apenas a visualização. Todas as tabelas envolvidas na criação da visualização devem fazer parte do compartilhamento entre contas.
- O nome do link do recurso criado nos recursos compartilhados deve corresponder ao nome do recurso na conta do proprietário. Se o nome não corresponder, será gerada uma mensagem de erro semelhante a Failed analyzing stored view 'awsdatacatalog.*`my-lf-resource-link`*.*`my-lf-view`*': line 3:3: Schema *`schema_name`* does not exist (Falha na análise da visualização compartilhada “awsdatacatalog.my-lf-resource-link.my-lf-view”: linha 3:3: o esquema schema_name não existe).

Para obter mais informações sobre o acesso entre contas no Lake Formation, consulte os seguintes recursos no Guia do desenvolvedor do AWS Lake Formation:

Acesso entre contas

How resource links work in Lake Formation (Como os links de recursos funcionam no Lake Formation)

Cross-account CloudTrail logging (Registro em log no CloudTrail entre contas)

Locais do Amazon S3 criptografados por CSE-KMS e registrados no Lake Formation

Tabelas em Open Table Format (OTF), como do Apache Iceberg, que têm as seguintes características, não podem ser consultadas com o Athena:

- As tabelas são baseadas nos locais de dados do Amazon S3 que estão registrados no Lake Formation.
- Os objetos no Amazon S3 são criptografados usando a criptografia do lado do cliente (CSE).
- A criptografia usa chaves do AWS KMS gerenciadas pelo cliente (`CSE_KMS`).

Para consultar tabelas não OTF criptografadas com uma chave `CSE_KMS`, adicione o bloco a seguir à política da chave AWS KMS que você usa para criptografia CSE. *<KMS_KEY_ARN>* indica o ARN da chave AWS KMS que criptografa os dados. *<IAM-ROLE-ARN>* indica o ARN do perfil do IAM que registra a localização do Amazon S3 no 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>"
        }
    }
}
```

Locais de dados particionados registrados no Lake Formation devem estar em subdiretórios de tabela

As tabelas particionadas registradas no Lake Formation devem ter dados particionados em diretórios que são subdiretórios da tabela no Amazon S3. Por exemplo, uma tabela com o local `s3://DOC-EXAMPLE-BUCKET/mytable` e as partições `s3://DOC-EXAMPLE-BUCKET/mytable/dt=2019-07-11`, `s3://DOC-EXAMPLE-BUCKET/mytable/dt=2019-07-12` etc. pode ser registrada no Lake Formation e consultada com o Athena. Por outro lado, uma tabela com o local `s3://DOC-EXAMPLE-BUCKET/mytable` e as partições localizadas em `s3://DOC-EXAMPLE-`

`BUCKET/dt=2019-07-11`, `s3://DOC-EXAMPLE-BUCKET/dt=2019-07-12` etc. não pode ser registrada no Lake Formation. Como essas partições não são subdiretórios de `s3://DOC-EXAMPLE-BUCKET/mytable`, elas também não podem ser lidas pelo Athena.

Consultas Create Table As Select (CTAS) exigem permissões de gravação do Amazon S3

As Create Table As Statements (CTAS) exigem acesso de gravação ao local das tabelas do Amazon S3. Para executar consultas CTAS em dados registrados no Lake Formation, os usuários do Athena devem ter as permissões do IAM para gravar nos locais de tabela do Amazon S3 e as devidas permissões do Lake Formation para ler os locais de dados. Para ter mais informações, consulte Criar uma tabela a partir de resultados de consultas (CTAS).

A permissão DESCRIBE é necessária no banco de dados padrão

A permissão DESCRIBE do Lake Formation é necessária no banco de dados `default`. O comando da AWS CLI de exemplo a seguir concede a permissão `DESCRIBE` no banco de dados `default` ao usuário `datalake_user1` na conta da AWS `111122223333`.

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --
permissions "DESCRIBE" --resource '{ "Database": {"Name":"default"}}
```

Para obter mais informações, consulte Lake Formation Permissions Reference (Referência de permissões do Lake Formation) no Guia do desenvolvedor do AWS Lake Formation.

Gerenciar permissões de usuário do Lake Formation e do Athena

O Lake Formation vende credenciais para consultar armazenamentos de dados do Amazon S3 registrados no Lake Formation. Se você já usou políticas do IAM para permitir ou negar permissões para ler locais de dados no Amazon S3, pode usar as permissões do Lake Formation. No entanto, outras permissões do IAM ainda são necessárias.

Sempre que você usar as políticas do IAM, siga as práticas recomendadas do IAM. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM no Manual do usuário do IAM.

As seções a seguir resumem as permissões necessárias para usar o Athena nas consultas de dados registrados no Lake Formation. Para obter mais informações, consulte Segurança no AWS Lake Formation no Guia do desenvolvedor do AWS Lake Formation.

Resumo de permissões

- Permissões baseadas em identidade para Lake Formation e Athena
- Permissões do Amazon S3 para locais de resultados de consultas do Athena
- Associações de grupos de trabalho do Athena para consultar o histórico
- Permissões do Lake Formation para dados
- Permissões do IAM para gravar em locais do Amazon S3
- Permissões para dados criptografados, metadados e resultados de consultas do Athena
- Permissões baseadas em recursos para buckets do Amazon S3 em contas externas (opcional)

Permissões baseadas em identidade para Lake Formation e Athena

Qualquer pessoa que usa o Athena para consultar dados registrados no Lake Formation deve ter uma política de permissões do IAM que permita a ação `lakeformation:GetDataAccess`. O Política gerenciada pela AWS: AmazonAthenaFullAccess permite essa ação. Se você usar políticas em linha, atualize as políticas de permissões para permitir essa ação.

No Lake Formation, um administrador de data lake tem permissões para criar objetos de metadados, como bancos de dados e tabelas, conceder permissões do Lake Formation a outros usuários e registrar novos locais do Amazon S3. Para registrar novos locais, são necessárias permissões para a função vinculada ao serviço no Lake Formation. Para obter mais informações, consulte Create a data lake administrator (Criar um administrador de data lake) e Service-linked role permissions for Lake Formation (Permissões de função vinculada ao serviço no Lake Formation) no Guia do desenvolvedor do AWS Lake Formation.

Um usuário do Lake Formation pode usar o Athena para consultar bancos de dados, tabelas, colunas de tabelas e armazenamentos de dados subjacentes do Amazon S3 com base nas permissões do Lake Formation concedidas a ele pelos administradores de data lake. Os usuários não podem criar bancos de dados ou tabelas, nem registrar novos locais do Amazon S3 no Lake Formation. Para obter mais informações, consulte Create a data lake user (Criar um usuário do data lake) no Guia do desenvolvedor do AWS Lake Formation.

No Athena, as políticas de permissões baseadas em identidade, incluindo aquelas para grupos de trabalho do Athena, ainda controlam o acesso às ações do Athena para usuários de contas da Amazon Web Services. Além disso, o acesso federado pode ser concedido por meio da autenticação baseada em SAML disponível nos drivers do Athena. Para obter mais informações, consulte Usar grupos de trabalho para controlar o acesso a consultas e os custos, Políticas do IAM para acessar grupos de trabalho e Permitir acesso federado à API do Athena.

Para obter mais informações, consulte Granting Lake Formation permissions (Conceder permissões do Lake Formation) no Guia do desenvolvedor do AWS Lake Formation.

Permissões do Amazon S3 para locais de resultados de consultas do Athena

O local dos resultados de consultas do Athena no Amazon S3 não pode ser registrado no Lake Formation. As permissões do Lake Formation não limitam o acesso a esses locais. A menos que você limite o acesso, os usuários do Athena podem acessar arquivos de resultados de consulta e metadados mesmo sem permissões do Lake Formation para os dados. Para evitar isso, recomendamos que você use grupos de trabalho para especificar o local dos resultados das consultas e alinhe a associação do grupo de trabalho com as permissões do Lake Formation. Depois disso, você pode usar as políticas de permissões do IAM para limitar o acesso aos locais de resultados de consultas. Para obter mais informações sobre resultados de consultas, veja Trabalhar com resultados de consultas, consultas recentes e arquivos de saída.

Associações de grupos de trabalho do Athena para consultar o histórico

O histórico de consultas do Athena expõe uma lista de consultas salvas e strings de consulta completas. A menos que você use grupos de trabalho para separar o acesso a históricos de consultas, os usuários do Athena que não estão autorizados a consultar os dados no Lake Formation poderão visualizar as strings de consultas executadas nesses dados, incluindo nomes de coluna, critérios de seleção e etc. Recomendamos que você use grupos de trabalho para separar históricos de consultas e alinhar a associação do grupo de trabalho do Athena com as permissões do Lake Formation para limitar o acesso. Para ter mais informações, consulte Usar grupos de trabalho para controlar o acesso a consultas e os custos.

Permissões do Lake Formation para dados

Além da permissão de linha de base para usar o Lake Formation, os usuários do Athena devem ter permissões do Lake Formation para acessar os recursos que eles consultam. Essas permissões são concedidas e gerenciadas por um administrador do Lake Formation. Para obter mais informações, consulte Security and access control to metadata and data (Controle de segurança e acesso a metadados e dados) no Guia do desenvolvedor do AWS Lake Formation.

Permissões do IAM para gravar em locais do Amazon S3

As permissões do Lake Formation para o Amazon S3 não incluem a capacidade de gravar no Amazon S3. As Create Table As Statements (CTAS) exigem acesso de gravação ao local das tabelas do Amazon S3. Para executar consultas CTAS em dados registrados no Lake Formation, os

usuários do Athena devem ter as permissões do IAM para gravar nos locais de tabela do Amazon S3 e as devidas permissões do Lake Formation para ler os locais de dados. Para ter mais informações, consulte Criar uma tabela a partir de resultados de consultas (CTAS).

Permissões para dados criptografados, metadados e resultados de consultas do Athena

É possível criptografar dados de origem subjacentes no Amazon S3 e metadados no Catálogo de dados registrados no Lake Formation. Não há alteração na maneira como o Athena processa a criptografia de resultados das consultas quando ele é usado para consultar dados registrados no Lake Formation. Para ter mais informações, consulte Criptografar resultados das consultas do Athena armazenados no Amazon S3.

- Criptografia de dados de origem: a criptografia dos dados de origem dos locais de dados do Amazon S3 é permitida. Os usuários do Athena que consultam locais criptografados do Amazon S3 registrados no Lake Formation precisam de permissões para criptografar e descriptografar dados. Para obter mais informações sobre os requisitos, consulte Opções de criptografia permitidas do Amazon S3 e Permissões para dados criptografados no Amazon S3.
- Criptografia de metadados: a criptografia de metadados no Catálogo de dados é permitida. Para os principais que usam o Athena, as políticas baseadas em identidade devem permitir as ações `"kms:GenerateDataKey"`, `"kms:Decrypt"` e `"kms:Encrypt"` à chave usada para criptografar os metadados. Para obter mais informações, consulte Encrypting your Data Catalog (Criptografar seu catálogo de dados) no Guia do desenvolvedor do AWS Glue e Acesso a metadados criptografados do Athena no AWS Glue Data Catalog.

Permissões baseadas em recursos para buckets do Amazon S3 em contas externas (opcional)

Para consultar um local de dados do Amazon S3 em uma conta diferente, uma política do IAM baseada em recurso (política de bucket) deve permitir o acesso ao local. Para ter mais informações, consulte Acesso entre contas no Athena aos buckets do Amazon S3.

Para obter informações sobre como acessar um catálogo de dados em outra conta, consulte Acesso ao catálogo de dados entre contas do Athena.

Aplicar permissões do Lake Formation a bancos de dados e tabelas existentes

Se você é novo no Athena e usa o Lake Formation para configurar o acesso aos dados da consulta, não precisa configurar políticas do IAM para que os usuários possam ler os dados do Amazon S3 e criar metadados. É possível usar o Lake Formation para administrar permissões.

Registrar dados no Lake Formation e atualizar políticas de permissões do IAM não são requisitos. Se os dados não estiverem registrados no Lake Formation, os usuários do Athena com as devidas permissões no Amazon S3 e no AWS Glue, se aplicável, poderão continuar consultando os dados não registrados no Lake Formation.

Se você tiver usuários existentes do Athena que consultam dados não registrados no Lake Formation, poderá atualizar as permissões do IAM para o Amazon S3 e o AWS Glue Data Catalog, se aplicável, de modo que você possa usar as permissões do Lake Formation para gerenciar o acesso dos usuários centralmente. Para obter permissão de leitura dos locais de dados do Amazon S3, é possível atualizar as políticas baseadas em recurso e em identidade para modificar as permissões do Amazon S3. Para obter acesso aos metadados, se você configurou políticas no nível do recurso para controle de acesso refinado com o AWS Glue, pode usar as permissões do Lake Formation para gerenciar o acesso.

Para obter mais informações, consulte Acesso granular a bancos de dados e tabelas no AWS Glue Data Catalog e Atualização das permissões de dados do AWS Glue para o modelo do AWS Lake Formation no Guia do desenvolvedor do AWS Lake Formation.

Usar o Lake Formation e os drivers JDBC e ODBC do Athena para acesso federado ao Athena

Os drivers Athena JDBC e ODBC permitem a federação baseada em SAML 2.0 com o Athena usando os provedores de identidade Okta e Microsoft Active Directory Federation Services (AD FS). Ao integrar o Amazon Athena com o AWS Lake Formation, você habilita a autenticação baseada em SAML no Athena com credenciais corporativas. Com o Lake Formation e o AWS Identity and Access Management (IAM), você pode manter um controle de acesso refinado no nível da coluna dos dados disponíveis para o usuário do SAML. Com os drivers Athena JDBC e ODBC, o acesso federado está disponível por ferramenta ou de modo programático.

Para usar o Athena para acessar uma origem dos dados controlada pelo Lake Formation, você precisa habilitar a federação baseada em SAML 2.0 configurando seu Identity Provider (IdP – Provedor de identidade) e as funções do AWS Identity and Access Management (IAM). Para obter detalhes das etapas, consulte, Tutorial: Configurar acesso federado para usuários do Okta ao Athena usando Lake Formation e JDBC.

Pré-requisitos

Para usar o Amazon Athena e o Lake Formation para acesso federado, você deve atender aos seguintes requisitos:

- Gerencie suas identidades corporativas usando um provedor de identidade baseado em SAML existente, como Okta ou Microsoft Active Directory Federation Services (AD FS).
- Use o AWS Glue Data Catalog como um armazenamento de metadados.
- Defina e gerencie as permissões no Lake Formation para acessar bancos de dados, tabelas e colunas no AWS Glue Data Catalog. Para mais informações, consulte o Guia do desenvolvedor do AWS Lake Formation.
- Use a versão 2.0.14 ou mais recente do driver JDBC do Athena ou a versão 1.1.3 ou mais recente do driver ODBC do Athena.

Considerações e limitações

Ao usar o driver Athena JDBC ou ODBC e o Lake Formation para configurar o acesso federado ao Athena, tenha em mente os seguintes pontos:

- No momento, o driver Athena JDBC e drivers ODBC são compatíveis com os provedores de identidade Okta, Microsoft Active Directory Federation Services (AD FS) e Azure AD. Embora o driver Athena JDBC tenha uma classe SAML genérica que pode ser estendida para usar outros provedores de identidade, o suporte a extensões personalizadas que permitem o uso de outros provedores de identidade (IdPs) com o Athena pode ser limitado.
- O acesso federado usando os drivers JDBC e ODBC não é compatível com o recurso de propagação de identidade confiável do Centro de Identidade do IAM.
- Atualmente, você não pode usar o console do Athena para configurar o suporte para uso de IdP e SAML com o Athena. Para configurar esse suporte, use o provedor de identidade de terceiro, os consoles de gerenciamento do Lake Formation e do IAM e o cliente de driver JDBC ou ODBC.
- Você deve entender a especificação SAML 2.0 e como ela funciona com seu provedor de identidade antes de configurar o provedor de identidade e o SAML para uso com o Lake Formation e o Athena.
- Os provedores SAML e os drivers Athena JDBC e ODBC são fornecidos por terceiros, portanto, o suporte da AWS para questões relacionadas ao uso deles pode ser limitado.

Tópicos

- Tutorial: Configurar acesso federado para usuários do Okta ao Athena usando Lake Formation e JDBC

Tutorial: Configurar acesso federado para usuários do Okta ao Athena usando Lake Formation e JDBC

Este tutorial mostra como configurar o Okta, o AWS Lake Formation, as permissões do AWS Identity and Access Management e o driver Athena JDBC para habilitar o uso federado baseado em SAML do Athena. O Lake Formation oferece controle de acesso refinado dos dados que estão disponíveis no Athena para o usuário baseado em SAML. Para definir essa configuração, o tutorial usa o console do desenvolvedor do Okta, os consoles do AWS IAM e do Lake Formation e a ferramenta SQL Workbench/J.

Pré-requisitos

Este tutorial pressupõe que você tenha feito o seguinte:

- Criado uma conta da Amazon Web Services. Para criar uma conta, acesse a home page da Amazon Web Services.
- Configurado um local de resultados de consultas do Athena no Amazon S3.
- Registrado um local de bucket de dados do Amazon S3 no Lake Formation.
- Definido um banco de dados e tabelas no Catálogo de dados do AWS Glue que aponte para seus dados no Amazon S3.
  - Se você ainda não definiu uma tabela, execute um crawler do AWS Glue ou use o Athena para definir um banco de dados e uma ou mais tabelas para os dados que deseja acessar.
  - Este tutorial usa uma tabela baseada no conjunto de dados de trajetos de táxi na cidade de Nova York disponível no Registry of Open Data on AWS (Registro aberto de dados na). O tutorial usa o nome do banco de dados `tripdb` e o nome da tabela`nyctaxi`.

Etapas do tutorial

- Etapa 1: Criar uma conta do Okta
- Etapa 2: Adicionar usuários e grupos ao Okta
- Etapa 3: Configurar uma aplicação do Okta para autenticação SAML
- Etapa 4: Criar um provedor de identidade SAML para AWS e uma função do IAM para acesso ao Lake Formation
- Etapa 5: Adicionar a função do IAM e o provedor de identidade SAML à aplicação do Okta
- Etapa 6: Conceder permissões de usuário e grupo pelo AWS Lake Formation
- Etapa 7: Verificar o acesso pelo cliente Athena JDBC

- Conclusão
- Recursos relacionados

Etapa 1: Criar uma conta do Okta

Este tutorial usa o Okta como provedor de identidade baseado em SAML. Se você ainda não tem uma conta do Okta, pode criar uma gratuitamente. Ela é necessária para que você possa criar uma aplicação do Okta para autenticação SAML.

Para criar uma conta do Okta

1. Para usar o Okta, navegue até a página de cadastro de desenvolvedor do Okta e crie uma conta de avaliação gratuita. O Developer Edition Service é gratuito dentro dos limites especificados pelo Okta em developer.okta.com/pricing.
2. Quando você receber o e-mail de ativação, ative sua conta.

   Um nome de domínio do Okta será atribuído a você. Salve-o para referência. Mais tarde, você usará o nome de domínio (*<okta-idp-domain>*) na string JDBC que se conecta ao Athena.

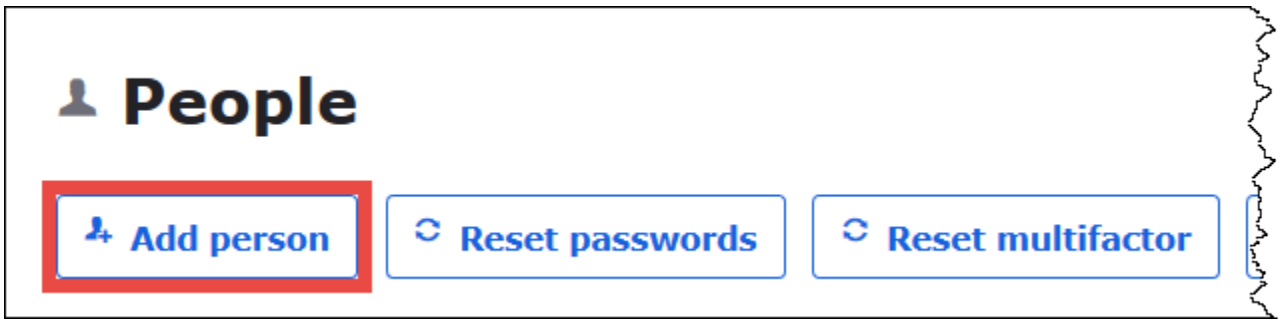
Etapa 2: Adicionar usuários e grupos ao Okta

Nesta etapa, você usa o console do Okta para executar as seguintes tarefas:

- Criar dois usuários do Okta.
- Criar dois grupos do Okta.
- Adicionar um usuário a cada grupo do Okta.

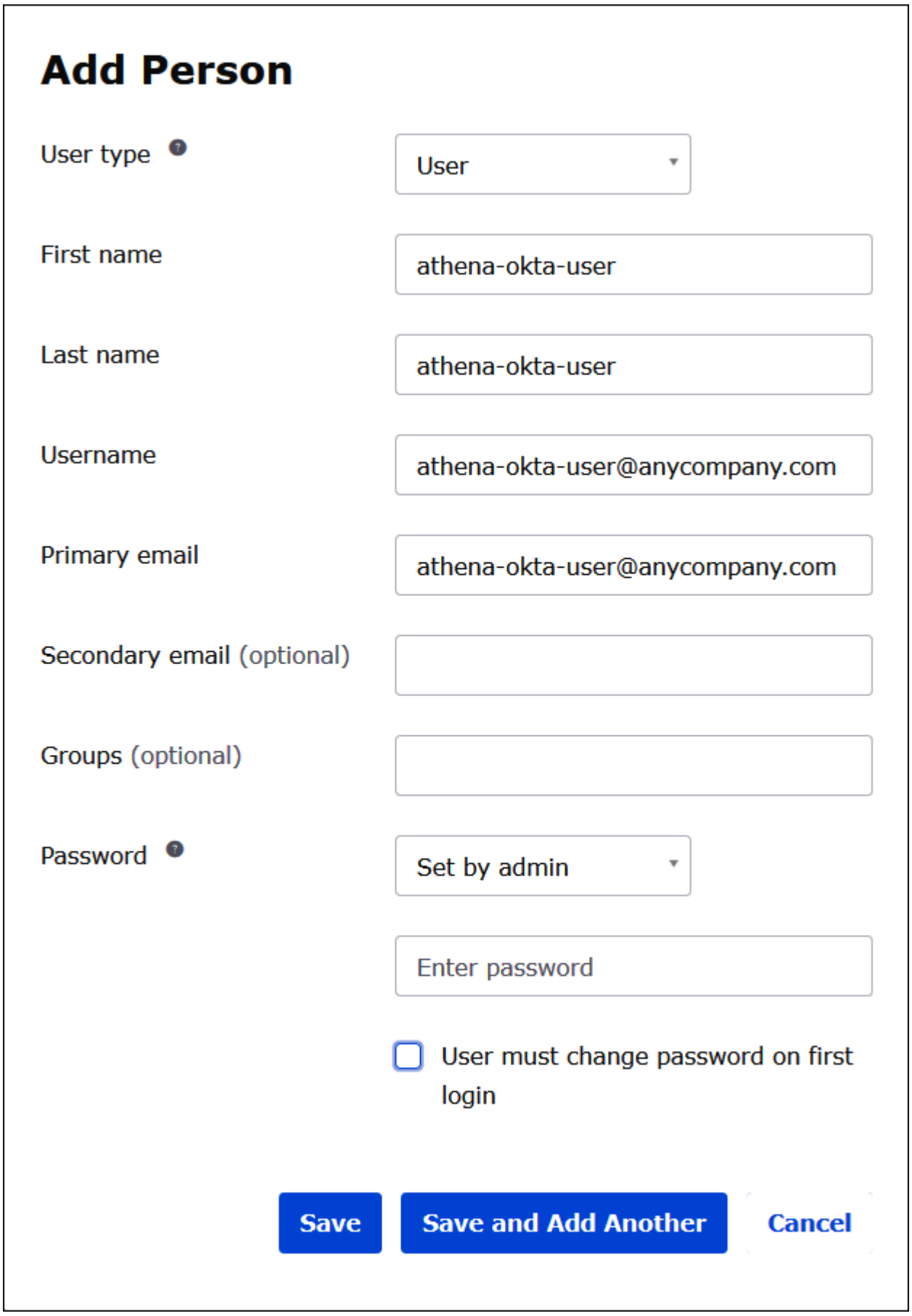
Para adicionar usuários ao Okta

1. Depois de ativar sua conta do Okta, faça login como usuário administrativo no domínio do Okta atribuído.
2. No painel de navegação à esquerda, escolha Directory (Diretório) e People (Pessoas).
3. Escolha Add Person (Adicionar pessoa) para adicionar um novo usuário que acessará o Athena por meio do driver JDBC.

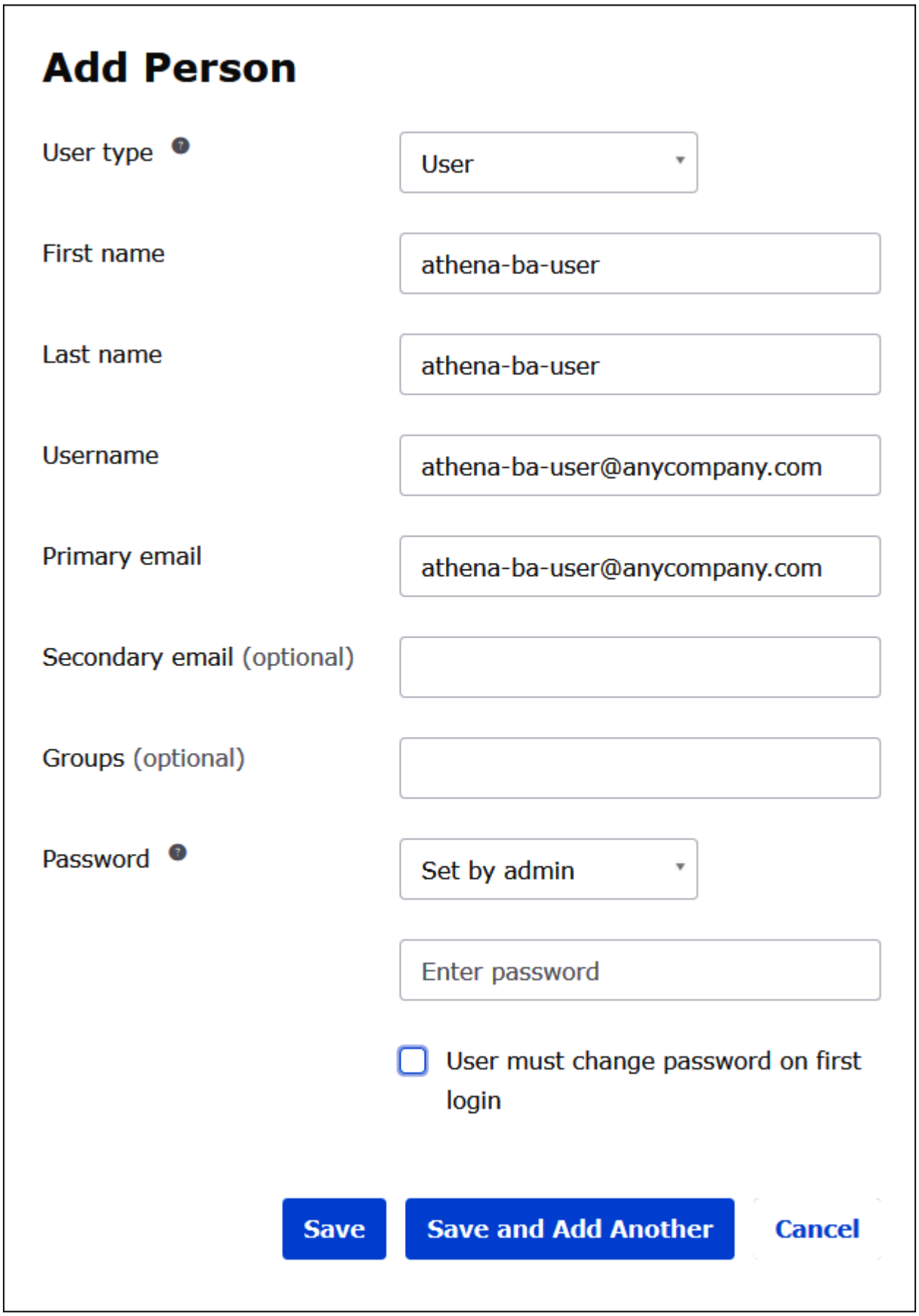


4. Na caixa de diálogo Add Person (Adicionar pessoa), insira as informações necessárias.

   - Insira os valores em First name (Nome) e Last name (Sobrenome). Este tutorial usa o *athena-okta-user*.
   - Insira um Username (Nome de usuário) e Primary email (E-mail principal). Este tutorial usa o *athena-okta-user@anycompany.com*.
   - Em Password (Senha), escolha Set by admin (Definir por administrador) e insira uma senha. Neste tutorial, a opção User must change password on first login (O usuário deve alterar a senha no primeiro login) está desmarcada, mas seus requisitos de segurança podem variar.


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. Escolha Save and Add Another (Salvar e adicionar outro).
6. Insira as informações do outro usuário. Este exemplo adiciona o usuário analista de negócios *athena-ba-user@anycompany.com*.

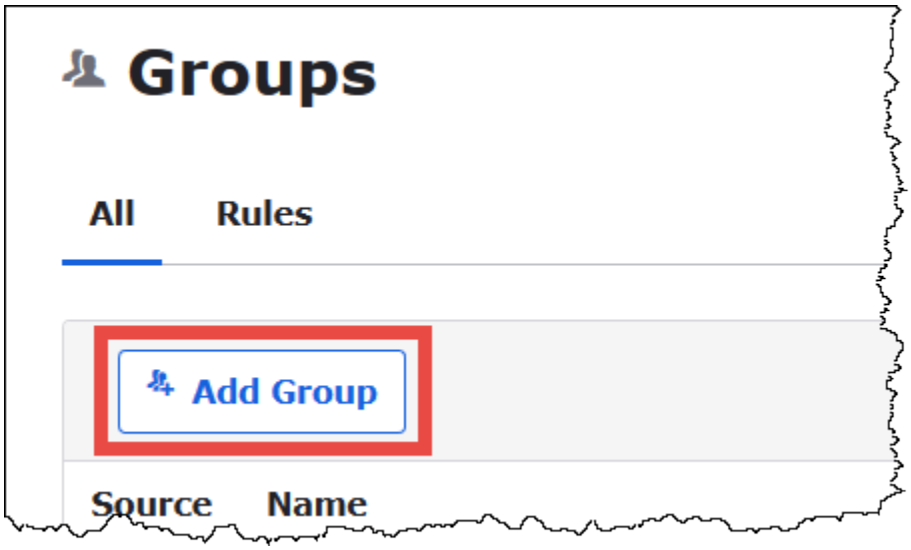

Add Person
User type
User
First name
athena-ba-user
Last name
athena-ba-user
Username
athena-ba-user@anycompany.com
Primary email
athena-ba-user@anycompany.com
Secondary email (optional)
Groups (optional)
Password
Set by admin
Enter password
User must change password on first login
Save
Save and Add Another
Cancel

7. Escolha Salvar.

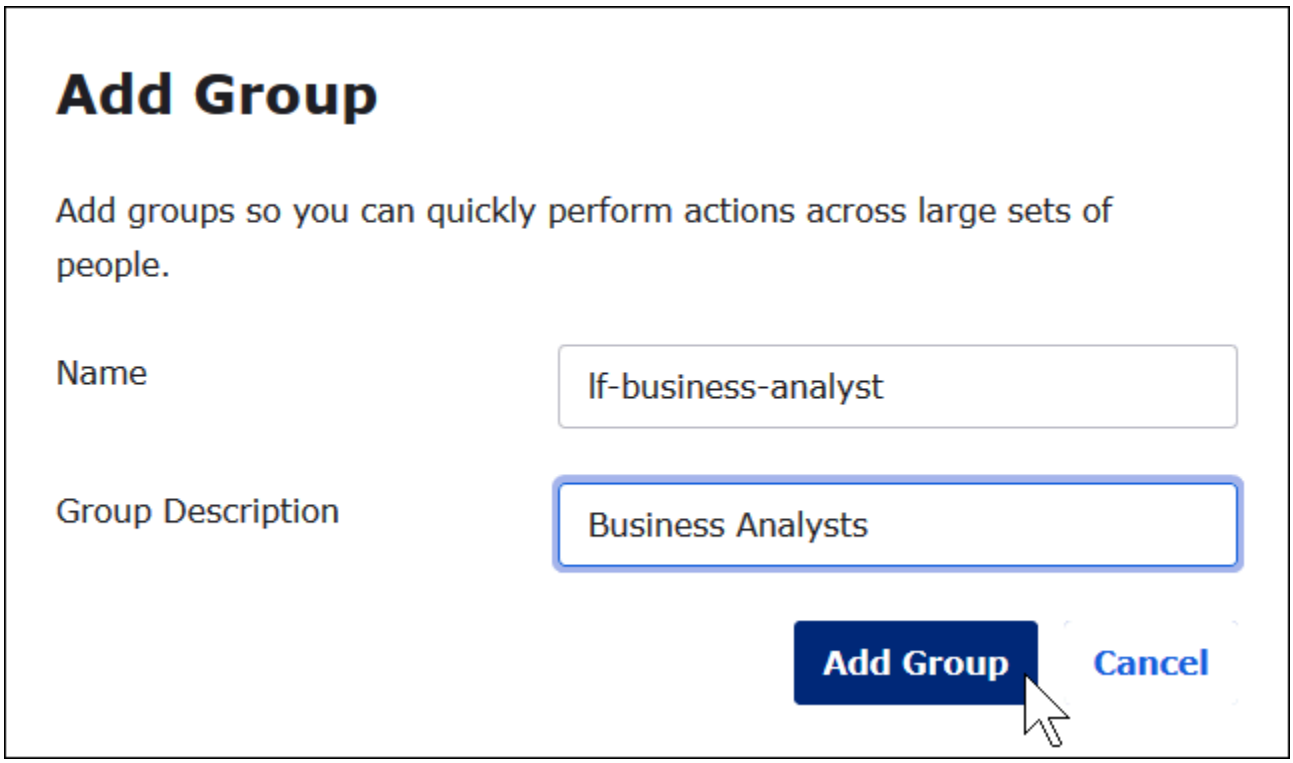
No procedimento a seguir, você concede acesso a dois grupos do Okta por meio do driver Athena JDBC adicionando um grupo “Business Analysts” e um grupo “Developer”.

Para adicionar grupos do Okta

1. No painel de navegação do Okta, escolha Directory (Diretório) e Groups (Grupos).
2. Na página Groups (Grupos), escolha Add Group (Adicionar grupo).



3. Na caixa de diálogo Add Group (Adicionar grupo), insira as informações necessárias.
   - Em Name (Nome), insira *`lf-business-analyst`*.
   - Em Group Description (Descrição do grupo), insira *`Business Analysts`* (Analistas de negócios).

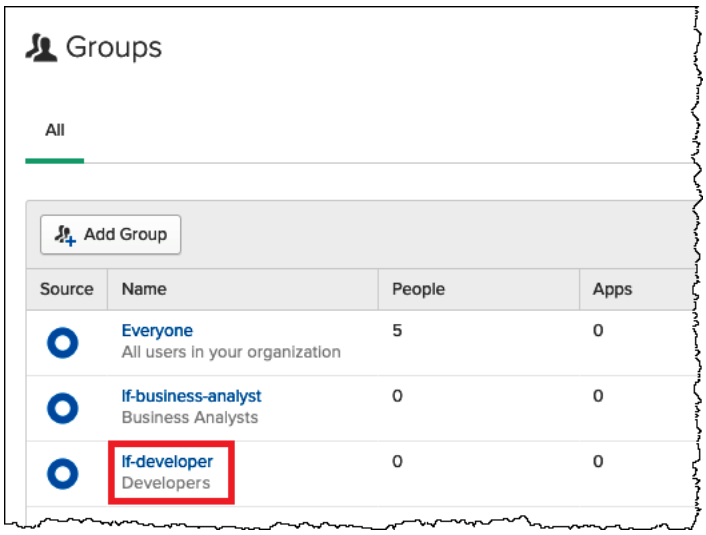


4. Escolha Adicionar grupo.
5. Na página Groups (Grupos), escolha Add Group (Adicionar grupo) novamente. Desta vez, você insere as informações do grupo Developer.
6. Insira as informações necessárias.
   - Em Name (Nome), insira `lf-developer`.
   - Em Group Description (Descrição do grupo), insira `Developers` (Desenvolvedores).
7. Escolha Adicionar grupo.

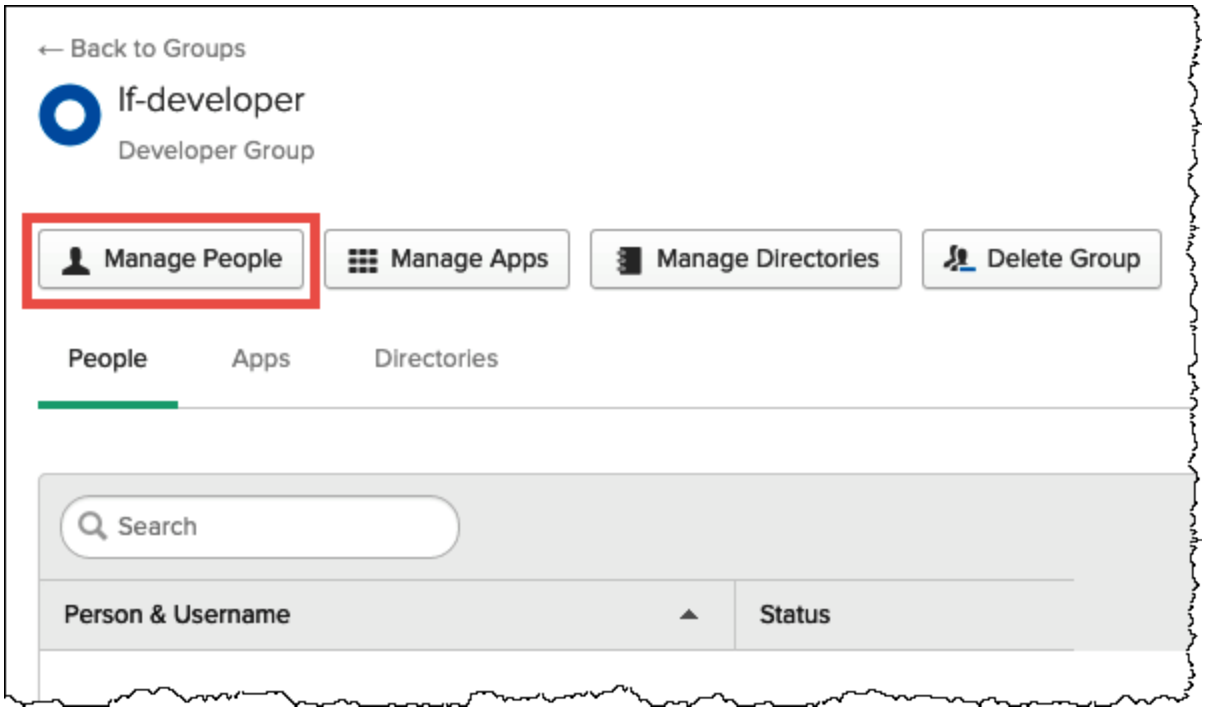
Agora que você tem dois usuários e dois grupos, está pronto para adicionar um usuário a cada grupo.

Para adicionar usuários a grupos

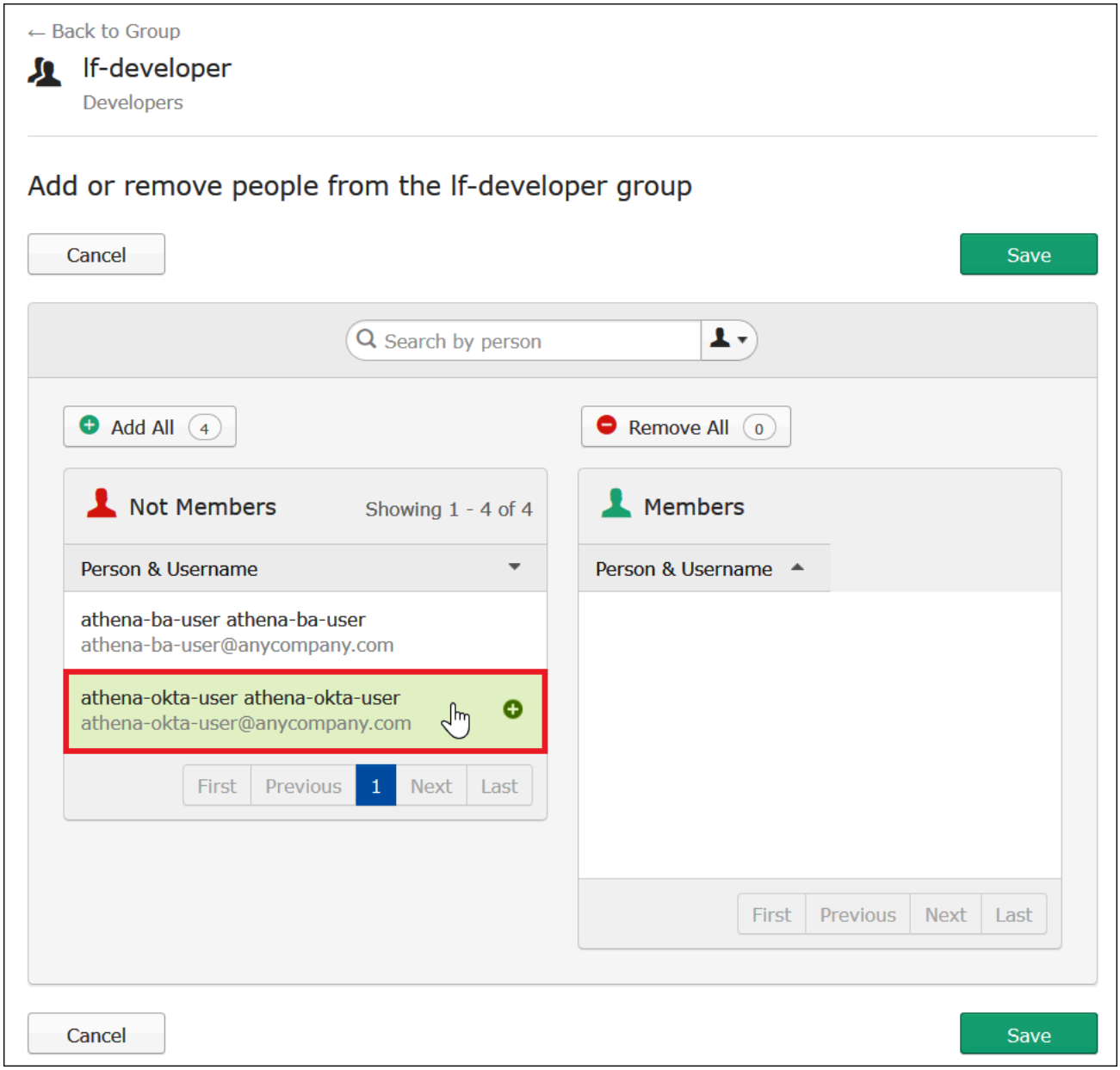
1. Na página Groups (Grupos), escolha o grupo lf-developer que você acabou de criar. Adicione a esse grupo um dos usuários do Okta que você criou como desenvolvedor.



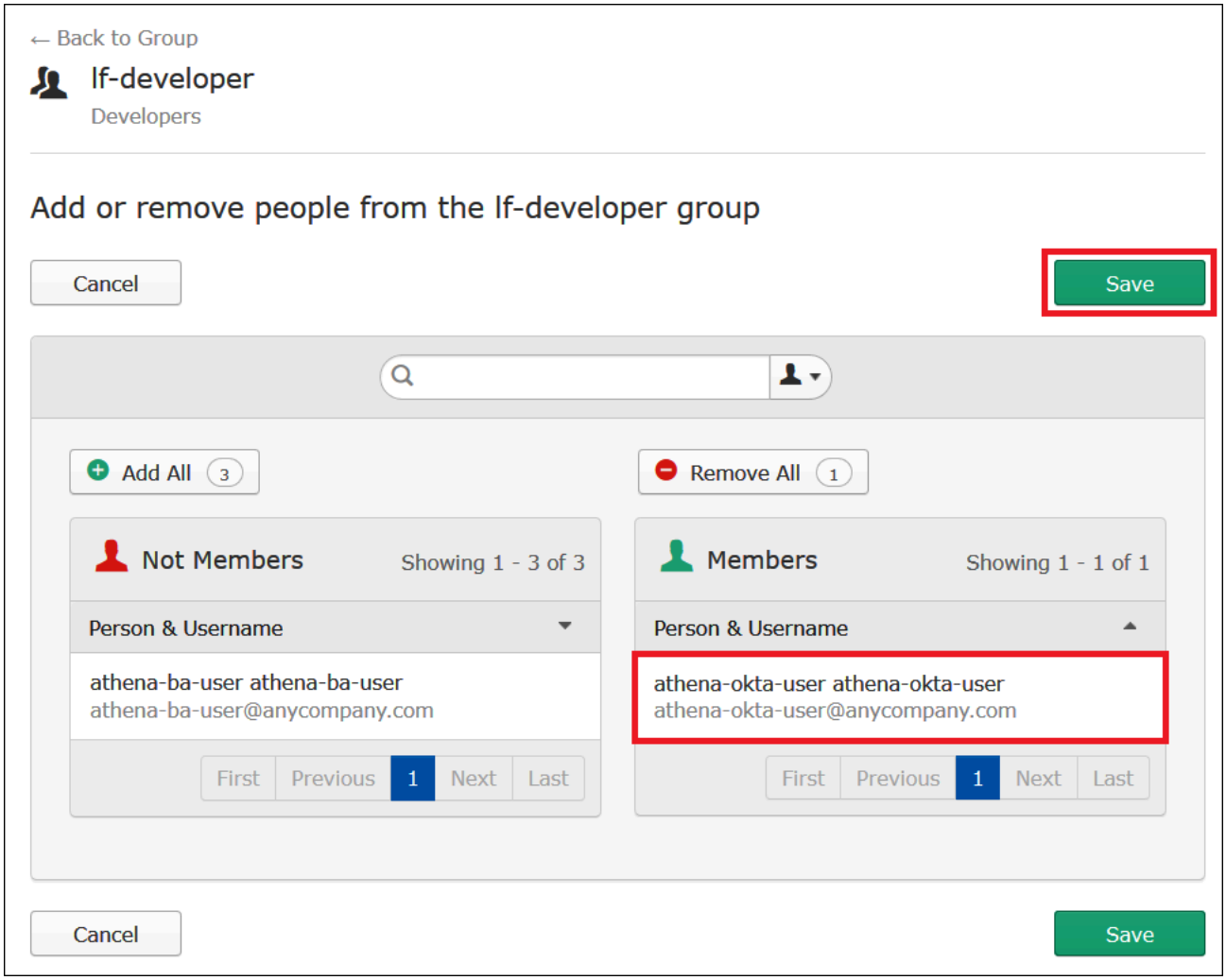
2. Escolha Manage People (Gerenciar pessoas).



3. Na lista Not Members (Não membros), escolha athena-okta-user.

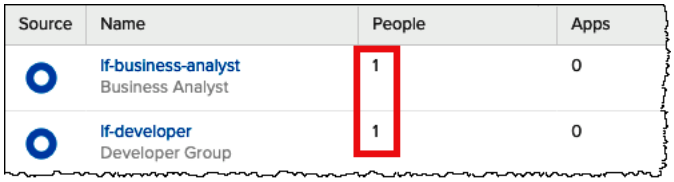


A entrada referente ao usuário é transferida da lista Not Members (Não membros) à esquerda para a lista Members (Membros) à direita.



4. Escolha Salvar.
5. Escolha Back to Group (Voltar ao grupo) ou Directory (Diretório) e selecione Groups (Grupos).
6. Escolha o grupo lf-business-analyst.
7. Escolha Manage People (Gerenciar pessoas).
8. Adicione o athena-ba-user à lista Members (Membros) do grupo lf-business-analyst e escolha Save (Salvar).
9. Escolha Back to Group (Voltar ao grupo) ou Directory (Diretório) e selecione Groups (Grupos).

   A página Groups (Grupos) agora mostra cada grupo com um usuário do Okta.



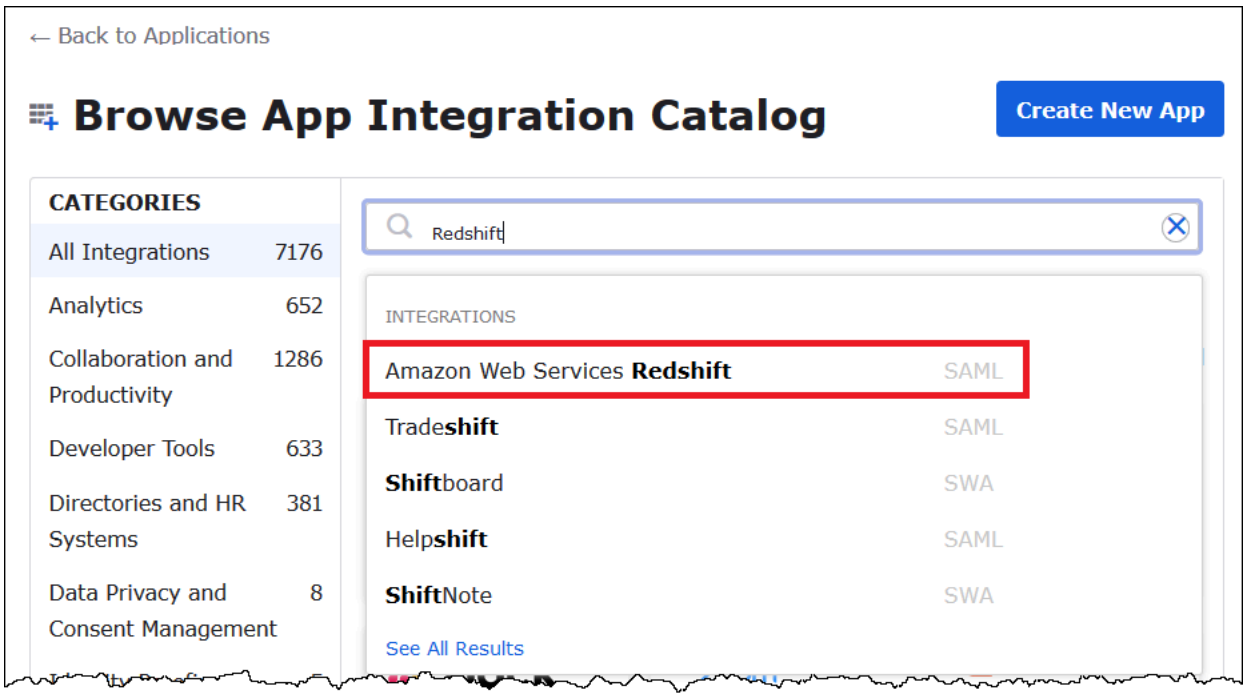
Etapa 3: Configurar uma aplicação do Okta para autenticação SAML

Nesta etapa, você usa o console de desenvolvedor do Okta para executar as seguintes tarefas:

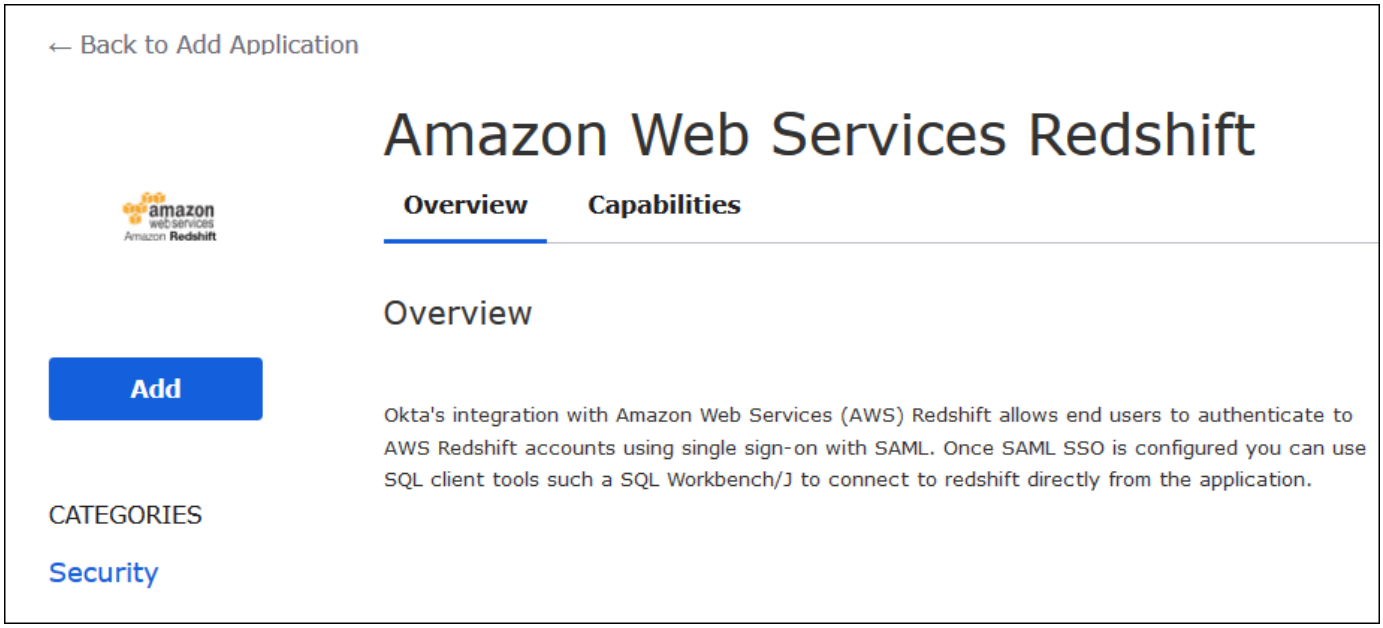
- Adicionar uma aplicação do SAML para usar com a AWS.
- Atribuir a aplicação a um usuário do Okta.
- Atribuir a aplicação a um grupo do Okta.
- Baixe os metadados resultantes do provedor de identidade para uso posterior com a AWS.

Para adicionar uma aplicação para autenticação SAML

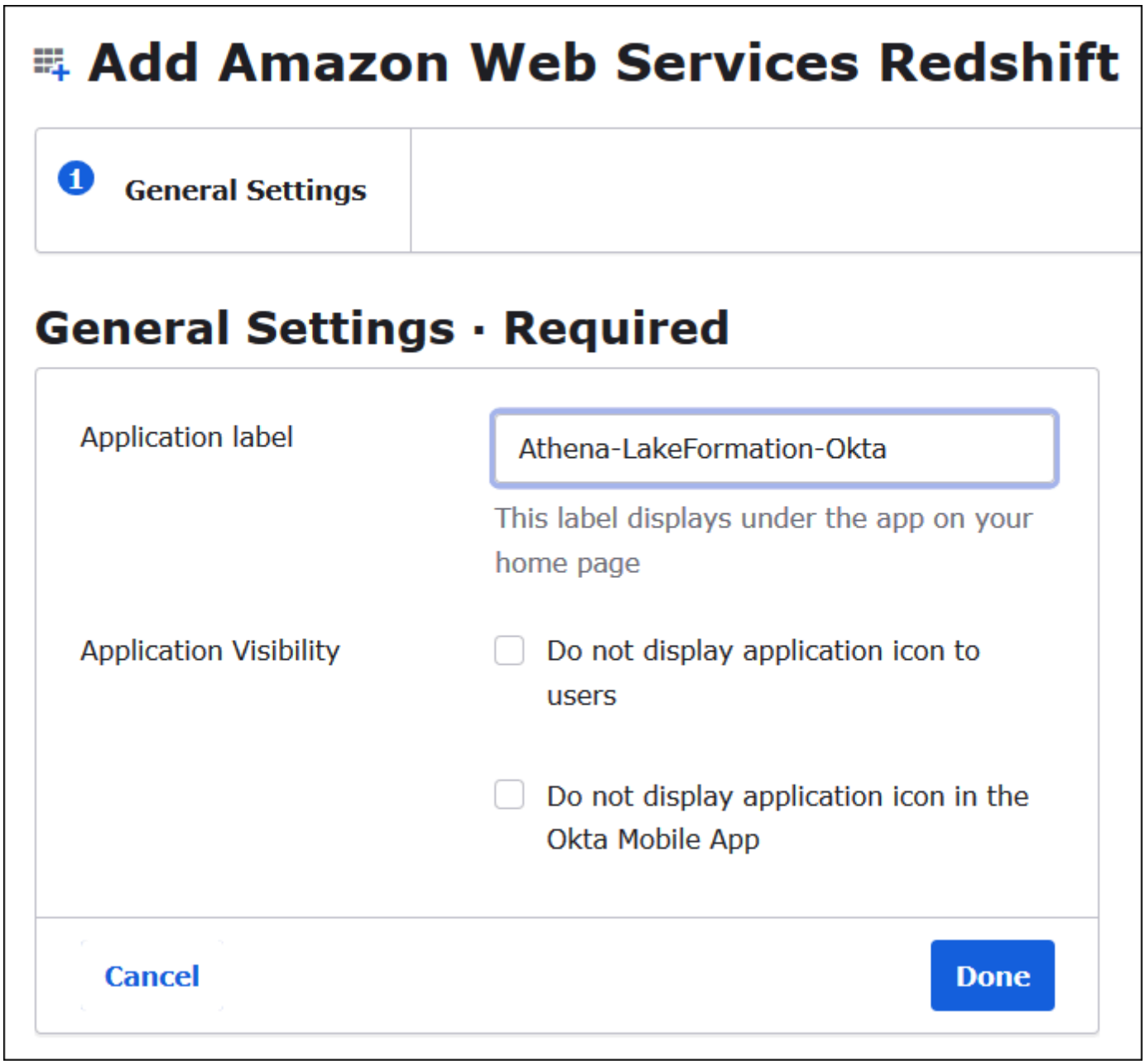
1. No painel de navegação do Okta, escolha Applications (Aplicações), Applications (Aplicações) para configurar uma aplicação do Okta para autenticação SAML no Athena.
2. Clique em Browse App Catalog (Procurar no catálogo de aplicações).
3. Na caixa de pesquisa, insira **`Redshift`**.
4. Escolha Amazon Web Services Redshift. Neste tutorial, a aplicação do Okta usa a integração existente com SAML para Amazon Redshift.



5. Na página Amazon Web Services Redshift, escolha Add (Adicionar) para criar uma aplicação baseada em SAML para o Amazon Redshift.



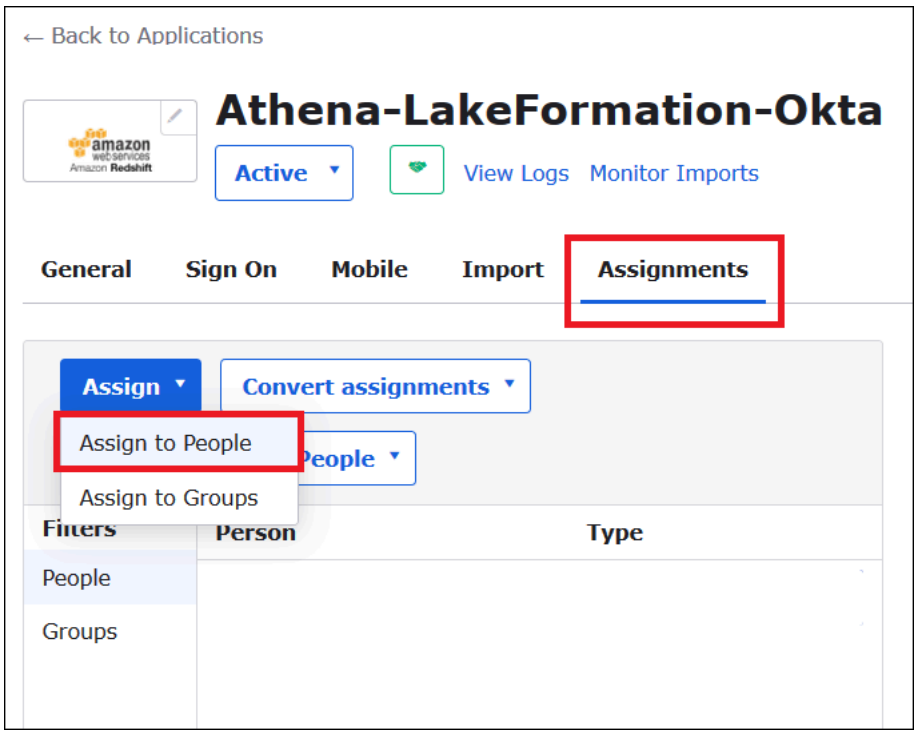
6. Em Application label (Rótulo da aplicação), insira `Athena-LakeFormation-Okta` e escolha Done (Concluído).



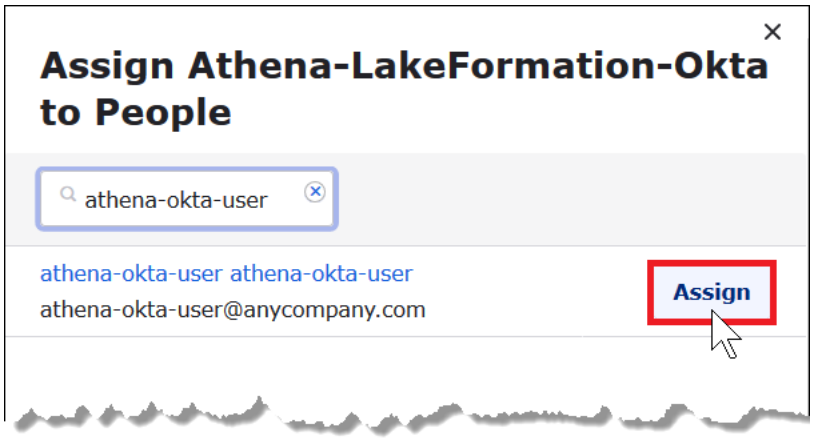
Agora que você criou uma aplicação do Okta, pode atribuí-la aos usuários e grupos criados.

Para atribuir uma aplicação a usuários e grupos

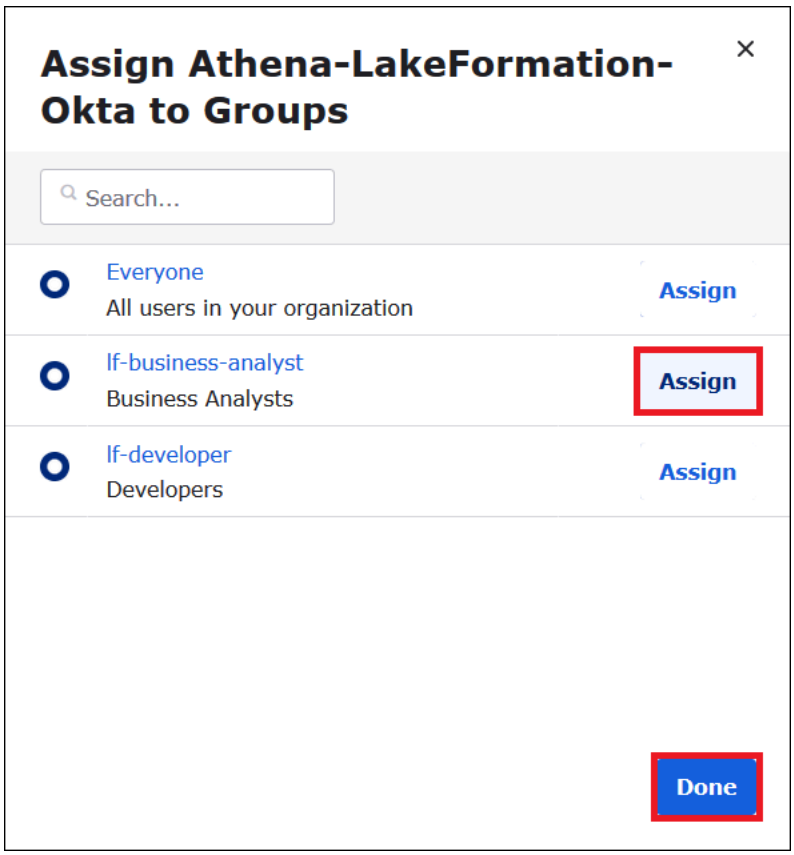
1. Na página Applications (Aplicações), escolha a aplicação Athena-LakeFormation-Okta.
2. Na guia Assignments (Atribuições), escolha Assign (Atribuir), Assign to People (Atribuir a pessoas).



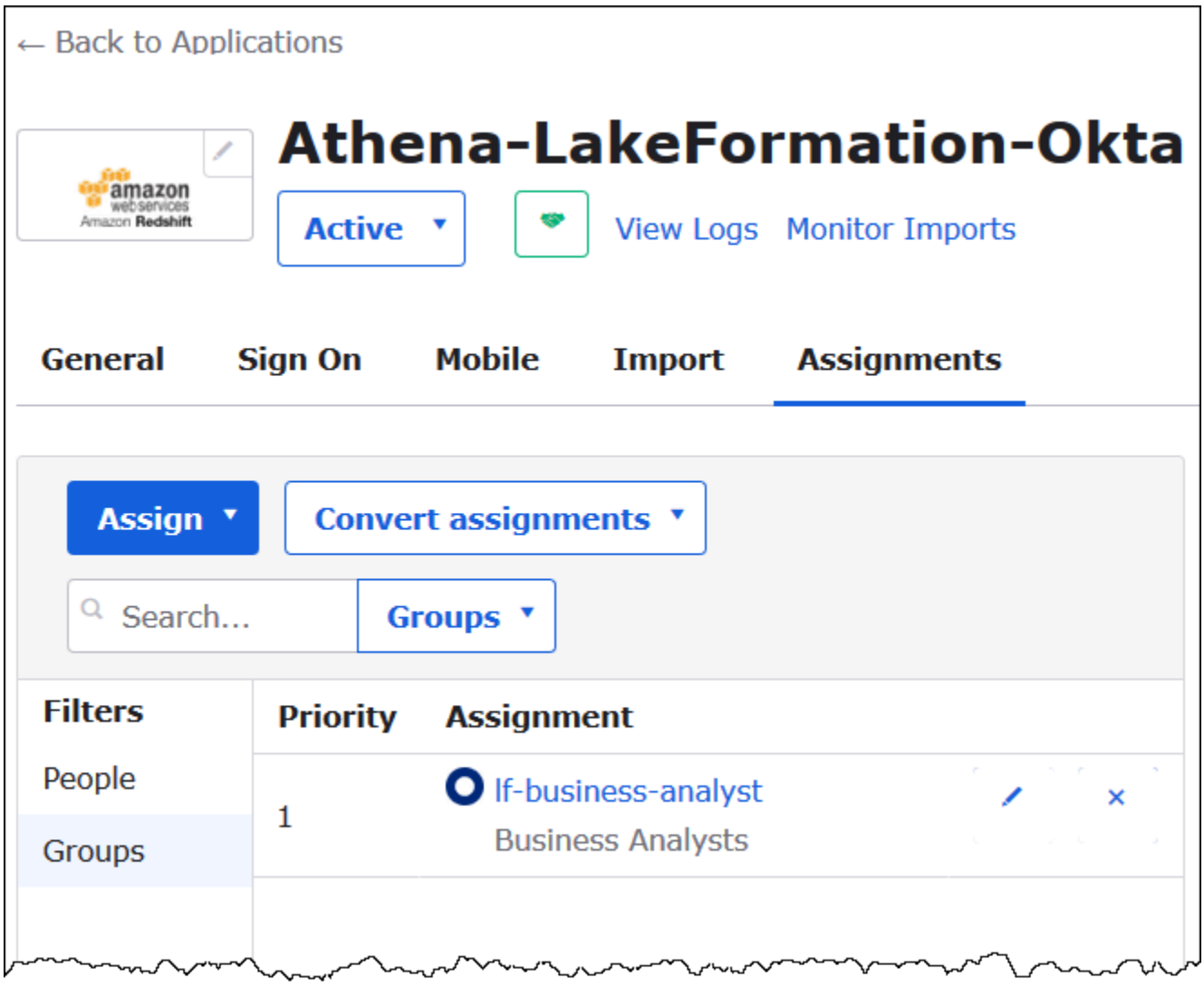
3. Na caixa de diálogo Assign Athena-LakeFormation-Okta to People (Atribuir Athena-LakeFormation-Okta a pessoas), encontre o usuário athena-okta-user que você criou.
4. Escolha Assign (Atribuir) para atribuir o usuário à aplicação.



5. Escolha Save and Go Back (Salvar e voltar).
6. Selecione Done (Concluído).
7. Na guia Assignments (Atribuições) da aplicação Athena-LakeFormation-Okta, escolha Assign (Atribuir), Assign to Groups (Atribuir a grupos).
8. Em lf-business-analyst, escolha Assign (Atribuir) para atribuir a aplicação Athena-LakeFormation-Okta ao grupo lf-business-analyst e escolha Done (Concluído).



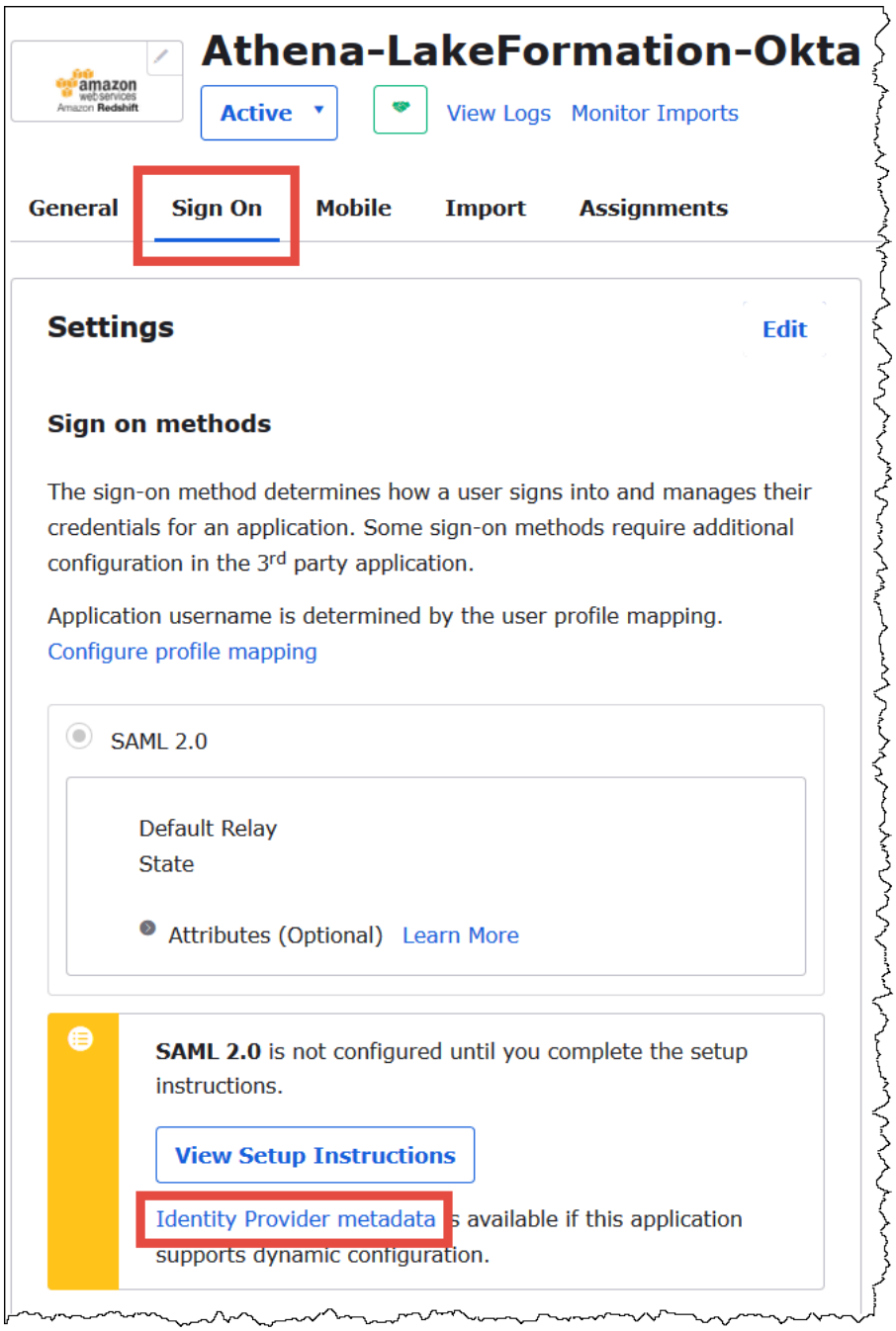
O grupo aparecerá na lista de grupos da aplicação.



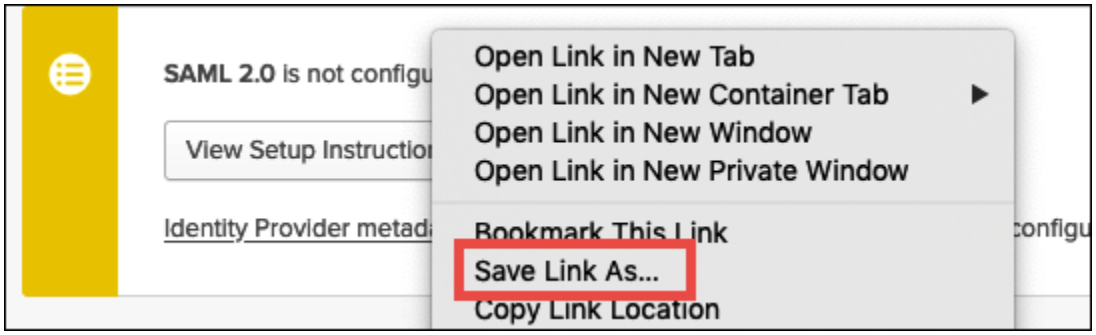
Agora você está pronto para baixar os metadados da aplicação do provedor de identidade para uso com a AWS.

Para baixar os metadados da aplicação

1. Escolha a guia Sign On (Logon) da aplicação do Okta e clique com o botão direito em Identity Provider metadata (Metadados do provedor de identidade).



2. Escolha Save Link As (Salvar link como) para salvar os metadados do provedor de identidade, que estão no formato XML, em um arquivo. Dê a ele um nome que você reconhece (por exemplo, `Athena-LakeFormation-idp-metadata.xml`).



Etapa 4: Criar um provedor de identidade SAML para AWS e uma função do IAM para acesso ao Lake Formation

Nesta etapa, use o console do AWS Identity and Access Management (IAM) para executar as seguintes tarefas:

- Criar um provedor de identidade para AWS.
- Criar uma função do IAM para acesso ao Lake Formation.
- Adicionar a política gerenciada AmazonAthenaFullAccess à função.
- Adicionar uma política para Lake Formation e AWS Glue à função.
- Adicionar uma política para os resultados das consultas do Athena à função.

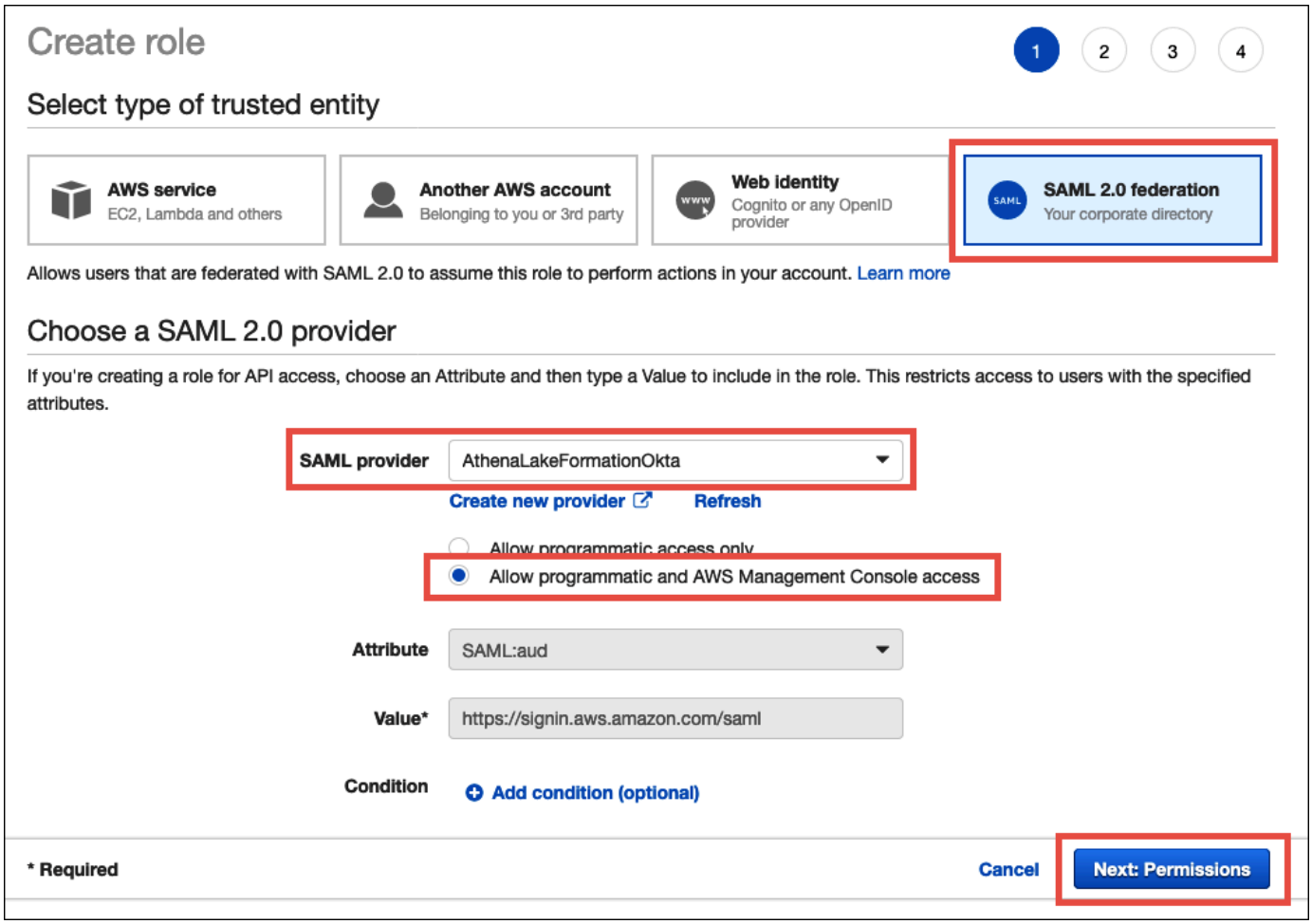
Para criar um provedor de identidade SAML para AWS

1. Faça login no console da conta da Amazon Web Services como administrador da conta da Amazon Web Services e navegue até o console do IAM (https://console.aws.amazon.com/iam/).
2. No painel de navegação, escolha Identity providers (Provedores de identidade) e clique em Add provider (Adicionar provedor).
3. Na tela Configure provider (Configurar provedor), insira as seguintes informações:
   - Em Provider type (Tipo de provedor), escolha SAML.
   - Em Provider name (Nome do provedor), insira `AthenaLakeFormationOkta`.
   - Em Metadata document (Documento de metadados), use a opção Choose file (Escolher arquivo) para carregar o arquivo XML de metadados do provedor de identidade (IdP) que você baixou.
4. Escolha Add provider (Adicionar provedor).

Em seguida, você cria uma função do IAM para acesso ao AWS Lake Formation. Adicione duas políticas em linha à função. Uma política concede permissões para acessar o Lake Formation e as APIs do AWS Glue. A outra política concede acesso ao Athena e ao local de resultados das consultas do Athena no Amazon S3.

Para criar uma função do IAM para acesso ao AWS Lake Formation

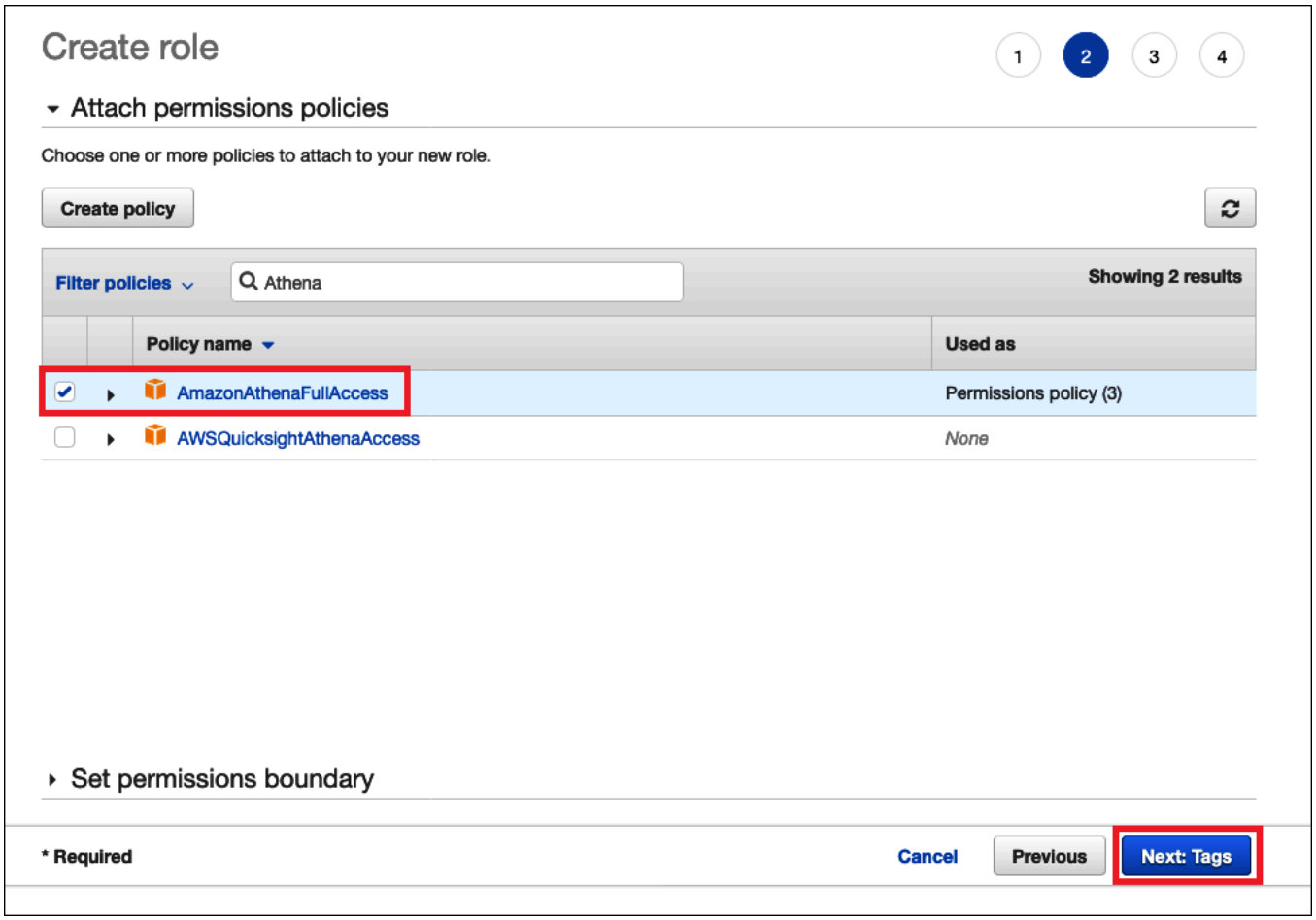
1. No painel de navegação do console do IAM, escolha Roles (Funções) e Create role (Criar função).
2. Na página Create role (Criar função), siga estas etapas:



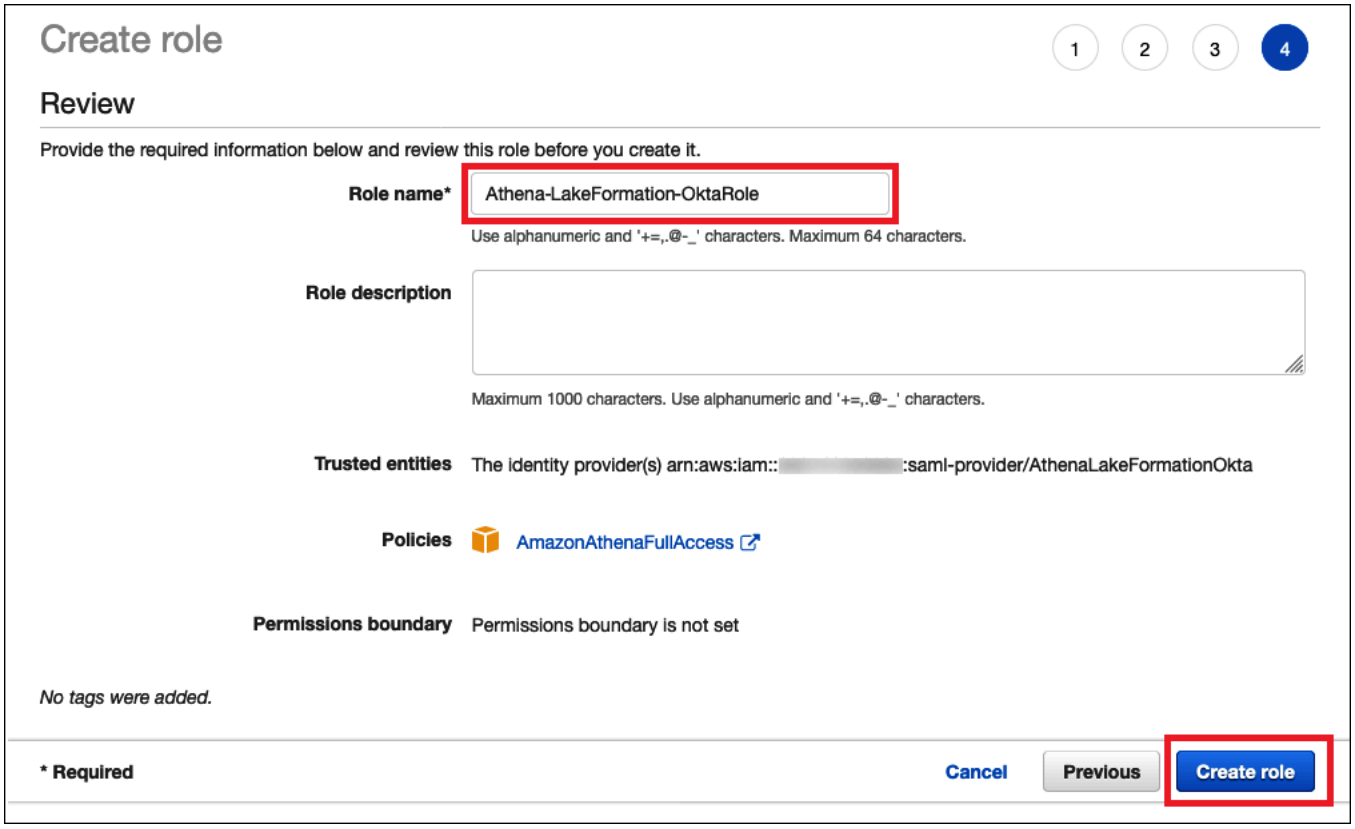
   a. Em Select type of trusted entity (Selecionar tipo de entidade confiável), escolha SAML 2.0 Federation.
   b. Em SAML provider (Provedor SAML), selecione AthenaLakeFormationOkta.
   c. Em SAML provider (Provedor SAML), selecione a opção Allow programmatic and AWS Management Console access (Permitir acesso programático e por console).

d. Escolha Próximo: permissões.

3. Na página Attach Permissions policies (Anexar políticas de permissões), em Filter policies (Filtrar políticas), insira **`Athena`**.
4. Selecione a política gerenciada AmazonAthenaFullAccess e escolha Next: Tags (Próximo: etiquetas).



5. Na página Add tags (Adicionar tags), escolha Next: Review(Próximo: Revisar).
6. Na página Review (Revisão), em Role name (Nome da função), insira um nome para a função (por exemplo, *`Athena-LakeFormation-OktaRole`*) e escolha Create role (Criar função).



Na sequência, adicione as políticas em linha que permitem o acesso ao Lake Formation, às APIs do AWS Glue e aos resultados das consultas do Athena no Amazon S3.

Sempre que você usar as políticas do IAM, siga as práticas recomendadas do IAM. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM no Manual do usuário do IAM.

Para adicionar uma política em linha à função para o Lake Formation e o AWS Glue

1. Na lista de funções no console do IAM, escolha a `Athena-LakeFormation-OktaRole` recém-criada.
2. Na página Summary (Resumo) da função, na guia Permissions (Permissões), escolha Add inline policy (Adicionar política em linha).
3. Na página Create policy (Criar política), escolha JSON.
4. Adicione uma política em linha, semelhante à mostrada abaixo, que concede acesso ao Lake Formation e às APIs do 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. Escolha Revisar política.
6. Em Name (Nome), insira um nome para a política (por exemplo, **`LakeFormationGlueInlinePolicy`**).
7. Escolha Criar política.

Para adicionar uma política em linha à função para o local de resultados das consultas do Athena

1. Na página Summary (Resumo) da função `Athena-LakeFormation-OktaRole`, na guia Permissions (Permissões), escolha Add inline policy (Adicionar política em linha).
2. Na página Create policy (Criar política), escolha JSON.
3. Adicione uma política em linha, semelhante à mostrada abaixo, que permite o acesso da função ao local de resultados das consultas do Athena. Substitua os espaços reservados *`<athena-query-results-bucket>`* no exemplo pelo nome do seu bucket do Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AthenaQueryResultsPermissionsForS3",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:PutObject",
                "s3:GetObject"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::<athena-query-results-bucket>",
                "arn:aws:s3:::<athena-query-results-bucket>/*"
            ]
        }
    ]
}
```

4. Escolha Revisar política.
5. Em Name (Nome), insira um nome para a política (por exemplo, **`AthenaQueryResultsInlinePolicy`**).
6. Escolha Criar política.

Na sequência, copie o ARN da função de acesso do Lake Formation e o ARN do provedor SAML criado. Eles serão necessários para você configurar a aplicação SAML do Okta na próxima seção do tutorial.

Para copiar o ARN da função e o ARN do provedor de identidade SAML

1. No console do IAM, na página Summary (Resumo) da função `Athena-LakeFormation-OktaRole`, escolha o ícone Copy to clipboard (Copiar para área de transferência) ao lado de Role ARN (ARN da função). O ARN tem o seguinte formato:

```
arn:aws:iam::<account-id>:role/Athena-LakeFormation-OktaRole
```

2. Salve o ARN completo com segurança para referência futura.
3. No painel de navegação do console do IAM, escolha Identity providers (Provedores de identidade).
4. Escolha o provedor AthenaLakeFormationOkta.
5. Na página Summary (Resumo), escolha o ícone Copy to clipboard (Copiar para área de transferência) ao lado de Provider ARN (ARN do provedor). O Nome de região da Amazon (ARN) deve se parecer com o seguinte:

```
arn:aws:iam::<account-id>:saml-provider/AthenaLakeFormationOkta
```

6. Salve o ARN completo com segurança para referência futura.

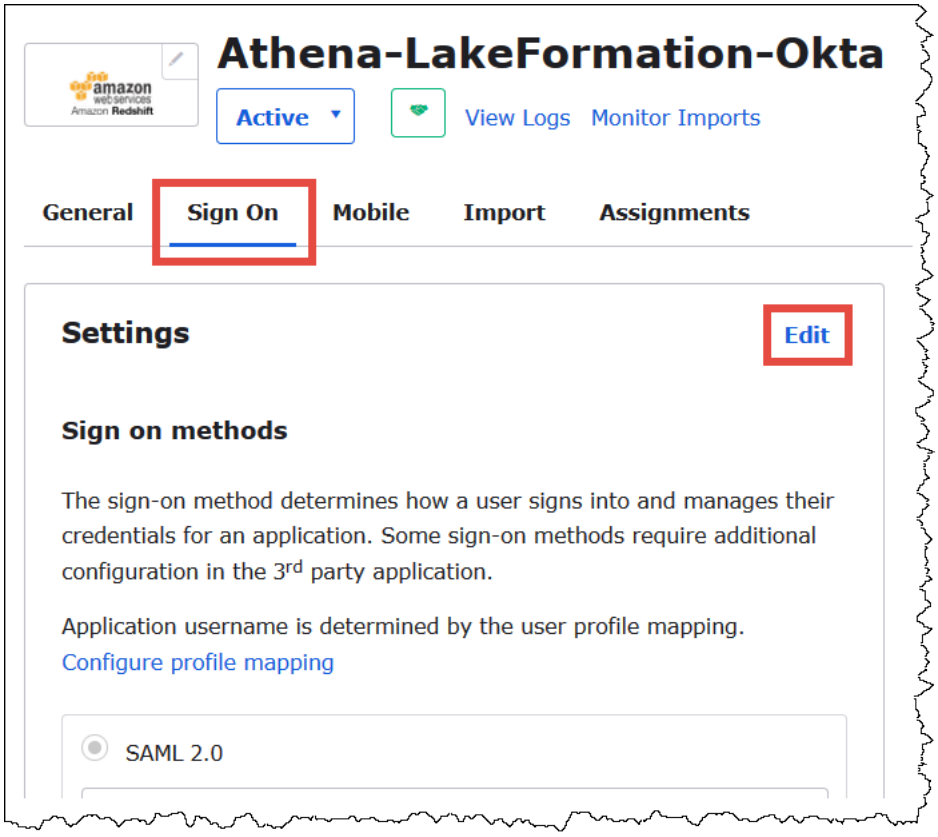
Etapa 5: Adicionar a função do IAM e o provedor de identidade SAML à aplicação do Okta

Nesta etapa, você retorna ao console de desenvolvedor do Okta e executa as seguintes tarefas:

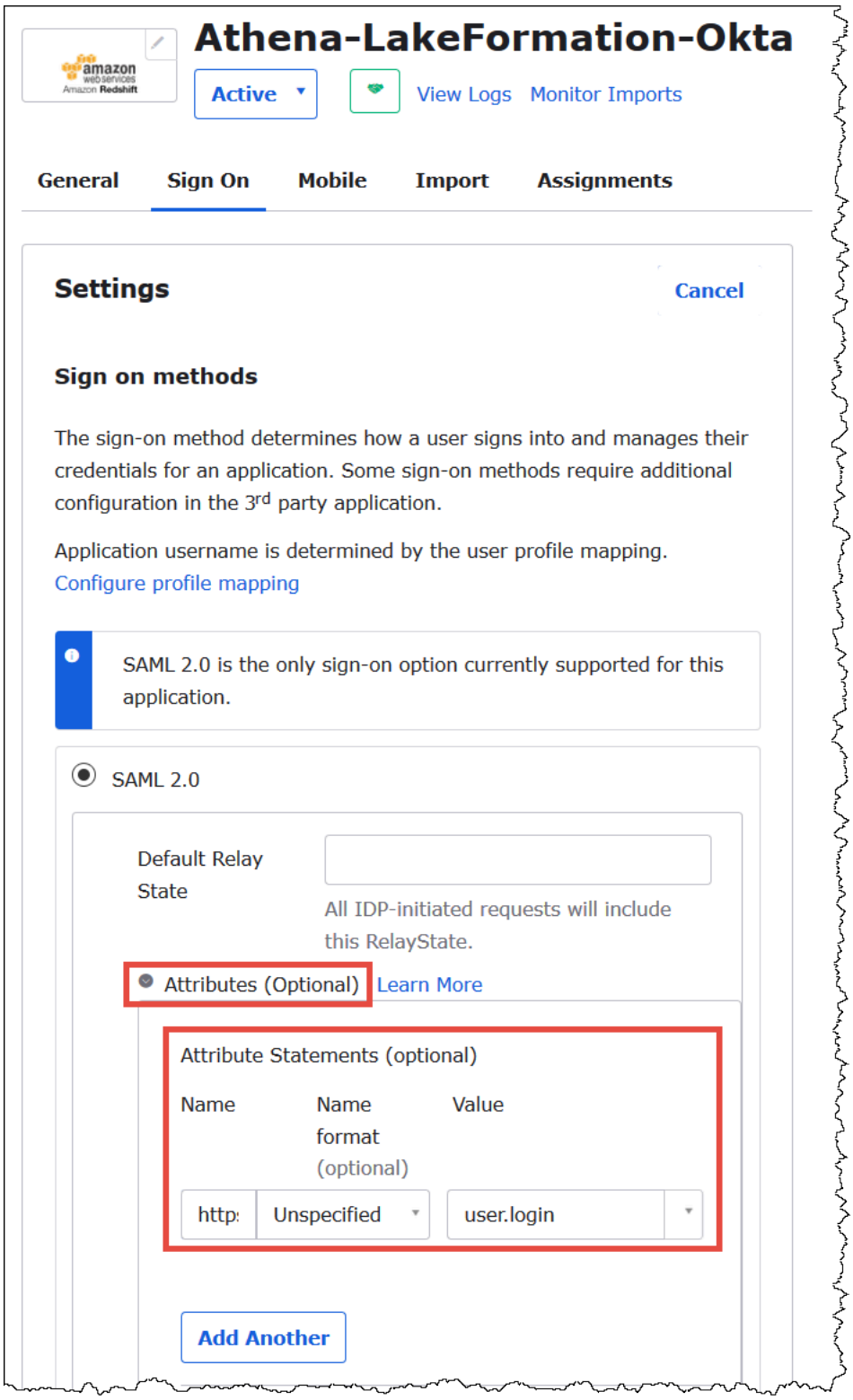
- Adicionar atributos de URL do Lake Formation de usuário e de grupo à aplicação do Okta.
- Adicionar o ARN do provedor de identidade e da função do IAM à aplicação do Okta.
- Copiar o ID da aplicação do Okta. O ID da aplicação do Okta é necessário no perfil JDBC que se conecta ao Athena.

Para adicionar atributos de URL do Lake Formation de usuário e de grupo à aplicação do Okta

1. Faça login no console de desenvolvedor do Okta.
2. Escolha a guia Applications (Aplicações) e selecione a aplicação `Athena-LakeFormation-Okta`.
3. Escolha a guia Sign On (Logon) da aplicação e selecione Edit (Editar).



4. Escolha Attributes (optional) (Atributos (opcional)) para expandi-los.

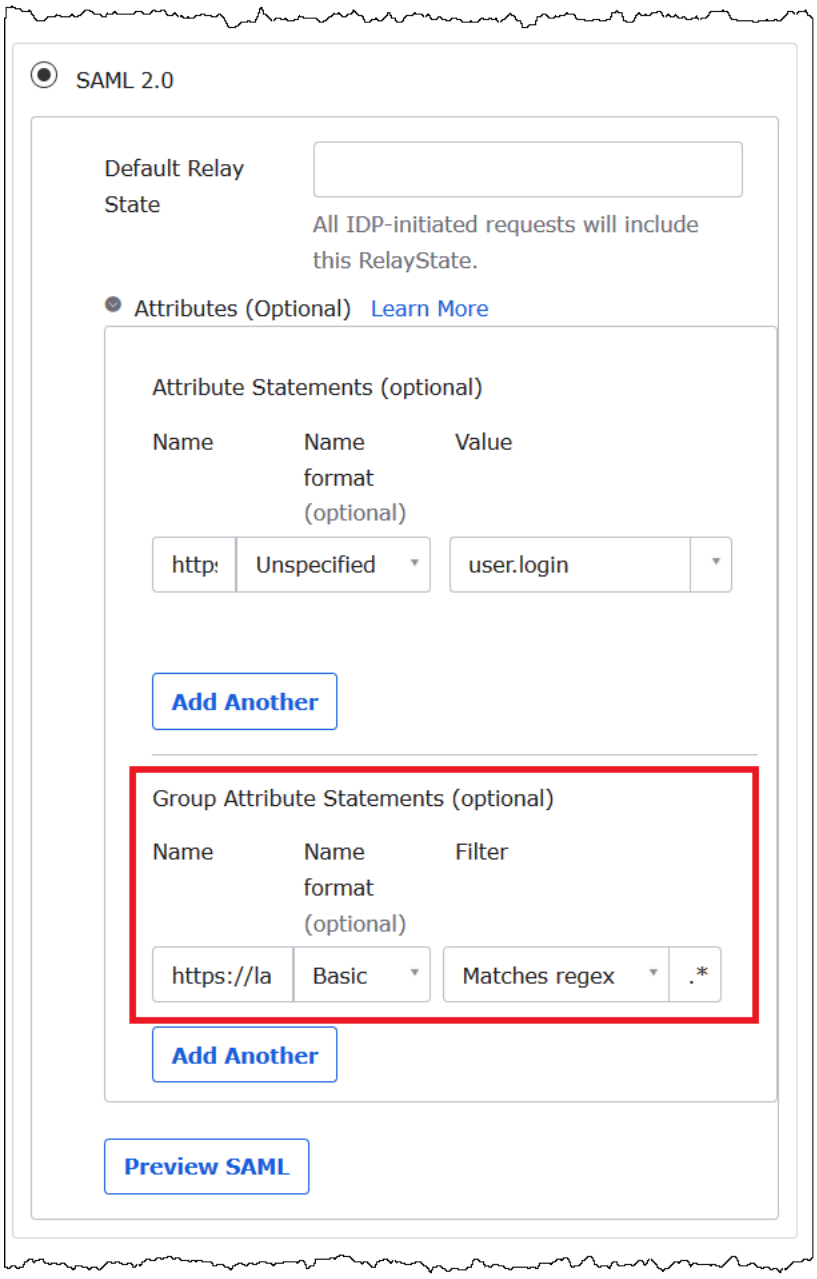


5. Em Attribute Statements (optional) (Instruções do atributo (opcional)), adicione o seguinte atributo:

- Em Nome, digite **`https://lakeformation.amazon.com/SAML/Attributes/Username`**.
- Em Value (Valor), insira **`user.login`**

6. Em Group Attribute Statements (optional) (Instruções do atributo de grupo (opcional)), adicione o seguinte atributo:

   - Em Nome, digite **`https://lakeformation.amazon.com/SAML/Attributes/Groups`**.
   - Em Name format (Formato de nome), insira **`Basic`**
   - Em Filter (Filtro), escolha Matches regex (Corresponde a regex) e insira **`.*`** na caixa do filtro.

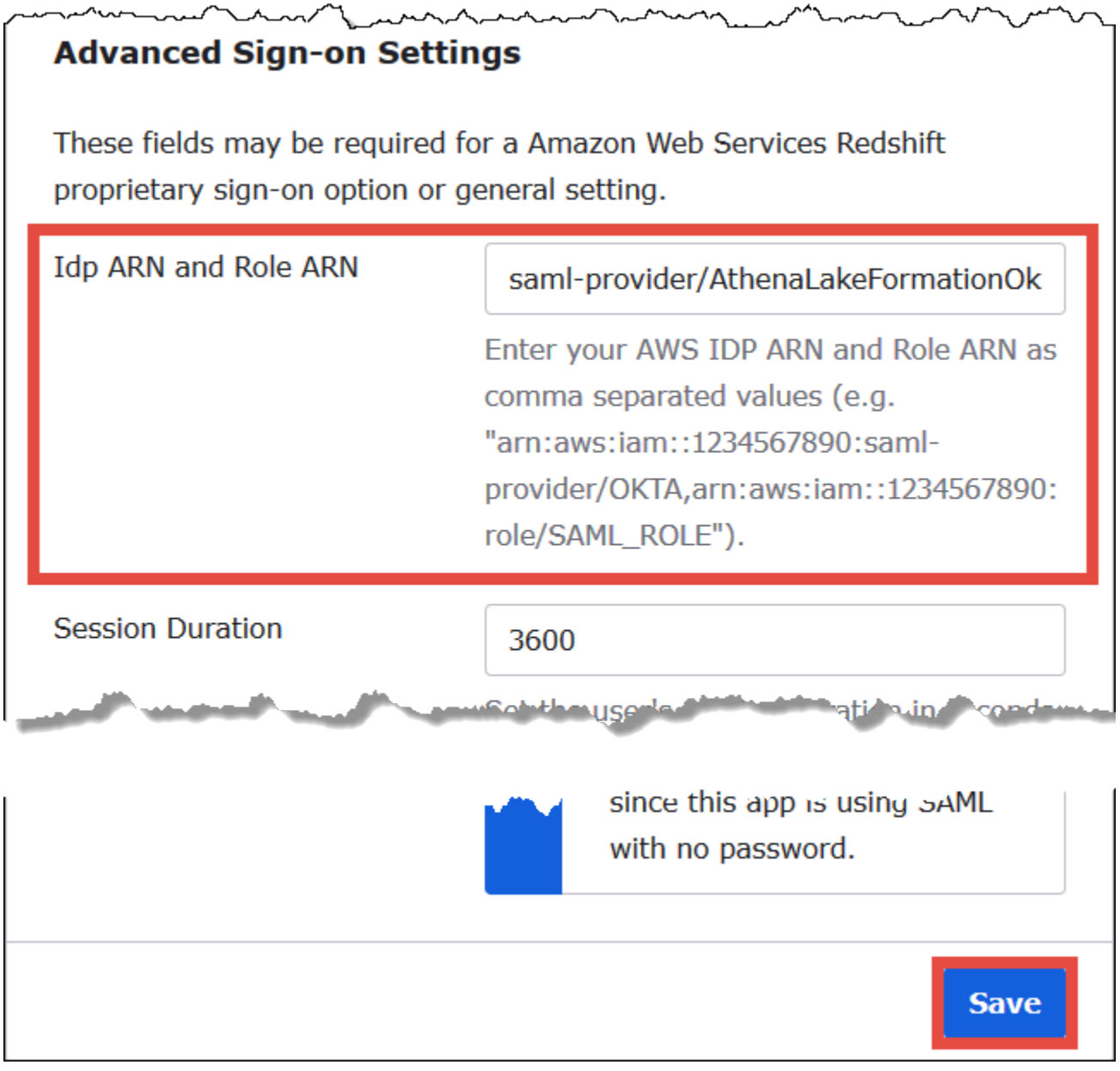


7. Role para baixo até a seção Advanced Sign-On Settings (Configurações avançadas de logon), na qual você adiciona os ARNs do provedor de identidade e da função do IAM à aplicação do Okta.

Para adicionar os ARNs do provedor de identidade e da função do IAM à aplicação do Okta

1. Em Idp ARN and Role ARN (ARN do Idp e ARN da função), insira o ARN do provedor de identidade da AWS e o ARN da função como valores separados por vírgulas no formato *<saml-arn>*,*<role-arn>*. A string combinada deve ter a seguinte aparência:

```
arn:aws:iam::<account-id>:saml-provider/
AthenaLakeFormationOkta,arn:aws:iam::<account-id>:role/Athena-LakeFormation-
OktaRole
```

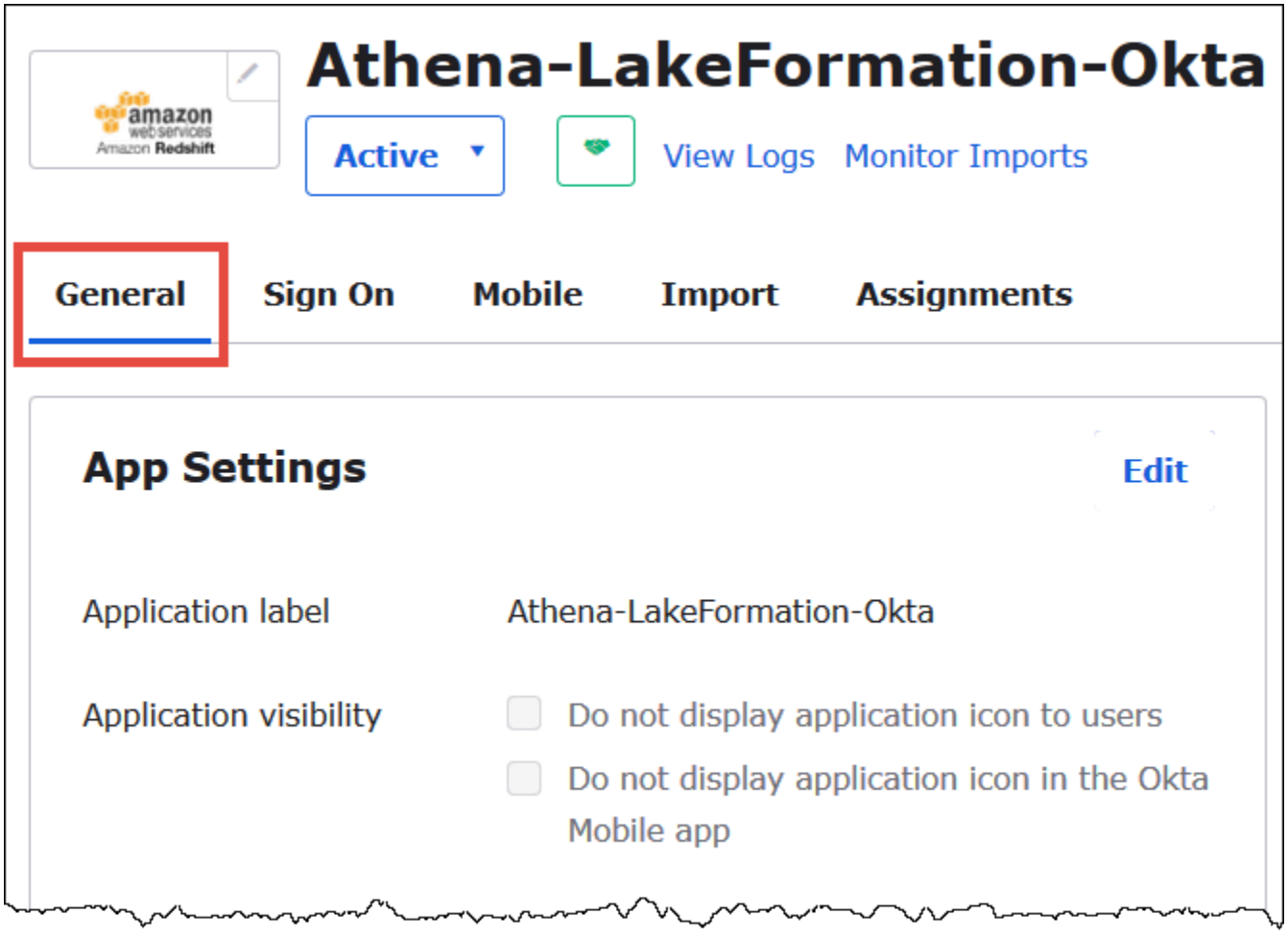


2. Escolha Salvar.

Em seguida, copie o ID da aplicação do Okta. Você precisará dele mais tarde para a string JDBC que se conecta ao Athena.

Para localizar e copiar o ID da aplicação do Okta.

1. Escolha a guia General (Geral) da aplicação do Okta.



2. Rola para baixo até a seção App Embed Link (Link de incorporação de aplicação).
3. Em Embed Link (Link de incorporação), copie e salve a parte do URL da aplicação do Okta com segurança. O ID da aplicação do Okta é a parte do URL que vem depois de `amazon_aws_redshift/`, mas antes da próxima barra. Por exemplo, se o URL incluir `amazon_aws_redshift/aaa/bbb`, o ID da aplicação será `aaa`.

> **Note**
>
> O link de incorporação não pode ser usado para fazer login diretamente no console do Athena para a exibição bancos de dados. As permissões do Lake Formation para usuários e grupos SAML são reconhecidas somente quando você utiliza o driver JDBC ou ODBC para enviar consultas ao Athena. Para exibir os bancos de dados, é possível utilizar usar a ferramenta SQL Workbench/J, que usa o driver JDBC para se conectar ao Athena. A ferramenta SQL Workbench/J é discutida em Etapa 7: Verificar o acesso pelo cliente Athena JDBC.

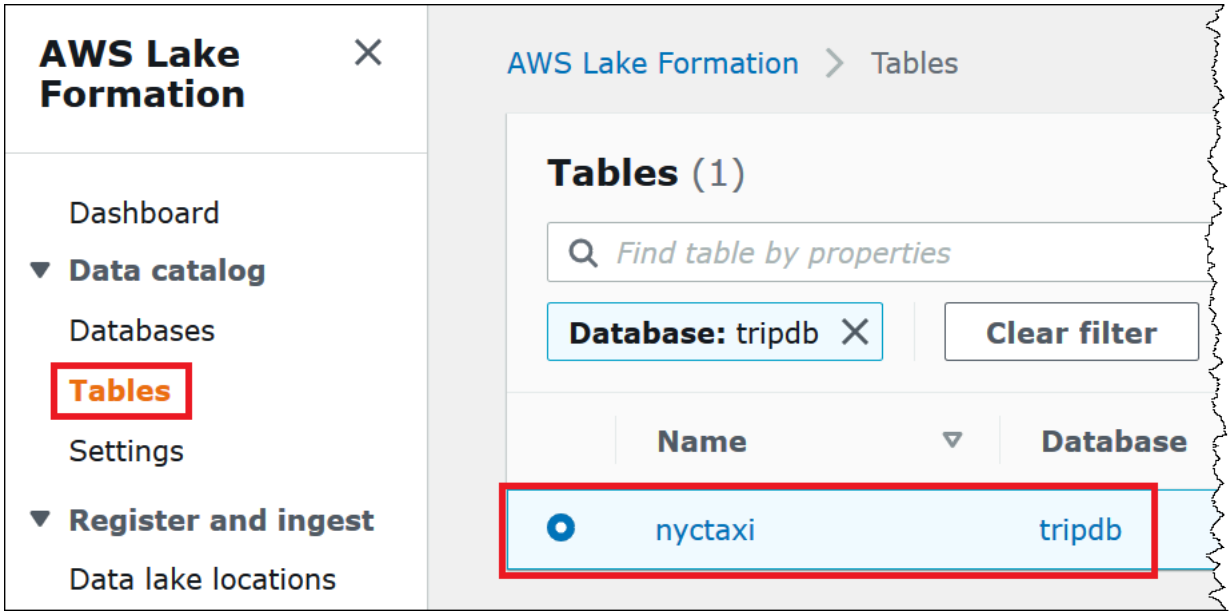
Etapa 6: Conceder permissões de usuário e grupo pelo AWS Lake Formation

Nesta etapa, você usa o console do Lake Formation para conceder permissões em uma tabela ao usuário e grupo do SAML. Você executa as seguintes tarefas:

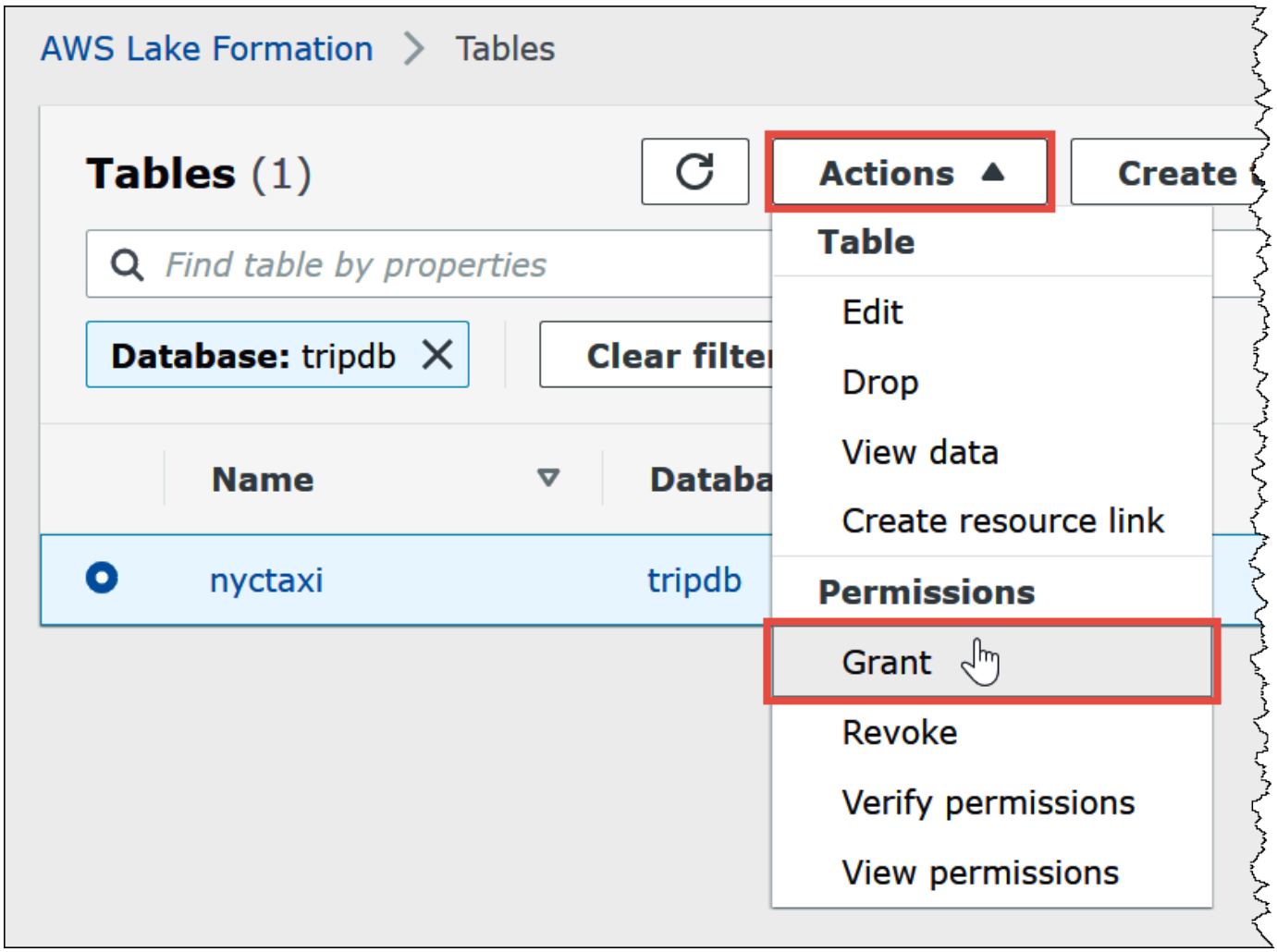
- Especificar o ARN do usuário SAML do Okta e as permissões do usuário associadas na tabela.
- Especificar o ARN do grupo SAML do Okta e as permissões do grupo associadas na tabela.
- Verificar as permissões que você concedeu.

Para conceder permissões no Lake Formation para o usuário do Okta

1. Faça login como administrador de data lake no AWS Management Console.
2. Abra o console do Lake Formation em https://console.aws.amazon.com/lakeformation/.
3. No painel de navegação, escolha Tables (Tabelas) e selecione a tabela à qual você deseja conceder as permissões. Este tutorial usa a tabela `nyctaxi` do banco de dados `tripdb`.



4. Em Actions (Ações), escolha Grant (Conceder).



5. Na caixa de diálogo Grant permissions (Conceder permissões), insira as seguintes informações:

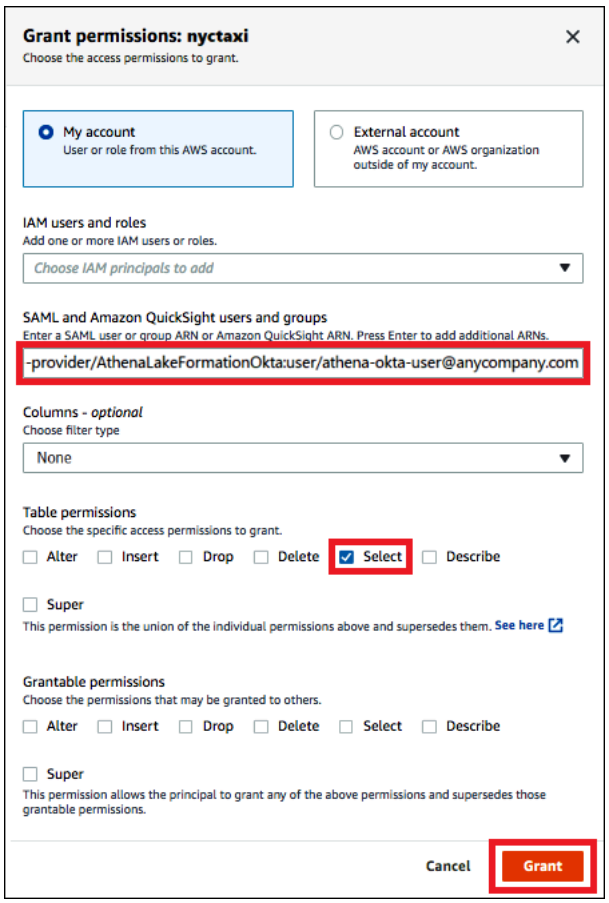
   a. Em SAML and Amazon QuickSight users and groups (Usuários e grupos do SAML e do Amazon QuickSight), insira o ARN do usuário SAML do Okta no seguinte formato:

   ```
   arn:aws:iam::<account-id>:saml-provider/AthenaLakeFormationOkta:user/<athena-okta-user>@<anycompany.com>
   ```

   b. Em Columns (Colunas), Choose filter type (Escolher tipo de filtro), escolha Include columns (Incluir colunas) ou Exclude columns (Excluir colunas) conforme desejado.

   c. Use o menu suspenso Choose one or more columns (Escolher uma ou mais colunas) do filtro para especificar as colunas que você deseja incluir ou excluir do usuário.

   d. Em Table permissions (Permissões de tabela), escolha Select (Selecionar). Este tutorial concede apenas as permissões `SELECT`, seus requisitos podem variar.



6. Escolha Grant (Conceder).

Agora você executa as mesmas etapas para o grupo do Okta.

Para conceder permissões no Lake Formation ao grupo do Okta

1. Na página Tables (Tabelas) do console do Lake Formation, verifique se a tabela nyctaxi ainda está selecionada.
2. Em Actions (Ações), escolha Grant (Conceder).
3. Na caixa de diálogo Grant permissions (Conceder permissões), insira as seguintes informações:

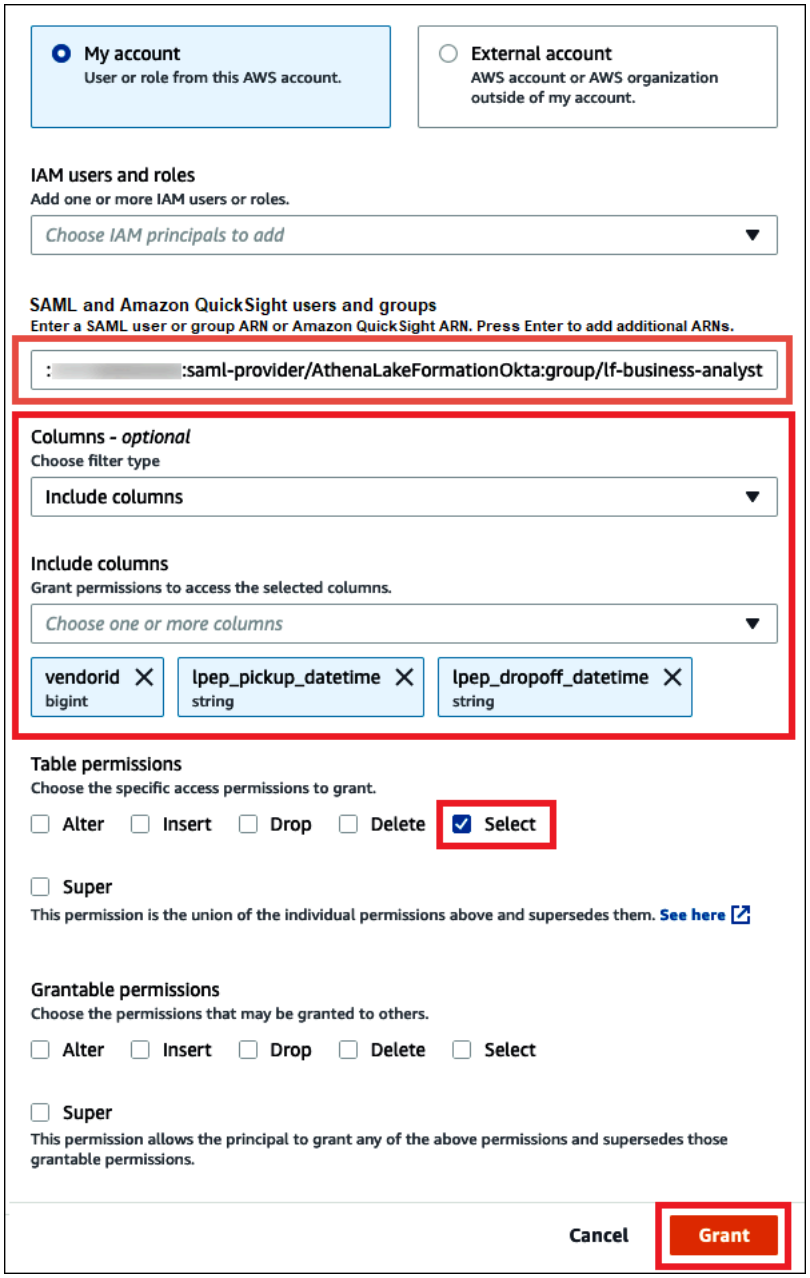
    a. Em SAML and Amazon QuickSight users and groups (Usuários e grupos do SAML e do Amazon QuickSight), insira o ARN do grupo SAML do Okta no seguinte formato:

```
arn:aws:iam::<account-id>:saml-provider/AthenaLakeFormationOkta:group/lf-business-analyst
```

b. Em Columns (Colunas), Choose filter type (Escolher tipo de filtro), escolha Include columns (Incluir colunas).

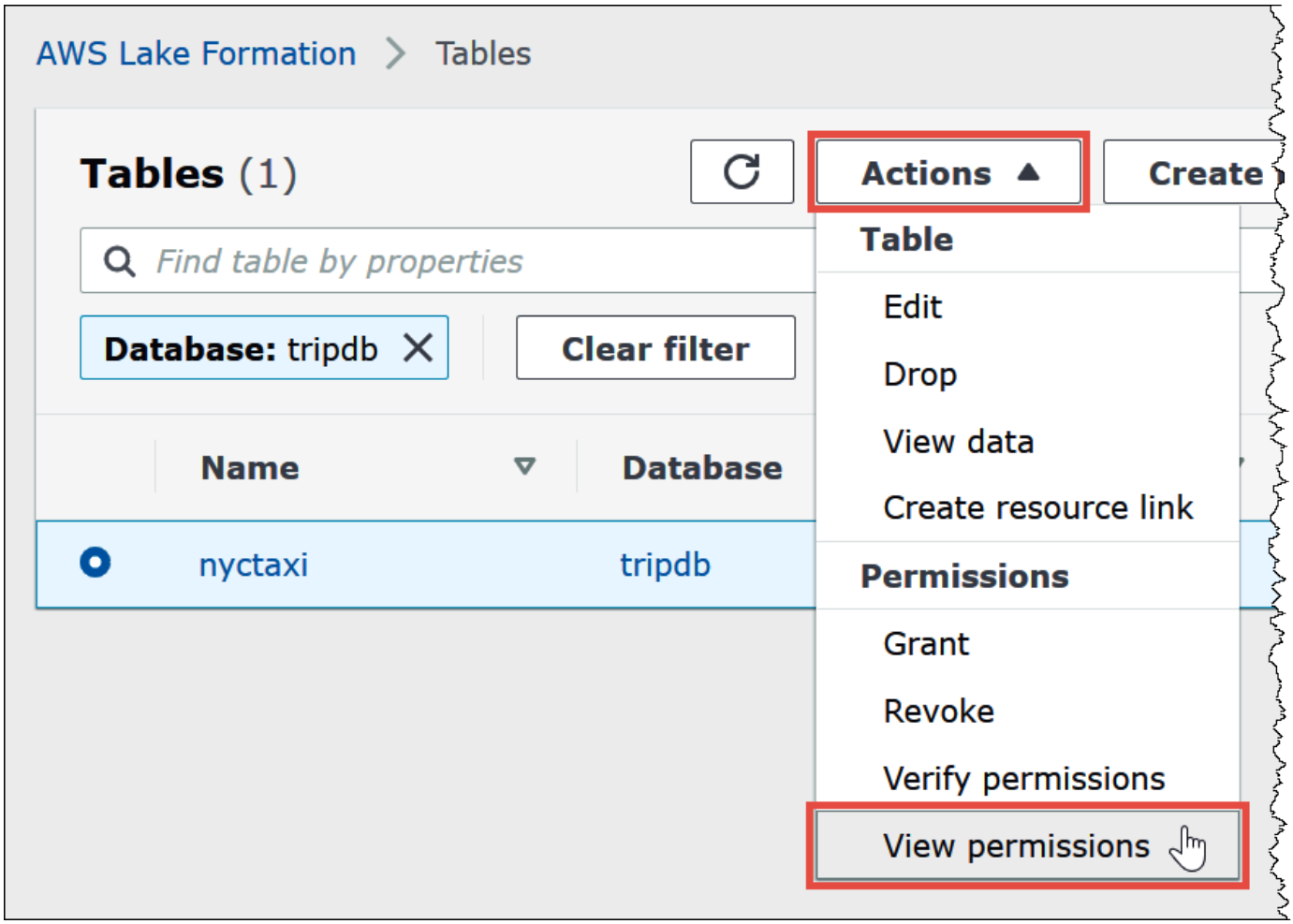
c. Em Choose one or more columns (Escolher uma ou mais colunas), escolha as primeiras três colunas da tabela.

d. Em Table permissions (Permissões de tabela), escolha as permissões de acesso específicas para conceder. Este tutorial concede apenas as permissões `SELECT`, seus requisitos podem variar.

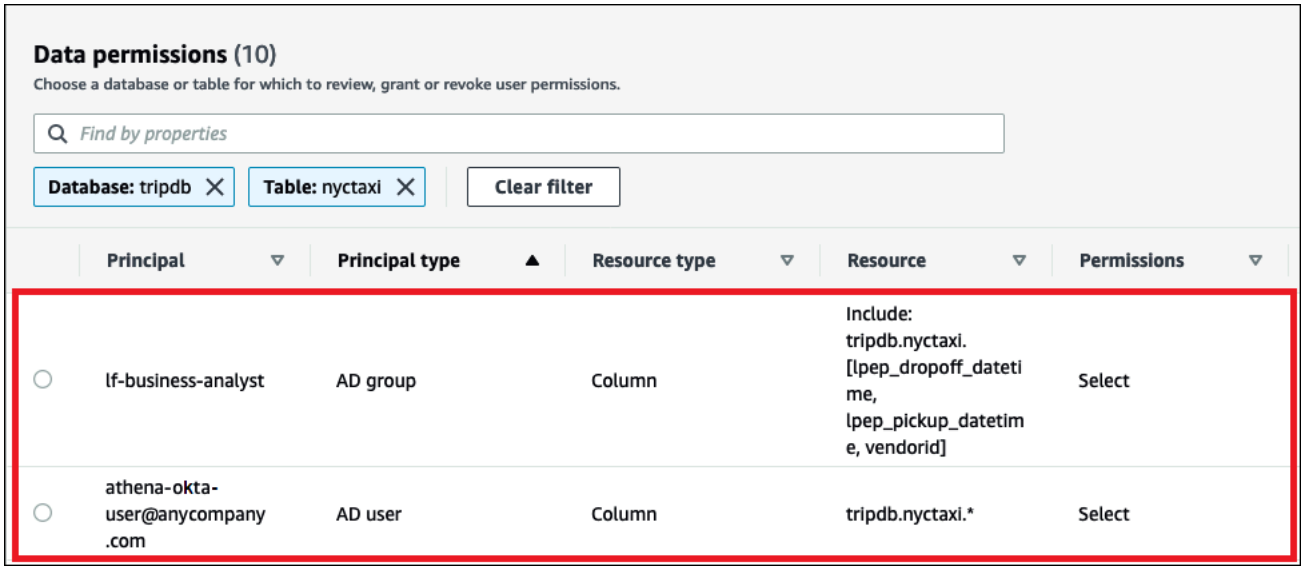


4. Escolha Grant (Conceder).

5. Para verificar as permissões concedidas, escolha Actions (Ações), View permissions (Visualizar permissões).



A página Data permissions (Permissões de dados) da tabela `nyctaxi` mostra as permissões do usuário athena-okta-user e do grupo lf-business-analyst.



Etapa 7: Verificar o acesso pelo cliente Athena JDBC

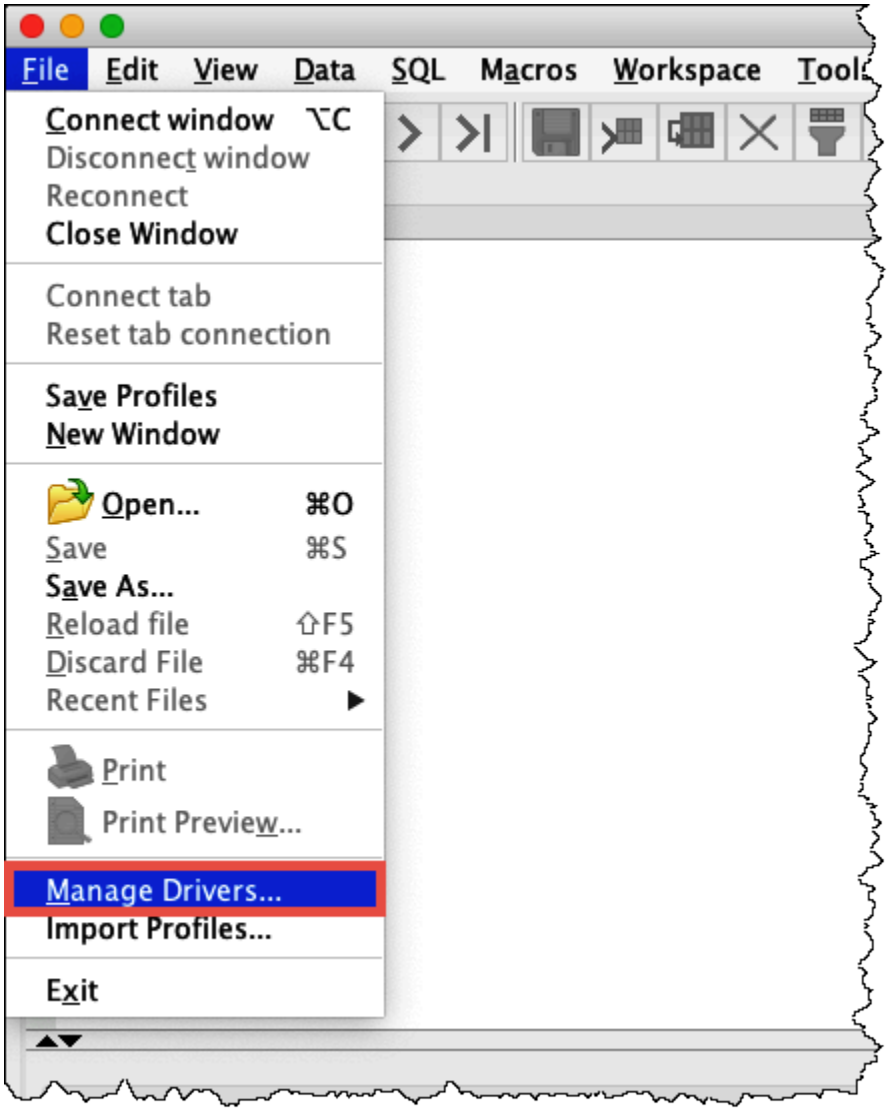
Agora você está pronto para usar um cliente JDBC para executar uma conexão de teste com o Athena como usuário SAML do Okta.

Nesta seção, você pode executar as seguintes tarefas:

- Preparar o cliente de teste: baixe o driver Athena JDBC, instale o SQL Workbench e adicione o driver ao Workbench. Este tutorial usa o SQL Workbench para acessar o Athena por meio da autenticação do Okta e verificar as permissões do Lake Formation.
- No SQL Workbench:
  - Crie uma conexão para o usuário do Okta no Athena.
  - Execute as consultas de teste como o usuário do Okta no Athena.
  - Crie e teste uma conexão para o usuário “analista de negócios”.
- No console do Okta, adicione o usuário “analista de negócios” ao grupo de desenvolvedores.
- No console do Lake Formation, configure as permissões de tabela para o grupo de desenvolvedores.
- No SQL Workbench, execute as consultas de teste como o usuário “analista de negócios” e verifique como a alteração nas permissões afeta os resultados.

Para preparar o cliente de teste

1. Baixe e extraia o driver Athena JDBC compatível com o Lake Formation (2.0.14 ou versão mais recente) em Conectar-se ao Amazon Athena com JDBC.
2. Baixe e instale a ferramenta de consulta SQL gratuita SQL Workbench/J, disponível na licença modificada do Apache 2.0.
3. No SQL Workbench, escolha File (Arquivo) e Manage Drivers (Gerenciar drivers).



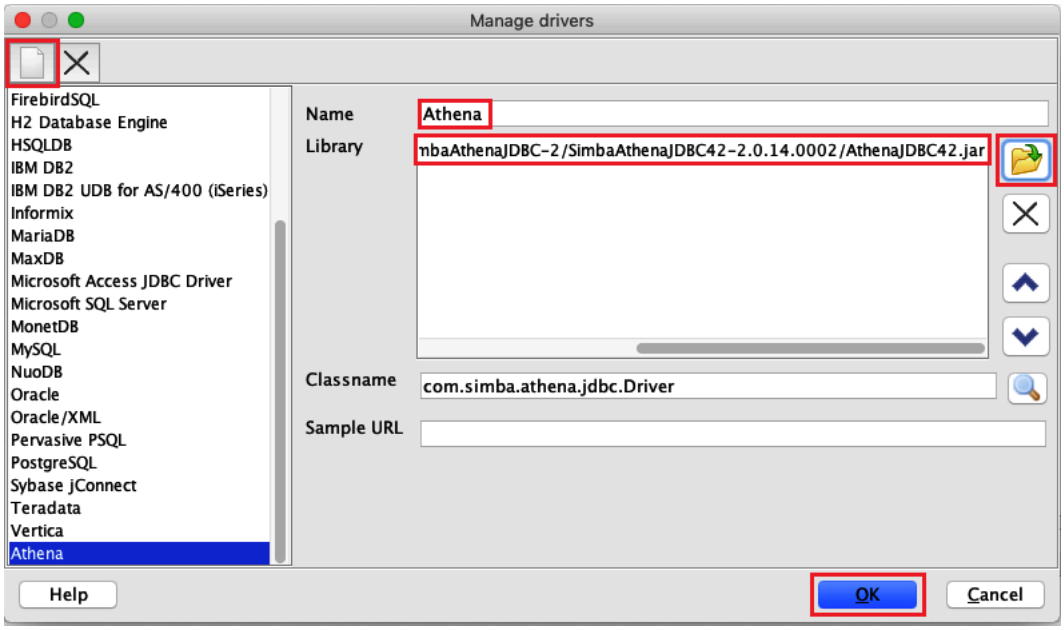
4. Na caixa de diálogo Manage Drivers (Gerenciar drivers), execute as seguintes etapas:

   a. Escolha o ícone do novo driver.

   b. Em Name (Nome), insira **`Athena`**.

   c. Em Library (Biblioteca), procure e escolha o arquivo `.jar` do Simba Athena JDBC que você acabou de baixar.

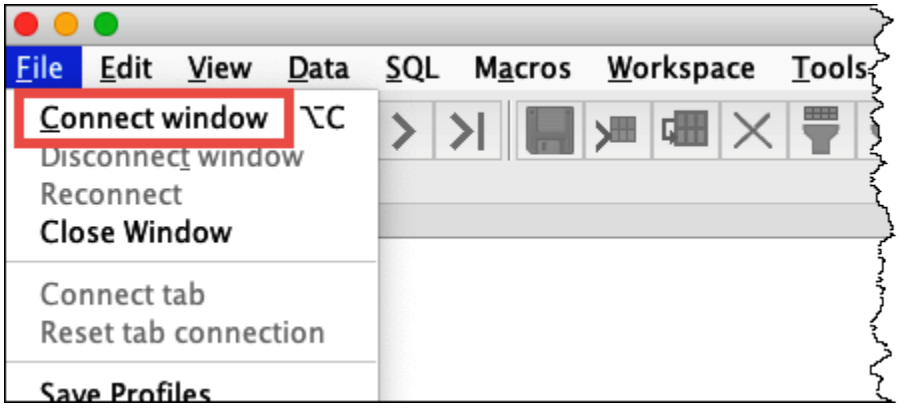
d. Escolha OK.



Agora você está pronto para criar e testar uma conexão para o usuário do Okta no Athena.

Para criar uma conexão para o usuário do Okta

1. Escolha File (Arquivo), Connect window (Conectar janela).



2. Na caixa de diálogo Connection profile (Perfil de conexão), crie uma conexão inserindo as seguintes informações:

   - Na caixa do nome, insira **`Athena_Okta_User_Connection`**.
   - Em Driver, escolha o driver Simba Athena JDBC.
   - Em URL, realize um destes procedimentos:

- Para usar um URL de conexão, insira uma string de conexão de linha única. O exemplo a seguir contém quebras de linha para facilitar a leitura.

```
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;
```

- Para usar um URL baseado no perfil da AWS, execute as seguintes etapas:

  1. Configure um perfil da AWS que tenha um arquivo de credenciais da AWS, como mostrado no exemplo a seguir.

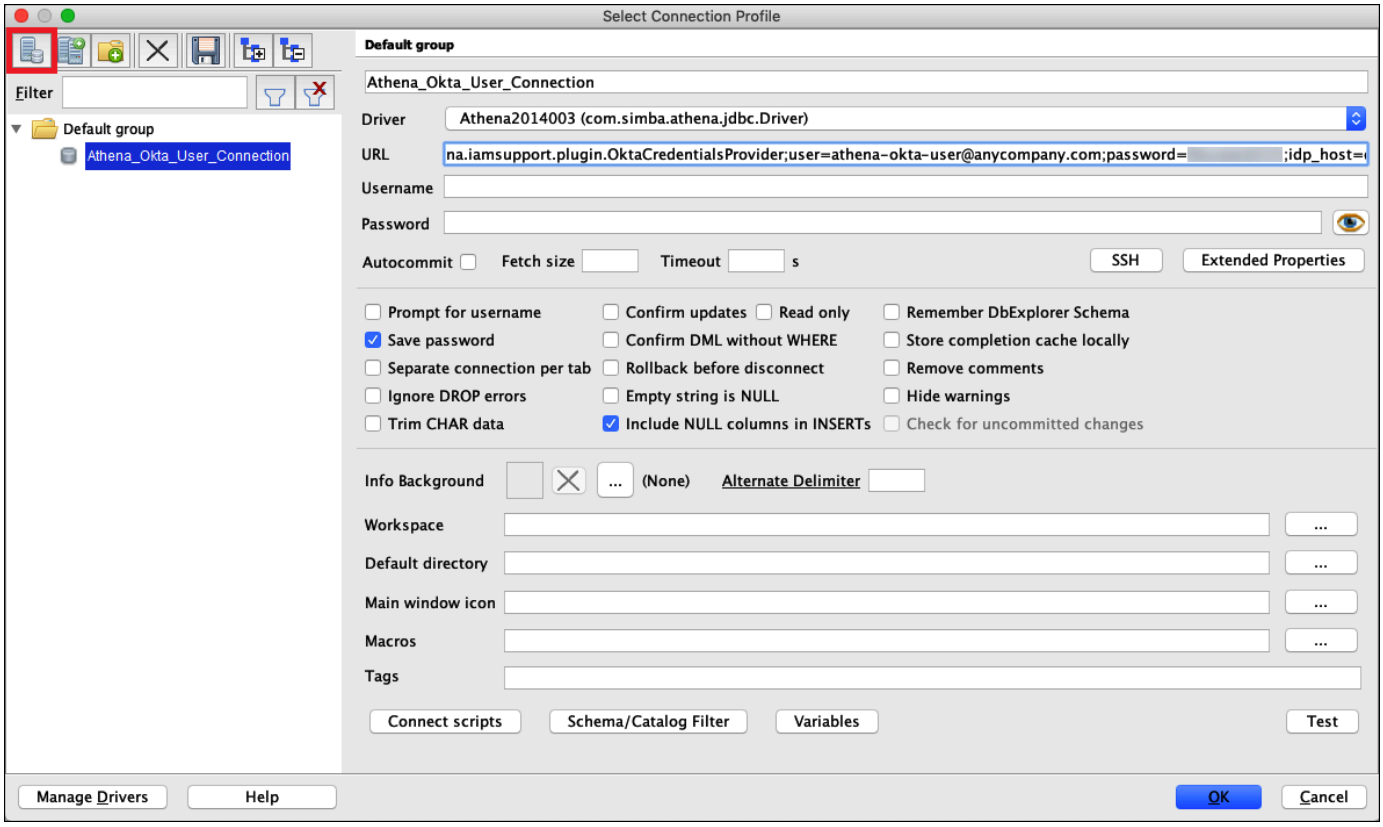
```
[athena_lf_dev]
plugin_name=com.simba.athena.iamsupport.plugin.OktaCredentialsProvider
idp_host=okta-idp-domain
app_id=okta-app-id
uid=athena-okta-user@anycompany.com
pwd=password
```

  2. Em URL, insira uma string de conexão de linha única, como mostrado no exemplo a seguir. O exemplo contém quebras de linha para facilitar a leitura.

```
jdbc:awsathena://AwsRegion=region-id;
S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/athena_results;
profile=athena_lf_dev;
SSL_Insecure=true;
LakeFormationEnabled=true;
```

Observe que esses exemplos são representações básicas do URL necessário para se conectar ao Athena. Para ver a lista completa dos parâmetros permitidos na URL, consulte a documentação do JDBC.

A imagem a seguir mostra um perfil de conexão do SQL Workbench que usa um URL de conexão.

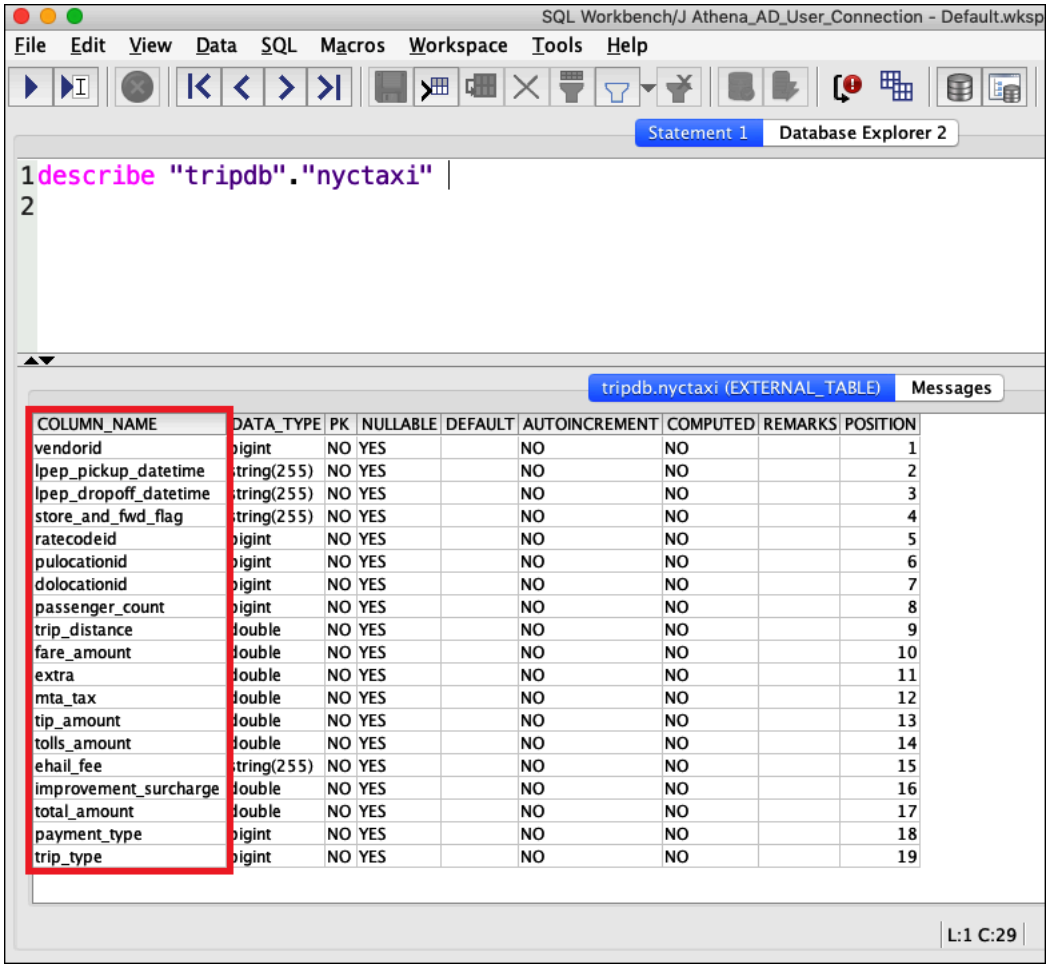


Agora que você estabeleceu uma conexão para o usuário do Okta, pode testá-la recuperando alguns dados.

Para testar a conexão do usuário do Okta

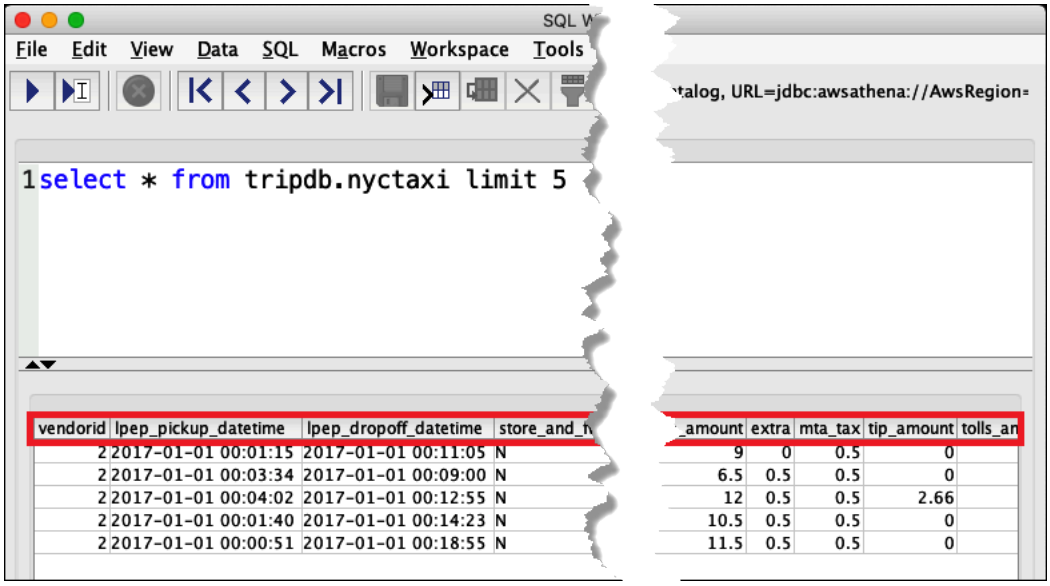
1. Escolha Test (Testar) e verifique se a conexão foi estabelecida.
2. Na janela Statement (Instrução) do SQL Workbench, execute o comando SQL `DESCRIBE` a seguir. Verifique se todas as colunas são exibidas.

```
DESCRIBE "tripdb"."nyctaxi"
```



3. Na janela Statement (Instrução) do SQL Workbench, execute o comando SQL `SELECT` a seguir. Verifique se todas as colunas são exibidas.

```
SELECT * FROM tripdb.nyctaxi LIMIT 5
```



Na sequência, verifique se o athena-ba-user, como membro do grupo lf-business-analyst, tem acesso apenas às primeiras três colunas da tabela que você especificou anteriormente no Lake Formation.

Para verificar o acesso do athena-ba-user

1. No SQL Workbench, na caixa de diálogo Connection profile (Perfil de conexão), crie outro perfil de conexão.
   - Para o nome do perfil de conexão, insira **`Athena_Okta_Group_Connection`**.
   - Em Driver, escolha o driver Simba Athena JDBC.
   - Em URL, realize um destes procedimentos:
     - Para usar um URL de conexão, insira uma string de conexão de linha única. O exemplo a seguir contém quebras de linha para facilitar a leitura.

```
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;
```

- Para usar um URL baseado no perfil da AWS, execute as seguintes etapas:

  1. Configure um perfil da AWS que tenha um arquivo de credenciais, como mostrado no exemplo a seguir.

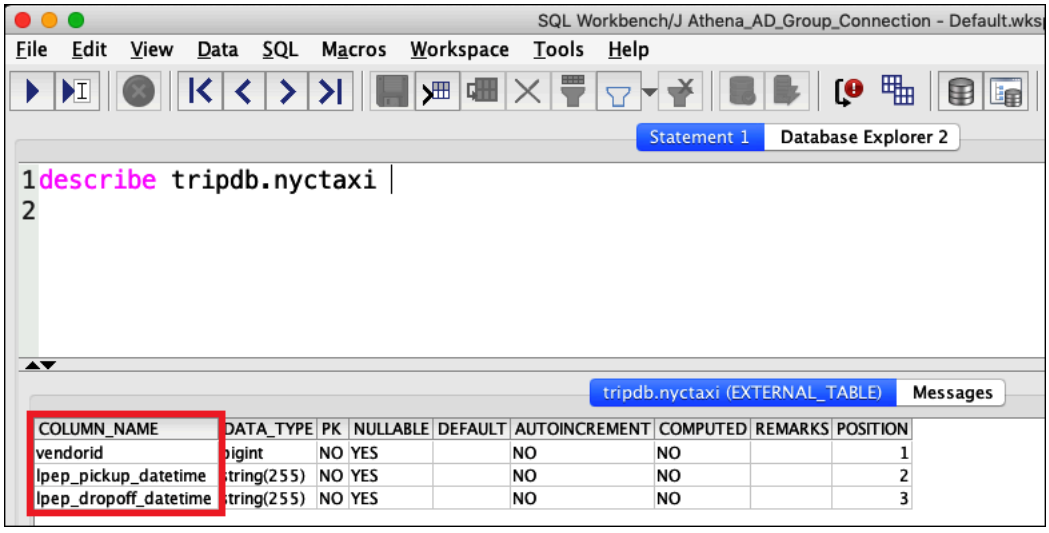
     ```
     [athena_lf_ba]
     plugin_name=com.simba.athena.iamsupport.plugin.OktaCredentialsProvider
     idp_host=okta-idp-domain
     app_id=okta-application-id
     uid=athena-ba-user@anycompany.com
     pwd=password
     ```

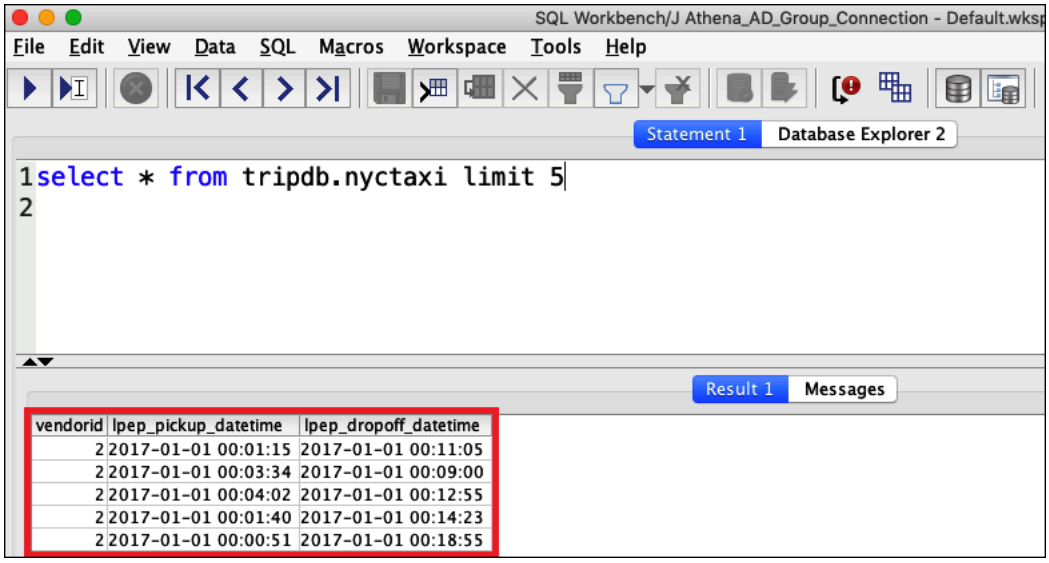
  2. Em URL, insira uma string de conexão de linha única, como mostrado a seguir. O exemplo contém quebras de linha para facilitar a leitura.

     ```
     jdbc:awsathena://AwsRegion=region-id;
     S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/athena_results;
     profile=athena_lf_ba;
     SSL_Insecure=true;
     LakeFormationEnabled=true;
     ```

2. Escolha Test (Testar) para confirmar se a conexão foi estabelecida.
3. Na janela SQL Statement (Instrução SQL), execute os mesmos comandos SQL `DESCRIBE` e `SELECT` de antes e confira os resultados.

   Como o athena-ba-user é membro do grupo lf-business-analyst, apenas as três primeiras colunas que você especificou no console do Lake Formation são retornadas.

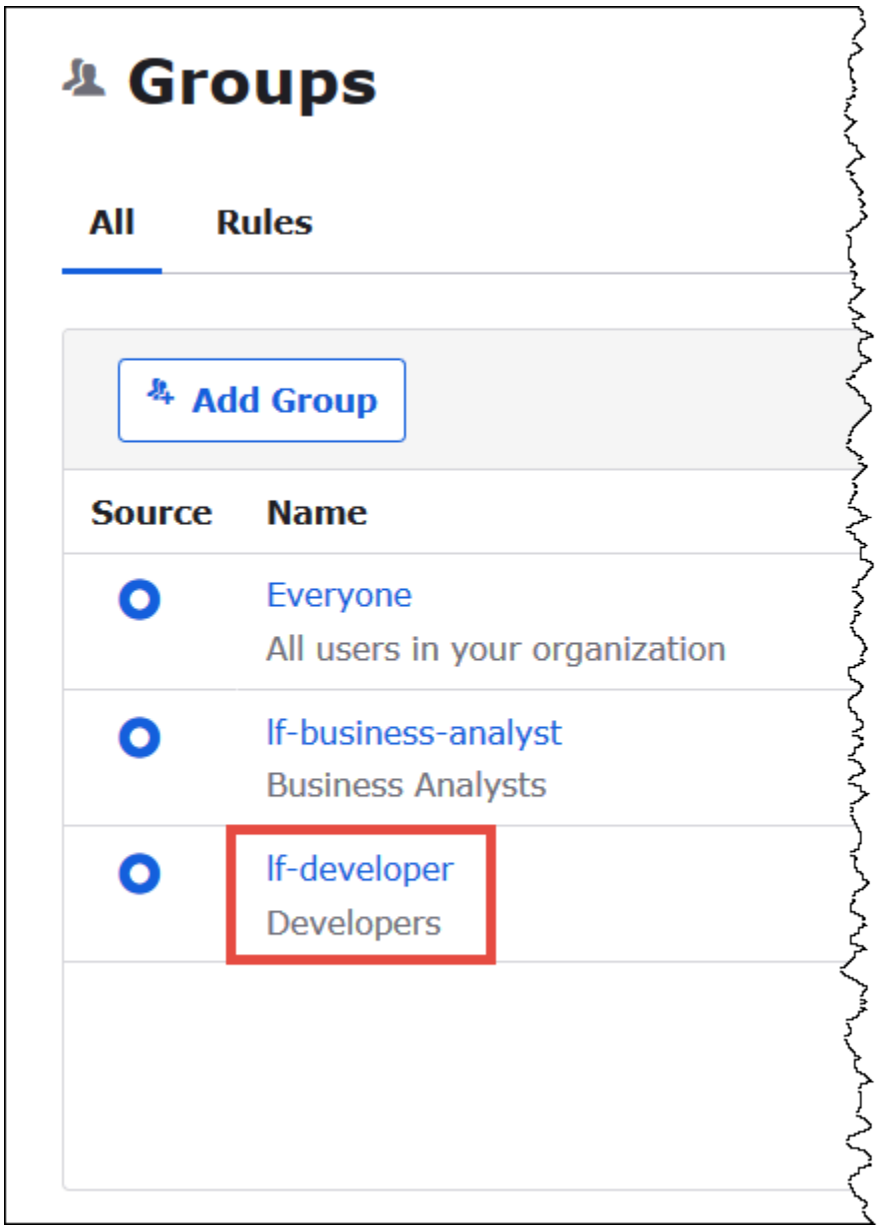




Depois disso, retorne ao console do Okta para adicionar o `athena-ba-user` ao grupo do Okta `lf-developer`.

Para adicionar o athena-ba-user ao grupo lf-developer

1. Faça login no console do Okta como usuário administrativo do domínio do Okta atribuído.
2. Escolha Directory (Diretório) e Groups (Grupos).
3. Na página Groups (Grupos), escolha o grupo lf-developer.



4. Escolha Manage People (Gerenciar pessoas).
5. Na lista Not Members (Não membros), escolha o athena-ba-user para adicionar ao grupo lf-developer.
6. Escolha Salvar.

Agora você retorna ao console do Lake Formation para configurar as permissões de tabela do grupo lf-developer.

Para configurar as permissões de tabela do grupo lf-developer

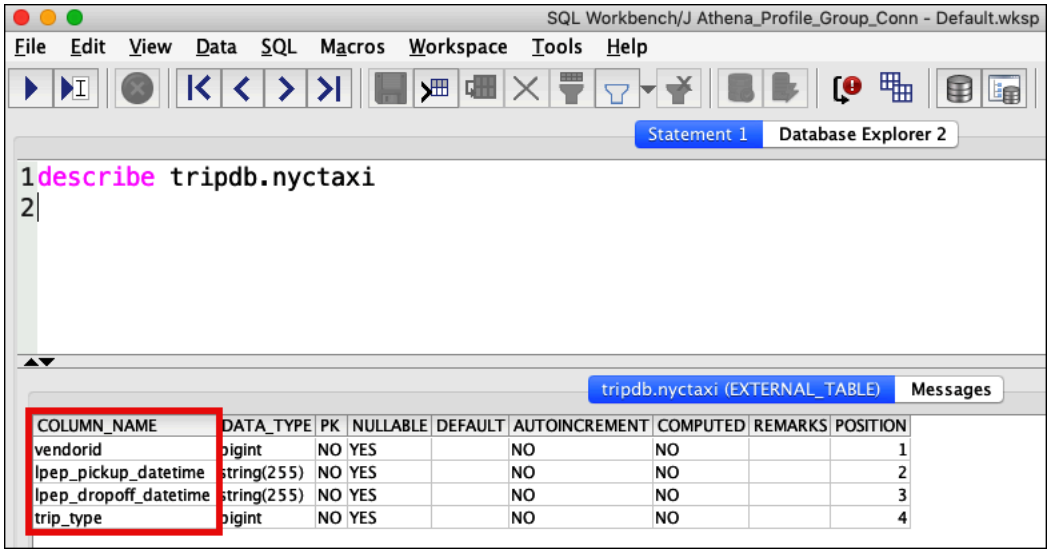
1. Faça login no console do Lake Formation como administrador do data lake.
2. No painel de navegação, selecione Tabelas.
3. Selecione a tabela nyctaxi.
4. Escolha Actions (Ações), Grant (Conceder).
5. Na caixa de diálogo Grant Permissions (Conceder permissões), insira as seguintes informações:
   - Em SAML and Amazon QuickSight users and groups (Usuários e grupos do SAML e do Amazon QuickSight), insira o ARN do grupo SAML lf-developer do Okta no seguinte formato:
   - Em Columns (Colunas), Choose filter type (Escolher tipo de filtro), escolha Include columns (Incluir colunas).
   - Escolha a coluna trip_type.
   - Em Table permissions (Permissões de tabela), escolha SELECT.
6. Escolha Grant (Conceder).

Agora você pode usar o SQL Workbench para verificar a alteração nas permissões do grupo lf-developer. A alteração deve ser refletida nos dados disponíveis ao athena-ba-user, que agora é membro do grupo lf-developer.

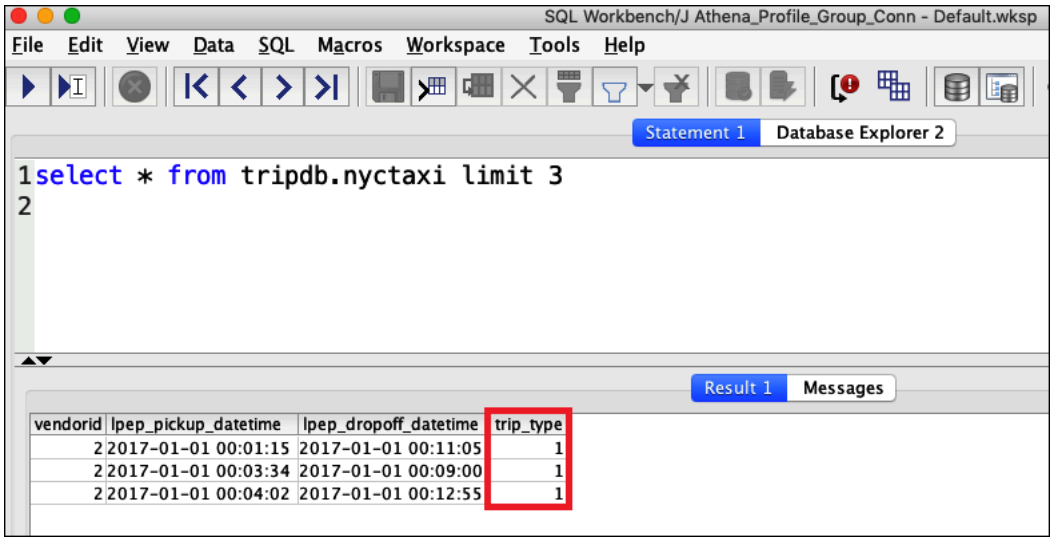
Para verificar a alteração nas permissões do athena-ba-user

1. Feche e reabra o programa SQL Workbench.
2. Conecte-se ao perfil do athena-ba-user.
3. Na janela Statement (Instrução), emita as mesmas instruções SQL que você executou antes:

   Desta vez, a coluna trip_type é exibida.



Como o athena-ba-user agora é membro dos dois grupos lf-developer e lf-business-analyst, a combinação das permissões do Lake Formation desses grupos determina as colunas que são retornadas.



Conclusão

Neste tutorial, você configurou a integração do Athena com o AWS Lake Formationusando o Okta como provedor SAML. Você usou o Lake Formation e o IAM para controlar os recursos disponíveis ao usuário SAML no Catálogo de dados do AWS Glue do seu data lake.

Recursos relacionados

Para obter informações relacionadas, consulte os recursos a seguir.

- Conectar-se ao Amazon Athena com JDBC
- Permitir acesso federado à API do Athena
- AWS Lake Formation Guia do desenvolvedor
- Granting and revoking Data Catalog permissions (Conceder e revogar permissões de catálogo de dados) no Guia do desenvolvedor do AWS Lake Formation.
- Provedores de identidade e federação no Guia do usuário do IAM.
- Criação de provedores de identidade SAML do IAM no Guia do usuário do IAM.
- Habilitar a federação na AWS usando Windows Active Directory, ADFS e SAML 2.0 no blog de segurança da AWS.

Gerenciamento do workload

Você pode usar os recursos de grupo de trabalho, gerenciamento de capacidade, ajuste de performance, suporte à compactação, etiquetas e cotas de serviço do Athena para gerenciar sua workload.

Tópicos

- Usar grupos de trabalho para controlar o acesso a consultas e os custos
- Como gerenciar a capacidade de processamento de consulta
- Ajuste de performance no Athena
- Suporte a compactação no Athena
- Etiquetar recursos do Athena
- Service Quotas

Usar grupos de trabalho para controlar o acesso a consultas e os custos

Use grupos de trabalho para separar usuários, equipes, aplicativos ou cargas de trabalho, para definir limites de quantidade de dados que cada consulta ou todo o grupo de trabalho pode processar e para acompanhar os custos. Como os grupos de trabalho atuam como recursos, você pode usar políticas baseadas em identidade no nível de recurso para controlar o acesso a um grupo de trabalho específico. Você também pode visualizar as métricas relacionadas à consulta no Amazon CloudWatch, controlar os custos configurando limites para a quantidade de dados verificada, criar limites e acionar ações, como o Amazon SNS, quando esses limites são violados.

Para controlar ainda mais os custos, é possível criar reservas de capacidade com o número de unidades de processamento de dados que você especificar e adicionar um ou mais grupos de trabalho à reserva. Para ter mais informações, consulte Como gerenciar a capacidade de processamento de consulta.

Os grupos de trabalho se integram ao IAM, ao CloudWatch, ao Amazon Simple Notification Service e aos Relatórios de custos e uso da AWS da seguinte forma:

- As políticas baseadas em identidade do IAM com permissões no nível do recurso controlam quem pode executar consultas em um grupo de trabalho.
- O Athena publicará as métricas de consulta do grupo de trabalho no CloudWatch se você habilitá-las.
- No Amazon SNS, você pode criar tópicos do Amazon SNS que emitem alarmes para usuários especificados do grupo de trabalho quando os controles de uso dos dados para consultas em um grupo de trabalho excedem os limites estabelecidos.
- Quando você aplica tags a um grupo de trabalho com uma tag configurada como de alocação de custos no console do Billing and Cost Management, os custos associados à execução de consultas nesse grupo de trabalho aparecerão em seu Relatório de custos e uso com essa tag de alocação de custos.

Tópicos

- Usar grupos de trabalho para executar consultas
- Controlar custos e monitorar consultas com métricas e eventos do CloudWatch

Consulte também a publicação no blog de big data da AWS: Consultas separadas e gerenciamento de custos usando grupos de trabalho do Amazon Athena (em inglês), que mostra como usar grupos de trabalho para separar workloads, controlar o acesso de usuários e gerenciar o uso e os custos das consultas.

Usar grupos de trabalho para executar consultas

Recomendamos usar grupos de trabalho para isolar consultas para equipes, aplicativos ou cargas de trabalho diferentes. Por exemplo, você pode criar grupos separados para duas equipes diferentes na sua organização. Você também pode separar cargas de trabalho. Por exemplo, você pode criar dois grupos de trabalho independentes: um para aplicativos programados automatizados, como geração de relatórios, e outro para uso ad-hoc por analistas. Você pode alternar entre os grupos de trabalho.

Tópicos

- Benefícios do uso de grupos de trabalho
- Como funcionam os grupos de trabalho
- Configurar grupos de trabalho
- Políticas do IAM para acessar grupos de trabalho
- Configurações do grupo de trabalho
- Gerenciar grupos de trabalho
- Uso de grupos de trabalho do Athena habilitados para o Centro de Identidade do IAM
- APIs de grupos de trabalho do Athena
- Resolver problemas nos grupos de trabalho

Benefícios do uso de grupos de trabalho

Os grupos de trabalho permitem que você:

Isole usuários, equipes, aplicativos ou cargas de trabalho em grupos.	Cada grupo de trabalho tem seu próprio histórico de consulta distinto e uma lista de consultas salvas. Para ter mais informações, consulte Como funcionam os grupos de trabalho. Para todas as consultas no grupo de trabalho, você pode optar por fazer as configurações dele. Elas incluem uma localização do Amazon S3 para o armazenamento de resultados de consultas , o proprietário do bucket esperado, a criptografia e o controle de objetos gravados no bucket de resultados de consultas. Você também pode aplicar as configurações do grupo de trabalho. Para ter mais informações, consulte Configurações do grupo de trabalho.
Imponha restrições de custos.	Você pode definir dois tipos de restrições de custo para as consultas de um grupo de trabalho: • Limite por consulta é um limite para a quantidade de dados verificada em cada consulta. O Athena cancela as consultas quando excedem o limite especificado. O limite se aplica a cada consulta em execução dentro de um grupo de trabalho. Você pode definir apenas um limite por consulta e atualizá-lo, se necessário.

	• O Per-workgroup limit (Limite por grupo de trabalho) é um limite que você pode definir para cada grupo de trabalho para a quantidade de dados verificada por consultas no grupo de trabalho. Violar um limite ativa um alarme do Amazon SNS que aciona uma ação de sua escolha, como enviar um e-mail para um usuário especificado. Você pode definir vários limites por grupo de trabalho para cada grupo de trabalho. Para obter detalhes das etapas, consulte, Definir limites de controle de uso de dados.
Acompanhe as métricas relacionadas a consultas para todas as consultas de grupo de trabalho no CloudWatch.	Para cada consulta executada em um grupo de trabalho, se você configurar o grupo de trabalho para publicar métricas, o Athena as publicará no CloudWatch. Você pode visualizar as métricas de consulta para cada um dos grupos de trabalho no console do Athena. No CloudWatch, você pode criar painéis personalizados e definir limites e alarmes para essas métricas.

Como funcionam os grupos de trabalho

Os grupos de trabalho no Athena têm as seguintes características:

- Por padrão, cada conta tem um grupo de trabalho principal e as permissões padrão permitem que todos os usuários autenticados tenham acesso a esse grupo de trabalho. O grupo de trabalho principal não pode ser excluído.
- Cada grupo de trabalho que você criar mostra as consultas salvas e o histórico de consultas somente para consultas executadas, não para todas as consultas na conta. Isso separa suas consultas de outras consultas dentro de uma conta e as deixa mais eficientes, para você localizar suas próprias consultas salvas e consultas no histórico.
- Desativar um grupo de trabalho impede que as consultas sejam executadas nele até você ativá-lo. As consultas enviadas para um grupo de trabalho falham até você habilitá-lo novamente.
- Se tiver permissões, você poderá excluir um grupo de trabalho vazio e um grupo de trabalho que contém consultas salvas. Nesse caso, antes de excluir um grupo de trabalho, o Athena avisa que as consultas salvas serão excluídas. Antes de excluir um grupo de trabalho ao qual outros usuários têm acesso, verifique se eles têm acesso a outros grupos de trabalho nos quais podem continuar a executar consultas.

- Você pode fazer as configurações em todo o grupo de trabalho e impor seu uso por todas as consultas executadas em um grupo de trabalho. As configurações incluem a localização de resultados de consultas no Amazon S3, o proprietário do bucket esperado, a criptografia e o controle de objetos gravados no bucket de resultados de consultas.

> **Important**
>
> Quando você impuser configurações em todo o grupo de trabalho, todas as consultas executadas nesse grupo de trabalho usarão configurações de grupo de trabalho. Isso acontece mesmo que as configurações do lado do cliente forem diferentes do grupo de configurações. Para ter mais informações, consulte Configurações do grupo de trabalho sobrepõem as configurações do lado do cliente.

Limitações de grupos de trabalho

- Você pode criar até 1000 grupos de trabalho por região na sua conta.
- O grupo de trabalho principal não pode ser excluído.
- Você pode abrir até dez guias de consulta dentro de cada grupo de trabalho. Quando alternar entre grupos de trabalho, suas guias de consulta permanecerão abertas para até três grupos de trabalho.

Configurar grupos de trabalho

Configuração de grupos de trabalho envolve a criação deles e a definição de permissões para seu uso. Primeiro, decida de quais grupos de trabalho sua organização precisa e crie-os. Na sequência, configure as políticas de grupo de trabalho do IAM que controlam o acesso e as ações do usuário em um recurso do `workgroup`. Os usuários com acesso a esses grupos de trabalho agora podem executar consultas neles.

> **Note**
>
> Use essas tarefas para configurar grupos de trabalho quando você começar a usá-los pela primeira vez. Se sua conta no Athena já usar grupos de trabalho, cada usuário da conta precisa de permissões para executar consultas em um ou mais grupos de trabalho na conta. Antes de executar consultas, verifique sua política do IAM para ver quais grupos de trabalho

você pode acessar, ajuste a política, se necessário, e alterne para um grupo de trabalho que você pretende usar.

Por padrão, se você ainda não criou nenhum grupo de trabalho, todas as consultas na sua conta são executadas no grupo de trabalho principal.

O Athena exibe o grupo de trabalho atual na opção Workgroup (Grupo de trabalho) no canto superior direito do console. Você pode usar essa opção para alternar grupos de trabalho. Quando você executar consultas, elas serão executadas no grupo de trabalho atual. Você pode executar consultas no contexto de um grupo de trabalho no console por meio de operações da API, da interface de linha de comando ou de uma aplicação cliente usando o driver JDBC ou ODBC. Quando você tiver acesso a um grupo de trabalho, poderá visualizar as configurações de grupo de trabalho, as métricas e os limites de controle de uso de dados. Com permissões adicionais, é possível editar as configurações e os limites de controle de uso de dados.

Como configurar grupos de trabalho

1. Decida quais grupos de trabalho deseja criar. Por exemplo, decida o seguinte:

   - Quem pode executar consultas em cada grupo de trabalho e de quem é a configuração do grupo de trabalho. Isso determina as políticas do IAM que você cria. Para ter mais informações, consulte Políticas do IAM para acessar grupos de trabalho.
   - Quais locais no Amazon S3 usar para os resultados das consultas executadas em cada grupo de trabalho. Deve existir um local no Amazon S3 antes de você especificá-lo para os resultados das consultas de grupo de trabalho. Todos os usuários que usam um grupo de trabalho devem ter acesso a esse local. Para ter mais informações, consulte Configurações do grupo de trabalho.
   - Se o proprietário do bucket de resultados de consultas do Amazon S3 tem ou não controle total sobre novos objetos que são gravados nesse bucket. Por exemplo, se a localização dos resultados das consultas pertencer a outra conta, será possível conceder propriedade e controle total sobre os resultados das consultas à outra conta. Para mais informações, consulte AclConfiguration.
   - Especifique o ID da Conta da AWS que você espera ser a proprietária do bucket do local de saída. Esta é uma medida de segurança adicional opcional. Se o ID da conta do proprietário do bucket não corresponder ao ID especificado aqui, as tentativas de saída para o bucket falharão. Para obter mais informações, consulte Verificar propriedade do bucket com a

condição de proprietário do bucket no Guia do usuário do Amazon S3. Essa configuração não se aplica às instruções CTAS, INSERT INTO ou UNLOAD.

- Quais configurações de criptografia são necessárias e quais grupos de trabalho têm consultas que devem ser criptografadas. Recomendamos que você crie grupos de trabalho separados para consultas criptografadas e não criptografadas. Dessa forma, você pode impor a criptografia a um grupo de trabalho que se aplica a todas as consultas executadas nele. Para ter mais informações, consulte Criptografar resultados das consultas do Athena armazenados no Amazon S3.

2. Crie grupos de trabalho conforme o necessário e adicione tags a eles. Para obter as etapas, consulte Criar um grupo de trabalho.
3. Crie políticas do IAM para seus usuários, grupos ou funções para habilitar o acesso deles aos grupos de trabalho. As políticas estabelecem a associação do grupo de trabalho e o acesso a ações em um recurso `workgroup`. Para obter detalhes das etapas, consulte, Políticas do IAM para acessar grupos de trabalho. Para obter exemplos de políticas em JSON, consulte Acesso a grupos de trabalho e etiquetas.
4. Definir as configurações do grupo de trabalho Especifique uma localização no Amazon S3 para os resultados de consultas e, opcionalmente, especifique o proprietário do bucket esperado, as configurações de criptografia e o controle de objetos gravados no bucket de resultados de consultas. Você pode impor as configurações do grupo de trabalho. Para obter mais informações, consulte configurações do grupo de trabalho.

   > **Important**
   >
   > Se você substituir configurações do lado do cliente, o Athena usará as configurações do grupo de trabalho. Isso afeta as consultas que você executar no console, usando os drivers, a interface de linha de comando ou as operações da API.
   > As consultas continuam sendo executadas, mas a automação integrada com base na disponibilidade dos resultados em um determinado bucket do Amazon S3 poderá falhar. Recomendamos que você informe seus usuários antes de fazer a substituição. Após definir as configurações do grupo de trabalho para que façam a substituição, você poderá omitir a especificação das configurações no lado do cliente nos drivers ou na API.

5. Notifique os usuários sobre quais grupos de trabalho devem ser usados para executar consultas. Envie um e-mail para informar os usuários da sua conta sobre os nomes de grupo de

trabalho que eles podem usar, as políticas do IAM e as configurações necessárias do grupo de trabalho.

6. Configure os limites de controle de custos, também conhecidos como limites de controle de uso de dados, para consultas e grupos de trabalho. Para notificar você quando o limite for violado, crie um tópico do Amazon SNS e configure assinaturas. Para obter as etapas em detalhes, consulte Definir limites de controle de uso de dados e Conceitos básicos do Amazon SNS no Guia do desenvolvedor do Amazon Simple Notification Service.
7. Alterne para o grupo de trabalho para que você possa executar consultas. Para executar consultas, alterne para o grupo de trabalho apropriado. Para obter detalhes das etapas, consulte, the section called “Especificar um grupo de trabalho no qual executar consultas”.

Políticas do IAM para acessar grupos de trabalho

Para controlar o acesso a grupos de trabalho, use permissões do IAM no nível do recurso ou políticas do IAM baseadas em identidade. Sempre que você usar as políticas do IAM, siga as práticas recomendadas do IAM. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

> **Note**
>
> Para acessar grupos de trabalho habilitados para a propagação de identidade confiável, os usuários do Centro de Identidade do IAM devem ser atribuídos ao `IdentityCenterApplicationArn` que é retornado pela resposta da ação de API GetWorkGroup do Athena.

O procedimento a seguir é específico ao Athena.

Para obter informações específicas do IAM, acesse os links listados no fim desta seção. Para obter informações sobre exemplos de políticas de grupo de trabalho em JSON, consulte Exemplo de políticas de grupo de trabalho.

Para usar o editor visual no console do IAM para criar uma política de grupo de trabalho

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Policies (Políticas) e Create policy (Criar política).

3. Na guia Editor visual, selecione Escolher um serviço. Em seguida, escolha o Athena para adicionar à política.
4. Escolha Select actions (Selecionar ações) e defina as ações para adicionar à política. O editor visual mostra as ações disponíveis no Athena. Para obter mais informações, consulte Ações, recursos e chaves de condição do Amazon Athena na Referência de autorização do serviço.
5. Escolha Add actions (Adicionar ações) para digitar uma ação específica ou use caracteres curinga (*) para especificar várias ações.

   Por padrão, a política que você está criando permite as ações que você escolhe. Se você escolher uma ou mais ações compatíveis com as permissões no nível do recurso `workgroup` no Athena, o editor listará o recurso `workgroup`.
6. Escolha Resources (Recursos) para especificar os grupos de trabalho específicos para sua política. Para obter exemplos de políticas de grupo de trabalho em JSON, consulte Exemplo de políticas de grupo de trabalho.
7. Especifique o recurso `workgroup` da seguinte forma:

   ```
   arn:aws:athena:<region>:<user-account>:workgroup/<workgroup-name>
   ```
8. Selecione Review Policy (Revisar política), digite um Name (Nome) e uma Description (Descrição) (opcional) para a política que você está criando. Revise o resumo da política para ter certeza de que você concedeu as permissões que pretendia.
9. Escolha Criar política para salvar sua nova política.
10. Anexe essa política baseada em política a um usuário, um grupo ou uma função.

Para obter mais informações, consulte os seguintes tópicos na Referência de autorização do serviço e no Manual do usuário do IAM:

- Ações, recursos e chaves de condição do Amazon Athena
- Criar políticas com o editor visual
- Adicionar e remover políticas do IAM
- Controlar o acesso aos recursos

Para obter exemplos de políticas de grupo de trabalho em JSON, consulte Exemplo de políticas de grupo de trabalho.

Para obter uma lista completa de ações do Amazon Athena, consulte os nomes das ações de API na Referência de API do Amazon Athena.

Exemplo de políticas de grupo de trabalho

Esta seção inclui exemplos de políticas que você pode usar para habilitar várias ações nos grupos de trabalho. Sempre que você usar as políticas do IAM, siga as práticas recomendadas do IAM. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

Um grupo de trabalho é um recurso do IAM gerenciado pelo Athena. Portanto, se sua política de grupo de trabalho usar ações que tomam `workgroup` como entrada, especifique o ARN do grupo de trabalho da seguinte forma:

```
"Resource": [arn:aws:athena:<region>:<user-account>:workgroup/<workgroup-name>]
```

Onde *<workgroup-name>* é o nome do seu grupo de trabalho. Por exemplo, para o grupo de trabalho chamado `test_workgroup`, especifique-o como um recurso da seguinte forma:

```
"Resource": ["arn:aws:athena:us-east-1:123456789012:workgroup/test_workgroup"]
```

Para obter uma lista completa de ações do Amazon Athena, consulte os nomes das ações de API na Referência de API do Amazon Athena. Para obter mais informações sobre as políticas do IAM, consulte Criar políticas com o editor visual no Guia do usuário do IAM. Para obter mais informações sobre como criar políticas do IAM para grupos de trabalho, consulte Políticas do IAM para acessar grupos de trabalho.

- 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 Exemplo de política para acesso total a todos os grupos de trabalho

A seguinte política permite acesso total a todos os recursos do grupo de trabalho que podem existir na conta. Recomendamos que você use essa política para os usuários da sua conta que devem administrar e gerenciar grupos de trabalho para todos os outros usuários.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Example Exemplo de política para acesso total a um grupo de trabalho especificado

A seguinte política permite acesso total ao único recurso do grupo de trabalho específico, denominado `workgroupA`. Você pode usar essa política para os usuários com controle total sobre um determinado grupo de trabalho.

```
{
    "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 Exemplo de política para execução de consultas em um grupo de trabalho especificado

Na política a seguir, um usuário tem permissão para executar consultas no `workgroupA` especificado e visualizá-las. O usuário não tem permissão de realizar tarefas de gerenciamento para o grupo de trabalho em si, como atualizá-las ou excluí-las.

```
{
    "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 Exemplo de política para execução de consultas em um grupo de trabalho principal

É possível modificar o exemplo anterior para permitir que um determinado usuário também execute consultas no grupo de trabalho principal.

> **Note**
>
> Recomendamos que você adicione o recurso do grupo de trabalho principal para todos os usuários configurados para executar consultas em seus grupos de trabalho designados. Adicionar esse recurso às políticas de usuário do grupo de trabalho será útil caso o grupo de trabalho designado seja excluído ou esteja desabilitado. Neste caso, eles podem continuar executando consultas no grupo de trabalho principal.

Para permitir que os usuários em sua conta executem consultas no grupo de trabalho principal, adicione uma linha que contenha o ARN do grupo de trabalho principal à seção de recursos do Example policy for running queries in a specified workgroup, como no exemplo a seguir.

```
arn:aws:athena:us-east-1:123456789012:workgroup/primary"
```

Example Exemplo de política para operações de gerenciamento em um grupo de trabalho especificado

Na política a seguir, o usuário tem permissão de criar, excluir, obter detalhes e atualizar um grupo de trabalho `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 Exemplo de política para listagem de grupos de trabalho

A política a seguir permite que todos os usuários listem todos os grupos de trabalho:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListWorkGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

Example Exemplo de política para execução e interrupção de consultas em um grupo de trabalho específico

Nesta política, o usuário tem a permissão de executar consultas no grupo de trabalho:

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

```
        {
            "Effect": "Allow",
            "Action": [
                "athena:StartQueryExecution",
                "athena:StopQueryExecution"
            ],
            "Resource": [
                "arn:aws:athena:us-east-1:123456789012:workgroup/test_workgroup"
            ]
        }
    ]
}
```

Example Exemplo de política para trabalhar com consultas nomeadas em um grupo de trabalho específico

Na política a seguir, o usuário tem permissões para criar, excluir e obter informações sobre consultas nomeadas no grupo de trabalho especificado:

```
{
    "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 Exemplo de política para trabalhar com cadernos Spark no Athena

Use uma política como a apresentada a seguir para trabalhar com cadernos Spark no 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": "*"
        }
    ]
}
```

Configurações do grupo de trabalho

Cada grupo de trabalho tem as seguintes configurações:

- Um nome exclusivo. Ele pode conter de 1 a 128 caracteres, incluindo caracteres alfanuméricos, traços e sublinhados. Depois de criar um grupo de trabalho, não é possível alterar o nome dele. No entanto, você pode criar um novo grupo de trabalho com as mesmas configurações e um nome diferente.
- Configurações que se aplicam a todas as consultas em execução no grupo de trabalho. Incluindo:
  - Um local no Amazon S3 para armazenar os resultados de todas as consultas executadas neste grupo de trabalho. Esse local deve existir antes de você especificá-lo para o grupo de trabalho ao criá-lo. Para obter informações sobre como criar um bucket do Amazon S3, consulte Criação de um bucket.
  - Bucket owner control of query results (Controle de resultados de consultas pelo proprietário do bucket) – Se o proprietário do bucket de resultados de consultas do Amazon S3 tem ou não controle total sobre novos objetos que são gravados nesse bucket. Por exemplo, se a localização dos resultados das consultas pertencer a outra conta, será possível conceder propriedade e controle total sobre os resultados das consultas à outra conta.
  - Proprietário do bucket esperado: o ID da Conta da AWS que você espera ser a proprietária do bucket de resultados da consulta. Essa é uma medida de segurança adicional. Se o ID da conta do proprietário do bucket não corresponder ao ID especificado aqui, as tentativas de saída para o bucket falharão. Para obter informações detalhadas, consulte Verificar propriedade do bucket com a condição de proprietário do bucket no Guia do usuário do Amazon S3.

    > **Note**
    >
    > A configuração esperada do proprietário do bucket se aplica somente ao local de saída do Amazon S3 que você especificar para os resultados da consulta do Athena. Ela não se aplica a outros locais do Amazon S3, como locais de origem dos dados em buckets externos do Amazon S3, locais da tabela de destino `CTAS` e `INSERT INTO`, locais de

saída da instrução `UNLOAD`, operações para buckets de vazamento para consultas federadas ou consultas `SELECT` executadas em uma tabela em outra conta.

- Uma configuração de criptografia, se você usar a criptografia para todas as consultas do grupo de trabalho. Você só pode criptografar todas as consultas em um grupo de trabalho, não apenas algumas delas. É recomendável criar grupos separados para conter consultas que são criptografadas ou não criptografadas.

Além disso, seu grupo de trabalho pode substituir as configurações no lado do cliente. Antes do lançamento dos grupos de trabalho, você especificava o local dos resultados e as opções de criptografia como parâmetros no driver JDBC ou ODBC ou na guia Properties (Propriedades) no console do Athena. Essas configurações também podem ser especificadas diretamente por meio das operações da API. Essas configurações são conhecidas como "configurações do lado do cliente". Com grupos de trabalho, é possível definir essas configurações no nível do grupo de trabalho para controlar as opções disponíveis no nível do cliente. A aplicação das configurações no nível do grupo de trabalho também evita que os usuários precisem definir as configurações do lado do cliente individualmente. Se você selecionar a opção Substituir configurações no lado do cliente para o grupo de trabalho, as consultas usarão as configurações do grupo de trabalho e ignorarão as configurações do lado do cliente.

Se Override Client-Side Settings (Substituir configurações no lado do cliente) for selecionado, o usuário será notificado no console que suas configurações foram alteradas. Se as configurações do grupo de trabalho forem impostas dessa forma, os usuários poderão omitir as configurações correspondentes do lado do cliente. Em seguida, as consultas executadas no console usam as configurações do grupo de trabalho, ainda que as configurações do lado do cliente estejam presentes. Além disso, quando as consultas do grupo de trabalho são executadas pela AWS CLI, por operações de API ou drivers JDBC ou ODBC, as configurações do lado do cliente, como localização dos resultados da consulta e criptografia, são substituídas pelas configurações do grupo de trabalho. Para ver as configurações do grupo de trabalho, visualize os detalhes do grupo de trabalho.

Você também pode definir limites de consulta para consultas nos grupos de trabalho.

Configurações do grupo de trabalho sobrepõem as configurações do lado do cliente

As caixas de diálogo Create workgroup (Criar grupo de trabalho) e Edit workgroup (Editar grupo de trabalho) têm um campo de nome Override client-side settings (Substituir configurações no lado do cliente). Esse campo não é selecionado por padrão. Dependendo da sua seleção, o Athena faz o seguinte:

- Se a opção Substituir configurações no lado do cliente não estiver selecionada, as configurações do grupo de trabalho não serão impostas no nível do cliente. Quando a opção de substituir as configurações do lado cliente não está selecionada para o grupo de trabalho, o Athena usa as configurações do cliente para todas as consultas executadas no grupo de trabalho, inclusive as configurações para localização dos resultados de consultas, proprietário do bucket esperado, criptografia e controle de objetos gravados no bucket de resultados de consultas. Cada usuário pode especificar as próprias configurações no menu Configurações no console. Se as configurações no lado do cliente não forem definidas, serão aplicadas as configurações de todo o grupo de trabalho. Se você usar a AWS CLI, ações de API ou os drivers JDBC e ODBC para executar consultas em um grupo de trabalho que não substitui as configurações do lado do cliente, suas consultas usarão as configurações especificadas em suas consultas.
- Se a opção Substituir configurações no lado do cliente for selecionada, as configurações do grupo de trabalho serão impostas no nível do cliente a todos os clientes do grupo de trabalho. Quando a opção de substituir as configurações do lado cliente está selecionada para o grupo de trabalho, o Athena usa as configurações do grupo de trabalho para todas as consultas executadas no grupo de trabalho, inclusive as configurações para localização dos resultados de consultas, proprietário do bucket esperado, criptografia e controle de objetos gravados no bucket de resultados de consultas. As configurações do grupo de trabalho substituem todas as configurações do lado do cliente especificadas para uma consulta ao usar o console, as ações da API ou os drivers JDBC ou ODBC.

  Se você substituir as configurações no lado do cliente, da próxima vez que você ou qualquer usuário do grupo de trabalho abrir o console do Athena, o Athena notificará você de que as consultas do grupo de trabalho usam as configurações do grupo de trabalho e solicitará a confirmação dessa alteração.

  > **Important**
  >
  > Se você usar ações de API, a AWS CLI ou os drivers JDBC e ODBC para executar consultas em um grupo de trabalho que substitui as configurações do lado do cliente, omita as configurações do lado do cliente em suas consultas ou atualize-as para corresponder às configurações do grupo de trabalho. Se você especificar configurações do lado do cliente nas consultas, mas executá-las em um grupo de trabalho que substitui as configurações, as consultas serão executadas, mas serão usadas as configurações do grupo de trabalho. Para obter informações sobre como visualizar as configurações de um grupo de trabalho, consulte Visualizar os detalhes do grupo de trabalho.

Gerenciar grupos de trabalho

Em https://console.aws.amazon.com/athena/, você pode executar as seguintes tarefas:

Statement	Descrição
Criar um grupo de trabalho	Criar um novo grupo de trabalho.
Editar um grupo de trabalho	Editar um grupo de trabalho e alterar as configurações. Você não pode alterar o nome de um grupo de trabalho, mas pode criar um novo grupo de trabalho com as mesmas configurações e um nome diferente .
Visualizar os detalhes do grupo de trabalho	Visualize os detalhes do grupo de trabalho, como nome, descrição , limites de uso de dados, localização de resultados de consultas, proprietário esperado do bucket de resultados de consultas, criptografia e controle de objetos gravados no bucket de resultados de consultas. Você também pode verificar se esse grupo de trabalho impõe suas configurações, caso a opção Override client-side settings (Substituir configurações no lado do cliente) esteja marcada.
Excluir um grupo de trabalho	Exclua um grupo de trabalho. Se você excluir um grupo de trabalho, serão excluídos o histórico de consultas, as consultas salvas, as configurações do grupo de trabalho e os controles do limite de dados por consulta. Os controles do limite de dados no âmbito do grupo de trabalho permanecem no CloudWatch, e você pode excluí-los um a um. O grupo de trabalho principal não pode ser excluído.
Alternar grupos de trabalho	Alterne entre grupos de trabalho aos quais você tem acesso.
Copiar uma consulta salva entre grupos de trabalho	Copie uma consulta salva entre grupos de trabalho. Convém fazer isso se, por exemplo, você criou uma consulta em um grupo de trabalho de previsualização e quiser disponibilizá-la em um grupo de trabalho que não seja de previsualização.

Statement	Descrição
Habilitar e desabilitar um grupo de trabalho	Habilite ou desabilite um grupo de trabalho. Quando um grupo de trabalho estiver desabilitado, os usuários não poderão executar consultas nem criar novas consultas com nome. Se você tiver acesso a ele, ainda pode visualizar métricas, os controles de limite de uso de dados, as configurações de grupo de trabalho, o histórico de consultas e as consultas salvas.
Especificar um grupo de trabalho no qual executar consultas	Antes de executar as consultas, especifique ao Athena qual grupo de trabalho deve ser usado. Você deve ter permissões para o grupo de trabalho.
Crie um grupo de trabalho do Athena que use a autenticação do Centro de Identidade do IAM	Para usar as identidades do Centro de Identidade do IAM com o Athena, é necessário criar um grupo de trabalho habilitado para o Centro de Identidade do IAM. Após criar o grupo de trabalho, você poderá usar o console ou a API do Centro de Identidade do IAM para atribuir usuários ou grupos do Centro de Identidade do IAM ao grupo de trabalho.

Criar um grupo de trabalho

Criar um grupo de trabalho requer permissões para ações da API `CreateWorkgroup`. Consulte Acesso a grupos de trabalho e etiquetas e Políticas do IAM para acessar grupos de trabalho. Se você adicionar tags, também precisará adicionar permissões para `TagResource`. Consulte Exemplos de política de etiquetas para grupos de trabalho.

Para criar um grupo de trabalho no console

1. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.

2. No painel de navegação do console do Athena, escolha Workgroups (Grupos de trabalho).
3. Na página Workgroups (Grupos de trabalho), escolha Create workgroup (Criar grupo de trabalho).
4. Na página Create workgroup (Criar grupo de trabalho), preencha os campos da seguinte forma:

Campo	Descrição
Workgroup name (Nome do grupo de trabalho)	Obrigatório. Digite um nome exclusivo para seu grupo de trabalho. Use 1 a 128 caracteres. (A-Z,a-z,0-9,_,-,.). Esse nome não pode ser alterado.
Descrição	Opcional. Insira uma descrição para seu grupo de trabalho. Ela pode conter até 1024 caracteres.
Choose the type of engine (Escolha o tipo de mecanismo)	Escolha o Athena SQL (SQL do Athena) se você deseja executar consultas SQL ad-hoc em dados no Amazon S3 ou usar um conector de fonte de dados criado previamente para executar consultas federadas em uma variedade de fontes de dados externas ao Amazon S3. É possível executar consultas usando o editor de consultas do Athena ou a AWS CLI ou as APIs do Athena. Escolha o Apache Spark se você deseja criar, editar e executar aplicações de caderno Jupyter usando o Python e o Apache Spark. Os cadernos Jupyter contêm uma lista de células que podem incluir

Campo	Descrição
	código, texto, Markdown, matemática, plotagens e mídia avançada. As células são executadas em ordem como cálculos em uma sessão interativa do caderno no Athena. Para obter informações sobre como criar e configurar um grupo de trabalho habilitado para Spark, consulte Criação de um grupo de trabalho habilitado para Spark no Athena. Após a criação de um grupo de trabalho, o mecanismo de análise poderá ser atualizado (por exemplo, da versão 2 do mecanismo do Athena para a versão 3 do mecanismo do Athena), mas o tipo de mecanismo não poderá ser alterado. Por exemplo, um grupo de trabalho que usa a versão 3 do mecanismo Athena não poderá ser alterado para um grupo de trabalho que usa a versão 3 do mecanismo PySpark.
Atualizar o mecanismo de consulta	Escolha como você deseja atualizar seu grupo de trabalho quando uma nova versão do mecanismo de pesquisa do Athena for lançada. Você pode deixar que o Athena decida quando fazer upgrade do seu grupo de trabalho ou escolher manualmente uma versão do mecanismo. Para ter mais informações, consulte Versionamento do mecanismo do Athena.
Modo de autenticação	Escolha AWS Identity and Access Management (IAM) para usar a autenticação ou a federação do IAM para o grupo de trabalho. Escolha Centro de Identidade do IAM se quiser oferecer suporte a identidades de força de trabalho, por exemplo, usuários e grupos de provedores de identidades do SAML 2.0, como o Microsoft Active Directory. Para obter mais informações, consulte Trusted identity propagation across applications no Guia do usuário do AWS IAM Identity Center.
Perfil de serviço para acesso ao Centro de Identidade do IAM	O Athena requer permissões do IAM para acessar o Centro de Identidade do IAM em seu nome. Para obter mais informações sobre os perfis de serviço do IAM, consulte Criar um perfil para delegar permissões a um serviço da AWS no Guia do usuário do IAM.

Campo	Descrição
Location of query result (Local do resultado da consulta)	Opcional. Insira um caminho para um prefixo ou bucket do Amazon S3. O bucket e o prefixo devem existir antes que você especificá-los. **Note** Se você executar consultas no console, especificar o local dos resultados da consulta é opcional. Se você não especificá-lo para o grupo de trabalho nem em Settings (Configurações), a Athena usará o local do resultado da consulta padrão. Se você executar consultas com a API ou os drivers, deverá especificar o local dos resultado s da consulta em pelo menos um de dois lugares: para consultas individuais, com OutputLocation, ou para o grupo de trabalho, com WorkGroupConfiguration.

Campo	Descrição
Expected bucket owner (Proprietário esperado do bucket)	Opcional. Insira o ID da Conta da AWS que você espera ser a proprietária do bucket do local de saída. Essa é uma medida de segurança adicional. Se o ID da conta do proprietário do bucket não corresponder ao ID especificado aqui, as tentativas de saída para o bucket falharão. Para obter informações detalhadas, consulte Verificar propriedade do bucket com a condição de proprietário do bucket no Guia do usuário do Amazon S3. **Note** A configuração esperada do proprietário do bucket se aplica somente ao local de saída do Amazon S3 que você especificar para os resultados da consulta do Athena. Ela não se aplica a outros locais do Amazon S3, como locais de origem dos dados em buckets externos do Amazon S3, locais da tabela de destino `CTAS` e `INSERT INTO`, locais de saída da instrução `UNLOAD`, operações para buckets de vazamento para consultas federadas ou consultas `SELECT` executadas em uma tabela em outra conta.

Campo	Descrição
Assign bucket owner full control over query results (Atribuir controle total ao proprietário do bucket sobre resultados de consultas)	Esse campo não é selecionado por padrão. Se essa opção for selecionada e ACLs estiverem habilitadas para o bucket de localização de resultados de consultas, você concede acesso de controle total sobre os resultados das consultas ao proprietário do bucket. Por exemplo, se a localização dos resultados das consultas pertencer a outra conta, será possível conceder propriedade e controle total sobre os resultados das consultas à outra conta. Se a configuração para S3 Object Ownership (Propriedade de objetos do S3) for Bucket owner preferred (Proprietário do bucket preferencial), o proprietário do bucket também possuirá todos os objetos de resultados de consultas gravados a partir deste grupo de trabalho. Por exemplo, quando o grupo de trabalho de uma conta externa habilita essa opção e define a localização dos resultados das consultas como o bucket do Amazon S3 da sua conta, cuja configuração S3 Object Ownership (Propriedade de objetos do S3) é Bucket owner preferred (Proprietário do bucket preferencial), você é o proprietário e tem acesso de controle total sobre os resultados da consulta do grupo de trabalho externo. A seleção dessa opção com a configuração configuração S3 Object Ownership (Propriedade de objetos do S3) definida como Bucket owner enforced (Proprietário do bucket imposto) não surte efeito. Para obter mais informações, consulte Configurações do Object Ownership no Guia do Usuário do Amazon S3.

Campo	Descrição
Encrypt query results (Criptografar resultados da consulta)	Opcional. Criptografe os resultados armazenados no Amazon S3. Se selecionada essa opção, todas as consultas do grupo de trabalho serão criptografadas. Se selecionada essa opção, você pode selecionar o Encryption type (Tipo de criptografia), a Encryption key (Chave de criptografia) e insira o KMS Key ARN (ARN da chave do KMS). Se você não tiver a chave, abra o console do AWS KMS para criá-la. Para obter mais informações, consulte Criação de chaves Guia do desenvolvedor do AWS Key Management Service.
Definir ***`encryption_type`*** como criptografia mínima	Opcional. Selecione essa opção para impor um tipo mínimo de criptografia aos resultados da consulta para todos os usuários do grupo de trabalho. Selecionar essa opção exibe uma tabela com a hierarquia dos tipos de criptografia. A tabela também mostra quais tipos de criptografia os usuários do grupo de trabalho poderão usar quando você especificar determinado tipo de criptografia como mínimo. Para usar essa opção, a opção Substituir configurações do lado do cliente não deve estar selecionada. Para ter mais informações, consulte Configurar a criptografia mínima para um grupo de trabalho.
Habilitar concessões de acesso do S3	Este campo é selecionado por padrão quando você escolhe o Centro de Identidade do IAM como o modo de autenticação. Quando selecionada, essa opção aplica permissões baseadas em usuários ou grupos do Centro de Identidade do IAM a locais do Amazon S3.
Criar prefixo do S3 baseado na identidade do usuário	Quando essa opção é selecionada, o Athena cria um prefixo do Amazon S3 ao armazenar os resultados da consulta. O prefixo é baseado na identidade do usuário do Centro de Identidade do IAM.

Campo	Descrição
Publicar métricas de consulta no CloudWatch	Esse campo é selecionado por padrão. Publicar métricas de consulta no CloudWatch. Consulte Monitorar consultas do Athena com métricas do CloudWatch.
Override client-side settings (Substituir configurações no lado do cliente)	Esse campo não é selecionado por padrão. Se você selecionar essa opção, as configurações do grupo de trabalho se aplicarão a todas as consultas do grupo de trabalho e ficarão sobrepostas às configurações no lado do cliente. Para ter mais informações, consulte Configurações do grupo de trabalho sobrepõem as configurações do lado do cliente.
Buckets do S3 de pagamento pelo solicitante	Opcional. Escolha Turn on queries on requester pays buckets in Amazon S3 (Ativar consultas em buckets de pagamento a cargo do solicitante no Amazon S3) se os usuários do grupo de trabalho executarem consultas em dados armazenados em buckets do Amazon S3 configurados como “pagamento a cargo do solicitante”. A conta do usuário que executa a consulta é cobrada pelas taxas de acesso e transferência de dados aplicáveis associadas à consulta. Para obter mais informações, consulte Buckets de pagamento a cargo do solicitante no Guia do usuário do Amazon Simple Storage Service.
Gerenciar o controle de uso de dados por consulta	Opcional. Define o limite para a quantidade máxima de dados que uma consulta tem permissão para verificar. Você pode definir apenas um limite por consulta para um grupo de trabalho. O limite se aplica a todas as consultas do grupo de trabalho e, se a consulta exceder o limite, ela será cancelada. Para ter mais informações, consulte Definir limites de controle de uso de dados.

Campo	Descrição
Workgroup data usage alerts (Alertas de uso de dados do grupo de trabalho)	Opcional. Defina vários limites de alerta quando as consultas em execução nesse grupo de trabalho verificam uma quantidade especificada de dados dentro de um período específico. Os alertas são implementados usando alarmes do Amazon CloudWatch e se aplicam a todas as consultas no grupo de trabalho. Para obter mais informações, consulte Uso de alarmes do Amazon CloudWatch no Manual do usuário do Amazon CloudWatch.
Tags	Opcional. Adicione uma ou mais tags a um grupo de trabalho. A tag é um rótulo atribuído a um recurso do grupo de trabalho do Athena. Ela é formada por uma chave e um valor. Use as práticas recomendadas de marcação com tags da AWS para criar um conjunto consistente de tags e categorizar os grupo de trabalho por finalidade, proprietário ou ambiente. Você também pode usar tags nas políticas do IAM e para controlar os custos de faturamen to. Não use chaves de tag duplicadas no mesmo grupo de trabalho. Para ter mais informações, consulte the section called “Marcar recursos”.

5. Escolha Create workgroup (Criar grupo de trabalho). O grupo de trabalho aparece na lista na página Workgroups (Grupos de trabalho).

Você também pode usar a operação da API CreateWorkGroup para criar um grupo de trabalho.

> **Important**
>
> Após criar grupos de trabalho, crie Políticas do IAM para acessar grupos de trabalho do IAM que permitam executar ações relacionadas ao grupo de trabalho.

Editar um grupo de trabalho

Como editar um grupo de trabalho requer permissões para operações da API `UpdateWorkgroup`. Consulte Acesso a grupos de trabalho e etiquetas e Políticas do IAM para acessar grupos de trabalho. Se você adicionar ou editar tags, também precisará ter permissões para `TagResource`. Consulte Exemplos de política de etiquetas para grupos de trabalho.

Para editar um grupo de trabalho no console

1. No painel de navegação do console do Athena, escolha Workgroups (Grupos de trabalho).
2. Na página Workgroups (Grupos de trabalho), selecione o botão do grupo de trabalho que você deseja editar.
3. Selecione Ações, Editar.
4. Altere os campos conforme necessário. Para ver a lista de campos, consulte Criar grupo de trabalho. Você pode alterar todos os campos, exceto pelo nome do grupo de trabalho. Se você precisar alterar o nome, crie outro grupo de trabalho com o novo nome e as mesmas configurações.
5. Escolha Salvar alterações. O grupo de trabalho atualizado aparece na lista na página Workgroups (Grupos de trabalho).

Visualizar os detalhes do grupo de trabalho

Para cada grupo de trabalho, você pode visualizar os detalhes. Os detalhes incluem o nome do grupo de trabalho, a descrição, se ele está habilitado ou desabilitado, e as configurações usadas para consultas executadas no grupo de trabalho, que incluem a localização dos resultados das consultas, o proprietário esperado do bucket, a criptografia e o controle de objetos gravados no bucket de resultados de consultas. Se um grupo de trabalho tiver limites de uso de dados, eles também serão exibidos.

Para exibir os detalhes do grupo de trabalho

1. No painel de navegação do console do Athena, escolha Workgroups (Grupos de trabalho).
2. Na página Workgroups (Grupos de trabalho), escolha o link do grupo de trabalho que você deseja visualizar. A página Overview Details (Visão geral dos detalhes) do grupo de trabalho é exibida.

Excluir um grupo de trabalho

Você pode excluir um grupo de trabalho se tiver permissões para fazê-lo. O grupo de trabalho principal não pode ser excluído.

Se tiver permissões, você poderá excluir um grupo de trabalho vazio a qualquer momento. Você também pode excluir um grupo de trabalho que contém as consultas salvas. Nesse caso, antes

de prosseguir para excluir um grupo de trabalho, o Athena avisa que as consultas salvas serão excluídas.

Se você excluir um grupo de trabalho enquanto estiver nele, o console vai alternar o foco para o grupo de trabalho principal. Se você tiver acesso a ele, pode executar consultas e exibir as configurações.

Se você excluir um grupo de trabalho, as configurações e os controles de limite de dados por consulta serão excluídos. Os controles de limite de dados no âmbito do grupo de trabalho permanecem no CloudWatch, e você pode excluí-los, se necessário.

> **Important**
>
> Antes de excluir um grupo de trabalho, verifique se os usuários também pertencem a outros grupos de trabalho onde podem continuar a executar consultas. Se as políticas do IAM dos usuários permitirem que eles executem consultas somente nesse grupo de trabalho e você o excluir, eles não terão mais permissões para executar as consultas. Para ter mais informações, consulte Example policy for running queries in the primary workgroup.

Para excluir um grupo de trabalho no console

1. No painel de navegação do console do Athena, escolha Workgroups (Grupos de trabalho).
2. Na página Workgroups (Grupos de trabalho), selecione o botão do grupo de trabalho que você deseja excluir.
3. Escolha Ações, Excluir.
4. No prompt de confirmação, insira Delete workgroup (Excluir grupo de trabalho) insira o nome do grupo de trabalho e, em seguida, escolha Delete (Excluir).

Para excluir um grupo de trabalho com a operação da API, use a ação `DeleteWorkGroup`.

Alternar grupos de trabalho

Você pode alternar de um grupo de trabalho para outro caso você tenha permissões para ambos.

Você pode abrir até dez guias de consulta dentro de cada grupo de trabalho. Quando alternar entre grupos de trabalho, suas guias de consulta permanecerão abertas para até três grupos de trabalho.

Para alternar grupos de trabalho

1. No console do Athena, escolha a opção Workgroup (Grupo de trabalho) no canto superior direito para escolher um grupo de trabalho.
2. Se a caixa de diálogo Workgroup ***workgroup-name*** settings (Configurações de nome-do-grupo-de-trabalho do grupo de trabalho) for exibida, escolha Acknowledge (Confirmar).

A opção Workgroup (Grupo de trabalho) mostra o nome do grupo de trabalho para o qual você mudou. Você já pode executar consultas nesse grupo de trabalho.

Copiar uma consulta salva entre grupos de trabalho

Atualmente, o console do Athena não tem uma opção para copiar uma consulta salva de um grupo de trabalho para outro diretamente, mas você pode executar a mesma tarefa manualmente usando o procedimento a seguir.

Para copiar uma consulta salva entre grupos de trabalho

1. No console do Athena, no grupo de trabalho do qual você deseja copiar a consulta, escolha a guia Saved queries (Consultas salvas).
2. Escolha o link da consulta salva que você deseja copiar. O Athena abre a consulta no editor de consultas.
3. No editor de consultas, selecione o texto da consulta e pressione **`Ctrl+C`** para copiá-lo.
4. Alterne para o grupo de trabalho de destino ou crie um grupo de trabalho e alterne para ele.
5. Abra uma nova guia no editor de consultas e pressione **`Ctrl+V`** para colar o texto na nova guia.
6. No editor de consultas, escolha Save as (Salvar como) para salvar a consulta no grupo de trabalho de destino.
7. Na caixa de diálogo Choose a name (Escolher um nome), insira um nome para a consulta e uma descrição opcional.
8. Escolha Salvar.

Habilitar e desabilitar um grupo de trabalho

Se você tiver permissões para fazê-lo, pode habilitar ou desabilitar grupos de trabalho no console usando as operações de API ou com os drivers JDBC e ODBC.

Para habilitar ou desabilitar um grupo de trabalho

1. No painel de navegação do console do Athena, escolha Workgroups (Grupos de trabalho).
2. Na página Workgroups (Grupos de trabalho), escolha o link do grupo de trabalho.
3. No canto superior direito, escolha Enable workgroup (Habilitar grupo de trabalho) ou Disable workgroup (Desativar grupo de trabalho).
4. No prompt de confirmação, escolha Enable (Habilitar) ou Disable (Desabilitar). Se você desativar um grupo de trabalho, os usuários não poderão executar consultas nem criar novas consultas nomeadas. Se você habilitar um grupo de trabalho, os usuários poderão usá-lo para executar consultas.

Especificar um grupo de trabalho no qual executar consultas

Para especificar um grupo de trabalho a ser usado, você deve ter permissões para ele.

Para especificar o grupo de trabalho a usar

1. Suas permissões devem permitir que você execute consultas no grupo de trabalho que você pretende usar. Para ter mais informações, consulte the section called “ Políticas do IAM para acessar grupos de trabalho”.
2. Para especificar o grupo de trabalho, use uma destas opções:

   - Se você estiver acessando o Athena por meio do console, defina o grupo de trabalho alternando os grupos de trabalho.
   - Se você usa as operações de API do Athena, especifique o nome do grupo de trabalho na ação da API. Por exemplo, você pode definir o nome do grupo de trabalho em StartQueryExecution, da seguinte forma:

     ```
     StartQueryExecutionRequest startQueryExecutionRequest = new
      StartQueryExecutionRequest()
                  .withQueryString(ExampleConstants.ATHENA_SAMPLE_QUERY)
                  .withQueryExecutionContext(queryExecutionContext)
                  .withWorkGroup(WorkgroupName)
     ```

   - Se você estiver usando o driver JDBC ou ODBC, defina o nome do grupo de trabalho na string de conexão usando o parâmetro de configuração `Workgroup`. O driver passa o nome do grupo de trabalho para o Athena. Especifique o parâmetro do grupo de trabalho na string de conexão, como no exemplo a seguir:

```
jdbc:awsathena://AwsRegion=<AWSREGION>;UID=<ACCESSKEY>;
PWD=<SECRETKEY>;S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/<athena-
output>-<AWSREGION>/;
Workgroup=<WORKGROUPNAME>;
```

Configurar a criptografia mínima para um grupo de trabalho

Como administrador de um grupo de trabalho do Athena SQL, você pode impor um nível mínimo de criptografia no Amazon S3 para todos os resultados de consultas do grupo de trabalho. Use esse recurso para garantir que os resultados da consulta nunca sejam armazenados em um bucket do Amazon S3 em estado não criptografado.

Quando os usuários de um grupo de trabalho com criptografia mínima ativada enviam uma consulta, eles podem definir apenas a criptografia para o nível mínimo que você configurou ou para um nível superior, se houver um disponível. O Athena criptografa os resultados da consulta no nível especificado quando o usuário executa a consulta ou no nível definido no grupo de trabalho.

Os seguintes níveis estão disponíveis:

- Básica: criptografia do lado do servidor do Amazon S3 com chaves gerenciadas pelo Amazon S3 (SSE-S3).
- Intermediária: criptografia do lado do servidor com chaves gerenciadas do KMS (SSE-KMS).
- Avançada: criptografia do lado do cliente com chaves gerenciadas do KMS (CSE-KMS).

Considerações e limitações

- O recurso de criptografia mínima não está disponível para grupos de trabalho habilitados para o Apache Spark.
- O recurso de criptografia mínima só funciona quando o grupo de trabalho não habilita a opção Substituir configurações do lado do cliente.
- Se o grupo de trabalho estiver com a opção Substituir configurações do lado do cliente habilitada, a configuração de criptografia do grupo de trabalho prevalecerá, e a configuração mínima de criptografia não terá efeito.
- Não há custos para habilitar esse recurso.

Habilitar criptografia mínima para um grupo de trabalho

É possível habilitar um nível mínimo de criptografia para os resultados da consulta do grupo de trabalho de SQL do Athena ao criar ou atualizar o grupo de trabalho. Para isso, você pode usar o console do Athena, a API do Athena ou a AWS CLI.

Usar o console do Athena para habilitar a criptografia mínima

Para começar a criar ou editar o grupo de trabalho usando o console do Athena, consulte Criar um grupo de trabalho ou Editar um grupo de trabalho. Ao configurar o grupo de trabalho, use as etapas a seguir para ativar a criptografia mínima.

Para configurar o nível mínimo de criptografia para resultados de consultas de grupos de trabalho

1. Na seção Configurações adicionais, expanda Configurações.
2. Desmarque a opção Substituir configurações do lado do cliente ou verifique se ela não está selecionada.
3. Na seção Configurações adicionais, expanda Configuração do resultado da consulta.
4. Selecione a opção Criptografar resultados da consulta.
5. Em Tipo de criptografia, selecione o método de criptografia que deseja que o Athena use para os resultados da consulta do grupo de trabalho (SSE_S3, SSE_KMS ou CSE_KMS). Esses tipos de criptografia correspondem aos níveis de segurança básico, intermediário e avançado.
6. Para impor o método de criptografia escolhido como o nível mínimo de criptografia para todos os usuários, selecione Definir ***`encryption_method`*** como criptografia mínima.

   Quando essa opção é selecionada, uma tabela mostra a hierarquia de criptografia e os níveis de criptografia permitidos aos usuários quando o tipo de criptografia escolhido torna-se o mínimo.
7. Depois de criar o grupo de trabalho ou atualizar a configuração de grupo de trabalho, escolha Criar grupo de trabalho ou Salvar alterações.

Usando a API do Athena ou a AWS CLI para habilitar a criptografia mínima

Ao usar a API CreateWorkGroup ou UpdateWorkGroup para criar ou atualizar um grupo de trabalho de SQL do Athena, defina EnforceWorkGroupConfiguration como `false`, EnableMinimumEncryptionConfiguration como `true` e use EncryptionOption para especificar o tipo de criptografia.

Na AWS CLI, use o comando `create-work-group` ou `update-work-group` com os parâmetros `--configuration` ou `--configuration-updates` e especifique as opções correspondentes às da API.

Uso de grupos de trabalho do Athena habilitados para o Centro de Identidade do IAM

O recurso de propagação de identidade confiável do AWS IAM Identity Center permite que as identidades da força de trabalho sejam usadas nos serviços de análise da AWS. A propagação de identidade confiável evita que você precise executar configurações específicas de serviço para o provedor de identidades ou definições para o perfil do IAM.

Com o Centro de Identidade do IAM, é possível gerenciar a segurança de login para as identidades da força de trabalho, também conhecidas como usuários da força de trabalho. O Centro de Identidade do IAM oferece um local no qual é possível criar ou conectar usuários da força de trabalho e gerenciar de forma centralizada o acesso deles em todas as contas e aplicações da AWS. É possível usar permissões de várias contas para atribuir acesso a Contas da AWS a esses usuários. Você pode usar as atribuições de aplicações para atribuir aos seus usuários acesso a aplicações habilitadas para o Centro de Identidade do IAM, aplicações em nuvem e aplicações do Security Assertion Markup Language (SAML 2.0) do cliente. Para obter mais informações, consulte Trusted identity propagation across applications no Guia do usuário do AWS IAM Identity Center.

No momento, o suporte do Athena SQL para a propagação de identidade confiável permite usar a mesma identidade para o Amazon EMR Studio e para a interface do Athena SQL no EMR Studio. Para usar as identidades do Centro de Identidade do IAM com o Athena SQL no EMR Studio, é necessário criar grupos de trabalho habilitados para o Centro de Identidade do IAM no Athena. Em seguida, é possível usar o console ou a API do Centro de Identidade do IAM para atribuir usuários ou grupos do Centro de Identidade do IAM aos grupos de trabalho do Athena habilitados para o Centro de Identidade do IAM. As consultas de um grupo de trabalho do Athena que usa propagação de identidade confiável devem ser executadas na interface do Athena SQL em um EMR Studio que tenha o Centro de Identidade do IAM habilitado.

Considerações e limitações

Ao usar a propagação de identidade confiável com o Amazon Athena, considere os seguintes pontos:

- Não é possível alterar o método de autenticação para o grupo de trabalho após sua criação.
  - Os grupos de trabalho do Athena SQL existentes não podem ser modificados para oferecer suporte aos grupos de trabalho habilitados para o Centro de Identidade do IAM.

  - Os grupos de trabalho habilitados para o Centro de Identidade do IAM não podem ser modificados para oferecer suporte a permissões do IAM em nível de recurso ou a políticas do IAM baseadas em identidade.
- Para acessar grupos de trabalho habilitados para a propagação de identidade confiável, os usuários do Centro de Identidade do IAM devem ser atribuídos ao `IdentityCenterApplicationArn` que é retornado pela resposta da ação de API GetWorkGroup do Athena.
- As concessões de acesso do Amazon S3 devem ser configuradas para usar identidades de propagação de identidade confiável. Para obter mais informações, consulte S3 Access Grants and corporate directory identities no Guia do usuário do Amazon S3.
- Os grupos de trabalho do Athena habilitados para o Centro de Identidade do IAM requerem que o Lake Formation seja configurado para usar as identidades do Centro de Identidade do IAM. Para obter informações sobre a configuração, consulte Integrating IAM Identity Center no Guia do desenvolvedor do AWS Lake Formation.
- Por padrão, as consultas atingem o tempo limite após 30 minutos em grupos de trabalho que usam a propagação de identidade confiável. É possível solicitar um aumento para o tempo limite da consulta, mas o tempo limite máximo em que uma consulta pode ser executada em grupos de trabalho de propagação de identidade confiável é de uma hora.
- As alterações de direitos de usuários ou de grupos em grupos de trabalho de propagação de identidade confiável podem demorar até uma hora para entrar em vigor.
- As consultas em um grupo de trabalho do Athena que usa propagação de identidade confiável não podem ser executadas usando o console do Athena de forma direta. As consultas devem ser executadas usando a interface do Athena em um EMR Studio que tenha o Centro de Identidade do IAM habilitado. Para obter mais informações sobre como usar o Athena no EMR Studio, consulte Use the Amazon Athena SQL editor in EMR Studio no Guia de gerenciamento do Amazon EMR.
- A propagação de identidade confiável não é compatível com os recursos do Athena apresentados a seguir.
  - Chaves de contexto `aws:CalledVia`.
  - Grupos de trabalho do Athena para Spark.
  - Acesso federado à API do Athena.
  - Acesso federado ao Athena usando o Lake Formation e os drivers JDBC e ODBC do Athena.
- É possível usar a propagação de identidade confiável com o Athena somente nas seguintes Regiões da AWS:
  - `us-east-2`: Leste dos EUA (Ohio)

- `us-east-1`: Leste dos EUA (Norte da Virgínia)
- `us-west-1`: Oeste dos EUA (Norte da Califórnia)
- `us-west-2`: Oeste dos EUA (Oregon)
- `af-south-1`: África (Cidade do Cabo)
- `ap-east-1`: Ásia-Pacífico (Hong Kong)
- `ap-southeast-3`: Ásia-Pacífico (Jacarta)
- `ap-south-1`: Ásia-Pacífico (Mumbai)
- `ap-northeast-3`: Asia Pacific (Osaka)
- `ap-northeast-2`: Ásia-Pacífico (Seul)
- `ap-southeast-1`: Ásia-Pacífico (Singapura)
- `ap-southeast-2`: Ásia-Pacífico (Sydney)
- `ap-northeast-1`: Ásia-Pacífico (Tóquio)
- `ca-central-1`: Canadá (Central)
- `eu-central-1`: Europa (Frankfurt)
- `eu-west-1`: Europa (Irlanda)
- `eu-west-2`: Europa (Londres)
- `eu-south-1`: Europa (Milão)
- `eu-west-3`: Europa (Paris)
- `eu-north-1`: Europa (Estocolmo)
- `me-south-1`: Oriente Médio (Bahrein)
- `sa-east-1`: América do Sul (São Paulo)

Permissões obrigatórias

O usuário do IAM do administrador que cria o grupo de trabalho habilitado para o Centro de Identidade do IAM no console do Athena deve ter as políticas apresentadas a seguir anexadas.

- A política gerenciada `AmazonAthenaFullAccess`. Para obter detalhes, consulte Política gerenciada pela AWS: AmazonAthenaFullAccess.
- A seguinte política em linha que permite ações do IAM e do Centro de Identidade do IAM:

```
{
    "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": [
                "*"
            ]
```

```
        }
    ]
}
```

Criação de um grupo de trabalho do Athena habilitado para o Centro de Identidade do IAM

O procedimento apresentado a seguir mostra as etapas e as opções relacionadas à criação de um grupo de trabalho do Athena habilitado para o Centro de Identidade do IAM. Para obter uma descrição das outras opções de configuração disponíveis para grupos de trabalho do Athena, consulte Criar um grupo de trabalho.

Como criar um grupo de trabalho habilitado para o SSO no console do Athena

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. No painel de navegação do console do Athena, escolha Workgroups (Grupos de trabalho).
3. Na página Workgroups (Grupos de trabalho), escolha Create workgroup (Criar grupo de trabalho).
4. Na página Criar grupo de trabalho, em Nome do grupo de trabalho, insira um nome para o grupo de trabalho.
5. Em Mecanismo de análise, use o padrão Athena SQL.
6. Em Autenticação, escolha Centro de Identidade do IAM.
7. Em Perfil de serviço para acesso ao Centro de Identidade do IAM, escolha um perfil de serviço existente ou crie um perfil novo.

   O Athena requer permissões para acessar o Centro de Identidade do IAM para você. Um perfil de serviço é necessário para que o Athena faça isso. Um perfil de serviço corresponde a um perfil do IAM gerenciado por você que autoriza um serviço da AWS a acessar outros serviços da AWS em seu nome. Para obter mais informações, consulte Criação de uma função para delegar permissões a um AWS serviço no Guia do usuário do IAM.
8. Expanda Configuração do resultado da consulta e, em seguida, insira ou escolha um caminho do Amazon S3 para Localização do resultado da consulta.
9. (Opcional) Escolha Criptografar resultados da consulta.
10. (Opcional) Escolha Criar prefixo do S3 baseado na identidade do usuário.

    Ao criar um grupo de trabalho habilitado para o Centro de Identidade do IAM, a opção Habilitar concessões de acesso ao S3 é selecionada por padrão. É possível usar as concessões de

acesso do Amazon S3 para controlar o acesso às localizações dos resultados das consultas do Athena (prefixos) no Amazon S3. Para obter mais informações sobre as concessões de acesso do Amazon S3, consulte Managing access with Amazon S3 Access Grants.

Em grupos de trabalho do Athena que usam a autenticação do Centro de Identidade do IAM, é possível habilitar a criação de localizações dos resultados das consultas baseados em identidade que são controlados pelas concessões de acesso do Amazon S3. Esses prefixos do Amazon S3 baseados na identidade do usuário permitem que os usuários em um grupo de trabalho do Athena mantenham os resultados de suas consultas isolados de outros usuários no mesmo grupo de trabalho.

Quando você habilita a opção de prefixo do usuário, o Athena anexa o ID do usuário como um prefixo de caminho do Amazon S3 à localização de saída do resultado da consulta para o grupo de trabalho (por exemplo, `s3://DOC-EXAMPLE-BUCKET/${user_id}`). Para usar esse recurso, você deve configurar concessões de acesso para permitir somente ao usuário a permissão para o local que tem o prefixo `user_id`. Para obter um exemplo da política de função de localização do Amazon S3 Access Grants que restringe o acesso aos resultados das consultas do Athena, consulte Exemplo de política de função.

> **Note**
>
> A seleção da opção de prefixo do S3 da identidade do usuário habilita automaticamente a opção de substituição de configurações do lado do cliente para o grupo de trabalho, conforme descrito na próxima etapa. A opção de substituir configurações do lado do cliente é um requisito para o recurso de prefixo da identidade do usuário.

11. Expanda Configurações e, em seguida, confirme se a opção Substituir configurações do lado do cliente está selecionada.

    Quando você seleciona Substituir configurações do lado do cliente, as configurações do grupo de trabalho são aplicadas no nível do grupo de trabalho para todos os clientes do grupo de trabalho. Para ter mais informações, consulte Configurações do grupo de trabalho sobrepõem as configurações do lado do cliente.

12. (Opcional) Faça quaisquer outras configurações necessárias, conforme descrito em Criar um grupo de trabalho.

13. Escolha Create workgroup (Criar grupo de trabalho).

14. Use a seção Grupos de trabalho do console do Athena para atribuir usuários ou grupos do seu diretório do IAM Identity Center ao seu grupo de trabalho do Athena habilitado para o IAM Identity Center.

Exemplo de política de função

O exemplo a seguir mostra uma política para uma função a ser anexada a um local do Amazon S3 Access Grants que restringe o acesso aos resultados da consulta do 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"
}
```

APIs de grupos de trabalho do Athena

Veja a seguir algumas das operações de API REST usadas para os grupos de trabalho Athena. Em todas as operações a seguir, com exceção de `ListWorkGroups`, você deve especificar um grupo de trabalho. Em outras operações, como `StartQueryExecution`, o parâmetro do grupo de trabalho é opcional e as operações não estão listadas aqui. Para obter uma lista de operações, consulte a Referência de API do Amazon Athena.

- CreateWorkGroup
- DeleteWorkGroup
- GetWorkGroup
- ListWorkGroups
- UpdateWorkGroup

Resolver problemas nos grupos de trabalho

Use as dicas a seguir para solucionar problemas com grupos de trabalho.

- Verifique as permissões para usuários individuais na sua conta. Eles devem ter acesso ao local para os resultados da consulta e ao grupo de trabalho no qual desejam executar consultas. Se eles quiserem alternar grupos de trabalho, vão precisar de permissões para ambos os grupos de trabalho. Para ter mais informações, consulte Políticas do IAM para acessar grupos de trabalho.
- Observe o contexto no console do Athena para ver em qual grupo de trabalho você vai executar as consultas. Se você usar o driver, defina o grupo de trabalho naquele de que você precisa. Para ter mais informações, consulte the section called “Especificar um grupo de trabalho no qual executar consultas”.
- Se você usar a API ou os drivers para executar as consultas, deverá especificar o local dos resultados das consultas usando uma destas formas: para consultas individuais, use OutputLocation (do lado do cliente). No grupo de trabalho, use WorkGroupConfiguration. Se o local não estiver especificado de nenhuma forma, o Athena emitirá um erro no runtime da consulta.
- Se você substituir as configurações do lado do cliente pelas configurações do grupo de trabalho, poderá encontrar erros com o local de resultados da consulta. Por exemplo, um usuário do grupo de trabalho pode não ter permissões para o local do grupo de trabalho no Amazon S3 para armazenar os resultados das consultas. Nesse caso, adicione as permissões necessárias.
- Os grupos de trabalho trazem mudanças no comportamento das operações da API. As chamadas para as operações de API existentes a seguir exigem que os usuários em sua conta tenham

permissões baseadas em recursos no IAM para os grupos de trabalho em que são criados. Se não houver permissões para o grupo de trabalho e para ações do grupo de trabalho, as seguintes ações da API emitirão `AccessDeniedException`: CreateNamedQuery, DeleteNamedQuery, GetNamedQuery, ListNamedQueries, StartQueryExecution, StopQueryExecution, ListQueryExecutions, GetQueryExecution, GetQueryResults e GetQueryResultsStream (essa ação de API está disponível somente para uso com o driver e, de outra forma, não é exposta para o uso público). Para obter mais informações, consulte Ações, recursos e chaves de condição do Amazon Athena na Referência de autorização do serviço.

As chamadas para as operações de API BatchGetQueryExecution e BatchGetNamedQuery retornam informações apenas sobre as execuções de consulta nos grupos de trabalho aos quais os usuários têm acesso. Se o usuário não tiver acesso ao grupo de trabalho, essas operações da API retornarão os IDs de consulta não autorizados como parte da lista de IDs não processados. Para ter mais informações, consulte the section called “ APIs de grupos de trabalho do Athena”.

- Se o grupo de trabalho em que uma consulta for executada estiver configurado com um local imposto para os resultados das consultas, não especifique um `external_location` para a consulta CTAS. O Athena emite um erro e falha com uma consulta que especifica um `external_location` nesse caso. Por exemplo, ocorrerá uma falha nesta consulta se você substituir as configurações do lado do cliente para o local dos resultados da consulta, forçando o grupo de trabalho a usar o próprio local: `CREATE TABLE <DB>.<TABLE1> WITH (format='Parquet', external_location='s3://DOC-EXAMPLE-BUCKET/test/') AS SELECT * FROM <DB>.<TABLE2> LIMIT 10;`

Você pode ver os erros a seguir. Esta tabela fornece uma lista de alguns dos erros relacionados a grupos de trabalho e sugere soluções.

Erros do grupo de trabalho

Erro	Ocorre quando...
query state CANCELED. Bytes scanned limit was exceeded. (o estado da consulta é CANCELED. O limite de bytes verificados foi excedido).	Uma consulta atinge um limite de dados por consulta e é cancelada. Considere reescrever a consulta para que leia menos dados ou entre em contato com o administrador da conta.
User: *arn:aws:iam::123456789012:user/abc* is not authorized to perform: athena:StartQueryExecution on resource:	Um usuário executa uma consulta em um grupo de trabalho, mas não tem acesso a ele.

Erro	Ocorre quando...
arn:aws:athena:us-east-1:123456789012:workgroup/workgroupname (O usuário: arn:aws:iam::123456789012:user/abc não está autorizado a executar: athena:StartQueryExecution no recurso: arn:aws:athena:us-east-1:123456789012:workgroup/workgroupname)	Atualize a política para ter acesso ao grupo de trabalho.
INVALID_INPUT. WorkGroup <name> is disabled. (INVALID_INPUT. O grupo de trabalho <nome> está desabilitado.)	Um usuário executa uma consulta em um grupo de trabalho, mas o grupo de trabalho está desabilitado. Seu grupo de trabalho poderia ser desabilitado pelo administrador. É possível também que você não tenha acesso a ele. Em ambos os casos, entre em contato com um administrador que tenha acesso para modificar grupos de trabalho.
INVALID_INPUT. WorkGroup <name> is not found. (INVALID_INPUT. O grupo de trabalho <nome> não foi localizado.)	Um usuário executa uma consulta em um grupo de trabalho, mas o grupo de trabalho não existe. Isso pode acontecer se o grupo de trabalho tiver sido excluído. Alterne para outro grupo de trabalho para executar a consulta.
InvalidRequestException: ao chamar a operação StartQueryExecution: nenhum local de saída fornecido. An output location is required either through the Workgroup result configuration setting or as an API input. (InvalidRequestException: ao chamar a operação StartQueryExecution: nenhum local de saída é fornecido. Um local de saída é necessário por meio da definição de configuração de resultados do grupo de trabalho ou como uma entrada de API).	Um usuário executa uma consulta com a API sem especificar o local dos resultados da consulta. Você deve definir o local de saída para resultados da consulta usando uma de duas formas: para consultas individuais, usando OutputLocation (no lado do cliente), ou, no grupo de trabalho, usando WorkGroup Configuration.

Erro	Ocorre quando...
The Create Table As Select query failed because it was submitted with an “external _location” property to an Athena Workgroup that enforces a centralized output location for all queries. Please remove the “external _location” property and resubmit the query. (A consulta “Create Table As Select” falhou porque foi enviada com uma propriedade “external_location” para um grupo de trabalho do Athena que impõe um local de saída centralizado para todas as consultas. Remova a propriedade “external_location” e reenvie a consulta).	Se o grupo de trabalho em que uma consulta for executada estiver configurado com um local imposto para os resultados da consulta e você especificar um `external_location` para a consulta do CTAS. Neste caso, remova o `external_location` e execute a consulta.
Cannot create prepared statement *`prepared_statement_name`* . The number of prepared statements in this workgroup exceeds the limit of 1000. (Não é possível criar a instrução preparada “prepare_statement_name”. O número de instruções preparadas neste grupo de trabalho excede o limite de mil).	O grupo de trabalho contém mais do que o limite de mil instruções preparadas. Para resolver esse problema, use DEALLOCATE PREPARE para remover uma ou mais instruções preparadas do grupo de trabalho. Como alternativa, crie um novo grupo de trabalho.

Controlar custos e monitorar consultas com métricas e eventos do CloudWatch

Os grupos de trabalho permitem que você defina os limites de controle de uso de dados por consulta ou por grupo de trabalho, configure alarmes quando esses limites são excedidos e publique métricas de consultas no CloudWatch.

Em cada grupo de trabalho, você pode:

- Configurar os Data usage controls (Controles de uso de dados) por consulta e por grupo de trabalho e estabelecer ações que serão executadas se as consultas violarem os limites.
- Visualizar e analisar as métricas de consulta e publicá-las no CloudWatch. Se você criar um grupo de trabalho no console, a configuração para publicação das métricas no CloudWatch será selecionada para você. Se você usar as operações de API, deve habilitar a publicação de

métricas. Quando são publicadas, as métricas são exibidas na guia Metrics (Métricas) do painel Workgroups (Grupos de trabalho). As métricas são desativadas por padrão para o grupo de trabalho principal.

Vídeo

O vídeo a seguir mostra como criar painéis personalizados e definir alarmes e acionadores com base nas métricas no CloudWatch. Você pode usar painéis pré-preenchidos diretamente do console do Athena para consumir essas métricas de consulta.

Monitoring Amazon Athena queries using Amazon CloudWatch (Monitorar consultas do Amazon Athena usando o Amazon CloudWatch)

Tópicos

- Habilitar métricas de consulta do CloudWatch
- Monitorar consultas do Athena com métricas do CloudWatch
- Monitorar consultas com eventos do Amazon EventBridge
- Monitorar as métricas de uso do Athena
- Definir limites de controle de uso de dados

Habilitar métricas de consulta do CloudWatch

Ao criar um grupo de trabalho no console, será selecionada por padrão a configuração para publicar as métricas de consulta no CloudWatch.

Para habilitar ou desabilitar as métricas de consulta no console do Athena para um grupo de trabalho

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.

3. No painel de navegação, escolha Global networks (Redes globais).
4. Escolha o link do grupo de trabalho que você deseja modificar.
5. Na página de detalhes do grupo de trabalho, escolha Edit (Editar).
6. Na seção Configurações, marque ou desmarque Publicar métricas de consulta no AWS CloudWatch.

Se você usar operações de API, a interface de linha de comando ou o aplicativo cliente com o driver JDBC para criar grupos de trabalho, de forma a habilitar a publicação de métricas de consulta, defina `PublishCloudWatchMetricsEnabled` como `true` em WorkGroupConfiguration. O exemplo a seguir mostra apenas a configuração das métricas e omite outra configuração:

```
"WorkGroupConfiguration": {
      "PublishCloudWatchMetricsEnabled": "true"
     ....
     }
```

Monitorar consultas do Athena com métricas do CloudWatch

O Athena publica as métricas relacionadas à consulta no Amazon CloudWatch, quando a opção Publish query metrics to CloudWatch (Publicar métricas de consulta no CloudWatch) está selecionada. Você pode criar painéis personalizados, definir alarmes e acionadores com base nas métricas no CloudWatch ou usar painéis pré-preenchidos diretamente no console do Athena.

Quando você habilitar as métricas de consulta nos grupos de trabalho, elas serão exibidas na guia Metrics (Métricas) do painel Workgroups (Grupos de trabalho) para cada grupo de trabalho do console do Athena.

O Athena publica as seguintes métricas no console do CloudWatch:

- `DPUAllocated`: o número total de DPUs (unidades de processamento de dados) provisionadas em uma reserva de capacidade para executar consultas.
- `DPUConsumed`: o número de DPUs consumidas ativamente por consultas em um estado de `RUNNING` em um dado momento em uma reserva. Métrica emitida somente quando o grupo de trabalho está associado a uma reserva de capacidade e inclui todos os grupos de trabalho associados a uma reserva.
- `DPUCount`: o número máximo de DPUs consumidas pela consulta, publicado exatamente uma vez quando a consulta é concluída.
- `EngineExecutionTime`: o número de milissegundos que a consulta levou para ser executada.
- `ProcessedBytes`: o número de bytes verificados pelo Athena por consulta DML.
- `QueryPlanningTime`: o número de milissegundos que o Athena levou para planejar o fluxo de processamento da consulta.
- `QueryQueueTime`: o número de milissegundos que a consulta ficou na fila de consultas aguardando recursos.
- `ServicePreProcessingTime`: o número de milissegundos que o Athena levou para pré-processar a consulta antes de enviá-la para o mecanismo de consulta.
- `ServiceProcessingTime`: o número de milissegundos que o Athena levou para processar os resultados da consulta depois que o mecanismo de consulta concluiu sua execução.
- `TotalExecutionTime`: o número de milissegundos que o Athena levou para executar uma consulta DDL ou DML.

Para obter descrições mais completas, consulte Lista de métricas e dimensões do CloudWatch para o Athena mais adiante neste documento.

Essas métricas têm as seguintes dimensões:

- `CapacityReservation`: o nome da reserva de capacidade usada para executar a consulta, se aplicável.
- `QueryState` – `SUCCEEDED`, `FAILED`, ou `CANCELED`

- `QueryType` – `DML`, `DDL`, ou `UTILITY`
- `WorkGroup`: nome do grupo de trabalho

O Athena publica a seguinte métrica no console do CloudWatch sob o namespace `AmazonAthenaForApacheSpark`:

- `DPUCount`: a quantidade de DPUs consumidas durante a sessão para executar os cálculos.

Essa métrica tem as seguintes dimensões:

- `SessionId`: o ID da sessão para a qual os cálculos são enviados.
- `WorkGroup`: o nome do grupo de trabalho.

Para obter mais informações, consulte Lista de métricas e dimensões do CloudWatch para o Athena posteriormente neste tópico. Para obter mais informações as métricas de uso do Athena, consulte Monitorar as métricas de uso do Athena.

Para visualizar as métricas de consulta para um grupo de trabalho no console

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.

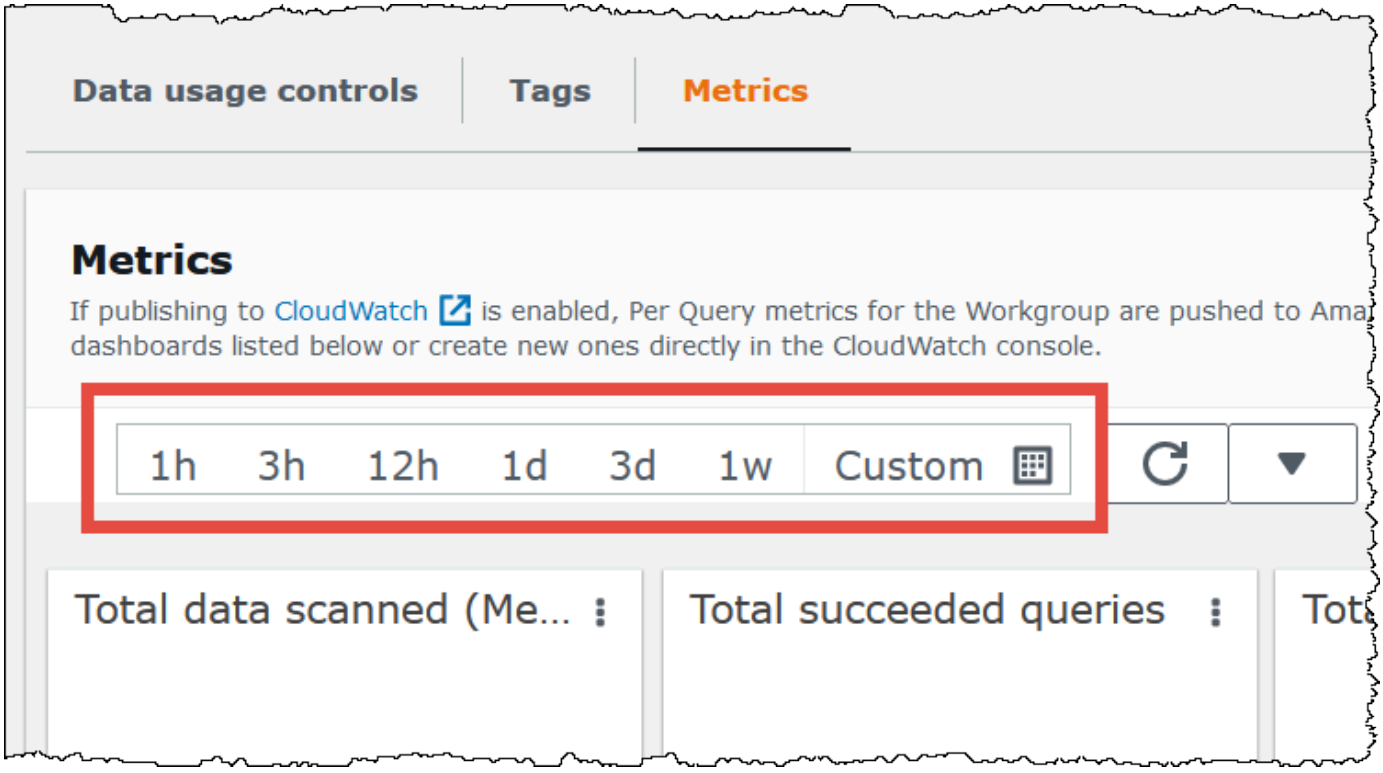
3. No painel de navegação, escolha Global networks (Redes globais).

4. Escolha o grupo de trabalho que você deseja na lista e escolha a guia Metrics (Métricas).

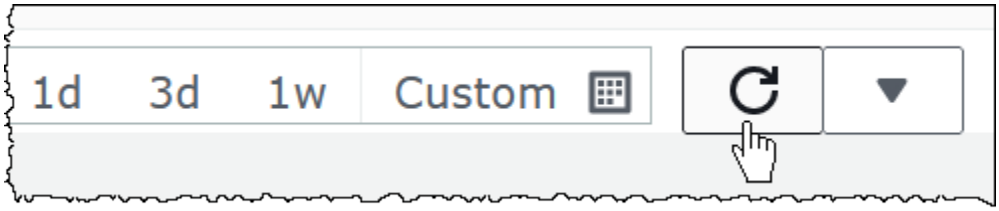
   O painel de métricas é exibido.

   > **Note**
   >
   > Se você ativou recentemente métricas para o grupo de trabalho e/ou não houve nenhuma atividade de consulta recente, os gráficos no painel poderão ficar vazios. A atividade de consulta é recuperada do CloudWatch dependendo do intervalo especificado na próxima etapa.

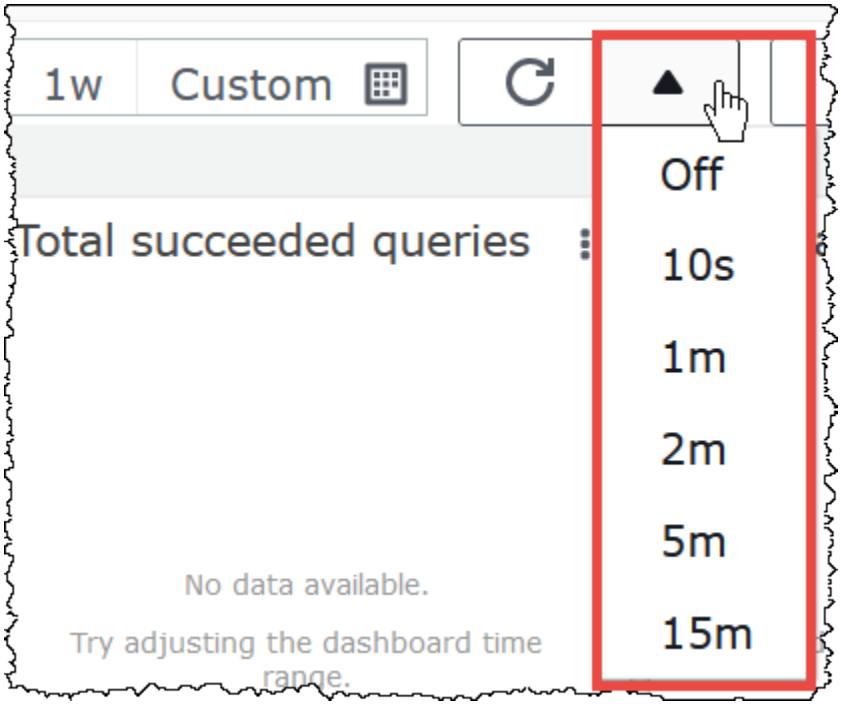
5. Na seção Metrics (Métricas), escolha o intervalo de métricas que o Athena deve usar para buscar as métricas de consulta do CloudWatch ou especifique um intervalo personalizado.



6. Para atualizar as métricas exibidas, escolha o ícone de atualização.



7. Clique na seta ao lado do ícone de atualização para escolher com que frequência você deseja que a exibição de métricas seja atualizada.



Para visualizar as métricas no console do Amazon CloudWatch

1. Abra o console CloudWatch em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, escolha Metrics (Métricas), All metrics (Todas as métricas).
3. Selecione o namespace AWS/Athena.

Para exibir métricas com a CLI

- Execute um destes procedimentos:
  - Para listar as métricas do Athena, abra um prompt de comando e use o seguinte comando:

```
aws cloudwatch list-metrics --namespace "AWS/Athena"
```

  - Para listar todas as métricas disponíveis, use o comando a seguir:

```
aws cloudwatch list-metrics"
```

Lista de métricas e dimensões do CloudWatch para o Athena

Se você habilitou as métricas do CloudWatch no Athena, ele enviará as métricas a seguir ao CloudWatch por grupo de trabalho. As métricas a seguir usam o namespace `AWS/Athena`.

Nome da métrica	Descrição
DPUAllocated	O número total de DPUs (unidades de processamento de dados) provisionadas em uma reserva de capacidade para executar consultas.
DPUConsumed	O número de DPUs consumidas ativamente por consultas em um estado de `RUNNING` em um dado momento em uma reserva. Essa métrica só é emitida quando o grupo de trabalho está associado a uma reserva de capacidade e inclui todos os grupos de trabalho associados a uma reserva. Se você mover um grupo de trabalho de uma reserva para outra, a métrica incluirá dados de quando o grupo de trabalho pertencia à primeira reserva. Para obter mais informações sobre reservas de capacidade, consulte Como gerenciar a capacidade de processamento de consulta.
DPUCount	O número máximo de DPUs consumidas pela consulta, publicado exatamente uma vez quando a consulta é concluída. Essa métrica só é emitida para grupos de trabalho associados a uma reserva de capacidade.
EngineExecutionTime	O número de milissegundos que a consulta levou para ser executada.
ProcessedBytes	O número de bytes verificados pelo Athena para cada consulta DML. Para consultas que foram canceladas (pelo usuários, ou automaticamente, se atingiram o limite), isso inclui a quantidad e de dados verificada antes da hora do cancelamento. Essa métrica não é relatada para consultas DDL.
QueryPlanningTime	O número de milissegundos que o Athena levou para planejar o fluxo de processamento da consulta. Isso inclui o tempo gasto recuperando partições de tabela da fonte de dados. Observe

Nome da métrica	Descrição
	que, como o mecanismo de consulta executa o planejamento da consulta, o tempo de planejamento de consulta é um subconjunto de EngineExecutionTime.
QueryQueueTime	O número de milissegundos que a consulta estava na fila de consultas aguardando recursos. Observe que, se ocorrerem erros temporários, a consulta poderá ser adicionada automaticamente de volta à fila.
ServicePreProcessingTime	O número de milissegundos que o Athena levou para pré-processar a consulta antes de enviá-la para o mecanismo de consulta.
ServiceProcessingTime	O número de milissegundos que o Athena levou para processar os resultados da consulta depois que o mecanismo de consulta concluiu a execução da consulta.
TotalExecutionTime	O número de milissegundos que o Athena levou para executar uma consulta DDL ou DML. TotalExecutionTime inclui QueryQueueTime, QueryPlanningTime, EngineExecutionTime e ServiceProcessingTime.

Essas métricas do Athena têm as dimensões a seguir.

Dimensão	Descrição
CapacityReservation	O nome da reserva de capacidade usada para executar a consulta, se aplicável. Quando uma reserva de capacidade não é usada, essa dimensão não retorna nenhum dado.
QueryState	O estado da consulta. Estatísticas válidas: SUCCEEDED, FAILED ou CANCELED.
QueryType	O tipo de consulta. Estatísticas válidas: `DDL`, `DML` ou `UTILITY`. O tipo de instrução de consulta que foi executada. `DDL` indica instruções de consulta

Dimensão	Descrição
	DDL (Data Definition Language). `DML` indica instruções de consulta DML (Data Manipulation Language), como `CREATE TABLE AS SELECT`. `UTILITY` indica instruções de consulta diferentes de DDL e DML, como `SHOW CREATE TABLE` ou `DESCRIBE TABLE`.
WorkGroup	O nome do grupo de trabalho.

Monitorar consultas com eventos do Amazon EventBridge

Você pode usar o Amazon Athena com o Amazon EventBridge para receber notificações em tempo real sobre o estado das consultas. Quando uma consulta enviada muda de estado, o Athena publica um evento no EventBridge com informações sobre a transição de estado da consulta. É possível gravar regras simples para eventos do seu interesse e realizar ações automatizadas quando um evento corresponder a uma regra. Por exemplo, é possível criar uma regra que invoca uma função do AWS Lambda quando uma consulta atinge um estado terminal. Os eventos são emitidos com base no melhor esforço.

Antes de criar regras de eventos para o Athena, faça o seguinte:

- Familiarize-se com os eventos, regras e destinos no EventBridge. Para obter mais informações, consulte O que é o Amazon EventBridge? Para obter mais informações sobre a configuração de regras, consulte Getting started with Amazon EventBridge.
- Crie os destinos para usar em suas regras de evento.

> **Note**
>
> Atualmente, o Athena oferece um tipo de evento, a alteração de estado da consulta do Athena, mas pode adicionar outros tipos de eventos e detalhes. Caso você esteja desserializando programaticamente dados JSON do evento, certifique-se de que a aplicação esteja preparada para lidar com propriedades desconhecidas caso propriedades adicionais sejam incluídas.

Formato de eventos do Athena

O item a seguir é o padrão básico de um evento do Amazon Athena.

```
{
    "source":[
        "aws.athena"
    ],
    "detail-type":[
        "Athena Query State Change"
    ],
    "detail":{
        "currentState":[
            "SUCCEEDED"
        ]
    }
}
```

Evento de alteração de estado da consulta do Athena

O exemplo a seguir mostra um evento de alteração de estado da consulta do Athena com o valor `currentState` de 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"
    }
```

```
}
```

O exemplo a seguir mostra um evento de alteração de estado da consulta do Athena com o valor `currentState` de `FAILED`. O bloco `athenaError` aparece somente quando `currentState` corresponde a `FAILED`. Para obter informações sobre os valores de `errorCategory` e `errorType`, consulte Catálogo de erros do 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"
    }
}
```

Propriedades de saída

A saída JSON inclui as seguintes propriedades.

Propriedade	Descrição
athenaError	Aparece somente quando currentState corresponde a FAILED. Contém informações sobre o erro ocorrido, incluindo a categoria do erro, o tipo de erro, a mensagem de erro e se a ação que levou ao erro pode ocorrer novamente. Os valores para cada um desses campos dependem da natureza do erro. Para obter informações sobre os valores de errorCategory e errorType , consulte Catálogo de erros do Athena.
versionId	O número da versão do esquema do objeto de detalhes.
currentState	O estado para o qual a consulta foi alterada no momento do evento.
previousState	O estado inicial da consulta no momento do evento.
statementType	O tipo de instrução da consulta executada.
queryExecutionId	O identificador exclusivo da execução da consulta.
workgroupName	O nome do grupo de trabalho no qual a consulta foi executada.
sequenceNumber	Um número continuamente crescente que permite a eliminação de duplicação e classificação de eventos de entrada que envolvem uma única execução de consulta. Quando eventos duplicados forem publicados para a mesma transição de estado, o valor sequenceNumber será o mesmo. Quando uma consulta apresentar mais de uma transição de estado, como consultas que apresentam raro reenfileiramento, é possível usar sequenceNumber para classificar eventos com valores idênticos de currentState e previousState .

Exemplo

O exemplo a seguir publica eventos em um tópico do Amazon SNS ao qual você se inscreveu. Quando o Athena é consultado, você recebe um e-mail. O exemplo pressupõe que o tópico do Amazon SNS existe e que você está inscrito nele.

Para publicar eventos do Athena em um tópico do Amazon SNS

1. Crie um destino para o tópico do Amazon SNS. Conceda a permissão `events.amazonaws.com` à entidade principal do serviço de eventos do EventBridge para publicar no tópico do Amazon SNS, como no exemplo a seguir.

```
{
    "Effect":"Allow",
    "Principal":{
        "Service":"events.amazonaws.com"
    },
    "Action":"sns:Publish",
    "Resource":"arn:aws:sns:us-east-1:111111111111:your-sns-topic"
}
```

2. Use o comando da AWS CLI `events put-rule` para criar uma regra para eventos do Athena, como no exemplo a seguir.

```
aws events put-rule --name {ruleName} --event-pattern '{"source": ["aws.athena"]}'
```

3. Use o comando da AWS CLI `events put-targets` para anexar o destino do tópico do Amazon SNS à regra, conforme o exemplo a seguir.

```
aws events put-targets --rule {ruleName} --targets Id=1,Arn=arn:aws:sns:us-
east-1:111111111111:your-sns-topic
```

4. Consulte o Athena e observe o destino sendo invocado. Você receberá e-mails correspondentes do tópico do Amazon SNS.

Uso do Notificações de Usuários da AWS com o Amazon Athena

É possível usar o Notificações de Usuários da AWS para configurar canais de entrega para receber notificações sobre eventos do Amazon Athena. Você recebe uma notificação quando um evento corresponde a uma regra especificada. É possível receber notificações de eventos por meio de vários canais, incluindo e-mail, notificações de chat do AWS Chatbot ou notificações por push do AWS Console Mobile Application. Você também pode visualizar notificações na Central de Notificações do Console. O Notificações de Usuários oferece é compatível com agregação, o que pode reduzir o número de notificações que você recebe durante eventos específicos.

Para mais informações, consulte o Guia do usuário do Notificações de Usuários da AWS.

Monitorar as métricas de uso do Athena

Você pode usar as métricas de uso do CloudWatch para fornecer visibilidade de como a conta usa os recursos, exibindo o uso do serviço atual nos gráficos e painéis do CloudWatch.

Para o Athena, as métricas de disponibilidade para uso correspondem às cotas de AWS service (Serviço da AWS) para o Athena. Também é possível configurar alarmes que alertem você quando o uso se aproximar de uma cota de serviço. Para obter mais informações sobre cotas de serviço do Athena, consulte Service Quotas. Para obter mais informações sobre métricas de uso do AWS, consulte métricas de uso da AWS no Guia do usuário do Amazon CloudWatch.

O Athena publica as métricas a seguir no namespace `AWS/Usage`.

Nome da métrica	Descrição
`ResourceCount`	A soma de todas as consultas enfileiradas e em execução por Região da AWS por conta, separada por tipo de consulta (DML ou DDL). O máximo é a única estatística útil para essa métrica. Essa métrica é publicada periodicamente, a cada minuto. Se você não estiver executando nenhuma consulta, a métrica não reportará nada (nem mesmo 0). A métrica será publicada somente se as consultas ativas estiverem sendo executadas no momento em que a métrica é verificada.

As dimensões a seguir são usadas para refinar as métricas de uso publicadas pelo Athena..

Dimensão	Descrição
`Service`	O nome do AWS service (Serviço da AWS) que contém o recurso. Para o Athena, o valor dessa dimensão é `Athena`.
`Resource`	O tipo de recurso que está em execução. O valor de recurso para o uso de consulta do Athena é `ActiveQueryCount` .
`Type`	O tipo de entidade que está sendo relatada. Atualmente, o único valor válido para métricas de uso do Athena é `Resource`.

Dimensão	Descrição
`Class`	A classe do recurso sob acompanhamento. Para o Athena, `Class` pode ser um `DML` ou um `DDL`.

Visualizar métricas de uso de recursos do Athena no console do CloudWatch

Você pode usar o console do CloudWatch para ver um gráfico das métricas de uso do Athena e configurar alarmes que o alertarão quando o uso se aproximar de uma cota de serviço.

Para ver as métricas de uso dos recursos do Athena

1. Abra o console CloudWatch em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, escolha Metrics (Métricas), All metrics (Todas as métricas).
3. Escolha Uso e depois Por recurso da AWS.

   A lista das métricas de uso da cota de serviço é exibida.
4. Marque a caixa de seleção ao lado de Athena e ActiveQueryCount.
5. Escolha a guia Graphed metrics (Métricas em gráfico).

   O gráfico acima exibe o uso atual do recurso da AWS.

Para obter informações sobre como adicionar cotas de serviço ao grafo e definir um alarme que notifique você caso o uso se aproxime da cota de serviço, consulte Visualizar cotas de serviço e definir alarmes no Guia do usuário do Amazon CloudWatch. Para obter informações sobre como definir limites de uso por grupo de trabalho, consulte Definir limites de controle de uso de dados.

Definir limites de controle de uso de dados

O Athena permite a definição de dois tipos de controles de custo: limite por consulta e limite por grupo de trabalho. Para cada grupo de trabalho, você só pode definir um limite por consulta e vários limites por grupo de trabalho.

- O limite de controle por consulta especifica a quantidade total de dados analisados por consulta. Se qualquer consulta que for executada no grupo de trabalho exceder o limite, ela será cancelada. Você pode criar apenas um limite de controle por consulta em um grupo de trabalho e aplicá-lo a cada consulta que for executada nele. Edite o limite se precisar alterá-lo. Para obter etapas detalhadas, consulte Para criar um controle de uso de dados por consulta.

- O limite de controle de uso de dados no âmbito do grupo de trabalho especifica a quantidade total de dados verificados para todas as consultas que são executadas neste grupo de trabalho durante o período especificado. Você pode criar vários limites por grupo de trabalho. O limite de consulta no âmbito do grupo de trabalho permite que você defina vários limites em agregados de hora e dia nos dados analisados por consultas em execução no grupo de trabalho.

  Se a quantidade somada de dados verificados exceder o limite, você poderá enviar uma notificação para um tópico do Amazon SNS. Para fazer isso, configure um alarme do Amazon SNS e uma ação no console do Athena para notificar um administrador quando o limite for violado. Para obter etapas detalhadas, consulte Para criar um controle de uso de dados por grupo de trabalho. Você também pode criar um alarme e uma ação com base em qualquer métrica que o Athena publicar pelo console do CloudWatch. Por exemplo, você pode configurar um alerta para um determinado número de consultas com falha. Esse alerta pode acionar um e-mail para um administrador se o número ultrapassar um determinado limite. Se o limite for excedido, uma ação enviará uma notificação de alarme do Amazon SNS aos usuários especificados.

  Outras ações que você pode executar:

  - Invocar uma função do Lambda. Para obter mais informações, consulte Invocar funções do Lambda usando notificações do Amazon SNS no Guia do desenvolvedor do Amazon Simple Notification Service.
  - Desabilite o grupo de trabalho para interromper a execução de outras consultas. Para obter as etapas, consulte Habilitar e desabilitar um grupo de trabalho.

Os limites por consulta e por grupo de trabalho são independentes um do outro. Uma ação especificada será executada sempre que o limite for excedido. Se dois ou mais usuários executarem consultas ao mesmo tempo no mesmo grupo de trabalho, é possível que cada consulta não exceda nenhum dos limites especificados, mas a soma dos dados verificados exceda o limite de uso de dados por grupo de trabalho. Nesse caso, um alarme do Amazon SNS será enviado ao usuário.

Para criar um controle de uso de dados por consulta

O limite de controle por consulta especifica a quantidade total de dados analisados por consulta. Se qualquer consulta que for executada no grupo de trabalho exceder o limite, ela será cancelada. As consultas canceladas são cobradas de acordo com os Preços do Amazon Athena.

> **Note**
>
> No caso de consultas canceladas ou com falha, o Athena pode já ter gravado resultados parciais no Amazon S3. Nesses casos, o Athena não excluirá os resultados parciais do prefixo do Amazon S3 em que os resultados são armazenados. Você deve remover o prefixo do Amazon S3 com os resultados parciais. O Athena usa carregamentos fracionados do Amazon S3 para gravar dados do Amazon S3. Recomendamos que você defina a política de ciclo de vida do bucket para encerrar os carregamentos fracionados nos casos em que as consultas falharem. Para obter mais informações, consulte Interromper um multipart upload incompleto usando uma política de ciclo de vida de bucket no Guia do usuário do Amazon Simple Storage Service.

Você pode criar apenas um limite de controle por consulta em um grupo de trabalho e aplicá-lo a cada consulta que for executada nele. Edite o limite se precisar alterá-lo.

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.

3. No painel de navegação, escolha Global networks (Redes globais).
4. Escolha o nome do grupo de trabalho na lista.
5. Na guia Data usage controls (Controles de uso de dados), na seção Per query data usage control (Controle de uso de dados por consulta), escolha Manage (Gerenciar).

6. Na página Manage per query data usage control (Gerenciar controle de uso de dados por consulta), especifique os seguintes valores:

   - Para Data limit (Limite de dados), especifique um valor entre 10 MB (mínimo) e 7 EB (máximo).

     > **Note**
     >
     > Esses são os limites impostos pelo console para controles de uso de dados dentro dos grupos de trabalho. Eles não representam nenhum limite de consulta no Athena.

   - Para unidades, selecione o valor unitário na lista suspensa (por exemplo, Kilobytes KB ou Exabytes EB).

   A ação padrão é cancelar a consulta se ela exceder o limite. Essa configuração não pode ser alterada.
7. Escolha Salvar.

Para criar ou editar um alerta de uso de dados por grupo de trabalho

Você pode definir vários limites de alerta quando as consultas em execução em um grupo de trabalho verificam uma quantidade especificada de dados dentro de um período específico. Os alertas são implementados usando alarmes do Amazon CloudWatch e se aplicam a todas as consultas no grupo de trabalho. Quando um limite é atingido, você pode fazer com que o Amazon SNS envie um e-mail aos usuários que você especificar. As consultas não são canceladas automaticamente quando um limite é atingido.

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.
3. No painel de navegação, escolha Global networks (Redes globais).
4. Escolha o nome do grupo de trabalho na lista.
5. Selecione Edit (Editar) para editar as configurações do grupo de trabalho.
6. Role para baixo até Workgroup data usage alerts - optional (Alertas de uso de dados do grupo de trabalho - opcional) e expanda essa opção.
7. Escolha Add alert (Adicionar alerta).

8. Em Data usage threshold configuration (Configuração de limite de uso de dados), especifique os valores da maneira a seguir:

   - Em Data threshold (Limite de dados), especifique um número e, em seguida, selecione um valor unitário na lista suspensa.
   - Em Time period (Período), escolha um período na lista suspensa.
   - Em SNS topic selection (Seleção de tópico do SNS), escolha um tópico do Amazon SNS na lista suspensa. Se preferir, escolha Create an SNS topic (Criar um tópico do SNS) para acessar diretamente o console do Amazon SNS, crie o tópico do Amazon SNS e configure uma assinatura para ele para um dos usuários em sua conta do Athena. Para obter mais informações, consulte Conceitos básicos do Amazon SNS no Guia do desenvolvedor do Amazon Simple Notification Service.

9. Escolha Add alert (Adicionar alerta) se estiver criando um alerta ou Save (Salvar) para salvar um alerta existente.

Como gerenciar a capacidade de processamento de consulta

Use as reservas de capacidade para obter capacidade de processamento dedicada para as consultas que você executa no Athena. Com as reservas de capacidade, é possível aproveitar os recursos de gerenciamento de workload que ajudam a priorizar, controlar e escalar suas workloads interativas mais importantes. Por exemplo, você pode adicionar capacidade a qualquer momento para aumentar o número de consultas que poderão ser executadas simultaneamente, controlar quais workloads poderão usar a capacidade e compartilhar a capacidade entre as workloads. A capacidade é totalmente gerenciada pelo Athena e armazenada pelo tempo que você precisar. A configuração é fácil, e não é necessário alterar as instruções SQL.

Para obter capacidade de processamento para suas consultas, crie uma reserva de capacidade, especifique o número de unidades de processamento de dados (DPUs) necessárias e atribua um ou mais grupos de trabalho à reserva.

Os grupos de trabalho têm um papel importante no uso das reservas de capacidade. Os grupos de trabalho permitem organizar consultas em agrupamentos lógicos. Com as reservas de capacidade, atribua seletivamente a capacidade aos grupos de trabalho para controlar como as consultas de cada grupo de trabalho se comportam e como são faturadas. Para obter mais informações sobre grupos de trabalho, consulte Usar grupos de trabalho para controlar o acesso a consultas e os custos.

Atribuir grupos de trabalho às reservas permite que você dê prioridade às consultas enviadas aos grupos de trabalho atribuídos. Por exemplo, é possível alocar capacidade para um grupo de trabalho usado para consultas urgentes de relatórios financeiros para isolá-las de consultas menos essenciais em outro grupo de trabalho. Isso permite a execução consistente de consultas para workloads essenciais, permitindo que outras workloads sejam executadas de forma independente.

Você pode usar reservas de capacidade e grupos de trabalho juntos para cumprir diferentes requisitos. Estes são alguns dos cenários de exemplo:

- Isolamento: para isolar uma workload importante, atribua um único grupo de trabalho a uma reserva. Somente as consultas do grupo de trabalho designado usam a capacidade de processamento da reserva escolhida.
- Compartilhamento: várias workloads compartilham a capacidade de uma única reserva. Por exemplo, se você quiser um custo mensal previsível para um conjunto específico de workloads, é possível atribuir vários grupos de trabalho a uma única reserva. Os grupos de trabalho designados compartilham a capacidade da reserva.
- Modelo misto: você pode usar reservas de capacidade e cobrança por consulta na mesma conta e ao mesmo tempo. Por exemplo, para garantir a execução confiável de consultas que oferecem suporte a uma aplicação de produção, atribua um grupo de trabalho para essas consultas a uma reserva de capacidade. Para desenvolver as consultas antes de passá-las para o grupo de trabalho de produção, use um grupo de trabalho separado que não esteja associado a uma reserva e que, portanto, use cobrança por consulta.

Noções básicas sobre DPUs

A capacidade é medida em unidades de processamento de dados (DPUs). As DPUs representam os recursos de computação e memória usados pelo Athena para acessar e processar dados para você. Uma DPU fornece 4 vCPUs e 16 GB de memória. O número especificado de DPUs influencia o número de consultas que você pode executar simultaneamente. Por exemplo, uma reserva com 256 DPUs permite aproximadamente duas vezes mais consultas simultâneas do que uma reserva com 128 DPUs.

É possível criar até 100 reservas de capacidade com até mil DPUs por conta e por região. O número mínimo de DPUs que você pode solicitar é 24. Caso precise de mais de mil DPUs para seu caso de uso, entre em contato com athena-feedback@amazon.com.

Para obter informações sobre como fazer a estimativa de seus requisitos de capacidade, consulte Determinar requisitos de capacidade. Para obter informações de preço, consulte Preços do Amazon Athena.

Considerações e limitações

- A propriedade requer o mecanismo Athena versão 3.
- Um único grupo de trabalho pode ser atribuído a, no máximo, uma reserva por vez, e você pode adicionar no máximo 20 grupos de trabalho a uma reserva.
- Não é possível adicionar grupos de trabalho habilitados para o Spark a uma reserva de capacidade.
- Para excluir um grupo de trabalho que tenha sido atribuído a uma reserva, remova antes o grupo de trabalho da reserva.
- O número mínimo de DPUs que você pode provisionar é 24.
- É possível criar até 100 reservas de capacidade com até mil DPUs por conta e por região.
- As solicitações de capacidade não são garantidas e podem levar até 30 minutos para serem concluídas.
- É cobrado um período mínimo de 1 hora por reserva. Depois de 1 hora, a capacidade é cobrada por minuto. Para obter informações de preço, consulte Preços do Amazon Athena.
- A capacidade reservada não é transferível para outra reserva de capacidade, Conta da AWS ou Região da AWS.
- Consultas DDL sobre reservas de capacidade consomem DPUs.
- As consultas executadas na capacidade provisionada não contam para os limites ativos de consulta de DDL e DML.
- Se todas as DPUs estiverem em uso, as consultas enviadas entrarão na fila. Essas consultas não são rejeitadas nem vão para a capacidade sob demanda.
- A métrica do `DPUConsumed` CloudWatch é por grupo de trabalho, não por reserva. Assim, se você mover um grupo de trabalho de uma reserva para outra, a métrica `DPUConsumed` incluirá dados de quando o grupo de trabalho pertencia à primeira reserva. Para obter mais informações sobre o uso de métricas do CloudWatch no Athena, consulte Monitorar consultas do Athena com métricas do CloudWatch.
- No momento, o recurso está disponível nas seguintes Regiões da AWS:
  - Leste dos EUA (Ohio)
  - Leste dos EUA (N. da Virgínia)

- Oeste dos EUA (Oregon)
- Ásia-Pacífico (Singapura)
- Ásia-Pacífico (Sydney)
- Ásia-Pacífico (Tóquio)
- Europa (Irlanda)
- Europa (Estocolmo)

Tópicos

- Determinar requisitos de capacidade
- Criar reservas de capacidade
- Gerenciar reservas
- Políticas do IAM para reservas de capacidade
- APIs de reserva de capacidade do Athena

Determinar requisitos de capacidade

Antes de criar uma reserva de capacidade, é possível fazer uma estimativa da capacidade necessária para atribuir a ela o número correto de DPUs. E depois que a reserva estiver em uso, convém verificar se a reserva tem capacidade insuficiente ou excessiva. Este tópico descreve as técnicas que você pode usar para fazer essas estimativas e também descreve algumas ferramentas da AWS para avaliar o uso e o custo.

Tópicos

- Fazer a estimativa da capacidade necessária
- Sinais da necessidade de mais capacidade
- Como verificar a capacidade ociosa
- Ferramentas para avaliar requisitos de capacidade e custo

Fazer a estimativa da capacidade necessária

Ao estimar os requisitos de capacidade, é útil considerar duas perspectivas: a quantidade de capacidade que uma consulta específica pode exigir e a quantidade de capacidade que você pode precisar em geral.

Fazer a estimativa dos requisitos de capacidade por consulta

Para determinar o número de DPUs que uma consulta pode exigir, é possível usar as seguintes diretrizes:

- As consultas DDL consomem 4 DPUs.
- As consultas DML normalmente consomem entre 4 e 124 DPUs.

O Athena determina o número de DPUs necessárias para uma consulta DML quando a consulta é enviada. O número varia conforme o tamanho dos dados, o formato de armazenamento, a estrutura da consulta e outros fatores. Geralmente, o Athena tenta selecionar o número de DPUs mais baixo e mais eficiente. Se o Athena determinar que é necessário obter mais capacidade computacional para que a consulta seja concluída com êxito, ele aumentará o número de DPUs atribuídas à consulta.

Como fazer a estimativa dos requisitos de capacidade específicos da workload

Para determinar a quantidade de capacidade necessária para executar várias consultas ao mesmo tempo, considere as diretrizes gerais na tabela a seguir:

Consultas simultâneas	DPUs necessárias
10	40 ou mais
20	96 ou mais
30 ou mais	240 ou mais

O número real de DPUs necessárias depende de suas metas e padrões de análise. Por exemplo, se você quiser que as consultas comecem imediatamente sem filas, determine o pico de demanda simultânea de consultas e provisione o número de DPUs de acordo.

Você pode provisionar menos DPUs do que sua demanda de pico, mas o enfileiramento poderá ocorrer quando ocorrer o pico de demanda. Quando ocorre o enfileiramento, o Athena mantém as consultas em uma fila e as executa quando a capacidade torna-se disponível.

Se sua meta for executar consultas dentro de um orçamento fixo, você poderá usar a Calculadora de preços da AWS para determinar o número de DPUs que cabem em seu orçamento.

Por fim, lembre-se de que o tamanho dos dados, o formato de armazenamento e a forma como uma consulta é escrita influenciam as DPUs necessárias para uma consulta. Para aumentar a performance da consulta, é possível compactar ou particionar os dados ou convertê-los para formatos em colunas. Para ter mais informações, consulte Ajuste de performance no Athena.

Sinais da necessidade de mais capacidade

Mensagens de erro de capacidade insuficiente e enfileiramento de consultas são duas indicações de que a capacidade atribuída é inadequada.

Se as consultas falharem com uma mensagem de erro de capacidade insuficiente, o número de DPUs da reserva de capacidade provavelmente será muito baixo para a consulta. Por exemplo, se você tiver uma reserva com 24 DPUs e executar uma consulta que exija mais de 24 DPUs, a consulta falhará. Para monitorar esse erro de consulta, use os eventos do EventBridge do Athena. Tente adicionar mais DPUs e executar a consulta novamente.

Se muitas consultas estiverem em fila, significa que a capacidade foi totalmente utilizada por outras consultas. Para reduzir o enfileiramento, realize uma destas ações:

- Adicione DPUs à reserva para aumentar a simultaneidade de consultas.
- Remova grupos de trabalho da reserva para liberar capacidade para outras consultas.

Para verificar se há excesso de filas de consultas, use a métrica do CloudWatch de tempo de fila de consultas do Athena para os grupos de trabalho na sua reserva de capacidade. Se o valor estiver acima de seu limite preferencial, você poderá adicionar DPUs à reserva de capacidade.

Como verificar a capacidade ociosa

Para verificar a capacidade ociosa, você pode diminuir o número de DPUs na reserva ou aumentar a workload e observar os resultados.

Para verificar a capacidade ociosa

1. Execute um destes procedimentos:
   - Reduza o número de DPUs da reserva (reduza os recursos disponíveis)
   - Adicione grupos de trabalho à reserva (aumente a workload)
2. Use o CloudWatch para medir o tempo da fila de consultas.
3. Se o tempo de espera ultrapassar um nível desejável, realize uma destas ações:

- Remova grupos de trabalho
- Adicione DPUs à reserva de capacidade

4. Após cada alteração, verifique a performance e o tempo da fila de consultas.
5. Continue ajustando a workload ou a contagem de DPUs para atingir o equilíbrio desejado.

Caso não queira manter a capacidade fora de um período de tempo preferencial, você pode cancelar a reserva e criar outra reserva posteriormente. Porém, mesmo que você tenha cancelado recentemente a capacidade de outra reserva, as solicitações de nova capacidade não são garantidas e as novas reservas demoram para serem criadas.

Ferramentas para avaliar requisitos de capacidade e custo

Você pode usar os serviços e recursos da AWS a seguir para medir o uso e os custos do Athena.

Métricas do CloudWatch

É possível configurar o Athena para publicar métricas relacionadas a consultas no Amazon CloudWatch no nível do grupo de trabalho. Depois que você habilitar as métricas para o grupo de trabalho, as métricas das consultas do grupo de trabalho serão exibidas no console do Athena na página de detalhes do grupo de trabalho.

Para obter informações sobre as métricas do Athena publicadas no CloudWatch e suas dimensões, consulte Monitorar consultas do Athena com métricas do CloudWatch.

Métricas de uso do CloudWatch

Você pode usar as métricas de uso do CloudWatch para fornecer visibilidade de como a conta usa os recursos, exibindo o uso do serviço atual nos gráficos e painéis do CloudWatch. Para o Athena, as métricas de disponibilidade para uso correspondem às cotas de serviço da AWS para o Athena. Também é possível configurar alarmes que alertem você quando o uso se aproximar de uma cota de serviço.

Para ter mais informações, consulte Monitorar as métricas de uso do Athena.

Eventos do Amazon EventBridge

Você pode usar o Amazon Athena com o Amazon EventBridge para receber notificações em tempo real sobre o estado das consultas. Quando o estado de uma consulta enviada é alterado, o Athena

publica um evento no EventBridge que contém informações sobre a transição de estado da consulta. É possível gravar regras simples para eventos do seu interesse e realizar ações automatizadas quando um evento corresponder a uma regra.

Para obter mais informações, consulte os recursos a seguir.

- Monitorar consultas com eventos do Amazon EventBridge
- O que é o Amazon EventBridge?
- Eventos do Amazon EventBridge

Tags

No Athena, as reservas de capacidade são compatíveis com etiquetas. Uma etiqueta consiste em uma chave e um valor. Para rastrear seus custos no Athena, você pode usar etiquetas de alocação de custos geradas pela AWS. A AWS utiliza as etiquetas de alocação de custos para organizar os custos de recursos em seu Relatório de Custos e Uso. Isso facilita a categorização e o controle de seus custos na AWS. Para ativar as etiquetas de alocação de custos para o Athena, use o console do AWS Billing and Cost Management.

Para obter mais informações, consulte os recursos a seguir.

- Etiquetar recursos do Athena
- Como ativar as etiquetas de alocação de custos geradas pela AWS
- Usar etiquetas de alocação de custos da AWS

Criar reservas de capacidade

Para começar, crie uma reserva de capacidade com o número de DPUs necessárias e atribua um ou mais grupos de trabalho que usarão essa capacidade nas consultas. É possível ajustar sua capacidade posteriormente, conforme necessário, para fornecer uma performance mais consistente ou gerenciar melhor os custos. Para obter informações sobre como fazer a estimativa de seus requisitos de capacidade, consulte Determinar requisitos de capacidade.

> **Important**
>
> As solicitações de capacidade não são garantidas e podem levar até 30 minutos para serem concluídas.

Para criar uma reserva de capacidade

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.
3. Escolha Administração, Reservas de capacidade.
4. Escolha Criar reserva de capacidade.
5. Na página Criar reserva de capacidade, em Nome da reserva de capacidade, insira o nome. O nome deve ser exclusivo, conter de 1 a 128 caracteres e usar apenas os seguintes caracteres: A-Z, 0-9, _ (sublinhado), . (ponto) e - (hífen). Não é possível alterar o nome depois de criar a reserva.
6. Em DPU, escolha ou insira o número de unidades de processamento de dados (DPUs) que deseja em incrementos de 4. Para ter mais informações, consulte Noções básicas sobre DPUs.
7. (Opcional) Expanda a opção Etiquetas e escolha Adicionar nova etiqueta para adicionar um ou mais pares de chave/valor personalizados para associar ao recurso de reserva de capacidade. Para ter mais informações, consulte Etiquetar recursos do Athena.
8. Escolha Revisar.
9. No prompt Confirmar criação de reserva de capacidade, confirme o número de DPUs, a Região da AWS e outras informações. Se você aceitar, escolha Enviar.

   Na página de detalhes, o Status da reserva de capacidade é exibido como Pendente. Quando sua capacidade de reserva estiver disponível para executar consultas, o status será exibido como Ativa.

Neste ponto, você está pronto para adicionar um ou mais grupos de trabalho à reserva. Para obter as etapas, consulte Adicionar grupos de trabalho a uma reserva.

Gerenciar reservas

Visualize e gerencie suas reservas de capacidade na página Reservas de capacidade. Você pode realizar tarefas de gerenciamento, como adicionar ou reduzir DPUs, modificar atribuições de grupos de trabalho e etiquetar ou cancelar reservas.

Para visualizar e gerenciar reservas de capacidade

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.

2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.
3. Escolha Administração, Reservas de capacidade.
4. Na página de reserva de capacidade, é possível realizar estas tarefas:

   - Para criar uma reserva de capacidade, escolha Criar reserva de capacidade.
   - Use a caixa de pesquisa para filtrar reservas por nome ou número de DPUs.
   - Escolha o menu suspenso de status para filtrar por status de reserva de capacidade (por exemplo, Ativa ou Cancelada). Para obter informações sobre status de reservas, consulte Noções básicas sobre status de reserva.
   - Para visualizar os detalhes de uma reserva de capacidade, escolha o link da reserva. A página de detalhes da reserva contém opções para editar a capacidade, adicionar grupos de trabalho, remover grupos de trabalho e cancelar a reserva.
   - Para editar uma reserva (por exemplo, adicionar ou remover DPUs), selecione o botão da reserva e escolha Editar.
   - Para cancelar uma reserva, selecione o botão da reserva e escolha Cancelar.

Noções básicas sobre status de reserva

A tabela a seguir descreve os possíveis valores de status de uma reserva de capacidade.

Status	Descrição
Pendente	O Athena está processando sua solicitação de capacidade. A capacidade não está pronta para executar consultas.
Ativo	A capacidade está disponível para executar consultas.
Com falha	A solicitação de capacidade não foi concluída com êxito. O cumprimento das solicitações de capacidade não é garantido. As reservas com falha contam para os limites de DPU da conta. Para liberar o uso, é necessário cancelar a reserva.
Atualização pendente	O Athena está processando uma alteração na reserva. Por exemplo, esse status ocorre depois que você edita a reserva para adicionar ou remover DPUs.

Status	Descrição
Cancelando	O Athena está processando uma solicitação para cancelar a reserva. As consultas que ainda estiverem em execução nos grupos de trabalho que estavam usando a reserva poderão ser concluídas, mas outras consultas do grupo de trabalho usarão capacidade sob demanda (não provisionada).
Cancelado	O cancelamento da reserva de capacidade foi concluído. As reservas canceladas permanecerão no console por 45 dias. Após 45 dias, o Athena excluirá a reserva. Durante os 45 dias, não será possível reaproveitar ou reutilizar a reserva, mas você poderá consultar as etiquetas e visualizar os detalhes para referência histórica. Não é garantido que a capacidade cancelada será reservada novamente em uma data futura. A capacidade não pode ser transferida para outra reserva, Conta da AWS ou Região da AWS.

Noções básicas sobre DPUs ativas e DPUs de destino

Na lista de reservas de capacidade no console Athena, a reserva exibe dois valores de DPU: DPU ativa e DPU de destino.

- DPU ativa: o número de DPUs que estão disponíveis na reserva para executar consultas. Por exemplo, se você solicitar 100 DPUs e a solicitação for atendida, a DPU ativa exibirá 100.
- DPU de destino: o número de DPUs para as quais a reserva está sendo transferida. A DPU de destino exibe um valor diferente da DPU ativa quando uma reserva está sendo criada ou há um aumento ou uma diminuição no número de DPUs pendente.

Por exemplo, depois que você enviar uma solicitação para criar uma reserva com 24 DPUs, o Status da reserva será Pendente, a DPU ativa será 0 e a DPU de destino será 24.

Se você tiver uma reserva com 100 DPUs e editar a reserva para solicitar um aumento de 20 DPUs, o Status será Atualização pendente, a DPU ativa será 100 e a DPU de destino será 120.

Se você tiver uma reserva com 100 DPUs e editar a reserva para solicitar uma redução de 20 DPUs, o Status será Atualização pendente, a DPU ativa será 100 e a DPU de destino será 80.

Durante essas transições, o Athena trabalha ativamente para adquirir ou reduzir o número de DPUs com base na solicitação. Quando a DPU ativa torna-se igual à DPU de destino, o número de destino foi atingido e não há alterações pendentes.

Para recuperar esses valores de forma programática, você pode chamar a ação da API GetCapacityReservation. A API refere-se à DPU ativa e à DPU de destino como `AllocatedDpus` e `TargetDpus`.

Tópicos

- Editar reservas de capacidade
- Adicionar grupos de trabalho a uma reserva
- Remover um grupo de trabalho de uma reserva
- Cancelar uma reserva de capacidade
- Excluir uma reserva de capacidade

Editar reservas de capacidade

Após criar uma reserva de capacidade, você pode ajustar o número de DPUs e adicionar ou remover as etiquetas personalizadas.

Para editar uma reserva de capacidade

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.
3. Escolha Administração, Reservas de capacidade.
4. Na lista de reservas de capacidade, siga um destes procedimentos:
   - Selecione o botão ao lado da reserva e escolha Editar.
   - Escolha o link da reserva e depois Editar.
5. Em DPU, escolha ou insira o número de unidades de processamento de dados desejada em incrementos de 4. O número mínimo de DPUs que você pode ter é 24. Para ter mais informações, consulte Noções básicas sobre DPUs.

> **Note**
>
> Você pode adicionar DPUs a uma reserva de capacidade existente a qualquer momento. Porém, você não pode diminuir o número de DPUs até 1 hora depois de criar a reserva ou adicionar DPUs a ela.

6. Em Etiquetas, selecione Remover para remover uma etiqueta ou escolha Adicionar etiqueta para adicionar uma nova etiqueta.
7. Selecione Enviar. A página de detalhes da reserva mostra a configuração atualizada.

Adicionar grupos de trabalho a uma reserva

Depois de criar uma reserva de capacidade, é possível adicionar até 20 grupos de trabalho à reserva. Adicionar um grupo de trabalho a uma reserva informa ao Athena quais consultas deverão ser executadas na capacidade reservada. As consultas de grupos de trabalho que não estão associados a uma reserva continuam sendo executadas usando o modelo de preço padrão por terabyte (TB) verificado.

Quando uma reserva tem dois ou mais grupos de trabalho, as consultas desses grupos de trabalho podem usar a capacidade da reserva. É possível adicionar e remover grupos de trabalho a qualquer momento. Quando você adicionar ou remover grupos de trabalho, as consultas em execução não serão interrompidas.

Quando a reserva está pendente, as consultas dos grupos de trabalho que você adicionou continuam sendo executadas usando o modelo de preço padrão por terabyte (TB) verificado até que a reserva torne-se ativa.

Para adicionar um ou mais grupos de trabalho à reserva de capacidade

1. Na página de detalhes da reserva de capacidade, escolha Adicionar grupos de trabalho.
2. Na página Adicionar grupos de trabalho, selecione os grupos de trabalho que deseja adicionar e escolha Adicionar grupos de trabalho. Não é possível atribuir um grupo de trabalho a mais de uma reserva.

   A página de detalhes da reserva de capacidade lista os grupos de trabalho que você adicionou. As consultas executadas nesses grupos de trabalho usarão a capacidade que você reservou quando a reserva estiver ativa.

Remover um grupo de trabalho de uma reserva

Caso não precise mais de capacidade dedicada para um grupo de trabalho ou se quiser mover um grupo de trabalho para sua própria reserva, poderá removê-lo a qualquer momento. A remoção de um grupo de trabalho de uma reserva é um processo simples. Depois de remover um grupo de trabalho de uma reserva, as consultas do grupo de trabalho removido usam como padrão a capacidade sob demanda (não provisionada) e são cobradas com base nos terabytes (TB) verificados.

Para remover um ou mais grupos de trabalho de uma reserva

1. Na página de detalhes da reserva de capacidade, selecione os grupos de trabalho que deseja remover.
2. Escolha Remover grupos de trabalho. O prompt Remover grupos de trabalho? informa que todas as consultas ativas no momento serão concluídas antes que o grupo de trabalho seja removido da reserva.
3. Escolha Remover. A página de detalhes da reserva de capacidade mostra que os grupos de trabalho removidos não estão mais presentes.

Cancelar uma reserva de capacidade

Caso não queira mais usar uma reserva de capacidade, você pode cancelá-la. As consultas que ainda estiverem em execução nos grupos de trabalho que estavam usando a reserva poderão ser concluídas, mas outras consultas do grupo de trabalho não usarão mais a reserva.

> **Note**
>
> Não é garantido que a capacidade cancelada será reservada novamente em uma data futura. A capacidade não pode ser transferida para outra reserva, Conta da AWS ou Região da AWS.

Para cancelar uma reserva de capacidade

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.
3. Escolha Administração, Reservas de capacidade.

4. Na lista de reservas de capacidade, siga um destes procedimentos:

   - Selecione o botão ao lado da reserva e escolha Cancelar.
   - Escolha o link da reserva e escolha Cancelar reserva de capacidade.

5. No prompt Cancelar reserva de capacidade?, insira cancel e escolha Cancelar reserva de capacidade.

   O status da reserva é alterado para Cancelando e um banner de progresso informa que o cancelamento está em andamento.

   Quando o cancelamento for concluído, a reserva de capacidade permanecerá, mas o status será exibido como Cancelada. A reserva será excluída 45 dias após o cancelamento. Durante os 45 dias, não será possível reaproveitar ou reutilizar a reserva que foi cancelada, mas você poderá consultar as etiquetas e visualizá-la para referência histórica.

Excluir uma reserva de capacidade

Se você quiser remover todas as referências a uma reserva de capacidade cancelada, poderá excluir a reserva. Uma reserva deve ser cancelada para poder ser excluída. Uma reserva excluída é imediatamente removida da conta e não pode mais ser referenciada, nem por seu ARN.

Para excluir uma reserva de capacidade

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.
3. Escolha Administração, Reservas de capacidade.
4. Na lista de reservas de capacidade, siga um destes procedimentos:

   - Selecione o botão ao lado da reserva e escolha Ações, Excluir.
   - Escolha o link da reserva e depois Excluir.

5. No prompt Excluir reserva de capacidade?, escolha Excluir.

   Um banner informa que a reserva de capacidade foi excluída com sucesso. A reserva excluída não aparece mais na lista de reservas de capacidade.

Políticas do IAM para reservas de capacidade

Para controlar o acesso a reservas de capacidade, use permissões do IAM no nível do recurso ou políticas do IAM baseadas em identidade. Sempre que você usar as políticas do IAM, siga as práticas recomendadas do IAM. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM no Manual do usuário do IAM.

O procedimento a seguir é específico ao Athena.

Para obter informações específicas do IAM, acesse os links listados no fim desta seção. Para obter informações sobre exemplos de políticas de reservas de capacidade do JSON, consulte Exemplos de políticas de reserva de capacidade.

Para usar o editor visual no console do IAM para criar uma política de reserva de capacidade

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Policies (Políticas) e Create policy (Criar política).
3. Na guia Editor visual, selecione Escolher um serviço. Em seguida, escolha o Athena para adicionar à política.
4. Escolha Select actions (Selecionar ações) e defina as ações para adicionar à política. O editor visual mostra as ações disponíveis no Athena. Para obter mais informações, consulte Ações, recursos e chaves de condição do Amazon Athena na Referência de autorização do serviço.
5. Escolha adicionar ações para digitar uma ação específica ou use caracteres curinga (*) para especificar várias ações.

   Por padrão, a política que você está criando permite as ações que você escolhe. Se você escolher uma ou mais ações compatíveis com as permissões no nível do recurso `capacity-reservation` no Athena, o editor listará o recurso `capacity-reservation`.
6. Escolha Recursos para especificar as reservas de capacidade específicas de sua política. Para obter exemplos de políticas de reserva de capacidade do JSON, consulte Exemplos de políticas de reserva de capacidade.
7. Especifique o recurso `capacity-reservation` da seguinte forma:

```
arn:aws:athena:<region>:<user-account>:capacity-reservation/<capacity-reservation-name>
```

8. Selecione Review Policy (Revisar política), digite um Name (Nome) e uma Description (Descrição) (opcional) para a política que você está criando. Revise o resumo da política para ter certeza de que você concedeu as permissões que pretendia.
9. Escolha Criar política para salvar sua nova política.
10. Anexe essa política baseada em política a um usuário, um grupo ou uma função.

Para obter mais informações, consulte os seguintes tópicos na Referência de autorização do serviço e no Manual do usuário do IAM:

- Ações, recursos e chaves de condição do Amazon Athena
- Criar políticas com o editor visual
- Adicionar e remover políticas do IAM
- Controlar o acesso aos recursos

Para obter exemplos de políticas de reserva de capacidade do JSON, consulte Exemplos de políticas de reserva de capacidade.

Para obter uma lista completa de ações do Amazon Athena, consulte os nomes das ações de API na Referência de API do Amazon Athena.

Exemplos de políticas de reserva de capacidade

Esta seção inclui exemplos de políticas que você pode usar para habilitar várias ações nas reservas de capacidade. Sempre que você usar as políticas do IAM, siga as práticas recomendadas do IAM. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM no Manual do usuário do IAM.

A reserva de capacidade é um recurso do IAM gerenciado pelo Athena. Portanto, se sua política de reserva de capacidade utiliza ações que usam `capacity-reservation` como entrada, especifique o ARN da reserva de capacidade da seguinte maneira:

```
"Resource": [arn:aws:athena:<region>:<user-account>:capacity-reservation/<capacity-reservation-name>]
```

Em que *<capacity-reservation-name>* é o nome da reserva de capacidade. Por exemplo, para uma reserva de capacidade denominada `test_capacity_reservation`, especifique-a como recurso da seguinte forma:

```
"Resource": ["arn:aws:athena:us-east-1:123456789012:capacity-reservation/
test_capacity_reservation"]
```

Para obter uma lista completa de ações do Amazon Athena, consulte os nomes das ações de API na Referência de API do Amazon Athena. Para obter mais informações sobre as políticas do IAM, consulte Criar políticas com o editor visual no Guia do usuário do IAM.

- Example policy to list capacity reservations
- Example policy for management operations

Example Exemplo de política para listar reservas de capacidade

A política a seguir permite que todos os usuários listem todas as reservas de capacidade.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListCapacityReservations"
            ],
            "Resource": "*"
        }
    ]
}
```

Example Exemplo de política para operações de gerenciamento

A política a seguir permite que o usuário crie, cancele, obtenha detalhes e atualize a reserva de capacidade `test_capacity_reservation`. A política também permite que o usuário atribua `workgroupA` e `workgroupB` a `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"
        ]
    }
  ]
}
```

APIs de reserva de capacidade do Athena

A lista a seguir contém links de referência para as ações de API de reserva de capacidade do Athena. Para obter estruturas de dados e outras ações de API do Athena, consulte Amazon Athena API Reference (Referência para APIs do Amazon Athena).

- CancelCapacityReservation
- CreateCapacityReservation
- GetCapacityAssignmentConfiguration
- GetCapacityReservation
- ListCapacityReservations
- PutCapacityAssignmentConfiguration
- UpdateCapacityReservation

Ajuste de performance no Athena

Este tópico fornece informações gerais e sugestões específicas para melhorar a performance de suas consultas do Athena e como resolver erros relacionados a limites e ao uso de recursos.

Cotas de serviço

O Athena impõe cotas para métricas como tempo de execução da consulta, número de consultas simultâneas na conta e taxas de solicitação de API. Para obter mais informações sobre essas cotas,

consulte Service Quotas. Exceder essas cotas faz com que uma consulta falhe, seja quando ela é enviada, seja durante a execução da consulta.

Muitas das dicas de otimização de performance desta página podem ajudar a reduzir o tempo de execução das consultas. A otimização libera capacidade para que você possa executar mais consultas dentro da cota de simultaneidade e evita que as consultas sejam canceladas por estarem em execução por muito tempo.

As cotas sobre o número de consultas simultâneas e solicitações de API são por Conta da AWS e Região da AWS. É recomendável executar uma workload por Conta da AWS (ou usar reservas de capacidade provisionadas separadas) para evitar que as workloads concorram à mesma cota.

Se você executar duas workloads na mesma conta, uma das workloads poderá executar uma intermitência de consultas. Isso pode fazer com que a workload restante seja limitada ou impedida de executar consultas. Para evitar isso, é possível mover as workloads para contas separadas para dar a cada workload sua própria cota de simultaneidade. Criar uma reserva de capacidade provisionada para uma ou ambas as workloads atinge o mesmo objetivo.

Cotas em outros serviços

Ao executar uma consulta, o Athena pode chamar outros serviços que impõem cotas. Durante a execução da consulta, o Athena pode fazer chamadas de API para o AWS Glue Data Catalog, o Amazon S3 e outros serviços da AWS, como o IAM e o AWS KMS. Se você usar consultas federadas, o Athena também chamará o AWS Lambda. Todos esses serviços têm seus próprios limites e cotas que podem ser excedidos. Quando a execução de uma consulta encontra erros nesses serviços, ela falha e inclui o erro do serviço de origem. Os erros recuperáveis são repetidos, mas as consultas ainda poderão falhar se o problema não for resolvido a tempo. Leia atentamente as mensagens de erro para determinar se elas vêm do Athena ou de outro serviço. Alguns erros relevantes são abordados neste documento.

Para obter mais informações sobre como resolver erros causados por cotas de serviço do Amazon S3, consulte Evite ter uma quantidade muito grande de arquivos mais adiante neste documento. Para obter mais informações sobre a otimização de performance do Amazon S3, consulte Padrões de design de práticas recomendadas: otimizar a performance do Amazon S3 no Guia do usuário do Amazon S3.

Limites de recurso

O Athena executa consultas em um mecanismo de consulta distribuído. Quando você envia uma consulta, o planejador de consultas do mecanismo do Athena estima a capacidade computacional

necessária para executar a consulta e prepara devidamente um cluster de nós de computação. Algumas consultas, como consultas DDL, são executadas em apenas um nó. Consultas complexas baseadas em grandes conjuntos de dados são executadas em clusters muito maiores. Os nós são uniformes e têm as mesmas configurações de memória, CPU e disco. O Athena aumenta a escala horizontalmente, não verticalmente, para processar consultas mais exigentes.

Às vezes, as demandas de uma consulta excedem os recursos disponíveis para o cluster que está executando a consulta. Quando isso acontece, a consulta falha com o erro Query exhausted resources at this scale factor.

O recurso que se esgota com mais frequência é a memória, mas em casos raros também pode ser o espaço em disco. Os erros de memória geralmente ocorrem quando o mecanismo executa uma função de junção ou janela, mas também podem ocorrer em contagens e agregações distintas.

Mesmo que a consulta falhe com um erro de “falta de recursos” uma vez, ela pode ter êxito ao ser executada novamente. A execução da consulta não é determinística. Fatores como o tempo necessário para carregar os dados e como os conjuntos de dados intermediários serão distribuídos pelos nós podem resultar em diferentes usos de recursos. Por exemplo, imagine uma consulta que une duas tabelas e tem uma grande distorção na distribuição dos valores da condição de junção. A consulta pode ter êxito na maioria das vezes, mas ocasionalmente falha quando os valores mais comuns acabam sendo processados pelo mesmo nó.

Para evitar que suas consultas excedam os recursos disponíveis, use as dicas de ajuste de performance mencionadas neste documento. Em particular, para obter dicas sobre como otimizar consultas que esgotam os recursos disponíveis, consulte Otimizar junções, Otimizar funções de janela e Otimizar consultas usando aproximações.

Técnicas de otimização de consulta

Use as técnicas de otimização de consulta descritas nesta seção para fazer com que as consultas sejam executadas mais rapidamente ou como soluções alternativas para consultas que excedam os limites de recursos no Athena.

Otimizar junções

Há muitas estratégias diferentes para executar junções em um mecanismo de consulta distribuído. Duas das mais comuns são as junções hash distribuídas e as consultas com condições de junção complexas.

Junção hash distribuída

O tipo mais comum de junção usa uma comparação de igualdade como condição de junção. O Athena executa esse tipo de junção como uma junção hash distribuída.

Em uma junção hash distribuída, o mecanismo cria uma tabela de pesquisa (tabela hash) com base em um dos lados da junção. Esse lado é chamado de lado da compilação. Os registros do lado da compilação são distribuídos entre os nós. Cada nó cria uma tabela de pesquisa para o respectivo subconjunto. O outro lado da junção, denominado lado de teste, é então transmitido pelos nós. Os registros do lado de teste são distribuídos pelos nós da mesma forma que no lado da compilação. Isso permite que cada nó realize a junção pesquisando os registros correspondentes em sua própria tabela de pesquisa.

Quando as tabelas de pesquisa criadas do lado da compilação da junção não cabem na memória, as consultas podem falhar. Mesmo que o tamanho total do lado da compilação seja menor que a memória disponível, as consultas poderão falhar se a distribuição dos registros tiver uma distorção significativa. Em um caso extremo, todos os registros poderiam ter o mesmo valor para a condição de junção e caber na memória em um único nó. Mesmo uma consulta com menos distorção poderá falhar se um conjunto de valores for enviado ao mesmo nó e a soma dos valores ultrapassar a memória disponível. Os nós têm a capacidade de vazar registros para o disco, mas o vazamento retarda a execução da consulta e pode ser insuficiente para evitar que a consulta falhe.

O Athena tenta reordenar as junções para usar a relação maior como lado de teste e a relação menor como o lado da compilação. No entanto, por não gerenciar os dados em tabelas, o Athena tem informações limitadas e muitas vezes deve presumir que a primeira tabela é a maior e a segunda tabela é a menor.

Ao gravar junções com condições de junção baseadas em igualdade, suponha que a tabela à esquerda da palavra-chave `JOIN` seja o lado de teste e que a tabela à direita seja o lado da compilação. Verifique se a tabela direita, o lado da compilação, é a menor das tabelas. Se não for possível diminuir o lado da compilação da junção o suficiente para caber na memória, considere executar várias consultas que unam subconjuntos da tabela de compilação.

Outros tipos de junção

Consultas com condições de junção complexas (por exemplo, consultas que usam `LIKE`, > ou outros operadores) geralmente demandam muito computacionalmente. Na pior das hipóteses, cada registro de um lado da junção deve ser comparado a todos os registros do outro lado da junção. Como o tempo de execução aumenta com o quadrado do número de registros, as consultas correm o risco de exceder o tempo máximo de execução.

Para descobrir como o Athena executará sua consulta com antecedência, você pode usar a instrução `EXPLAIN`. Para obter mais informações, consulte Usar EXPLAIN e EXPLAIN ANALYZE no Athena e Noções básicas sobre os resultados da instrução EXPLAIN do Athena.

Otimizar funções de janela

Por serem operações que fazem uso intensivo de recursos, as funções de janela podem fazer com que as consultas sejam executadas lentamente ou até mesmo falhem com a mensagem Query exhausted resources at this scale factor. As funções de janela mantêm todos os registros em que operam na memória para calcular o resultado. Quando a janela é muito grande, a função de janela pode ficar sem memória.

Para garantir que suas consultas sejam executadas dentro dos limites de memória disponíveis, reduza o tamanho das janelas em que suas funções de janela operam. Para fazer isso, adicione uma cláusula `PARTITIONED BY` ou restrinja o escopo das cláusulas de particionamento existentes.

Em vez disso, use funções que não sejam de janela

Às vezes, consultas com funções de janela podem ser gravadas sem funções de janela. Por exemplo, em vez de usar `row_number` para encontrar os principais registros `N`, você pode usar `ORDER BY` e `LIMIT`. Em vez de usar `row_number` ou `rank` para desduplicar registros, você pode usar funções agregadas como max_by, min_by e arbitrary.

Por exemplo, digamos que você tenha um conjunto de dados com atualizações de um sensor. O sensor informa periodicamente o status da bateria e inclui alguns metadados, como localização. Para saber o último status da bateria de cada sensor e sua localização, você pode usar esta consulta:

```
SELECT sensor_id,
       arbitrary(location) AS location,
       max_by(battery_status, updated_at) AS battery_status
FROM sensor_readings
GROUP BY sensor_id
```

Os metadados, como a localização, são os mesmos para cada registro, então você pode usar a função `arbitrary` para escolher qualquer valor do grupo.

Para obter o último status da bateria, você pode usar a função `max_by`. A função `max_by` escolhe o valor de uma coluna do registro em que o valor máximo de outra coluna foi encontrado. Nesse caso, ela retorna o status da bateria do registro com a hora da última atualização dentro do grupo. Essa

consulta é executada mais rapidamente e usa menos memória do que uma consulta equivalente com função de janela.

Otimizar agregações

Ao realizar uma agregação, o Athena distribui os registros entre os nós de processamento usando as colunas na cláusula `GROUP BY`. Para tornar a tarefa de combinar registros com grupos o mais eficiente possível, os nós tentam manter os registros na memória, mas os vazam para o disco, se necessário.

Também é uma boa ideia evitar incluir colunas redundantes nas cláusulas `GROUP BY`. Como menos colunas exigem menos memória, uma consulta que usa menos colunas para descrever um grupo é mais eficiente. As colunas numéricas também usam menos memória do que as strings. Por exemplo, ao agregar um conjunto de dados que tenha um ID numérico de categoria e um nome de categoria, use somente a coluna ID da categoria na cláusula `GROUP BY`.

Às vezes, as consultas incluem colunas na cláusula `GROUP BY` para contornar o fato de que uma coluna deve ser parte da cláusula `GROUP BY` ou ser uma expressão agregada. Se essa regra não for seguida, você poderá receber uma mensagem de erro como esta:

EXPRESSION_NOT_AGGREGATE: line 1:8: 'category' must be an aggregate expression or appear in GROUP BY clause

Para evitar a necessidade de adicionar colunas redundantes à cláusula `GROUP BY`, use a função arbitrary, como no exemplo a seguir.

```
SELECT country_id,
       arbitrary(country_name) AS country_name,
       COUNT(*) AS city_count
FROM world_cities
GROUP BY country_id
```

A função `ARBITRARY` retorna um valor de arbitrary do grupo. A função é útil quando você sabe que todos os registros do grupo têm o mesmo valor para uma coluna, mas o valor não identifica o grupo.

Otimizar as principais consultas N

A cláusula `ORDER BY` retorna os resultados de uma consulta em ordem de classificação. O Athena usa a classificação distribuída para executar a operação de classificação paralelamente em vários nós.

Se você não precisar estritamente que seu resultado seja classificado, evite adicionar uma cláusula `ORDER BY`. Além disso, evite adicionar `ORDER BY` a consultas internas se não for estritamente necessário. Em muitos casos, o planejador de consultas poderá remover a classificação redundante, mas isso não é garantido. Uma exceção a essa regra é se uma consulta interna estiver realizando uma operação principal `N`, como encontrar o `N` mais recente ou os valores de `N` mais comuns.

Quando o Athena vê `ORDER BY` junto com `LIMIT`, ele entende que você está executando uma consulta principal `N` e usa as operações dedicadas adequadamente.

> **Note**
>
> Embora o Athena muitas vezes também possa detectar funções de janela como `row_number` que usa `N` principal, recomendamos a versão mais simples que usa `ORDER BY` e `LIMIT`. Para obter mais informações, consulte Otimizar funções de janela.

Incluir somente as colunas obrigatórias

Caso não precise estritamente de uma coluna, não a inclua na consulta. Quanto menos dados a consulta precisar processar, mais rápido ela será executada. Isso reduz a quantidade de memória necessária e a quantidade de dados que devem ser enviados entre os nós. Se você estiver usando um formato de arquivo colunar, reduzir o número de colunas também reduzirá a quantidade de dados lidos do Amazon S3.

O Athena não tem limite específico para o número de colunas de um resultado, mas a forma como as consultas são executadas limita o possível tamanho combinado das colunas. O tamanho combinado das colunas inclui os nomes e tipos.

Por exemplo, o seguinte erro é causado por uma relação que excede o limite de tamanho de um descritor de relação:

GENERIC_INTERNAL_ERROR: io.airlift.bytecode.CompilationException

Para resolver esse problema, reduza o número de colunas na consulta ou crie subconsultas e use `JOIN` para recuperar uma quantidade menor de dados. Se você tiver consultas que executam `SELECT *` na consulta mais externa, altere `*` para uma lista com somente as colunas necessárias.

Otimizar consultas usando aproximações

O Athena tem suporte para funções agregadas de aproximação para contar valores distintos, valores mais frequentes, percentis (incluindo medianas aproximadas) e criar histogramas. Use essas funções sempre que não for necessário obter valores exatos.

Diferentemente das operações `COUNT(DISTINCT col)`, o approx_distinct usa muito menos memória e é executado mais rápido. Da mesma forma, usar numeric_histogram em vez de histogram utiliza métodos aproximados e, portanto, menos memória.

Otimizar LIKE

É possível usar `LIKE` para encontrar strings correspondentes, mas com strings longas, demanda uso intensivo de computação. Na maioria dos casos, a função regexp_like é uma alternativa mais rápida e também oferece mais flexibilidade.

Muitas vezes, é possível otimizar uma pesquisa ancorando a substring que você está procurando. Por exemplo, se você estiver procurando por um prefixo, é muito melhor usar "*`substr`*%" em vez de "*`substr`*%". Ou, se você estiver usando `regexp_like`, "^*`substr`*".

Usar UNION ALL em vez de UNION

`UNION ALL` e `UNION` são duas maneiras de combinar os resultados de duas consultas em um único resultado. `UNION ALL` concatena os registros da primeira consulta com a segunda e `UNION` faz o mesmo, mas também remove as duplicatas. `UNION` precisa processar todos os registros e encontrar as duplicatas, o que requer uso intensivo de memória e computação, mas `UNION ALL` é uma operação relativamente rápida. A menos que você precise desduplicar registros, use `UNION ALL` para obter a melhor performance.

Usar UNLOAD para grandes conjuntos de resultados

Quando se espera que os resultados da consulta sejam grandes (por exemplo, dezenas de milhares de linhas ou mais), use UNLOAD para exportar os resultados. Na maioria dos casos, isso é mais rápido do que executar uma consulta normal, e usar `UNLOAD` também oferece mais controle sobre a saída.

Quando a consulta acaba de ser executada, o Athena armazena o resultado como um único arquivo CSV não compactado no Amazon S3. Isso leva mais tempo que usar `UNLOAD`, não apenas porque o resultado não está compactado, mas também porque a operação não pode ser paralelizada. Por sua vez, `UNLOAD` grava os resultados diretamente dos nós de trabalho e faz uso total do paralelismo

do cluster de computação. Além disso, é possível configurar UNLOAD para gravar os resultados em formato compactado e em outros formatos de arquivo, como JSON e Parquet.

Para obter mais informações, consulte UNLOAD.

Use CTAS ou o Glue ETL para materializar agregações usadas com frequência

“Materializar” uma consulta é uma forma de acelerar a performance da consulta armazenando resultados de consultas complexas pré-computados (por exemplo, agregações e junções) para reutilização em consultas subsequentes.

Se muitas de suas consultas incluírem as mesmas junções e agregações, você poderá materializar a subconsulta comum como uma nova tabela e executar consultas nessa tabela. É possível criar a nova tabela com Criar uma tabela a partir de resultados de consultas (CTAS) ou com uma ferramenta ETL dedicada, como o Glue ETL.

Por exemplo, digamos que você tenha um painel com widgets que exibem diferentes aspectos de um conjunto de dados de pedidos. Cada widget tem sua própria consulta, mas todas as consultas compartilham as mesmas junções e filtros. Uma tabela de pedidos é unida a uma tabela de itens de linha, e há um filtro para exibir somente os últimos três meses. Se você identificar os atributos comuns dessas consultas, poderá criar uma nova tabela que os widgets possam usar. Isso reduz a duplicação e melhora a performance. A desvantagem é que você deverá manter a nova tabela atualizada.

Reutilizar resultados da consulta

É comum que a mesma consulta seja executada várias vezes em pouco tempo. Por exemplo, isso pode ocorrer quando várias pessoas abrem o mesmo painel de dados. Ao executar uma consulta, você pode pedir para o Athena reutilizar os resultados calculados anteriormente. Especifique a idade máxima dos resultados a serem reutilizados. Se a mesma consulta for executada anteriormente dentro desse período, o Athena retornará esses resultados em vez de executar a consulta novamente. Para obter mais informações, consulte Reutilização dos resultados da consulta aqui no Guia do usuário do Amazon Athena e Reduce cost and improve query performance with Amazon Athena Query Result Reuse no blog de big data da AWS.

Técnicas de otimização de dados

A performance não depende apenas das consultas, mas também, principalmente, de como seu conjunto de dados está organizado e do formato de arquivo e da compactação que ele usa.

Particionar dados

O particionamento divide a tabela em partes e mantém os dados relacionados juntos com base em propriedades como data, país ou região. As chaves de partição atuam como colunas virtuais. Você define as chaves de partição na criação da tabela e as utiliza para filtrar consultas. Quando você filtra as colunas das chaves de partição, somente os dados das partições correspondentes são lidos. Por exemplo, se o conjunto de dados estiver particionado por data e a consulta tiver um filtro que corresponda somente à última semana, somente os dados da última semana serão lidos. Para obter mais informações sobre particionamento, consulte Particionar dados no Athena.

Escolher chaves de partição que oferecerão suporte às suas consultas

Como o particionamento tem um impacto significativo na performance da consulta, pense com atenção em como você particionará ao criar seu conjunto de dados e tabelas. Ter muitas chaves de partição pode resultar em conjuntos de dados fragmentados com excesso de arquivos e arquivos muito pequenos. Por outro lado, ter poucas chaves de partição ou não ter particionamento leva a consultas que examinam mais dados do que o necessário.

Evitar otimização de consultas raras

Uma boa estratégia é otimizar as consultas mais comuns e evitar a otimização de consultas raras. Por exemplo, se as consultas analisarem períodos de dias, não particione por hora, mesmo que algumas consultas sejam filtradas até esse nível. Se seus dados tiverem uma coluna de timestamp granular, as consultas raras que filtram por hora poderão usar a coluna de timestamp. Mesmo que casos raros verifiquem um pouco mais de dados do que o necessário, reduzir a performance geral em prol de casos raros geralmente não é vantajoso.

Para reduzir a quantidade de dados que as consultas precisam verificar e, assim, melhorar a performance, use um formato de arquivo colunar e mantenha os registros ordenados. Em vez de particionar por hora, mantenha os registros classificados por carimbo de data e hora. Para consultas em janelas de tempo mais curtas, a classificação por carimbo de data e hora é quase tão eficiente quanto a partição por hora. Além disso, a classificação por carimbo de data e hora normalmente não prejudica a performance das consultas nas janelas de tempo contadas em dias. Para obter mais informações, consulte Usar formatos de arquivo colunares.

As consultas em tabelas com dezenas de milhares de partições terão melhor performance se houver predicados em todas as chaves de partição. Esse é outro motivo para criar um esquema de particionamento para as consultas mais comuns. Para obter mais informações, consulte Consultar partições por igualdade.

Usar projeção de partições

A projeção de partição é um atributo do Athena que armazena informações de partição não no AWS Glue Data Catalog, mas como regras nas propriedades da tabela no AWS Glue. Ao planejar uma consulta em uma tabela configurada com projeção de partição, o Athena lê as regras de projeção de partição da tabela. O Athena calcula as partições a serem lidas na memória com base na consulta e nas regras, em vez de procurar partições no AWS Glue Data Catalog.

Além de simplificar o gerenciamento de partições, a projeção de partições pode melhorar a performance de conjuntos de dados que têm um grande número de partições. Quando a consulta contém intervalos em vez de valores específicos para chaves de partição, a busca por partições correspondentes no catálogo demora mais quanto mais partições houver. Com a projeção de partição, o filtro pode ser computado na memória sem acessar o catálogo e pode ser muito mais rápido.

Em determinadas circunstâncias, a projeção de partições pode resultar em pior performance. Um exemplo ocorre quando uma tabela é “avulsa”. Uma tabela avulsa não tem dados para cada permutação dos valores da chave de partição descritos pela configuração da projeção da partição. Com uma tabela avulsa, o conjunto de partições calculado com base na consulta e a configuração da projeção da partição são listados no Amazon S3, mesmo quando não contêm dados.

Ao usar a projeção de partição, verifique se incluiu predicados em todas as chaves de partição. Limite o escopo dos valores possíveis para evitar listas desnecessárias do Amazon S3. Imagine uma chave de partição que contenha um intervalo de um milhão de valores e uma consulta que não tenha nenhum filtro nessa chave de partição. Para executar a consulta, o Athena deverá realizar pelo menos um milhão de operações de lista do Amazon S3. As consultas são mais rápidas quando você consulta valores específicos, independentemente de usar a projeção de partição ou armazenar informações de partição no catálogo. Para obter mais informações, consulte Consultar partições por igualdade.

Ao configurar uma tabela para projeção de partições, verifique se os intervalos especificados são razoáveis. Se a consulta não incluir um predicado em uma chave de partição, todos os valores no intervalo dessa chave serão usados. Se o conjunto de dados foi criado em uma data específica, use essa data como ponto de partida para qualquer intervalo de datas. Use `NOW` como fim de intervalos de datas. Evite intervalos numéricos que tenham um grande número de valores e considere usar o tipo injected em vez disso.

Para obter mais informações sobre projeção de partições, consulte Projeção de partições com o Amazon Athena.

Usar índices de partição

Os índices de partição são um atributo do AWS Glue Data Catalog que melhoram a performance da pesquisa de partições para tabelas que têm um grande número de partições.

A lista de partições do catálogo é como uma tabela em um banco de dados relacional. A tabela contém colunas para as chaves de partição e uma coluna adicional para o local da partição. Quando você consulta uma tabela particionada, os locais das partições são pesquisados por meio da verificação dessa tabela.

Assim como nos bancos de dados relacionais, é possível aumentar a performance das consultas adicionando índices. É possível adicionar vários índices para oferecer suporte a diferentes padrões de consulta. O índice de partição do AWS Glue Data Catalog é compatível com operadores de igualdade e comparação, como >, >= e < combinados com o operador `AND`. Para obter mais informações, consulte Working with partition indexes in AWS Glue no Guia do desenvolvedor do AWS Glue e Improve Amazon Athena query performance using AWS Glue Data Catalog partition indexes no blog de big data da AWS.

Use sempre STRING como o tipo para chaves de partição

Ao consultar chaves de partição, lembre-se de que o Athena exige que as chaves de partição sejam do tipo `STRING` para inserir a filtragem de partição no AWS Glue. Se o número de partições não for pequeno, usar outros tipos poderá reduzir a performance. Se os valores da chave de partição forem semelhantes a uma data ou a um número, converta-os ao tipo apropriado em sua consulta.

Remover partições antigas e vazias

Se você remover dados de uma partição no Amazon S3 (por exemplo, usando o ciclo de vida do Amazon S3), também deverá remover a entrada da partição do AWS Glue Data Catalog. Durante o planejamento da consulta, toda partição correspondente à consulta é listada no Amazon S3. Se houver muitas partições vazias, a sobrecarga de listar essas partições poderá ser prejudicial.

Além disso, se houver milhares de partições, considere remover os metadados da partição de dados antigos que não são mais relevantes. Por exemplo, se as consultas nunca buscarem dados com mais de um ano, você poderá remover periodicamente os metadados das partições mais antigas. Se o número de partições aumentar para dezenas de milhares, remover partições não utilizadas pode acelerar as consultas que não incluem predicados em todas as chaves de partição. Para obter informações sobre como incluir predicados em todas as chaves de partição de suas consultas, consulte Consultar partições por igualdade.

Consultar partições por igualdade

As consultas que contêm predicados de igualdade em todas as chaves de partição são executadas mais rapidamente porque os metadados da partição podem ser carregados diretamente. Evite consultas nas quais uma ou mais chaves de partição não tenham um predicado, ou o predicado seleciona uma faixa de valores. Para essas consultas, a lista de todas as partições deverá ser filtrada para encontrar valores correspondentes. Para a maioria das tabelas, a sobrecarga é mínima, mas para tabelas com dezenas de milhares de partições ou mais, a sobrecarga pode ser considerável.

Se não for possível gravar novamente suas consultas para filtrar partições por igualdade, você pode tentar a projeção de partições. Para obter mais informações, consulte Usar projeção de partições.

Evitar usar MSCK REPAIR TABLE para manutenção de partições

Como `MSCK REPAIR TABLE` pode levar muito tempo para ser executado, apenas adiciona novas partições e não remove partições antigas, não é uma forma eficiente de gerenciar partições (consulte Considerações e limitações).

É melhor gerenciar partições manualmente usando as APIs do AWS Glue Data Catalog, ALTER TABLE ADD PARTITION ou crawlers do AWS Glue. Se preferir, você pode usar a projeção de partições, que elimina totalmente a necessidade de gerenciar as partições. Para obter mais informações, consulte Projeção de partições com o Amazon Athena.

Verifique se suas consultas são compatíveis com o esquema de particionamento

É possível verificar com antecedência quais partições uma consulta examinará usando a instrução `EXPLAIN`. Prefixe sua consulta com a palavra-chave `EXPLAIN` e procure o fragmento de origem (por exemplo `Fragment 2 [SOURCE]`) para cada tabela na parte inferior da saída de `EXPLAIN`. Procure atribuições em que o lado direito esteja definido como chave de partição. A linha abaixo contém uma lista de todos os valores dessa chave de partição que serão verificados quando a consulta for executada.

Por exemplo, digamos que você tenha uma consulta em uma tabela com uma chave de partição `dt` e prefixe a consulta com `EXPLAIN`. Se os valores da consulta forem datas e um filtro selecionar um intervalo de três dias, a saída de `EXPLAIN` poderá ser mais ou menos assim:

```
dt := dt:string:PARTITION_KEY
    :: [[2023-06-11], [2023-06-12], [2023-06-13]]
```

A saída de `EXPLAIN` mostra que o planejador encontrou três valores para essa chave de partição que corresponderam à consulta. Ela também mostra quais são esses valores. Para obter mais

informações sobre o uso de `EXPLAIN`, consulte Usar EXPLAIN e EXPLAIN ANALYZE no Athena e Noções básicas sobre os resultados da instrução EXPLAIN do Athena.

Usar formatos de arquivo colunares

Formatos de arquivo colunar, como Parquet e ORC, foram criados para workloads de análise distribuídas. Eles organizam os dados por coluna, não por linha. Organizar os dados em formato colunar oferece as seguintes vantagens:

- Apenas as colunas necessárias para a consulta são carregadas
- Reduz-se a quantidade geral de dados que precisam ser carregados
- Os valores das colunas são armazenados juntos, de modo que os dados possam ser compactados com eficiência
- Os arquivos podem conter metadados que permitam que o mecanismo ignore o carregamento de dados desnecessários

Como exemplo de como podemos usar os metadados do arquivo, os metadados do arquivo podem conter informações sobre os valores mínimo e máximo em uma página de dados. Se os valores consultados não estiverem no intervalo indicado nos metadados, a página poderá ser ignorada.

Uma forma de usar esses metadados para melhorar a performance é garantir que os dados dos arquivos sejam classificados. Por exemplo, digamos que você tenha consultas que buscam registros em que a entrada `created_at` esteja em um curto espaço de tempo. Se seus dados forem classificados pela coluna `created_at`, o Athena poderá usar os valores mínimo e máximo dos metadados do arquivo para ignorar as partes desnecessárias dos arquivos de dados.

Ao usar formatos de arquivo colunar, verifique se seus arquivos não são pequenos demais. Conforme observado em Evite ter uma quantidade muito grande de arquivos, conjuntos de dados com muitos arquivos pequenos causam problemas de performance. Isso ocorre especialmente com formatos de arquivo colunar. Em arquivos pequenos, a sobrecarga do formato de arquivo colunar supera os benefícios.

O Parquet e o ORC são organizados internamente por grupos de linhas (Parquet) e faixas (ORC). O tamanho padrão para grupos de linhas é 128 MB, e para faixas, 64 MB. Se houver muitas colunas, você poderá aumentar o grupo de linhas e o tamanho da faixa para melhorar a performance. Não é recomendável diminuir o tamanho do grupo de linhas ou da faixa para um valor inferior ao padrão.

Para converter outros formatos de dados em Parquet ou ORC, você pode usar o AWS Glue ETL ou o Athena. Para obter mais informações sobre como usar o Athena para ETL, consulte Usar CTAS e INSERT INTO para ETL e análise de dados.

Compactar dados

O Athena é compatível com uma ampla variedade de formatos de compactação. Consultar dados compactados é mais rápido e mais barato, pois você paga pelo número de bytes verificados antes da descompactação.

O formato gzip fornece boas taxas de compressão e oferece amplo suporte a outras ferramentas e serviços. O formato zstd (Zstandard) é um formato de compactação mais recente com um bom equilíbrio entre performance e taxa de compactação.

Ao compactar arquivos de texto, como dados JSON e CSV, tente chegar a um equilíbrio entre o número de arquivos e o tamanho dos arquivos. A maioria dos formatos de compactação exige que o leitor leia os arquivos desde o início. Isso significa que, em geral, os arquivos de texto compactados não podem ser processados em paralelo. Arquivos grandes não compactados muitas vezes são divididos entre operadores para obter maior paralelismo durante o processamento da consulta, mas isso não é possível com a maioria dos formatos de compactação.

Conforme discutido em Evite ter uma quantidade muito grande de arquivos, é melhor não ter uma quantidade muito grande nem muito pequena de arquivos. Como o número de arquivos é o limite de quantos operadores podem processar a consulta, essa regra se aplica principalmente a arquivos compactados.

Para obter mais informações sobre uso de compactação no Athena, consulte Suporte a compactação no Athena.

Usar bucketing para pesquisas em chaves com alta cardinalidade

O bucketing é uma técnica para distribuir registros em arquivos separados com base no valor de uma das colunas. Isso garante que todos os registros com o mesmo valor estejam no mesmo arquivo. O bucketing é útil quando você tem uma chave com alta cardinalidade e muitas de suas consultas buscam valores específicos da chave.

Por exemplo, digamos que você consulte um conjunto de registros de um usuário específico. Se os dados forem agrupados em bucket por ID de usuário, o Athena saberá com antecedência quais arquivos contêm registros para um ID específico e quais não. Isso permite que o Athena leia somente os arquivos que possam conter o ID, reduzindo consideravelmente a quantidade de dados

lidos. Também reduz o tempo de computação que, de outra forma, seria necessário para pesquisar os dados em busca do ID específico.

Desvantagens do bucketing

O bucketing é menos vantajoso quando as consultas frequentemente pesquisam vários valores na coluna pela qual os dados são agrupados em bucket. Quanto mais valores forem consultados, maior será a probabilidade de que seja necessário ler todos ou a maioria dos arquivos. Por exemplo, se você tiver três buckets e uma consulta procurar três valores diferentes, talvez seja necessário ler todos os arquivos. O bucketing funciona melhor quando as consultas pesquisam valores únicos.

Para obter mais informações, consulte Particionamento e bucketing no Athena.

Evite ter uma quantidade muito grande de arquivos

Conjuntos de dados que consistem em muitos arquivos pequenos resultam em baixo desempenho geral da consulta. Ao planejar uma consulta, o Athena lista todos os locais das partições, o que leva tempo. O tratamento e a solicitação de cada arquivo também geram uma sobrecarga computacional. Portanto, é mais rápido carregar um único arquivo maior do Amazon S3 do que carregar os mesmos registros de muitos arquivos menores.

Em casos extremos, é possível encontrar limites de serviço do Amazon S3. O Amazon S3 é compatível com até 5.500 solicitações por segundo para uma única partição de índice. Inicialmente, o bucket é tratado como uma única partição de índice, mas à medida que as cargas de solicitações aumentam, ele pode ser dividido em várias partições de índice.

O Amazon S3 analisa os padrões de solicitação e as divisões com base nos prefixos de chave. Se seu conjunto de dados consistir em muitos milhares de arquivos, as solicitações provenientes do Athena poderão exceder a cota de solicitações. Mesmo com menos arquivos, a cota poderá ser excedida se forem feitas várias consultas simultâneas no mesmo conjunto de dados. Outras aplicações que acessam os mesmos arquivos podem contribuir para o número total de solicitações.

Quando o `limit` da taxa de solicitação é excedido, o Amazon S3 retorna o erro a seguir. Esse erro está incluído nas informações de status da consulta no Athena.

SlowDown: Please reduce your request rate

Para solucionar problemas, comece determinando se o erro foi causado por uma única consulta ou por várias consultas que leem os mesmos arquivos. No último caso, coordene a execução das consultas de modo que não sejam executadas ao mesmo tempo. Para obter isso, adicione um mecanismo de enfileiramento ou até mesmo novas tentativas em sua aplicação.

Se a execução de uma única consulta acionar o erro, tente combinar arquivos de dados ou modificar a consulta para ler menos arquivos. O melhor momento para combinar arquivos pequenos é antes de serem gravados. Para isso, considere as seguintes técnicas:

- Altere o processo que grava arquivos para gravar arquivos maiores. Por exemplo, é possível armazenar registros em buffer por mais tempo antes de serem gravados.
- Coloque arquivos em um local do Amazon S3 e use uma ferramenta como o Glue ETL para combiná-los em arquivos maiores. Em seguida, mova os arquivos maiores para o local ao qual a tabela aponta. Para obter mais informações, consulte Reading input files in larger groups no Guia do desenvolvedor do AWS Glue ou How can I configure an AWS Glue ETL job to output larger files? no Centro de Conhecimento AWS re:Post.
- Reduza o número de chaves de partição. Quando há muitas chaves de partição, cada partição pode ter poucos registros, resultando em um número excessivo de arquivos pequenos. Para obter informações sobre como decidir quais partições criar, consulte Escolher chaves de partição que oferecerão suporte às suas consultas.

Evitar hierarquias de armazenamento adicionais além da partição

Para evitar a sobrecarga do planejamento de consultas, armazene os arquivos em uma estrutura plana em cada local da partição. Não use hierarquias de diretório adicionais.

Ao planejar uma consulta, o Athena lista todos os arquivos em todas as partições correspondentes à consulta. Embora o Amazon S3 não tenha diretórios em si, a convenção é interpretar a barra direta `/` como separador de diretório. Ao listar os locais das partições, o Athena lista recursivamente qualquer diretório que encontrar. Quando os arquivos de uma partição são organizados em hierarquia, ocorrem várias rodadas de listagens.

Quando todos os arquivos estão diretamente no local da partição, na maioria das vezes, apenas uma operação de lista precisa ser executada. Porém, várias operações de lista sequencial serão necessárias se você tiver mais de mil arquivos em uma partição porque o Amazon S3 retorna somente mil objetos por operação de lista. Ter mais de mil arquivos em uma partição também pode criar outros problemas de performance mais graves. Para obter mais informações, consulte Evite ter uma quantidade muito grande de arquivos.

Use SymlinkTextInputFormat somente quando necessário

Usar a técnica SymlinkTextInputFormat pode ser uma forma de resolver situações em que os arquivos da tabela não estão bem organizados em partições. Por exemplo, links simbólicos podem

ser úteis quando todos os arquivos estão no mesmo prefixo ou quando há arquivos com esquemas diferentes no mesmo local.

Porém, usar links simbólicos adiciona níveis de indireção à execução da consulta. Esses níveis de indireção afetam a performance geral. Os arquivos de links simbólicos precisam ser lidos, e os locais que eles definem devem ser listados. Isso adiciona muitas viagens de ida e volta ao Amazon S3 que as tabelas habituais do Hive não exigem. Concluindo, você deverá usar `SymlinkTextInputFormat` somente quando não houver opções melhores disponíveis, como reorganizar arquivos.

Recursos adicionais do

Para obter mais informações sobre ajuste de performance no Athena, acesse estes recursos:

- Leia a publicação no blog sobre big data da AWS Top 10 Performance Tuning Tips for Amazon Athena (As 10 melhores dicas de ajuste de performance do Amazon Athena)
- Para ver um artigo sobre como usar o pushdown de predicado para melhorar o desempenho em consultas federadas, consulte Melhorar as consultas federadas com o pushdown de predicados no Amazon Athena no AWSBlog de Big Data.
- Para ver um artigo sobre as otimizações de desempenho no mecanismo de consulta do Athena, consulte Execute consultas 3 vezes mais rapidamente e com até 70% de economia de custos no mecanismo mais recente do Amazon Athena no blog AWS Big Data.
- Leia outras publicações do Athena no blog sobre big data da AWS
- Faça uma pergunta em AWS re:Post usando a etiqueta Amazon Athena
- Consulte os tópicos do Athena na Central de Conhecimento da AWS
- Entre em contato com AWS Support (no AWS Management Console, clique em Support (Suporte), Support Center (Central de Suporte))

Como prevenir o controle de utilização do Amazon S3

O controle de utilização é o processo de limitar sua taxa de uso de um serviço, uma aplicação ou um sistema. Na AWS, você pode usar o controle de utilização para evitar o uso excessivo do serviço Amazon S3 e aumentar a disponibilidade e a capacidade de resposta do Amazon S3 para todos os usuários. Porém, como o controle de utilização limita a taxa de transferências de dados de entrada e saída do Amazon S3, é importante tentar evitar que suas interações sejam limitadas.

Reduzir o controle de utilização no nível de serviço

Para evitar o controle de utilização do Amazon S3 no nível do serviço, é possível monitorar o uso e ajustar as cotas de serviço ou usar certas técnicas, como particionamento. Estas são algumas das condições que podem levar ao controle de utilização:

- Exceder os limites de solicitação de API da conta: o Amazon S3 tem limites de solicitação de API padrão baseados no tipo e no uso da conta. Se você exceder o número máximo de solicitações por segundo para um único objeto, as solicitações poderão ser limitadas para evitar a sobrecarga do serviço Amazon S3.
- Particionamento de dados insuficiente: se você não particionar os dados corretamente e transferir uma grande quantidade de dados, o Amazon S3 poderá limitar as solicitações. Para obter mais informações sobre particionamento, consulte a seção Usar particionamento deste documento.
- Grande quantidade de objetos pequenos: se possível, evite uma grande quantidade de arquivos pequenos. O Amazon S3 tem um limite de 5500 solicitações GET por segundo por prefixo particionado, e suas consultas do Athena compartilham esse mesmo limite. Se você verificar milhões de objetos pequenos em uma única consulta, provavelmente o Amazon S3 limitará a consulta.

Para evitar verificação em excesso, você pode usar o recurso ETL do AWS Glue para compactar periodicamente os arquivos ou particionar a tabela e adicionar filtros de chave de partição. Para obter mais informações, consulte os recursos a seguir.

- Como posso configurar um trabalho de ETL do AWS Glue para gerar arquivos maiores? (Central de conhecimento da AWS
- Ler arquivos de entrada em grupos maiores (Guia do desenvolvedor do AWS Glue)

Como otimizar as tabelas

Caso você encontre problemas de controle de utilização, é importante estruturar os dados. Embora o Amazon S3 consiga lidar com grandes quantidades de dados, às vezes o controle de utilização ocorre devido à forma como os dados são estruturados.

As seções a seguir oferecem algumas sugestões sobre como estruturar dados no Amazon S3 para evitar problemas de controle de utilização.

Usar particionamento

É possível usar o particionamento para reduzir o controle de utilização limitando a quantidade de dados que precisam ser acessados a qualquer momento. Ao particionar dados em colunas específicas, você pode distribuir as solicitações de maneira uniforme entre vários objetos e reduzir o número de solicitações para um único objeto. Reduzir a quantidade de dados que devem ser verificados melhora a performance da consulta e reduz os custos.

Ao criar uma tabela, você pode definir partições que atuam como colunas virtuais. Para criar uma tabela com partições em uma instrução `CREATE TABLE`, use a cláusula `PARTITIONED BY (`*`column_name data_type`*`)` para definir as chaves para particionar dados.

Para restringir as partições verificadas por uma consulta, é possível especificá-las como predicados em uma cláusula `WHERE` da consulta. Portanto, colunas usadas como filtros com frequência são boas candidatas para particionamento. Uma prática comum é particionar os dados com base na hora, o que pode acarretar um esquema de particionamento em vários níveis.

O particionamento também tem um custo. Quando você aumenta o número de partições na tabela, o tempo necessário para recuperar e processar os metadados da partição também aumenta. Assim, o particionamento excessivo pode remover os benefícios obtidos ao particionar de forma mais criteriosa. Se os dados estiverem muito distorcidos para um valor de partição e a maioria das consultas usar esse valor, você poderá incorrer em aumento de sobrecarga.

Para obter mais informações sobre particionamento no Athena, consulte O que é particionamento?.

Classificar dados em bucket

Outra forma de particionar dados é classificá-los em buckets dentro de uma única partição. Com o bucketing, você especifica uma ou mais colunas que contêm linhas que deseja agrupar. Em seguida, você coloca essas linhas em vários buckets. Dessa forma, você consulta somente o bucket que deve ser lido, reduzindo o número de linhas de dados que devem ser verificados.

Ao selecionar uma coluna que será usada para bucketing, selecione a coluna que tenha alta cardinalidade (ou seja, que tenha muitos valores distintos), esteja uniformemente distribuída e seja usada com frequência para filtrar os dados. Um exemplo de uma boa coluna a ser usada para bucketing é uma chave primária, como uma coluna de ID.

Para obter mais informações sobre bucketing no Athena, consulte O que é bucketing?

Usar índices de partição do AWS Glue

Você pode usar índices de partição do AWS Glue para organizar dados em uma tabela com base nos valores de uma ou mais partições. Os índices de partição do AWS Glue podem reduzir o número de transferências de dados, a quantidade de processamento de dados e o tempo de processamento das consultas.

Um índice de partição do AWS Glue é um arquivo de metadados que contém informações sobre as partições da tabela, inclusive as chaves da partição e seus valores. O índice da partição é armazenado em um bucket do Amazon S3 e atualizado automaticamente pelo AWS Glue à medida que novas partições são adicionadas à tabela.

Quando há um índice de partição do AWS Glue presente, as consultas tentam buscar um subconjunto das partições em vez de carregar todas as partições na tabela. As consultas são executadas somente no subconjunto de dados relevante para a consulta.

Ao criar uma tabela no AWS Glue, é possível criar um índice de partição em qualquer combinação de chaves de partição definidas na tabela. Depois de criar um ou mais índices de partição em uma tabela, é necessário adicionar uma propriedade à tabela que habilite a filtragem de partições. Em seguida, você pode consultar a tabela no Athena.

Para obter informações sobre como criar índices de partição no AWS Glue, consulte Trabalhar com índices de partição no AWS Glue no Guia do desenvolvedor do AWS Glue. Para obter informações sobre como adicionar uma propriedade de tabela para habilitar a filtragem de partições, consulte Indexação e filtragem de partições do AWS Glue.

Usar compactação de dados e divisão de arquivos

A compactação de dados pode acelerar consideravelmente as consultas se os arquivos estiverem no tamanho ideal ou se puderem ser divididos em grupos lógicos. Geralmente, taxas de compactação mais altas necessitam de mais ciclos de CPU para compactar e descompactar os dados. Para o Athena, recomendamos usar o Apache Parquet ou o Apache ORC, que compactam dados por padrão. Para obter mais informações sobre compactação de dados no Athena, consulte Suporte a compactação no Athena.

A divisão de arquivos aumenta o paralelismo ao permitir que o Athena distribua a tarefa de ler um único arquivo entre vários leitores. Se um único arquivo não foi divisível, somente um único leitor poderá ler o arquivo enquanto outros leitores estiverem ociosos. O Apache Parquet e o Apache ORC também são compatíveis com arquivos divisíveis.

Use armazenamentos de dados em colunas otimizados

A performance da consulta do Athena melhorará consideravelmente se você converter os dados para um formato em colunas. Ao gerar arquivos em colunas, uma técnica de otimização a ser considerada é ordenar os dados com base na chave de partição.

O Apache Parquet e o Apache ORC são armazenamentos de dados em colunas de código aberto bastante usados. Para obter informações sobre como converter a fonte de dados existente do Amazon S3 para um desses formatos, consulte Converter em formatos colunares.

Usar um tamanho de bloco do Parquet ou tamanho de faixa ORC maior

O Parquet e o ORC têm parâmetros de armazenamento de dados que podem ser ajustados para otimização. No Parquet, é possível otimizar o tamanho do bloco. No ORC, é possível otimizar o tamanho da faixa. Quanto maior o bloco ou a faixa, mais linhas você poderá armazenar em cada um deles. Por padrão, o tamanho do bloco Parquet é 128 MB, e o tamanho da faixa ORC é 64 MB.

Se uma faixa ORC for menor que 8 MB (o valor padrão de `hive.orc.max_buffer_size`), o Athena lerá toda a faixa ORC. É a compensação que o Athena faz entre a seletividade da coluna e as operações de entrada e saída por segundo para faixas menores.

Se houver tabelas com um número muito grande de colunas, um tamanho pequeno de bloco ou de faixa pode fazer com que sejam verificados mais dados do que o necessário. Nesses casos, um tamanho maior de bloco pode ser mais eficiente.

Usar ORC para tipos complexos

Atualmente, ao consultar colunas armazenadas em Parquet que tenham tipos de dados complexos (por exemplo, `array`, `map` ou `struct`), o Athena lê a linha de dados inteira em vez de somente as colunas especificadas seletivamente. Este é um problema conhecido do Athena. Como solução alternativa, considere usar o ORC.

Escolher um algoritmo de compactação

Outro parâmetro que pode ser configurado é o algoritmo de compactação nos blocos de dados. Para obter informações sobre os algoritmos de compactação compatíveis com Parquet e ORC no Athena, consulte Suporte a compactação no Athena.

Para obter mais informações sobre a otimização de formatos de armazenamento em colunas no Athena, consulte a seção “Otimizar a geração de armazenamento de dados em colunas” na publicação do AWS Big Data Blog As dez melhores dicas de ajuste de performance para o Amazon Athena.

Usar tabelas Iceberg

Apache Iceberg é um formato de tabela aberta para conjuntos de dados analíticos muito grandes, criado para otimizar o uso do Amazon S3. Você pode usar as tabelas Iceberg para ajudar a reduzir o controle de utilização no Amazon S3.

As tabelas Iceberg oferecem as seguintes vantagens:

- É possível dividir as tabelas Iceberg em partições de uma ou mais colunas. Isso otimiza o acesso aos dados e reduz a quantidade de dados que deverão ser verificados por meio de consultas.
- Como o modo de armazenamento de objetos do Iceberg otimiza as tabelas Iceberg para funcionar com o Amazon S3, ele pode processar grandes volumes de dados e workloads de consultas pesadas.
- As tabelas Iceberg no modo de armazenamento de objetos são escaláveis, tolerantes a falhas e duráveis, o que pode ajudar a reduzir o controle de utilização.
- Com o suporte à transação ACID, vários usuários podem adicionar e excluir objetos do Amazon S3 de forma atômica.

Para obter mais informações sobre o Apache Iceberg, consulte Apache Iceberg. Para obter mais informações sobre como usar tabelas Apache Iceberg no Athena, consulte Usar tabelas Iceberg.

Otimizar consultas

Use as sugestões desta seção para otimizar as consultas SQL no Athena.

Usar LIMIT com a cláusula ORDER BY

A cláusula `ORDER BY` retorna dados em uma ordem classificada. Isso exige que o Athena envie todas as linhas de dados para um único nó de processamento e classifique as linhas. Esse tipo de consulta pode ser executado por muito tempo ou pode até falhar.

Para aumentar a eficiência das consultas, observe os valores *N* superiores ou inferiores e use também uma cláusula `LIMIT`. Isso reduz significativamente o custo de classificação, inserindo tanto a classificação como a limitação em nós de processamento individuais, e não em um único operador.

Otimizar cláusulas JOIN

Quando você une duas tabelas, o Athena distribui a tabela à direita para os nós de processamento e transmite a tabela à esquerda para realizar a junção.

Por isso, especifique a tabela maior no lado esquerdo da junção e a tabela menor no lado direito da junção. Assim, o Athena usa menos memória e executa a consulta com menor latência.

Observe também os seguintes pontos:

- Ao usar vários comandos `JOIN`, especifique as tabelas da maior para a menor.
- Evite junções cruzadas, a menos que sejam exigidas pela consulta.

Otimizar cláusulas GROUP BY

O operador `GROUP BY` distribui as linhas com base nas colunas `GROUP BY` para os nós de processamento. Essas colunas são referenciadas na memória, e os valores são comparados à medida que as linhas são ingeridas. Os valores são agregados quando a coluna `GROUP BY` coincide. Considerando a forma como o processo funciona, é aconselhável ordenar as colunas da maior cardinalidade para a menor.

Usar números em vez de strings

Como os números exigem menos memória e são mais rápidos de processar em comparação com as strings, use números em vez de strings quando possível.

Limitar o número de colunas

Para reduzir a quantidade total de memória necessária para armazenar os dados, limite o número de colunas especificado na instrução `SELECT`.

Usar expressões regulares em vez de LIKE

Consultas que incluem cláusulas como `LIKE '%string%'` em strings grandes podem fazer uso muito intensivo de computação. Ao filtrar por vários valores em uma coluna de string, use a função regexp_like() e uma expressão regular. Isso é útil especialmente ao comparar uma longa lista de valores.

Usar a cláusula LIMIT

Em vez de selecionar todas as colunas ao executar uma consulta, use a cláusula `LIMIT` para retornar somente as colunas necessárias. Essa ação reduz o tamanho do conjunto de dados que é processado pelo pipeline de execução da consulta. As cláusulas `LIMIT` são mais úteis ao consultar tabelas que têm um grande número de colunas baseadas em strings. As cláusulas `LIMIT` também são úteis ao executar várias junções ou agregações em qualquer consulta.

Recursos adicionais do

Padrões de design de práticas recomendadas: otimizar a performance do Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

Ajuste de performance no Athena

Suporte a compactação no Athena

Tópicos

- Especificar os formatos de compactação
- Especificar a não compactação
- Observações e recursos
- Suporte à compressão de tabelas do Hive por formato de arquivo
- Suporte à compressão de tabelas do Iceberg por formato de arquivo
- Usar níveis de compressão ZSTD no Athena

O Athena suporta uma variedade de formatos de compactação para leitura e gravação de dados, incluindo a leitura de uma tabela que usa vários formatos de compactação. Por exemplo, o Athena pode ler com sucesso os dados de uma tabela que usa o formato de arquivo Parquet quando alguns arquivos Parquet são compactados com o Snappy e outros arquivos Parquet são compactados com o GZIP. O mesmo princípio se aplica aos formatos de armazenamento ORC, arquivo de texto e JSON.

O Athena aceita os seguintes formatos de compactação:

- BZIP2: formato que usa o algoritmo de Burrows-Wheeler.
- DEFLATE: algoritmo de compactação de dados baseado em codificação LZSS e Huffman. Deflate só é relevante para o formato de arquivo Avro.
- GZIP: algoritmo de compactação baseado em Deflate. Para tabelas do Hive no mecanismo do Athena versões 2 e 3, e tabelas do Iceberg no mecanismo do Athena versão 2, o GZIP é o formato de compressão de gravação padrão para arquivos nos formatos de armazenamento de arquivos Parquet e de texto. Arquivos no formato `tar.gz` não são suportados.
- LZ4: este membro da família Lempel-Ziv 77 (LZ7) também se concentra na velocidade de compactação e descompactação, não na compactação máxima dos dados. O LZ4 tem os seguintes formatos de enquadramento:

  - LZ4 Raw/Unframed: uma implementação sem quadros padrão do formato de compactação de blocos LZ4. Para obter mais informações, consulte LZ4 Block Format Description (Descrição do formato de bloco LZ4) no GitHub.
  - LZ4 framed: a implementação de enquadramento usual do LZ4. Para obter mais informações, consulte LZ4 Frame Format Description (Descrição do formato de quadro LZ4) no GitHub.
  - LZ4 Hadoop-compatible: a implementação do Apache Hadoop do LZ4. Esta implementação envolve a compactação LZ4 com a classe BlockCompressorStream.java.
- LZO: formato que usa o algoritmo Lempel-Ziv-Oberhumer, que se concentra na alta velocidade de compactação e descompactação, não na compactação máxima dos dados. O LZO tem duas implementações:
  - Standard LZO: para obter mais informações, consulte o resumo do LZO no site da Oberhumer.
  - LZO Hadoop-compatible: esta implementação envolve o algoritmo LZO com a classe BlockCompressorStream.java.
- SNAPPY: algoritmo de compactação que faz parte da família Lempel-Ziv 77 (LZ7). O Snappy se concentra na alta velocidade de compactação e descompactação, não na compactação máxima do dados.
- ZLIB: com base no Deflate, ZLIB é o formato de compactação de gravação padrão para arquivos no formato de armazenamento de dados ORC. Para obter mais informações, consulte a página zib no GitHub.
- ZSTD: o algoritmo de compactação de dados Zstandard é um algoritmo de compactação rápida que fornece altas taxas de compactação. A biblioteca Zstandard (ZSTD) é fornecida como software de código aberto usando uma licença BSD. O ZSTD é a compressão padrão para tabelas do Iceberg. O Athena usa o nível 3 de compressão ZSTD por padrão quando grava dados compactados como ZSTD. Para obter mais informações sobre o uso de níveis de compressão ZSTD no Athena, consulte Usar níveis de compressão ZSTD no Athena.

Especificar os formatos de compactação

Ao escrever instruções CREATE TABLE ou CTAS, você pode especificar propriedades de compactação que especifiquem o tipo de compactação a ser usado quando o Athena gravar nessas tabelas.

- Para CTAS, consulte Propriedades da tabela CTAS. Para ver exemplos, consulte Exemplos de consultas CTAS.

- Para CREATE TABLE, consulte ALTER TABLE SET TBLPROPERTIES para obter uma lista de propriedades da tabela de compactação.

Especificar a não compactação

As instruções CREATE TABLE são compatíveis com a gravação de arquivos descompactados. Para gravar arquivos descompactados, use a seguinte sintaxe:

- CREATE TABLE (arquivo de texto ou JSON): Em `TBLPROPERTIES`, especifique `write.compression = NONE`.
- CREATE TABLE (Parquet): em `TBLPROPERTIES`, especifique `parquet.compression = UNCOMPRESSED`.
- CREATE TABLE (ORC): em `TBLPROPERTIES`, especifique `orc.compress = NONE`.

Observações e recursos

- Atualmente, extensões de arquivos em letras maiúsculas, como `.GZ` ou `.BZIP2`, não são reconhecidas pelo Athena. Evite usar conjuntos de dados com extensões de arquivos em letras maiúsculas ou altere essas extensões para letras minúsculas.
- Para dados em CSV, TSV e JSON, o Athena determina o tipo de compactação com base na extensão do arquivo. Se nenhuma extensão de arquivo estiver presente, o Athena tratará os dados como texto simples sem compactação. Se os dados estiverem compactados, certifique-se de que o nome do arquivo inclua a extensão de compactação, como `gz`.
- O formato de arquivo ZIP não é compatível.
- Para consultar os logs do Amazon Data Firehose usando o Athena, os formatos compatíveis incluem a compactação GZIP ou os arquivos ORC com compactação SNAPPY.
- Para obter mais informações sobre como usar a compactação, consulte a seção 3 (“Compress and split files” [Compactar e dividir arquivos]) da publicação do blog da AWS sobre big data: Top 10 performance tuning tips for Amazon Athena (As dez melhores dicas de ajuste de performance do Amazon Athena).

Suporte à compressão de tabelas do Hive por formato de arquivo

O suporte à compressão do Hive no Athena depende da versão do mecanismo.

Suporte à compressão do Hive na versão 3 do mecanismo do Athena

A tabela a seguir resume o suporte a formatos de compressão na versão 3 do mecanismo no Athena para formatos de arquivo de armazenamento no Apache Hive. O formato de arquivo de texto inclui TSV, CSV, JSON e SerDes personalizado para texto. “Sim” ou “Não” em uma célula se aplicam igualmente às operações de leitura e gravação, exceto quando indicado. Para as finalidades desta tabela, CREATE TABLE, CTAS e INSERT INTO serão consideradas operações de gravação. Para obter mais informações sobre o uso de níveis de compressão ZSTD no Athena, consulte Usar níveis de compressão ZSTD no Athena.

	Avro	Ion	ORC	Parquet	Arquivo de texto
BZIP2	Sim	Sim	Não	Não	Sim
DEFLATE	Sim	Não	Não	Não	Não
GZIP	Não	Sim	Não	Sim	Sim
LZ4	Não	Sim	Sim	Sim	Sim
LZO	Não	Gravação: não Leitura: sim	Não	Sim	Gravação: não Leitura: sim
SNAPPY	Sim	Sim	Sim	Sim	Sim
ZLIB	Não	Não	Sim	Não	Não
ZSTD	Sim	Sim	Sim	Sim	Sim
NONE	Sim	Sim	Sim	Sim	Sim

Suporte à compressão do Hive na versão 2 do mecanismo do Athena

A tabela a seguir resume o suporte a formatos de compressão na versão 2 do mecanismo do Athena para Apache Hive. O formato de arquivo de texto inclui TSV, CSV, JSON e SerDes personalizado para texto. “Sim” ou “Não” em uma célula se aplicam igualmente às operações de leitura e gravação,

exceto quando indicado. Para as finalidades desta tabela, CREATE TABLE, CTAS e INSERT INTO serão consideradas operações de gravação.

	Avro	Ion	ORC	Parquet	Arquivo de texto
BZIP2	Sim	Sim	Não	Não	Sim
DEFLATE	Sim	Não	Não	Não	Não
GZIP	Não	Sim	Não	Sim	Sim
LZ4	Não	Não	Sim	Gravação: sim Leitura: não	Gravação: não Leitura: sim
LZO	Não	Gravação: não Leitura: sim	Não	Sim	Gravação: não Leitura: sim
SNAPPY	Sim	Sim	Sim	Sim	Sim
ZLIB	Não	Não	Sim	Não	Não
ZSTD	Não	Sim	Sim	Sim	Sim
NONE	Sim	Sim	Sim	Sim	Sim

Suporte à compressão de tabelas do Iceberg por formato de arquivo

O suporte à compressão do Apache Iceberg no Athena depende da versão do mecanismo.

Suporte à compressão do Iceberg na versão 3 do mecanismo do Athena

A tabela a seguir resume o suporte a formatos de compressão na versão 3 do mecanismo no Athena para formatos de arquivo de armazenamento no Apache Iceberg. “Sim” ou “Não” em uma célula se

aplicam igualmente às operações de leitura e gravação, exceto quando indicado. Para as finalidades desta tabela, CREATE TABLE, CTAS e INSERT INTO serão consideradas operações de gravação. O formato de armazenamento padrão para o Iceberg no mecanismo do Athena versão 3 é Parquet. O formato de compactação padrão para o Iceberg no mecanismo do Athena versão 3 é ZSTD. Para obter mais informações sobre o uso de níveis de compressão ZSTD no Athena, consulte Usar níveis de compressão ZSTD no Athena.

	Avro	ORC	Parquet (padrão)
BZIP2	Não	Não	Não
GZIP	Sim	Não	Sim
LZ4	Não	Sim	Não
SNAPPY	Sim	Sim	Sim
ZLIB	Não	Sim	Não
ZSTD	Sim	Sim	Sim (padrão)
NONE	Sim (especificar `None` ou `Deflate`)	Sim	Sim (especificar `None` ou `Uncompressed`)

Suporte à compressão do Iceberg na versão 2 do mecanismo do Athena

A tabela a seguir resume o suporte a formatos de compressão na versão 2 do mecanismo do Athena para Apache Iceberg. “Sim” ou “Não” em uma célula se aplicam igualmente às operações de leitura e gravação, exceto quando indicado. Para as finalidades desta tabela, CREATE TABLE, CTAS e INSERT INTO serão consideradas operações de gravação. O formato de armazenamento padrão para o Iceberg no mecanismo do Athena versão 2 é Parquet. O formato de compactação padrão para o Iceberg no mecanismo do Athena versão 2 é GZIP.

	Avro (Sem suporte)	ORC (Não suportado)	Parquet (padrão)
BZIP2	Não	Não	Não

	Avro (Sem suporte)	ORC (Não suportado)	Parquet (padrão)
GZIP	Não	Não	Sim (padrão)
LZ4	Não	Não	Não
SNAPPY	Não	Não	Sim
ZLIB	Não	Não	Não
ZSTD	Não	Não	Sim
NONE	Não	Não	Sim

Usar níveis de compressão ZSTD no Athena

O algoritmo de compressão de dados em tempo real Zstandard é um algoritmo de compressão rápida que fornece altas taxas de compressão. A biblioteca Zstandard (ZSTD) é um software de código aberto e usa uma licença BSD. O Athena oferece suporte a leitura e a gravação de dados nos formatos ORC, Parquet e arquivo de texto compactados como ZSTD.

Você pode usar os níveis de compressão ZSTD para ajustar a taxa e a velocidade de compressão de acordo com suas necessidades. A biblioteca ZSTD é compatível com níveis de compressão de 1 a 22. O Athena usa o nível 3 de compressão ZSTD por padrão.

Os níveis de compressão oferecem compensações granulares entre a velocidade e a quantidade de compressão alcançadas. Níveis de compressão mais baixos oferecem maior velocidade, mas tamanhos de arquivo maiores. Por exemplo, você pode usar o nível 1 se a velocidade for mais importante e o nível 22 se o tamanho for mais importante. O nível 3 é adequado para muitos casos de uso e é o padrão. Use níveis maiores que 19 com cuidado, pois eles exigem mais memória. A biblioteca ZSTD também oferece níveis de compressão negativos que ampliam a faixa de velocidade e taxas de compressão. Para obter mais informações, consulte Zstandard Compression RFC.

A abundância de níveis de compressão oferece oportunidades substanciais para ajustes finos. No entanto, certifique-se de medir seus dados e considerar as desvantagens ao decidir sobre um nível de compressão. Recomendamos usar o nível padrão de 3 ou um nível na faixa de 6 a 9 para uma compensação razoável entre a velocidade de compressão e o tamanho dos dados compactados.

Reserve níveis 20 ou mais para casos em que o tamanho é mais importante e a velocidade de compressão não é uma preocupação.

Considerações e limitações

Ao usar o nível de compressão ZSTD no Athena, acesse os seguintes pontos.

- A propriedade `compression_level` da ZSTD tem suporte apenas no mecanismo Athena versão 3.
- A propriedade `compression_level` da ZSTD é compatível com as instruções `ALTER TABLE`, `CREATE TABLE`, `CREATE TABLE AS` (CTAS) e `UNLOAD`.
- A propriedade `compression_level` é opcional.
- A propriedade `compression_level` tem suporte apenas para compressão ZSTD.
- Os níveis de compressão possíveis são de 1 a 22.
- O nível de compressão padrão é 3.

Para obter informações sobre suporte à compactação do Apache Hive ZSTD no Athena, consulte Suporte à compressão de tabelas do Hive por formato de arquivo. Para obter informações sobre suporte à compactação do Apache Iceberg ZSTD no Athena, consulte Suporte à compressão de tabelas do Iceberg por formato de arquivo.

Especificar os níveis de compressão ZSTD

Para especificar o nível de compressão ZSTD para as instruções `ALTER TABLE`, `CREATE TABLE`, `CREATE TABLE AS` e `UNLOAD`, use a propriedade `compression_level`. Para especificar a compressão ZSTD em si, você deve usar a propriedade de compressão individual que a sintaxe da instrução usa.

ALTER TABLE SET TBLPROPERTIES

Na cláusula `SET TBLPROPERTIES` da instrução ALTER TABLE SET TBLPROPERTIES, especifique a compressão ZSTD usando `'write.compression' = ' ZSTD'` ou `'parquet.compression' = 'ZSTD'`. Em seguida, use a propriedade `compression_level` para especificar um valor de 1 a 22 (por exemplo, `'compression_level' = 5`). Se você não especificar uma propriedade de nível de compressão, o padrão será 3.

Exemplo

O exemplo a seguir modifica a tabela `existing_table` para usar o formato de arquivo Parquet com compressão ZSTD nível 4. O valor do nível de compactação deve ser inserido como uma string em vez de um número inteiro.

```
ALTER TABLE existing_table
SET TBLPROPERTIES ('parquet.compression' = 'ZSTD', 'compression_level' = 4)
```

CRIAR TABELA

Na cláusula `TBLPROPERTIES` da instrução CREATE TABLE, especifique `'write.compression' = 'ZSTD'` ou `'parquet.compression' = 'ZSTD'`, em seguida, use `compression_level = compression_level` e especifique um valor de 1 a 22. Se a propriedade `compression_level` não for especificada, o nível de compressão padrão será 3.

Exemplo

O exemplo a seguir cria uma tabela no formato de arquivo Parquet usando compressão ZSTD nível 4.

```
CREATE EXTERNAL TABLE new_table (
  `col0` string COMMENT '',
  `col1` string COMMENT ''
)
STORED AS PARQUET
LOCATION 's3://DOC-EXAMPLE-BUCKET/'
TBLPROPERTIES ('write.compression' = 'ZSTD', 'compression_level' = 4)
```

CREATE TABLE AS (CTAS)

Na cláusula `WITH` da instrução CREATE TABLE AS, especifique `write_compression = 'ZSTD'` ou `parquet_compression = 'ZSTD'`, em seguida, use `compression_level = compression_level` e especifique um valor de 1 a 22. Se a propriedade `compression_level` não for especificada, o nível de compressão padrão será 3.

Exemplo

O exemplo de CTAS a seguir especifica o Parquet como formato de arquivo usando compressão ZSTD nível 4.

```
CREATE TABLE new_table
WITH ( format = 'PARQUET', write_compression = 'ZSTD', compression_level = 4)
AS SELECT * FROM old_table
```

UNLOAD

Na cláusula `WITH` da instrução UNLOAD, especifique `compression = 'ZSTD'`, em seguida, use `compression_level = compression_level` e especifique um valor de 1 a 22. Se a propriedade `compression_level` não for especificada, o nível de compressão padrão será 3.

Exemplo

O exemplo a seguir descarrega os resultados da consulta no local especificado usando o formato de arquivo Parquet, a compressão ZSTD e o nível 4 de compressão ZSTD.

```
UNLOAD (SELECT * FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/'
WITH (format = 'PARQUET', compression = 'ZSTD', compression_level = 4)
```

Etiquetar recursos do Athena

Uma tag consiste em uma chave e um valor, ambos definidos por você. Ao marcar um recurso do Athena, você atribui metadados personalizados a ele. Você pode usar etiquetas para categorizar seus recursos da AWS de diferentes formas, como por finalidade, proprietário ou ambiente. No Athena, recursos como grupos de trabalho, catálogos de dados e reservas de capacidade são recursos etiquetáveis. Por exemplo, você pode criar um conjunto de tags para grupos de trabalho na sua conta que ajuda a rastrear os proprietários do grupo de trabalho ou identificar grupos de trabalho por finalidade. Se você também habilitar as tags como tags de alocação de custos no console do Billing and Cost Management, os custos associados à execução de consultas aparecerão em seu Relatório de custos e uso com essa tag de alocação de custos. Recomendamos usar as práticas recomendadas de marcação com tags da AWS para criar um conjunto consistente de tags que atenda às necessidades da sua organização.

Você pode trabalhar com etiquetas usando o console do Athena ou as operações de API.

Tópicos

- Conceitos Básicos de Tags
- Restrições de tags

- Trabalhar com etiquetas em grupos de trabalho no console
- Usar operações de etiquetas
- Políticas de controle de acesso do IAM baseadas em etiquetas

Conceitos Básicos de Tags

Uma etiqueta é um rótulo atribuído a um recurso do Athena. Cada tag consiste em uma chave e um valor opcional, ambos definidos por você.

As etiquetas permitem categorizar seus recursos da AWS de maneiras diferentes. Por exemplo, você pode definir um conjunto de tags para os grupos de trabalho da sua conta que ajudem a rastrear o proprietário ou a finalidade de cada grupo de trabalho.

É possível adicionar etiquetas ao criar um grupo de trabalho ou catálogo de dados do Athena ou adicionar, editar ou remover etiquetas dele. Você pode editar uma tag no console. Se você usar as operações da API para editar uma tag, remova a tag antiga e adicione uma nova. Caso exclua um recurso, todas as respectivas tags também serão excluídas.

O Athena não atribui etiquetas automaticamente aos recursos. É possível editar chaves de tags e valores, e é possível remover as tags de um recurso a qualquer momento. É possível definir o valor de uma tag a uma string vazia, mas não pode configurar o valor de um tag como nula. Não adicione chaves de tag duplicadas ao mesmo recurso. Se você fizer isso, o Athena emitirá uma mensagem de erro. Se você usar a ação TagResource para marcar um recurso usando uma chave de tag existente, o novo valor da tag substituirá o valor antigo.

No IAM, é possível controlar quais usuários na sua conta da Amazon Web Services têm permissão para criar, editar, remover ou listar etiquetas. Para ter mais informações, consulte Políticas de controle de acesso do IAM baseadas em etiquetas.

Para obter uma lista completa de ações de etiqueta do Amazon Athena, consulte os nomes das ações de API na Referência de API do Amazon Athena.

Você pode usar tags para faturamento. Para obter mais informações, consulte Usar etiquetas para faturamento no Guia do usuário do AWS Billing and Cost Management.

Para ter mais informações, consulte Restrições de tags.

Restrições de tags

As tags têm as seguintes restrições:

- No Athena, você pode marcar grupos de trabalho e catálogos de dados. Você não pode marcar consultas com tags.
- O número máximo de tags por recurso é 50. Para permanecer no limite, revise e exclua tags não utilizadas.
- Em todos os recursos, cada chave de tag deve ser exclusiva e pode ter apenas um valor. Não adicione chaves de tag duplicadas ao mesmo tempo ao mesmo recurso. Se você fizer isso, o Athena emitirá uma mensagem de erro. Se você marcar um grupo de trabalho usando uma chave de tag existente em uma ação `TagResource` separada, o novo valor da tag substituirá o valor antigo.
- O comprimento da chave da tag é de 1-128 caracteres Unicode em UTF-8.
- O comprimento do valor da tag é de 0-256 caracteres Unicode em UTF-8.

  Marcar operações, como adicionar, editar, remover ou listar tags, exigem que você especifique um ARN para o recurso do grupo de trabalho.
- O Athena permite usar letras, números, espaços representados em UTF-8 e os seguintes caracteres: + - = . _ : / @.
- As chaves e os valores de tags diferenciam maiúsculas de minúsculas.
- O prefixo `"aws:"` nas chaves de tag é reservado para uso da AWS. Você não pode editar nem excluir chaves de tag com esse prefixo. As tags com esse prefixo não contam contra o limite de tags por recurso.
- As etiquetas atribuídas estão disponíveis somente para a sua conta da Amazon Web Services.

Trabalhar com etiquetas em grupos de trabalho no console

Usando o console do Athena, você pode ver quais etiquetas estão em uso por cada grupo de trabalho na sua conta. Você só pode visualizar as tags por grupo de trabalho. Você também pode usar o console do Athena para aplicar, editar ou remover as etiquetas de um grupo de trabalho por vez.

Você pode pesquisar grupos de trabalho usando as tags criadas.

Tópicos

- Exibir etiquetas para grupos de trabalho individuais
- Adicionar e excluir etiquetas em um grupo de trabalho individual

Exibir etiquetas para grupos de trabalho individuais

Para exibir as etiquetas de um grupo de trabalho individual no console do Athena

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.

3. No menu de navegação, escolha Workgroups (Grupos de trabalho) e escolha o grupo de trabalho desejado.
4. Execute um destes procedimentos:
   - Escolha a guia Tags. Se a lista de etiquetas for longa, use a caixa de pesquisa.
   - Escolha Edit (Editar) e, em seguida, role para baixo até a seção Tags (Etiquetas).

Adicionar e excluir etiquetas em um grupo de trabalho individual

Você pode gerenciar as tags de um único grupo de trabalho diretamente pela guia Workgroups (Grupos de trabalho).

> **Note**
>
> Se você deseja que os usuários adicionem etiquetas ao criarem um grupo de trabalho no console ou especifiquem etiquetas quando usarem a ação CreateWorkGroup, conceda a eles permissões do IAM para as ações TagResource e CreateWorkGroup.

Para adicionar uma etiqueta ao criar um novo grupo de trabalho

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. No menu de navegação, escolha Workgroups (Grupos de trabalho).
3. Escolha Create workgroup (Criar grupo de trabalho) e preencha os valores, conforme necessário. Para obter detalhes das etapas, consulte, Criar um grupo de trabalho.
4. Na seção Tags (Etiquetas), adicione uma ou mais etiquetas especificando chaves e valores. Não adicione chaves de tag duplicadas ao mesmo tempo ao mesmo grupo de trabalho. Se você fizer isso, o Athena emitirá uma mensagem de erro. Para ter mais informações, consulte Restrições de tags.
5. Ao concluir, escolha Create workgroup (Criar grupo de trabalho).

Para adicionar ou editar uma tag a um grupo de trabalho existente

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. No painel de navegação, escolha Global networks (Redes globais).
3. Escolha o grupo de trabalho que você deseja modificar.
4. Execute um destes procedimentos:
   - Escolha a guia Tags e escolha Gerenciar tags.
   - Escolha Edit (Editar) e, em seguida, role para baixo até a seção Tags (Etiquetas).
5. Especifique uma chave e um valor para cada etiqueta. Para ter mais informações, consulte Restrições de tags.
6. Escolha Salvar.

Para excluir uma tag de um grupo de trabalho individual

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. No painel de navegação, escolha Global networks (Redes globais).
3. Escolha o grupo de trabalho que você deseja modificar.
4. Execute um destes procedimentos:
   - Escolha a guia Tags e escolha Gerenciar tags.
   - Escolha Edit (Editar) e, em seguida, role para baixo até a seção Tags (Etiquetas).

5. Na lista de etiquetas, escolha Remove (Remover) para a etiqueta que você quer excluir e, em seguida, escolha Save (Salvar).

Usar operações de etiquetas

Use as seguintes operações de tag para adicionar, remover ou listar tags em um recurso.

API	CLI	Descrição da ação
`TagResource`	`tag-resource`	Adicione ou substitua uma ou mais tags no recurso que tem o ARN especificado.
`UntagResource`	`untag-resource`	Exclua uma ou mais tags do recurso que tem o ARN especificado.
`ListTagsForResource`	`list-tags-for-resource`	Liste uma ou mais tags para o recurso que tem o ARN especificado.

Adicionar tags ao criar um recurso

Para adicionar tags ao criar um grupo de trabalho ou catálogo de dados, use o parâmetro `tags` com as operações `CreateDataCatalog` ou `CreateWorkGroup` da API ou com os comandos `create-work-group` ou `create-data-catalog` da AWS CLI.

Gerenciar etiquetas usando operações da API

Os exemplos nesta seção mostram como usar operações da API de tags para gerenciar tags em grupos de trabalho e catálogos de dados. Os exemplos estão na linguagem de programação Java.

Example TagResource

O exemplo a seguir adiciona duas tags ao grupo de trabalho `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);
```

O exemplo a seguir adiciona duas tags ao catálogo de dados `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**
>
> Não adicione chaves de tag duplicadas ao mesmo recurso. Se você fizer isso, o Athena emitirá uma mensagem de erro. Se você marcar um grupo de trabalho usando uma chave de tag existente em uma ação `TagResource` separada, o novo valor da tag substituirá o valor antigo.

Example UntagResource

O exemplo a seguir remove `tagKey2` do grupo de trabalho `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);
```

O exemplo a seguir remove `tagKey2` do catálogo de dados `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

O exemplo a seguir lista as tags do grupo de trabalho `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();
```

O exemplo a seguir lista as tags do catálogo de dados `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();
```

Gerenciar tags usando o AWS CLI

As seções a seguir mostram como usar a AWS CLI para criar e gerenciar tags em catálogos de dados.

Adicionar etiquetas a um recurso: tag-resource

O comando `tag-resource` adiciona uma ou mais tags a um recurso especificado.

Sintaxe

```
aws athena tag-resource --resource-arn
arn:aws:athena:region:account_id:datacatalog/catalog_name --tags
Key=string,Value=string Key=string,Value=string
```

O parâmetro `--resource-arn` especifica o recurso ao qual as tags são adicionadas. O parâmetro `--tags` especifica uma lista de pares de chave-valor separados por espaço a serem adicionados como tags ao recurso.

Example

O exemplo a seguir adiciona tags ao catálogo de dados `mydatacatalog`.

```
aws athena tag-resource --resource-arn arn:aws:athena:us-
east-1:111122223333:datacatalog/mydatacatalog --tags Key=Color,Value=Orange
 Key=Time,Value=Now
```

Para mostrar o resultado, use o comando `list-tags-for-resource`.

Para obter informações sobre como adicionar etiquetas ao usar o comando `create-data-catalog`, consulte Registrar um catálogo: create-data-catalog.

Listar as etiquetas de um recurso: list-tags-for-resource

O comando `list-tags-for-resource` lista as tags do recurso especificado.

Sintaxe

`aws athena list-tags-for-resource --resource-arn arn:aws:athena:`*`region`*`:`*`account_id`*`:datacatalog/`*`catalog_name`*

O parâmetro `--resource-arn` especifica o recurso para o qual as tags são listadas.

O exemplo a seguir lista as tags do catálogo de dados `mydatacatalog`.

```
aws athena list-tags-for-resource --resource-arn arn:aws:athena:us-
east-1:111122223333:datacatalog/mydatacatalog
```

O resultado de exemplo a seguir está em formato JSON.

```
{
    "Tags": [
        {
            "Key": "Time",
            "Value": "Now"
        },
        {
            "Key": "Color",
```

```
            "Value": "Orange"
        }
    ]
}
```

Remover etiquetas de um recurso: untag-resource

O comando `untag-resource` remove as chaves de tag especificadas e seus valores associados do recurso especificado.

Sintaxe

`aws athena untag-resource --resource-arn arn:aws:athena:`*`region`*`:`*`account_id`*`:datacatalog/`*`catalog_name`* `--tag-keys` *`key_name`* `[`*`key_name`* `...]`

O parâmetro `--resource-arn` especifica o recurso do qual as tags são removidas. O parâmetro `--tag-keys` usa uma lista separada por espaços de nomes de chave. Para cada nome de chave especificado, o comando `untag-resource` remove a chave e seu valor.

O exemplo a seguir remove as chaves `Color` e `Time` e seus valores do recurso de catálogo `mydatacatalog`.

```
aws athena untag-resource --resource-arn arn:aws:athena:us-
east-1:111122223333:datacatalog/mydatacatalog --tag-keys Color Time
```

Políticas de controle de acesso do IAM baseadas em etiquetas

As etiquetas permitem que você escreva uma política do IAM que inclua o bloco `Condition` para controlar o acesso a um recurso com base em suas etiquetas.

Exemplos de política de etiquetas para grupos de trabalho

Example 1. Política básica de marcação com etiquetas

A seguinte política do IAM permite que você execute consultas e interaja com as etiquetas do grupo de trabalho chamado `workgroupA`:

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

```
            "Effect": "Allow",
            "Action": [
                "athena:ListWorkGroups",
                "athena:ListEngineVersions",
                "athena:ListDataCatalogs",
                "athena:ListDatabases",
                "athena:GetDatabase",
                "athena:ListTableMetadata",
                "athena:GetTableMetadata"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "athena:GetWorkGroup",
                "athena:TagResource",
                "athena:UntagResource",
                "athena:ListTagsForResource",
                "athena:StartQueryExecution",
                "athena:GetQueryExecution",
                "athena:BatchGetQueryExecution",
                "athena:ListQueryExecutions",
                "athena:StopQueryExecution",
                "athena:GetQueryResults",
                "athena:GetQueryResultsStream",
                "athena:CreateNamedQuery",
                "athena:GetNamedQuery",
                "athena:BatchGetNamedQuery",
                "athena:ListNamedQueries",
                "athena:DeleteNamedQuery",
                "athena:CreatePreparedStatement",
                "athena:GetPreparedStatement",
                "athena:ListPreparedStatements",
                "athena:UpdatePreparedStatement",
                "athena:DeletePreparedStatement"
            ],
            "Resource": "arn:aws:athena:us-east-1:123456789012:workgroup/workgroupA"
        }
    ]
}
```

Example 2: Bloco de política que nega ações em um grupo de trabalho com base em par de chave e valor de etiqueta

As tags associadas a um recurso, como um grupo de trabalho, são chamadas de tags de recurso. As tags de recursos permitem escrever blocos de política, como os seguintes, que negam as ações listadas em qualquer grupo de trabalho marcado com um par de chave-valor como `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. Bloco de política que restringe solicitações de ação de mudança de etiqueta a etiquetas específicas

As tags que são passadas como parâmetros para operações que alteram tags (por exemplo, `TagResource`, `UntagResource` ou `CreateWorkGroup` com tags) são chamadas de tags de solicitação. O seguinte exemplo de bloco de política permite a operação `CreateWorkGroup` apenas se uma das tags passadas tiver a chave `costcenter` e o valor `1`, `2` ou `3`.

> **Note**
>
> Se você deseja permitir que um perfil do IAM especifique as etiquetas como parte de uma operação `CreateWorkGroup`, certifique-se de conceder ao perfil as permissões para as ações `TagResource` e `CreateWorkGroup`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:CreateWorkGroup",
                "athena:TagResource"
            ],
            "Resource": "arn:aws:athena:us-east-1:123456789012:workgroup/*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/costcenter": [
                        "1",
                        "2",
                        "3"
                    ]
                }
            }
        }
    ]
```

```
}
```

Exemplos de políticas de etiquetas para catálogos de dados

Example 1. Política básica de marcação com etiquetas

A seguinte política do IAM permite interagir com as etiquetas do catálogo de dados chamado `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: Bloco de política que nega ações em um grupo de trabalho com base em um par de chave e valor de etiqueta

Você pode usar tags de recursos para gravar blocos de política que negam ações específicas em catálogos de dados marcados com pares de chave-valor de tag específicos. O exemplo a seguir nega ações em catálogos de dados que têm o par de chave-valor da 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. Bloco de política que restringe solicitações de ação de mudança de etiqueta a etiquetas específicas

As tags que são passadas como parâmetros para operações que alteram tags (por exemplo, `TagResource`, `UntagResource` ou `CreateDataCatalog` com tags) são chamadas de tags de solicitação. O seguinte exemplo de bloco de política permite a operação `CreateDataCatalog` apenas se uma das tags passadas tiver a chave `costcenter` e o valor 1, 2 ou 3.

> **Note**
>
> Se você deseja permitir que um perfil do IAM especifique as etiquetas como parte de uma operação `CreateDataCatalog`, certifique-se de conceder ao perfil as permissões para as ações `TagResource` e `CreateDataCatalog`.

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

```
                "athena:CreateDataCatalog",
                "athena:TagResource"
            ],
            "Resource": "arn:aws:athena:us-east-1:123456789012:datacatalog/*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/costcenter": [
                        "1",
                        "2",
                        "3"
                    ]
                }
            }
        }
    ]
}
```

Service Quotas

> **Note**
>
> O console do Service Quotas inclui informações sobre as cotas do Amazon Athena. Você também pode usar o console do Service Quotas para solicitar aumentos de cotas para aquelas que são ajustáveis. Para obter as limitações de esquema relacionadas ao AWS Glue, consulte a página Endpoints e cotas do AWS Glue. Para obter informações gerais sobre cotas de serviço da AWS, consulte Cotas de serviço da AWS na Referência geral da AWS.

Consultas

Sua conta tem as seguintes cotas relacionadas a consultas para o Amazon Athena. Para obter detalhes, consulte a página Endpoints e cotas do Amazon Athena da Referência geral da AWS.

- Active DDL queries (Consultas DDL ativas): o número de consultas DDL ativas. As consultas DDL incluem consultas `CREATE TABLE` e `ALTER TABLE ADD PARTITION`.
- DDL query timeout (Tempo limite de consulta DDL): a quantidade máxima de tempo em minutos que uma consulta DDL pode ser executada antes de ser cancelada.

- Active DML queries (Consultas DML ativas): o número de consultas DML ativas. As consultas DML incluem consultas `SELECT`, `CREATE TABLE AS` (CTAS) e `INSERT INTO`. As cotas específicas variam de acordo com a região da AWS.
- DML query timeout (Tempo limite de consulta DML): a quantidade máxima de tempo em minutos que uma consulta DML pode ser executada antes de ser cancelada. É possível solicitar um aumento desse tempo limite para até 240 minutos.

Para solicitar aumentos de cotas, você pode usar o console do Service Quotas para o Athena.

O Athena processa as consultas atribuindo recursos com base na carga geral do serviço e no número de solicitações recebidas. Suas consultas podem ser temporariamente enfileiradas antes da execução. Os processos assíncronos retiram as consultas das filas e as executam nos recursos físicos assim que eles se tornam disponíveis e conforme permitido na configuração da sua conta.

Uma cota de consulta DML ou DDL inclui consultas tanto em execução quanto enfileiradas. Por exemplo, se sua conta de consultas DML for 25 e seu total de consultas em execução e em fila for 26, a consulta 26 resultará em um erro TooManyRequestsException.

> **Note**
>
> Se você deseja controlar a concorrência diretamente para as consultas que você executa no Athena, pode usar reservas de capacidade. Para obter mais informações, consulte Como gerenciar a capacidade de processamento de consulta.

Comprimento da string de consulta

O comprimento máximo permitido da string da consulta é de 262.144 bytes, pois as strings são codificadas em UTF-8. Esta não é uma cota ajustável. No entanto, você pode resolver esse problema dividindo as consultas grandes em várias consultas menores. Para obter mais informações, consulte Como posso aumentar o tamanho máximo da string de consulta no Athena? (em inglês) na Central de Conhecimento da AWS.

Grupos de trabalho

Ao usar os grupos de trabalho do Athena, lembre-se dos seguintes pontos:

- As cotas do serviço do Athena são compartilhadas com todos os grupos de trabalho em uma conta.

- O número máximo de grupos de trabalho que podem ser criados por região em uma conta é 1000.
- O número máximo de instruções preparadas em um grupo de trabalho é mil.
- O número máximo de tags por grupo de trabalho é 50. Para obter mais informações, consulte Restrições de tags.

Bancos de dados, tabelas e partições

- Se você estiver usando o AWS Glue Data Catalog com o Athena, consulte Endpoints e cotas do AWS Glue para conhecer as cotas de serviço referentes a tabelas, bancos de dados e partições, por exemplo, o número máximo de bancos de dados ou de tabelas por conta.
  - Embora o Athena ofereça suporte a consulta a tabelas do AWS Glue com 10 milhões de partições, o Athena não pode ler mais de 1 milhão de partições em uma única varredura.
- Se você não estiver usando o AWS Glue Data Catalog, o número de partições por tabela é 20.000. É possível solicitar um aumento da cota.

Buckets do Amazon S3

Ao trabalhar com buckets do Amazon S3, lembre-se dos seguintes pontos:

- O Amazon S3 tem uma cota de serviço padrão de 100 buckets por conta.
- O Athena requer um bucket separado para registrar os resultados.
- Você pode solicitar um aumento da cota de até 1.000 buckets do Amazon S3 por conta da AWS.

Cotas de chamada de API por conta

As APIs do Athena têm as seguintes cotas padrão para o número de chamadas à API por conta (não por consulta):

Nome da API	Número padrão de chamadas por segundo	Capacidade de expansão
`BatchGetNamedQuery` , `ListNamedQueries` , `ListQueryExecutions`	5	até 10

Nome da API	Número padrão de chamadas por segundo	Capacidade de expansão
`CreateNamedQuery` , `DeleteNamedQuery` , `GetNamedQuery`	5	até 20
`BatchGetQueryExecution`	20	até 40
`StartQueryExecution` , `StopQueryExecution`	20	até 80
`GetQueryExecution` , `GetQueryResults`	100	até 200

Por exemplo, para `StartQueryExecution`, é possível fazer até 20 chamadas por segundo. Além disso, se essa API não for chamada por 4 segundos, sua conta acumulará uma capacidade de intermitência de até 80 chamadas. Nesse caso, o aplicativo pode fazer até 80 chamadas para essa API no modo de intermitência.

Se você usar qualquer uma dessas APIs e exceder a cota padrão de número de chamadas por segundo ou a capacidade de expansão da sua conta, a API do Athena emitirá um erro semelhante ao seguinte: “ClientError: Ocorreu um erro (ThrottlingException) ao chamar a operação da <nome_da_API>: taxa excedida.” Reduza o número de chamadas por segundo ou a capacidade de intermitência da API para essa conta.

A cota do Athena para chamadas de API por conta não pode ser alterada no console Service Quotas do Athena. Para solicitar um aumento de cota para chamadas da API Athena, navegue até a página deAWS Support aumento do limite de serviço, preencha e envie o formulário.

Versionamento do mecanismo do Athena

Esporadicamente, o Athena lança uma nova versão do mecanismo para oferecer melhor performance, funcionalidade e correções de código. Quando uma nona versão está disponível, o Athena notifica você no console do Athena e no seu AWS Health Dashboard. O AWS Health Dashboard notifica você sobre os eventos que podem afetar seus serviços ou sua conta da AWS. Para obter mais informações sobre o AWS Health Dashboard, consulte Conceitos básicos do AWS Health Dashboard.

O versionamento do mecanismo é configurado por grupo de trabalho. É possível usar grupos de trabalho para controlar qual o mecanismo de consultas usado por suas consultas e se o Athena fará ou não upgrade automático dos grupos de trabalho. O mecanismo de consulta que está em uso aparece no editor de consultas, na página de detalhes do grupo de trabalho e fica disponível por meio das APIs do Athena.

- Por padrão, os grupos de trabalho são configurados para atualização automática. Quando um grupo de trabalho estiver configurado para fazer upgrade automático, ele fará upgrade do Athena, a menos que encontre incompatibilidades.
- Se você configurar um grupo de trabalho para usar uma determinada versão, o Athena não alterará a versão do grupo de trabalho.

Em ambos os casos, o Athena atualizará seus grupos de trabalho quando uma versão não estiver mais disponível. O Athena notificará no AWS Health Dashboard sobre quando uma versão do mecanismo deixará de ser oferecida. O AWS Health Dashboard notifica você sobre os eventos que podem afetar seus serviços ou sua conta da AWS. Para obter mais informações sobre o AWS Health Dashboard, consulte Conceitos básicos do AWS Health Dashboard.

Quando você começar a usar uma nova versão do mecanismo, um pequeno subconjunto de consultas pode ser interrompido devido a incompatibilidades. As mudanças disruptivas são anunciadas no lançamento de uma nova versão do Athena. Você deve usar os grupos de trabalho para testar as consultas antes do upgrade, criando um grupo de trabalho de teste que use o novo mecanismo ou fazendo um upgrade de teste de um grupo de trabalho existente. Para obter mais informações, consulte Testar as consultas antes do upgrade da versão do mecanismo.

Tópicos

- Alterar versões do mecanismo do Athena
- Referência da versão do mecanismo do Athena

Alterar versões do mecanismo do Athena

Esporadicamente, o Athena lança uma nova versão do mecanismo para oferecer melhor performance, funcionalidade e correções de código. Quando uma nona versão está disponível, o Athena notifica você no console. Você pode deixar que o Athena decida quando fazer upgrade ou especificar manualmente uma versão do mecanismo do Athena por grupo de trabalho.

Tópicos

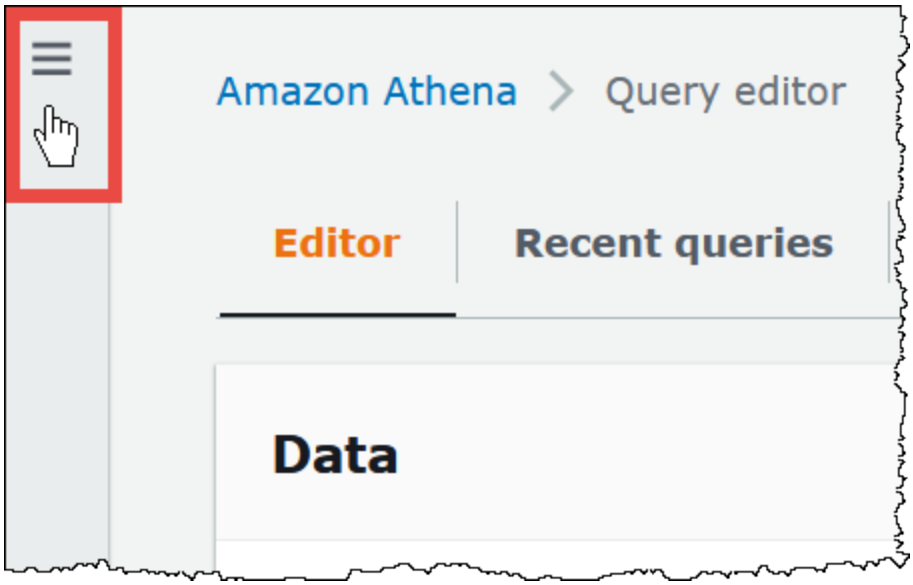
- Localizar a versão do mecanismo de consulta de um grupo de trabalho
- Alteração da versão do mecanismo no console do Athena
- Alterar a versão do mecanismo usando a AWS CLI
- Especificar a versão do mecanismo ao criar um grupo de trabalho
- Testar as consultas antes do upgrade da versão do mecanismo
- Solução de problemas de consultas com falha

Localizar a versão do mecanismo de consulta de um grupo de trabalho

Você pode usar a página Workgroups (Grupos de trabalho) para localizar a versão atual do mecanismo de qualquer grupo de trabalho.

Para localizar a versão atual do mecanismo de qualquer grupo de trabalho

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.



3. No painel de navegação do console do Athena, escolha Workgroups (Grupos de trabalho).
4. Na página Workgroups, encontre o grupo de trabalho desejado. A coluna Query engine version (Versão do mecanismo de consulta) do grupo de trabalho exibe a versão do mecanismo de consulta.

Alteração da versão do mecanismo no console do Athena

Quando uma nova versão do mecanismo está disponível, você pode deixar que o Athena decida quando fazer upgrade do grupo de trabalho ou especificar manualmente a versão do mecanismo do Athena que o grupo de trabalho usa. Se apenas uma versão estiver disponível no momento, não é possível especificar manualmente uma versão diferente.

> **Note**
>
> Para alterar a versão do mecanismo de um grupo de trabalho, você deve ter permissão para executar a ação `athena:ListEngineVersions` no grupo de trabalho. Para ver exemplos de política do IAM, consulte Exemplo de políticas de grupo de trabalho.

Para deixar que o Athena decida quando fazer upgrade do grupo de trabalho

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.
3. No painel de navegação do console, escolha Workgroups (Grupos de trabalho).
4. Na lista de grupos de trabalho, escolha link para o grupo de trabalho que deseja configurar.
5. Selecione a opção Editar.
6. Na seção Query engine version (Versão do mecanismo de consulta), para Update query engine (Atualizar mecanismo de consulta), escolha Automatic (Automático) para permitir que o Athena escolha quando fazer o upgrade de seu grupo de trabalho. Essa é a configuração padrão.
7. Escolha Salvar alterações.

   Na lista de grupos de trabalho, o Query engine update status (Status de atualização do mecanismo de consulta) para o grupo de trabalho indica Automatic (Automático).

Para escolher manualmente a versão de um mecanismo

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.
3. No painel de navegação do console, escolha Workgroups (Grupos de trabalho).

4. Na lista de grupos de trabalho, escolha link para o grupo de trabalho que deseja configurar.
5. Selecione a opção Editar.
6. Na seção Query engine version (Versão do mecanismo de consulta), em Update query engine (Atualizar mecanismo de consulta), escolha Manual para escolher manualmente uma versão do mecanismo).
7. Use a opção Query engine version (Versão do mecanismo de consulta) para escolher a versão do mecanismo que você deseja que o grupo de trabalho use. Se uma versão de mecanismo diferente não estiver disponível, uma versão de mecanismo diferente não poderá ser especificada.
8. Escolha Salvar alterações.

   Na lista de grupos de trabalho, o Query engine update status (Status de atualização do mecanismo de consulta) para o grupo de trabalho indica Manual (Manual).

Alterar a versão do mecanismo usando a AWS CLI

Para alterar a versão do mecanismo usando a AWS CLI, use a sintaxe no seguinte exemplo.

```
aws athena update-work-group --work-group workgroup-name --configuration-updates
 EngineVersion={SelectedEngineVersion='Athena engine version 3'}
```

Especificar a versão do mecanismo ao criar um grupo de trabalho

Ao criar um grupo de trabalho, você pode especificar a versão do mecanismo que ele usará ou deixar que o Athena decida quando fazer upgrade do grupo de trabalho. Se uma nova versão do mecanismo estiver disponível, uma prática recomendada é criar um grupo de trabalho para testar o novo mecanismo antes de fazer upgrade dos outros grupos de trabalho. Para especificar a versão do mecanismo de um grupo de trabalho, você deve ter a permissão `athena:ListEngineVersions` no grupo de trabalho. Para ver exemplos de política do IAM, consulte Exemplo de políticas de grupo de trabalho.

Especificar a versão do mecanismo quando você cria um grupo de trabalho

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.
3. No painel de navegação do console, escolha Workgroups (Grupos de trabalho).

4. Na página Workgroups (Grupos de trabalho), escolha Create workgroup (Criar grupo de trabalho).
5. Na página Create workgroup (Criar grupo de trabalho), na seção Query engine version (Versão do mecanismo de consulta), execute um destes procedimentos:
   - Escolha Automatic (Automático) para permitir que o Athena escolha quando fazer upgrade de seu grupo de trabalho. Essa é a configuração padrão.
   - Selecione Manual para escolher manualmente uma versão de mecanismo diferente, se houver uma disponível.
6. Digite as informações nos outros campos conforme necessário. Para obter informações sobre os outros campos, consulte Criar um grupo de trabalho.
7. Escolha Create workgroup (Criar grupo de trabalho).

Testar as consultas antes do upgrade da versão do mecanismo

Quando o upgrade de um grupo de trabalho é feito para uma nova versão do mecanismo, algumas das suas consultas podem ser interrompidas devido a incompatibilidades. Para garantir que o upgrade da versão do mecanismo seja feito sem problemas, você pode testar as consultas com antecedência.

Para testar as consultas antes do upgrade de uma versão do mecanismo

1. Verifique a versão do mecanismo do grupo de trabalho que você está usando. A versão do mecanismo que você está usando aparece na página Workgroups (Grupos de trabalho) na coluna Query engine version (Versão do mecanismo de consulta) do grupo de trabalho. Para ter mais informações, consulte Localizar a versão do mecanismo de consulta de um grupo de trabalho.
2. Crie um grupo de trabalho de teste que usa a nova versão do mecanismo. Para ter mais informações, consulte Especificar a versão do mecanismo ao criar um grupo de trabalho.
3. Use o novo grupo de trabalho para executar as consultas que você deseja testar.
4. Se uma consulta falhar, use o Referência da versão do mecanismo do Athena para verificar se há alterações inválidas que possam estar afetando a consulta. Algumas alterações podem exigir que você atualize a sintaxe das suas consultas.
5. Se a falha persistir, entre em contato com AWS Support para obter ajuda. No AWS Management Console, escolha Support (Suporte), Support Center (Central de Suporte) ou faça uma pergunta em AWS re:Post usando a etiqueta Amazon Athena.

Solução de problemas de consultas com falha

Se uma consulta falhar após o upgrade de uma versão do mecanismo, use o Referência da versão do mecanismo do Athena para verificar se há alterações inválidas, inclusive que possam afetar a sintaxe das consultas.

Se a falha persistir, entre em contato com AWS Support para obter ajuda. No AWS Management Console, escolha Support (Suporte), Support Center (Central de Suporte) ou faça uma pergunta em AWS re:Post usando a etiqueta Amazon Athena.

Referência da versão do mecanismo do Athena

Esta seção lista as alterações feitas no mecanismo de consulta do Athena.

Tópicos

- Mecanismo Athena versão 3
- Mecanismo do Athena versão 2

Mecanismo Athena versão 3

Para a versão 3 do mecanismo, o Athena também está disponibilizando uma abordagem de integração contínua para o gerenciamento de software de código aberto que melhora a simultaneidade com os projetos Trino e Presto, para que você tenha acesso mais rápido às melhorias da comunidade, integradas e ajustadas no mecanismo do Athena.

Este lançamento do mecanismo Athena versão 3 é compatível com todos os recursos do mecanismo Athena versão 2. Este documento destaca as principais diferenças entre o mecanismo Athena versão 2 e o mecanismo Athena versão 3. Para obter mais informações, consulte o AWSBlog Big Data com o artigo Atualizar para a versão 3 do mecanismo Athena para aumentar o desempenho das consultas e acessar mais recursos de análise.

- Conceitos básicos
- Melhorias e novos recursos
  - Recursos adicionados
  - Funções adicionadas
  - Melhorias de performance
  - Melhorias na confiabilidade

  - Melhorias na sintaxe da consulta
  - Melhorias no formato e no tipo de dados
- Alterações que podem causar interrupções
  - Mudanças na sintaxe de consulta
  - Alterações no processamento de dados
  - Alterações de carimbo de data/hora
- Limitações

Conceitos básicos

Para começar, crie um novo grupo de trabalho do Athena que use o mecanismo Athena versão 3 ou configure um grupo de trabalho existente para usar a versão 3. Qualquer grupo de trabalho do Athena pode fazer o upgrade da versão 2 para a versão 3 do mecanismo sem interromper sua capacidade de enviar consultas.

Para obter mais informações, consulte Alterar versões do mecanismo do Athena.

Melhorias e novos recursos

Os recursos e atualizações listados incluem melhorias do próprio Athena e da funcionalidade incorporada do Trino de código aberto. Para obter uma lista completa de operadores e funções de consulta SQL, consulte a documentação do Trino.

Recursos adicionados

Compatibilidade com o algoritmo de bucketing do Apache Spark

O Athena pode ler buckets gerados pelo algoritmo de hash do Spark. Para especificar que os dados foram originalmente gravados pelo algoritmo de hash do Spark, insira `('bucketing_format'='spark')` na cláusula `TBLPROPERTIES` da sua declaração `CREATE TABLE`. Se essa propriedade não for especificada, o algoritmo de hash do Hive será usado.

```
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')
```

Funções adicionadas

As funções nesta seção são novas no mecanismo Athena versão 3.

Funções agregadas

listagg(x, separator): retorna os valores de entrada concatenados, separados pela string separadora.

```
SELECT listagg(value, ',') WITHIN GROUP (ORDER BY value) csv_value
FROM (VALUES 'a', 'c', 'b') t(value);
```

Funções de array

contains_sequence(x, seq): retorna verdadeiro se a matriz x contiver toda a matriz seq como um subconjunto sequencial (todos os valores na mesma ordem consecutiva).

```
SELECT contains_sequence(ARRAY [1,2,3,4,5,6], ARRAY[1,2]);
```

Funções binárias

murmur3(binary): calcula o hash MurmurHash3 de 128 bits do binário.

```
SELECT murmur3(from_base64('aaaaaa'));
```

Funções de conversão

format_number(number): retorna uma string formatada usando um símbolo de unidade.

```
SELECT format_number(123456); -- '123K'
```

```
SELECT format_number(1000000); -- '1M'
```

Perfis de data e hora

timezone_hour(timestamp): retorna a hora da compensação do fuso horário do carimbo de data/hora.

```
SELECT EXTRACT(TIMEZONE_HOUR FROM TIMESTAMP '2020-05-10 12:34:56 +08:35');
```

timezone_minute(timestamp): retorna o minuto da compensação do fuso horário do carimbo de data/hora.

```
SELECT EXTRACT(TIMEZONE_MINUTE FROM TIMESTAMP '2020-05-10 12:34:56 +08:35');
```

Funções geoespaciais

to_encoded_polyline(Geometry): codifica uma cadeia de linha ou multiponto em uma polilinha.

```
SELECT to_encoded_polyline(ST_GeometryFromText(
   'LINESTRING (-120.2 38.5, -120.95 40.7, -126.453 43.252)'));
```

from_encoded_polyline(varchar): decodifica uma polilinha em uma string de linha.

```
SELECT ST_AsText(from_encoded_polyline('_p~iF~ps|U_ulLnnqC_mqNvxq`@'));
```

to_geojson_geometry(SphericalGeography): retorna a geografia esférica especificada no formato GeoJSON.

```
SELECT to_geojson_geometry(to_spherical_geography(ST_GeometryFromText(
   'LINESTRING (0 0, 1 2, 3 4)')));
```

from_geojson_geometry(varchar): retorna o objeto do tipo geografia esférica da representação GeoJSON, removendo chave/valores que não sejam geométricos. `Feature` e `FeatureCollection` não são compatíveis.

```
SELECT
 from_geojson_geometry(to_geojson_geometry(to_spherical_geography(ST_GeometryFromText(
   'LINESTRING (0 0, 1 2, 3 4)'))));
```

geometry_nearest_points(Geometry, Geometry): retorna os pontos em cada geometria que estão mais próximos uns dos outros. Se a geometria especificada estiver vazia, retornará NULL. Caso contrário, retornará uma linha de dois objetos `Point` que têm a distância mínima de quaisquer dois pontos nas geometrias. O primeiro ponto é do primeiro argumento Geometry (Geometria), o segundo do segundo argumento Geometry (Geometria). Se houver vários pares com a mesma distância mínima, um par será escolhido arbitrariamente.

```
SELECT geometry_nearest_points(ST_GeometryFromText(
   'LINESTRING (50 100, 50 200)'), ST_GeometryFromText(
```

```
   'LINESTRING (10 10, 20 20)'));
```

Funções de definição de resumo

make_set_digest(x): compõe todos os valores de entrada de x em um setdigest.

```
SELECT make_set_digest(value) FROM (VALUES 1, 2, 3) T(value);
```

Funções de string

soundex(char): retorna uma string contendo a representação fonética de char.

```
SELECT name
FROM nation
WHERE SOUNDEX(name) = SOUNDEX('CHYNA'); -- CHINA
```

concat_ws(string0, string1, ..., stringN): retorna a concatenação de `string1, string2, ..., stringN` usando `string0` como separador. Se `string0` for null, o valor retornado será nulo. Todos os valores nulos fornecidos nos argumentos após o separador são ignorados.

```
SELECT concat_ws(',', 'def', 'pqr', 'mno');
```

Funções de janela

GROUPS: adiciona compatibilidade com frames de janela com base em grupos.

```
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);
```

Melhorias de performance

As melhorias na performance do mecanismo Athena versão 3 incluem os seguintes exemplos.

- Recuperação mais rápida de metadados de tabelas do AWS Glue: as consultas que envolvem diversas tabelas terão um tempo de planejamento de consulta reduzido.
- Filtragem dinâmica para RIGHT JOIN: agora, a filtragem dinâmica está habilitada para uniões direitas que têm condições de união de igualdade, como no exemplo a seguir.

  ```
  SELECT *
  ```

```
FROM lineitem RIGHT JOIN tpch.tiny.supplier
ON lineitem.suppkey = supplier.suppkey
WHERE supplier.name = 'abc';
```

- Instruções elaboradas abrangentes: aumento do tamanho padrão do cabeçalho de solicitação/resposta HTTP para 2 MB visando permitir instruções elaboradas abrangentes.
- approx_percentile () - A `approx_percentile` função agora usa `tdigest` em vez de `qdigest` para recuperar valores quantis aproximados das distribuições. Isto resulta em um desempenho superior e em um menor uso de memória. Observe que, como resultado dessa alteração, a função retorna resultados diferentes do que fazia na versão 2 do mecanismo Athena. Para ter mais informações, consulte A função approx_percentile retorna resultados diferentes..

Melhorias na confiabilidade

O uso geral da memória do mecanismo e o rastreamento no mecanismo Athena versão 3 passaram por aprimoramento. Consultas grandes estão menos suscetíveis a falhas causadas por falhas em nós.

Melhorias na sintaxe da consulta

INTERSECT ALL: adição de compatibilidade com `INTERSECT ALL`.

```
SELECT * FROM (VALUES 1, 2, 3, 4) INTERSECT ALL SELECT * FROM (VALUES 3, 4);
```

EXCEPT ALL: adição de compatibilidade com `EXCEPT ALL`.

```
SELECT * FROM (VALUES 1, 2, 3, 4) EXCEPT ALL SELECT * FROM (VALUES 3, 4);
```

RANGE PRECEDING: adição de compatibilidade com `RANGE PRECEDING` em funções de janela.

```
SELECT sum(x) over (order by x range 1 preceding)
FROM (values (1), (1), (2), (2)) t(x);
```

MATCH_RECOGNIZE: adição de compatibilidade com correspondência de padrões de linha, como no exemplo a seguir.

```
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;
```

Melhorias no formato e no tipo de dados

O mecanismo Athena versão 3 recebeu os seguintes aprimoramentos de formato e tipo de dados.

- LZ4 e ZSTD: adição de compatibilidade com leitura de dados compactados de LZ4 e ZSTD em Parquet. Adição de compatibilidade com gravação de dados ORC compactados em ZSTD.
- Tabelas baseadas em links simbólicos: adição de compatibilidade com a criação de tabelas baseadas em links simbólicos em arquivos Avro. Veja a seguir um exemplo.

  ```
  CREATE TABLE test_avro_symlink
  ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
  ...
  INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
  ```

- SphericalGeography: o tipo SphericalGeography fornece suporte nativo para características espaciais representadas em coordenadas geográficas (às vezes chamadas de coordenadas geodésicas, lat/lon ou lon/lat). As coordenadas geográficas são coordenadas esféricas expressas em unidades angulares (graus).

  A função `to_spherical_geography` retorna coordenadas geográficas (esféricas) de coordenadas geométricas (planares), como no exemplo a seguir.

  ```
  SELECT to_spherical_geography(ST_GeometryFromText(
     'LINESTRING (-40.2 28.9, -40.2 31.9, -37.2 31.9)'));
  ```

Alterações que podem causar interrupções

Quando você migrar do mecanismo Athena versão 2 para a versão 3, certas alterações podem afetar o esquema da tabela, a sintaxe ou o uso do tipo de dados. Esta seção lista as mensagens de erro associadas e fornece sugestões de soluções alternativas.

Mudanças na sintaxe de consulta

IGNORE NULLS não pode ser usado com funções de janela sem valor

Mensagem de erro: não é possível especificar a cláusula de tratamento nula para a função `bool_or`.

Causa: o `IGNORE NULLS` agora pode ser usado somente com as funções de valor `first_value`, `last_value`, `nth_value`, `lead` `elag`. Essa alteração foi feita para estar em conformidade com a especificação ANSI SQL.

Solução sugerida: Remover o `IGNORE NULLS` de funções de janela sem valor em cadeias de caracteres de consulta.

A função CONCAT deve ter dois ou mais argumentos

Mensagem de erro: INVALID_FUNCTION_ARGUMENT: There must be two or more concatenation arguments (INVALID_FUNCTION_ARGUMENT: deve haver dois ou mais argumentos de concatenação).

Causa: anteriormente, a função de string `CONCAT` aceitava somente um argumento. Na versão 3 do mecanismo do Athena, a função `CONCAT` requer, no mínimo, dois argumentos.

Solução sugerida: altere as ocorrências de `CONCAT(str)` para `CONCAT(str, '')`.

No mecanismo Athena versão 3, as funções não podem ter mais de 127 argumentos. Para ter mais informações, consulte Excesso de argumentos para chamada de função.

A função approx_percentile retorna resultados diferentes.

A função `approx_percentile` retorna resultados diferentes no mecanismo Athena versão 3 do mecanismo Athena versão 2.

Mensagem de erro: nenhum

Causa: A função `approx_percentile` está sujeita a alterações de versão.

> **Important**
>
> Como as saídas da função `approx_percentile` são aproximações e as aproximações estão sujeitas a alterações de uma versão para outra, você não deve confiar na função para aplicações críticas `approx_percentile`.

Solução sugerida: para aproximar o comportamento `approx_percentile` do motor Athena versão 2, você pode usar um conjunto diferente de funções no motor Athena versão 3. Por exemplo, suponha que você tenha a seguinte consulta no mecanismo Athena versão 2:

```
SELECT approx_percentile(somecol, 2E-1)
```

Para obter aproximadamente a mesma saída na versão 3 do motor Athena, você pode experimentar `qdigest_agg` as funções `value_at_quantile` e, como no exemplo a seguir. Observe que, mesmo com esta solução alternativa, o mesmo comportamento não é garantido.

```
SELECT value_at_quantile(qdigest_agg(somecol, 1), 2E-1)
```

A função geoespacial não é compatível com entrada varibinária

Mensagem de erro: FUNCTION_NOT_FOUND for st_XXX (FUNCTION_NOT_FOUND para st_XXX).

Causa: algumas funções geoespaciais não são mais compatíveis com o tipo de entrada `VARBINARY` legado ou as assinaturas de funções relacionadas ao texto.

Solução sugerida: use funções geoespaciais para converter os tipos de entrada em tipos compatíveis. Os tipos de entrada compatíveis são indicados na mensagem de erro.

Em cláusulas GROUP BY, as colunas aninhadas devem ter aspas duplas

Mensagem de erro: “*`column_name`*” “*`nested_column`*” deve ser uma expressão agregada ou aparecer na cláusula GROUP BY

Causa: o mecanismo Athena versão 3 exige que os nomes das colunas aninhadas nas cláusulas `GROUP BY` tenham aspas duplas. Por exemplo, a consulta a seguir produz o erro porque, na cláusula `GROUP BY`, `user.name` não está entre aspas duplas.

```
SELECT "user"."name" FROM dataset
GROUP BY user.name
```

Solução sugerida: coloque aspas duplas ao redor dos nomes das colunas aninhadas nas cláusulas `GROUP BY`, como no exemplo a seguir.

```
SELECT "user"."name" FROM dataset
GROUP BY "user"."name"
```

Erro de FilterNode inesperado ao usar OPTIMIZE em uma tabela Iceberg

Mensagem de erro: FilterNode inesperado encontrado no plano; provavelmente o conector não conseguiu lidar com a expressão WHERE fornecida.

Causa: a instrução `OPTIMIZE` que foi executada na tabela Iceberg usou uma cláusula `WHERE` que incluía uma coluna sem partição em sua expressão de filtro.

Solução sugerida: a instrução `OPTIMIZE` suporta a filtragem somente por partições. Ao executar `OPTIMIZE` em tabelas particionadas, inclua somente colunas de partição na cláusula `WHERE`. Se você executar `OPTIMIZE` em uma tabela não particionada, não especifique uma cláusula `WHERE`.

Ordem dos argumentos na função Log()

Na versão 2 do mecanismo do Athena, a ordem dos argumentos para a função `log()` era `log(value, base)`. Na versão 3 do mecanismo do Athena, isso foi alterado para `log(base, value)` por questão de conformidade com os padrões SQL.

A função Minute() não é compatível com intervalos de ano a mês

Mensagem de erro: Unexpected parameters (interval year to month) for function minute. (Parâmetros inesperados [intervalo de ano a mês] para a função “minute” [minutos]). Expected: minute(timestamp with time zone) , minute(time with time zone) , minute(timestamp) , minute(time) , minute(interval day to second).

Causa: no mecanismo Athena versão 3, as verificações estão mais precisas para `EXTRACT` de acordo com a especificação ANSI SQL.

Solução sugerida: atualize as consultas para garantir que os tipos correspondam às assinaturas de função sugeridas.

Expressões ORDER BY devem aparecer na lista SELECT

Mensagem de erro: For SELECT DISTINCT, ORDER BY expressions must appear in SELECT list (Para SELECT DISTINCT, expressões ORDER BY devem aparecer na lista SELECT).

Causa: o alias de tabela incorreto é usado em uma cláusula `SELECT`.

Solução sugerida: verifique se todas as colunas na expressão `ORDER BY` têm referências adequadas na cláusula `SELECT DISTINCT`.

Falha na consulta ao comparar várias colunas retornadas de uma subconsulta

Exemplo de mensagem de erro: a expressão do valor e o resultado da subconsulta devem ser do mesmo tipo: row(varchar, varchar) vs. row(row(varchar, varchar))

Causa: devido a uma atualização de sintaxe no mecanismo Athena versão 3, esse erro ocorre quando uma consulta tenta comparar vários valores retornados de uma subconsulta, e a instrução `SELECT` da subconsulta coloca a lista de colunas entre parênteses, como no exemplo a seguir.

```
SELECT *
FROM table1
WHERE (t1_col1, t1_col2)
IN (SELECT (t2_col1, t2_col2) FROM table2)
```

Solução: no mecanismo Athena versão 3, remova os parênteses ao redor da lista de colunas na instrução `SELECT` da subconsulta, como no exemplo de consulta atualizado a seguir.

```
SELECT *
FROM table1
WHERE (t1_col1, t1_col2)
IN (SELECT t2_col1, t2_col2 FROM table2)
```

SKIP é uma palavra reservada para consultas DML

Agora a palavra `SKIP` é reservada para consultas DML, como `SELECT`. Para usar `SKIP` como identificador em uma consulta DML, coloque-a entre aspas duplas.

Para obter mais informações sobre as palavras reservadas no Athena, consulte Palavras-chave reservadas.

Cláusulas SYSTEM_TIME e SYSTEM_VERSION estão obsoletas para passagem de tempo

Mensagem de erro: mismatched input ‘SYSTEM_TIME’ (entrada “SYSTEM_TIME” incompatível). Expecting: ‘TIMESTAMP’, ‘VERSION’ (Esperado: “TIMESTAMP”, “VERSION”).

Causa: na versão 2 do mecanismo do Athena, as tabelas do Iceberg usavam as cláusulas `FOR SYSTEM_TIME AS OF` e `FOR SYSTEM_VERSION AS OF` como o carimbo de data/hora e a versão de passagem de tempo. A versão 3 do mecanismo do Athena usa as cláusulas `FOR TIMESTAMP AS OF` e `FOR VERSION AS OF`.

Solução sugerida: atualize a consulta SQL para usar as cláusulas `TIMESTAMP AS OF` e `VERSION AS OF` para operações de passagem de tempo, como nos exemplos a seguir.

Passagem de tempo por data e hora:

```
SELECT * FROM TABLE FOR TIMESTAMP AS OF (current_timestamp - interval '1' day)
```

Passagem de tempo por versão:

```
SELECT * FROM TABLE FOR VERSION AS OF 949530903748831860
```

Excesso de argumentos para um construtor de matrizes

Mensagem de erro: TOO_MANY_ARGUMENTS: excesso de argumentos para o construtor de matrizes.

Causa: o número máximo de elementos em um construtor de matrizes agora está definido como 254.

Solução sugerida: divida os elementos em várias matrizes com até 254 elementos cada e use a função `CONCAT` para concatenar as matrizes, como no exemplo a seguir.

```
CONCAT(
ARRAY[x1,x2,x3...x254],
ARRAY[y1,y2,y3...y254],
...
)
```

Identificador delimitado com comprimento zero não permitido

Mensagem de erro: Zero-length delimited identifier not allowed (Identificadores delimitados por comprimento zero não são permitidos).

Causa: uma consulta usou uma string vazia como alias de coluna.

Solução sugerida: atualize a consulta para usar um alias que não esteja vazio para a coluna.

Alterações no processamento de dados

Validação de bucket

Mensagem de erro: HIVE_INVALID_BUCKET_FILES: a tabela do Hive está corrompida..

Causa: a tabela pode ter sido corrompida. O mecanismo Athena versão 3 permite validação adicional de tabelas em buckets para assegurar que as consultas estejam corretas e evitar falhas inesperadas no runtime.

Solução sugerida: recrie a tabela usando o mecanismo Athena versão 3.

Agora a conversão de um struct para JSON retorna nomes de campos

Agora, ao converter um `struct` para JSON em uma consulta `SELECT` no mecanismo do Athena versão 3, a conversão retorna os nomes dos campos e os valores (por exemplo, `"useragent":null` em vez de apenas os valores (por exemplo, `null`).

Alteração da imposição de segurança em nível de coluna da tabela do Iceberg

Mensagem de erro: Access Denied: Cannot select from columns (Acesso negado: não é possível realizar a seleção entre as colunas).

Causa: a tabela do Iceberg foi criada exteriormente ao Athena e usa uma versão do SDK do Apache Iceberg anterior à versão 0.13.0. Como as versões anteriores do SDK não preenchem as colunas no AWS Glue, o Lake Formation não é capaz de determinar as colunas autorizadas para o acesso.

Solução sugerida: realize uma atualização usando a instrução ALTER TABLE SET PROPERTIES do Athena ou use o SDK mais recente do Iceberg para corrigir a tabela e atualizar as informações da coluna no AWS Glue.

Agora, os nulos nos tipos de dados List (Lista) são propagados para UDFs

Mensagem de erro: Null Pointer Exception.

Causa: esse problema pode afetar você se você usar o conector UDF e tiver implementado uma função Lambda definida pelo usuário.

O mecanismo Athena versão 2 filtrou os nulos nos tipos de dados List (Lista) que foram transmitidos para uma função definida pelo usuário. No mecanismo Athena versão 3, agora os nulos são preservados e transmitidos para a UDF. Isso pode causar uma exceção de ponteiro nulo se a UDF tentar desfazer a referência ao elemento nulo sem verificação.

Por exemplo, se você tiver os dados `[null, 1, null, 2, 3, 4]` em uma fonte de dados de origem como o DynamoDB, o seguinte será transmitido para a função Lambda definida pelo usuário:

Mecanismo Athena versão 2: `[1,2,3,4]`

Mecanismo Athena versão 3: `[null, 1, null, 2, 3, 4]`

Solução sugerida: certifique-se de que sua função do Lambda definida pelo usuário trate os elementos nulos nos tipos de dados de lista.

Subsequências de matrizes de caracteres não contêm mais espaços preenchidos

Mensagem de erro: nenhuma mensagem de erro é gerada, mas a string retornada não contém mais os espaços preenchidos. Por exemplo, agora `substr(char[20],1,100)` retorna uma string com comprimento 20 em vez de 100.

Solução sugerida: nenhuma ação é necessária.

Coerção de tipo de coluna decimal não compatível

Mensagens de erro: HIVE_CURSOR_ERROR: Failed to read Parquet file: s3://DOC-EXAMPLE-BUCKET/*caminho*/*nome_arquivo*.parquet ou Unsupported column type (varchar) for Parquet column ([*nome_coluna*]

Causa: o mecanismo Athena versão 2 teve sucesso algumas vezes (mas falhou frequentemente) ao tentar conversões de tipo de dados de `varchar` para decimal. Como a versão 3 do mecanismo Athena tem validação de tipo, que verifica se o tipo é compatível antes de tentar ler o valor, essas tentativas de coerção agora sempre falham.

Solução sugerida: tanto para o mecanismo Athena versão 2 quanto para o mecanismo Athena versão 3, modifique o esquema no AWS Glue para usar um tipo de dado numérico em vez `varchar` para colunas decimais em arquivos do Parquet. Ou recrawle os dados e garanta que o novo tipo de dados da coluna seja do tipo decimal, ou recrie manualmente a tabela no Athena e use a sintaxe `decimal(`*`precision`*`, `*`scale`*`)` para especificar um tipo de dados decimal para a coluna.

Valores NaN de ponto flutuante ou precisão dupla não podem mais ser convertidos para bigint

Mensagem de erro: INVALID_CAST_ARGUMENT: Cannot cast real/double NaN to bigint

Causa: na versão 3 do mecanismo do Athena, `NaN` não pode mais ser convertido para 0 como `bigint`.

Solução sugerida: certifique-se de que os valores `NaN` não estejam presentes nas colunas `float` e `double` ao converter para `bigint`.

alteração do tipo de retorno da função uuid()

O problema a seguir afeta tanto as tabelas como as visualizações.

Mensagem de erro: Unsupported Hive type: uuid

Causa: no mecanismo Athena versão 2, a função `uuid()` retornou uma string, mas no mecanismo Athena versão 3, ela retorna um pseudo UUID gerado aleatoriamente (tipo 4). Como o tipo de

dados da coluna UUID não é compatível com o Athena, a função `uuid()` não pode mais ser usada diretamente em consultas CTAS para gerar colunas UUID no mecanismo Athena versão 3.

Por exemplo, a seguinte instrução `CREATE TABLE` é concluída com êxito no mecanismo do Athena versão 2, mas retorna NOT_SUPPORTED: Unsupported Hive type: uuid no mecanismo do Athena versão 3:

```
CREATE TABLE uuid_table AS
   SELECT uuid() AS myuuid
```

Da mesma forma, a seguinte instrução `CREATE VIEW` é concluída com êxito no mecanismo do Athena versão 2, mas retorna NOT_SUPPORTED: Unsupported Hive type: uuid no mecanismo do Athena versão 3:

```
CREATE VIEW uuid_view AS
   SELECT uuid() AS myuuid
```

Quando uma visualização assim criada no mecanismo do Athena versão 2 é consultada no mecanismo do Athena versão 3, ocorre um erro como este:

VIEW_IS_STALE: linha 1:15: a exibição 'awsdatacatalog.mydatabase.uuid_view' está esmaecida ou em estado inválido: a coluna [myuuid] do tipo uuid projetada da exibição da consulta na posição 0 não pode ser forçada na coluna [myuuid] do tipo varchar armazenada na definição da exibição

Solução sugerida: ao criar ou visualizar a tabela, use a função `cast()` para converter a saída de `uuid()` para `varchar`, como no seguinte exemplo:

```
CREATE TABLE uuid_table AS
   SELECT CAST(uuid() AS VARCHAR) AS myuuid
```

```
CREATE VIEW uuid_view AS
   SELECT CAST(uuid() AS VARCHAR) AS myuuid
```

Problemas de coerção de CHAR e VARCHAR

Use as soluções alternativas desta seção caso encontre problemas de coerção de `varchar` e `char` no mecanismo do Athena versão 3. Se não conseguir usar essas soluções alternativas, entre em contato com o AWS Support.

Falha na função CONCAT com entradas CHAR e VARCHAR mistas

Problema: a consulta a seguir tem êxito no mecanismo do Athena versão 2.

```
SELECT concat(CAST('abc' AS VARCHAR(20)), '12', CAST('a' AS CHAR(1)))
```

Porém, no mecanismo do Athena versão 3, a mesma consulta falha com o seguinte:

Mensagem de erro: FUNCTION_NOT_FOUND: line 1:8: Unexpected parameters (varchar(20), varchar(2), char(1)) for function concat. Esperado: concat(char(x), char(y)), concat(array(E), E) E, concat(E, array(E)) E, concat(array(E)) E, concat(varchar), concat(varbinary)

Solução sugerida: ao usar a função `concat`, converta para `char` ou `varchar`, mas não para uma mistura de ambas.

Falha de concatenação de || do SQL com entradas CHAR e VARCHAR

No mecanismo do Athena versão 3, o operador de concatenação de barras verticais duplas `||` requer `varchar` como entradas. As entradas não podem ser uma combinação de tipos `varchar` e `char`.

Mensagem de erro: TYPE_NOT_FOUND: line 1:26: Unknown type: char(65537)

Causa: uma consulta que usa `||` para concatenar um `char` e um `varchar` pode produzir o erro, como no exemplo a seguir.

```
SELECT CAST('a' AS CHAR) || CAST('b' AS VARCHAR)
```

Solução sugerida: concatene `varchar` com `varchar`, como no exemplo a seguir.

```
SELECT CAST('a' AS VARCHAR) || CAST('b' AS VARCHAR)
```

Falha na consulta CHAR e VARCHAR UNION

Mensagem de erro: NOT_SUPPORTED: Unsupported Hive type: char(65536). Tipos de CHAR compatíveis: CHAR (<=255)

Causa: uma consulta que tenta combinar `char` e `varchar`, como no seguinte exemplo:

```
CREATE TABLE t1 (c1) AS SELECT CAST('a' as CHAR) as c1 UNION ALL SELECT CAST('b' AS
 VARCHAR) AS c1
```

Solução sugerida: na consulta de exemplo, converta `'a'` para `varchar` em vez de `char`.

Espaços vazios indesejados após a coerção de CHAR ou VARCHAR

Na versão 3 do mecanismo Athena, quando os dados `char(X)` e `varchar` são forçados para um único tipo formando uma matriz ou coluna única, `char(65535)` é o tipo de destino, e cada campo contém muitos espaços indesejados à direita.

Causa: o mecanismo Athena versão 3 força `varchar` e `char(X)` para `char(65535)` e preenche os dados com espaços à direita.

Solução sugerida: converta cada campo explicitamente para `varchar`.

Alterações de carimbo de data/hora

Mudança no comportamento da transmissão de um carimbo de data/hora com fuso horário para varchar

No mecanismo Athena versão 2, a conversão de um `Timestamp` com fuso horário para `varchar` causava a mudança de alguns literais de fuso horário (p. ex., `US/Eastern` mudava para `America/New_York`). Esse comportamento não ocorre no mecanismo do Athena versão 3.

O estouro de carimbo de data gera um erro

Mensagem de erro: Millis overflow: XXX (Estouro de milissegundos: XXX)

Causa: como as datas ISO 8601 não foram verificadas quanto ao estouro na versão 2 do mecanismo Athena, algumas datas produziram um carimbo de data/hora negativo. O mecanismo Athena versão 3 verifica esse estouro e gera uma exceção.

Solução sugerida: certifique-se de que o carimbo de data/hora esteja dentro do intervalo.

Não há compatibilidade com fusos horários políticos com TIME

Mensagem de erro: INVALID LITERAL

Causa: consultas como `SELECT TIME '13:21:32.424 America/Los_Angeles'`.

Solução sugerida: evite usar fusos horários políticos com `TIME`.

Divergência de precisão nas colunas Timestamp causa erro de serialização

Mensagem de erro: SERIALIZATION_ERROR: Could not serialize column '*COLUMNZ*' of type 'timestamp(3)' at position *X*:*Y* (SERIALIZATION_ERROR: não foi possível serializar a coluna "COLUMNZ" do tipo "timestamp(3)" na posição X:Y).

COLUMNZ é o nome de saída da coluna que causa o problema. Os números *X*:*Y* indicam a posição da coluna na saída.

Causa: o mecanismo Athena versão 3 verifica se a precisão dos registros de data e hora nos dados é igual à precisão especificada para o tipo de dados da coluna na especificação da tabela. No momento, essa precisão é sempre 3. Se os dados tiverem uma precisão maior do que isso, as consultas falharão com o erro observado.

Solução sugerida: verifique seus dados para garantir que os timestamps tenham a precisão de milissegundos.

Precisão incorreta do carimbo de data e hora nas consultas UNLOAD e CTAS para tabelas Iceberg

Mensagem de erro: precisão incorreta do carimbo de data e hora para timestamp(6); a precisão configurada é MILLISECONDS

Causa: o mecanismo Athena versão 3 verifica se a precisão dos registros de data e hora nos dados é igual à precisão especificada para o tipo de dados da coluna na especificação da tabela. No momento, essa precisão é sempre 3. Se os dados tiverem uma precisão maior do que isso (por exemplo, microssegundos em vez de milissegundos), as consultas poderão falhar com o erro observado.

Solução: para resolver esse problema, primeiro CAST a precisão do timestamp para 6, como no exemplo de CTAS a seguir, que cria uma tabela Iceberg. Observe que a precisão deve ser especificada como 6 em vez de 3 para evitar o erro Timestamp precision (3) not supported for 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
```

Então, como o Athena não é compatível com o timestamp 6, converta o valor novamente para timestamp (por exemplo, em uma exibição). O exemplo a seguir cria uma visualização da tabela `my_iceberg_ctas`.

```
CREATE OR REPLACE VIEW my_iceberg_ctas_view AS
SELECT cast(dt AS timestamp) AS dt
FROM my_iceberg_ctas
```

Ler o tipo Long como Timestamp ou vice-versa em arquivos ORC agora causa um erro de arquivo ORC malformado.

Mensagem de erro: Error opening Hive split ‘FILE (SPLIT POSITION)’ Malformed ORC file. (Erro ao abrir o arquivo ORC malformado da divisão Hive ‘FILE (SPLIT POSITION)’). Cannot read SQL type timestamp from ORC stream .long_type of type LONG

Causa: agora, o mecanismo Athena versão 3 rejeita a coerção implícita do tipo de dados `Long` para `Timestamp` ou de `Timestamp` para `Long`. Anteriormente, os valores `Long` eram convertidos implicitamente em carimbo de data/hora como se fossem milissegundos de época.

Solução sugerida: use a função `from_unixtime` para converter explicitamente a coluna ou use a função `from_unixtime` para criar uma coluna adicional para consultas futuras.

Não há compatibilidade com tempo e intervalo de ano a mês

Mensagem de erro: TYPE MISMATCH

Causa: o mecanismo Athena versão 3 não é compatível com tempo e intervalo de ano a mês (p. ex., `SELECT TIME '01:00' + INTERVAL '3' MONTH`).

Estouro de carimbo de data/hora para o formato Parquet int96

Mensagem de erro: Invalid timeOfDayNanos (timeOfDayNanos inválido).

Causa: um estouro de carimbo de data/hora para o formato Parquet `int96`.

Solução sugerida: identifique os arquivos específicos que apresentam o problema. Em seguida, gere o arquivo de dados novamente com uma biblioteca Parquet atualizada e conhecida ou use o Athena CTAS. Se o problema persistir, entre em contato com o suporte do Athena e nos informe como os arquivos de dados são gerados.

Espaço necessário entre os valores de data e hora ao converter de string para carimbo de data e hora

Mensagem de erro: INVALID_CAST_ARGUMENT: o valor não pode ser convertido para carimbo de data e hora.

Causa: o mecanismo Athena versão 3 não aceita mais um hífen como separador válido entre valores de data e hora na string de entrada para `cast`. Por exemplo, a consulta a seguir funciona no mecanismo Athena versão 2, mas não no mecanismo Athena versão 3:

```
SELECT CAST('2021-06-06-23:38:46' AS timestamp) AS this_time
```

Solução sugerida: no mecanismo Athena versão 3, substitua o hífen entre a data e a hora por um espaço, como no exemplo a seguir.

```
SELECT CAST('2021-06-06 23:38:46' AS timestamp) AS this_time
```

alteração do valor de retorno do carimbo de data e hora to_iso8601()

Mensagem de erro: nenhuma

Causa: no mecanismo Athena versão 2, a função `to_iso8601` retorna um carimbo de data e hora com fuso horário, mesmo que o valor passado para a função não inclua o fuso horário. No mecanismo Athena versão 3, a função `to_iso8601` retorna um carimbo de data e hora com fuso horário somente quando o argumento passado inclui o fuso horário.

Por exemplo, a consulta a seguir passa a data atual para a função `to_iso8601` duas vezes: primeiro como carimbo de data e hora com fuso horário e depois como um carimbo de data e hora.

```
SELECT TO_ISO8601(CAST(CURRENT_DATE AS TIMESTAMP WITH TIME ZONE)),
 TO_ISO8601(CAST(CURRENT_DATE AS TIMESTAMP))
```

A saída a seguir mostra o resultado da consulta em cada mecanismo.

Mecanismo Athena versão 2:

#	_col0	_col1
1	2023-02-24T00:00:00.000Z	2023-02-24T00:00:00.000Z

Mecanismo Athena versão 3:

#	_col0	_col1
1	2023-02-24T00:00:00.000Z	2023-02-24T00:00:00.000

Solução sugerida: para replicar o comportamento anterior, você pode passar o valor do carimbo de data e hora para a função `with_timezone` antes de passá-lo para `to_iso8601`, como no exemplo a seguir:

```
SELECT to_iso8601(with_timezone(TIMESTAMP '2023-01-01 00:00:00.000', 'UTC'))
```

Resultado

#	_col0
1	2023-01-01T00:00:00.000Z

o primeiro parâmetro at_timezone() deve especificar uma data

Problema: no mecanismo Athena versão 3, a função `at_timezone` não pode ter um valor `time_with_timezone` como o primeiro parâmetro.

Causa: sem informações de data, não é possível determinar se o valor passado está no horário de verão ou no horário padrão. Por exemplo, `at_timezone('12:00:00 UTC', 'America/Los_Angeles')` é ambíguo, pois não há como determinar se o valor passado está no Horário de Verão do Pacífico (PDT) ou no Horário Padrão do Pacífico (PST).

Limitações

O mecanismo Athena versão 3 tem as seguintes limitações.

- Performance de consulta: muitas consultas são executadas mais rapidamente no mecanismo Athena versão 3, mas alguns planos de consulta podem ser diferentes do mecanismo Athena versão 2. Como resultado, algumas consultas podem diferir em latência ou custo.
- Conectores Trino e Presto: não há compatibilidade com conectores Trino nem Presto. Use a consulta federada do Amazon Athena para conectar origens de dados. Para ter mais informações, consulte Usar a consulta federada do Amazon Athena.

- Execução tolerante a falhas: não há compatibilidade com a execução tolerante a falhas do Trino (Trino Tardigrade).
- Limite de parâmetros de função: as funções não podem ter mais de 127 parâmetros. Para ter mais informações, consulte Excesso de argumentos para chamada de função.

Os limites a seguir foram introduzidos no mecanismo do Athena versão 2 para garantir que as consultas não falhem devido a limitações de recursos. Esses limites não são configuráveis pelos usuários.

- Número de elementos do resultado: o número de elementos do resultado n é restrito a no máximo 10.000 para as seguintes funções: `min(col, n)`, `max(col, n)`, `min_by(col1, col2, n)` e `max_by(col1, col2, n)`.
- GROUPING SETS: o número máximo de fatias em um conjunto de agrupamento é 2.048.
- Comprimento máximo de linha de arquivo de texto: o comprimento máximo de linha padrão para arquivos de texto é de 200 MB.
- Tamanho máximo do resultado da função de sequência: o tamanho máximo do resultado de uma função de sequência é 50.000 entradas. Por exemplo, `SELECT sequence(0,45000,1)` é bem-sucedido, mas `SELECT sequence(0,55000,1)` falha com a mensagem de erro: The result of the sequence function must not have more than 50000 entries (O resultado da função de sequência não deve ter mais de 50.000 entradas). Esse limite se aplica a todos os tipos de entrada das funções de sequência, incluindo carimbos de data/hora.

Mecanismo do Athena versão 2

A versão 2 do mecanismo do Athena introduz as alterações a seguir.

- Melhorias e novos recursos
  - Melhorias de agrupamento, junção e subconsulta
  - Melhorias de tipos de dados
  - Funções adicionadas
  - Melhorias de performance
  - Melhorias relacionadas ao JSON
- Alterações que podem causar interrupções
  - Correções de erros

- Alterações nas funções geoespaciais
- Compatibilidade com ANSI SQL
- Funções substituídas
- Limites

Melhorias e novos recursos

- EXPLAIN e EXPLAIN ANALYZE: agora é possível usar a instrução `EXPLAIN` no Athena para visualizar o plano de execução das suas consultas SQL. Use o `EXPLAIN ANALYZE` para visualizar o plano de execução distribuído para suas consultas SQL e o custo de cada operação. Para obter mais informações, consulte Usar EXPLAIN e EXPLAIN ANALYZE no Athena.
- Consultas federadas: estão disponíveis no mecanismo do Athena versão 2. Para obter mais informações, consulte Usar a consulta federada do Amazon Athena.
- Funções geoespaciais: mais de 25 funções geoespaciais foram adicionadas. Para obter mais informações, consulte Novas funções geoespaciais no mecanismo do Athena versão 2.
- Esquema aninhado: suporte incluído para leitura de esquema aninhado, o que reduz os custos.
- Instruções preparadas: Use instruções preparadas para a execução repetida da mesma consulta com diferentes parâmetros de consulta. Uma instrução preparada contém parâmetros de espaço reservado cujos valores você passa no runtime. Instruções preparadas ajudam a evitar ataques com injeções de SQL. Para obter mais informações, consulte Utilizar consultas parametrizadas.
- Suporte à evolução do esquema: suporte incluído para evolução do esquema de dados no formato Parquet.
  - Suporte incluído para leitura de colunas do tipo array, mapa ou linha de partições em que o esquema de partição é diferente do esquema de tabela. Isso pode ocorrer quando o esquema de tabela é atualizado após a criação da partição. Os tipos de coluna alterados devem ser compatíveis. Para os tipos de linha, é possível adicionar ou descartar campos finais, mas os campos correspondentes (por ordinal) devem ter o mesmo nome.
  - Os arquivos ORC agora podem ter colunas struct com campos ausentes. Isso permite que o esquema de tabela seja alterado sem reescrever os arquivos ORC.
  - As colunas struct ORC agora são mapeadas por nome, em vez de por ordinal. Isso processa corretamente os campos struct ausentes ou extras no arquivo ORC.
- SQL OFFSET: a cláusula SQL `OFFSET` agora é suportada nas instruções `SELECT`. Para obter mais informações, consulte SELECT.

- Instrução UNLOAD: é possível usar a instrução `UNLOAD` para gravar a saída de uma consulta `SELECT` nos formatos PARQUET, ORC, AVRO e JSON. Para obter mais informações, consulte UNLOAD.

Melhorias de agrupamento, junção e subconsulta

- Agrupamento complexo: suporte incluído para operações de agrupamento complexas.
- Subconsultas correlacionadas: suporte incluído para subconsultas correlacionadas nos predicados `IN` e para subconsultas correlacionadas que exigem coerções.
- CROSS JOIN: suporte incluído para `CROSS JOIN` em tabelas derivadas de `LATERAL`.
- GROUPING SETS: suporte incluído para cláusulas `ORDER BY` em agregações para consultas que usam `GROUPING SETS`.
- Expressões do Lambda: suporte incluído para cancelar a referência de campos de linha em expressões do Lambda.
- Valores nulos em semijunções: suporte incluído para valores nulos no lado esquerdo de uma semijunção (ou seja, um predicado `IN` com subconsultas).
- Junções espaciais: suporte incluído para junções espaciais de transmissão e à esquerda.
- Vazamento para disco: para operações `INNER JOIN` e `LEFT JOIN` de uso intensivo de memória, o Athena descarrega os resultados da operação intermediária no disco. Esse procedimento permite executar consultas que exigem grandes quantidades de memória.

Melhorias de tipos de dados

- INT para INTEGER: suporte incluído para `INT` como um alias do tipo de dados `INTEGER`.
- Tipos de INTERVAL: suporte incluído para conversão nos tipos de `INTERVAL`.
- IPADDRESS: adição de um novo tipo de `IPADDRESS` para representar endereços IP em consultas DML. Suporte incluído para conversão entre os tipos `VARBINARY` e `IPADDRESS`. O tipo `IPADDRESS` não é reconhecido em consultas DDL.
- IS DISTINCT FROM: suporte incluído de `IS DISTINCT FROM` para os tipos `JSON` e `IPADDRESS`.
- Verificações de igualdade de nulos: agora são permitidas as verificações de igualdade de valores nulos nas estruturas de dados `ARRAY`, `MAP` e `ROW`. Por exemplo, a expressão `ARRAY ['1', '3', null] = ARRAY ['1', '2', null]` retorna `false`. Antes, um elemento nulo retornava a mensagem de erro comparison not supported (comparação não permitida).

- Coerção de tipo de linha: agora é permitida a coerção entre tipos de linha, seja qual for os nomes dos campos. Antes, a coerção de um tipo de linha para outro era possível apenas se o nome do campo no tipo de origem correspondesse ao tipo de destino, ou quando o tipo de destino tinha um nome de campo anônimo.
- Subtração de tempo: subtração implementada para todos os tipos `TIME` e `TIMESTAMP`.
- Unicode: suporte incluído para sequências Unicode escapadas em literais de string.
- Concatenação de VARBINARY: suporte incluído para concatenação de valores `VARBINARY`.

  Funções de valor de janela: as funções de valor da janela agora suportam `IGNORE NULLS` e `RESPECT NULLS`.

Tipos de entrada adicionais para funções

As funções abaixo agora aceitam tipos de entrada adicionais. Para obter mais informações sobre cada função, acesse o link correspondente para a documentação do Presto.

- approx_distinct(): a função approx_distinct() agora aceita os seguintes tipos: `INTEGER`, `SMALLINT`, `TINYINT`, `DECIMAL`, `REAL`, `DATE`, `TIMESTAMP`, `TIMESTAMP WITH TIME ZONE`, `TIME`, `TIME WITH TIME ZONE`, `IPADDRESS` e `CHAR`.
- avg(), sum(): as funções de agregação avg() e sum() agora aceitam o tipo de dado `INTERVAL`.
- lpad(), rpad(): as funções lpad e rpad agora funcionam com as entradas `VARBINARY`.
- min(), max(): as funções de agregação min() e max() agora permitem tipos de entrada desconhecidos no momento da análise da consulta para que você possa usar as funções com `NULL` literais.
- regexp_replace(): a variante da função regexp_replace() foi adicionada para executar uma função do Lambda em cada substituição.
- sequence(): as variantes `DATE` foram adicionadas à função sequence(), incluindo a variante com um incremento implícito de um dia.
- ST_Area(): a função geoespacial ST_Area() agora aceita todos os tipos de geometria.
- substr(): a função substr agora funciona com as entradas `VARBINARY`.
- zip_with(): arrays de comprimento divergente agora podem ser usados com zip_with(). As posições ausentes são preenchidas com nulo. Antes, um erro era gerado quando arrays de comprimentos diferentes eram especificados. Essa alteração pode dificultar a distinção entre os valores que inicialmente eram nulos dos valores que foram adicionados para preencher os arrays para ficarem com o mesmo comprimento.

Funções adicionadas

A lista a seguir contém as funções novas a partir do mecanismo Athena versão 2. A lista não inclui as funções geoespaciais. Para obter uma lista das funções geoespaciais, consulte Novas funções geoespaciais no mecanismo do Athena versão 2.

Para obter mais informações sobre cada função, acesse o link correspondente para a documentação do Presto.

Funções agregadas

reduce_agg()

Funções e operadores de matriz

array_sort(): variante dessa função adicionada para assumir uma função do Lambda como um comparador.

ngrams()

Funções e operadores binários

from_big_endian_32()

from_ieee754_32()

from_ieee754_64()

hmac_md5()

hmac_sha1()

hmac_sha256()

hmac_sha512()

spooky_hash_v2_32()

spooky_hash_v2_64()

to_big_endian_32()

to_ieee754_32()

to_ieee754_64()

Funções e operadores de data e hora

millisecond()

parse_duration()

to_milliseconds()

Funções e operadores de mapa

multimap_from_entries()

Funções e operadores matemáticos

inverse_normal_cdf()

wilson_interval_lower()

wilson_interval_upper()

Funções de resumo de quantil

funções de resumo de quantil e o tipo de resumo de quantil `qdigest` foram adicionados.

Funções e operadores de string

hamming_distance()

split_to_multimap()

Melhorias de performance

A performance dos recursos abaixo foi aprimorada no mecanismo do Athena versão 2.

Performance da consulta

- Performance de junção de transmissão: aprimorada a performance da junção de transmissão com a aplicação de redução dinâmica de partição no nó de processamento.
- Tabelas em bucket: performance aprimorada de gravação nas tabelas em bucket quando os dados que estão sendo gravados já foram devidamente particionados (por exemplo, quando a saída é de uma junção em bucket).

- DISTINCT: performance aprimorada de algumas consultas que usam `DISTINCT`.

  Filtragem dinâmica e redução de partições: as melhorias aumentam a performance e reduzem a quantidade de dados verificados nas consultas.
- Operações de filtro e projeção: as operações de filtro e projeção agora sempre são processadas por colunas, se possível. O mecanismo aproveita automaticamente as codificações do dicionário quando são eficazes.
- Trocas de coleta: performance aprimorada das consultas com trocas de coleta.
- Agregações globais: performance aprimorada de algumas consultas que executam agregações globais filtradas.
- GROUPING SETS, CUBE, ROLLUP: performance aprimorada das consultas que envolvem `GROUPING SETS`, `CUBE` ou `ROLLUP`, que você pode usar para agregar vários conjuntos de colunas em uma única consulta.
- Filtros altamente seletivos: performance aprimorada das consultas com filtros altamente seletivos.
- Operações JOIN e AGGREGATE: a performance das operações `JOIN` e `AGGREGATE` foi aprimorada.
- LIKE: a performance foi aprimorada das consultas que usam os predicados `LIKE` nas colunas das tabelas `information_schema`.
- ORDER BY e LIMIT: os planos, a performance e o uso de memória foram aprimorados para as consultas que envolvem `ORDER BY` e `LIMIT` para evitar trocas desnecessárias de dados.
- ORDER BY: as operações `ORDER BY` agora são distribuídas por padrão, o que permite usar cláusulas `ORDER BY` maiores.
- Conversões do tipo ROW: performance aprimorada da conversão entre os tipos `ROW`.
- Tipos estruturais: performance aprimorada das consultas que processam tipos estruturais e contêm verificações, junções, agregações ou gravações de tabela.
- Varreduras de tabela: uma regra de otimização foi introduzida para evitar varreduras de tabelas duplicadas em determinados casos.
- UNION: performance aprimorada das consultas `UNION`.

Performance do planejamento de consultas

- Performance de planejamento: performance aprimorada de planejamento das consultas que unem várias tabelas com um grande número de colunas.

- Avaliações de predicados: performance aprimorada da avaliação de predicados durante a aplicação de predicados ao planejamento.
- Suporte à aplicação de predicados para conversão: suporte à aplicação do predicado `<column> IN <values list>`, em que os valores na lista de valores exigem conversão para corresponder ao tipo de coluna.
- Inferência e aplicação de predicados: inferência e aplicação de predicados estendidas para as consultas que usam um predicado `<symbol> IN <subquery>`.
- Tempos limites: corrigido um bug que, em casos raros, causava o esgotamento dos tempos limites de planejamento de consulta.

Performance de junções

- Junções com colunas de mapa: performance aprimorada das junções e agregações que incluem colunas de mapa.
- Junções apenas com condições de não igualdade: performance aprimorada de junções apenas com condições de não igualdade usando uma junção de loop aninhada, em vez de uma junção de hash.
- Junções externas: o tipo de distribuição de junção agora é selecionado automaticamente para as consultas que envolvem junções externas.
- Junções de intervalo em uma função: performance aprimorada das junções em que a condição é um intervalo em uma função (por exemplo, `a JOIN b ON b.x < f(a.x) AND b.x > g(a.x)`).
- Vazamento em disco: os bugs relacionados a vazamento em disco (spill-to-disk) e os problemas de memória foram corrigidos para melhorar a performance e reduzir erros de memória nas operações `JOIN`.

Performance de subconsultas

- Subconsultas EXISTS correlacionadas: performance aprimorada das subconsultas `EXISTS` correlacionadas.
- Subconsultas correlacionadas com igualdades: suporte aprimorado para subconsultas correlacionadas que contêm predicados de igualdade.
- Subconsultas correlacionadas com desigualdades: performance aprimorada das subconsultas correlacionadas que contêm desigualdades.

- Agregações count(*) em subconsultas: performance aprimorada das agregações `count(*)` em subconsultas com cardinalidade de constantes conhecidas.
- Propagação do filtro de consulta externa: performance aprimorada das subconsultas correlacionadas quando os filtros da consulta externa podem ser propagados para a subconsulta.

Performance de funções

- Funções agregadas da janela: performance aprimorada das funções agregadas da janela.
- element_at(): performance aprimorada de `element_at()` para que os mapas sejam constantes com horário, em vez de serem proporcionais ao tamanho do mapa.
- grouping(): performance aprimorada das consultas que envolvem `grouping()`.
- Conversão de JSON: performance aprimorada da conversão de `JSON` nos tipos `ARRAY` ou `MAP`.
- Funções de retorno de mapas: performance aprimorada das funções que retornam mapas.
- Conversão de mapa em mapa: performance aprimorada da conversão de mapa em mapa.
- min() e max(): as funções `min()` e `max()` foram otimizadas para evitar a criação de objetos desnecessários, o que reduz a sobrecarga de coleta de resíduos.
- row_number(): performance e uso de memória aprimorados para consultas que usam `row_number()` seguido de um filtro nos números gerados das linhas.
- Funções da janela: performance aprimorada das consultas que contêm funções da janela com cláusulas `PARTITION BY` e `ORDER BY` idênticas.
- Funções da janela: performance aprimorada de determinadas funções da janela (por exemplo, `LAG`) que têm especificações semelhantes.

Performance geoespacial

- Serialização de geometria: a performance de serialização dos valores de geometria foi aprimorada.
- Funções geoespaciais: performance aprimorada de `ST_Intersects()`, `ST_Contains()`, `ST_Touches()`, `ST_Within()`, `ST_Overlaps()`, `ST_Disjoint()`, `transform_values()`, `ST_XMin()`, `ST_XMax()`, `ST_YMin()`, `ST_YMax()`, `ST_Crosses()` e `array_intersect()`.
- ST_Distance(): performance aprimorada das consultas de junção que envolvem a função `ST_Distance()`.
- ST_Intersection(): a função `ST_Intersection()` foi otimizada para retângulos alinhados com eixos de coordenadas (por exemplo, polígonos gerados pelas funções `ST_Envelope()` e `bing_tile_polygon()`).

Melhorias relacionadas ao JSON

Funções de mapa

- Performance aprimorada do subscrito de mapa de `O(n)` para `O(1)` em todos os casos. Antes, apenas os mapas produzidos por determinadas funções e leitores aproveitavam essa melhoria.
- As funções `map_from_entries()` e `map_entries()` foram adicionadas.

Conversão

- Recurso adicionado para converter de `REAL`, `TINYINT` ou `SMALLINT` em `JSON`.
- Agora você pode converter `JSON` em `ROW`, mesmo que `JSON` não contenha todos os campos em `ROW`.
- Performance aprimorada de `CAST(json_parse(...) AS ...)`.
- A performance de conversão de `JSON` nos tipos `ARRAY` ou `MAP` foi aprimorada.

Novas funções JSON

- is_json_scalar()

Alterações que podem causar interrupções

As alterações que podem causar interrupções são correções de bugs, alterações em funções geoespaciais, funções substituídas e a introdução de limites. As melhorias feitas de compatibilidade com ANSI SQL podem interromper as consultas que dependiam do comportamento não padrão.

Correções de erros

As alterações a seguir corrigem problemas comportamentais que faziam com que as consultas fossem executadas com êxito, mas com resultados imprecisos.

- As colunas fixed_len_byte_array do Parquet agora são aceitas como DECIMAL: as consultas nas colunas do Parquet do tipo `fixed_len_byte_array` são bem-sucedidas e retornam os valores corretos quando são anotadas como `DECIMAL` no esquema do Parquet. As consultas nas colunas `fixed_len_byte_array` sem a anotação `DECIMAL` resultam em falha com um erro. Antes, as consultas nas colunas `fixed_len_byte_array` sem a anotação DECIMAL eram bem-sucedidas, mas retornavam valores incompreensíveis.

- json_parse() não ignora mais caracteres à direita: antes, as entradas como `[1,2]abc` eram analisadas com êxito como `[1,2]`. O uso de caracteres à direita agora gera a mensagem de erro: Cannot convert '[1, 2]abc' to JSON (Não é possível converter '[1, 2]abc' em JSON).
- Precisão decimal de round() corrigida: `round(x, d)` agora arredonda `x` corretamente quando `x` é um DECIMAL ou quando `x` é um DECIMAL com escala 0 e `d` é um número inteiro negativo. Antes, não havia arredondamento nesses casos.
- round(x, d) e truncate(x, d): o parâmetro `d` na assinatura das funções `round(x, d)` e `truncate(x, d)` agora é do tipo `INTEGER`. Antes, `d` podia ser do tipo `BIGINT`.
- map() com chaves duplicadas: `map()` agora gera um erro em caso de chaves duplicadas, em vez de produzir um mapa corrompido em modo silencioso. As consultas que criam valores de mapa usando chaves duplicadas agora falham com um erro.
- map_from_entries() gera um erro com entradas nulas: `map_from_entries()` agora gera um erro quando o array de entrada contém uma entrada nula. As consultas que criam um mapa especificando `NULL` como valor agora falham.
- Tabelas: as tabelas que têm tipos de partição incompatíveis não podem mais ser criadas.
- Melhoria na estabilidade numérica das funções estatísticas: a estabilidade numérica das funções estatísticas `corr()`, `covar_samp()`, `regr_intercept()` e `regr_slope()` foi melhorada.
- A precisão de TIMESTAMP definida no Parquet agora é respeitada: a precisão dos valores de `TIMESTAMP` e a precisão definida para a coluna `TIMESTAMP` no esquema do Parquet agora devem ser equivalentes. As precisões divergentes resultam em carimbos de data/hora incorretos.
- Informações de fuso horário: as informações de fuso horário agora são calculadas usando o pacote java.time do SDK for Java 1.8.
- SUM dos tipos de dados INTERVAL_DAY_TO_SECOND e INTERVAL_YEAR_TO_MONTH: não é mais possível usar `SUM(NULL)` diretamente. Para usar `SUM(NULL)`, converta `NULL` em um tipo de dados como `BIGINT`, `DECIMAL`, `REAL`, `DOUBLE`, `INTERVAL_DAY_TO_SECOND` ou `INTERVAL_YEAR_TO_MONTH`.

Alterações nas funções geoespaciais

As alterações feitas nas funções geoespaciais incluem o seguinte.

- Alterações nos nomes das funções: alguns nomes de funções foram alterados. Para obter mais informações, consulte Alterações de nomes de funções geoespaciais no mecanismo do Athena versão 2.

- Entrada VARBINARY: o tipo `VARBINARY` não é mais permitido diretamente como entrada para funções geoespaciais. Por exemplo, para calcular a área de uma geometria diretamente, a geometria agora deve ser inserida no formato `VARCHAR` ou `GEOMETRY`. A solução alternativa é usar as funções de transformação, como nos exemplos a seguir.
  - Para usar `ST_area()` para calcular a área da entrada `VARBINARY` no formato Well-known binary (WKB – Binário bem conhecido), especifique primeiro a entrada para `ST_GeomFromBinary()`, por exemplo:

    ```
    ST_area(ST_GeomFromBinary(<wkb_varbinary_value>))
    ```

  - Para usar `ST_area()` para calcular a área da entrada `VARBINARY` no formato binário herdado, especifique primeiro a entrada para a função `ST_GeomFromLegacyBinary()`, por exemplo:

    ```
    ST_area(ST_GeomFromLegacyBinary(<legacy_varbinary_value>))
    ```

- ST_ExteriorRing() e ST_Polygon(): `ST_ExteriorRing()` e `ST_Polygon()` agora aceitam apenas polígonos como entradas. Antes, essas funções aceitavam erroneamente outras geometrias.
- ST_Distance(): conforme exigido pela especificação SQL/MM, a função `ST_Distance()` agora retorna `NULL` se uma das entradas é uma geometria vazia. Antes era retornado `NaN`.

Compatibilidade com ANSI SQL

Os problemas de sintaxe e comportamentais a seguir foram corrigidos para cumprir o padrão ANSI SQL.

- Operações Cast(): as operações Cast() de REAL ou DOUBLE para DECIMAL agora cumprem o padrão SQL. Por exemplo, antes `cast (double '10000000000000000000000000000000' as decimal(38))` retornava `10000000000000000536616220439 3472`, mas agora retorna `10000000000000000000000000000000`.
- JOIN... USING: `JOIN ... USING` agora segue a semântica do padrão SQL. Antes, `JOIN ... USING` exigia a qualificação do nome da tabela em colunas, e a coluna de ambas as tabelas estava presente na saída. As qualificações da tabela agora são inválidas, e a coluna está presente apenas uma vez na saída.
- Literais do tipo ROW removidos: o formato literal do tipo ROW `ROW<int, int>(1, 2)` não é mais permitido. Em vez dele, use a sintaxe `ROW(1 int, 2 int)`.

- Semântica de agregação agrupada: as agregações agrupadas usam `IS NOT DISTINCT FROM` em vez da semântica de igualdade. As agregações agrupadas agora retornam resultados corretos e apresentam uma performance aprimorada ao agrupar com base nos valores de ponto flutuante `NaN`. O agrupamento com base nos tipos de mapa, lista e linha que contêm nulos é permitido.
- Tipos com aspas não são mais permitidos: de acordo com o padrão ANSI SQL, os tipos de dados não podem mais ser colocados entre aspas. Por exemplo, `SELECT "date" '2020-02-02'` não é mais uma consulta válida. Em vez disso, use a sintaxe `SELECT date '2020-02-02'`.
- Acesso anônimo a campos de linha: os campos de linha anônimos não podem mais ser acessados usando a sintaxe `[.field0, .field1, ...]`.
- Operações de agrupamento complexas: as operações de agrupamento complexas `GROUPING SETS`, `CUBE` e `ROLLUP` não permitem agrupamento de expressões compostas de colunas de entrada. Somente nomes de coluna são permitidos.

Funções substituídas

As funções a seguir não são mais aceitas e foram substituídas pela sintaxe que gera os mesmos resultados.

- information_schema.__internal_partitions__: o uso de `__internal_partitions__` não é mais permitido no mecanismo Athena versão 2. Como sintaxe equivalente, use `SELECT * FROM "<table_name>$partitions"` ou `SHOW PARTITIONS`. Para obter mais informações, consulte Listar partições de uma tabela específica.
- Funções geoespaciais substituídas: para obter uma lista de funções geoespaciais que tiveram os nomes alterados, consulte Alterações de nomes de funções geoespaciais no mecanismo do Athena versão 2.

Limites

Os limites a seguir foram introduzidos no mecanismo do Athena versão 2 para garantir que as consultas não falhem devido a limitações de recursos. Esses limites não são configuráveis pelos usuários.

- Número de elementos do resultado: o número de elementos do resultado n é restrito a no máximo 10.000 para as seguintes funções: `min(col, n)`, `max(col, n)`, `min_by(col1, col2, n)` e `max_by(col1, col2, n)`.
- GROUPING SETS: o número máximo de fatias em um conjunto de agrupamento é 2.048.

- Comprimento máximo de linha de arquivo de texto: o comprimento máximo de linha padrão para arquivos de texto é de 200 MB.
- Tamanho máximo do resultado da função de sequência: o tamanho máximo do resultado de uma função de sequência é 50.000 entradas. Por exemplo, `SELECT sequence(0,45000,1)` é bem-sucedido, mas `SELECT sequence(0,55000,1)` falha com a mensagem de erro: The result of the sequence function must not have more than 50000 entries (O resultado da função de sequência não deve ter mais de 50.000 entradas). Esse limite se aplica a todos os tipos de entrada das funções de sequência, incluindo carimbos de data/hora.

Referência do SQL para o Athena

O Amazon Athena oferece um subconjunto de instruções, funções, operadores e tipos de dados em Data Definition Language (DDL – Linguagem de definição de dados) e Data Manipulation Language (DML – Linguagem de manipulação de dados). Com algumas exceções, a DDL do Athena é baseada na DDL do HiveQL e a DML do Athena é baseada no Trino. Para obter informações sobre as versões do mecanismo do Athena, consulte Versionamento do mecanismo do Athena.

Tópicos

- Tipos de dados no Amazon Athena
- Consultas, funções e operadores em DML
- Instruções DDL
- Considerações e limitações das consultas SQL no Amazon Athena

Tipos de dados no Amazon Athena

Ao executar `CREATE TABLE`, você especifica os nomes de coluna e o tipo de dados que cada coluna pode conter. As tabelas que você cria são armazenadas no AWS Glue Data Catalog.

Para facilitar a interoperabilidade com outros mecanismos de consulta, o Athena usa nomes de tipos de dados do Apache Hive para instruções DDL, como `CREATE TABLE`. Para consultas DML como `SELECT`, `CTAS` e `INSERT INTO`, o Athena usa nomes de tipo de dados Trino. A tabela a seguir mostra os tipos de dados compatíveis com o Athena. Quando os tipos DDL e DML forem diferentes em termos de nome, disponibilidade ou sintaxe, eles serão apresentados em colunas separadas.

DDL	DML	Descrição
BOOLEAN		Os valores são `true` e `false`.
TINYINT		Um número inteiro de 8 bits com sinal no formato de complemento de 2, com um valor mínimo de -2^7 e um valor máximo de 2^7-1.
SMALLINT		Um número inteiro de 16 bits com sinal no formato de complemento de 2, com um valor mínimo de -2^{15} e um valor máximo de $2^{15}-1$.
INT, INTEGER		Um valor de 32 bits com sinal no formato de complemento de 2, com um valor mínimo de -2^{31} e um valor máximo de $2^{31}-1$.
BIGINT		Um número inteiro de 64 bits com sinal no formato de complemento de 2, com um valor mínimo de -2^{63} e um valor máximo de $2^{63}-1$.
FLOAT	REAL	Um número de 32 bits com sinal de ponto flutuante de precisão única. O intervalo é de 1,40129846432481707e-45 a 3,40282346638528860e+38, positivo ou negativo. Segue o padrão IEEE para aritmética de ponto flutuante (IEEE 754).
DOUBLE		Um número de 64 bits com sinal de ponto flutuante de precisão dupla. O intervalo é de 4,94065645841246544e-324d a 1,79769313486231570e+308d, positivo ou negativo. Segue o padrão IEEE para aritmética de ponto flutuante (IEEE 754).
DECIMAL(*precisão*, *escala*)		*precision* é o número total de dígitos. *scale* (opcional) é o número de dígitos na parte da fração com um padrão de 0. Por exemplo, use estas definições de tipo: `decimal(11,5)` , `decimal(15)` . O valor máximo para *precisão* é 38 e o valor máximo para *escala* é 38.

DDL	DML	Descrição
CHAR, CHAR(*comprimento*)		Dados de caractere de comprimento fixo, com um comprimento especificado entre 1 e 255, p. ex., char(10). Se o *comprimento* for especificado, as strings serão truncadas no comprimento especificado quando lidas. Se a string de dados subjacente for mais longa, a string de dados subjacente permanecerá inalterada. Para obter mais informações, consulte CHAR Hive data type (Tipo de dado CHAR do Hive).
STRING	VARCHAR	Dados de caracteres de comprimento variável.
VARCHAR(*comprimento*)		Dados de caracteres de comprimento variável com um comprimento máximo de leitura. As strings serão truncadas no comprimento especificado quando lidas. Se a string de dados subjacente for mais longa, a string de dados subjacente permanecerá inalterada.
BINARY	VARBINARY	Dados binários de comprimento variável.
TIME		Uma hora do dia com precisão de milissegundos.
Indisponível	TIME(*precisão*)	Uma hora do dia com uma precisão específica. `TIME(3)` é equivalente a `TIME`.
Indisponível	TIME WITH TIME ZONE	Uma hora do dia em um fuso horário. Os fusos horários devem ser especificados como desvios em relação ao horário UTC.
DATA		Uma data do calendário com o ano, o mês e o dia.
TIMESTAMP	TIMESTAMP , TIMESTAMP WITHOUT TIME ZONE	Uma data do calendário e hora do dia com precisão de milissegundos.

DDL	DML	Descrição
Indisponível	TIMESTAMP (*precisão*), TIMESTAMP (*precisão*) WITHOUT TIME ZONE	Uma data do calendário e hora do dia com uma precisão específica. `TIMESTAMP(3)` é equivalente a `TIMESTAMP` .
Indisponível	TIMESTAMP WITH TIME ZONE	Uma data do calendário e hora do dia em um fuso horário. É possível especificar os fusos horários como desvios em relação ao horário UTC, como nomes de fuso horário IANA ou usando UTC, UT, Z ou GMT.
Indisponível	TIMESTAMP (*precisão*) WITH TIME ZONE	Uma data do calendário e hora do dia com uma precisão específica, em um fuso horário.
Indisponível	INTERVALO ENTRE UM ANO E UM MÊS	Um intervalo de um ou mais meses inteiros
Indisponível	INTERVALO ENTRE UM DIA E UM SEGUNDO	Um intervalo de um ou mais segundos, minutos, horas ou dias
ARRAY<*tipo_elemento* >	ARRAY[*tipo_elemento*]	Uma matriz de valores. Todos os valores devem ser do mesmo tipo.
MAP<*tipo_chave* , *tipo_valor* >	MAP(*tipo_chave* , *tipo_valor*)	Um mapa no qual é possível consultar os valores por chave. Todas as chaves devem ter o mesmo valor e todos os valores devem ter o mesmo valor.

DDL	DML	Descrição
STRUCT<*nome_campo_1* :*tipo_campo_1* , *nome_campo_2* :*tipo_campo_2* , ...>	ROW(*nome_campo_1* :*tipo_campo_1* , *nome_campo_2* :*tipo_campo_2* , ...)	Uma estrutura de dados com campos nomeados e seus valores.
Indisponível	JSON	Tipo de valor JSON, que pode ser um objeto JSON, uma matriz JSON, um número JSON, uma string JSON, `true`, `false` ou `null`.
Indisponível	UUID	Um UUID (identificador exclusivo universal).
Indisponível	IPADDRESS	Endereço IPv4 ou IPv6.
Indisponível	Log de HyperLogLog	Esses tipos de dados são compatíveis com mecanismos internos de funções aproximadas. Para obter mais informações sobre cada tipo, acesse o link da entrada correspondente na documentação do Trino.
	P4HyperLogLog	
	SetDigest	
	QDigest	
	TDigest	

Exemplos de tipo de dados

A tabela a seguir apresenta exemplos de literais para tipos de dados DML.

Tipo de dados	Exemplos
BOOLEAN	`true` `false`
TINYINT	`TINYINT '123'`

Tipo de dados	Exemplos
SMALLINT	`SMALLINT '123'`
INT, INTEGER	`123456790`
BIGINT	`BIGINT '1234567890'` `2147483648`
REAL	`'123456.78'`
DOUBLE	`1.234`
DECIMAL(*precisão*, *escala*)	`DECIMAL '123.456'`
CHAR, CHAR(*comprimento*)	`CHAR 'hello world'`, `CHAR 'hello ''world''!'`
VARCHAR, VARCHAR(*comprimento*)	`VARCHAR 'hello world'`, `VARCHAR 'hello ''world''!'`
VARBINARY	`X'00 01 02'`
TIME, TIME(*precisão*)	`TIME '10:11:12'` , `TIME '10:11:12.345'`
TIME WITH TIME ZONE	`TIME '10:11:12.345 -06:00'`
DATA	`DATE '2024-03-25'`
TIMESTAMP, TIMESTAMP WITHOUT TIME ZONE, TIMESTAMP(*precisão*), TIMESTAMP(*precisão*) WITHOUT TIME ZONE	`TIMESTAMP '2024-03-25 11:12:13'` , `TIMESTAMP '2024-03-25 11:12:13.456'`

Tipo de dados	Exemplos
TIMESTAMP WITH TIME ZONE, TIMESTAMP (*precisão*) WITH TIME ZONE	`TIMESTAMP '2024-03-25 11:12:13.456 Europe/Berlin'`
INTERVALO ENTRE UM ANO E UM MÊS	`INTERVAL '3' MONTH`
INTERVALO ENTRE UM DIA E UM SEGUNDO	`INTERVAL '2' DAY`
ARRAY[*tipo_elemento*]	`ARRAY['one', 'two', 'three']`
MAP(*tipo_chave* , *tipo_valor*)	`MAP(ARRAY['one', 'two', 'three'], ARRAY[1, 2, 3])` Observe que os mapas são criados com base em uma matriz de chaves e uma matriz de valores.

ROW(*nome_campo_1* :*tipo_campo_1* , *nome_campo_2* :*tipo_campo_2* , ...)	`ROW('one', 'two', 'three')`
	
	Observe que as linhas criadas dessa forma não têm nomes de colunas. Para adicionar nomes de colunas, você pode usar CAST, como no exemplo a seguir:
	
	`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'`

Considerações sobre tipos de dados

Limites de tamanho

Para tipos de dados que não especificam um limite de tamanho, lembre-se de que há um limite prático de 32 MB para todos os dados em uma única linha. Para obter mais informações, consulte Row or column size limitation em Considerações e limitações das consultas SQL no Amazon Athena.

CHAR e VARCHAR

Um valor `CHAR(n)` sempre tem uma contagem de *n* caracteres. Por exemplo, se você lançar “abc” para `CHAR(7)`, 4 espaços finais serão adicionados.

As comparações de valores `CHAR` incluem espaços à esquerda e à direita.

Se um comprimento for especificado para `CHAR` ou `VARCHAR`, as strings serão truncadas no comprimento especificado quando lidas. Se a string de dados subjacente for mais longa, a string de dados subjacente permanecerá inalterada.

Para escapar uma aspas simples em um `CHAR` ou `VARCHAR`, use uma aspas simples adicional.

Para converter um tipo de dados que não seja uma string em uma consulta DML, converta para o tipo de dados `VARCHAR`.

Para usar a função `substr` para retornar uma substring de comprimento específico proveniente de um tipo de dados `CHAR`, primeiro é necessário lançar o valor `CHAR` como `VARCHAR`. No exemplo a seguir, `col1` usa o tipo de dados `CHAR`.

```
substr(CAST(col1 AS VARCHAR), 1, 4)
```

DECIMAL

Para especificar valores decimais como literais em consultas `SELECT`, como ao selecionar linhas com um valor decimal específico, é possível especificar o tipo `DECIMAL` e listar o valor decimal como um literal entre aspas simples na consulta, como nos exemplos a seguir.

```
SELECT * FROM my_table
WHERE decimal_value = DECIMAL '0.12'
```

```
SELECT DECIMAL '44.6' + DECIMAL '77.2'
```

Trabalhar com dados de carimbo de data e hora

Esta seção descreve algumas considerações para trabalhar com dados de carimbo de data e hora no Athena.

> **Note**
>
> O tratamento dos carimbos de data e hora mudou um pouco entre o mecanismo do Athena versão 2 e o mecanismo do Athena versão 3. Para obter informações sobre erros relacionados ao carimbo de data e hora que podem ocorrer no mecanismo do Athena versão 3 e soluções sugeridas, consulte Alterações de carimbo de data/hora na referência Mecanismo Athena versão 3.

Formato para gravar dados de carimbo de data e hora em objetos do Amazon S3

O formato no qual os dados de carimbo de data e hora devem ser gravados em objetos do Amazon S3 depende do tipo de dados da coluna e da biblioteca SerDe utilizada.

- Se você tiver uma coluna de tabela do tipo `DATE`, o Athena espera que a coluna ou propriedade correspondente dos dados seja uma string no formato ISO `YYYY-MM-DD` ou um tipo de data incorporado, como aqueles para Parquet ou ORC.
- Se você tiver uma coluna de tabela do tipo `TIME`, o Athena espera que a coluna ou propriedade correspondente dos dados seja uma string no formato ISO `HH:MM:SS` ou um tipo de hora incorporado, como aqueles para Parquet ou ORC.
- Se você tiver uma coluna de tabela do tipo `TIMESTAMP`, o Athena espera que a coluna ou propriedade correspondente dos dados seja uma string no formato `YYYY-MM-DD HH:MM:SS.SSS` (observe o espaço entre a data e a hora) ou um tipo de hora incorporado, como aqueles para Parquet, ORC ou Ion.

> **Note**
>
> Os carimbos de data e hora do OpenCsvSerDe são uma exceção e devem ser codificados como epochs UNIX com resolução de milissegundos.

Garantir que os dados particionados por tempo correspondam ao campo de carimbo de data e hora em um registro

O produtor dos dados deve garantir que os valores da partição estejam alinhados com os dados na partição. Por exemplo, se os dados tiverem uma propriedade `timestamp` e você usar o Firehose para carregá-los no Amazon S3, será necessário usar o particionamento dinâmico porque o particionamento padrão do Firehose é baseado em hora real do relógio.

Usar string como o tipo de dados para chaves de partição

Por motivos de performance, é preferível usar `STRING` como tipo de dados para chaves de partição. Embora o Athena reconheça valores de partição no formato `YYYY-MM-DD` como datas quando você usa o tipo `DATE`, isso pode levar a uma performance ruim. Por isso, recomenda-se usar o tipo de dados `STRING` para chaves de partição.

Como gravar consultas em campos de carimbo de data e hora que também são particionados por tempo

A forma como você grava consultas em campos de carimbo de data e hora que são particionados por hora depende do tipo de tabela que você deseja consultar.

Tabelas Hive

Com as tabelas Hive mais usadas no Athena, o mecanismo de consulta não tem conhecimento das relações entre as colunas e as chaves de partição. Por isso, você sempre deve adicionar predicados às consultas tanto para a coluna como para a chave de partição.

Por exemplo, suponha que você tenha uma coluna `event_time` e uma chave de partição `event_date` e queira consultar eventos entre 23:00 e 03:00. Nesse caso, é necessário incluir predicados em sua consulta para a coluna e a chave de partição, como no exemplo a seguir.

```
WHERE event_time BETWEEN start_time AND end_time
  AND event_date BETWEEN start_time_date AND end_time_date
```

Tabelas Iceberg

Com as tabelas Iceberg, é possível usar valores de partição computados, o que simplifica suas consultas. Por exemplo, suponha que sua tabela do Iceberg tenha sido criada com uma cláusula `PARTITIONED BY` como esta:

```
PARTITIONED BY (event_date month(event_time))
```

Nesse caso, o mecanismo de consulta remove automaticamente as partições com base nos valores dos predicados `event_time`. Por isso, sua consulta só precisa especificar um predicado para `event_time`, como no exemplo a seguir.

```
WHERE event_time BETWEEN start_time AND end_time
```

Para ter mais informações, consulte Criar tabelas Iceberg.

Consultas, funções e operadores em DML

Em geral, o mecanismo de consulta DML do Athena é compatível com a sintaxe do Trino e do Presto e acrescenta suas próprias melhorias. O Athena não é compatível com todos os recursos do Trino ou Presto. Para obter mais informações, consulte os tópicos das instruções específicas nesta seção e Considerações e limitações. Para obter informações sobre as funções, consulte Funções no Amazon Athena. Para obter informações sobre as versões do mecanismo do Athena, consulte Versionamento do mecanismo do Athena.

Para obter mais informações sobre instruções DDL, consulte Instruções DDL. Para ver uma lista de instruções DDL não permitidas, consulte DDL incompatível.

SELECT

Recupera linhas de dados de zero ou mais tabelas.

> **Note**
>
> Este tópico fornece informações resumidas para referência. Informações abrangentes sobre o uso de `SELECT` e a linguagem SQL estão além do escopo desta documentação. Para obter informações sobre como usar o SQL específico do Athena, consulte Considerações e limitações das consultas SQL no Amazon Athena e Executar consultas SQL usando o Amazon Athena. Para ver um exemplo de como criar um banco de dados, criar uma tabela e executar uma consulta `SELECT` na tabela do Athena, consulte Conceitos básicos.

Resumo

```
[ 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**
>
> Palavras reservadas em instruções SQL SELECT devem ficar entre aspas duplas. Para ter mais informações, consulte Lista de palavras-chave reservadas em instruções SQL SELECT.

Parâmetros

[WITH with_query [, ...]]

Você pode usar `WITH` para nivelar consultas aninhadas ou para simplificar subconsultas.

O uso da cláusula `WITH` para criar consultas recursivas é possível a partir da versão 3 do mecanismo Athena. A profundidade de recursão máxima é 10.

A cláusula `WITH` precede a lista `SELECT` em uma consulta e define uma ou mais subconsultas a serem usadas dentro da consulta `SELECT`.

Cada subconsulta define uma tabela temporária, semelhante a uma definição de exibição, que você pode referenciar na cláusula `FROM`. As tabelas são usadas apenas quando a consulta é executada.

A sintaxe `with_query` é:

```
subquery_table_name [ ( column_name [, ...] ) ] AS (subquery)
```

Em que:

- `subquery_table_name` é um nome exclusivo para uma tabela temporária que define os resultados da subconsulta de cláusula `WITH`. Cada `subquery` deve ter um nome de tabela que possa ser referenciado na cláusula `FROM`.

- `column_name [, ...]` é uma lista opcional de nomes de coluna de saída. O número de nomes de coluna deve ser igual a ou menor que o número de colunas definido por `subquery`.
- `subquery` é uma instrução da consulta qualquer.

[ALL | DISTINCT] select_expression

`select_expression` determina as linhas a serem selecionadas. Uma `select_expression` pode usar um dos seguintes formatos:

```
expression [ [ AS ] column_alias ] [, ...]
```

```
row_expression.* [ AS ( column_alias [, ...] ) ]
```

```
relation.*
```

```
*
```

- A sintaxe `expression [ [ AS ] column_alias ]` especifica uma coluna de saída. A sintaxe opcional `[AS] column_alias` especifica um nome de título personalizado a ser usado para a coluna na saída.
- Para `row_expression.* [ AS ( column_alias [, ...] ) ]`, `row_expression` é uma expressão arbitrária do tipo de dados `ROW`. Os campos da linha definem as colunas de saída a serem incluídas no resultado.
- Para `relation.*`, as colunas de `relation` são incluídas no resultado. Essa sintaxe não permite o uso de aliases de coluna.
- O asterisco `*` especifica que todas as colunas sejam incluídas no conjunto de resultados.
- No conjunto de resultados, a ordem das colunas é igual à ordem de sua especificação pela expressão de seleção. Se uma expressão de seleção retornar várias colunas, a ordem das colunas segue a ordem usada na relação de origem ou na expressão do tipo de linha.
- Quando os aliases de coluna são especificados, os aliases substituem os nomes de campos de coluna ou linha pré-existentes. Se a expressão de seleção não tiver nomes de coluna, nomes de colunas anônimas com índice zero (`_col0`,`_col1` e `_col2, ...`) serão exibidos na saída.
- `ALL` é o padrão. Usar `ALL` será tratado da mesma maneira como se tivesse sido omitido. Todas as linhas de todas as colunas são selecionadas, e as duplicações são mantidas.
- Use `DISTINCT` para retornar somente valores distintos quando uma coluna contém valores duplicados.

FROM from_item [, ...]

Indica a entrada para a consulta, em que `from_item` pode ser uma exibição, um construto de união ou uma subconsulta conforme descrito abaixo.

O `from_item` pode ser:

- `table_name [ [ AS ] alias [ (column_alias [, ...]) ] ]`

  Em que `table_name` é o nome da tabela de destino da qual selecionar linhas, `alias` é o nome para indicar a saída da instrução `SELECT` e `column_alias` define as colunas para o `alias` especificado.

-OU-

- `join_type from_item [ ON join_condition | USING ( join_column [, ...] ) ]`

  Em que `join_type` é um dos:

  - `[ INNER ] JOIN`
  - `LEFT [ OUTER ] JOIN`
  - `RIGHT [ OUTER ] JOIN`
  - `FULL [ OUTER ] JOIN`
  - `CROSS JOIN`
  - `ON join_condition | USING (join_column [, ...])` Em que usar `join_condition` permite especificar nomes de coluna para chaves de união em várias tabelas e usar `join_column` exige que `join_column` exista em ambas as tabelas.

[WHERE condição]

Filtra os resultados de acordo com a `condition` que você especificar, em que `condition` costuma ter a sintaxe abaixo.

```
column_name operator value [[[AND | OR] column_name operator value] ...]
```

O *`operador`* pode ser um dos comparadores =, >, <, >=, <=, <>, !=.

As expressões de subconsulta a seguir também podem ser usadas na cláusula `WHERE`.

- `[NOT] BETWEEN` *`integer_A`* `AND` *`integer_B`*: especifica um intervalo entre dois inteiros, como no exemplo a seguir. Se o tipo de dados da coluna for `varchar`, a coluna deverá ser convertida para inteiro primeiro.

```
SELECT DISTINCT processid FROM "webdata"."impressions"
WHERE cast(processid as int) BETWEEN 1500 and 1800
ORDER BY processid
```

- `[NOT] LIKE value`: pesquisa o padrão especificado. Use o sinal de porcentagem (`%`) como caractere curinga, conforme mostrado no exemplo a seguir.

```
SELECT * FROM "webdata"."impressions"
WHERE referrer LIKE '%.org'
```

- `[NOT] IN (value[, value[, ...])`: especifica uma lista de valores possíveis para uma coluna, como no exemplo a seguir.

```
SELECT * FROM "webdata"."impressions"
WHERE referrer IN ('example.com','example.net','example.org')
```

[GROUP BY [ALL | DISTINCT] grouping_expressions [, ...]]

Divide a saída da instrução `SELECT` em linhas com valores correspondentes.

`ALL` e `DISTINCT` determinam se conjuntos de agrupamentos duplicados produzem linhas de saída distintas. Se omitido, `ALL` será pressuposto.

`grouping_expressions` permite realizar operações de agrupamento complexas. Você pode usar operações de agrupamento complexas para realizar uma análise que exija agregação de vários conjuntos de colunas em uma única consulta.

O elemento `grouping_expressions` pode ser qualquer função, como `SUM`, `AVG` ou `COUNT`, executada nas colunas de entrada.

As expressões `GROUP BY` podem agrupar a saída por nomes de coluna de entrada não exibidos na saída da instrução `SELECT`.

Todas as expressões de saída devem ser funções agregadas ou colunas presentes na cláusula `GROUP BY`.

Você pode usar uma única consulta para realizar uma análise que exija a agregação de vários conjuntos de colunas.

O Athena aceita agregações complexas que usam `GROUPING SETS`, `CUBE` e `ROLLUP`. `GROUP BY GROUPING SETS` especifica várias listas de colunas para agrupamento. `GROUP BY CUBE`

gera todos os conjuntos de agrupamento possíveis para um determinado conjunto de colunas. `GROUP BY ROLLUP` gera todos os subtotais possíveis para um determinado conjunto de colunas. As operações de agrupamento complexas não permitem agrupamento de expressões compostas de colunas de entrada. Somente nomes de coluna são permitidos.

Você normalmente pode usar `UNION ALL` para obter os mesmos resultados dessas operações `GROUP BY`, mas consultas que usam `GROUP BY` têm a vantagem de ler os dados uma vez, e `UNION ALL` lê os dados subjacentes três vezes e podem produzir resultados inconsistentes quando a fonte de dados está sujeita a alterações.

[HAVING condition]

Usada com funções de agregação e a cláusula `GROUP BY`. Controla quais grupos são selecionados, eliminando grupos que não atendam a `condition`. A filtragem ocorrerá depois de grupos, e as agregações serão calculadas.

[{ UNION | INTERSECT | EXCEPT } [ALL | DISTINCT] union_query]]

`UNION`, `INTERSECT` e `EXCEPT` combinam os resultados de mais de uma instrução `SELECT` em uma única consulta. `ALL` ou `DISTINCT` controla a exclusividade das linhas incluídas no conjunto final de resultados.

`UNION` combina as linhas resultantes da primeira consulta com as linhas resultantes da segunda consulta. Para eliminar duplicatas, `UNION` cria uma tabela de hash, que consome memória. Para melhor performance, considere usar `UNION ALL` se a consulta não exigir eliminação de duplicatas. Várias cláusulas `UNION` são processadas da esquerda para a direita, a menos que você use parênteses para definir explicitamente a ordem de processamento.

`INTERSECT` retorna apenas as linhas que estão presentes nos resultados da primeira e da segunda consultas.

`EXCEPT` retorna as linhas dos resultados da primeira consulta, excluindo as linhas encontradas pela segunda consulta.

`ALL` faz com que todas as linhas sejam incluídas, mesmo se elas forem idênticas.

`DISTINCT` faz com que apenas as linhas exclusivas sejam incluídas no conjunto de resultados combinados.

[ORDER BY expression [ASC | DESC] [NULLS FIRST | NULLS LAST] [, ...]]

Classifica um conjunto de resultados por um ou mais `expression` de saída.

Quando a cláusula contém várias expressões, o conjunto de resultados é classificado de acordo com o primeiro `expression`. Em seguida, o segundo `expression` é aplicado a linhas que tenham valores correspondentes da primeira expressão, e assim por diante.

Cada `expression` pode especificar colunas de saída de `SELECT` ou um número ordinal para uma coluna de saída por posição, a partir de um.

`ORDER BY` é avaliada como a última etapa após qualquer cláusula `GROUP BY` ou `HAVING`. `ASC` e `DESC` determinam se os resultados são classificados em ordem crescente ou decrescente.

A ordem nula padrão é `NULLS LAST`, independentemente da ordem de classificação crescente ou decrescente.

[Contagem de DESLOCAMENTO [LINHA | LINHAS]]

Use a cláusula `OFFSET` para descartar várias linhas iniciais do conjunto de resultados. Se a cláusula `ORDER BY` estiver presente, a cláusula `OFFSET` será avaliada em um conjunto de resultados classificados e o conjunto permanecerá classificado após as linhas ignoradas serem descartadas. Se a consulta não tiver cláusula `ORDER BY`, a definição de quais linhas serão descartadas é arbitrária. Se a contagem especificada por `OFFSET` for igual ou exceder o tamanho do conjunto de resultados, o resultado final será vazio.

LIMIT [count | ALL]

Restringe o número de linhas no conjunto de resultados a `count`. `LIMIT ALL` é igual à omissão da cláusula `LIMIT`. Se a consulta não tiver a cláusula `ORDER BY`, os resultados serão arbitrários.

TABLESAMPLE [BERNOULLI | SYSTEM] (porcentagem)

Operador operacional para selecionar linhas de uma tabela com base em um método de amostragem.

`BERNOULLI` seleciona cada linha para estar no exemplo da tabela com uma probabilidade de `percentage`. Todos os blocos físicos da tabela são examinados, e determinadas linhas são ignoradas com base em uma comparação entre o `percentage` de exemplo e um valor aleatório calculado no runtime

Com `SYSTEM`, a tabela é dividida em segmentos lógicos de dados, e a tabela mostra um exemplo dessa granularidade.

Todas as linhas de um determinado segmento são selecionadas, ou o segmento é ignorado com base em uma comparação entre o `percentage` de exemplo e um valor aleatório calculado

no runtime. A amostragem de SYSTEM depende do conector. Esse método não garante probabilidades de amostragem independentes.

[UNNEST (array_or_map) [WITH ORDINALITY]]

Expande uma matriz ou um mapa para uma relação. As matrizes são expandidas para uma única coluna. Os mapas são expandidos para duas colunas (chave, valor).

Você pode usar UNNEST com vários argumentos, que são expandidos para várias colunas com o máximo de linhas do maior argumento de cardinalidade.

Outras colunas são preenchidas com nulos.

A cláusula WITH ORDINALITY adiciona uma coluna de ordinalidade ao final.

UNNEST costuma ser usado com um JOIN e pode fazer referência a colunas de relações no lado esquerdo do JOIN.

Obter os locais de arquivos dos dados de origem no Amazon S3

Para ver o local do arquivo do Amazon S3 referente aos dados em uma linha da tabela, você pode usar "$path" em uma consulta SELECT, como no seguinte exemplo:

```
SELECT "$path" FROM "my_database"."my_table" WHERE year=2019;
```

Essa consulta retorna um resultado semelhante a este:

```
s3://DOC-EXAMPLE-BUCKET/datasets_mytable/year=2019/data_file1.json
```

Para retornar uma lista classificada e exclusiva dos caminhos de nome de arquivo do S3 para os dados em uma tabela, você pode usar SELECT DISTINCT e ORDER BY, como no exemplo a seguir.

```
SELECT DISTINCT "$path" AS data_source_file
FROM sampledb.elb_logs
ORDER By data_source_file ASC
```

Para retornar somente os nomes de arquivo sem o caminho, você pode especificar "$path" como um parâmetro para a função regexp_extract, conforme mostrado no exemplo a seguir.

```
SELECT DISTINCT regexp_extract("$path", '[^/]+$') AS data_source_file
```

```
FROM sampledb.elb_logs
ORDER By data_source_file ASC
```

Para retornar os dados de um arquivo específico, especifique o arquivo na cláusula `WHERE`, como no exemplo a seguir.

```
SELECT *,"$path" FROM my_database.my_table WHERE "$path" = 's3://DOC-EXAMPLE-BUCKET/
my_table/my_partition/file-01.csv'
```

Para obter mais informações e exemplos, consulte o artigo da Central de Conhecimento Como posso ver o arquivo de origem do Amazon S3 para uma linha em uma tabela do Athena?

> **Note**
>
> No Athena, as colunas ocultas de metadados do Hive ou do Iceberg `$bucket`, `$file_modified_time`, `$file_size` e `$partition` não são compatíveis para visualizações.

Caractere de escape para aspas simples

Para inserir caracteres de escape em aspas simples, preceda-as com outras aspas simples, conforme o exemplo a seguir. Não confunda isso com aspas duplas.

```
Select 'O''Reilly'
```

Resultados

```
O'Reilly
```

Recursos adicionais do

Para obter mais informações sobre como usaras instruções `SELECT` no Athena, consulte os recursos abaixo.

Para obter informações sobre este tópico	consulte esta referência
Executar consultas no Athena	Executar consultas SQL usando o Amazon Athena

Para obter informações sobre este tópico	consulte esta referência
Usar `SELECT` para criar uma tabela	Criar uma tabela a partir de resultados de consultas (CTAS)
Inserir dados de uma consulta `SELECT` em outra tabela	INSERT INTO
Usar funções integradas nas instruções `SELECT`	Funções no Amazon Athena
Usar funções definidas pelo usuário nas instruções `SELECT`	Executar consultas com funções definidas pelo usuário
Consultar metadados do catálogo de dados	Consulta no AWS Glue Data Catalog

INSERT INTO

Insere novas linhas em uma tabela de destino com base em uma instrução de consulta `SELECT` executada em uma tabela de origem ou com base em um conjunto de `VALUES` fornecidos como parte da instrução. Quando a tabela de origem é baseada em dados subjacentes em um formato, como CSV ou JSON, e a tabela de destino é baseada em outro formato, como Parquet ou ORC, você pode usar as consultas `INSERT INTO` para transformar os dados selecionados no formato da tabela de destino.

Considerações e limitações

Considere o seguinte ao usar as consultas `INSERT` com o Athena.

- Ao executar uma consulta `INSERT` em uma tabela com dados subjacentes criptografados no Amazon S3, os arquivos de saída que a consulta `INSERT` grava não são criptografados por padrão. Recomendamos que você criptografe os resultados da consulta `INSERT` se estiver inserindo em tabelas com dados criptografados.

  Para obter mais informações sobre como criptografar resultados da consulta usando o console, consulte Criptografar resultados das consultas do Athena armazenados no Amazon S3. Para habilitar a criptografia usando a AWS CLI ou a API do Athena, use as propriedades `EncryptionConfiguration` da ação StartQueryExecution para especificar as opções de criptografia do Amazon S3 de acordo com os seus requisitos.

- Para instruções `INSERT INTO`, a configuração do proprietário do bucket esperado não se aplica ao local da tabela de destino no Amazon S3. A configuração esperada do proprietário do bucket se aplica somente ao local de saída do Amazon S3 que você especificar para os resultados da consulta do Athena. Para obter mais informações, consulte Especificar um local para resultados de consultas usando o console do Athena.
- Para obter instruções `INSERT INTO` em conformidade com ACID, consulte a seção `INSERT INTO` em Atualizar dados nas tabelas Iceberg.

Formatos compatíveis e SerDes

É possível executar uma consulta `INSERT` em tabelas criadas de dados com os seguintes formatos e SerDes.

Formato de dados	SerDe
Avro	org.apache.hadoop.hive.serde2.avro.AvroSerDe
Ion	com.amazon.ionhiveserde.IonHiveSerDe
JSON	org.apache.hive.hcatalog.data.JsonSerDe
ORC	org.apache.hadoop.hive.ql.io.orc.OrcSerde
Parquet	org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe
Arquivo de texto	org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe **Note** Arquivos CSV, TSV e delimitados personalizados são compatíveis.

Tabelas em bucket sem suporte

`INSERT INTO` não é compatível com tabelas em bucket. Para obter mais informações, consulte Particionamento e bucketing no Athena.

Consultas federadas sem suporte

`INSERT INTO` não é suportado para consultas federadas. Tentar fazer isso pode gerar a mensagem de erro: This operation is currently not supported for external catalogs (Atualmente, esta operação não é suportada para catálogos externos). Para obter informações sobre consultas federadas, consulte Usar a consulta federada do Amazon Athena.

Particionamento

Considere os pontos desta seção ao usar o particionamento com as consultas `INSERT INTO` ou `CREATE TABLE AS SELECT`.

Limites

A instrução `INSERT INTO` comporta a gravação de no máximo 100 combinações de partições na tabela de destino. Se você executar a cláusula `SELECT` em uma tabela com mais de 100 partições, a consulta falhará a menos que a consulta `SELECT` seja limitada a 100 partições ou menos.

Para obter informações sobre como contornar essa limitação, consulte Usar CTAS e INSERT INTO para resolver o limite de 100 partições.

Ordem das colunas

As instruções `INSERT INTO` ou `CREATE TABLE AS SELECT` esperam que a coluna particionada seja a última na lista de colunas projetadas em uma instrução `SELECT`.

Se a tabela de origem não for particionada ou for particionada em colunas diferentes em comparação com a tabela de destino, as consultas como `INSERT INTO` *`destination_table`* `SELECT * FROM` *`source_table`* vão considerar os valores na última coluna da tabela de origem como os valores de uma coluna de partição na tabela de destino. Tenha isso em mente ao tentar criar uma tabela particionada com base em uma tabela não particionada.

Recursos

Para obter mais informações sobre como usar `INSERT INTO` com particionamento, consulte os recursos abaixo.

- Para inserir dados particionados em uma tabela particionada, consulte Usar CTAS e INSERT INTO para resolver o limite de 100 partições.
- Para inserir dados não particionados em uma tabela particionada, consulte Usar CTAS e INSERT INTO para ETL e análise de dados.

Arquivos gravados no Amazon S3

O Athena grava arquivos nos locais dos dados de origem no Amazon S3 como resultado do comando `INSERT`. Cada operação `INSERT` cria um novo arquivo, em vez de anexar a um arquivo existente. Os locais de arquivos dependem da estrutura da tabela e da consulta `SELECT`, se houver. O Athena gera um arquivo manifesto de dados para cada consulta `INSERT`. O manifesto rastreia os arquivos que a consulta gravou. Ele é salvo no local dos resultados das consultas do Athena no Amazon S3. Para obter mais informações, consulte Identificar os arquivos de saída das consultas.

Evitar atualizações altamente transacionais

Quando você usa `INSERT INTO` para adicionar linhas a uma tabela no Amazon S3, o Athena não reescreve nem modifica arquivos existentes. Em vez disso, ele grava as linhas como um ou mais novos arquivos. Como tabelas com muitos arquivos pequenos resultam em desempenho inferior de consultas, e operações de gravação e leitura, como `PutObject` e `GetObject`, resultam em custos mais altos no Amazon S3, considere as seguintes opções ao usar `INSERT INTO`:

- Execute operações `INSERT INTO` com menor frequência em lotes maiores de linhas.
- Para grandes volumes de ingestão de dados, considere usar um serviço como o Amazon Data Firehose.
- Evite completamente o uso de `INSERT INTO`. Em vez disso, acumule linhas em arquivos maiores e faça o upload deles diretamente para o Amazon S3, onde poderão ser consultados pelo Athena.

Localizar arquivos órfãos

Se uma instrução `INSERT INTO` ou `CTAS` falhar, os dados órfãos poderão ser deixados no local de dados e ser lidos em consultas subsequentes. Para localizar arquivos órfãos para inspeção ou exclusão, é possível usar o arquivo do manifesto de dados que o Athena oferece para rastrear a lista de arquivos a serem gravados. Para obter mais informações, consulte Identificar os arquivos de saída das consultas e DataManifestLocation.

INSERT INTO...SELECT

Especifica a consulta a ser executada em uma tabela, `source_table`, que determina as linhas a serem inseridas em uma segunda tabela, `destination_table`. Se a consulta `SELECT` especificar colunas na `source_table`, as colunas deverão corresponder precisamente àquelas na `destination_table`.

Para obter mais informações sobre consultas `SELECT`, consulte SELECT.

Resumo

```
INSERT INTO destination_table
SELECT select_query
FROM source_table_or_view
```

Exemplos

Selecione todas as linhas na tabela `vancouver_pageviews` e insira-as na tabela `canada_pageviews`:

```
INSERT INTO canada_pageviews
SELECT *
FROM vancouver_pageviews;
```

Selecione apenas as linhas na tabela `vancouver_pageviews` em que a coluna `date` tenha um valor entre `2019-07-01` e `2019-07-31`, e insira-as em `canada_july_pageviews`:

```
INSERT INTO canada_july_pageviews
SELECT *
FROM vancouver_pageviews
WHERE date
    BETWEEN date '2019-07-01'
        AND '2019-07-31';
```

Selecione os valores nas colunas `city` e `state` na tabela `cities_world` somente dessas linhas com um valor `usa` na coluna `country` e insira-os nas colunas `city` e `state` na tabela `cities_usa`:

```
INSERT INTO cities_usa (city,state)
SELECT city,state
FROM cities_world
    WHERE country='usa'
```

INSERT INTO...VALUES

Insere linhas em uma tabela existente especificando colunas e valores. As colunas especificadas e os tipos de dados associados devem corresponder precisamente às colunas e aos tipos de dados na tabela de destino.

> **Important**
>
> Não recomendamos inserir linhas com `VALUES` porque o Athena gera arquivos para cada operação `INSERT`. Isso pode fazer com que muitos arquivos pequenos sejam criados e degradem a performance de consulta da tabela. Para identificar arquivos que uma consulta `INSERT` cria, examine o arquivo manifesto de dados. Para obter mais informações, consulte Trabalhar com resultados de consultas, consultas recentes e arquivos de saída.

Resumo

```
INSERT INTO destination_table [(col1,col2,...)]
VALUES (col1value,col2value,...)[,
       (col1value,col2value,...)][,
       ...]
```

Exemplos

Nos exemplos a seguir, a tabela de cidades tem três colunas: `id`, `city`, `state`, `state_motto`. A coluna `id` é do tipo `INT`, e todas as outras colunas são do tipo `VARCHAR`.

Insira uma única linha na tabela `cities`, com todos os valores da coluna especificados:

```
INSERT INTO cities
VALUES (1,'Lansing','MI','Si quaeris peninsulam amoenam circumspice')
```

Insira duas linhas na tabela `cities`:

```
INSERT INTO cities
VALUES (1,'Lansing','MI','Si quaeris peninsulam amoenam circumspice'),
       (3,'Boise','ID','Esto perpetua')
```

DELETE

Exclui linhas em uma tabela do Apache Iceberg. `DELETE` é transacional e é compatível somente com tabelas do Apache Iceberg.

Resumo

Para excluir as linhas de uma tabela do Iceberg, use a sintaxe a seguir.

```
DELETE FROM [db_name.]table_name [WHERE predicate]
```

Para obter mais informações e exemplos, consulte a seção DELETE da Atualizar dados nas tabelas Iceberg.

UPDATE

Atualiza linhas em uma tabela do Apache Iceberg. UPDATE é transacional e é compatível somente com tabelas do Apache Iceberg.

Resumo

Para atualizar as linhas em uma tabela do Iceberg, use a sintaxe a seguir.

```
UPDATE [db_name.]table_name SET xx=yy[,...] [WHERE predicate]
```

Para obter mais informações e exemplos, consulte a seção UPDATE da Atualizar dados nas tabelas Iceberg.

MERGE INTO

Atualiza, exclui ou insere linhas de forma condicional em uma tabela do Apache Iceberg. Uma única instrução pode combinar ações de atualização, exclusão e inserção.

> **Note**
>
> MERGE INTO é transacional e é compatível somente com tabelas do Apache Iceberg na versão 3 do mecanismo do Athena.

Resumo

Para atualizar, excluir ou inserir linhas de forma condicional em uma tabela do Iceberg, use a sintaxe a seguir.

```
MERGE INTO target_table [ [ AS ]  target_alias ]
USING { source_table | query } [ [ AS ] source_alias ]
ON search_condition
```

```
when_clause [...]
```

A *when_clause* corresponde a uma das seguintes:

```
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 é compatível com um número arbitrário de cláusulas WHEN com diferentes condições MATCHED. As cláusulas de condição executam as operações DELETE, UPDATE ou INSERT na primeira cláusula WHEN selecionada pelo estado MATCHED e pela condição de correspondência.

Para cada linha de origem, as cláusulas WHEN são processadas por ordem. Somente a primeira cláusula WHEN correspondente é executada. As cláusulas subsequentes são ignoradas. Um erro de usuário é gerado quando uma única linha da tabela de destino corresponde a mais de uma linha de origem.

Se uma linha de origem não corresponder a nenhuma cláusula WHEN e não houver uma cláusula WHEN NOT MATCHED, a linha de origem será ignorada.

Nas cláusulas WHEN que têm operações UPDATE, as expressões de valor da coluna podem se referir a qualquer campo de destino ou de origem. No caso de NOT MATCHED, as expressões INSERT podem se referir a qualquer campo de origem.

Exemplo

O exemplo a seguir mescla linhas da segunda tabela com a primeira tabela, se as linhas não existirem na primeira tabela. As colunas listadas na cláusula VALUES devem ser prefixadas pelo alias da tabela de origem. As colunas de destino listadas na cláusula INSERT não devem ter esse prefixo.

```
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)
```

Para obter mais exemplos de `MERGE INTO`, consulte Atualizar dados nas tabelas Iceberg.

OPTIMIZE

Otimize as linhas em uma tabela do Apache Iceberg ao gravar novamente arquivos de dados em um layout mais otimizado com base em no tamanho e no número de arquivos excluídos associados.

> **Note**
>
> `OPTIMIZE` é transacional e é compatível somente para tabelas do Apache Iceberg.

Sintaxe

O resumo da sintaxe a seguir mostra como otimizar o layout de dados para uma tabela Iceberg.

```
OPTIMIZE [db_name.]table_name REWRITE DATA USING BIN_PACK
  [WHERE predicate]
```

> **Note**
>
> Somente colunas de partição são permitidas no *`predicado`* da cláusula `WHERE`. Especificar uma coluna sem partição fará com que a consulta falhe.

A ação de compactação é cobrada pela quantidade de dados verificados durante o processo de regravação. A ação `REWRITE DATA` usa predicados para selecionar arquivos que contenham linhas iguais. Se alguma linha no arquivo corresponder ao predicado, o arquivo será selecionado para otimização. Assim, para controlar o número de arquivos afetados pela operação de compactação, você pode especificar uma cláusula `WHERE`.

Configurar propriedades de compactação

Para controlar o tamanho dos arquivos a serem selecionados para compactação e o tamanho do arquivo resultante após a compactação, você pode usar parâmetros de propriedade de tabela. Você

pode usar o comando ALTER TABLE SET PROPERTIES para configurar as propriedades de tabela a seguir.

Recursos adicionais do

Otimizar tabelas Iceberg

VACUUM

A instrução `VACUUM` realiza a manutenção da tabela para as tabelas do Apache Iceberg ao remover os arquivos de dados que não são mais necessários.

> **Note**
>
> `VACUUM` é transacional e é compatível somente com tabelas do Apache Iceberg na versão 3 do mecanismo do Athena.

A execução da instrução `VACUUM` em tabelas do Iceberg é recomendada para remover arquivos de dados que não são mais relevantes e reduzir o tamanho dos metadados e o consumo de armazenamento. Como a instrução `VACUUM` faz chamadas de API para o Amazon S3, as cobranças se aplicam às solicitações associadas ao Amazon S3.

> **Warning**
>
> Se você executar uma operação de expiração de snapshot, não poderá mais fazer viagens no tempo para snapshots expirados.

Resumo

Para remover os arquivos de dados que não são mais necessários para uma tabela do Iceberg, use a sintaxe a seguir.

```
VACUUM [database_name.]target_table
```

Para executar `VACUUM` em uma tabela com um nome que comece com um sublinhado (por exemplo, `_mytable`), coloque o nome da tabela entre acentos maiúsculos, como no exemplo a seguir. Se

você prefixar o nome da tabela com um nome de banco de dados, não coloque o nome do banco de dados entre aspas. Observe que aspas duplas não funcionarão no lugar dos acentos.

Esse comportamento é específico do `VACUUM`. As instruções `CREATE` e `INSERT INTO` não exigem acentos graves para nomes de tabelas que começam com sublinhados.

```
VACUUM `_mytable`
VACUUM my_database.`_mytable`
```

Observe também que `VACUUM` espera que os dados do Iceberg estejam em uma pasta do Amazon S3 em vez de em um bucket do Amazon S3. Por exemplo, se seus dados do Iceberg estiverem em `s3://DOC-EXAMPLE-BUCKET/` em vez de `s3://DOC-EXAMPLE-BUCKET/myicebergfolder/`, a instrução `VACUUM` falhará com a mensagem de erro GENERIC_INTERNAL_ERROR: Path missing in file system location: `s3://DOC-EXAMPLE-BUCKET`.

Operações realizadas

`VACUUM` realiza as seguintes operações:

- Remove os snapshots que são mais antigos do que o tempo especificado pela propriedade de tabela `vacuum_max_snapshot_age_seconds`. Por padrão, essa propriedade é definida para 432 mil segundos (cinco dias).
- Remove os snapshots que não estão dentro do período de retenção e que excedem o número especificado pela propriedade de tabela `vacuum_min_snapshots_to_keep`. O padrão é um.

  É possível especificar essas propriedades de tabela em sua instrução `CREATE TABLE`. Após a criação da tabela, é possível usar a instrução ALTER TABLE SET PROPERTIES para atualizá-la.
- Remove todos os metadados e os arquivos de dados inacessíveis como resultado da remoção do snapshot. Você pode configurar o número de arquivos de metadados antigos a serem retidos definindo a propriedade da tabela `vacuum_max_metadata_files_to_keep`. O valor padrão é 100.
- Remove arquivos órfãos que são mais antigos do que o tempo especificado na propriedade de tabela `vacuum_max_snapshot_age_seconds`. Arquivos órfãos corresponde a arquivos no diretório de dados da tabela que não fazem parte do estado da tabela.

Para obter mais informações sobre como criar e gerenciar tabelas do Apache Iceberg no Athena, consulte Criar tabelas Iceberg e Gerenciar tabelas Iceberg.

Usar EXPLAIN e EXPLAIN ANALYZE no Athena

A instrução `EXPLAIN` mostra o plano de execução lógico ou distribuído de uma instrução SQL especificada ou valida a instrução SQL. Você pode gerar os resultados no formato de texto ou em um formato de dados para renderização em gráfico.

> **Note**
>
> É possível visualizar representações gráficas de planos lógicos e distribuídos para suas consultas no console do Athena sem usar a sintaxe `EXPLAIN`. Para ter mais informações, consulte Visualizar planos de execução para consultas SQL.

O `EXPLAIN ANALYZE` mostra o plano de execução distribuído de uma instrução SQL especificada e o custo computacional de cada operação em uma consulta SQL. Você pode gerar os resultados no formato de texto ou JSON.

Considerações e limitações

As instruções `EXPLAIN` e `EXPLAIN ANALYZE` no Athena têm as limitações a seguir.

- Como as consultas `EXPLAIN` não verificam os dados, o Athena não cobra por elas. Entretanto, como as consultas `EXPLAIN` fazem chamadas ao AWS Glue para recuperar metadados de tabela, poderá haver cobranças do Glue se as chamadas ultrapassarem o limite do nível gratuito do Glue.
- Como consultas `EXPLAIN ANALYZE` são executadas, elas verificam os dados, e o Athena cobra pela quantidade de dados verificados.
- As informações de filtragem de linha ou de célula definidas no Lake Formation e as informações de estatísticas de consulta não são mostradas na saída de `EXPLAIN` e de `EXPLAIN ANALYZE`.

Sintaxe de EXPLAIN

```
EXPLAIN [ ( option [, ...]) ] statement
```

opção pode ser uma das seguintes:

```
FORMAT { TEXT | GRAPHVIZ | JSON }
TYPE { LOGICAL | DISTRIBUTED | VALIDATE | IO }
```

Se a opção `FORMAT` não for especificada, o padrão de saída será o formato `TEXT`. O tipo `IO` mostra informações sobre as tabelas e os esquemas que a consulta lê. O `IO` é compatível apenas com o mecanismo do Athena versão 2 e somente pode ser retornado no formato JSON.

Sintaxe de EXPLAIN ANALYZE

Além da saída incluída em `EXPLAIN`, a saída de `EXPLAIN ANALYZE` também inclui estatísticas de runtime para a consulta especificada, como o uso da CPU, o número de linhas da entrada e o número de linhas da saída.

```
EXPLAIN ANALYZE [ ( option [, ...]) ] statement
```

opção pode ser uma das seguintes:

```
FORMAT { TEXT | JSON }
```

Se a opção `FORMAT` não for especificada, o padrão de saída será o formato `TEXT`. Porque todas as consultas para `EXPLAIN ANALYZE` são `DISTRIBUTED`, a opção `TYPE` não está disponível para `EXPLAIN ANALYZE`.

A *instrução* pode ser uma das seguintes:

```
SELECT
CREATE TABLE AS SELECT
INSERT
UNLOAD
```

Exemplos de EXPLAIN

Os exemplos a seguir representam a progressão de `EXPLAIN`, do mais simples ao mais complexo.

Exemplo 1 de EXPLAIN: usar a instrução EXPLAIN para mostrar um plano de consulta no formato de texto

No exemplo a seguir, `EXPLAIN` mostra o plano de execução de uma consulta `SELECT` nos logs do Elastic Load Balancing. O formato é o padrão para saída de texto.

```
EXPLAIN
```

```
SELECT
   request_timestamp,
   elb_name,
   request_ip
FROM sampledb.elb_logs;
```

Resultados

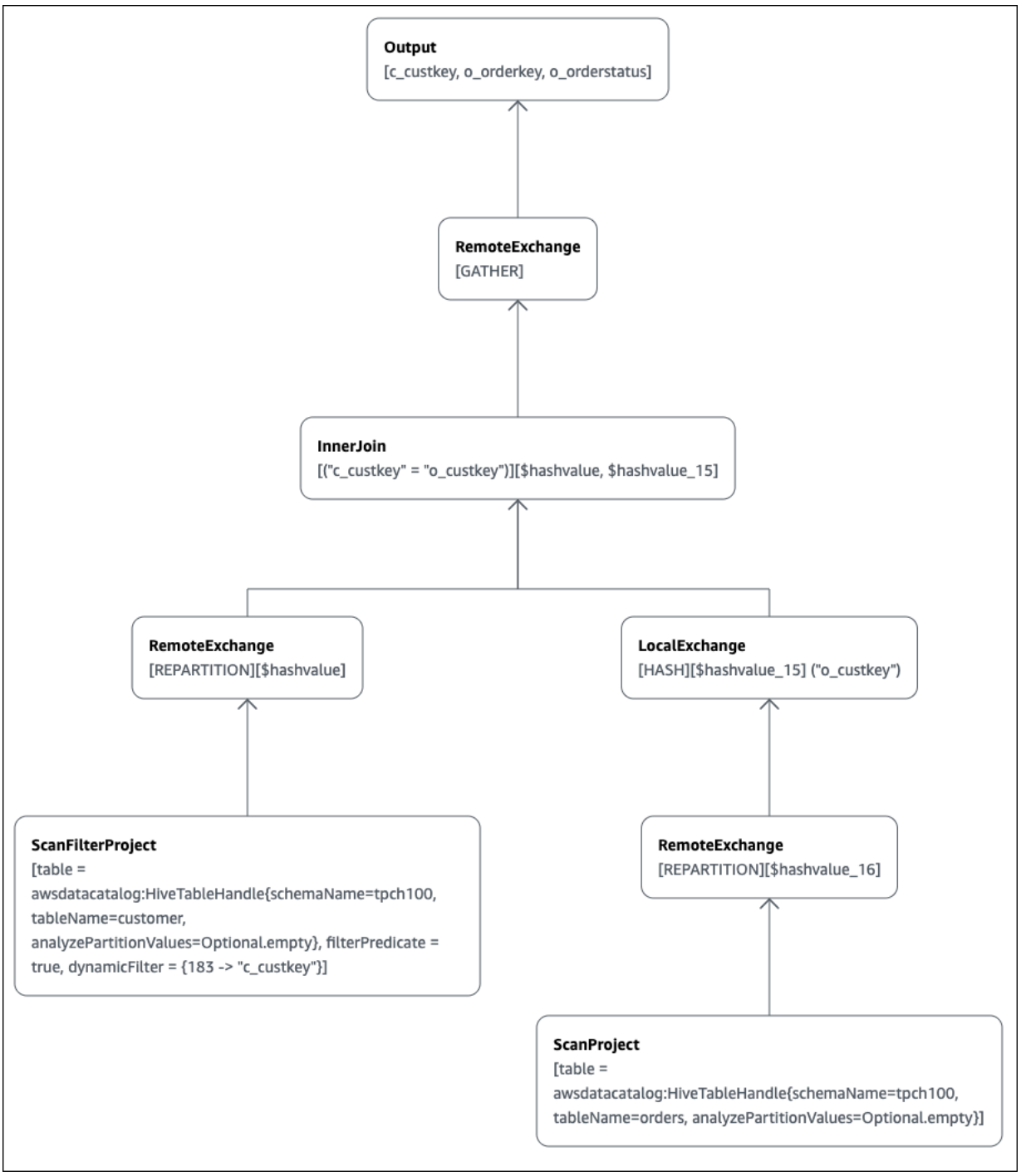
```
- Output[request_timestamp, elb_name, request_ip] => [[request_timestamp, elb_name,
 request_ip]]
    - RemoteExchange[GATHER] => [[request_timestamp, elb_name, request_ip]]
        - TableScan[awsdatacatalog:HiveTableHandle{schemaName=sampledb,
 tableName=elb_logs,
analyzePartitionValues=Optional.empty}] => [[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
```

Exemplo 2 de EXPLAIN: representar um plano de consulta graficamente

É possível utilizar o console do Athena para representar um plano de consulta graficamente para você. Insira uma instrução SELECT como a seguinte no editor de consultas do Athena e escolha EXPLAIN.

```
SELECT
      c.c_custkey,
      o.o_orderkey,
      o.o_orderstatus
   FROM tpch100.customer c
   JOIN tpch100.orders o
       ON c.c_custkey = o.o_custkey
```

A página Explain (Explicar) do editor de consultas do Athena é aberta e mostra um plano distribuído e um plano lógico para a consulta. O gráfico a seguir mostra o plano lógico do exemplo.



Important

Alguns filtros de partição podem não estar visíveis no gráfico de árvore do operador aninhado, mesmo que o Athena os aplique à sua consulta. Para verificar o efeito desses filtros, execute `EXPLAIN` ou `EXPLAIN ANALYZE` na sua consulta e visualize os resultados.

Para obter mais informações sobre como usar os recursos de gráfico do plano de consulta no console do Athena, consulte Visualizar planos de execução para consultas SQL.

Exemplo 3 de EXPLAIN: usar a instrução EXPLAIN para verificar a redução da partição

Quando você usa um predicado de filtragem em uma chave particionada para consultar uma tabela particionada, o mecanismo de consulta aplica o predicado à chave particionada para reduzir a quantidade de dados lidos.

O exemplo a seguir usa uma consulta `EXPLAIN` para verificar a remoção da partição de uma consulta `SELECT` em uma tabela particionada. Primeiro, a instrução `CREATE TABLE` cria a tabela `tpch100.orders_partitioned`. A tabela é particionada na coluna `o_orderdate`.

```
CREATE TABLE `tpch100.orders_partitioned`(
  `o_orderkey` int,
  `o_custkey` int,
  `o_orderstatus` string,
  `o_totalprice` double,
  `o_orderpriority` string,
  `o_clerk` string,
  `o_shippriority` int,
  `o_comment` string)
PARTITIONED BY (
  `o_orderdate` string)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET/<your_directory_path>/'
```

A tabela `tpch100.orders_partitioned` tem várias partições em `o_orderdate`, como mostrado pelo comando `SHOW PARTITIONS`.

```
SHOW PARTITIONS tpch100.orders_partitioned;

o_orderdate=1994
o_orderdate=2015
o_orderdate=1998
o_orderdate=1995
```

```
o_orderdate=1993
o_orderdate=1997
o_orderdate=1992
o_orderdate=1996
```

A consulta EXPLAIN a seguir verifica a remoção da partição com base na instrução SELECT especificada.

```
EXPLAIN
SELECT
   o_orderkey,
   o_custkey,
   o_orderdate
FROM tpch100.orders_partitioned
WHERE o_orderdate = '1995'
```

Resultados

```
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
```

O texto em negrito no resultado mostra que o predicado o_orderdate = '1995' foi aplicado a PARTITION_KEY.

Exemplo 4 de EXPLAIN: usar uma consulta EXPLAIN para verificar a ordem e o tipo de junção

A consulta EXPLAIN a seguir verifica o tipo e a ordem de junção da instrução SELECT. Use uma consulta como esta para examinar o uso da memória de consulta para que você possa reduzir as chances de receber um erro EXCEEDED_LOCAL_MEMORY_LIMIT.

```
EXPLAIN (TYPE DISTRIBUTED)
   SELECT
      c.c_custkey,
```

```
      o.o_orderkey,
      o.o_orderstatus
   FROM tpch100.customer c
   JOIN tpch100.orders o
       ON c.c_custkey = o.o_custkey
   WHERE c.c_custkey = 123
```

Resultados

```
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
```

A consulta de exemplo foi otimizada em uma junção cruzada para uma performance melhor. Os resultados mostram que `tpch100.orders` será distribuído como o tipo de distribuição `BROADCAST`. Isso indica que a tabela `tpch100.orders` será distribuída para todos os nós que executam a operação de junção. O tipo de distribuição `BROADCAST` exige que todos os resultados filtrados da tabela `tpch100.orders` estejam dentro da capacidade de memória de cada nó que executa a operação de junção.

No entanto, a tabela `tpch100.customer` é menor que `tpch100.orders`. Como `tpch100.customer` requer menos memória, você pode reescrever a consulta como `BROADCAST tpch100.customer` em vez de `tpch100.orders`. Desse modo, a consulta tem menos chance de receber o erro `EXCEEDED_LOCAL_MEMORY_LIMIT`. Essa estratégia considera os seguintes pontos:

- `tpch100.customer.c_custkey` é exclusivo na tabela `tpch100.customer`.
- Existe um relacionamento de mapeamento um para muitos entre `tpch100.customer` e `tpch100.orders`.

O exemplo a seguir mostra a consulta reescrita.

```
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
```

Exemplo 5 de EXPLAIN: usar uma consulta EXPLAIN para remover predicados sem efeito

Você pode usar uma consulta `EXPLAIN` para verificar a eficácia dos predicados de filtragem. Você pode usar os resultados para remover os predicados que não têm efeito, como no exemplo a seguir.

```
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
```

Resultados

```
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
```

O `filterPredicate` nos resultados mostra que o otimizador mesclou os três predicados originais em dois predicados e alterou a ordem de aplicação deles.

```
filterPredicate = (("c_custkey" = 1500) AND ("c_custkey" = CAST(("random"() * 1E3) AS
 int)))
```

Como os resultados mostram que o predicado `AND c.c_custkey BETWEEN 1000 AND 2000` não tem efeito, você pode removê-lo sem alterar os resultados da consulta.

Para obter informações sobre os termos usados nos resultados das consultas `EXPLAIN`, veja Noções básicas sobre os resultados da instrução EXPLAIN do Athena.

Exemplos de EXPLAIN ANALYZE

Os exemplos a seguir mostram exemplos de consultas e saídas de `EXPLAIN ANALYZE`.

Exemplo 1 de EXPLAIN ANALYZE: usar EXPLAIN ANALYZE para mostrar um plano de consulta e o custo computacional em formato de texto

No exemplo a seguir, `EXPLAIN ANALYZE` mostra o plano de execução e os custos computacionais de uma consulta `SELECT` em logs do CloudFront. O formato é o padrão para saída de texto.

```
EXPLAIN ANALYZE SELECT FROM cloudfront_logs LIMIT 10
```

Resultados

```
 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
```

Exemplo 2 de EXPLAIN ANALYZE: usar EXPLAIN ANALYZE para mostrar um plano de consulta em formato JSON

O exemplo a seguir mostra o plano de execução e os custos computacionais de uma consulta SELECT em logs do CloudFront. O exemplo especifica JSON como formato de saída.

```
EXPLAIN ANALYZE (FORMAT JSON) SELECT * FROM cloudfront_logs LIMIT 10
```

Resultados

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

Recursos adicionais do

Para obter mais informações, consulte os recursos a seguir.

- Noções básicas sobre os resultados da instrução EXPLAIN do Athena
- Visualizar planos de execução para consultas SQL
- Visualizar estatísticas e detalhes de execução para consultas concluídas
- Documentação Trino `EXPLAIN`
- Documentação Trino `EXPLAIN ANALYZE`

- Otimize o desempenho de consultas federadas usando EXPLAIN e EXPLAIN ANALYZE no Amazon Athena no AWSBlog de Big Data.

Noções básicas sobre os resultados da instrução EXPLAIN do Athena

Esse tópico apresenta uma rápida orientação sobre os termos operacionais usados nos resultados da instrução `EXPLAIN` do Athena.

Tipos de saída da instrução EXPLAIN

As saídas da instrução `EXPLAIN` podem ser de um destes dois tipos:

- Plano lógico: mostra o plano lógico que o mecanismo SQL usa para executar uma instrução. A sintaxe para essa opção é `EXPLAIN` ou `EXPLAIN (TYPE LOGICAL)`.
- Plano distribuído: mostra um plano de execução em um ambiente distribuído. A saída mostra fragmentos, que são os estágios de processamento. Cada fragmento do plano é processado por um ou mais nós. Os dados podem ser trocados entre os nós que processam os fragmentos. A sintaxe para essa opção é `EXPLAIN (TYPE DISTRIBUTED)`.

  Na saída de um plano de distribuição, os fragmentos (estágios de processamento) são indicados por `Fragment` *`number`* [*`fragment_type`*], em que *`number`* é um número inteiro baseado em zero e *`fragment_type`* especifica como os nós executam o fragmento. Os tipos de fragmento, que mostram insights do layout da troca de dados, estão descritos na tabela a seguir.

  Tipos de fragmento do plano distribuído

Tipo de fragmento	Descrição
`SINGLE`	O fragmento é executado em um único nó.
`HASH`	O fragmento é executado em um número fixo de nós. Os dados de entrada são distribuídos usando uma função hash.
`ROUND_ROBIN`	O fragmento é executado em um número fixo de nós. Os dados de entrada são distribuídos como “round-robin”.
`BROADCAST`	O fragmento é executado em um número fixo de nós. Os dados de entrada são transmitidos para todos os nós.

Tipo de fragmento	Descrição
SOURCE	O fragmento é executado nos nós onde as divisões de entrada são acessadas.

Exchange

Os termos relacionados à troca descrevem como os dados são trocados entre os nós de processamento. As transferências podem ser locais ou remotas.

LocalExchange [***exchange_type***]

Transfere os dados localmente nos nós de processamento para estágios diferentes de uma consulta. O valor de *exchange_type* pode ser um dos tipos de troca lógicos ou distribuídos, conforme descrito mais adiante nesta seção.

RemoteExchange [***exchange_type***]

Transfere os dados entre os nós de processamento para estágios diferentes de uma consulta. O valor de *exchange_type* pode ser um dos tipos de troca lógicos ou distribuídos, conforme descrito mais adiante nesta seção.

Tipos de troca lógicos

Os tipos de troca a seguir descrevem as ações executadas durante a fase de troca de um plano lógico.

- **GATHER**: um único nó de processamento combina a saída de todos os outros nós de processamento. Por exemplo, o último estágio de uma consulta selecionada coleta os resultados de todos os nós e grava esses resultados no Amazon S3.
- **REPARTITION**: envia os dados da linha para um operador específico com base no esquema de particionamento que deverá ser aplicado ao próximo operador.
- **REPLICATE**: copia os dados da linha para todos os operadores.

Tipos de troca distribuídos

Os tipos de troca a seguir indicam o layout dos dados quando eles são trocados entre os nós em um plano distribuído.

- **`HASH`**: a troca distribui os dados para vários destinos usando uma função hash.
- **`SINGLE`**: a troca distribui os dados para um único destino.

Verificação

Os termos a seguir descrevem como os dados são verificados durante uma consulta.

TableScan

Verifica os dados de origem de uma tabela do Amazon S3 ou de um conector do Apache Hive e aplica a remoção de partição gerada do predicado de filtro.

ScanFilter

Verifica os dados de origem de uma tabela do Amazon S3 ou de um conector do Hive e aplica a remoção de partição gerada do predicado de filtro e dos outros predicados de filtro não aplicados por meio da remoção de partição.

ScanFilterProject

Primeiramente, verifica os dados de origem de uma tabela do Amazon S3 ou de um conector do Hive e aplica a remoção de partição gerada do predicado de filtro e dos outros predicados de filtro não aplicados por meio da remoção de partição. Na sequência, modifica o layout da memória dos dados de saída para uma nova projeção a fim de melhorar a performance dos estágios posteriores.

Ingressar

Faz a junção dos dados entre duas tabelas. É possível categorizar as junções por tipo de junção e por tipo de distribuição.

Tipos de junção

Os tipos de junção definem como é feita a operação de junção.

CrossJoin: gera o produto cartesiano de duas tabelas unidas.

InnerJoin: seleciona os registros que têm valores correspondentes nas duas tabelas.

LeftJoin: seleciona todos os registros da tabela esquerda e os registros correspondentes da tabela direita. Se não houver nenhuma correspondência, o resultado no lado direito será NULL.

RightJoin: seleciona todos os registros da tabela direita e os registros correspondentes da tabela esquerda. Se não houver nenhuma correspondência, o resultado no lado direito será NULL.

FullJoin: seleciona todos os registros em que há correspondência com os registros da tabela esquerda ou direita. A tabela unida contém todos os registros das duas tabelas e preenche as correspondências ausentes com NULL em um dos lados.

> **Note**
>
> Por motivos de performance, o mecanismo de consulta pode reescrever uma consulta de junção com um tipo de junção diferente para retornar os mesmos resultados. Por exemplo, uma consulta de junção interna com predicado em uma tabela pode ser reescrita como `CrossJoin`. Isso leva o predicado à fase de verificação da tabela para que menos dados sejam verificados.

Tipos de distribuição de junção

Os tipos de distribuição definem como os dados são trocados entre os nós de processamento quando a operação de junção é executada.

Particionada: as tabelas tanto esquerda quanto direita são particionadas por hash em todos os nós de processamento. A distribuição particionada consome menos memória em cada nó. A distribuição particionada pode ser muito mais lenta do que as junções replicadas. As junções particionadas são ideais para unir duas tabelas grandes.

Replicada: uma tabela é particionada por hash em todos os nós de processamento, e a outra tabela é replicada para todos os nós de processamento para executar a operação de junção. A distribuição replicada pode ser muito mais rápida do que as junções particionadas, mas consome mais memória em cada nó de processamento. Se a tabela replicada for muito grande, o nó de processamento poderá receber um erro de memória insuficiente. As junções replicadas são ideais quando uma das tabelas unidas é pequena.

PREPARE

Cria uma instrução SQL com o nome `statement_name` para execução posterior. A instrução pode incluir parâmetros representados por pontos de interrogação. Para fornecer valores para os parâmetros e executar a instrução preparada, utilize EXECUTE.

Resumo

```
PREPARE statement_name FROM statement
```

A tabela a seguir descreve os parâmetros.

Parâmetro	Descrição
`statement_name`	O nome da instrução que será preparada. O nome deve ser exclusivo no grupo de trabalho.
`statement`	Uma consulta `SELECT`, `CTAS` ou `INSERT INTO`.

> **Note**
>
> O número máximo de instruções preparadas em um grupo de trabalho é mil.

Exemplos

O seguinte exemplo prepara uma consulta selecionada sem parâmetros.

```
PREPARE my_select1 FROM
SELECT * FROM nation
```

O seguinte exemplo prepara uma consulta selecionada que inclui parâmetros. Os valores para `productid` e `quantity` serão fornecido pela cláusula `USING` de uma instrução `EXECUTE`:

```
PREPARE my_select2 FROM
SELECT order FROM orders WHERE productid = ? and quantity < ?
```

O seguinte exemplo prepara uma consulta de inserção.

```
PREPARE my_insert FROM
INSERT INTO cities_usa (city, state)
SELECT city, state
FROM cities_world
WHERE country = ?
```

Recursos adicionais do

Executar consultas com instruções preparadas

EXECUTE

DEALLOCATE PREPARE

INSERT INTO

EXECUTE

Executa uma instrução preparada com o nome `statement_name`. Os valores dos parâmetros para os pontos de interrogação na declaração preparada estão definidos na cláusula `USING` em uma lista separada por vírgulas. Para criar uma instrução preparada, utilize PREPARE.

Resumo

```
EXECUTE statement_name [ USING parameter1[, parameter2, ... ] ]
```

Exemplos

O seguinte exemplo prepara e executa uma consulta sem parâmetros.

```
PREPARE my_select1 FROM
SELECT name FROM nation
EXECUTE my_select1
```

O seguinte exemplo prepara e executa uma consulta com um único parâmetro.

```
PREPARE my_select2 FROM
SELECT * FROM "my_database"."my_table" WHERE year = ?
EXECUTE my_select2 USING 2012
```

Isso é equivalente a:

```
SELECT * FROM "my_database"."my_table" WHERE year = 2012
```

O seguinte exemplo prepara e executa uma consulta com dois parâmetros.

```
PREPARE my_select3 FROM
SELECT order FROM orders WHERE productid = ? and quantity < ?
EXECUTE my_select3 USING 346078, 12
```

Recursos adicionais do

Executar consultas com instruções preparadas

PREPARE

INSERT INTO

DEALLOCATE PREPARE

Remove a instrução preparada com o nome especificado do conjunto de instruções preparadas no grupo de trabalho atual.

Resumo

```
DEALLOCATE PREPARE statement_name
```

Exemplos

O exemplo a seguir remove a instrução preparada `my_select1` do grupo de trabalho atual.

```
DEALLOCATE PREPARE my_select1
```

Recursos adicionais do

Executar consultas com instruções preparadas

PREPARE

UNLOAD

Grava os resultados da consulta de uma instrução `SELECT` no formato de dados especificado. Os formatos permitidos para `UNLOAD` são Apache Parquet, ORC, Apache Avro e JSON. CSV é o único

formato de saída compatível com o comando SELECT do Athena, mas você pode usar o comando UNLOAD, que é compatível com vários formatos de saída, para delimitar sua consulta SELECT e reescrever sua saída em um dos formatos compatíveis com UNLOAD.

Você pode usar a instrução CTAS para gerar dados em formatos diferentes do CSV, mas essa instrução também exige a criação de uma tabela no Athena. A instrução UNLOAD é útil para gerar os resultados de uma consulta SELECT em um formato não CSV, mas quando não há necessidade da tabela associada. Por exemplo, uma aplicação downstream pode exigir que os resultados de uma consulta SELECT estejam no formato JSON, e o Parquet ou o ORC pode ter uma performance melhor do que o CSV se você pretende usar os resultados da consulta SELECT para realizar outras análises.

Considerações e limitações

Quando você usar a instrução UNLOAD no Athena, lembre-se dos seguintes pontos:

- Sem ordenação global de arquivos: os resultados de UNLOAD são gravados em vários arquivos em paralelo. Se a consulta SELECT na instrução UNLOAD especificar uma ordem de classificação, o conteúdo de cada arquivo estará na ordem classificada, mas os arquivos não serão classificados entre eles.
- Dados órfãos não excluídos: em caso de falha, o Athena não tenta excluir os dados órfãos. Esse comportamento é o mesmo nas instruções CTAS e INSERT INTO.
- Partições máximas: o número máximo de partições que podem ser usadas com UNLOAD é 100.
- Arquivos manifesto e de metadados: o Athena gera um arquivo de metadados e um arquivo manifesto de dados para cada consulta UNLOAD. O manifesto rastreia os arquivos que a consulta gravou. Os dois arquivos são salvos no local dos resultados das consultas do Athena no Amazon S3. Para ter mais informações, consulte Identificar os arquivos de saída das consultas.
- Criptografia: os arquivos de saída de UNLOAD são criptografados de acordo com a configuração de criptografia usada no Amazon S3. Para definir a configuração para criptografar o resultado de UNLOAD, você pode usar a API EncryptionConfiguration.
- Instruções preparadas: UNLOAD pode ser usado com instruções preparadas. Para obter informações sobre instruções preparadas no Athena, consulte Utilizar consultas parametrizadas.
- Cotas de serviço: o UNLOAD usa as cotas de consulta DML. Para obter informações sobre cotas, consulte Service Quotas.
- Proprietário do bucket esperado: a configuração esperada do proprietário do bucket não se aplica ao local de destino do Amazon S3 especificado na consulta UNLOAD. A configuração esperada do

proprietário do bucket se aplica somente ao local de saída do Amazon S3 que você especificar para os resultados da consulta do Athena. Para ter mais informações, consulte Especificar um local para resultados de consultas usando o console do Athena.

Sintaxe

A instrução UNLOAD usa a sintaxe a seguir.

```
UNLOAD (SELECT col_name[, ...] FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/my_folder/'
WITH ( property_name = 'expression' [, ...] )
```

Exceto ao gravar nas partições, o destino TO deve especificar um local no Amazon S3 que não tenha dados. Antes de gravar no local especificado, a consulta UNLOAD verifica se o local do bucket está vazio. Como UNLOAD não grava dados no local especificado se ele já tiver dados, UNLOAD não substitui os dados existentes. Para reutilizar um local de bucket como destino para UNLOAD, exclua os dados do local do bucket e execute a consulta novamente.

Observe que, quando UNLOAD grava em partições, esse comportamento é diferente. Se você executar a mesma consulta UNLOAD várias vezes com a mesma instrução SELECT, o mesmo local TO e as mesmas partições, cada consulta UNLOAD descarrega os dados no Amazon S3 no local e nas partições especificadas.

Parâmetros

Veja abaixo os valores possíveis para *property_name*.

format = '***file_format***'

Obrigatório. Especifica o formato de arquivo da saída. Os valores possíveis para *file_format* são ORC, PARQUET, AVRO, JSON ouTEXTFILE.

compression = '***compression_format***'

Opcional. Essa opção é específica aos formatos ORC e Parquet. Para ORC, o padrão é zlib, e para Parquet, o padrão é gzip. Para obter informações sobre formatos de compactação compatíveis, consulte Suporte a compactação no Athena.

> **Note**
>
> Essa opção não se aplica ao formato `AVRO`. O Athena usa `gzip` para os formatos `JSON` e `TEXTFILE`.

compression_level = ***`compression_level`***

Opcional. O nível de compactação a ser usado para compactação ZSTD. Essa propriedade se aplica apenas à compressão ZSTD. Para ter mais informações, consulte Usar níveis de compressão ZSTD no Athena.

field_delimiter = '***`delimiter`***'

Opcional. Especifica um delimitador de campo de caractere único para arquivos em CSV, TSV e outros formatos de texto. O exemplo a seguir especifica o delimitador como vírgula.

```
WITH (field_delimiter = ',')
```

Atualmente, não há suporte para delimitadores de campo de vários caracteres. Se você não especificar um delimitador de campo, o caractere octal `\001` (^A) será usado.

partitioned_by = ARRAY[***`col_name`***[,...]]

Opcional. Uma lista matriz de colunas pela qual a saída é particionada.

> **Note**
>
> Em sua instrução `SELECT`, verifique se os nomes das colunas particionadas estão listados por último na lista de colunas.

Exemplos

O exemplo a seguir grava a saída de uma consulta `SELECT` no local do Amazon S3 `s3://DOC-EXAMPLE-BUCKET/unload_test_1/` usando o formato JSON.

```
UNLOAD (SELECT * FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/unload_test_1/'
WITH (format = 'JSON')
```

O exemplo a seguir grava a saída de uma consulta SELECT no formato Parquet usando a compactação Snappy.

```
UNLOAD (SELECT * FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/'
WITH (format = 'PARQUET',compression = 'SNAPPY')
```

O exemplo a seguir grava quatro colunas no formato de texto, com a saída particionada pela última coluna.

```
UNLOAD (SELECT name1, address1, comment1, key1 FROM table1)
TO 's3://DOC-EXAMPLE-BUCKET/ partitioned/'
WITH (format = 'TEXTFILE', partitioned_by = ARRAY['key1'])
```

O exemplo a seguir descarrega os resultados da consulta no local especificado usando o formato de arquivo Parquet, a compressão ZSTD e o nível 4 de compressão ZSTD.

```
UNLOAD (SELECT * FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/'
WITH (format = 'PARQUET', compression = 'ZSTD', compression_level = 4)
```

Recursos adicionais do

- Simplifique seus pipelines de ETL e ML usando o recurso Amazon Athena UNLOAD no AWSBlog de Big Data.

Funções no Amazon Athena

Para obter informações sobre as mudanças nas funções entre as versões do mecanismo do Athena, consulte Referência da versão do mecanismo do Athena. Para ver uma lista dos fusos horários que podem ser usados com o operador AT TIME ZONE, consulte Fusos horários suportados:.

Mecanismo Athena versão 3

As funções no mecanismo Athena versão 3 são baseadas no Trino. Para obter informações sobre as funções, operadores e expressões do Trino, consulte Functions and operators (Funções e operadores) e as seguintes subseções na documentação do Trino.

- Aggregate

- Array
- Binário
- Bitwise
- Color (Cor)
- Comparação
- Condicional
- Conversão
- Data e hora
- Decimal
- Geoespacial
- Log de HyperLogLog
- Endereço IP
- JSON
- Lambda
- Lógico
- Machine learning
- Mapa
- Math (Matemática)
- Resumo quantil
- Expressão regular
- Sessão
- Definir resumo
- String
- Tabela
- Teradata
- Resumo em T
- URL
- UUID
- Window

Mecanismo do Athena versão 2

As funções no mecanismo do Athena versão 2 são baseadas no Presto 0.217. Para as funções geoespaciais no mecanismo do Athena versão 2, consulte Funções geoespaciais no mecanismo do Athena versão 2.

> **Note**
>
> A documentação específica da versão para as funções do Presto 0.217 não está mais disponível. Para obter informações sobre funções, operadores e expressões atuais do Presto, consulte Funções e operadores do Presto, ou visite os links das subcategorias nesta seção.

- Operadores lógicos
- Funções e operadores comparativos
- Expressões condicionais
- Funções de conversão
- Funções e operadores matemáticos
- Funções bitwise
- Funções e operadores decimais
- Funções e operadores de string
- Funções binárias
- Funções e operadores de data e hora
- Funções de expressões regulares
- Funções e operadores JSON
- Funções de URL
- Funções agregadas
- Funções de janela
- Funções de cor
- Funções e operadores de matriz
- Funções e operadores de mapa
- Expressões e funções do Lambda
- Funções de teradados

Fusos horários suportados:

É possível utilizar o operador `AT TIME ZONE` em uma instrução para especificar o fuso horário do carimbo de data/hora que será retornado, como no seguinte exemplo:

```
SELECT timestamp '2012-10-31 01:00 UTC' AT TIME ZONE 'America/Los_Angeles' AS la_time;
```

Resultados

```
la_time

2012-10-30 18:00:00.000 America/Los_Angeles
```

A lista a seguir contém os fusos horários que podem ser usados com o operador `AT TIME ZONE` no Athena. Para obter funções e exemplos adicionais relacionados a fuso horário, consulte Funções e exemplos de fuso horário.

```
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
```

Funções e exemplos de fuso horário

Veja a seguir algumas funções e exemplos adicionais relacionados a fuso horário.

- at_timezone(***timestamp***, ***zone***): retorna o valor de *timestamp* no horário local correspondente para *zone*.

  Exemplo

```
SELECT at_timezone(timestamp '2021-08-22 00:00 UTC', 'Canada/Newfoundland')
```

Resultado

```
2021-08-21 21:30:00.000 Canada/Newfoundland
```

- timezone_hour(***timestamp***): retorna a hora do deslocamento do fuso horário do timestamp como um `bigint`.

  Exemplo

  ```
  SELECT timezone_hour(timestamp '2021-08-22 04:00 UTC' AT TIME ZONE 'Canada/
  Newfoundland')
  ```

  Resultado

  ```
  -2
  ```

- timezone_minute(***timestamp***): retorna a hora do deslocamento do fuso horário do *timestamp* como um `bigint`.

  Exemplo

  ```
  SELECT timezone_minute(timestamp '2021-08-22 04:00 UTC' AT TIME ZONE 'Canada/
  Newfoundland')
  ```

  Resultado

  ```
  -30
  ```

- with_timezone(***timestamp***, ***zone***): retorna um timestamp com fuso horário dos valores *timestamp* e *zone* especificado.

  Exemplo

  ```
  SELECT with_timezone(timestamp '2021-08-22 04:00', 'Canada/Newfoundland')
  ```

  Resultado

```
2021-08-22 04:00:00.000 Canada/Newfoundland
```

Instruções DDL

Use as instruções DDL a seguir diretamente no Athena.

O mecanismo de consulta do Athena é baseado em parte na DDL do HiveQL.

O Athena não aceita todas as instruções DDL, e há algumas diferenças entre a DDL do HiveQL e do Athena. Para obter mais informações, consulte os tópicos de referência nesta seção e em DDL incompatível.

Tópicos

- DDL incompatível
- 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 incompatível

As seguintes instruções DDL não são aceitas no Athena:

- ALTER INDEX
- ALTER TABLE *table_name* ARCHIVE PARTITION
- ALTER TABLE *table_name* CLUSTERED BY
- ALTER TABLE *table_name* EXCHANGE PARTITION
- ALTER TABLE *table_name* NOT CLUSTERED
- ALTER TABLE *table_name* NOT SKEWED
- ALTER TABLE *table_name* NOT SORTED
- ALTER TABLE *table_name* NOT STORED AS DIRECTORIES
- ALTER TABLE *table_name* partitionSpec CHANGE COLUMNS
- ALTER TABLE *table_name* partitionSpec COMPACT
- ALTER TABLE *table_name* partitionSpec CONCATENATE
- ALTER TABLE *table_name* partitionSpec SET FILEFORMAT
- ALTER TABLE *table_name* SET SERDEPROPERTIES
- ALTER TABLE *table_name* SET SKEWED LOCATION
- ALTER TABLE *table_name* SKEWED BY
- ALTER TABLE *table_name* TOUCH
- ALTER TABLE *table_name* UNARCHIVE PARTITION

- COMMIT
- CREATE INDEX
- CRIAR PERFIL
- 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
- MOSTRAR ESTATÍSTICAS
- SHOW TRANSACTIONS
- START TRANSACTION

- UNLOCK DATABASE
- UNLOCK TABLE

ALTER DATABASE SET DBPROPERTIES

Cria uma ou mais propriedades para um banco de dados. O uso de `DATABASE` e `SCHEMA` é intercambiável. Eles são iguais.

Resumo

```
ALTER {DATABASE|SCHEMA} database_name
  SET DBPROPERTIES ('property_name'='property_value' [, ...] )
```

Parâmetros

SET DBPROPERTIES ('property_name'='property_value' [, ...]

Especifica uma propriedade ou propriedades para o banco de dados chamado `property_name` e estabelece o valor para cada uma das propriedades, respectivamente como `property_value`. Se `property_name` já existir, o valor anterior será substituído por `property_value`.

Exemplos

```
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

Adicione uma ou mais colunas a uma tabela existente. Quando a sintaxe opcional de `PARTITION` é usada, os metadados da partição são atualizados.

Resumo

```
ALTER TABLE table_name
  [PARTITION
```

```
   (partition_col1_name = partition_col1_value
   [,partition_col2_name = partition_col2_value][,...])]
  ADD COLUMNS (col_name data_type)
```

Parâmetros

PARTITION (partition_col_name = partition_col_value [,...])

Cria uma partição com as combinações de nome/valor de coluna que você especificar. Coloque `partition_col_value` entre aspas somente se o tipo de dados da coluna for uma string.

ADD COLUMNS (col_name data_type [,col_name data_type,...])

Adiciona colunas após colunas existentes, mas antes das colunas de partição.

Exemplos

```
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)
```

Observações

- Para ver uma nova coluna de tabela no painel de navegação do editor de consultas do Athena depois de executar `ALTER TABLE ADD COLUMNS`, atualize manualmente a lista de tabelas no editor e expanda a tabela outra vez.
- `ALTER TABLE ADD COLUMNS` não funciona em colunas com o tipo de dados `date`. Para contornar esse problema, use o tipo de dados `timestamp`.

ALTER TABLE ADD PARTITION

Cria uma ou mais colunas de partição para a tabela. Cada partição consiste em uma ou mais combinações de nome/valor de coluna distintas. Um diretório de dados à parte é criado para cada combinação especificada, o que pode melhorar a performance da consulta em algumas circunstâncias. As colunas particionadas não existem dentro da própria tabela de dados. Dessa

forma, se usar o nome de uma coluna com o mesmo nome de uma coluna na própria tabela, você receberá um erro. Para ter mais informações, consulte Particionar dados no Athena.

No Athena, uma tabela e suas partições devem usar os mesmos formatos de dados, mas os esquemas podem ser diferentes. Para ter mais informações, consulte Atualizações em tabelas com partições.

Para obter informações sobre as permissões no nível do recurso necessárias nas políticas do IAM (incluindo `glue:CreatePartition`), consulte Permissões da API do AWS Glue: referência de ações e recursos e Acesso granular a bancos de dados e tabelas no AWS Glue Data Catalog. Para obter informações sobre solução de problemas de permissões ao usar o Athena, consulte a seção Permissões do tópico Solução de problemas no Athena.

Resumo

```
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']
  [,...]
```

Parâmetros

Ao adicionar uma partição, você especifica um ou mais pares de nome/valor de coluna para a partição e o caminho do Amazon S3 onde residem os arquivos de dados dessa partição.

[IF NOT EXISTS]

Suprimirá o erro se uma partição com a mesma definição já existir.

PARTITION (partition_col_name = partition_col_value [,...])

Cria uma partição com as combinações de nome/valor de coluna que você especificar. Coloque `partition_col_value` entre caracteres de string somente se o tipo de dados da coluna for uma string.

[LOCATION 'location']

Especifica o diretório no qual armazenar a partição definida pela instrução anterior. A cláusula `LOCATION` é opcional quando os dados usam particionamento no estilo Hive (`pk1=v1/pk2=v2/pk3=v3`). Com o particionamento no estilo Hive, o URI completo do Amazon S3 é estruturado automaticamente com base na localização da tabela, nos nomes das chaves de partição e nos valores de chave de partição. Para ter mais informações, consulte Particionar dados no Athena.

Considerações

O Amazon Athena não impõe um limite específico ao número de partições que você pode adicionar em uma única instrução DDL `ALTER TABLE ADD PARTITION`. No entanto, se você precisar adicionar um número significativo de partições, considere dividir a operação em lotes menores para evitar possíveis problemas de desempenho. O exemplo a seguir usa comandos sucessivos para adicionar partições individualmente e usa `IF NOT EXISTS` para evitar a adição de duplicatas.

```
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')
```

Ao trabalhar com partições no Athena, lembre-se dos seguintes pontos:

- Embora o Athena ofereça suporte a consulta a tabelas do AWS Glue com 10 milhões de partições, o Athena não pode ler mais de 1 milhão de partições em uma única varredura.
- Para otimizar suas consultas e reduzir o número de partições verificadas, considere estratégias como remoção de partições ou uso de índices de partição.
- Se não estiver usando o AWS Glue Data Catalog, o número máximo de partições por tabela será 20.000. É possível solicitar um aumento da cota.

Para considerações adicionais sobre como trabalhar com partições no Athena, consulte Particionar dados no Athena.

Exemplos

O exemplo a seguir adiciona uma única partição a uma tabela para dados particionados no estilo Hive.

```
ALTER TABLE orders ADD
  PARTITION (dt = '2016-05-14', country = 'IN');
```

O exemplo a seguir adiciona múltiplas partições a uma tabela para dados particionados no estilo Hive.

```
ALTER TABLE orders ADD
  PARTITION (dt = '2016-05-31', country = 'IN')
  PARTITION (dt = '2016-06-01', country = 'IN');
```

Quando a tabela não serve para dados particionados no estilo Hive, a cláusula `LOCATION` é obrigatória e deve ser o URI completo do Amazon S3 para o prefixo que contém os dados da partição.

```
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/';
```

Para ignorar erros quando a partição já existe, use a cláusula `IF NOT EXISTS`, como no exemplo a seguir.

```
ALTER TABLE orders ADD IF NOT EXISTS
  PARTITION (dt = '2016-05-14', country = 'IN');
```

Arquivos de zero byte no formato **`_$folder$`**

Se você executar uma instrução `ALTER TABLE ADD PARTITION` e especificar erroneamente uma partição que já existe e uma localização incorreta do Amazon S3, serão criados arquivos de espaço reservado de zero byte do formato *`partition_value`*`_$folder$` no Amazon S3. Você precisa remover esses arquivos manualmente.

Para evitar que isso aconteça, use a sintaxe `ADD IF NOT EXISTS` na instrução `ALTER TABLE ADD PARTITION`, como no exemplo a seguir.

```
ALTER TABLE table_name ADD IF NOT EXISTS PARTITION […]
```

ALTER TABLE DROP PARTITION

Descarta uma ou mais partições especificadas para a tabela nomeada.

Resumo

```
ALTER TABLE table_name DROP [IF EXISTS] PARTITION (partition_spec) [, PARTITION
 (partition_spec)]
```

Parâmetros

[SE EXISTIR]

Suprime a mensagem de erro se a partição especificada não existir.

PARTIÇÃO (partition_spec)

Cada `partition_spec` especifica uma combinação de nome de coluna e valor no formato `partition_col_name = partition_col_value [,...]`.

Exemplos

```
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');
```

Observações

A instrução `ALTER TABLE DROP PARTITION` não fornece uma sintaxe única para descartar todas as partições de uma só vez nem suporta critérios de filtragem para especificar um intervalo de partições a serem eliminadas.

Como solução de contorno, use as ações GetPartitions e BatchDeletePartition da API do AWS Glue no desenvolvimento de scripts. A ação `GetPartitions` suporta expressões de filtro complexas como as de uma expressão `WHERE` em SQL. Depois de usar `GetPartitions` para criar uma lista filtrada das partições a serem descartadas, você pode usar a ação `BatchDeletePartition` para descartar as partições em lotes de 25.

Important

Devido a um problema conhecido, quando uma partição inválida é especificada para a instrução `ALTER TABLE DROP PARTITION` , todas as partições da tabela são descartadas no AWS Glue. Por exemplo, a instrução a seguir descartará todas as partições da tabela *`my_table`* mesmo que a partição especificada não exista. Como solução alternativa, certifique-se de inserir as informações da partição corretamente antes de executar a instrução `ALTER TABLE DROP PARTITION`.

```
ALTER TABLE my_table DROP IF EXISTS PARTITION(zzz='');
```

ALTER TABLE RENAME PARTITION

Renomeia um valor de partição.

Note

ALTER TABLE RENAME PARTITION não renomeia as colunas de partição. Para alterar o nome de uma coluna de partição, você pode usar o console do AWS Glue. Para obter mais informações, consulte Renomear uma coluna de partição no AWS Glue adiante neste documento.

Resumo

Para a tabela chamada `table_name`, renomeia o valor da partição especificado por `partition_spec` para o valor especificado por `new_partition_spec`.

```
ALTER TABLE table_name PARTITION (partition_spec) RENAME TO PARTITION
 (new_partition_spec)
```

Parâmetros

PARTIÇÃO (partition_spec)

Cada `partition_spec` especifica uma combinação de nome de coluna e valor no formato `partition_col_name = partition_col_value [,...]`.

Exemplos

```
ALTER TABLE orders
PARTITION (dt = '2014-05-14', country = 'IN') RENAME TO PARTITION (dt = '2014-05-15',
 country = 'IN');
```

Renomear uma coluna de partição no AWS Glue

Use o procedimento a seguir para renomear colunas de partição no console do AWS Glue.

Para renomear uma coluna de partição de tabela no console do AWS Glue

1. Faça login no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, selecione Tabelas.
3. Na página Tabelas, use a caixa de pesquisa Filtrar tabelas para localizar a tabela que você deseja alterar.
4. Na coluna Nome, escolha o link da tabela que você deseja alterar.
5. Na página de detalhes da tabela, na seção Esquema, siga um destes procedimentos:
   - Para fazer a alteração do nome no formato JSON, escolha Editar esquema como JSON.
   - Para alterar o nome diretamente, escolha Editar esquema. Esse procedimento escolhe Editar esquema.
6. Marque a caixa de seleção referente à coluna particionada que você deseja renomear e escolha Editar.
7. Na caixa de diálogo Editar entrada do esquema, em Nome, insira o novo nome para a coluna de partição.
8. Selecione Salvar como nova versão da tabela. Essa ação atualiza o nome da coluna da partição e preserva o histórico de evolução do esquema sem criar outra cópia física dos dados.
9. Para comparar as versões da tabela, na página de detalhes da tabela, escolha Ações e, em seguida, Comparar versões.

Recursos adicionais do

Para obter mais informações sobre particionamento, consulte Particionar dados no Athena.

ALTER TABLE REPLACE COLUMNS

Remove todas as colunas existentes de uma tabela criada com LazySimpleSerDe e as substitui pelo conjunto de colunas especificado. Quando a sintaxe opcional de `PARTITION` é usada, os metadados da partição são atualizados. Você também pode usar `ALTER TABLE REPLACE COLUMNS` para descartar as colunas especificando apenas as colunas que deseja manter.

Resumo

```
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, ...])
```

Parâmetros

PARTITION (partition_col_name = partition_col_value [,...])

Especifica uma partição com as combinações de nome/valor de coluna que você especificar. Coloque `partition_col_value` entre aspas somente se o tipo de dados da coluna for uma string.

REPLACE COLUMNS (col_name data_type [,col_name data_type,...])

Substitui as colunas existentes pelos nomes e tipos de dados de coluna especificados.

Observações

- Para ver a alteração nas colunas de tabela no painel de navegação do editor de consultas do Athena após executar `ALTER TABLE REPLACE COLUMNS`, pode ser necessário atualizar manualmente a tabela no editor e depois expandir a tabela outra vez.
- `ALTER TABLE REPLACE COLUMNS` não funciona em colunas com o tipo de dados `date`. Para contornar esse problema, use o tipo de dados `timestamp` na tabela.
- Observe que, mesmo que se você for substituir apenas uma coluna, a sintaxe deverá ser `ALTER TABLE` *`table-name`* `REPLACE COLUMNS`, com columns no plural. Você deve especificar não apenas a coluna que deseja substituir, mas as colunas que deseja manter, do contrário, as colunas que não especificadas serão descartadas. Essa sintaxe e esse comportamento derivam da DDL do Apache Hive. Para referência, consulte Add/Replace Columns (Adicionar/substituir colunas) na documentação do Apache.

Exemplo

No exemplo a seguir, a tabela `names_cities`, que foi criada usando o LazySimpleSerDe, tem três colunas chamadas `col1`, `col2` e `col3`. Todas as colunas são do tipo `string`. Para mostrar as colunas na tabela, o comando a seguir usa a instrução SHOW COLUMNS.

```
SHOW COLUMNS IN names_cities
```

Resultado da consulta:

```
col1
col2
col3
```

O comando `ALTER TABLE REPLACE COLUMNS` a seguir substitui os nomes das colunas por `first_name`,`last_name` e `city`. Os dados de origem subjacentes não são afetados.

```
ALTER TABLE names_cities
REPLACE COLUMNS (first_name string, last_name string, city string)
```

Para testar o resultado, `SHOW COLUMNS` é executado novamente.

```
SHOW COLUMNS IN names_cities
```

Resultado da consulta:

```
first_name
last_name
city
```

Outra maneira de mostrar os novos nomes de coluna é exibir a previsualização da tabela no editor de consultas do Athena ou executar o sua própria consulta `SELECT`.

ALTER TABLE SET LOCATION

Altera o local da tabela chamada `table_name` e, como opção, uma partição com `partition_spec`.

Resumo

```
ALTER TABLE table_name [ PARTITION (partition_spec) ] SET LOCATION 'new location'
```

Parâmetros

PARTIÇÃO (partition_spec)

Especifica a partição com parâmetros `partition_spec` cujo local você deseja alterar. O `partition_spec` especifica uma combinação nome/valor na forma `partition_col_name = partition_col_value`.

SET LOCATION 'new location'

Especifica o novo local, que deve ser um local do Amazon S3. Para obter informações sobre sintaxe, consulte Local da tabela no Amazon S3.

Exemplos

```
ALTER TABLE customers PARTITION (zip='98040', state='WA') SET LOCATION 's3://DOC-
EXAMPLE-BUCKET/custdata/';
```

ALTER TABLE SET TBLPROPERTIES

Adiciona propriedades de metadados personalizadas ou predefinidas a uma tabela e define seus valores atribuídos. Para ver as propriedades em uma tabela, use o comando SHOW TBLPROPERTIES.

As tabelas gerenciadas do Apache Hive não são permitidas, portanto, a definição de `'EXTERNAL'='FALSE'` não tem efeito.

Resumo

```
ALTER TABLE table_name SET TBLPROPERTIES ('property_name' = 'property_value' [ , ... ])
```

Parâmetros

SET TBLPROPERTIES ('property_name' = 'property_value' [, ...])

Especifica as propriedades de metadados a serem adicionadas como `property_name` e o valor para cada uma como `property value`. Se `property_name` já existir, o valor será definido como o `property_value` recém-especificado.

As propriedades de tabela predefinidas a seguir têm usos especiais.

Propriedade predefinida	Descrição
`classification`	Indica o tipo de dados de AWS Glue. Os valores possíveis são `csv`, `parquet`, `orc`, `avro` ou `json`. As tabelas criadas para o Athena no console do CloudTrail adicionam `cloudtrail` como um valor para a propriedade `classification` . Para obter mais informações, consulte a seção TBLPROPERTIES de CREATE TABLE.
`has_encrypted_data`	Indica se o conjunto de dados especificado por `LOCATION` está criptografado. Para obter mais informações, consulte a seção TBLPROPERTIES de CREATE TABLE e Criar tabelas com base em conjuntos de dados criptografados no Amazon S3.
`orc.compress`	Especifica o formato de compactação de dados no formato ORC. Para ter mais informações, consulte ORC SerDe.
`parquet.compression`	Especifica o formato de compactação de dados no formato Parquet. Para ter mais informações, consulte Parquet SerDe.
`write.compression`	Especifica o formato de compressão de dados nos formatos de arquivo de texto ou JSON. Para os formatos Parquet e ORC, use as propriedades `parquet.compression` e `orc.compress` , respectivamente.
`compression_level`	Especifica um nível de compressão a ser usado. Essa propriedade se aplica apenas à compressão ZSTD. Os valores possíveis são de 1 a 22. O valor padrão é 3. Para ter mais informações, consulte Usar níveis de compressão ZSTD no Athena.

Propriedade predefinida	Descrição
`projection.*`	As propriedades personalizadas usadas na projeção da partição que permitem que o Athena saiba quais padrões de partição esperar ao executar uma consulta em uma tabela. Para ter mais informações, consulte Projeção de partições com o Amazon Athena.
`skip.header.line.count`	Ignora os cabeçalhos nos dados quando você define uma tabela. Para ter mais informações, consulte Ignorar cabeçalhos.
`storage.location.template`	Especifica um modelo de caminho personalizado do Amazon S3 para partições projetadas. Para ter mais informações, consulte Configurar a projeção de partições.

Exemplos

O exemplo a seguir adiciona uma nota de comentário às propriedades da tabela.

```
ALTER TABLE orders
SET TBLPROPERTIES ('notes'="Please don't drop this table.");
```

O exemplo a seguir modifica a tabela `existing_table` para usar o formato de arquivo Parquet com compressão ZSTD nível 4.

```
ALTER TABLE existing_table
SET TBLPROPERTIES ('parquet.compression' = 'ZSTD', 'compression_level' = 4)
```

CREATE DATABASE

Cria um banco de dados. O uso de `DATABASE` e `SCHEMA` é intercambiável. Eles significam a mesma coisa.

> **Note**
>
> Para ver um exemplo de como criar um banco de dados, criar uma tabela e executar uma consulta `SELECT` na tabela do Athena, consulte Conceitos básicos.

Resumo

```
CREATE {DATABASE|SCHEMA} [IF NOT EXISTS] database_name
  [COMMENT 'database_comment']
  [LOCATION 'S3_loc']
  [WITH DBPROPERTIES ('property_name' = 'property_value') [, ...]]
```

Parâmetros

[IF NOT EXISTS]

Fará o erro ser suprimido se um banco de dados chamado `database_name` já existir.

[COMMENT database_comment]

Estabelece o valor de metadados para a propriedade de metadados interna chamada `comment` e o valor fornecido por você para `database_comment`. No AWS Glue, o conteúdo de `COMMENT` é gravado no campo `Description` das propriedades do banco de dados.

[LOCATION S3_loc]

Especifica o local onde arquivos de banco de dados e metastore existirão como `S3_loc`. O local deve ser um local do Amazon S3.

[WITH DBPROPERTIES ('property_name' = 'property_value') [, ...]]

Permite especificar propriedades de metadados personalizados para a definição do banco de dados.

Exemplos

```
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');
```

Visualizar as propriedades do banco de dados

Para visualizar as propriedades de um banco de dados criado no AWSDataCatalog usando `CREATE DATABASE`, você pode usar o comando `aws glue get-database` da AWS CLI, como no seguinte exemplo:

```
aws glue get-database --name <your-database-name>
```

Na saída do JSON, o resultado é semelhante ao seguinte:

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

Para obter mais informações sobre a AWS CLI, consulte o Manual do usuário da AWS Command Line Interface.

CREATE TABLE

Cria uma tabela com o nome e os parâmetros especificados por você.

> **Note**
>
> Esta página contém informações de referência resumidas. Para obter mais informações sobre como criar tabelas no Athena e um exemplo de instrução `CREATE TABLE`, consulte Criar tabelas no Athena. Para ver um exemplo de como criar um banco de dados, criar uma tabela e executar uma consulta `SELECT` na tabela do Athena, consulte Conceitos básicos.

Resumo

```
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 [, ...] ) ]
```

Parâmetros

EXTERNAL

Especifica que a tabela é baseada em um arquivo de dados subjacente existente no Amazon S3, no `LOCATION` que você especificar. Exceto ao criar tabelas do Iceberg, use sempre a palavra-chave `EXTERNAL`. Se você usar `CREATE TABLE` sem a palavra-chave `EXTERNAL` para tabelas que não são do Iceberg, o Athena emitirá um erro. Quando você cria uma tabela externa, os dados referenciados devem estar em conformidade com o formato padrão ou o formato especificado por você com as cláusulas `ROW FORMAT`, `STORED AS` e `WITH SERDEPROPERTIES`.

[IF NOT EXISTS]

Este parâmetro verifica se já existe uma tabela com o mesmo nome. Se existir, o parâmetro retornará `TRUE` e o Amazon Athena cancelará a ação `CREATE TABLE`. Como o cancelamento ocorre antes que o Athena chame o catálogo de dados, ele não emite um evento AWS CloudTrail.

[db_name.]table_name

Especifica um nome para a tabela a ser criada. O parâmetro `db_name` opcional especifica o banco de dados no qual a tabela existe. Se omitido, o banco de dados atual será assumido. Se o nome da tabela inclui números, coloque `table_name` entre aspas, por exemplo `"table123"`. Se `table_name` começar com um sublinhado, use acentos graves, por exemplo, `` `_mytable` ``. Os caracteres especiais (que não sejam sublinhado) não são compatíveis.

Os nomes de tabela do Athena não diferenciam maiúsculas de minúsculas. No entanto, se você trabalhar com o Apache Spark, o Spark exigirá nomes de tabela em letras minúsculas.

[(col_name data_type [COMMENT col_comment] [, ...])]

Especifica o nome de cada coluna a ser criada, além do tipo de dados da coluna. Os nomes de coluna não permitem caracteres especiais além de sublinhado (`_`). Se `col_name` começar com um sublinhado, coloque o nome da coluna entre acentos graves, por exemplo `` `_mycolumn` ``.

O valor `data_type` pode ser qualquer um dos seguintes:

- `boolean`: os valores são `true` e `false`.
- `tinyint`: um inteiro com sinal de 8 bits no formato de complemento de dois, com um valor mínimo de -2^7 e um valor máximo de 2^7-1.
- `smallint`: um inteiro com sinal de 16 bits no formato de complemento de dois, com um valor mínimo de -2^15 e um valor máximo de 2^15-1.
- `int`: nas consultas em Data Definition Language (DDL), como `CREATE TABLE`, use a palavra-chave `int` para representar um número inteiro. Em outras consultas, use a palavra-chave `integer`, em que `integer` é representado por um valor com sinal de 32 bits no formato de complemento de dois, com um valor mínimo de -2^31 e um valor máximo de 2^31-1. No driver JDBC, `integer` é retornado para garantir a compatibilidade com os aplicativos de análise de negócios.
- `bigint`: um inteiro com sinal de 64 bits no formato de complemento de dois, com um valor mínimo de -2^63 e um valor máximo de 2^63-1.
- `double`: um número com sinal de ponto flutuante de precisão dupla de 64 bits. O intervalo é de 4.94065645841246544e-324d a 1.79769313486231570e+308d, positivo ou negativo. `double` segue o padrão para aritmética de ponto flutuante do IEEE (IEEE 754).
- `float`: um número com sinal de ponto flutuante de precisão única de 32 bits. O intervalo é de 1.40129846432481707e-45 a 3.40282346638528860e+38, positivo ou negativo. `float`

segue o padrão para aritmética de ponto flutuante do IEEE (IEEE 754). Equivalente a `real` no Presto. No Athena, use `float` nas instruções DDL, como `CREATE TABLE`, e `real` nas funções SQL, como `SELECT CAST`. O crawler do AWS Glue retorna os valores em `float`, e o Athena converte os tipos `real` e `float` internamente (leia as notas de release 5 de junho de 2018).

- `decimal [ (precision, scale) ]`, onde *`precision`* é o número total de dígitos e *`scale`* (opcional) é o número de dígitos na parte fracionada, o padrão é 0. Por exemplo, use estas definições de tipo: `decimal(11,5)`, `decimal(15)`. O valor máximo para *`precisão`* é 38 e o valor máximo para *`escala`* é 38.

  Para especificar valores decimais como literais, como ao selecionar filas com um valor decimal específico em uma expressão de consulta DDL, especifique a definição de tipo `decimal` e liste o valor decimal como um literal (em aspas simples) na consulta, como neste exemplo: `decimal_value = decimal '0.12'`.
- `char`: os dados de caractere de comprimento fixo, com um tamanho especificado entre 1 e 255, como `char(10)`. Para obter mais informações, consulte CHAR Hive data type (Tipo de dado CHAR do Hive).
- `varchar`: os dados de caractere de comprimento variável, com um tamanho especificado entre 1 e 65535, como `varchar(10)`. Para obter mais informações, consulte VARCHAR Hive data type (Tipo de dado VARCHAR do Hive).
- `string`: um literal de string entre aspas simples ou duplas.

  > **Note**
  >
  > Os tipos de dados que não são de string não podem ser convertidos em `string` no Athena. Em vez disso, converta-os em `varchar`.

- `binary`: (para dados em Parquet)
- `date`: uma data no formato ISO, como *`YYYY`*`-`*`MM`*`-`*`DD`*. Por exemplo, `date '2008-09-15'`. Uma exceção é o OpenCSVSerDe, que usa o número de dias decorridos desde 1° de janeiro de 1970. Para ter mais informações, consulte OpenCSVSerDe para processar CSV.
- `timestamp`: instante de data e hora em um formato compatível com `java.sql.Timestamp` até uma resolução máxima de milissegundos, como *`yyyy`*`-`*`MM`*`-`*`dd HH`*`:`*`mm`*`:`*`ss`*`[.f...]`. Por exemplo, `timestamp '2008-09-15 03:04:05.324'`. Uma exceção é o OpenCSVSerDe, que usa os dados de `TIMESTAMP` no formato numérico UNIX (por exemplo, `1579059880000`). Para ter mais informações, consulte OpenCSVSerDe para processar CSV.

- `array` < data_type >
- `map` < primitive_type, data_type >
- `struct` < col_name : data_type [comment col_comment] [, ...] >

[COMMENT table_comment]

Cria a propriedade da tabela `comment` e a preenche com o `table_comment` especificado por você.

[PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]

Cria uma tabela particionada com uma ou mais colunas de partição que tenham `col_name`, `data_type` e `col_comment` especificados. A tabela pode ter uma ou mais partições, que consistem em uma combinação distinta de nome e valor de coluna. Um diretório de dados à parte é criado para cada combinação especificada, o que pode melhorar a performance da consulta em algumas circunstâncias. As colunas particionadas não existem na própria tabela de dados. Se você usar um valor para `col_name` que é o mesmo valor usado na coluna da tabela, obterá um erro. Para obter mais informações, consulte Particionar dados.

> **Note**
>
> Depois de criar uma tabela com partições, execute uma consulta que consista na cláusula MSCK REPAIR TABLE para atualizar metadados de partição, por exemplo, `MSCK REPAIR TABLE cloudfront_logs;`. Para partições que não são compatíveis com o Hive, use o ALTER TABLE ADD PARTITION a fim de carregar as partições para que você consiga consultar os dados.

[CLUSTERED BY (col_name, col_name, ...) INTO num_buckets BUCKETS]

Divide, com ou sem particionamento, os dados nas colunas `col_name` especificadas em subconjuntos de dados chamados buckets. O parâmetro `num_buckets` especifica o número de buckets que serão criados. A criação de buckets pode melhorar a performance de algumas consultas em grandes conjuntos de dados.

[ROW FORMAT row_format]

Especifica o formato de linha da tabela e os dados de origem subjacente, se aplicável. Para `row_format`, você pode especificar um ou mais delimitadores com a cláusula `DELIMITED` ou,

como alternativa, usar a cláusula `SERDE` conforme descrito abaixo. Se `ROW FORMAT` for omitido ou `ROW FORMAT DELIMITED` for especificado, um SerDe nativo será usado.

- [DELIMITED FIELDS TERMINATED BY char [ESCAPED BY char]]
- [DELIMITED COLLECTION ITEMS TERMINATED BY char]
- [MAP KEYS TERMINATED BY char]
- [LINES TERMINATED BY char]
- [NULL DEFINED AS char]

  Disponível somente com o Hive 0.13 e quando o formato de arquivo em STORED AS (ARMAZENADO COMO) for `TEXTFILE`.

--OU--

- SERDE 'serde_name' [WITH SERDEPROPERTIES ("property_name" = "property_value", "property_name" = "property_value" [, ...])]

  O `serde_name` indica o SerDe a ser usado. A cláusula `WITH SERDEPROPERTIES` permite fornecer uma ou mais propriedades personalizadas permitidas pelo SerDe.

[STORED AS formato do arquivo]

Especifica o formato de arquivo para dados da tabela. Se omitido, `TEXTFILE` será o padrão. As opções de `file_format` são:

- SEQUENCEFILE
- TEXTFILE
- RCFILE
- ORC
- PARQUET
- AVRO
- ION
- INPUTFORMAT input_format_classname OUTPUTFORMAT output_format_classname

[LOCATION 's3://DOC-EXAMPLE-BUCKET/[pasta]/']

Especifica o local dos dados subjacentes no Amazon S3 dos quais a tabela é criada. O caminho do local deve ser um nome de bucket ou um nome de bucket e uma ou mais pastas. Se você estiver usando partições, especifique a raiz dos dados particionados. Para obter mais informações sobre a localização da tabela, consulte Local da tabela no Amazon S3. Para obter

informações sobre formatos de dados e permissões, consulte Requisitos para tabelas no Athena e dados no Amazon S3.

Use uma barra à direita para a pasta ou o bucket. Não use nomes de arquivo ou caracteres glob.

Use:

`s3://DOC-EXAMPLE-BUCKET/`

`s3://DOC-EXAMPLE-BUCKET/folder/`

`s3://DOC-EXAMPLE-BUCKET/folder/anotherfolder/`

Não use:

`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 [, ...])]

Especifica pares de chave/valor de metadados personalizados para a definição da tabela, além das propriedades da tabela predefinidas, como `"comment"`.

has_encrypted_data: o Athena tem uma propriedade integrada, `has_encrypted_data`. Defina essa propriedade como `true` para indicar que o conjunto de dados subjacente especificado por `LOCATION` está criptografado. Se omitido e se as configurações do grupo de trabalho não substituírem as configurações do lado do cliente, a pressuposição será `false`. Se omitido ou definido como `false` quando os dados subjacentes estiverem criptografados, a consulta resultará em um erro. Para ter mais informações, consulte Criptografia inativa.

classificação: as tabelas criadas para o Athena no console do CloudTrail adicionam `cloudtrail` como um valor para a propriedade `classification`. Para executar trabalhos ETL, o AWS Glue requer a criação de uma tabela com a propriedade `classification` para indicar o tipo de dados do AWS Glue como `csv`, `parquet`, `orc`, `avro` ou `json`. Por exemplo, `'classification'='csv'`. Os trabalhos ETL falharão se você não especificar essa propriedade. Você poderá especificá-la mais tarde usando o console do AWS Glue, a API ou a CLI. Para obter mais informações, consulte Usar trabalhos do AWS Glue para ETL com o Athena e Criar trabalhos no AWS Glue no Guia do desenvolvedor do AWS Glue.

compression_level: a propriedade `compression_level` especifica o nível de compactação a ser usado. Essa propriedade se aplica apenas à compressão ZSTD. Os valores possíveis são de 1 a 22. O valor padrão é 3. Para ter mais informações, consulte Usar níveis de compressão ZSTD no Athena.

Para obter mais informações sobre outras propriedades de tabelas, consulte ALTER TABLE SET TBLPROPERTIES.

Exemplos

A instrução de exemplo `CREATE TABLE` a seguir cria uma tabela com base em dados de planetas separados por tabulações armazenados no 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/'
```

Observe os seguintes pontos:

- A cláusula `ROW FORMAT DELIMITED` indica que os dados são delimitados por um caractere específico.
- A cláusula `FIELDS TERMINATED BY '\t'` especifica que os campos nos dados do TSV sejam separados pelo caractere de tabulação ('\t').
- A cláusula `STORED AS TEXTFILE` indica que os dados sejam armazenados como arquivos de texto simples no Amazon S3.

Para consultar os dados, você pode usar uma instrução simples `SELECT` como a seguinte:

```
SELECT * FROM planet_data
```

Para usar o exemplo para criar sua própria tabela TSV no Athena, substitua os nomes das tabelas e colunas pelos nomes e tipos de dados de sua própria tabela e colunas e atualize a cláusula `LOCATION` para apontar para o caminho do Amazon S3 em que seus arquivos TSV estão armazenados.

Para obter mais informações sobre como criar tabelas, consulte Criar tabelas no Athena.

CREATE TABLE AS

Cria uma tabela preenchida com os resultados de uma consulta SELECT. Para criar uma tabela vazia, use CREATE TABLE. `CREATE TABLE AS` combina uma instrução DDL `CREATE TABLE` com uma instrução DML `SELECT` e, portanto, tecnicamente contém DDL e DML. Observe que, embora `CREATE TABLE AS` esteja agrupado aqui com outras instruções DDL, as consultas CTAS no Athena são tratadas como DML para fins de cotas de serviço. Para obter informações sobre as cotas de serviço do Athena, consulte Service Quotas.

> **Note**
>
> Para instruções CTAS, a configuração esperada do proprietário do bucket não se aplica ao local da tabela de destino no Amazon S3. A configuração esperada do proprietário do bucket se aplica somente ao local de saída do Amazon S3 que você especificar para os resultados da consulta do Athena. Para ter mais informações, consulte Especificar um local para resultados de consultas usando o console do Athena.

Para obter outras informações sobre `CREATE TABLE AS` que não fazem parte do escopo deste tópico de referência, consulte Criar uma tabela a partir de resultados de consultas (CTAS).

Tópicos

- Resumo
- Propriedades da tabela CTAS
- Exemplos

Resumo

```
CREATE TABLE table_name
[ WITH ( property_name = expression [, ...] ) ]
```

```
AS query
[ WITH [ NO ] DATA ]
```

Em que:

COM (property_name = expression [, ...])

Uma lista de propriedades opcionais da tabela CTAS, algumas das quais são específicas do formato de armazenamento de dados. Consulte Propriedades da tabela CTAS.

consulta

A consulta SELECT que é usada para criar uma tabela.

> **Important**
>
> Se você planeja criar uma consulta com partições, especifique os nomes das colunas particionadas por último na lista de colunas na `SELECT` instrução.

[WITH [NO] DATA]

Se `WITH NO DATA` for usado, uma tabela vazia com o mesmo esquema da tabela original será criada.

> **Note**
>
> Para incluir cabeçalhos de coluna na saída do resultado da consulta, você pode usar uma consulta `SELECT` simples em vez de uma consulta CTAS. Você pode recuperar os resultados no local dos resultados da consulta ou baixá-los diretamente usando o console do Athena. Para ter mais informações, consulte Trabalhar com resultados de consultas, consultas recentes e arquivos de saída.

Propriedades da tabela CTAS

Cada tabela CTAS no Athena tem uma lista de propriedades opcionais que você especifica usando `WITH (property_name = expression [, ...] )`. Para obter mais informações sobre como usar esses parâmetros, consulte Exemplos de consultas CTAS.

WITH (property_name = expression [, ...],)

table_type = ['HIVE', 'ICEBERG']

Opcional. O padrão é `HIVE`. Especifica o tipo de tabela da tabela resultante.

Exemplo:

```
WITH (table_type ='ICEBERG')
```

external_location = [location]

> **Note**
>
> Como as tabelas do Iceberg não são externas, essa propriedade não se aplicará a elas. Para definir o local raiz de uma tabela do Iceberg em uma instrução CTAS, use a propriedade `location`, que será descrita posteriormente nesta seção.

Opcional. O local no qual o Athena salvará sua consulta CTAS no Amazon S3.

Exemplo:

```
WITH (external_location ='s3://DOC-EXAMPLE-BUCKET/tables/parquet_table/')
```

O Athena não usa o mesmo caminho duas vezes para os resultados das consultas. Se você especificar o local manualmente, verifique se o local especificado no Amazon S3 não contém dados. O Athena nunca tenta excluir os dados. Para usar o mesmo local novamente, exclua os dados manualmente. Caso contrário, a consulta CTAS falhará.

Se você executar uma consulta CTAS que especifica um `external_location` em um grupo de trabalho que impõe um local para os resultados de consultas, a consulta falhará com uma mensagem de erro. Para ver o local dos resultados de consultas especificado para o grupo de trabalho, consulte os detalhes do grupo de trabalho.

Se o grupo de trabalho substituir a configuração de local dos resultados das consultas do lado do cliente, o Athena criará sua tabela no seguinte local:

```
s3://DOC-EXAMPLE-BUCKET/tables/query-id/
```

Se você não usar a propriedade `external_location` para especificar um local, e o grupo de trabalho não substituir as configurações do lado do cliente, o Athena usará a configuração do lado do cliente de local dos resultados das consultas para criar sua tabela no seguinte local:

```
s3://DOC-EXAMPLE-BUCKET/Unsaved-or-query-name/year/month/date/tables/query-id/
```

`is_external = [boolean]`

Opcional. Indica se a tabela corresponde a uma tabela externa. O padrão é true. Para tabelas do Iceberg, deve ser definido como “false” (falso).

Exemplo:

```
WITH (is_external = false)
```

`location = [location]`

Obrigatório para tabelas do Iceberg. Especifica o local raiz da tabela do Iceberg que será criada a partir dos resultados da consulta.

Exemplo:

```
WITH (location ='s3://DOC-EXAMPLE-BUCKET/tables/iceberg_table/')
```

`field_delimiter = [delimiter]`

Opcionais e específicos para formatos de armazenamento físico de dados com base em texto. O delimitador de campo de caractere único para arquivos em CSV, TSV e de texto. Por exemplo, `WITH (field_delimiter = ',')`. Atualmente, os delimitadores de campo de vários caracteres não são permitidos em consultas CTAS. Se você não especificar um delimitador do campo, `\001` será usado por padrão.

`format = [storage_format]`

O formato de armazenamento dos resultados de consultas CTAS, como `ORC`, `PARQUET`, `AVRO`, `JSON`, `ION` ou `TEXTFILE`. Para tabelas do Iceberg, os formatos permitidos são `ORC`, `PARQUET` e `AVRO`. Se for omitido, `PARQUET` é usado por padrão. O nome deste parâmetro, `format`, deve estar listado em minúsculas, ou sua consulta CTAS falhará.

Exemplo:

```
WITH (format = 'PARQUET')
```

bucketed_by = ARRAY[column_name[,…], bucket_count = [int]]

> **Note**
>
> Essa propriedade não se aplica para tabelas do Iceberg. Para tabelas do Iceberg, use o particionamento com transformação de bucket.

Uma lista matriz de buckets para dados do bucket. Se omitida, o Athena não armazenará os dados dessa consulta em bucket.

bucket_count = [int]

> **Note**
>
> Essa propriedade não se aplica para tabelas do Iceberg. Para tabelas do Iceberg, use o particionamento com transformação de bucket.

O número de buckets para armazenar seus dados em um bucket. Se omitido, o Athena não armazenará os dados em bucket. Exemplo:

```
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**
>
> Essa propriedade não se aplica para tabelas do Iceberg. Para usar transformações de partição para tabelas do Iceberg, use a propriedade `partitioning`, que será descrita posteriormente nesta seção.

Opcional. Uma lista matriz de colunas pela qual a tabela CTAS será particionada. Verifique se os nomes das colunas particionadas estão listados por último na lista de colunas da instrução `SELECT`.

partitioning = ARRAY[partition_transform, ...]

Opcional. Especifica o particionamento da tabela do Iceberg que será criada. O Iceberg é compatível com uma ampla variedade de transformações e evoluções de partições. As transformações de partição estão resumidas na tabela a seguir.

Transformação	Descrição
`year(ts)`	Cria uma partição para cada ano. O valor da partição corresponde a diferença em números inteiros, em anos, entre `ts` e 1.º de janeiro de 1970.
`month(ts)`	Cria uma partição para cada mês de cada ano. O valor da partição corresponde a diferença em números inteiros, em meses, entre `ts` e 1.º de janeiro de 1970.
`day(ts)`	Cria uma partição para cada dia de cada ano. O valor da partição corresponde a diferença em números inteiros, em dias, entre `ts` e 1.º de janeiro de 1970.
`hour(ts)`	Cria uma partição para cada hora de cada dia. O valor da partição corresponde a um carimbo de data/hora com os minutos e segundos definidos como zero.

Transformação	Descrição
`bucket(x, nbuckets)`	Realiza o hash dos dados em um número especificado de buckets. O valor da partição corresponde a um hash em números inteiros de `x`, com um valor entre 0 e `nbuckets - 1` incluso.
`truncate(s, nchars)`	Transforma o valor da partição nos primeiros caracteres `nchars` de `s`.

Exemplo:

```
WITH (partitioning = ARRAY['month(order_date)',
                           'bucket(account_number, 10)',
                           'country']))
```

`optimize_rewrite_min_data_file_size_bytes = [long]`

Opcional. Configuração específica de otimização de dados. Arquivos menores que o valor especificado são incluídos para otimização. O padrão é 0,75 vezes o valor de `write_target_data_file_size_bytes`. Essa propriedade se aplica apenas a tabelas do Iceberg. Para ter mais informações, consulte Otimizar tabelas Iceberg.

Exemplo:

```
WITH (optimize_rewrite_min_data_file_size_bytes = 402653184)
```

`optimize_rewrite_max_data_file_size_bytes = [long]`

Opcional. Configuração específica de otimização de dados. Arquivos maiores que o valor especificado são incluídos para otimização. O padrão é 1,8 vezes o valor de `write_target_data_file_size_bytes`. Essa propriedade se aplica apenas a tabelas do Iceberg. Para ter mais informações, consulte Otimizar tabelas Iceberg.

Exemplo:

```
WITH (optimize_rewrite_max_data_file_size_bytes = 966367641)
```

`optimize_rewrite_data_file_threshold = [int]`

Opcional. Configuração específica de otimização de dados. Se houver menos arquivos de dados que exigem otimização do que o limite fornecido, os arquivos não serão regravados. Isso permite acumular mais arquivos de dados para produzir arquivos mais próximos do tamanho de destino e ignorar a computação desnecessária para gerar economia de custos. O padrão é 5. Essa propriedade se aplica apenas a tabelas do Iceberg. Para ter mais informações, consulte Otimizar tabelas Iceberg.

Exemplo:

```
WITH (optimize_rewrite_data_file_threshold = 5)
```

`optimize_rewrite_delete_file_threshold = [int]`

Opcional. Configuração específica de otimização de dados. Se houver menos arquivos de exclusão associados a um arquivo de dados do que o limite, o arquivo de dados não será regravado. Isso permite acumular mais arquivos de exclusão para cada arquivo de dados a fim de gerar economia de custos. O padrão é 2. Essa propriedade se aplica apenas a tabelas do Iceberg. Para ter mais informações, consulte Otimizar tabelas Iceberg.

Exemplo:

```
WITH (optimize_rewrite_delete_file_threshold = 2)
```

`vacuum_min_snapshots_to_keep = [int]`

Opcional. Configuração específica para vácuo. O número mínimo de snapshots mais recentes a serem retidos. O padrão é um. Essa propriedade se aplica apenas a tabelas do Iceberg. Para ter mais informações, consulte VACUUM.

> **Note**
>
> A propriedade `vacuum_min_snapshots_to_keep` requer a versão 3 do mecanismo do Athena.

Exemplo:

```
WITH (vacuum_min_snapshots_to_keep = 1)
```

`vacuum_max_snapshot_age_seconds = [long]`

Opcional. Configuração específica para vácuo. Um período, em segundos, que representa o tempo pelo qual os snapshots serão retidos. O padrão é 432 mil (5 dias). Essa propriedade se aplica apenas a tabelas do Iceberg. Para ter mais informações, consulte VACUUM.

> **Note**
>
> A propriedade `vacuum_max_snapshot_age_seconds` requer a versão 3 do mecanismo do Athena.

Exemplo:

```
WITH (vacuum_max_snapshot_age_seconds = 432000)
```

`write_compression = [compression_format]`

O tipo de compactação a ser usado para qualquer formato de armazenamento que permita que a compactação seja especificada. O valor `compression_format` especifica a compactação a ser usada quando os dados são gravados na tabela. Você pode especificar a compactação para os formatos de arquivo `TEXTFILE`, `JSON`, `PARQUET` e `ORC`.

Por exemplo, se a propriedade `format` especificar `PARQUET` como o formato de armazenamento, o valor para `write_compression` especificará o formato de compactação para Parquet. Nesse caso, especificar um valor para `write_compression` é equivalente a especificar um valor para `parquet_compression`.

Por exemplo, se a propriedade `format` especificar `ORC` como o formato de armazenamento, o valor para `write_compression` especificará o formato de compactação para ORC. Nesse caso, especificar um valor para `write_compression` é equivalente a especificar um valor para `orc_compression`.

Não é possível especificar várias propriedades da tabela de formato de compactação na mesma consulta CTAS. Por exemplo, não é possível especificar `write_compression` e `parquet_compression` na mesma consulta. O mesmo se aplica a `write_compression` e

`orc_compression`. Para obter mais informações sobre os tipos de compactação suportados para cada formato de arquivo, consulte Suporte a compactação no Athena.

`orc_compression = [compression_format]`

O tipo de compactação a ser usado para o formato de arquivo ORC quando dados ORC são gravados na tabela. Por exemplo, `WITH (orc_compression = 'ZLIB')`. As partes dentro do arquivo ORC (exceto o ORC Postscript) são compactadas usando a compactação que você especificar. Se não especificada, a compactação ZLIB será usada por padrão para ORC.

> **Note**
>
> Para consistência, recomendamos que você use a propriedade `write_compression` em vez de `orc_compression`. Use a propriedade `format` para especificar o formato de armazenamento como ORC e, em seguida, use a propriedade `write_compression` para especificar o formato de compactação que ORC usará.

`parquet_compression = [compression_format]`

O tipo de compactação a ser usado para o formato de arquivo Parquet quando os dados do Parquet são gravados na tabela. Por exemplo, `WITH (parquet_compression = 'SNAPPY')`. Essa compactação é aplicada a blocos de colunas em arquivos Parquet. Se não especificada, a compactação GZIP será usada por padrão para Parquet.

> **Note**
>
> Para consistência, recomendamos que você use a propriedade `write_compression` em vez de `parquet_compression`. Use a propriedade `format` para especificar o formato de armazenamento como PARQUET e, em seguida, use a propriedade `write_compression` para especificar o formato de compactação que PARQUET usará.

`compression_level = [compression_level]`

O nível de compressão a ser usado. Essa propriedade se aplica apenas à compressão ZSTD. Os valores possíveis são de 1 a 22. O valor padrão é 3. Para ter mais informações, consulte Usar níveis de compressão ZSTD no Athena.

Exemplos

Para obter exemplos de consultas CTAS, consulte os seguintes recursos.

- Exemplos de consultas CTAS
- Usar CTAS e INSERT INTO para ETL e análise de dados
- Use CTAS statements with Amazon Athena to reduce cost and improve performance (Usar instruções CTAS com o Amazon Athena para reduzir custos e melhorar a performance)
- Usar CTAS e INSERT INTO para resolver o limite de 100 partições

CREATE VIEW

Cria uma nova exibição a partir de uma consulta `SELECT` especificada. A exibição é uma tabela lógica que pode ser referenciada por futuras consultas. As visualizações não contêm todos os dados e não gravam dados. Em vez disso, a consulta especificada pela exibição é executada sempre que você fizer referência à exibição por outra consulta.

> **Note**
>
> Este tópico fornece informações resumidas para referência. Para obter informações mais detalhadas sobre como usar as visualizações no Athena, consulte Trabalhar com visualizações. Para obter informações sobre as limitações de exibição, consulte Limitações das visualizações.

Resumo

```
CREATE [ OR REPLACE ] VIEW view_name AS query
```

A cláusula `OR REPLACE` opcional permite atualizar a exibição existente substituindo-a Para ter mais informações, consulte Criar visualizações.

Exemplos

Para criar uma exibição `test` a partir da tabela `orders`, use uma consulta semelhante à seguinte:

```
CREATE VIEW test AS
```

```
SELECT
orderkey,
orderstatus,
totalprice / 2 AS half
FROM orders;
```

Para criar uma exibição `orders_by_date` a partir da tabela `orders`, use a seguinte consulta:

```
CREATE VIEW orders_by_date AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate;
```

Para atualizar uma exibição existente, use um exemplo semelhante ao seguinte:

```
CREATE OR REPLACE VIEW test AS
SELECT orderkey, orderstatus, totalprice / 4 AS quarter
FROM orders;
```

Consulte também SHOW COLUMNS, SHOW CREATE VIEW, DESCRIBE VIEW e DROP VIEW.

DESCRIBE

Mostra uma ou mais colunas, inclusive de partição, da tabela especificada. Esse comando é útil para examinar os atributos de colunas complexas.

Resumo

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

> **Important**
>
> A sintaxe para essa declaração é `DESCRIBE` *`table_name`*, não `DESCRIBE TABLE` *`table_name`*. O uso da última sintaxe resulta na mensagem de erro FAILED: SemanticException [Error 10001]: Table not found table (FALHA: SemanticException [Erro 10001]: Tabela não encontrada).

Parâmetros

[EXTENDED | FORMATTED]

Determina o formato da saída. A omissão desses parâmetros mostra nomes de colunas e os tipos de dados correspondentes, incluindo colunas de partição, em formato tabular. Especificando `FORMATTED` não só mostra nomes de colunas e tipos de dados em formato tabular, mas também traz informações detalhadas de tabela e armazenamento. `EXTENDED` mostra informações de coluna e tipos de dados em formato tabular, além de metadados detalhados para a tabela no formato serializado Thrift. Esse formato é menos legível e ajuda principalmente na depuração.

[PARTITION partition_spec]

Se incluído, lista os metadados para a partição especificada por `partition_spec`, onde `partition_spec` está no formato `(partition_column = partition_col_value, partition_column = partition_col_value, ...)`.

[col_name ([.field_name] | [.'$elem$'] | [.'key'] | [.'$value$'])*]

Especifica a coluna e os atributos a serem examinados. Você pode especificar `.field_name` para um elemento de uma struct, `'$elem$'` para um elemento de matriz, `'$key$'` para uma chave de mapa e `'$value$'` para um valor de mapa. Você pode especificar isso de maneira recursiva para explorar mais a coluna complexa.

Exemplos

```
DESCRIBE orders
```

```
DESCRIBE FORMATTED mydatabase.mytable PARTITION (part_col = 100) columnA;
```

A consulta e a saída a seguir mostram informações de coluna e tipos de dados de uma tabela de `impressions` baseada em dados de amostra do 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
```

Os exemplos de consulta e saída a seguir mostram o resultado da mesma tabela quando a opção `FORMATTED` é usada.

```
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
```

Os exemplos de consulta e saída a seguir mostram o resultado da mesma tabela quando a opção EXTENDED é usada. As informações detalhadas da tabela são geradas em uma única linha, mas foram formatadas aqui para facilitar a leitura.

```
DESCRIBE EXTENDED impressions
```

```
requestbegintime          string                                              from
 deserializer
adid                      string                                              from
 deserializer
impressionid              string                                              from
 deserializer
referrer                  string                                              from
 deserializer
useragent                 string                                              from
 deserializer
usercookie                string                                              from
 deserializer
ip                        string                                              from
 deserializer
number                    string                                              from
 deserializer
processid                 string                                              from
 deserializer
```

```
browsercokie                string                                              from
 deserializer
requestendtime              string                                              from
 deserializer
timers                      struct<modellookup:string,requesttime:string>  from
 deserializer
threadid                    string                                              from
 deserializer
hostname                    string                                              from
 deserializer
sessionid                   string                                              from
 deserializer
dt                          string

# Partition Information
# col_name                  data_type                 comment

dt                          string

Detailed Table Information       Table(tableName:impressions, dbName:sampledb,
 owner:hadoop, createTime:1587610521,
lastAccessTime:0, retention:0, sd:StorageDescriptor(cols:
[FieldSchema(name:requestbegintime, type:string, comment:null),
FieldSchema(name:adid, type:string, comment:null), FieldSchema(name:impressionid,
 type:string, comment:null),
FieldSchema(name:referrer, type:string, comment:null), FieldSchema(name:useragent,
 type:string, comment:null),
FieldSchema(name:usercookie, type:string, comment:null), FieldSchema(name:ip,
 type:string, comment:null),
FieldSchema(name:number, type:string, comment:null), FieldSchema(name:processid,
 type:string, comment:null),
FieldSchema(name:browsercokie, type:string, comment:null),
 FieldSchema(name:requestendtime, type:string, comment:null),
FieldSchema(name:timers, type:struct<modellookup:string,requesttime:string>,
 comment:null), FieldSchema(name:threadid,
type:string, comment:null), FieldSchema(name:hostname, type:string, comment:null),
 FieldSchema(name:sessionid,
type:string, comment:null)], location:s3://us-east-1.elasticmapreduce/samples/hive-ads/
tables/impressions,
inputFormat:org.apache.hadoop.mapred.TextInputFormat,
outputFormat:org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat, compressed:false,
 numBuckets:-1,
serdeInfo:SerDeInfo(name:null, serializationLib:org.openx.data.jsonserde.JsonSerDe,
 parameters:{serialization.format=1,
```

```
paths=requestbegintime, adid, impressionid, referrer, useragent, usercookie, ip}),
 bucketCols:[], sortCols:[], parameters:{},
skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[],
 skewedColValueLocationMaps:{}),
storedAsSubDirectories:false), partitionKeys:[FieldSchema(name:dt, type:string,
 comment:null)],
parameters:{EXTERNAL=TRUE, transient_lastDdlTime=1587610521}, viewOriginalText:null,
 viewExpandedText:null,
tableType:EXTERNAL_TABLE)
```

DESCRIBE VIEW

Mostra a lista de colunas para a exibição nomeada. Isso permite examinar os atributos de uma exibição complexa.

Resumo

```
DESCRIBE [db_name.]view_name
```

Exemplo

```
DESCRIBE orders;
```

Consulte também SHOW COLUMNS, SHOW CREATE VIEW, SHOW VIEWS e DROP VIEW.

DROP DATABASE

Remove o banco de dados nomeado do catálogo. Se o banco de dados incluir tabelas, você deverá descartá-las antes de executar `DROP DATABASE` ou usar a cláusula `CASCADE`. O uso de `DATABASE` e `SCHEMA` é intercambiável. Eles significam a mesma coisa.

Resumo

```
DROP {DATABASE | SCHEMA} [IF EXISTS] database_name [RESTRICT | CASCADE]
```

Parâmetros

[SE EXISTIR]

Fará o erro ser suprimido se `database_name` não existir.

[RESTRICT|CASCADE]

Determina como tabelas dentro de `database_name` são consideradas durante a operação `DROP`. Se você especificar `RESTRICT`, o banco de dados não será ignorado se ele contiver tabelas. Esse é o comportamento padrão. Especificar `CASCADE` faz o banco de dados e todas as tabelas serem ignorados.

Exemplos

```
DROP DATABASE clickstreams;
```

```
DROP SCHEMA IF EXISTS clickstreams CASCADE;
```

> **Note**
>
> Quando você tenta eliminar um banco de dados cujo nome tem caracteres especiais (p. ex., `my-database`), você pode receber uma mensagem de erro. Para resolver esse problema, tente delimitar o nome do banco de dados com caracteres de crase (`). Para obter mais informações sobre nomenclatura de bancos de dados no Athena, consulte Nomes de tabelas, bancos de dados e colunas.

DROP TABLE

Remove a definição da tabela de metadados da tabela nomeada `table_name`. Quando você descarta uma tabela externa, os dados subjacentes permanecem intactos.

Resumo

```
DROP TABLE [IF EXISTS] table_name
```

Parâmetros

[IF EXISTS]

Fará o erro ser suprimido se `table_name` não existir.

Exemplos

```
DROP TABLE fulfilled_orders
```

```
DROP TABLE IF EXISTS fulfilled_orders
```

Ao usar o editor de consultas do console do Athena para descartar uma tabela que tenha caracteres especiais diferentes de sublinhados (_), use acentos graves, como no exemplo a seguir.

```
DROP TABLE `my-athena-database-01.my-athena-table`
```

Ao usar o conector JDBC para soltar uma tabela que tenha caracteres especiais, os caracteres de acento grave não são necessários.

```
DROP TABLE my-athena-database-01.my-athena-table
```

DROP VIEW

Exclui uma exibição existente. A cláusula `IF EXISTS` opcional faz com que o erro seja suprimido se a exibição não existir.

Para ter mais informações, consulte Trabalhar com visualizações.

Resumo

```
DROP VIEW [ IF EXISTS ] view_name
```

Exemplos

```
DROP VIEW orders_by_date
```

```
DROP VIEW IF EXISTS orders_by_date
```

Consulte também CREATE VIEW, SHOW COLUMNS, SHOW CREATE VIEW, SHOW VIEWS e DESCRIBE VIEW.

MSCK REPAIR TABLE

Use o comando `MSCK REPAIR TABLE` para atualizar os metadados no catálogo depois de adicionar partições compatíveis com o Hive.

O comando `MSCK REPAIR TABLE` verifica um sistema de arquivos, como o Amazon S3, para procurar se há partições compatíveis com o Hive que foram adicionadas ao sistema de arquivos após a criação da tabela. `MSCK REPAIR TABLE` compara as partições nos metadados da tabela e as partições no S3. Se houver novas partições no local do S3 que você especificou quando criou a tabela, ele as adicionará aos metadados e à tabela do Athena.

Quando você adiciona partições físicas, os metadados no catálogo ficam inconsistentes com o layout dos dados no sistema de arquivos, e é necessário adicionar informações sobre as novas partições ao catálogo. Para atualizar os metadados, execute `MSCK REPAIR TABLE` para que você possa consultar os dados nas novas partições do Athena.

> **Note**
>
> `MSCK REPAIR TABLE` somente adiciona partições aos metadados, não as remove. Para remover partições dos metadados depois que elas foram excluídas manualmente do Amazon S3, execute o comando `ALTER TABLE` *`table-name`* `DROP PARTITION`. Para obter mais informações, consulte ALTER TABLE DROP PARTITION.

Considerações e limitações

Ao usar `MSCK REPAIR TABLE`, lembre-se dos seguintes pontos:

- É possível que demore algum tempo para adicionar todas as partições. Se expirar, essa operação estará em um estado incompleto, quando somente algumas partições são adicionadas ao catálogo. Você deve executar `MSCK REPAIR TABLE` na mesma tabela até que todas as partições sejam adicionadas. Para ter mais informações, consulte Particionar dados no Athena.
- Para as partições que não são compatíveis com o Hive, use ALTER TABLE ADD PARTITION para carregá-las para poder consultar os dados.
- Os locais das partições que serão usados com o Athena devem aplicar o protocolo do `s3` (por exemplo, `s3://DOC-EXAMPLE-BUCKET/`*`folder`*`/`). No Athena, os locais que usam outros protocolos (por exemplo, `s3a://`*`bucket`*`/`*`folder`*`/`) resultam em falhas nas consultas `MSCK REPAIR TABLE` quando elas são executadas nas tabelas que os contêm.

- Como `MSCK REPAIR TABLE` verifica uma pasta e as subpastas para encontrar um esquema de partição correspondente, mantenha os dados das tabelas separadas em hierarquias de pastas separadas. Por exemplo, suponha que você tenha dados na tabela 1 em `s3://DOC-EXAMPLE-BUCKET1` e dados na tabela 2 em `s3://DOC-EXAMPLE-BUCKET1/table-2-data`. Se ambas as tabelas forem particionadas por string, `MSCK REPAIR TABLE` adicionará as partições da tabela 2 à tabela 1. Para evitar isso, use estruturas de pastas separadas, como `s3://DOC-EXAMPLE-BUCKET1` e `s3://DOC-EXAMPLE-BUCKET2`. Observe que esse comportamento é consistente com o Amazon EMR e o Apache Hive.
- Devido a um problema conhecido, a `MSCK REPAIR TABLE` falha silenciosamente quando os valores da partição contêm dois pontos (`:`), por exemplo, quando o valor da partição é um carimbo de data/hora. Como solução alternativa, use ALTER TABLE ADD PARTITION.
- `MSCK REPAIR TABLE` X não adiciona nomes de colunas de partição que começam com um sublinhado (_). Para contornar essa limitação, utilize ALTER TABLE ADD PARTITION.

Resumo

```
MSCK REPAIR TABLE table_name
```

Exemplos

```
MSCK REPAIR TABLE orders;
```

Solução de problemas

Depois que você executar `MSCK REPAIR TABLE`, se o Athena não adicionar as partições à tabela no AWS Glue Data Catalog, verifique o seguinte:

- Acesso do AWS Glue: certifique-se de que o perfil do AWS Identity and Access Management (IAM) tenha uma política que permita a ação `glue:BatchCreatePartition`. Para obter mais informações, consulte Permitir glue:BatchCreatePartition na política do IAM adiante neste documento.
- Acesso do Amazon S3: certifique-se de que o perfil tenha uma política com permissões suficientes para acessar o Amazon S3, incluindo a ação `s3:DescribeJob`. Para ver um exemplo das ações do Amazon S3 que devem ser permitidas, consulte o exemplo de política de bucket em Acesso entre contas no Athena aos buckets do Amazon S3.
- Uso de maiúsculas e minúsculas em chaves de objeto do Amazon S3: verifique se o caminho do Amazon S3 está em letras minúsculas em vez de minúsculas concatenadas (por exemplo,

`userid` em vez de `userId`) ou use `ALTER TABLE ADD PARTITION` para especificar os nomes de chaves de objeto. Para obter mais informações, consulte Alterar ou redefinir o caminho do Amazon S3 adiante neste documento.

- Tempo limite de consulta esgotado: é melhor usar `MSCK REPAIR TABLE` para criar uma tabela pela primeira vez ou quando há incerteza sobre a paridade entre os dados e os metadados da partição. Se você usa `MSCK REPAIR TABLE` para adicionar novas partições com frequência (por exemplo, diariamente) e sempre enfrenta problemas de tempo limite de consulta esgotado, considere usar ALTER TABLE ADD PARTITION.
- Partições ausentes do sistema de arquivos: se você excluir manualmente uma partição do Amazon S3 e executar `MSCK REPAIR TABLE`, poderá receber a mensagem de erro: Partições ausentes do sistema de arquivos. Isso ocorre porque `MSCK REPAIR TABLE` não remove partições obsoletas dos metadados da tabela. Em vez disso, execute ALTER TABLE DROP PARTITION para remover as partições excluídas dos metadados da tabela. Do mesmo modo, veja que SHOW PARTITIONS lista apenas as partições nos metadados, e não as partições no sistema de arquivos.
- Erro “NullPointerException name is null” (O nome de NullPointerException é nulo)

  Se você usar a operação de API do AWS Glue CreateTable ou o modelo `AWS::Glue::Table` do AWS CloudFormation para criar uma tabela para uso no Athena sem especificar a propriedade `TableType` e, depois, executar uma consulta DDL, como `SHOW CREATE TABLE` ou `MSCK REPAIR TABLE`, poderá receber a mensagem de erro FALHA: o nome de NullPointerException é nulo.

  Para resolver o erro, especifique um valor para o atributo TableInput `TableType` como parte da chamada de API `CreateTable` do AWS Glue ou do modelo do AWS CloudFormation. Os valores possíveis para `TableType` são `EXTERNAL_TABLE` ou `VIRTUAL_VIEW`.

  Esse requisito é aplicado somente quando você cria uma tabela usando a operação de API do AWS Glue `CreateTable` ou o modelo do `AWS::Glue::Table`. Se você criar uma tabela do Athena usando uma instrução DDL ou um crawler do AWS Glue, a propriedade `TableType` será definida automaticamente para você.

As seções a seguir apresentam mais detalhes.

Permitir glue:BatchCreatePartition na política do IAM

Analise as políticas do IAM vinculadas ao perfil que você usa para executar `MSCK REPAIR TABLE`. Quando você usa o AWS Glue Data Catalog com o Athena, a política do IAM deve

permitir a ação `glue:BatchCreatePartition`. Para ver um exemplo de uma política do IAM que permite a ação `glue:BatchCreatePartition`, consulte Política gerenciada pela AWS: AmazonAthenaFullAccess.

Alterar ou redefinir o caminho do Amazon S3

Se uma ou mais chaves de objeto no caminho do Amazon S3 estiverem em letras minúsculas concatenadas em vez de minúsculas, talvez `MSCK REPAIR TABLE` não adicione as partições ao AWS Glue Data Catalog. Por exemplo, se o caminho do Amazon S3 incluir o nome da chave do objeto `userId`, talvez as seguintes partições não sejam adicionadas ao 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/
```

Para resolver esse problema, execute um dos seguintes procedimentos:

- Use letras minúsculas em vez de minúsculas concatenadas ao criar chaves de objeto do Amazon S3:

  ```
  s3://DOC-EXAMPLE-BUCKET/path/userid=1/

  s3://DOC-EXAMPLE-BUCKET/path/userid=2/

  s3://DOC-EXAMPLE-BUCKET/path/userid=3/
  ```

- Use ALTER TABLE ADD PARTITION para redefinir o local, como no seguinte exemplo:

  ```
  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/'
  ```

Embora os nomes de chave de objeto do Amazon S3 possam usar letras maiúsculas, os nomes de bucket do Amazon S3 devem estar sempre em letras minúsculas. Para obter mais informações,

consulte Diretrizes de nomeação de chave de objeto e Regras de nomeação de bucket no Guia do usuário do Amazon S3.

SHOW COLUMNS

Exibe somente os nomes de coluna de uma única tabela ou visualização especificada. Para obter informações mais detalhadas, consulte o AWS Glue Data Catalog. Para obter informações e exemplos, consulte as seguintes seções do tópico Consulta no AWS Glue Data Catalog:

- Para visualizar os metadados de coluna, como tipo de dados, consulte Listar ou pesquisar colunas de uma tabela ou visualização especificada.
- Para visualizar todas as colunas de todas as tabelas em um banco de dados específico no `AwsDataCatalog`, consulte Listar ou pesquisar colunas de uma tabela ou visualização especificada.
- Para visualizar todas as colunas de todas as tabelas em todos os bancos de dados no `AwsDataCatalog`, consulte Listagem de todas as colunas de todas as tabelas.
- Para visualizar as colunas que tabelas específicas têm em comum em um banco de dados, consulte Listagem das colunas que tabelas específicas têm em comum.

Resumo

```
SHOW COLUMNS {FROM|IN} database_name.table_name
```

```
SHOW COLUMNS {FROM|IN} table_name [{FROM|IN} database_name]
```

É possível usar as palavras-chave `FROM` e `IN` de forma intercambiável. Se *`table_name`* ou *`database_name`* tiver caracteres especiais como hifens, coloque-os entre acentos graves (por exemplo, `` `my-database`.`my-table` ``). Não coloque *`table_name`* ou *`database_name`* entre aspas simples ou duplas. Atualmente, o uso de `LIKE` e das expressões de correspondência de padrões não é permitido.

Exemplos

Os exemplos equivalentes a seguir mostram as colunas da tabela `orders` no banco de dados `customers`. Os dois primeiros exemplos consideram o banco de dados `customers` como o atual.

```
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

Analisa uma tabela existente chamada `table_name` para gerar a consulta que a criou.

Resumo

```
SHOW CREATE TABLE [db_name.]table_name
```

Parâmetros

TABLE [db_name.]table_name

O parâmetro db_name é opcional. Se omitido, o contexto assumirá como padrão o banco de dados atual.

> **Note**
>
> O nome da tabela é obrigatório.

Exemplos

```
SHOW CREATE TABLE orderclickstoday;
```

```
SHOW CREATE TABLE `salesdata.orderclickstoday`;
```

Solução de problemas

Se você usar a operação de API do AWS Glue CreateTable ou o modelo `AWS::Glue::Table` do AWS CloudFormation para criar uma tabela para uso no Athena sem especificar a propriedade `TableType` e, depois, executar uma consulta DDL, como `SHOW CREATE TABLE` ou `MSCK REPAIR TABLE`, poderá receber a mensagem de erro FALHA: o nome de NullPointerException é nulo.

Para resolver o erro, especifique um valor para o atributo TableInput `TableType` como parte da chamada de API `CreateTable` do AWS Glue ou do modelo do AWS CloudFormation. Os valores possíveis para `TableType` são `EXTERNAL_TABLE` ou `VIRTUAL_VIEW`.

Esse requisito é aplicado somente quando você cria uma tabela usando a operação de API do AWS Glue `CreateTable` ou o modelo do `AWS::Glue::Table`. Se você criar uma tabela do Athena usando uma instrução DDL ou um crawler do AWS Glue, a propriedade `TableType` será definida automaticamente para você.

SHOW CREATE VIEW

Mostra a instrução SQL que cria a exibição específica.

Resumo

```
SHOW CREATE VIEW view_name
```

Exemplos

```
SHOW CREATE VIEW orders_by_date
```

Consulte também CREATE VIEW e DROP VIEW.

SHOW DATABASES

Lista todos os bancos de dados definidos na metastore. Você pode usar `DATABASES` ou `SCHEMAS`. Eles significam a mesma coisa.

O equivalente programático de `SHOW DATABASES` é a ação da API ListDatabases do Athena. O método equivalente em AWS SDK for Python (Boto3) é list_databases.

Resumo

```
SHOW {DATABASES | SCHEMAS} [LIKE 'regular_expression']
```

Parâmetros

[LIKE '*regular_expression*']

Filtra a lista de bancos de dados aos correspondentes à *regular_expression* especificada por você. Para correspondência de caracteres curinga, você pode usar a combinação `.*`, que associa qualquer caractere zero a multiplicações ilimitadas.

Exemplos

```
SHOW SCHEMAS;
```

```
SHOW DATABASES LIKE '.*analytics';
```

SHOW PARTITIONS

Lista todas as partições em uma tabela do Athena em uma ordem não classificada.

Resumo

```
SHOW PARTITIONS table_name
```

- Para exibir as partições em uma tabela e listá-las em uma ordem específica, consulte a seção Listar partições de uma tabela específica na página Consulta no AWS Glue Data Catalog.
- Para visualizar o conteúdo de uma partição, consulte a seção Consultar os dados na página Particionar dados no Athena.
- `SHOW PARTITIONS` não lista as partições que foram projetadas pelo Athena, mas que não estão registradas no catálogo do AWS Glue. Para obter informações sobre a projeção de partições, consulte Projeção de partições com o Amazon Athena.
- `SHOW PARTITIONS` lista as partições nos metadados, não as partições no sistema de arquivos real. Para atualizar os metadados depois que você excluir manualmente as partições do Amazon S3, execute ALTER TABLE DROP PARTITION.

Exemplos

A consulta de exemplo a seguir mostra as partições da tabela `flight_delays_csv`, que inclui os dados da tabela de voos do Departamento de Transporte dos EUA. Para obter mais informações

sobre a tabelas `flight_delays_csv` de exemplo, consulte LazySimpleSerDe para arquivos CSV, TSV e com delimitação personalizada. A tabela está particionada por ano.

```
SHOW PARTITIONS flight_delays_csv
```

Resultados

```
year=2007
year=2015
year=1999
year=1993
year=1991
year=2003
year=1996
year=2014
year=2004
year=2011
...
```

A consulta de exemplo a seguir mostra as partições da tabela `impressions`, que inclui amostra de dados de navegação na Web. Para obter mais informações sobre a tabelas `impressions` de exemplo, consulte Particionar dados no Athena. A tabela está particionada pela coluna `dt` (data e hora).

```
SHOW PARTITIONS impressions
```

Resultados

```
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
...
```

Listar partições em ordem de classificação

Para ordenar as partições na lista de resultados, use a sintaxe `SELECT` a seguir, em vez de `SHOW PARTITIONS`.

```
SELECT * FROM database_name."table_name$partitions" ORDER BY column_name
```

A consulta a seguir mostra a lista das partições do exemplo de `flight_delays_csv`, mas agora classificada.

```
SELECT * FROM "flight_delays_csv$partitions" ORDER BY year
```

Resultados

```
year
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
...
```

Para obter mais informações, consulte a seção Listar partições de uma tabela específica na página Consulta no AWS Glue Data Catalog.

SHOW TABLES

Lista todas as tabelas base e exibe em um banco de dados.

Resumo

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

Parâmetros

[IN database_name]

Especifica o `database_name` de quais tabelas serão listadas. Se omitido, o banco de dados do contexto atual é assumido.

> **Note**
>
> `SHOW TABLES` poderá falhar se `database_name` usar um caractere sem suporte, como o hífen. Como solução alternativa, tente delimitar o nome do banco de dados com acentos graves.

['regular_expression']

Filtra a lista de tabelas às correspondentes ao `regular_expression` especificado por você. Para indicar qualquer caractere nas tabelas `AWSDataCatalog`, você pode usar a expressão curinga `*` ou `.*`. Para bancos de dados do Apache Hive, use a expressão curinga `.*`. Para indicar uma opção entre caracteres, use o caractere `|`.

Exemplos

Example – mostrar todas as tabelas no banco de dados **`sampledb`**

```
SHOW TABLES IN sampledb
```

`Results`

```
alb_logs
cloudfront_logs
elb_logs
flights_2016
flights_parquet
view_2016_flights_dfw
```

Example – mostrar os nomes de todas as tabelas em **`sampledb`** que incluem a palavra “flights”

```
SHOW TABLES IN sampledb '*flights*'
```

Results

```
flights_2016
flights_parquet
view_2016_flights_dfw
```

Example – mostrar os nomes de todas as tabelas em **sampledb** que terminam com a palavra “logs”

```
SHOW TABLES IN sampledb '*logs'
```

Results

```
alb_logs
cloudfront_logs
elb_logs
```

SHOW TBLPROPERTIES

Lista as propriedades da tabela nomeada.

Resumo

```
SHOW TBLPROPERTIES table_name [('property_name')]
```

Parâmetros

[('property_name')]

Se incluído, somente o valor da propriedade chamada `property_name` será listado.

Exemplos

```
SHOW TBLPROPERTIES orders;
```

```
SHOW TBLPROPERTIES orders('comment');
```

SHOW VIEWS

Lista as exibições no banco de dados especificado ou no banco de dados atual se você omitir o nome do banco de dados. Use a cláusula `LIKE` opcional com uma expressão regular para restringir a lista de nomes de exibições.

O Athena retorna uma lista de valores do tipo `STRING`, em que cada valor é um nome de visualização.

Resumo

```
SHOW VIEWS [IN database_name] [LIKE 'regular_expression']
```

Parâmetros

[IN database_name]

Especifica o `database_name` do qual as exibições serão listadas. Se omitido, o banco de dados do contexto atual é assumido.

[LIKE 'regular_expression']

Filtra a lista de exibições às correspondentes ao `regular_expression` especificado por você. Somente o caractere curinga `*`, o que indica qualquer caractere, ou `|`, o que indica uma escolha entre caracteres, podem ser usados.

Exemplos

```
SHOW VIEWS;
```

```
SHOW VIEWS IN marketing_analytics LIKE 'orders*'
```

Consulte também SHOW COLUMNS, SHOW CREATE VIEW, DESCRIBE VIEW e DROP VIEW.

Considerações e limitações das consultas SQL no Amazon Athena

Ao executar consultas no Athena, tenha em mente as considerações e limitações a seguir.

- Procedimentos armazenados: não há suporte para procedimentos armazenados.

- Número máximo de partições: o número máximo de partições que você pode criar com instruções `CREATE TABLE AS SELECT` (CTAS) é 100. Para obter informações, consulte CREATE TABLE AS. Para obter uma solução alternativa, consulte Usar CTAS e INSERT INTO para resolver o limite de 100 partições.
- Instruções não permitidas: as seguintes instruções não são permitidas:
  - Não há suporte ao `CREATE TABLE LIKE`.
  - `DESCRIBE INPUT` e `DESCRIBE OUTPUT` não são compatíveis.
  - A instrução `MERGE` é compatível somente com formatos de tabela transacional. Para ter mais informações, consulte MERGE INTO.
  - As instruções `UPDATE` não são compatíveis.
- Conectores Trino e Presto : não há compatibilidade com conectores Trino nem Presto. Use a consulta federada do Amazon Athena para conectar origens de dados. Para ter mais informações, consulte Usar a consulta federada do Amazon Athena.
- Tempo limite esgotado em tabelas com muitas partições: o Athena pode esgotar o tempo limite ao consultar uma tabela com milhares de partições. Isso pode acontecer quando a tabela tem muitas partições que não são do tipo `string`. Quando você usa o tipo `string`, o Athena remove as partições no nível do metastore. No entanto, quando você usa outros tipos de dados, o Athena remove as partições do servidor. Quanto mais partições você tiver, mais tempo esse processo levará e maior será a probabilidade de suas consultas atingirem o tempo limite. Para resolver esse problema, defina o tipo de partição como `string` para que o Athena remova as partições no nível do metastore. Isso reduz a sobrecarga e evita que as consultas atinjam o tempo limite.
- Compatibilidade com o S3 Glacier: para obter informações sobre como consultar objetos restaurados do Amazon S3 Glacier, consulte Consultar objetos restaurados do Amazon S3 Glacier.
- Arquivos tratados como ocultos: o Athena trata os arquivos de origem que começam com sublinhado (_) ou ponto (`.`) como ocultos. Para contornar essa limitação, renomeie os arquivos.
- Limitação de tamanho de linha ou coluna: o tamanho de uma única linha ou de suas colunas não pode exceder 32 megabytes. Esse limite pode ser excedido quando, por exemplo, uma linha em um arquivo CSV ou JSON contém uma única coluna de 300 megabytes. Exceder esse limite também pode gerar a mensagem de erro: Line too long in text file (Linha muito longa no arquivo de texto). Para resolver essa limitação, certifique-se de que a soma dos dados das colunas em qualquer linha seja inferior a 32 MB.
- Cláusula LIMIT máxima: o número máximo de linhas que podem ser especificadas para a cláusula `LIMIT` é

9223372036854775807. Ao usar `ORDER BY`, o número máximo de linhas permitido para a cláusula LIMIT é 2.147.483.647. Exceder esse limite resultará na mensagem de erro NOT_SUPPORTED: ORDER BY LIMIT > 2147483647 is not supported [Não há suporte para ORDER BY LIMIT > 2147483647].

- information_schema: as consultas de `information_schema` apresentam melhor desempenho se você tiver uma quantidade pequena a moderada de metadados do AWS Glue. Pode haver erros se você tiver uma grande quantidade de metadados. Para obter informações sobre como consultar o banco de dados `information_schema` para metadados do AWS Glue, consulte Consulta no AWS Glue Data Catalog.
- Inicializações de matriz: devido a uma limitação do Java, não é possível inicializar no Athena uma matriz que tenha mais de 254 argumentos.
- Colunas ocultas de metadados: as colunas de metadados ocultas do Hive ou do Iceberg `$bucket`, `$file_modified_time`, `$file_size` e `$partition` não são compatíveis para visualizações. Para obter informações sobre como usar a coluna `$path` de metadados no Athena, consulte Obter os locais de arquivos dos dados de origem no Amazon S3 .

Para obter informações sobre o tamanho máximo da cadeia de caracteres de consulta, cotas para tempos limite de consulta e cotas para o número ativo de consultas DML, acesse Service Quotas.

Solução de problemas no Athena

A equipe do Athena reuniu as seguintes informações de solução de problemas dos clientes. Elas não abrangem tudo, mas incluem orientações sobre alguns problemas comuns de performance, tempo limite e falta de memória.

Tópicos

- CREATE TABLE AS SELECT (CTAS)
- Problemas no arquivo de dados
- Tabelas do Linux Foundation Delta Lake
- Consultas federadas
- Erros relacionados ao JSON
- MSCK REPAIR TABLE
- Problemas de saída

- Problemas do Parquet
- Problemas de particionamento
- Permissões
- Problemas de sintaxe da consulta
- Problemas de tempo limite de consulta
- Problemas de controle de utilização
- Visões
- Grupos de trabalho
- Recursos adicionais do
- Catálogo de erros do Athena

CREATE TABLE AS SELECT (CTAS)

Dados duplicados ocorrem com instruções CTAS simultâneas

O Athena não mantém a validação simultânea de CTAS. Verifique se não há instruções CTAS duplicadas para o mesmo local ao mesmo tempo. Mesmo se uma instrução CTAS ou INSERT INTO falhar, os dados órfãos podem permanecer no local de dados especificado na instrução.

HIVE_TOO_MANY_OPEN_PARTITIONS

Ao usar uma instrução CTAS para criar uma tabela com mais de 100 partições, você pode receber o erro HIVE_TOO_MANY_OPEN_PARTITIONS: Exceeded limit of 100 open writers for partitions/buckets (Limite excedido de 100 gravadores abertos para partições/buckets). Para contornar essa limitação, é possível usar uma instrução CTAS e uma série de instruções `INSERT INTO` que criam ou inserem até 100 partições cada. Para ter mais informações, consulte Usar CTAS e INSERT INTO para resolver o limite de 100 partições.

Problemas no arquivo de dados

O Athena não pode ler arquivos ocultos

O Athena trata os arquivos de origem que começam com sublinhado (_) ou ponto (.) como ocultos. Para contornar essa limitação, renomeie os arquivos.

O Athena lê arquivos que eu excluí do crawler do AWS Glue

O Athena não reconhece os padrões de exclusão que você especifica para um crawler do AWS Glue. Por exemplo, se você tem um bucket do Amazon S3 com os arquivos `.csv` e `.json` e exclui os arquivos `.json` do crawler, o Athena consulta os dois grupos de arquivos. Para evitar isso, coloque os arquivos que você deseja excluir em um local diferente.

HIVE_BAD_DATA: erro ao analisar o valor do campo

Esse erro pode ocorrer nos seguintes cenários:

- O tipo de dados definido na tabela não corresponde aos dados de origem ou um único campo contém tipos de dados diferentes. Para ver as resoluções sugeridas, consulte My Amazon Athena query fails with the error “HIVE_BAD_DATA: Error parsing field value for field x: For input string: ‘12312845691’” (Minha consulta do Amazon Athena falha com o erro “HIVE_BAD_DATA: erro ao analisar o valor do campo x: para a string de entrada: ‘12312845691’”) na Central de Conhecimento da AWS.
- Há valores nulos em um campo de número inteiro. Uma solução alternativa é criar a coluna com os valores nulos como `string` e usar `CAST` para converter o campo em uma consulta especificando um valor padrão de `0` para nulos. Para obter mais informações, consulte When I query CSV data in Athena, I get the error “HIVE_BAD_DATA: Error parsing field value “ for field x: For input string: ””” (Quando consulto dados CSV no Athena, aparece o erro “HIVE_BAD_DATA: erro ao analisar o valor do campo x: para a string de entrada: ‘’") na Central de Conhecimento da AWS.

HIVE_CANNOT_OPEN_SPLIT: erro ao abrir divisão do Hive s3://DOC-EXAMPLE-BUCKET

Esse erro pode ocorrer quando você consulta um prefixo de bucket do Amazon S3 que tenha um grande número de objetos. Para obter mais informações, consulte How do I resolve the "HIVE_CANNOT_OPEN_SPLIT: Error opening Hive split s3://DOC-EXAMPLE-BUCKET/: Slow down" error in Athena? no Centro de Conhecimentos da AWS.

HIVE_CURSOR_ERROR: com.amazonaws.services.s3.model.AmazonS3Exception: a chave especificada não existe

Geralmente, esse erro ocorre quando um arquivo é removido durante a execução de uma consulta. Execute novamente a consulta ou examine o fluxo de trabalho para ver se outro trabalho ou processo está modificando os arquivos durante a execução da consulta.

HIVE_CURSOR_ERROR: fim inesperado do stream de entrada

Essa mensagem indica que o arquivo está corrompido ou vazio. Verifique a integridade do arquivo e execute a consulta novamente.

HIVE_FILESYSTEM_ERROR: Incorrect fileSize ***1234567*** for file (HIVE_FILESYSTEM_ERROR: tamanho de arquivo 1234567 incorreto para o arquivo)

Essa mensagem pode ocorrer quando um arquivo foi alterado entre o planejamento da consulta e a execução da consulta. Geralmente ocorre quando um arquivo no Amazon S3 é substituído no local (por exemplo, um PUT é executado em uma chave em que um objeto já existe). O Athena não suporta a exclusão ou a substituição do conteúdo de um arquivo quando uma consulta está em execução. Para evitar esse erro, agende trabalhos que substituam ou excluam arquivos para quando consultas não são executadas ou simplesmente grave os dados em novos arquivos ou partições.

HIVE_UNKNOWN_ERROR: não é possível criar o formato de entrada

Esse erro pode ser resultado de problemas como estes:

- O crawler do AWS Glue não conseguiu classificar o formato de dados
- Certas propriedades de definição de tabela do AWS Glue estão vazias
- O Athena não é compatível com o formato de dados dos arquivos no Amazon S3

Para obter mais informações, consulte Como resolvo o erro “não é possível criar o formato de entrada” no Athena? (em inglês) ou assista ao vídeo na Central de Conhecimento da AWS.

O local do S3 especificado para salvar os resultados das consultas é inválido.

Verifique se você especificou um local válido do S3 para os resultados das consultas. Para obter mais informações, consulte Especificar um local para resultados de consultasTrabalhar com resultados de consultas, consultas recentes e arquivos de saída no tópico.

Tabelas do Linux Foundation Delta Lake

O esquema das tabelas do Delta Lake não está sincronizado

Quando você consulta uma tabela do Delta Lake que tem um esquema no AWS Glue, pode receber a seguinte mensagem de erro:

```
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.
```

O esquema pode ficar desatualizado se for modificado no AWS Glue depois de ser adicionado ao Athena. Para atualizar o esquema, faça uma das etapas a seguir:

- No AWS Glue, execute o crawler do AWS Glue.
- No Athena, descarte a tabela e torne a criá-la.
- Adicione manualmente as colunas que estiverem faltando, usando a instrução ALTER TABLE ADD COLUMNS no Athena ou editando o esquema da tabela no AWS Glue.

Consultas federadas

Tempo limite ao chamar ListTableMetadata

Uma chamada para a API ListTableMetadata pode atingir o tempo limite se houver muitas tabelas na fonte de dados, se a fonte de dados estiver lenta ou se a rede estiver lenta. Para solucionar esse problema, experimente as seguintes etapas:

- Verifique o número de tabelas: se você tiver mais de 1.000 tabelas, tente reduzir esse número. Para uma resposta mais rápida de `ListTableMetadata`, recomendamos ter menos de 1000 tabelas por catálogo.
- Verifique a configuração do Lambda — Monitorar o comportamento da função do Lambda é fundamental. Ao usar catálogos federados, examine os logs de execução da função do Lambda. Com base nos resultados, ajuste os valores de memória e tempo limite. Para identificar possíveis problemas de tempo limite, revise sua configuração do Lambda. Para obter mais informações, consulte Configurar o tempo de limite da função (console) no Guia do desenvolvedor do AWS Lambda.
- Verifique os logs da fonte de dados federada — Examine os logs e as mensagens de erro da fonte de dados federada para ver se há algum problema ou erro. Os logs podem fornecer informações valiosas sobre a causa do tempo limite.
- Use **`StartQueryExecution`** para buscar os metadados: se você tiver mais de 1.000 tabelas, recuperar os metadados usando o conector federado pode levar mais tempo que o esperado. Como a natureza assíncrona de StartQueryExecution garante que o Athena execute a consulta da

maneira ideal, considere usar `StartQueryExecution` como alternativa a `ListTableMetadata`. Os exemplos da AWS CLI a seguir mostram como `StartQueryExecution` pode ser usado em vez de `ListTableMetadata` para obter todos os metadados das tabelas do catálogo de dados.

Primeiro, execute uma consulta que obtenha todas as tabelas, como no exemplo a seguir.

```
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"
```

Em seguida, recupere os metadados de uma tabela individual, como no exemplo a seguir.

```
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"
```

O tempo necessário para obter os resultados depende do número de tabelas do catálogo.

Para obter mais informações sobre como solucionar problemas de consultas federadas, acesse Common_Problems na seção awslabs/aws-athena-query-federation do GitHub ou consulte a documentação de cada conector de fonte de dados do Athena.

Erros relacionados ao JSON

Erros de dados NULL ou incorretos ao tentar ler dados JSON

Os erros de dados NULL ou incorretos quando você tenta ler dados JSON podem ter diversas causas. Para identificar as linhas que estão causando erros quando você usa o OpenX SerDe, defina `ignore.malformed.json` como `true`. Registros malformados serão retornados como NULL. Para obter mais informações, consulte Recebo erros ao tentar ler dados JSON no Amazon Athena (em inglês) ou assista ao vídeo na Central de Conhecimento da AWS.

HIVE_BAD_DATA: erro ao analisar o valor do campo 0: java.lang.String não pode ser convertido em org.openx.data.jsonserde.json.JSONObject

O OpenX JSON SerDe gera esse erro quando não consegue analisar uma coluna em uma consulta do Athena. Isso poderá acontecer se você definir uma coluna como `map` ou `struct`, mas os dados subjacentes são, na verdade, `string`, `int` ou outro tipo primitivo.

HIVE_CURSOR_ERROR: Row is not a valid JSON object - JSONException: Duplicate key (HIVE_CURSOR_ERROR: a linha não é um objeto JSON válido - JSONException: chave duplicada)

Esse erro ocorre ao usar o Athena para consultar recursos de AWS Config que têm várias etiquetas com o mesmo nome com letras maiúsculas e minúsculas diferentes. A solução é executar `CREATE TABLE` usando `WITH SERDEPROPERTIES 'case.insensitive'='false'` e mapear os nomes. Para obter mais informações sobre `case.insensitive` e mapeamento, consulte Bibliotecas SerDe JSON. Para obter mais informações, consulte Como resolvo o erro “HIVE_CURSOR_ERROR: a linha não é um objeto JSON válido - JSONException: chave duplicada” ao ler arquivos do AWS Config no Athena? no Centro de Conhecimento da AWS.

Mensagens HIVE_CURSOR_ERROR com JSON formatado para impressão

As bibliotecas Hive JSON SerDe e OpenX JSON SerDe esperam que cada documento JSON esteja em uma única linha de texto, sem caracteres de terminação de linha separando os campos no registro. Se o texto JSON estiver formatado para impressão, você poderá receber uma mensagem de erro como HIVE_CURSOR_ERROR: Row is not a valid JSON Object (HIVE_CURSOR_ERROR: a linha não é um objeto JSON válido) ou HIVE_CURSOR_ERROR: JsonParseException: Unexpected end-of-input: expected close marker for OBJECT (HIVE_CURSOR_ERROR: JSONParseException: Fim de entrada inesperado: marcador de fechamento esperado para OBJECT) quando tentar consultar a tabela após criá-la. Para obter mais informações, consulte JSON Data Files (Arquivos de dados do JSON) na documentação do OpenX SerDe no GitHub.

Vários registros JSON retornam SELECT COUNT de 1

Se você usa OpenX JSON SerDe, verifique se os registros estão separados por um caractere de nova linha. Para obter mais informações, consulte A consulta SELECT COUNT no Amazon Athena retorna somente um registro, embora o arquivo JSON de entrada tenha vários registros (em inglês) na Central de Conhecimento da AWS.

Não é possível consultar uma tabela criada por um crawler do AWS Glue que usa um classificador JSON personalizado

O mecanismo do Athena não é compatível com classificadores JSON personalizados. Para resolver esse problema, crie uma nova tabela sem o classificador personalizado. Para transformar o JSON, você pode usar CTAS ou criar uma visualização. Por exemplo, se você trabalha com arrays, pode usar a opção UNNEST para nivelar o JSON. Outra opção é usar um trabalho ETL AWS Glue que aceita o classificador personalizado, converter os dados em parquet no Amazon S3 e consultá-los no Athena.

MSCK REPAIR TABLE

Para obter informações sobre problemas relacionados a MSCK REPAIR TABLE, consulte as seções Considerações e limitações e Solução de problemas da página MSCK REPAIR TABLE.

Problemas de saída

Não é possível verificar/criar bucket de saída

Esse erro poderá ocorrer se o local de resultados das consultas especificado não existir ou se as permissões apropriadas não estiverem presentes. Para obter mais informações, consulte How do I resolve the “unable to verify/create output bucket” error in Amazon Athena? (Como resolvo o erro “Não é possível verificar/criar bucket de saída” no Amazon Athena?) na Central de Conhecimento da AWS.

O resultado de TIMESTAMP é vazio

O Athena requer o formato de TIMESTAMP do Java. Para obter mais informações, consulte Quando consulto uma tabela no Amazon Athena, o resultado de TIMESTAMP é vazio (em inglês) na Central de Conhecimento da AWS.

Armazenar saída de consulta do Athena em um formato diferente de CSV

Por padrão, o Athena só gera arquivos de saída no formato CSV. Para gerar os resultados de uma consulta `SELECT` em outro formato, você pode usar a instrução `UNLOAD`. Para ter mais informações, consulte UNLOAD. Você também pode usar uma consulta do CTAS que usa a propriedade de tabela `format` para configurar o formato de saída. Ao contrário de `UNLOAD`, a técnica CTAS requer a criação de uma tabela. Para obter mais informações, consulte Como posso armazenar uma saída de

consulta do Athena em um formato diferente de CSV, como um formato compactado? (em inglês) na Central de Conhecimento da AWS.

O local do S3 especificado para salvar os resultados das consultas é inválido

Você poderá receber essa mensagem de erro se o local do bucket de saída não estiver na mesma região onde você executa sua consulta. Para evitar isso, especifique um local de resultados de consulta na região onde você executa a consulta. Para obter as etapas, consulte Especificar um local para resultados de consultas.

Problemas do Parquet

org.apache.parquet.io.GroupColumnIO não pode ser convertido em org.apache.parquet.io.PrimitiveColumnIO

Esse erro é causado por uma incompatibilidade de esquema do parquet. Uma coluna com um tipo não primitivo (por exemplo, `array`) foi declarado como um tipo primitivo (por exemplo, `string`) no AWS Glue. Para solucionar esse problema, verifique o esquema de dados nos arquivos e compare-o com o esquema declarado no AWS Glue.

Problemas estatísticos do Parquet

Quando você lê dados Parquet, pode receber mensagens de erro como as seguintes:

```
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.
```

Para contornar esse problema, use a instrução CREATE TABLE ou ALTER TABLE SET TBLPROPERTIES para definir a `parquet.ignore.statistics` propriedade Parquet SerDe como `true`, como nos exemplos a seguir.

Exemplo de CREATE TABLE

```
...
ROW FORMAT SERDE
'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
WITH SERDEPROPERTIES ('parquet.ignore.statistics'='true')
```

```
STORED AS PARQUET
...
```

Exemplo de ALTER TABLE

```
ALTER TABLE ... SET TBLPROPERTIES ('parquet.ignore.statistics'='true')
```

Para obter mais informações sobre o Parquet Hive SerDe, consulte Parquet SerDe.

Problemas de particionamento

MSCK REPAIR TABLE não remove partições obsoletas

Se você excluir uma partição manualmente no Amazon S3 e executar MSCK REPAIR TABLE, poderá receber a mensagem de erro Partições ausentes do sistema de arquivos. Isso ocorre porque MSCK REPAIR TABLE não remove as partições obsoletas dos metadados da tabela. Use ALTER TABLE DROP PARTITION para remover as partições obsoletas manualmente. Para obter mais informações, consulte a seção “Solução de problemas” do tópico MSCK REPAIR TABLE.

Falha em MSCK REPAIR TABLE

Quando uma grande quantidade de partições (por exemplo, mais de 100.000) são associadas a uma tabela específica, pode haver falha em `MSCK REPAIR TABLE` devido às limitações de memória. Para contornar esse limite, use ALTER TABLE ADD PARTITION no lugar dele.

MSCK REPAIR TABLE detecta partições, mas não as adiciona ao AWS Glue

Esse problema poderá ocorrer se um caminho do Amazon S3 estiver em maiúsculas e minúsculas, em vez de apenas minúsculas, ou se uma política do IAM não permitir a ação `glue:BatchCreatePartition`. Para obter mais informações, consulte MSCK REPAIR TABLE detecta partições no Athena, mas não as adiciona ao AWS Glue Data Catalog (em inglês) na Central de Conhecimento da AWS.

Os intervalos de projeção de partições com o formato de data dd-MM-yyyy-HH-mm-ss ou yyyy-MM-dd não funcionam

Para funcionar corretamente, o formato de data deve ser definido como `yyyy-MM-dd HH:00:00`. Para obter mais informações, consulte a publicação do Stack Overflow Athena Partition Projection Not Working As Expected (A projeção de partições do Athena não está funcionado como esperado).

PARTITITION BY não é compatível com o tipo BIGINT

Converta o tipo de dados em `string` e tente novamente.

Não há partições significativas disponíveis

Geralmente, essa mensagem de erro significa que as configurações de partição foram corrompidas. Para resolver esse problema, descarte a tabela e crie outra com novas partições.

A projeção de partições não funciona em conjunto com as partições do intervalo

Verifique se a unidade do intervalo de tempo projection.*<columnName>*.interval.unit corresponde ao delimitador das partições. Por exemplo, se as partições forem delimitadas por dias, uma unidade de intervalo de horas não funcionará.

Erro de projeção de partição quando o intervalo é especificado por hífen

Especificar a propriedade da tabela `range` com um hífen em vez de uma vírgula produz um erro como INVALID_TABLE_PROPERTY: For input string: "*number*-*number*". Certifique-se de que os valores do intervalo estejam separados por uma vírgula, não por um hífen. Para ter mais informações, consulte Tipo integer.

HIVE_UNKNOWN_ERROR: não é possível criar o formato de entrada

Uma ou mais das partições do Glue são declaradas em um formato diferente porque cada uma tem um formato próprio de entrada específico que é independente. Verifique como suas partições estão definidas no AWS Glue.

HIVE_PARTITION_SCHEMA_MISMATCH

Se o esquema de uma partição for diferente do esquema da tabela, uma consulta poderá falhar com a mensagem de erro HIVE_PARTITION_SCHEMA_MISMATCH. Para ter mais informações, consulte Sincronizar o esquema da partição para evitar “HIVE_PARTITION_SCHEMA_MISMATCH”.

A tabela SemanticException não foi particionada, mas a especificação da partição existe

Esse erro pode ocorrer quando não há partições definidas na instrução `CREATE TABLE`. Para obter mais informações, consulte Como solucionar o erro "FALHA: a tabela SemanticException não foi particionada, mas a especificação da partição existe" no Athena? (em inglês) na Central de Conhecimento da AWS.

Arquivos de zero byte no formato `_$folder$`

Se você executar uma instrução `ALTER TABLE ADD PARTITION` e especificar erroneamente uma partição que já existe e uma localização incorreta do Amazon S3, serão criados arquivos de espaço reservado de zero byte do formato *`partition_value`*`_$folder$` no Amazon S3. Você precisa remover esses arquivos manualmente.

Para evitar que isso aconteça, use a sintaxe `ADD IF NOT EXISTS` em sua instrução `ALTER TABLE ADD PARTITION`, assim:

```
ALTER TABLE table_name ADD IF NOT EXISTS PARTITIION […]
```

Zero registro retornado dos dados particionados

Esse problema pode ocorrer por vários motivos. Para saber as causas possíveis e as resoluções, consulte Criei uma tabela no Amazon Athena com partições definidas, mas quando consulto a tabela, zero registro é retornado (em inglês) na Central de Conhecimento da AWS.

Consulte também HIVE_TOO_MANY_OPEN_PARTITIONS.

Permissões

Erro de acesso negado ao consultar o Amazon S3

Isso pode ocorrer quando você não tem permissão para ler os dados no bucket, permissão para gravar no bucket de resultados ou o caminho do Amazon S3 contém um endpoint de região como `us-east-1.amazonaws.com`. Para obter mais informações, consulte When I run an Athena query, I get an “access denied” error (Quando executo uma consulta do Athena, recebo um erro “Acesso negado”) na Central de Conhecimento da AWS.

Acesso negado com o erro de código de status 403 ao executar consultas DDL em dados criptografados no Amazon S3

Você poderá receber a mensagem de erro Acesso negado (serviço: Amazon S3; código de status: 403; código de erro: AccessDenied; ID da solicitação: *`<request_id>`*) se as seguintes condições forem verdadeiras:

1. Execute uma consulta DDL, como `ALTER TABLE ADD PARTITION` ou `MSCK REPAIR TABLE`.
2. Você tem um bucket com a criptografia padrão configurada para usar `SSE-S3`.

3. O bucket também tem uma política como a seguinte que força as solicitações `PutObject` a especificar os cabeçalhos PUT `"s3:x-amz-server-side-encryption": "true"` e `"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"
                }
            }
        }
    ]
}
```

Nesse caso, a solução recomendada é remover a política de bucket, como a que foi mostrada acima, quando a criptografia padrão do bucket já está presente.

Acesso negado com código de status 403 ao consultar um bucket do Amazon S3 em outra conta

Esse erro pode ocorrer quando você tenta consultar logs escritos por outro AWS service (Serviço da AWS) e a segunda conta é o proprietária do bucket, mas não tem os objetos no bucket. Para obter mais informações, consulte I get the Amazon S3 exception “access denied with status code: 403” in

Amazon Athena when I query a bucket in another account (Vejo a exceção do Amazon S3 “Acesso negado com código de status: 403” no Amazon Athena quando consulto um bucket em outra conta) ou assista ao vídeo na Central de Conhecimento da AWS.

Usar credenciais de função do IAM para se conectar ao driver JDBC do Athena

Você pode recuperar as credenciais temporárias de uma função para autenticar o conector JDBC ao Athena. As credenciais temporárias têm uma vida útil máxima de 12 horas. Para obter mais informações, consulte Como posso usar minhas credenciais de função do IAM ou alternar para outra função do IAM ao me conectar ao Athena usando o driver JDBC? (em inglês) na Central de Conhecimento da AWS.

Problemas de sintaxe da consulta

FAILED: NullPointerException name is null (FALHA: o nome de NullPointerException é nulo)

Se você usar a operação de API do AWS Glue CreateTable ou o modelo `AWS::Glue::Table` do AWS CloudFormation para criar uma tabela para uso no Athena sem especificar a propriedade `TableType` e, depois, executar uma consulta DDL, como `SHOW CREATE TABLE` ou `MSCK REPAIR TABLE`, poderá receber a mensagem de erro FALHA: o nome de NullPointerException é nulo.

Para resolver o erro, especifique um valor para o atributo TableInput `TableType` como parte da chamada de API `CreateTable` do AWS Glue ou do modelo do AWS CloudFormation. Os valores possíveis para `TableType` são `EXTERNAL_TABLE` ou `VIRTUAL_VIEW`.

Esse requisito é aplicado somente quando você cria uma tabela usando a operação de API do AWS Glue `CreateTable` ou o modelo do `AWS::Glue::Table`. Se você criar uma tabela do Athena usando uma instrução DDL ou um crawler do AWS Glue, a propriedade `TableType` será definida automaticamente para você.

Função não registrada

Esse erro ocorre quando você tenta usar uma função que o Athena não permite. Para ver a lista de funções permitidas pelo Athena, consulte Funções no Amazon Athena ou execute a instrução `SHOW FUNCTIONS` no editor de consultas. Você também pode escrever sua própria User Defined Function (UDF – Função definida pelo usuário). Para obter mais informações, consulte Como resolver o erro de sintaxe “função não registrada” no Athena? (em inglês) na Central de Conhecimento da AWS.

Exceções GENERIC_INTERNAL_ERROR

As exceções `GENERIC_INTERNAL_ERROR` podem ter várias causas, incluindo as seguintes:

- GENERIC_INTERNAL_ERROR: Null (GENERIC_INTERNAL_ERROR: nulo): você pode ver essa exceção em qualquer uma das seguintes condições:
  - Existe discrepância de esquema entre o tipo de dados de uma coluna na definição de tabela e o tipo de dados real do conjunto de dados.
  - Você está executando uma consulta (CTAS) `CREATE TABLE AS SELECT` com uma sintaxe inexata.
- GENERIC_INTERNAL_ERROR: parent builder is null (GENERIC_INTERNAL_ERROR: o criador pai é nulo): você pode ver essa exceção quando consulta uma tabela de colunas com tipo de dado `array` e está usando a biblioteca OpenCSVSerde. O formato OpenCSVSerde não suporta o tipo de dados `array`.
- GENERIC_INTERNAL_ERROR: Value exceeds MAX_INT (GENERIC_INTERNAL_ERROR: o valor excede MAX_INT): você pode ver essa exceção quando a coluna de dados de origem é definida com o tipo de dados `INT` e tem um valor numérico maior que 2.147.483.647.
- GENERIC_INTERNAL_ERROR: Value exceeds MAX_BYTE (GENERIC_INTERNAL_ERROR: o valor excede MAX_BYTE): você pode ver essa exceção quando a coluna de dados de origem tem um valor numérico que excede o tamanho permitido para o tipo de dadosBYTE. O tipo de dado `BYTE` é equivalente a `TINYINT`. `TINYINT` é um número inteiro com sinal de 8 bits no formato de complemento de dois com um valor mínimo de -128 e um valor máximo de 127.
- GENERIC_INTERNAL_ERROR: Number of partition values does not match number of filters (GENERIC_INTERNAL_ERROR: o número de partições não corresponde ao número de filtros): você pode ver essa exceção se tiver partições inconsistentes nos dados do Amazon Simple Storage Service (Amazon S3). Você pode ter partições inconsistentes em qualquer uma das seguintes condições:
  - As partições no Amazon S3 foram alteradas (exemplo: novas partições foram adicionadas).
  - O número de colunas de partição na tabela não corresponde às dos metadados da partição.

Para obter informações mais detalhadas sobre cada um desses erros, consulte How do I resolve the error “GENERIC_INTERNAL_ERROR" when I query a table in Amazon Athena? na Central de Conhecimento da AWS.

O número de grupos correspondentes não corresponde ao número de colunas

Esse erro ocorre ao usar Regex SerDe em uma instrução CREATE TABLE e o número de grupos correspondentes de regex não corresponde ao número de colunas que você especificou para a tabela. Para obter mais informações, consulte How do I resolve the RegexSerDe error “number of matching groups doesn't match the number of columns” in Amazon Athena? (Como resolvo o erro RegexSerDe “O número de grupos correspondentes não corresponde ao número de colunas” no Amazon Athena?) na Central de Conhecimento da AWS.

queryString não atende à restrição: o tamanho do membro deve ser menor ou igual a 262144

O tamanho máximo da string de consulta no Athena (262.144 bytes) não é uma cota ajustável. O AWS Support não pode aumentar a cota para você, mas você pode tentar resolver o problema dividindo as consultas longas em tamanhos menores. Para obter mais informações, consulte Como posso aumentar o tamanho máximo da string de consulta no Athena? (em inglês) na Central de Conhecimento da AWS.

SYNTAX_ERROR: a coluna não pode ser resolvida

Esse erro pode ocorrer quando você consulta uma tabela criada por um crawler do AWS Glue de um arquivo CSV codificado em UTF-8 que tem uma Byte Order Mark (BOM – Marca de ordem de byte). O AWS Glue não reconhece as BOMs e as altera para pontos de interrogação, que o Amazon Athena não reconhece. A solução é remover o ponto de interrogação no Athena ou no AWS Glue.

Excesso de argumentos para chamada de função

No mecanismo Athena versão 3, as funções não podem receber mais de 127 argumentos. Essa limitação é proposital. Se você usar uma função com mais de 127 parâmetros, ocorrerá uma mensagem de erro como esta:

TOO_MANY_ARGUMENTS: linha *nnn*:*nn*: excesso de argumentos para a chamada da função *function_name*().

Para resolver esse problema, utilize menos parâmetros por chamada de função.

Problemas de tempo limite de consulta

Se você tiver erros de tempo limite nas suas consultas do Athena, verifique os logs do CloudTrail. O tempo limite das consultas pode ser atingido devido ao controle de utilização das APIs do AWS Glue

ou do Lake Formation. Quando esses erros ocorrem, as mensagens de erro correspondentes podem indicar um problema de tempo limite de consulta em vez de um problema de controle de utilização. Para solucionar o problema, você pode verificar os logs do CloudTrail antes de entrar em contato com o AWS Support. Para obter mais informações, consulte Consultar os logs do AWS CloudTrail e Registro de chamadas de API do Amazon Athena com o AWS CloudTrail.

Vaja informações sobre problemas de tempo limite de consulta com as consultas federadas quando você chama a API `ListTableMetadata` em Tempo limite ao chamar ListTableMetadata.

Problemas de controle de utilização

Se suas consultas excederem os limites dos serviços dependentes, como o Amazon S3, AWS KMS, AWS Glue ou AWS Lambda, as mensagens a seguir podem ser esperadas. Para resolver esses problemas, reduza o número de chamadas simultâneas originadas da mesma conta.

Serviço	Mensagem de erro
AWS Glue	AWSGlueException: Rate exceeded. (AWSGlueException: Taxa excedida)
AWS KMS	Você excedeu a taxa na qual pode chamar o KMS. Reduza a frequência de suas chamadas.
AWS Lambda	Rate exceeded (Taxa excedida) TooManyRequestsException
Amazon S3	AmazonS3Exception: Please reduce your request rate. (AmazonS3Exception: Reduza sua taxa de solicitação.)

Para obter informações sobre formas de evitar o controle de utilização do Amazon S3 ao usar o Athena, consulte Como prevenir o controle de utilização do Amazon S3.

Visões

Visualizações criadas no shell do Apache Hive não funcionam no Athena

Devido a suas implementações fundamentalmente diferentes, as visualizações criadas no shell do Apache Hive não são compatíveis com o Athena. Para resolver esse problema, recrie as visualizações no Athena.

A visualização é obsoleta e deve ser recriada

Você pode receber esse erro se a tabela subjacente a uma visualização foi alterada ou descartada. A resolução é recriar a visualização. Para obter mais informações, consulte How can I resolve the "view is stale; it must be re-created" error in Athena? (Como posso resolver o erro "A visualização é obsoleta e deve ser recriada" no Athena?) na Central de Conhecimento da AWS.

Grupos de trabalho

Para obter informações sobre como solucionar problemas de grupos de trabalho, consulte Resolver problemas nos grupos de trabalho.

Recursos adicionais do

As páginas a seguir apresentam mais informações para solucionar problemas com o Amazon Athena.

- Catálogo de erros do Athena
- Service Quotas
- Considerações e limitações das consultas SQL no Amazon Athena
- DDL incompatível
- Nomes de tabelas, bancos de dados e colunas
- Tipos de dados no Amazon Athena
- SerDes e formatos de dados compatíveis
- Suporte a compactação no Athena
- Palavras-chave reservadas
- Resolver problemas nos grupos de trabalho

Os seguintes recursos da AWS também podem ajudar:

- Tópicos do Athena na Central de Conhecimento da AWS
- Perguntas do Amazon Athena em AWS re:Post
- Publicações do Athena no blog sobre big data da AWS

Geralmente, a solução de problemas requer consulta e descoberta iterativas por um especialista ou de uma comunidade de ajudantes. Se continuar a enfrentar problemas após tentar as sugestões

nesta página, entre em contato com o AWS Support (no AWS Management Console, clique em Support (Suporte), Support Center (Central de Suporte)) ou faça uma pergunta no AWS re:Post usando a etiqueta Amazon Athena.

Catálogo de erros do Athena

O Athena fornece informações de erro padronizadas para ajudar você a compreender consultas com falha e tomar medidas após uma falha de consulta. O recurso `AthenaError` inclui um campo `ErrorCategory` e um campo `ErrorType`. `ErrorCategory` especifica se a causa da consulta com falha é um erro do sistema, erro do usuário ou outro erro. `ErrorType` fornece informações mais detalhadas sobre a origem da falha. Ao combinar esses dois campos, você entende melhor as circunstâncias relacionadas e as causas do erro específico que ocorreu.

Categoria do erro

A seguinte tabela lista os valores da categoria de erro do Athena e seus significados.

Categoria do erro	Origem
1	SYSTEM
2	USER
3	OUTRO

Referência de tipos de erros

A seguinte tabela lista os valores do tipo de erro do Athena e seus significados.

Tipo de erro	Descrição
0	A consulta esgotou recursos esgotados nesse fator de escala
1	A consulta esgotou recursos esgotados nesse fator de escala
2	A consulta esgotou recursos esgotados nesse fator de escala
3	A consulta esgotou recursos esgotados nesse fator de escala

Tipo de erro	Descrição
4	A consulta esgotou recursos esgotados nesse fator de escala
5	A consulta esgotou recursos esgotados nesse fator de escala
6	A consulta esgotou recursos esgotados nesse fator de escala
7	A consulta esgotou recursos esgotados nesse fator de escala
8	A consulta esgotou recursos esgotados nesse fator de escala
100	Erro de serviço interno
200	O mecanismo de consulta apresentou um erro interno
201	O mecanismo de consulta apresentou um erro interno
202	O mecanismo de consulta apresentou um erro interno
203	Erro do driver
204	Erro no metastore
205	O mecanismo de consulta apresentou um erro interno
206	Tempo limite da consulta
207	O mecanismo de consulta apresentou um erro interno
208	O mecanismo de consulta apresentou um erro interno
209	Falha no cancelamento da consulta
210	Tempo limite da consulta
211	O mecanismo de consulta apresentou um erro interno
212	O mecanismo de consulta apresentou um erro interno
213	O mecanismo de consulta apresentou um erro interno

Tipo de erro	Descrição
214	O mecanismo de consulta apresentou um erro interno
215	O mecanismo de consulta apresentou um erro interno
216	O mecanismo de consulta apresentou um erro interno
217	O mecanismo de consulta apresentou um erro interno
218	O mecanismo de consulta apresentou um erro interno
219	O mecanismo de consulta apresentou um erro interno
220	O mecanismo de consulta apresentou um erro interno
221	O mecanismo de consulta apresentou um erro interno
222	O mecanismo de consulta apresentou um erro interno
223	O mecanismo de consulta apresentou um erro interno
224	O mecanismo de consulta apresentou um erro interno
225	O mecanismo de consulta apresentou um erro interno
226	O mecanismo de consulta apresentou um erro interno
227	O mecanismo de consulta apresentou um erro interno
228	O mecanismo de consulta apresentou um erro interno
229	O mecanismo de consulta apresentou um erro interno
230	O mecanismo de consulta apresentou um erro interno
231	O mecanismo de consulta apresentou um erro interno
232	O mecanismo de consulta apresentou um erro interno
233	Erro no Iceberg

Tipo de erro	Descrição
234	Erro no Lake Formation
235	O mecanismo de consulta apresentou um erro interno
236	O mecanismo de consulta apresentou um erro interno
237	Erro de serialização
238	Falha ao carregar metadados no Amazon S3
239	Erro geral de persistência
240	Falha no envio da consulta
300	Erro de serviço interno
301	Erro de serviço interno
302	Erro de serviço interno
303	Erro de serviço interno
400	Erro de serviço interno
401	Falha na gravação de resultados da consulta no Amazon S3
402	Falha na gravação de resultados da consulta no Amazon S3
1000	Erro do usuário
1001	Erro de dados
1.002	Erro de dados
1003	Falha na tarefa DDL
1004	Erro de esquema
1005	Erro de serialização

Tipo de erro	Descrição
1006	Erro de sintaxe
1007	Erro de dados
1008	Consulta rejeitada
1009	Falha na consulta
1010	Erro de serviço interno
1011	Cancelamento da consulta pelo usuário
1012	O mecanismo de consulta apresentou um erro interno
1013	O mecanismo de consulta apresentou um erro interno
1014	Cancelamento da consulta pelo usuário
1100	Argumento inválido fornecido
1101	Propriedade inválida fornecida
1102	O mecanismo de consulta apresentou um erro interno
1103	Propriedade de tabela inválida fornecida
1104	O mecanismo de consulta apresentou um erro interno
1105	O mecanismo de consulta apresentou um erro interno
1106	Argumento de função inválido fornecido
1107	Exibição inválida
1108	Falha no registro da função
1109	Caminho fornecido do Amazon S3 não encontrado
1110	Tabela ou exibição fornecida inexistente

Tipo de erro	Descrição
1200	Consulta sem suporte
1201	Decodificador fornecido sem suporte
1202	Tipo de consulta sem suporte
1300	Erro geral de não encontrado
1301	Entidade geral não encontrada
1302	Arquivo não encontrado
1303	Função ou implementação de função fornecida não encontrada
1304	O mecanismo de consulta apresentou um erro interno
1305	O mecanismo de consulta apresentou um erro interno
1306	Bucket do Amazon S3 não encontrado
1307	Mecanismo selecionado não encontrado
1308	O mecanismo de consulta apresentou um erro interno
1400	Erros de controle de utilização
1401	Falha na consulta devido a controle de utilização de AWS Glue
1402	Falha na consulta devido a muitas versões de tabela na AWS Glue
1403	A consulta falhou como resultado do controle de utilização do Amazon S3
1404	A consulta falhou como resultado do controle de utilização do Amazon Athena
1405	A consulta falhou como resultado do controle de utilização do Amazon Athena
1406	A consulta falhou como resultado do controle de utilização do Amazon Athena
1500	Erro de permissão

Tipo de erro	Descrição
1501	Erro de permissão do Amazon S3
1602	Excedeu o limite da capacidade reservada. Capacidade insuficiente para executar esta consulta.
1700	A consulta falhou devido a uma exceção interna do Lake Formation
1701	A consulta falhou devido a uma exceção interna do AWS Glue
9999	Erro de serviço interno

Exemplos de código

Os exemplos apresentados neste tópico usam o SDK para Java 2.x como ponto de partida para a criação de aplicações do Athena.

> **Note**
>
> Para obter informações sobre como programar o Athena usando outros AWS SDKs específicos de linguagem, consulte os seguintes recursos:
>
> - 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`)

Para obter mais informações sobre como executar os exemplos de código para Java nesta seção, consulte Amazon Athena Java readme no repositório de exemplos de código da AWS no GitHub.

Para obter a referência de programação em Java para o Athena, consulte AthenaClient no AWS SDK for Java 2.x.

- Exemplos de código Java
  - Constantes
  - Criar um cliente para acessar o Athena
  - Trabalhar com execuções de consulta
    - Iniciar a execução da consulta
    - Interromper a execução da consulta
    - Listar execuções de consulta
  - Trabalhar com consultas nomeadas
    - Criar uma consulta nomeada
    - Excluir uma consulta nomeada
    - Listar execuções de consulta

> **Note**
>
> Estes exemplos usam constantes (por exemplo, `ATHENA_SAMPLE_QUERY`) para strings, definidas em uma declaração da classe `ExampleConstants.java`. Substitua essas constantes pelas próprias strings ou constantes definidas.

Constantes

A classe `ExampleConstants.java` demonstra como consultar uma tabela criada pelo tutorial Conceitos básicos no 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

}
```

Criar um cliente para acessar o Athena

A classe `AthenaClientFactory.java` mostra como criar e configurar um cliente do 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();
    }
}
```

Iniciar a execução da consulta

O `StartQueryExample` mostra como enviar uma consulta ao Athena, aguardar até que os resultados estejam disponíveis e processá-los.

```
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());
            }
        }
    }
}
```

Interromper a execução da consulta

O `StopQueryExecutionExample` executa um exemplo de consulta, imediatamente interrompe a consulta e verifica o status dela para garantir que foi cancelada.

```
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;

    }
}
```

Listar execuções de consulta

O `ListQueryExecutionsExample` mostra como obter uma lista de IDs de execução de consulta.

```
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);
        }
    }
}
```

Criar uma consulta nomeada

O `CreateNamedQueryExample` mostra como criar uma consulta nomeada.

```
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);
        }
    }
}
```

Excluir uma consulta nomeada

O `DeleteNamedQueryExample` mostra como excluir uma consulta nomeada usando o ID da consulta nomeada.

```
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;
    }
}
```

Listar consultas nomeadas

O `ListNamedQueryExample` mostra como obter uma lista de IDs de consultas nomeadas.

```
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);
        }
    }
}
```

Uso do Apache Spark no Amazon Athena

O Amazon Athena facilita a execução interativa de análises e exploração de dados usando o Apache Spark, sem a necessidade de planejamento, configuração ou gerenciamento de recursos. Executar aplicações do Apache Spark no Athena significa enviar o código do Spark para processamento e receber os resultados diretamente sem a necessidade de uma configuração adicional. É possível usar a experiência simplificada de cadernos no console do Amazon Athena para desenvolver aplicações do Apache Spark usando APIs de cadernos do Python ou do Athena. O Apache Spark no Amazon Athena corresponde a uma tecnologia sem servidor e oferece uma escalabilidade automática sob demanda que fornece computação instantânea para atender aos volumes de dados em constante mudança e aos requisitos de processamento.

O Amazon Athena oferece os recursos a seguir:

- Uso do console: envie suas aplicações do Spark usando o console do Amazon Athena.
- Criação de scripts: crie e depure aplicações do Apache Spark de forma rápida e interativa em Python.
- Escalabilidade dinâmica: o Amazon Athena determina automaticamente a memória e os recursos de computação necessários para executar um trabalho, além de escalar continuamente esses recursos de acordo com os máximos especificados. Essa escalabilidade dinâmica reduz os custos sem afetar a velocidade.
- Experiência de cadernos: use o editor de cadernos do Athena para criar, editar e executar cálculos usando uma interface familiar. Os cadernos do Athena são compatíveis com os cadernos Jupyter e contêm uma lista de células que são executadas por ordem como cálculos. O conteúdo da célula pode incluir código, texto, Markdown, matemática, plotagens e mídia avançada.

Para obter informações adicionais, consulte Explore seu data lake usando o Amazon Athena para Apache Spark no AWSBlog de Big Data.

Considerações e limitações

- No momento, o Amazon Athena para Apache Spark está disponível nas seguintes Regiões da AWS:
  - Ásia-Pacífico (Mumbai)
  - Ásia-Pacífico (Singapura)

  - Ásia-Pacífico (Sydney)
  - Ásia-Pacífico (Tóquio)
  - Europa (Frankfurt)
  - Europa (Irlanda)
  - Leste dos EUA (N. da Virgínia)
  - Leste dos EUA (Ohio)
  - Oeste dos EUA (Oregon)
- Não há suporte ao AWS Lake Formation.
- Tabelas que usam projeção de partição não são compatíveis.
- Grupos de trabalho habilitados para o Apache Spark podem usar o editor de cadernos do Athena, mas não o editor de consultas do Athena. Somente os grupos de trabalho do Athena podem usar o editor de consultas do Athena.
- Não há suporte para consultas de visualização entre mecanismos. As visualizações criadas pelo SQL do Athena não podem ser consultadas pelo Athena para Spark. Como as visualizações dos dois mecanismos são implementadas de maneira diferente, elas não são compatíveis para uso entre mecanismos.
- Não há compatibilidade com MLLib (biblioteca de machine learning do Apache Spark) e com o pacote `pyspark.ml`. Para obter uma lista de bibliotecas Python compatíveis, consulte a Lista de bibliotecas Python pré-instaladas.
- No momento, `pip install` não é compatível com as sessões do Athena para Spark.
- Somente uma sessão ativa por caderno é permitida.
- Quando vários usuários usam o console para abrir uma sessão existente em um grupo de trabalho, eles acessam o mesmo caderno. Para evitar confusão, abra apenas sessões criadas por você mesmo.
- Os domínios de hospedagem para aplicações do Apache Spark que você pode usar com o Amazon Athena (por exemplo, `analytics-gateway.us-east-1.amazonaws.com`) estão registrados na Lista Public Suffix List (PSL) da Internet. Se você precisar definir cookies confidenciais em seus domínios, recomendamos que use cookies com um prefixo `__Host-` para ajudar a defender o domínio contra tentativas de falsificação de solicitações entre sites (CSRF). Para obter mais informações, consulte a página Set-Cookie na documentação da Mozilla.org. para desenvolvedores.
- Para obter informações sobre como solucionar problemas relacionados com cadernos, sessões e grupos de trabalho do Spark no Athena, consulte Solução de problemas do Athena para Spark.

Conceitos básicos do Apache Spark no Amazon Athena

Para começar a usar o Apache Spark no Amazon Athena, primeiro, você deve criar um grupo de trabalho habilitado para Spark. Depois de alternar para o grupo de trabalho, é possível criar um caderno ou abrir um caderno existente. Quando você abre um caderno no Athena, uma nova sessão é iniciada para ele automaticamente e você pode trabalhar com ele diretamente no editor de cadernos do Athena.

> **Note**
>
> Certifique-se de criar um grupo de trabalho habilitado para Spark antes de tentar criar um caderno.

Criação de um grupo de trabalho habilitado para Spark no Athena

É possível usar grupos de trabalho no Athena para agrupar usuários, equipes, aplicações ou workloads, além de monitorar custos. Para usar o Apache Spark no Amazon Athena, crie um grupo de trabalho do Amazon Athena que usa um mecanismo Spark.

> **Note**
>
> Grupos de trabalho habilitados para o Apache Spark podem usar o editor de cadernos do Athena, mas não o editor de consultas do Athena. Somente os grupos de trabalho do Athena podem usar o editor de consultas do Athena.

Para criar um grupo de trabalho habilitado para Spark no Athena

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.

3. No painel de navegação, escolha Global networks (Redes globais).
4. Na página Workgroups (Grupos de trabalho), escolha Create workgroup (Criar grupo de trabalho).
5. Em Workgroup name (Nome do grupo de trabalho), insira um nome para seu grupo de trabalho do Apache Spark.
6. (Opcional) Em Description (Descrição), insira uma descrição para seu grupo de trabalho.
7. Em Analytics engine (Mecanismo de análise), escolha Apache Spark.

> **Note**
>
> Após a criação de um grupo de trabalho, o tipo de mecanismo de análise do grupo de trabalho não poderá ser alterado. Por exemplo, um grupo de trabalho que usa a versão 3 do mecanismo Athena não poderá ser alterado para um grupo de trabalho que usa a versão 3 do mecanismo PySpark.

8. Para fins deste tutorial, selecione Turn on example notebook (Ativar caderno de exemplo). Esse recurso opcional adiciona um caderno de exemplo com o nome `example-notebook-`*`random_string`* ao seu grupo de trabalho e adiciona permissões relacionadas ao AWS Glue que o caderno usa para criar, apresentar e excluir tabelas e bancos de dados específicos em sua conta, além de permissões de leitura no Amazon S3 para o conjunto de dados de amostra. Para visualizar as permissões adicionadas, escolha View additional permissions details (Apresentar detalhes adicionais das permissões).

> **Note**
>
> A execução do caderno de exemplo poderá gerar custos adicionais.

9. Em Additional configuration (Configurações adicionais), siga um destes procedimentos:

   - Use a configuração Use defaults (Usar padrões). Essa opção é a padrão e ajuda você a começar com seu grupo de trabalho habilitado para Spark. Com essa opção, o Athena cria um perfil do IAM e um local para os resultados dos cálculos no Amazon S3 para você. O nome do perfil do IAM e o local do bucket S3 a serem criados são exibidos na caixa abaixo do cabeçalho Additional configurations (Configurações adicionais).
   - Desative a configuração Use defaults (Usar padrões) e, em seguida, prossiga com as etapas na seção Como especificar suas próprias configurações de grupo de trabalho para configurar seu grupo de trabalho manualmente.

10. (Opcional) Tags (Etiquetas): use essa opção para adicionar etiquetas ao seu grupo de trabalho. Para ter mais informações, consulte Etiquetar recursos do Athena.
11. Escolha Create workgroup (Criar grupo de trabalho). Uma mensagem informará que o grupo de trabalho foi criado com êxito e o grupo de trabalho aparecerá na lista de grupos de trabalho.

Como especificar suas próprias configurações de grupo de trabalho

Se você deseja especificar o próprio perfil do IAM e o local para os resultados dos cálculos para seu caderno, siga as etapas nesta seção. Se você escolheu Use defaults (Usar padrões) para a opção Additional configurations (Configurações adicionais), ignore esta seção e vá diretamente para Como abrir o explorador de cadernos e alternar grupos de trabalho .

O procedimento a seguir pressupõe que você concluiu as etapas 1 a 9 do procedimento Para criar um grupo de trabalho habilitado para Spark no Athena na seção anterior.

Para especificar suas próprias configurações de grupo de trabalho

1. Se você deseja criar ou usar o próprio perfil do IAM, ou configurar a criptografia de caderno, expanda IAM role configuration (Configuração do perfil do IAM).

   - Em Service Role (Perfil de serviço), escolha uma das opções a seguir:

- Create a service role (Criar um perfil de serviço): escolha essa opção para que o Athena crie um perfil de serviço para você. Para visualizar as permissões concedidas pelo perfil, selecione View permission details (Exibir detalhes da permissão).
- Choose an existing service role (Escolher um perfil de serviço existente): selecione um perfil a partir do menu suspenso. O perfil que você escolher deve incluir as permissões na primeira opção. Para obter mais informações sobre permissões para grupos de trabalho habilitados para cadernos, consulte Solução de problemas de grupos de trabalho habilitados para Spark.

- Em Notebook and calculation code encryption key management (Gerenciamento de chave de criptografia para códigos de cálculo e cadernos), escolha uma das opções a seguir:
  - Owned by Amazon Athena (Propriedade do Amazon Athena): a chave AWS KMS pertence e é gerenciada pelo Amazon Athena. Não há custo adicional para usar essa chave.
  - A symmetric key stored in your account, owned and managed by you (Uma chave simétrica armazenada em sua conta, de sua propriedade e gerenciada por você): para essa opção, siga um destes procedimentos:
    - Para usar uma chave existente, use a caixa de pesquisa para escolher um AWS KMS ou insira um ARN de chave.
    - Para criar uma chave no console do AWS KMS, escolha Criar uma chave do AWS KMS. Seu perfil de execução deve ter permissões para usar a chave que você criar.

> **Important**
>
> Quando você altera a AWS KMS key para um grupo de trabalho, os cadernos gerenciados antes da atualização ainda fazem referência à antiga chave do KMS. Os cadernos gerenciados após a atualização usarão a nova chave do KMS. Para atualizar os cadernos antigos para fazer referência à nova chave do KMS, exporte e importe cada um dos cadernos antigos. Se você excluir a antiga chave do KMS antes de atualizar as referências dos cadernos antigos para a nova chave do KMS, os cadernos antigos não poderão mais ser descriptografados e não poderão ser recuperados.
>
> Esse comportamento também se aplica a atualizações de aliases, que são nomes amigáveis para chaves do KMS. Quando você atualiza um alias de chave do KMS para direcionar para uma nova chave do KMS, os cadernos gerenciados antes da atualização do alias ainda fazem referência à antiga chave do KMS e os cadernos

gerenciados após a atualização do alias usam a nova chave do KMS. Considere esses pontos antes de atualizar suas chaves ou aliases do KMS.

2. Se você desejar especificar as próprias configurações para os resultados dos cálculos, expanda Calculation result settings (Configurações de resultados dos cálculos) e, em seguida, escolha uma das opções a seguir.

   - Create a new S3 bucket (Criar um novo bucket do S3): essa opção cria um bucket do Amazon S3 em sua conta para os resultados dos cálculos. O nome do bucket tem o formato `account_id-region-athena-results-bucket-alphanumeric_id` e usa as configurações ACLs desabilitadas, o acesso público bloqueado, o controle de versionamento desabilitado e o proprietário do depósito aplicado.
   - Choose an existing S3 location (Escolher um local existente do S3): para essa opção, faça o seguinte:
     - Insira o caminho no S3 para um local existente na caixa de pesquisa ou escolha Browse S3 (Procurar no S3) para escolher um bucket em uma lista.

       > **Note**
       >
       > Ao selecionar um local existente no Amazon S3, não acrescente uma barra (/) ao local. Isso faz com que o link para o local dos resultados dos cálculos na página de detalhes dos cálculo direcione para o diretório incorreto. Se isso ocorrer, edite o local dos resultados do grupo de trabalho para remover a barra ao final.

     - (Opcional) Escolha View (Visualizar) para abrir a página Buckets do console do Amazon S3, na qual você pode visualizar mais informações sobre o bucket existente escolhido.
     - (Opcional) Em Expected bucket owner (Proprietário esperado do bucket), insira o ID da conta da AWS de quem você espera que seja o proprietário do bucket do local de saída dos resultados da consulta. Recomendamos que você escolha essa opção, como uma medida de segurança adicional, sempre que possível. Se o ID da conta do proprietário do bucket não corresponder ao ID especificado, as tentativas de saída para o bucket falharão. Para obter informações detalhadas, consulte Verificar propriedade do bucket com a condição de proprietário do bucket no Guia do usuário do Amazon S3.
     - (Opcional) Selecione Assign bucket owner full control over query results (Atribuir controle total ao proprietário do bucket sobre os resultados da consulta) se o local para os resultados dos cálculos pertencer a outra conta e você desejar conceder controle total sobre os resultados da consulta à outra conta.

3. (Opcional) Selecione Encrypt calculation results (Criptografar resultados dos cálculos) e escolha uma das opções a seguir:

   - SSE_S3: essa é uma chave de criptografia do lado do servidor gerenciada pelo S3.
   - SSE_KMS: uma chave que você fornece. Em Escolher uma chave do AWS KMS, é possível escolher uma das seguintes opções:
     - Usar chave de propriedade da AWS: use uma chave de propriedade da AWS e gerenciada para você.
     - Escolher uma outra chave do AWS KMS (avançado): escolha ou crie uma chave.
       - Para usar uma chave existente, use a caixa de pesquisa para escolher um AWS KMS ou insira um ARN de chave.
       - Para criar uma chave no console do KMS, escolha Criar uma chave do AWS KMS. Após terminar a criação da chave no console do KMS, retorne à página Criar grupo de trabalho no console do Athena e use a caixa de pesquisa Escolher uma chave do AWS KMS ou inserir um ARN para escolher a chave que você acabou de criar.
4. (Opcional) Other settings (Outras configurações): expanda essa opção para habilitar ou desabilitar a opção Publish CloudWatch metrics (Publicar métricas do CloudWatch) para o grupo de trabalho. Esse campo é selecionado por padrão. Para ter mais informações, consulte Monitoramento de cálculos do Apache Spark com métricas do CloudWatch.
5. (Opcional) Tags (Etiquetas): use essa opção para adicionar etiquetas ao seu grupo de trabalho. Para ter mais informações, consulte Etiquetar recursos do Athena.
6. Escolha Create workgroup (Criar grupo de trabalho). Uma mensagem informará que o grupo de trabalho foi criado com êxito e o grupo de trabalho aparecerá na lista de grupos de trabalho.

Como abrir o explorador de cadernos e alternar grupos de trabalho

Antes de poder usar o grupo de trabalho habilitado para Spark que acabou de criar, você deve alternar para o grupo de trabalho. Para alternar os grupos de trabalho habilitados para Spark, você pode usar a opção Workgroup (Grupo de trabalho) no explorador de cadernos ou no editor de cadernos.

> **Note**
>
> Antes de começar, verifique se o navegador não bloqueia cookies de terceiros. Qualquer navegador que bloqueie cookies de terceiros por padrão ou graças a uma configuração

habilitada pelo usuário impedirá a inicialização dos cadernos. Para mais informações sobre o gerenciamento de cookies, consulte:

- Chrome
- Firefox
- Safari

Para abrir o explorador de cadernos e alternar grupos de trabalho

1. No painel de navegação, escolha Notebook explorer (Explorador de cadernos).
2. Use a opção Workgroup (Grupo de trabalho) no canto superior direito do console para escolher o grupo de trabalho habilitado para Spark que você criou. O caderno de exemplo é mostrado na lista de cadernos.

   É possível usar o explorador de cadernos das seguintes maneiras:

   - Escolha o nome vinculado de um caderno para abrí-lo em uma nova sessão.
   - Para renomear, excluir ou exportar seu caderno, use o menu Actions (Ações).
   - Para importar um arquivo do caderno, escolha Import file (Importar arquivo).
   - Para criar um caderno, escolha Create notebook (Criar caderno).

Como executar o caderno de exemplo

O caderno de amostra consulta dados de um conjunto de dados de uma viagem de táxi da cidade de Nova York disponível publicamente. O caderno tem exemplos que mostram como trabalhar com o Spark DataFrames, o Spark SQL e o AWS Glue Data Catalog.

Para executar o caderno de exemplo

1. No explorador de cadernos, escolha o nome vinculado ao caderno de exemplo.

   Isso inicia uma sessão do caderno com os parâmetros padrões e abre o caderno no editor de cadernos. Uma mensagem informará que uma nova sessão do Apache Spark foi iniciada usando os parâmetros padrões (máximo de 20 DPUs).
2. Para executar as células em ordem e observar os resultados, escolha o botão Run (Executar) uma vez para cada célula do caderno.

- Role para baixo para visualizar os resultados e exibir novas células.
- Para as células que têm cálculos, uma barra de progresso mostra a porcentagem concluída, o tempo decorrido e o tempo restante.
- O caderno de exemplo cria um banco de dados e uma tabela de amostra em sua conta. A célula final os remove como uma etapa de limpeza.

> **Note**
>
> Se você alterar os nomes de pastas, tabelas ou bancos de dados no caderno de exemplo, certifique-se de que essas alterações sejam refletidas nos perfis do IAM usados. Caso contrário, o caderno poderá apresentar falhas devido a permissões insuficientes.

Como editar os detalhes da sessão

Depois de iniciar uma sessão de caderno, é possível editar os detalhes da sessão, como formato de tabela, criptografia, tempo limite de inatividade da sessão e o número máximo simultâneo de unidades de processamento de dados (DPUs) que deseja usar. Uma DPU é uma medida relativa do poder de processamento que consiste em uma capacidade computacional de 4 vCPUs e 16 GB de memória.

Para editar os detalhes da sessão

1. No editor de cadernos, no menu Session (Sessão) no canto superior direito, escolha Edit session (Editar sessão).
2. Na caixa de diálogo Editar detalhes da sessão, na seção Propriedades do Spark, escolha ou insira valores para as seguintes opções:

   - Formato de tabela adicional: escolha Linux Foundation Delta Lake, Apache Hudi, Apache Iceberg ou Personalizado.
     - Para as opções de tabela Delta, Hudi ou Iceberg, as propriedades de tabela necessárias para o formato de tabela correspondente são fornecidas automaticamente nas opções Editar na tabela e Editar em JSON. Para obter mais informações sobre como usar esses formatos de tabela, consulte Usar formatos de tabela que não sejam do Hive no Amazon Athena para Apache Spark.

- Para adicionar ou remover propriedades de tabela para Personalizada ou outros tipos de tabela, use as opções Editar na tabela e Editar em JSON.
- Para a opção Editar na tabela, escolha Adicionar propriedade para adicionar uma propriedade ou Remover para remover uma propriedade. Para inserir os nomes das propriedades e os respectivos valores, use as caixas Chave e Valor.
- Na opção Editar em JSON, use o editor de texto JSON para editar a configuração diretamente.
  - Para copiar o texto JSON para a área de transferência, escolha Copiar.
  - Para remover todo o texto do editor JSON, escolha Limpar.
  - Para configurar a quebra de linha ou escolher um tema de cores para o editor JSON, escolha o ícone de configurações (engrenagem).

- Ativar a criptografia do Spark: selecione esta opção para criptografar dados gravados em disco e enviados pelos nós de rede do Spark. Para ter mais informações, consulte Habilitar a criptografia do Apache Spark.

3. Na seção Parâmetros da sessão, escolha ou insira valores para as seguintes opções:

- Session idle timeout (Tempo limite de inatividade da sessão): escolha ou insira um valor entre 1 e 480 minutos. O padrão é 20.
- Coordinator size (Tamanho do coordenador): um coordenador corresponde a um executor especial que realiza a orquestração do trabalho de processamento e gerencia outros executores em uma sessão do caderno. Atualmente, 1 DPU é o valor padrão e o único possível.
- Executor size (Tamanho do executor): um executor corresponde a menor unidade de computação que uma sessão do caderno pode solicitar do Athena. Atualmente, 1 DPU é o valor padrão e o único possível.
- Max concurrent value (Valor simultâneo máximo): o número máximo de DPUs que podem ser executadas simultaneamente. O padrão é 20, o mínimo é 3 e o máximo é 60. Aumentar esse valor não alocará recursos adicionais automaticamente, mas o Athena tentará alocar até o máximo especificado quando a carga computacional exigir e quando os recursos estiverem disponíveis.

4. Escolha Salvar.
5. Na solicitação Confirm edit (Confirmar edição), escolha Confirm (Confirmar).

O Athena salva seu caderno e inicia uma nova sessão com os parâmetros que você especificou. Uma barra de notificação no editor do caderno informará que uma nova sessão foi iniciada com os parâmetros modificados.

> **Note**
>
> O Athena lembrará suas configurações de sessão para o caderno. Se você editar os parâmetros de uma sessão e, em seguida, encerrá-la, o Athena usará os parâmetros de sessão que você configurou na próxima vez que iniciar uma sessão para o caderno.

Como visualizar detalhes da sessão e dos cálculos

Após executar o caderno, você poderá visualizar os detalhes da sessão e dos cálculos.

Para visualizar detalhes da sessão e dos cálculos

1. No menu Session (Sessão) no canto superior direito, escolha View details (Exibir detalhes).
   - A guia Current session (Sessão atual) apresenta informações sobre a sessão atual, incluindo o ID da sessão, o horário de criação, o status e o grupo de trabalho.
   - A guia History (Histórico) lista os IDs das sessões anteriores. Para visualizar os detalhes de uma sessão anterior, escolha a guia History (Histórico) e, em seguida, escolha um ID de sessão na lista.
   - A seção Calculations (Cálculos) apresenta uma lista dos cálculos executados na sessão.
2. Para visualizar os detalhes de um cálculo, escolha o ID do cálculo.
3. Na página Calculation details (Detalhes do cálculo), você pode fazer o seguinte:
   - Para visualizar o código para o cálculo, consulte a seção Code (Código).
   - Para visualizar os resultados do cálculo, selecione a guia Results (Resultados).
   - Para baixar dos resultados que você visualiza em formato de texto, escolha Download results (Fazer download dos resultados).
   - Para visualizar informações sobre os resultados dos cálculos no Amazon S3, escolha View in S3 (Visualizar no S3).

Encerrar uma sessão

Para encerrar uma sessão do caderno

1. No editor de cadernos, no menu Session (Sessão) no canto superior direito, escolha Terminate (Encerrar).
2. Na solicitação Confirm session termination (Confirmar encerramento da sessão), escolha Confirm (Confirmar). Seu caderno será salvo e você retornará ao editor de cadernos.

> **Note**
>
> Fechar uma guia do caderno no editor de cadernos não encerrará por si só a sessão de um caderno ativo. Se você deseja garantir que a sessão seja encerrada, use a opção Terminate (Encerrar) em Session (Sessão).

Como criar seu próprio caderno

Após a criação de um grupo de trabalho do Athena habilitado para Spark, é possível criar seu próprio caderno.

Para criar um caderno

1. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.
2. No painel de navegação do console do Athena, escolha Notebook explorer (Explorador de cadernos) ou Notebook editor (Editor de cadernos).
3. Execute um destes procedimentos:
   - No Notebook explorer (Explorador de cadernos), escolha Create notebook (Criar caderno).
   - No Notebook editor (Editor de cadernos), escolha Create notebook (Criar caderno) ou selecione o ícone de adição (+) para adicionar um caderno.
4. Na caixa de diálogo Create notebook (Criar caderno), em Notebook name (Nome do caderno), insira um nome.
5. (Opcional) Expanda Propriedades do Spark e escolha ou insira valores para as seguintes opções:

- Formato de tabela adicional: escolha Linux Foundation Delta Lake, Apache Hudi, Apache Iceberg ou Personalizado.
  - Para as opções de tabela Delta, Hudi ou Iceberg, as propriedades de tabela necessárias para o formato de tabela correspondente são fornecidas automaticamente nas opções Editar na tabela e Editar em JSON. Para obter mais informações sobre como usar esses formatos de tabela, consulte Usar formatos de tabela que não sejam do Hive no Amazon Athena para Apache Spark.
  - Para adicionar ou remover propriedades de tabela para Personalizada ou outros tipos de tabela, use as opções Editar na tabela e Editar em JSON.
  - Para a opção Editar na tabela, escolha Adicionar propriedade para adicionar uma propriedade ou Remover para remover uma propriedade. Para inserir os nomes das propriedades e os respectivos valores, use as caixas Chave e Valor.
  - Na opção Editar em JSON, use o editor de texto JSON para editar a configuração diretamente.
    - Para copiar o texto JSON para a área de transferência, escolha Copiar.
    - Para remover todo o texto do editor JSON, escolha Limpar.
    - Para configurar a quebra de linha ou escolher um tema de cores para o editor JSON, escolha o ícone de configurações (engrenagem).
- Ativar a criptografia do Spark: selecione esta opção para criptografar dados gravados em disco e enviados pelos nós de rede do Spark. Para ter mais informações, consulte Habilitar a criptografia do Apache Spark.

6. (Opcional) Expanda Session parameters (Parâmetros da sessão) e, em seguida, escolha ou insira valores para as opções a seguir:

   - Session idle timeout (Tempo limite de inatividade da sessão): escolha ou insira um valor entre 1 e 480 minutos. O padrão é 20.
   - Coordinator size (Tamanho do coordenador): um coordenador corresponde a um executor especial que realiza a orquestração do trabalho de processamento e gerencia outros executores em uma sessão do caderno. Atualmente, 1 DPU é o valor padrão e o único possível. Uma DPU (unidade de processamento de dados) corresponde a uma medida relativa de poder de processamento que consiste em 4 vCPUs de capacidade computacional e 16 GB de memória.

- Executor size (Tamanho do executor): um executor corresponde a menor unidade de computação que uma sessão do caderno pode solicitar do Athena. Atualmente, 1 DPU é o valor padrão e o único possível.
- Max concurrent value (Valor simultâneo máximo): o número máximo de DPUs que podem ser executadas simultaneamente. O padrão é 20 e o máximo é 60. Aumentar esse valor não alocará recursos adicionais automaticamente, mas o Athena tentará alocar até o máximo especificado quando a carga computacional exigir e quando os recursos estiverem disponíveis.

7. Escolha Criar. Seu caderno será aberto em uma nova sessão no editor de cadernos.

Como abrir um caderno criado anteriormente

Para abrir um caderno criado anteriormente

1. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.
2. No painel de navegação do console do Athena, escolha Notebook editor (Editor de cadernos) ou Notebook explorer (Explorador de cadernos).
3. Execute um destes procedimentos:

   - No Notebook editor (Editor de cadernos), escolha um caderno na lista Recent notebooks (Cadernos recentes) ou Saved notebooks (Cadernos salvos). O caderno será aberto em uma nova sessão.
   - No Notebook explorer (Explorador de cadernos), escolha o nome de um caderno na lista. O caderno será aberto em uma nova sessão.

Para obter mais informações sobre como gerenciar seus arquivos de cadernos, consulte Gerenciamento de arquivos de cadernos.

Como trabalhar com cadernos

Você gerencia seus cadernos no explorador de cadernos do Athena e os edita e executa em sessões usando o editor de cadernos do Athena. É possível configurar o uso da DPU para suas sessões de cadernos de acordo com seus requisitos.

Ao interromper um caderno, você encerra a sessão associada. Todos os arquivos serão salvos, mas as alterações em andamento nas variáveis, funções e classes declaradas serão perdidas. Quando você reinicia o caderno, o Athena recarrega os arquivos do caderno e você pode executar o código novamente.

Sessões e cálculos

Cada caderno está associado a um único kernel do Python e executa o código Python. Um caderno pode ter uma ou mais células que contêm comandos. Para executar as células em um caderno, primeiro crie uma sessão para ele. As sessões acompanham as variáveis e o estado dos cadernos.

Executar uma célula em um caderno significa executar um cálculo na sessão atual. Os cálculos avançam no estado do caderno e podem executar tarefas como leitura do Amazon S3 ou gravação para outros armazenamentos de dados. Enquanto uma sessão estiver em execução, os cálculos usam e modificam o estado que é mantido para o caderno.

Quando você não precisar mais do estado, poderá encerrar uma sessão. Ao encerrar uma sessão, o caderno permanecerá, mas as variáveis e outras informações de estado serão destruídas. Se você precisar trabalhar em diversos projetos ao mesmo tempo, poderá criar uma sessão para cada projeto, e as sessões serão independentes umas das outras.

As sessões têm capacidade de computação dedicada, que é medida em DPU. Ao criar uma sessão, é possível atribuir à sessão um número de DPUs. Sessões diferentes podem ter capacidades diferentes, dependendo dos requisitos para a tarefa.

Uso do editor de cadernos do Athena

O editor de cadernos do Athena é um ambiente interativo para gravar e executar códigos. As seções a seguir descrevem os recursos do ambiente.

Comparação entre o modo de comando e o modo de edição

O editor de cadernos tem uma interface de usuário modal: um modo de edição para inserir texto em uma célula e um modo de comando para emitir comandos para o próprio editor, como copiar, colar ou executar.

Para usar o modo de edição e o modo de comando, você pode executar as tarefas a seguir:

- Para entrar no modo de edição, pressione **ENTER** ou selecione uma célula. Quando uma célula está no modo de edição, ela tem uma margem esquerda na cor verde.

- Para entrar no modo de comando, pressione **ESC** ou clique na parte exterior de uma célula. Observe que os comandos geralmente se aplicam somente à célula selecionada no momento, não a todas as células. Quando o editor está no modo de comando, a célula tem uma margem esquerda na cor azul.
- No modo de comando, é possível usar atalhos de teclado e o menu acima do editor, mas não é possível inserir texto em células individuais.
- Para selecionar uma célula, escolha a célula.
- Para selecionar todas as células, pressione **Ctrl+A** (Windows) ou **Cmd+A** (Mac).

Menu do editor de cadernos

Os ícones no menu presente na parte superior do editor de cadernos oferecem as seguintes opções:

- Salvar: salva o estado atual do caderno.
- Inserir célula abaixo: adiciona uma nova célula (vazia) abaixo da célula selecionada no momento.
- Recortar células selecionadas: remove a célula selecionada de sua localização atual e copia a célula para a memória.
- Copiar células selecionadas: copia a célula selecionada para a memória.
- Colar células abaixo: cola a célula copiada abaixo da célula atual.
- Mover as células selecionadas para cima: move a célula atual acima da célula que está na posição acima.
- Mover as células selecionadas para baixo: move a célula atual abaixo da célula que está na posição abaixo.
- Executar: executa a célula atual (selecionada). A saída é exibida logo abaixo da célula atual.
- Executar tudo: executa todas as células presentes no caderno. A saída para cada célula é exibida logo abaixo das respectivas células.
- Parar (interromper o kernel): interrompe o caderno atual ao interromper o kernel.
- Opção de formato: seleciona o formato da célula, que pode ser um dos seguintes:
  - Código: use para códigos do Python (padrão).
  - Markdown: use para inserir texto no formato markdown estilo GitHub. Para renderizar o markdown, realize a execução da célula.
  - Raw NBConvert: use para inserir texto em um formato não modificado. As células marcadas como Raw NBConvert podem ser convertidas em um formato diferente, como HTML, pela ferramenta de linha de comando nbconvert do Jupyter.

- Cabeçalho: use para alterar o nível do cabeçalho da célula.
- Paleta de comandos: contém comandos do caderno Jupyter e seus atalhos de teclado. Para obter mais informações sobre os atalhos de teclado, consulte as seções posteriores neste documento.
- Sessão: use as opções desse menu para visualizar os detalhes de uma sessão, editar os parâmetros da sessão ou encerrá-la.

Atalhos de teclado do modo de comando

A seguir estão alguns atalhos de teclado conhecidos do modo de comando do editor de cadernos. Esses atalhos estão disponíveis após pressionar **ESC** para entrar no modo de comando. Para visualizar uma lista completa de comandos disponíveis no editor, pressione **ESC + H**.

Chave	Ação
1 - 6	Alterar o tipo de célula para markdown e definir o nível do cabeçalho para o número digitado
a	Criar uma célula acima da célula atual
b	Criar uma célula abaixo da célula atual
c	Copiar a célula atual para a memória
d d	Excluir a célula atual
h	Exibir a tela de ajuda para os atalhos de teclado
j	Ir uma célula abaixo
k	Ir uma célula acima
m	Alterar o formato atual da célula para markdown
r	Alterar o formato atual da célula para raw
s	Salvar o caderno
v	Colar o conteúdo da memória na célula atual

Chave	Ação
x	Recortar a célula ou as células selecionadas
y	Alterar o formato da célula para código
z	Desfazer
Ctrl+Enter	Executar a célula atual e entrar no modo de comando
Shift+Enter ou **Alt+Enter**	Executar a célula atual e criar uma nova célula abaixo da saída, além de inserir a nova célula no modo de edição
Space	Uma página abaixo
Shift+Space	Uma página acima
Shift + L	Alternar a visibilidade dos números de linha nas células

Como editar atalhos do modo de comando

O editor de cadernos tem a opção para personalizar os atalhos de teclado do modo de comando.

Para editar atalhos do modo de comando

1. No menu do editor de cadernos, escolha a Command palette (Paleta de comandos).
2. Na paleta de comandos, escolha o comando Edit command mode keyboard shortcuts (Editar atalhos de teclado do modo de comando).
3. Use a interface Edit command mode shortcuts (Editar atalhos do modo de comando) para mapear ou mapear novamente os comandos que você deseja para o teclado.

   Para visualizar as instruções para edição dos atalhos do modo de comando, role até a parte inferior da tela do Edit command mode shortcuts (Editar atalhos do modo de comando).

Para obter informações sobre como usar comandos magic no Athena para Apache Spark, consulte Como usar comandos mágicos.

Como usar comandos mágicos

Comandos mágicos, ou mágicas, correspondem a comandos especiais que você pode executar em uma célula do caderno. Por exemplo, `%env` apresenta as variáveis de ambiente em uma sessão do caderno. O Athena oferece suporte para as funções mágicas no IPython 6.0.3.

Esta seção apresenta alguns dos principais comandos magic no Athena para Apache Spark.

- Para visualizar uma lista de comandos magic no Athena, execute o comando **`%lsmagic`** em uma célula de caderno.
- Para obter informações sobre como usar magic para criar grafos em cadernos do Athena, consulte Magics para criar grafos de dados.
- Para obter informações sobre outros comandos magic, consulte Comandos magic integrados na documentação do IPython.

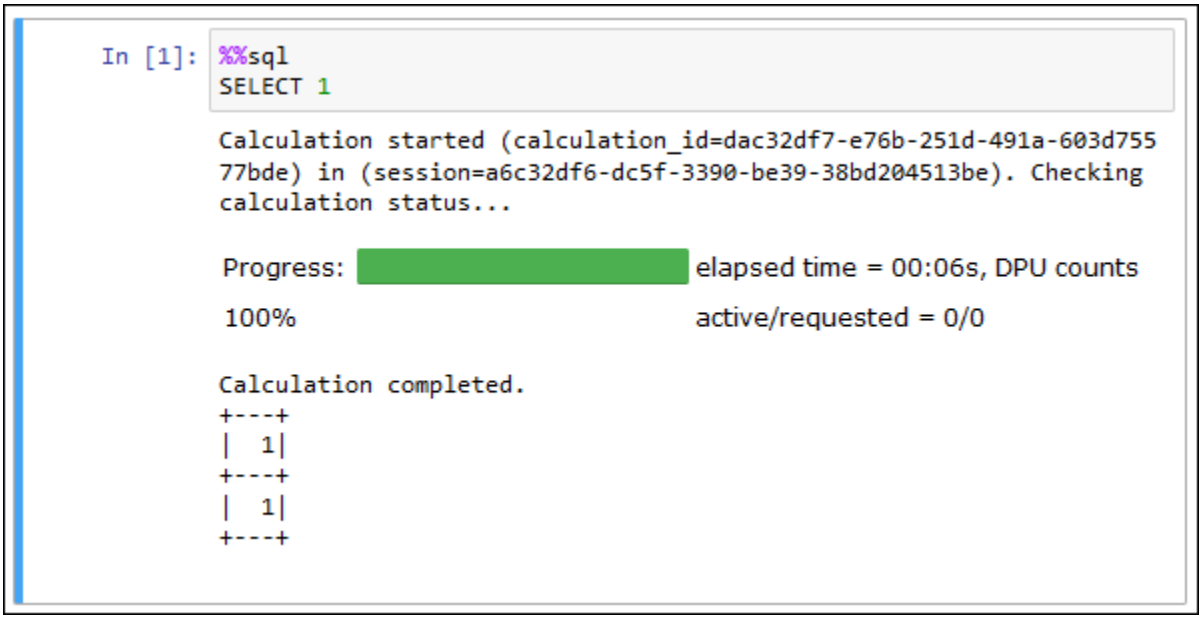
> **Note**
>
> Atualmente, o comando `%pip` apresenta falhas quando é executado. Esse é um problema conhecido.

Magics de células

Mágicas que são gravadas em diversas linhas são precedidas por um sinal de porcentagem duplo (`%%`) e são chamadas de funções mágicas de células ou mágicas de células.

%%sql

O magic de células permite executar instruções SQL diretamente sem precisar decorá-las com a instrução SQL do Spark. O comando também exibe a saída chamando implicitamente `.show()` no quadro de dados retornado.



O comando `%%sql` trunca automaticamente as saídas da coluna para uma largura de 20 caracteres. Essa definição não é configurável no momento. Para contornar essa limitação, use a sintaxe completa a seguir e modifique os parâmetros do método `show` adequadamente.

```
spark.sql("""YOUR_SQL""").show(n=number, truncate=number, vertical=bool)
```

- n `int`, opcional. O número de linhas a serem exibidas.
- truncar: `bool` ou `int`, opcional. Se `true`, trunca strings com mais de 20 caracteres. Quando definido como um número maior que 1, trunca strings de caracteres longas até o comprimento especificado e alinha as células à direita.
- vertical: `bool`, opcional. Se `true`, imprime as linhas de saída verticalmente (uma linha por valor de coluna).

Magics de linha

Mágicas que estão em uma única linha são precedidas por um sinal de porcentagem (%) e são chamadas de funções mágicas de linha ou mágicas de linha.

%help

Exibe descrições dos comandos magic disponíveis.

```
In [6]: %help

Available Magic Commands:
Magic | Input | Description
%session_id  | None | Return the session ID for the running session.
%status | None | Describes the current session and display SessionID, State,
WorkGroup, EngineVersion and StartTime
%help | None | Displays list of supported magics
%set_log_level | String | Sets the current log level to the provided log leve
ls (ERROR|INFO|WARNING etc)
%list_sessions | None | Lists the most recent sessions associated with the cu
rrent workgroup
%%sql | String | Run an SQL command against SparkSQL.
```

%list_sessions

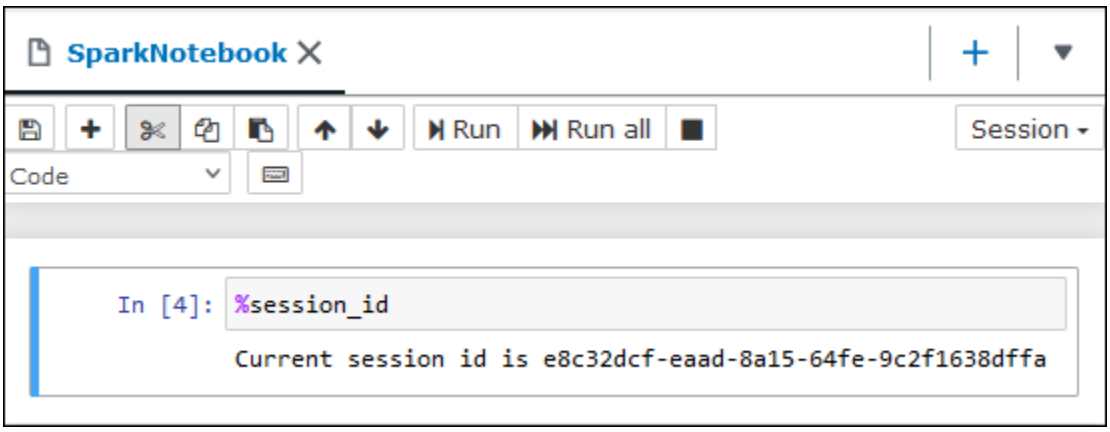
Lista as sessões associadas ao caderno. As informações de cada sessão incluem o ID da sessão, o status da sessão e a data e hora em que a sessão iniciou e terminou.

```
In [12]: %list_sessions
```

SessionId	Status	StartDateTime	EndDateTime
66c32de7-78b9-f2ee-6eb9-d8d9716c6ac8	IDLE	02/16/2023, 19:58:54	
ccc32dda-6dea-6277-d434-5c5da5e1a882	TERMINATED	02/16/2023, 19:30:24	02/16/2023, 19:51:53
e8c32dcf-eaad-8a15-64fe-9c2f1638dffa	TERMINATED	02/16/2023, 19:07:26	02/16/2023, 19:28:53

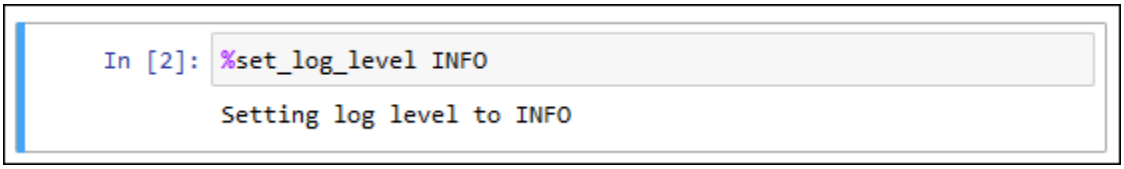
%session_id

Recupera o ID da sessão atual.



%set_log_level

Define ou redefine o logger para usar o nível de log especificado. Os valores possíveis são `DEBUG`, `ERROR`, `FATAL`,`INFO` e `WARN` ou `WARNING`. Os valores devem estar em letras maiúsculas e não devem estar delimitados entre aspas simples ou duplas.



%status

Descreve a sessão atual. A saída inclui o ID da sessão, o estado da sessão, o nome do grupo de trabalho, a versão do mecanismo PySpark e a hora de início da sessão. Esse comando magic requer uma sessão ativa para recuperar os detalhes da sessão.

Os seguintes valores de status são possíveis:

CREATING: a sessão está sendo iniciada, incluindo a aquisição de recursos.

CREATED: a sessão foi iniciada.

IDLE: a sessão pode aceitar um cálculo.

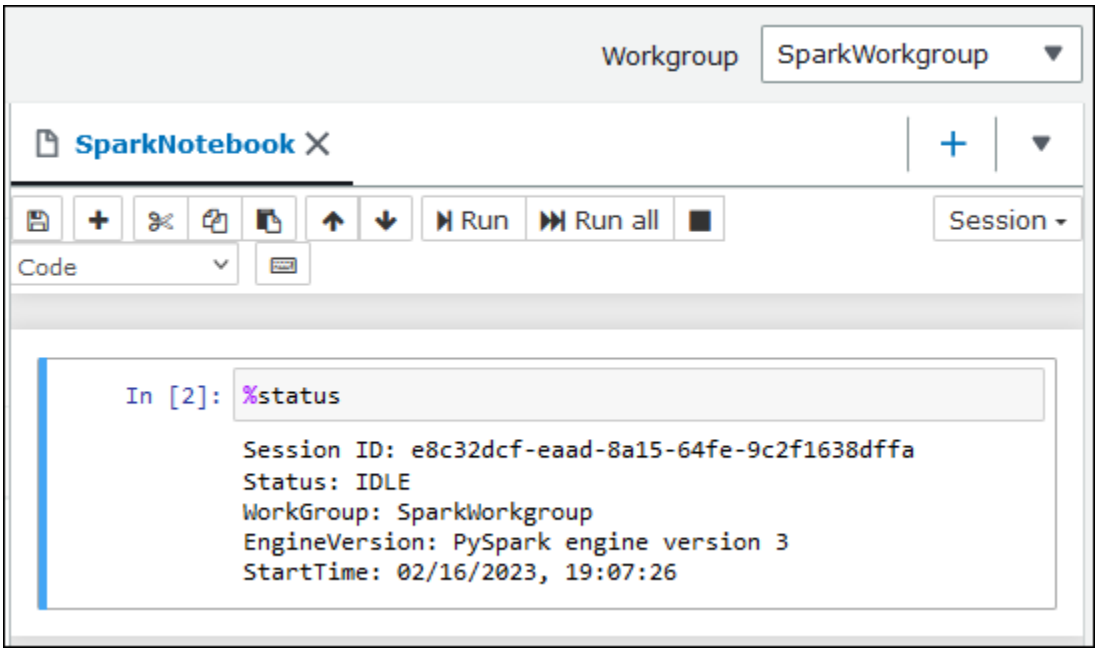
BUSY: a sessão está processando outra tarefa e não pode aceitar um cálculo.

TERMINATING: a sessão está no processo de desligamento.

TERMINATED: a sessão e os respectivos recursos não estão mais em execução.

DEGRADED: a sessão não tem coordenadores íntegros.

FAILED: devido a uma falha, a sessão e os respectivos recursos não estão mais em execução.



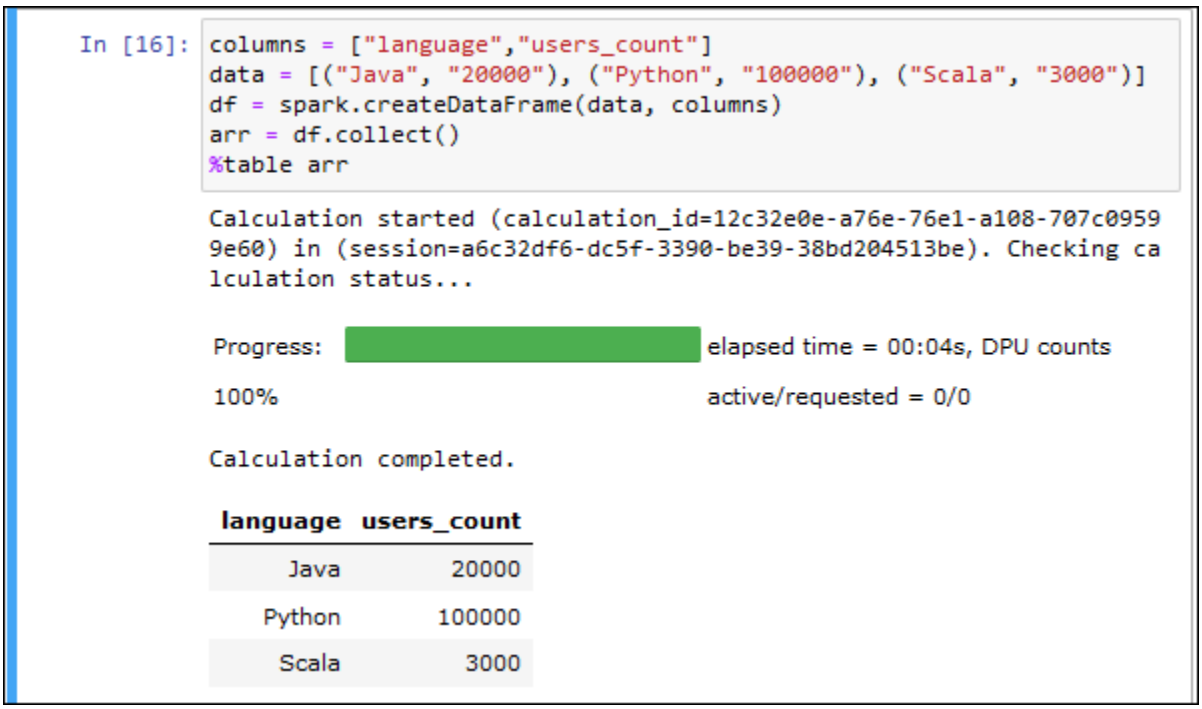
Magics para criar grafos de dados

Os magics de linha desta seção são especializados em renderizar dados para tipos específicos de dados ou em conjunto com bibliotecas de grafos.

%table

Use o comando magic `%table` para exibir dados do quadro de dados em formato de tabela.

O exemplo a seguir cria um quadro de dados com duas colunas e três linhas de dados e exibe os dados em formato de tabela.



%matplot

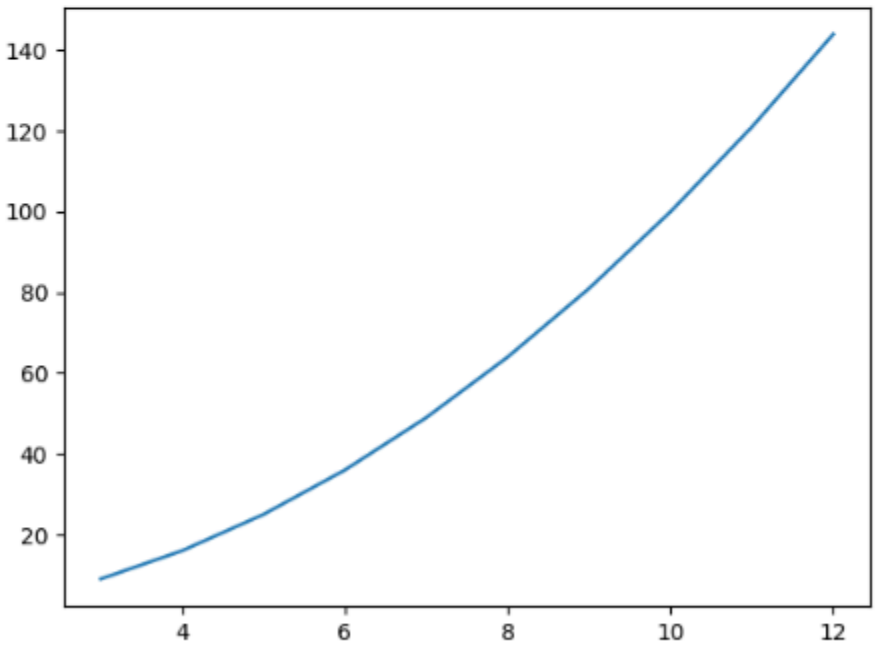
Matplotlib é uma biblioteca completa para criar visualizações estáticas, animadas e interativas em Python. Você pode usar o comando magic `%matplot` para criar um grafo depois de importar a biblioteca matplotlib para uma célula do caderno.

O exemplo a seguir importa a biblioteca matplotlib, cria um conjunto de coordenadas x e y e usa o comando magic `%matplot` para criar um grafo dos pontos.

```
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 stat
us...

Progress:                                elapsed time =

100%                                     00:02s

Calculation completed.
```




```
[<matplotlib.lines.Line2D object at 0x7f6e29e580>]
```

Usar bibliotecas matplotlib e seaborn em conjunto

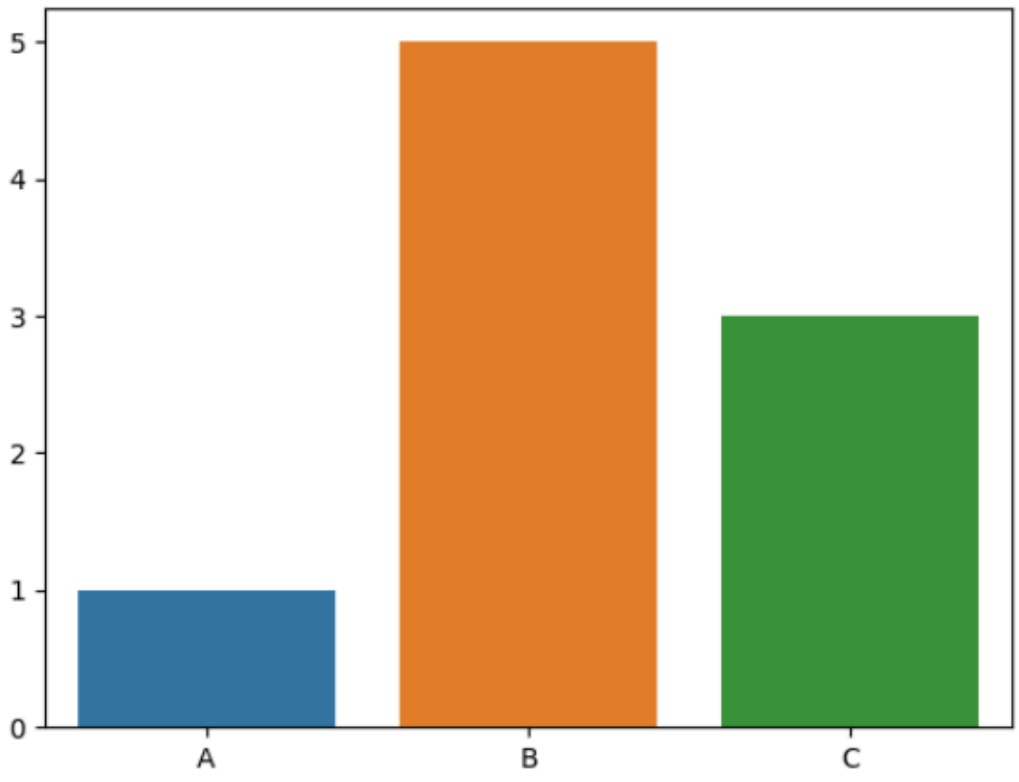
Seaborn é uma biblioteca para criação de gráficos estatísticos em Python. Baseia-se no matplotlib e integra-se estreitamente às estruturas de dados pandas (análise de dados Python). Também é possível usar o comando magic `%matplot` para renderizar dados seaborn.

O exemplo a seguir usa as bibliotecas matplotlib e seaborn para criar um grafo de barras simples.

```
import matplotlib.pyplot as plt
import seaborn as sns

x = ['A', 'B', 'C']
y = [1, 5, 3]

sns.barplot(x, y)
%matplot plt
```

In [1]:
```
import matplotlib.pyplot as plt
import seaborn as sns


x = ['A', 'B', 'C']
y = [1, 5, 3]

sns.barplot(x, y)
%matplot plt
```

```
Calculation started (calculation_id=08c32e1b-233b-4a72-6571-1ae7a28a7b78)
in (session=64c32e1a-f45e-1d52-b54b-85202e2a9233). Checking calculation st
atus...

Progress: 100% elapsed time = 00:04s

Calculation completed.
```

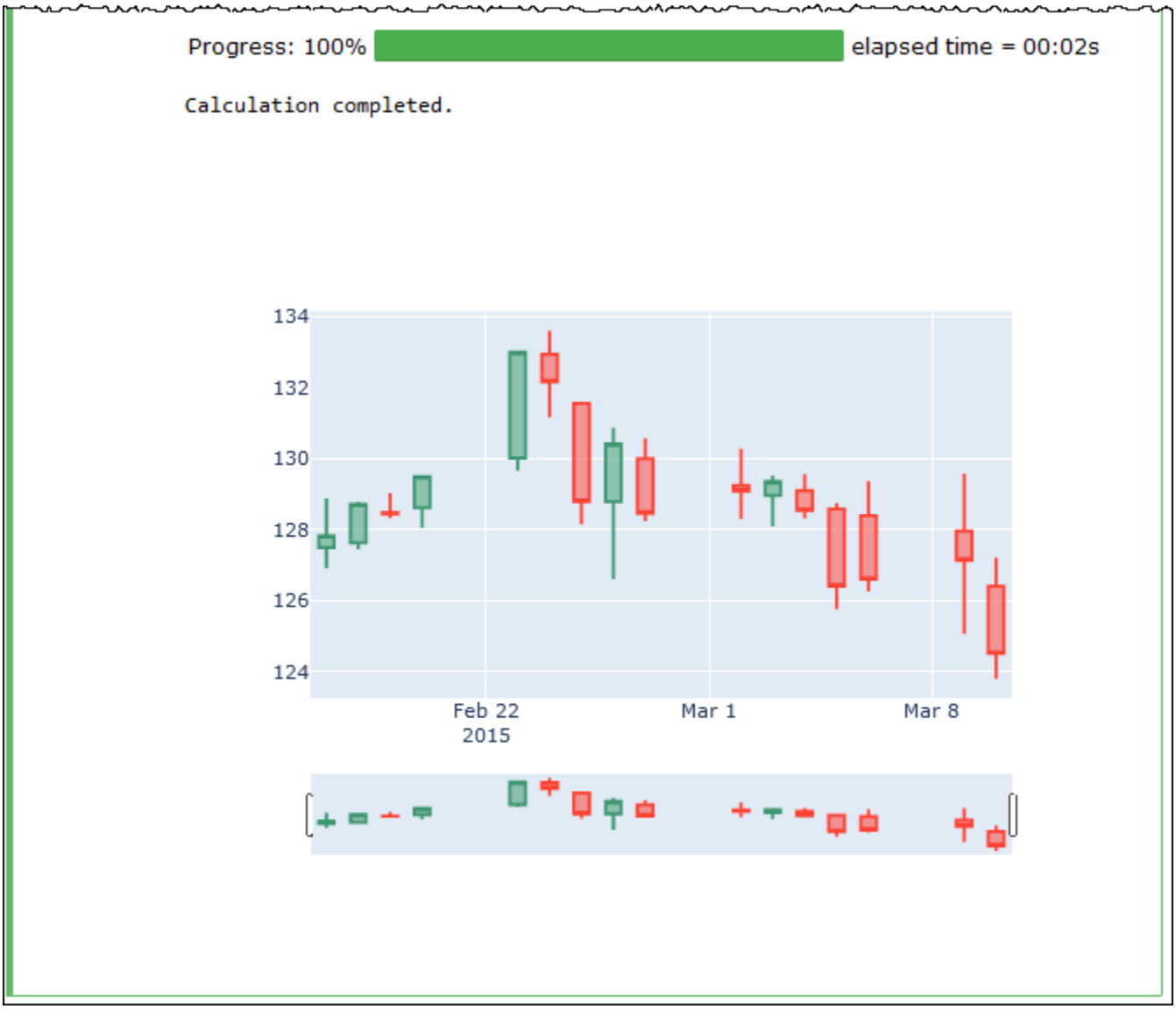


%plotly

Plotly é uma biblioteca de grafos de código aberto para Python que você pode usar para criar grafos interativos. Use o comando magic `%ploty` para renderizar dados ploty.

O exemplo a seguir usa as bibliotecas StringIO, plotly e pandas em dados de preços de ações para criar um grafo da atividade de ações de fevereiro e março de 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
```



Gerenciamento de arquivos de cadernos

Além de usar o explorador de cadernos para criar e abrir cadernos, também é possível usá-lo para renomear, excluir, exportar ou importar cadernos ou visualizar o histórico da sessão de um caderno.

Para renomear um caderno

1. Encerre todas as sessões ativas dos cadernos que deseja renomear. As sessões ativas dos cadernos devem ser encerradas antes que você possa renomeá-los.
2. Abra o Notebook explorer (Explorador de cadernos).
3. Na lista Notebooks (Cadernos), selecione o botão de opção do caderno que deseja renomear.
4. A partir do menu Actions (Ações), escolha Rename (Renomear).

5. Na solicitação Rename notebook (Renomear caderno), digite o novo nome e, em seguida, selecione Save (Salvar). O novo nome do caderno aparecerá na lista de cadernos.

Para excluir um caderno

1. Encerre todas as sessões ativas dos cadernos que deseja excluir. As sessões ativas dos cadernos devem ser encerradas antes que você possa excluí-los.
2. Abra o Notebook explorer (Explorador de cadernos).
3. Na lista Notebooks (Cadernos), selecione o botão de opção do caderno que deseja excluir.
4. No menu Actions (Ações), escolha Delete (Excluir).
5. Na solicitação Delete notebook? (Excluir caderno?), insira o nome do caderno e, em seguida, selecione Delete (Excluir) para confirmar a exclusão. O nome do caderno será removido da lista de cadernos.

Para exportar um caderno

1. Abra o Notebook explorer (Explorador de cadernos).
2. Na lista Notebooks (Cadernos), selecione o botão de opção do caderno que deseja exportar.
3. A partir do menu Actions (Ações), escolha Export (Exportar).

Para importar um caderno

1. Abra o Notebook explorer (Explorador de cadernos).
2. Escolha Import file (Importar arquivo).
3. Navegue até o local em que se encontra o arquivo que deseja importar em seu computador local e, em seguida, escolha Open (Abrir). O caderno importado aparecerá na lista de cadernos.

Para visualizar o histórico da sessão para um caderno

1. Abra o Notebook explorer (Explorador de cadernos).
2. Na lista Notebooks (Cadernos), selecione o botão de opção do caderno cujo histórico de sessão você deseja visualizar.
3. A partir do menu Actions (Ações), escolha Session history (Histórico da sessão).

4. Na guia History (Histórico), escolha um Session ID (ID de sessão) para visualizar as informações sobre a sessão e seus cálculos.

Usar formatos de tabela que não sejam do Hive no Amazon Athena para Apache Spark

Ao trabalhar com sessões e cadernos no Athena para Spark, é possível usar tabelas do Linux Foundation Delta Lake, do Apache Hudi e do Apache Iceberg, além das tabelas do Apache Hive.

Considerações e limitações

Ao usar formatos de tabela que não sejam do Apache Hive com o Athena para Spark, considere os seguintes pontos:

- Além do Apache Hive, somente um formato de tabela por caderno é compatível. Para usar vários formatos de tabela no Athena para Spark, crie um caderno separado para cada formato de tabela. Para obter informações sobre como criar cadernos no Athena para Spark, consulte Como criar seu próprio caderno.
- Os formatos de tabela Delta Lake, Hudi e Iceberg foram testados no Athena para Spark usando o AWS Glue como o metastore. Talvez seja possível usar outros metastores, mas não há suporte para esse uso atualmente.
- Para usar os outros formatos de tabela, substitua a propriedade `spark_catalog` padrão, conforme indicado no console do Athena e nesta documentação. Esses catálogos que não são do Hive podem ler tabelas do Hive, além de seus próprios formatos de tabela.

Versões de tabela

A tabela a seguir mostra as versões de tabela que não são do Hive compatíveis com o Amazon Athena para Apache Spark.

Formato da tabela	Versão com suporte
Apache Iceberg	1.2.1
Apache Hudi	0.13
Linux Foundation Delta Lake	2.0.2

No Athena para Spark, esses arquivos de formato de tabela `.jar` e suas dependências são carregados no caminho de classe dos drivers e executores do Spark.

Tópicos

- Apache Iceberg
- Apache Hudi
- Linux Foundation Delta Lake

Apache Iceberg

O Apache Iceberg é um formato de tabela aberta para grandes conjuntos de dados no Amazon Simple Storage Service (Amazon S3). Ele fornece performance rápida de consultas em tabelas grandes, confirmações atômicas, gravações simultâneas e evolução de tabelas compatível com SQL.

Para usar tabelas do Apache Iceberg no Athena para Spark, configure as propriedades do Spark a seguir. Essas propriedades são configuradas por padrão no console do Athena para Spark quando você escolhe Apache Iceberg como formato de tabela. Para obter as etapas, consulte Como editar os detalhes da sessão ou Como criar seu próprio caderno.

```
"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"
```

O procedimento a seguir mostra como usar uma tabela do Apache Iceberg em um caderno do Athena para Spark. Execute cada etapa em uma nova célula no caderno.

Para usar tabelas do Apache Iceberg no Athena para Spark

1. Defina as constantes a serem usadas no caderno.

   ```
   DB_NAME = "NEW_DB_NAME"
   TABLE_NAME = "NEW_TABLE_NAME"
   TABLE_S3_LOCATION = "s3://DOC-EXAMPLE-BUCKET"
   ```

2. Crie um DataFrame do Apache Spark.

```
columns = ["language","users_count"]
data = [("Golang", 3000)]
df = spark.createDataFrame(data, columns)
```

3. Crie um banco de dados.

```
spark.sql("CREATE DATABASE {} LOCATION '{}'".format(DB_NAME, TABLE_S3_LOCATION))
```

4. Crie uma tabela vazia do Apache Iceberg.

```
spark.sql("""
CREATE TABLE {}.{} (
language string,
users_count int
) USING ICEBERG
""".format(DB_NAME, TABLE_NAME))
```

5. Insira uma linha de dados na tabela.

```
spark.sql("""INSERT INTO {}.{} VALUES ('Golang',
 3000)""".format(DB_NAME, TABLE_NAME))
```

6. Confirme se é possível consultar a nova tabela.

```
spark.sql("SELECT * FROM {}.{}".format(DB_NAME, TABLE_NAME)).show()
```

Para obter mais informações e exemplos sobre como trabalhar com tabelas DataFrames e Iceberg do Spark, consulte Spark Queries na documentação do Apache Iceberg.

Apache Hudi

O Apache Hudi é um framework de gerenciamento de dados de código aberto que simplifica o processamento incremental de dados. As ações de inserção, atualização, upsert e exclusão em nível de registro são processadas com maior precisão, o que reduz a sobrecarga.

Para usar tabelas do Apache Hudi no Athena para Spark, configure as propriedades do Spark a seguir. Essas propriedades são configuradas por padrão no console do Athena para Spark quando você escolhe Apache Hudi como formato de tabela. Para obter as etapas, consulte Como editar os detalhes da sessão ou Como criar seu próprio caderno.

```
"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"
```

O procedimento a seguir mostra como usar uma tabela do Apache Hudi em um caderno do Athena para Spark. Execute cada etapa em uma nova célula no caderno.

Para usar tabelas do Apache Hudi no Athena para Spark

1. Defina as constantes a serem usadas no caderno.

```
DB_NAME = "NEW_DB_NAME"
TABLE_NAME = "NEW_TABLE_NAME"
TABLE_S3_LOCATION = "s3://DOC-EXAMPLE-BUCKET"
```

2. Crie um DataFrame do Apache Spark.

```
columns = ["language","users_count"]
data = [("Golang", 3000)]
df = spark.createDataFrame(data, columns)
```

3. Crie um banco de dados.

```
spark.sql("CREATE DATABASE {} LOCATION '{}'".format(DB_NAME, TABLE_S3_LOCATION))
```

4. Crie uma tabela vazia do Apache Hudi.

```
spark.sql("""
CREATE TABLE {}.{} (
language string,
users_count int
) USING HUDI
TBLPROPERTIES (
primaryKey = 'language',
type = 'mor'
);
""".format(DB_NAME, TABLE_NAME))
```

5. Insira uma linha de dados na tabela.

```
spark.sql("""INSERT INTO {}.{} VALUES ('Golang',
 3000)""".format(DB_NAME,TABLE_NAME))
```

6. Confirme se é possível consultar a nova tabela.

```
spark.sql("SELECT * FROM {}.{}".format(DB_NAME, TABLE_NAME)).show()
```

Linux Foundation Delta Lake

O Linux Foundation Delta Lake é um formato de tabela que pode ser usado para análise de big data. Use o Athena para Spark para ler tabelas do Delta Lake armazenadas diretamente no Amazon S3.

Para usar tabelas do Delta Lake no Athena para Spark, configure as propriedades do Spark a seguir. Essas propriedades são configuradas por padrão no console do Athena para Spark quando você escolhe Delta Lake como formato de tabela. Para obter as etapas, consulte Como editar os detalhes da sessão ou Como criar seu próprio caderno.

```
"spark.sql.catalog.spark_catalog" : "org.apache.spark.sql.delta.catalog.DeltaCatalog",
"spark.sql.extensions" : "io.delta.sql.DeltaSparkSessionExtension"
```

O procedimento a seguir mostra como usar uma tabela do Delta Lake em um caderno do Athena para Spark. Execute cada etapa em uma nova célula no caderno.

Para usar uma tabela do Delta Lake no Athena para Spark

1. Defina as constantes a serem usadas no caderno.

```
DB_NAME = "NEW_DB_NAME"
TABLE_NAME = "NEW_TABLE_NAME"
TABLE_S3_LOCATION = "s3://DOC-EXAMPLE-BUCKET"
```

2. Crie um DataFrame do Apache Spark.

```
columns = ["language","users_count"]
data = [("Golang", 3000)]
df = spark.createDataFrame(data, columns)
```

3. Crie um banco de dados.

```
spark.sql("CREATE DATABASE {} LOCATION '{}'".format(DB_NAME, TABLE_S3_LOCATION))
```

4. Crie uma tabela vazia do Delta Lake.

```
spark.sql("""
CREATE TABLE {}.{} (
  language string,
  users_count int
) USING DELTA
""".format(DB_NAME, TABLE_NAME))
```

5. Insira uma linha de dados na tabela.

```
spark.sql("""INSERT INTO {}.{} VALUES ('Golang',
 3000)""".format(DB_NAME, TABLE_NAME))
```

6. Confirme se é possível consultar a nova tabela.

```
spark.sql("SELECT * FROM {}.{}".format(DB_NAME, TABLE_NAME)).show()
```

Suporte à biblioteca Python no Amazon Athena para Apache Spark

Esta página descreve a terminologia usada e o gerenciamento do ciclo de vida seguido para os tempos de execução, as bibliotecas e os pacotes usados no Amazon Athena para Apache Spark.

Definições

- O Amazon Athena para Apache Spark corresponde a uma versão personalizada do Apache Spark de código aberto. Para ver a versão atual, execute o comando `print(f'{spark.version}')` em uma célula de caderno.
- O runtime do Athena corresponde ao ambiente no qual o código é executado. O ambiente inclui um intérprete do Python e bibliotecas PySpark.
- Uma biblioteca ou um pacote externo corresponde a uma biblioteca Java, Scala JAR ou Python que não faz parte do runtime do Athena, mas pode ser incluída nos trabalhos do Athena para Spark. Os pacotes externos podem ser criados por você ou pela Amazon.
- Um pacote de conveniência corresponde a uma coleção de pacotes externos selecionados pelo Athena que você pode optar por incluir em suas aplicações Spark.

- Um pacote combina o runtime do Athena e um pacote de conveniência.
- Uma biblioteca de usuário corresponde a uma biblioteca ou a um pacote externo que você adiciona explicitamente ao seu trabalho do Athena para Spark.
  - Uma biblioteca de usuário é um pacote externo que não faz parte de um pacote de conveniência. Uma biblioteca de usuário requer carregamento e instalação, como quando você grava alguns arquivos `.py`, compacta-os e, em seguida, adiciona o arquivo `.zip` à aplicação.
- Uma aplicação do Athena para Spark corresponde a um trabalho ou a uma consulta que você envia ao Athena para Spark.

Gerenciamento de ciclo de vida

Versionamento e substituição do runtime

O componente principal no runtime do Athena é o intérprete do Python. Como o Python é uma linguagem em evolução, novas versões são lançadas com regularidade e o suporte é removido para as versões mais antigas. O Athena não recomenda que você execute programas com versões obsoletas do intérprete do Python e recomenda fortemente que você use o runtime mais recente do Athena sempre que possível.

O cronograma de substituição do runtime do Athena é o seguinte:

1. Depois que o Athena fornecer um novo runtime, ele continuará a oferecer suporte ao runtime anterior por seis meses. Durante esse período, o Athena aplicará patches e atualizações de segurança ao runtime anterior.
2. Após seis meses, o Athena encerrará o suporte para o runtime anterior. O Athena não aplicará mais patches de segurança e outras atualizações ao runtime anterior. As aplicações Spark que usam o runtime anterior não serão mais elegíveis para o suporte técnico.
3. Após 12 meses, não será mais possível atualizar ou editar aplicações Spark em um grupo de trabalho que usa o runtime anterior. Recomendamos que você atualize suas aplicações Spark antes que esse período se encerre. Após o término do período, você ainda poderá executar os cadernos existentes, mas todos os cadernos que ainda usarem o runtime anterior registrarão em log um aviso nesse sentido.
4. Após 18 meses, você não poderá mais executar trabalhos no grupo de trabalho usando o runtime anterior.

Versionamento e substituição do pacote de conveniência

O conteúdo dos pacotes de conveniência é alterado ao longo do tempo. Ocasionalmente, o Athena adiciona, remove ou atualiza os pacotes de conveniência.

O Athena usa as diretrizes a seguir para os pacotes de conveniência:

- Os pacotes de conveniência têm um esquema de versionamento simples, como 1, 2 e 3.
- Cada versão de pacote de conveniência inclui versões específicas de pacotes externos. Depois que o Athena cria um pacote de conveniência, o conjunto de pacotes externos do pacote de conveniência e suas versões correspondentes não são alterados.
- O Athena cria uma nova versão de pacote de conveniência ao incluir um novo pacote externo, remover um pacote externo ou atualizar a versão de um ou mais pacotes externos.

O Athena substitui um pacote de conveniência quando substitui o runtime do Athena que o pacote usa. O Athena pode substituir os pacotes antecipadamente para limitar o número de pacotes que ele oferece suporte.

O cronograma de substituição de pacote de conveniência segue o cronograma de substituição de runtime do Athena.

Lista de bibliotecas Python pré-instaladas

As bibliotecas Python pré-instaladas incluem o seguinte.

```
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
```

Observações

- Não há compatibilidade com MLLib (biblioteca de machine learning do Apache Spark) e com o pacote `pyspark.ml`.
- No momento, `pip install` não é compatível com as sessões do Athena para Spark.

Para obter informações sobre como importar bibliotecas Python para o Amazon Athena para Apache Spark, consulte Importação de arquivos e de bibliotecas Python para o Amazon Athena para Apache Spark.

Importação de arquivos e de bibliotecas Python para o Amazon Athena para Apache Spark

Este documento fornece exemplos de como importar arquivos e bibliotecas Python para o Amazon Athena para Apache Spark.

Condições e limitações

- Versão Python: atualmente, o Athena para Spark usa o Python versão 3.9.16. Observe que os pacotes Python são sensíveis às versões secundárias do Python.
- Arquitetura do Athena para Spark: o Athena para Spark usa o Amazon Linux 2 na arquitetura ARM64. Observe que algumas bibliotecas Python não distribuem binários para essa arquitetura.
- Objetos compartilhados binários (SOs): como o método addPyFile do SparkContext não detecta objetos binários compartilhados, ele não pode ser usado no Athena para Spark para adicionar pacotes Python que dependam de objetos compartilhados.
- Conjuntos de dados resilientes distribuídos (RDDs): RDDs não são compatíveis.
- Dataframe.foreach: o método DataFrame.foreach do PySpark não é compatível.

Exemplos

Os exemplos usam as convenções a seguir:

- O espaço reservado para `s3://DOC-EXAMPLE-BUCKET` no local do Amazon S3. Substitua isso pelo seu próprio local para buckets do S3.
- Todos os blocos de código executados a partir de um shell Unix são apresentados como *`directory_name`* `$`. Por exemplo, o comando `ls` no diretório `/tmp` e sua saída são exibidos da seguinte forma:

```
/tmp $ ls
```

Saída

```
file1 file2
```

- Como adicionar um arquivo a um caderno após gravá-lo no diretório temporário local
- Como importar um arquivo do Amazon S3
- Como adicionar arquivos Python e registrar uma UDF
- Importação de um arquivo .zip do Python
- Importação de duas versões de uma biblioteca Python como módulos separados
- Importação de um arquivo .zip do Python do PyPI

- Importação de um arquivo .zip do Python do PyPI que tem dependências

Importação de arquivos de texto para uso em cálculos

Os exemplos nesta seção mostram como importar arquivos de texto para uso em cálculos em seus cadernos no Athena para Spark.

Como adicionar um arquivo a um caderno após gravá-lo no diretório temporário local

O exemplo a seguir mostra como gravar um arquivo em um diretório temporário local, adicioná-lo a um caderno e testá-lo.

```
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()
```

Saída

```
Calculation completed.
+---+---+-------+
| _1| _2|    col|
+---+---+-------+
|  1|  a|[aaaaa]|
|  2|  b|[bbbbb]|
```

```
+---+---+-------+
```

Como importar um arquivo do Amazon S3

O exemplo a seguir mostra como importar um arquivo do Amazon S3 para um caderno e testá-lo.

Para importar um arquivo do Amazon S3 para um caderno

1. Crie um arquivo denominado `test.txt` que tenha uma única linha com o valor `5`.
2. Adicione o arquivo a um bucket no Amazon S3. Este exemplo usa o local `s3://DOC-EXAMPLE-BUCKET`.
3. Use o código a seguir para importar o arquivo para seu caderno e realizar o teste do arquivo.

```
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()
```

Saída

```
Calculation completed.
+---+---+-------+
| _1| _2|    col|
+---+---+-------+
|  1|  a|[aaaaa]|
|  2|  b|[bbbbb]|
+---+---+-------+
```

Adição de arquivos Python

Os exemplos nesta seção mostram como adicionar arquivos e bibliotecas Python aos seus cadernos Spark no Athena.

Como adicionar arquivos Python e registrar uma UDF

O exemplo a seguir mostra como adicionar arquivos Python do Amazon S3 ao seu caderno e registrar uma UDF.

Para adicionar arquivos Python ao seu caderno e registrar uma UDF

1. Usando seu próprio local do Amazon S3, crie o arquivo `s3://DOC-EXAMPLE-BUCKET/file1.py` com o seguinte conteúdo:

```
def xyz(input):
    return 'xyz  - udf ' + str(input);
```

2. No mesmo local do S3, crie o arquivo `s3://DOC-EXAMPLE-BUCKET/file2.py` com o seguinte conteúdo:

```
from file1 import xyz
def uvw(input):
    return 'uvw -> ' + xyz(input);
```

3. Em seu caderno do Athena para Spark, execute os comandos a seguir.

```
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)
```

Saída

```
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... |
+---+---+--------------------+
```

Importação de um arquivo .zip do Python

É possível usar os métodos `addPyFile` e `import` do Python para importar um arquivo .zip do Python para o seu caderno.

> **Note**
>
> Os arquivos `.zip` que você importa para o Athena Spark podem incluir somente pacotes Python. Por exemplo, a inclusão de pacotes com arquivos baseados em C não é compatível.

Para importar um arquivo **`.zip`** do Python para seu caderno

1. Em seu computador local, em um diretório da área de trabalho, como `\tmp`, crie um diretório denominado `moduletest`.
2. No diretório `moduletest`, crie um arquivo denominado `hello.py` com o conteúdo a seguir:

   ```
   def hi(input):
       return 'hi ' + str(input);
   ```

3. No mesmo diretório, adicione um arquivo vazio com o nome `__init__.py`.

   Se você listar o conteúdo do diretório, ele deverá se parecer com o seguinte:

   ```
   /tmp $ ls moduletest
   __init__.py       hello.py
   ```

4. Use o comando `zip` para inserir os dois arquivos do módulo em um arquivo denominado `moduletest.zip`.

```
moduletest $ zip -r9 ../moduletest.zip *
```

5. Carregue o arquivo `.zip` em seu bucket no Amazon S3.
6. Use o código a seguir para importar o arquivo `.zip` do Python para seu caderno.

```
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()
```

Saída

```
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|
+---+---+----+
```

Importação de duas versões de uma biblioteca Python como módulos separados

Os exemplos de código a seguir mostram como adicionar e importar duas versões diferentes de uma biblioteca Python de um local no Amazon S3 como dois módulos separados. O código adicionará cada arquivo da biblioteca do S3, realizará a importação deles e, em seguida, imprimirá a versão da biblioteca para verificar a importação.

```
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__)
```

Saída

```
3.15.0
```

```
import simplejson_v3_17_6
print(simplejson_v3_17_6.__version__)
```

Saída

```
3.17.6
```

Importação de um arquivo .zip do Python do PyPI

Este exemplo usa o comando `pip` para baixar de um arquivo .zip do Python referente ao projeto bpabel/piglatin do Python Package Index (PyPI).

Para importar um arquivo .zip do Python do PyPI

1. Em sua área de trabalho local, use os comandos a seguir para criar um diretório denominado `testpiglatin` e criar um ambiente virtual.

   ```
   /tmp $ mkdir testpiglatin
   /tmp $ cd testpiglatin
   testpiglatin $ virtualenv .
   ```

   Saída

   ```
   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. Crie um subdiretório chamado `unpacked` para manter o projeto.

```
testpiglatin $ mkdir unpacked
```

3. Use o comando `pip` para instalar o projeto no diretório `unpacked`.

```
testpiglatin $ bin/pip install -t $PWD/unpacked piglatin
```

Saída

```
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. Verifique o conteúdo do diretório.

```
testpiglatin $ ls
```

Saída

```
bin lib pyvenv.cfg unpacked
```

5. Altere para o diretório `unpacked` e exiba seu conteúdo.

```
testpiglatin $ cd unpacked
unpacked $ ls
```

Saída

```
piglatin piglatin-1.0.6.dist-info
```

6. Use o comando `zip` para inserir o conteúdo do projeto piglatin em um arquivo denominado `library.zip`.

```
unpacked $ zip -r9 ../library.zip *
```

Saída

```
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. (Opcional) Use os seguintes comandos para testar a importação em nível local.

   a. Defina o caminho do Python para o local do arquivo `library.zip` e inicie o Python.

   ```
   /home $ PYTHONPATH=/tmp/testpiglatin/library.zip
   /home $ python3
   ```

   Saída

   ```
   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. Importe a biblioteca e execute um comando para teste.

   ```
   >>> import piglatin
   >>> piglatin.translate('hello')
   ```

   Saída

   ```
   'ello-hay'
   ```

8. Use comandos, como apresentado a seguir, para adicionar o arquivo `.zip` do Amazon S3, importá-lo para seu caderno no Athena e testá-lo.

```
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()
```

Saída

```
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|
+---+-----+--------+
```

Importação de um arquivo .zip do Python do PyPI que tem dependências

Este exemplo realiza a importação do pacote md2gemini, que converte texto em markdown para o formato de texto Gemini do PyPI. O pacote tem as seguintes dependências:

```
cjkwrap
mistune
wcwidth
```

Para importar um arquivo .zip do Python que tem dependências

1. Em seu computador local, use os comandos a seguir para criar um diretório denominado `testmd2gemini` e criar um ambiente virtual.

```
/tmp $ mkdir testmd2gemini
/tmp $ cd testmd2gemini
```

```
testmd2gemini$ virtualenv .
```

2. Crie um subdiretório chamado `unpacked` para manter o projeto.

```
testmd2gemini $ mkdir unpacked
```

3. Use o comando `pip` para instalar o projeto no diretório `unpacked`.

```
/testmd2gemini $ bin/pip install -t $PWD/unpacked md2gemini
```

Saída

```
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. Altere para o diretório `unpacked` e verifique seu conteúdo.

```
testmd2gemini $ cd unpacked
unpacked $ ls -lah
```

Saída

```
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. Use o comando `zip` para inserir o conteúdo do projeto md2gemini em um arquivo denominado `md2gemini.zip`.

```
unpacked $ zip -r9 ../md2gemini *
```

Saída

```
  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. (Opcional) Use os comandos a seguir para testar se a biblioteca funciona em seu computador local.

   a. Defina o caminho do Python para o local do arquivo `md2gemini.zip` e inicie o Python.

```
/home $ PYTHONPATH=/tmp/testmd2gemini/md2gemini.zip
/home python3
```

   b. Importe a biblioteca e execute um teste.

```
>>> from md2gemini import md2gemini
>>> print(md2gemini('[abc](https://abc.def)'))
```

Saída

```
https://abc.def abc
```

7. Use os comandos a seguir para adicionar o arquivo `.zip` do Amazon S3, importá-lo para seu caderno no Athena e executar um teste que não seja UDF.

```
# (non udf test)
sc.addPyFile('s3://DOC-EXAMPLE-BUCKET/md2gemini.zip')
from md2gemini import md2gemini
```

```
print(md2gemini('[abc](https://abc.def)'))
```

Saída

```
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. Use os comandos a seguir para executar um teste que seja 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()
```

Saída

```
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...|
+---+--------------------+--------------------+
```

Adicionar arquivos JAR e configuração personalizada do Spark

Ao criar ou editar uma sessão no Amazon Athena para Apache Spark, você pode usar as propriedades do Spark para especificar arquivos `.jar`, pacotes ou outra configuração

personalizada para a sessão. Para especificar propriedades do Spark, use o console do Athena, a AWS CLI ou a API do Athena.

Usar o console do Athena para especificar propriedades do Spark

No console do Athena, é possível especificar as propriedades do Spark ao criar um caderno ou editar uma sessão atual.

Para adicionar propriedades na caixa de diálogo Criar caderno ou Editar detalhes da sessão

1. Expanda as propriedades do Spark.
2. Para adicionar propriedades, use a opção Editar na tabela ou Editar em JSON.
   - Para a opção Editar na tabela, escolha Adicionar propriedade para adicionar uma propriedade, ou escolha Remover para remover uma propriedade. Use as caixas Chave e Valor para inserir os nomes das propriedades e os respectivos valores.
     - Para adicionar um arquivo `.jar` personalizado, use a propriedade `spark.jars`.
     - Para especificar um arquivo de pacote, especifique a propriedade `spark.jars.packages`.
   - Para inserir e editar sua configuração diretamente, escolha a opção Editar em JSON. No editor de texto JSON, você pode executar as seguintes tarefas:
     - Escolha Copiar para copiar o texto JSON para a área de transferência.
     - Escolha Limpar para remover todo o texto do editor JSON.
     - Escolha o ícone de configurações (engrenagem) para configurar a quebra de linha ou escolha um tema de cores para o editor JSON.

Observações

- É possível definir propriedades no Athena para Spark, que é o mesmo que definir as propriedades do Spark diretamente em um objeto SparkConf.
- Inicie todas as propriedades do Spark com o prefixo `spark.`. As propriedades com outros prefixos são ignoradas.
- Nem todas as propriedades do Spark estão disponíveis para configuração personalizada no Athena. Se você enviar uma solicitação `StartSession` com uma configuração restrita, a sessão não será iniciada.
  - Não é possível usar o prefixo `spark.athena.` porque ele é reservado.

Usar a AWS CLI ou API do Athena para fornecer uma configuração personalizada

Para usar a AWS CLI ou a API do Athena para fornecer sua configuração de sessão, use a ação de API StartSession ou o comando start-session da CLI. Em sua solicitação `StartSession`, use o campo `SparkProperties` do objeto EngineConfiguration para passar suas informações de configuração no formato JSON. Isso iniciará uma sessão com a configuração especificada. Para obter a sintaxe da solicitação, consulte StartSession na Amazon Athena API Reference.

Solução de erros de início de sessão

Quando ocorre um erro de configuração personalizada durante o início da sessão, o console do Athena para Spark mostra um banner de mensagem de erro. Para solucionar erros de início de sessão, verifique a alteração do estado da sessão ou as informações de registro.

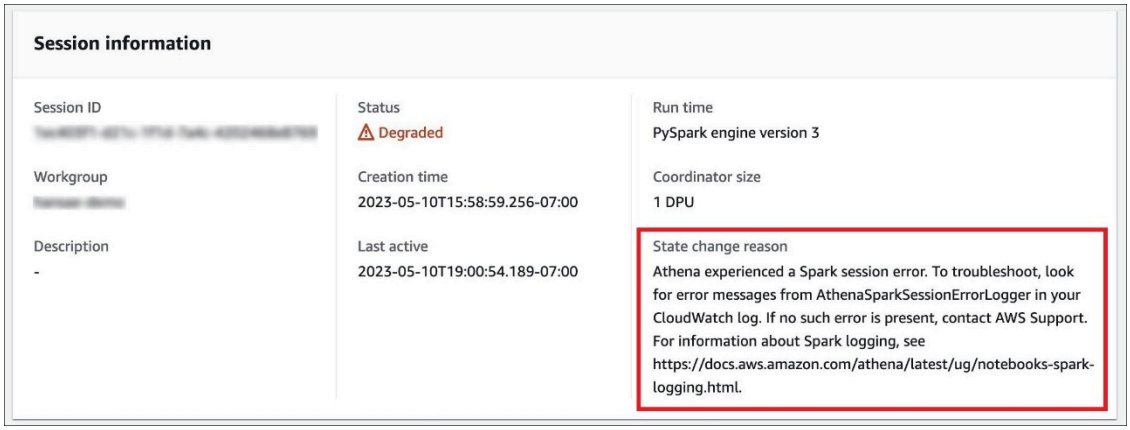
Visualizar informações de alteração do estado da sessão

Você pode obter detalhes sobre uma alteração no estado da sessão no editor do caderno do Athena ou na API do Athena.

Para visualizar as informações do estado da sessão no console do Athena

1. No editor de cadernos do Athena, no menu Sessão no canto superior direito, escolha Visualizar detalhes.
2. Visualize a guia Sessão atual. A seção Informações da sessão exibe informações como ID da sessão, grupo de trabalho, status e motivo da mudança de estado.

   O exemplo de captura de tela a seguir mostra informações na seção Motivo da mudança de estado da caixa de diálogo Informações da sessão para um erro de sessão do Spark no Athena.



Para visualizar as informações do estado da sessão usando a API do Athena

- Na API do Athena, você pode encontrar informações sobre alteração do estado da sessão no campo `StateChangeReason` do objeto SessionStatus.

> **Note**
>
> Após interromper manualmente uma sessão ou se a sessão for interrompida após um tempo limite de inatividade (o padrão é 20 minutos), o valor de StateChangeReason é alterado para A sessão foi encerrada por solicitação.

Usar registro em log para solucionar erros de início de sessão

Os erros de configuração personalizada que ocorrem durante o início da sessão são registrados em log pelo Amazon CloudWatch. No CloudWatch Logs, pesquise mensagens de erro de `AthenaSparkSessionErrorLogger` para solucionar uma falha no início da sessão.

Para obter mais informações sobre registro em log do Spark, consulte Registro em log de eventos da aplicação Spark no Athena.

Para obter mais informações sobre sessões de solução de problemas no Athena para Spark, consulte Solução de problemas de sessões.

Formatos de dados e de armazenamento compatíveis

A tabela a seguir apresenta os formatos com suporte nativo para Apache Spark no Athena.

Formato de dados	Read	Write	Compressão de gravação
parquet	sim	sim	nenhum, descompactado, snappy, gzip
orc	sim	sim	nenhum, snappy, zlib, lzo
json	sim	sim	bzip2, gzip, desinflar

Formato de dados	Read	Write	Compressão de gravação
csv	sim	sim	bzip2, gzip, desinflar
text	sim	sim	nenhum, bzip2, gzip, desinflar
arquivos binários	sim	N/D	N/D

Monitoramento de cálculos do Apache Spark com métricas do CloudWatch

O Athena publica métricas relacionadas a cálculos no Amazon CloudWatch quando a opção Publish CloudWatch metrics para seu grupo de trabalho habilitado para Spark está selecionada. É possível criar painéis personalizados, definir alarmes e acionar métricas no console do CloudWatch.

O Athena publica a seguinte métrica no console do CloudWatch sob o namespace `AmazonAthenaForApacheSpark`:

- `DPUCount`: a quantidade de DPUs consumidas durante a sessão para executar os cálculos.

Essa métrica tem as seguintes dimensões:

- `SessionId`: o ID da sessão para a qual os cálculos são enviados.
- `WorkGroup`: o nome do grupo de trabalho.

Para visualizar as métricas para grupos de trabalho habilitados para Spark no console do Amazon CloudWatch

1. Abra o console CloudWatch em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, escolha Metrics (Métricas), All metrics (Todas as métricas).
3. Selecione o namespace AmazonAthenaForApacheSpark.

Para exibir métricas com a CLI

- Execute um destes procedimentos:
  - Para listar as métricas dos grupos de trabalho habilitados para Spark do Athena, abra uma solicitação de comando e use o comando a seguir:

```
aws cloudwatch list-metrics --namespace "AmazonAthenaForApacheSpark"
```

  - Para listar todas as métricas disponíveis, use o comando a seguir:

```
aws cloudwatch list-metrics
```

Lista de métricas e dimensões do CloudWatch para cálculos do Apache Spark no Athena

Se você habilitou as métricas do CloudWatch em seu grupo de trabalho habilitado para Spark do Athena, o Athena enviará a métrica a seguir para o CloudWatch por grupo de trabalho. A métrica usa o namespace `AmazonAthenaForApacheSpark`.

Nome da métrica	Descrição
DPUCount	A quantidade de DPUs (unidades de processamento de dados) consumidas durante a sessão para executar os cálculos. Uma DPU é uma medida relativa do poder de processamento que consiste em uma capacidade computacional de 4 vCPUs e 16 GB de memória.

Essa métrica tem as seguintes dimensões.

Dimensão	Descrição
SessionId	O ID da sessão para a qual os cálculos são enviados.
WorkGroup	O nome do grupo de trabalho.

Habilita buckets de pagamentos pelo solicitante do Amazon S3 no Athena para Spark

Quando um bucket do Amazon S3 é configurado como pagamento pelo solicitante, a conta do usuário que executa a consulta é cobrada pelas taxas de acesso e transferência de dados associadas à consulta. Para obter mais informações, consulte Configuração de buckets de Pagamento pelo solicitante para transferências de armazenamento e uso no Guia do usuário do Amazon S3.

No Athena para Spark, os buckets de pagamentos pelo solicitante são habilitados por sessão, não por grupo de trabalho. Em um nível alto, habilitar os buckets de pagamentos pelo solicitante inclui as seguintes etapas:

1. No console do Amazon S3, habilite os pagamentos pelo solicitante nas propriedades do bucket e adicione uma política de bucket para especificar o acesso.
2. No console do IAM, crie uma política do IAM para permitir o acesso ao bucket e, em seguida, anexe a política ao perfil do IAM que será usado para acessar o bucket de pagamentos pelo solicitante.
3. No Athena para Spark, adicione uma propriedade de sessão para habilitar o recurso de pagamento pelo solicitante.

1. Habilitar pagamentos pelo solicitante em um bucket do Amazon S3 e adicionar uma política de bucket

Habilitar o pagamento pelo solicitante para um bucket do Amazon S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Na lista de buckets, escolha o link do bucket para o qual você deseja habilitar o pagamento pelo solicitante.
3. Na página do bucket, escolha a aba Propriedades.
4. Role para baixo até a seção Pagamentos pelo solicitante e depois escolha Editar.
5. Na página Editar pagamentos pelo solicitante, escolha Habilitar e, em seguida, escolha Salvar alterações.
6. Escolha a aba Permissions (permissões).
7. Na seção Política de bucket, escolha Editar.

8. Na página Editar política de bucket, aplique a política de bucket que você deseja ao bucket de origem. O exemplo de política a seguir dá acesso a todas as entidades principais da AWS (`"AWS": "*"`), mas seu acesso pode ser mais granular. Por exemplo, talvez você queira especificar somente um perfil do IAM específico em outra conta.

```
{
    "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. Criar uma política do IAM e anexá-la ao perfil do IAM

Depois, você cria uma política do IAM para permitir acesso ao bucket. Em seguida, você anexa a política ao perfil que será usado para acessar o bucket de pagamentos pelo solicitante.

Criar uma política do IAM para o bucket de pagamentos pelo solicitante e anexá-la a um perfil

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação do console IAM, escolha Políticas.
3. Escolha Criar política.
4. Selecione JSON.
5. No Editor de política, adicione uma política como a seguinte:

```
{
```

```
    "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. Escolha Next (próximo).
7. Na página Revisar e criar, digite um nome e uma descrição opcional para a política e, em seguida, escolha Criar política.
8. No painel de navegação, escolha Perfis.
9. Na página Perfis, localize o perfil que você deseja usar e escolha o link do nome do perfil.
10. Na seção Políticas de permissões, escolha Adicionar permissões, Anexar políticas.
11. Na seção Outras políticas de permissões, selecione a caixa de seleção da política que você criou e escolha Adicionar permissões.

3. Adicionar uma propriedade de sessão do Athena para Spark

Depois de configurar o bucket do Amazon S3 e as permissões associadas para pagamentos pelo solicitante, você pode habilitar o atributo em uma sessão do Athena para Spark.

Habilitar buckets de pagamentos pelo solicitante em uma sessão do Athena para Spark

1. No editor do notebook, no menu Session (sessão) no canto superior direito, escolha Edit session (editar sessão).
2. Expanda as propriedades do Spark.
3. Escolha Editar em JSON.
4. No editor de texto JSON, insira o seguinte:

```
{
  "spark.hadoop.fs.s3.useRequesterPaysHeader":"true"
}
```

5. Escolha Salvar.

Habilitar a criptografia do Apache Spark

Você pode habilitar a criptografia do Apache Spark no Athena. Essa ação criptografa dados em trânsito entre nós do Spark e criptografa dados em repouso armazenados localmente pelo Spark. Para aumentar a segurança desses dados, o Athena usa esta configuração de criptografia:

```
spark.io.encryption.keySizeBits="256"
spark.io.encryption.keygen.algorithm="HmacSHA384"
```

Para habilitar a criptografia do Spark, você pode usar o console do Athena, a AWS CLI ou a API do Athena.

Usar o console do Athena para habilitar a criptografia do Spark

Para criar um novo caderno que tenha a criptografia do Spark habilitada

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.
3. Execute um destes procedimentos:
   - No Notebook explorer (Explorador de cadernos), escolha Create notebook (Criar caderno).
   - No Notebook editor (Editor de cadernos), escolha Create notebook (Criar caderno) ou selecione o ícone de adição (+) para adicionar um caderno.
4. Em Nome do caderno, insira um nome para o caderno.
5. Expanda a opção Propriedades do Spark.
6. Selecione Ativar a criptografia do Spark.
7. Escolha Criar.

A sessão do caderno que você cria é criptografada. Use o novo caderno como faria normalmente. Futuramente, quando você iniciar novas sessões que usarem o caderno, as novas sessões também serão criptografadas.

Você também pode usar o console do Athena para ativar a criptografia do Spark para um caderno existente.

Para habilitar a criptografia para um caderno existente

1. Abra uma nova sessão para um caderno criado anteriormente.
2. No editor do notebook, no menu Session (sessão) no canto superior direito, escolha Edit session (editar sessão).
3. Na caixa de diálogo Editar detalhes da sessão, expanda Propriedades do Spark.
4. Selecione Ativar a criptografia do Spark.
5. Escolha Salvar.

O console inicia uma nova sessão que tem a criptografia habilitada. As sessões posteriores que você criar para o caderno também terão a criptografia habilitada.

Usar a AWS CLI para habilitar a criptografia do Spark

Você pode usar a AWS CLI para habilitar a criptografia ao iniciar uma sessão especificando as propriedades corretas do Spark.

Para usar a AWS CLI para habilitar a criptografia do Spark

1. Use um comando como o exemplo a seguir para criar um objeto JSON de configuração do mecanismo que especifique as propriedades de criptografia do 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. Na AWS CLI, use o comando `athena start-session` e passe o objeto JSON que você criou para o argumento `--engine-configuration`, como no seguinte exemplo:

```
aws athena start-session \
   --region "region" \
   --work-group "your-work-group" \
   --engine-configuration "$ENGINE_CONFIGURATION_JSON"
```

Usar a API do Athena para habilitar a criptografia do Spark

Para habilitar a criptografia do Spark com a API do Athena, use a ação StartSession e o respectivo parâmetro EngineConfiguration de `SparkProperties` para especificar a configuração de criptografia na solicitação `StartSession`.

Como configurar o acesso entre contas do AWS Glue no Athena para Spark

Este tópico mostra como a conta do consumidor *666666666666* e a conta do proprietário *999999999999* podem ser configuradas para acesso entre contas do AWS Glue. Quando as contas são configuradas, a conta do consumidor pode executar consultas do Athena para Spark nos bancos de dados e tabelas do AWS Glue do proprietário.

1. No AWS Glue, forneça acesso aos perfis do consumidor

No AWS Glue, o proprietário cria uma política que fornece aos perfis do consumidor acesso ao catálogo de dados do AWS Glue do proprietário.

Adicionar uma política do AWS Glue que permita que um perfil de consumidor acesse o catálogo de dados do proprietário

1. Usando a conta do proprietário do catálogo, faça login no AWS Management Console.
2. Abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
3. No painel de navegação, expanda Data Catalog e escolha Configurações do catálogo.

4. Na página Configurações do catálogo de dados, na seção Permissões, adicione uma política como a seguir. Essa política fornece perfis para a conta do consumidor *666666666666* acessar o catálogo de dados na conta do proprietário *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. Configurar a conta do consumidor para acesso

Na conta do consumidor, crie uma política para permitir o acesso ao AWS Glue Data Catalog, aos bancos de dados e às tabelas do proprietário e anexe a política a um perfil. O exemplo a seguir usa a conta de consumidor *666666666666*.

Criar uma política do AWS Glue para acesso ao AWS Glue Data Catalog do proprietário

1. Usando a conta do consumidor, faça login no AWS Management Console.
2. Abra o console IAM em https://console.aws.amazon.com/iam/.
3. No painel de navegação, expanda Gerenciamento de acesso e escolha Políticas.
4. Escolha Criar política.
5. Na página Especificar permissões, escolha JSON.

6. No Editor de políticas, insira uma instrução JSON como a seguir que permite ações do AWS Glue no catálogo de dados da conta do proprietário.

```
{
    "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. Escolha Próximo.
8. Na página Revisar e criar, para Nome da política, insira um nome para a política.
9. Escolha Criar política.

Em seguida, você usa o console do IAM na conta do consumidor para anexar a política que você acabou de criar ao perfil ou aos perfis do IAM que a conta do consumidor usará para acessar o catálogo de dados do proprietário.

Anexar a política do AWS Glue aos perfis na conta do consumidor

1. No painel de navegação do console do IAM da conta do consumidor, escolha Perfis.
2. Na página Perfis, encontre o perfil ao qual você deseja anexar a política.
3. Escolha Adicionar permissões e depois Anexar políticas.
4. Encontre a política que você acabou de criar.
5. Marque a caixa de seleção da política e escolha Adicionar permissões.
6. Repita as etapas para adicionar a política a outros perfis que você deseja usar.

3. Configurar uma sessão e criar uma consulta

No Athena Spark, na conta do solicitante, usando o perfil especificado, crie uma sessão para testar o acesso criando um caderno ou editando uma sessão atual. Ao configurar as propriedades da sessão, especifique uma das seguintes opções:

- O separador de catálogo do Glue: com essa abordagem, você inclui o ID da conta do proprietário em suas consultas. Use esse método se você for usar a sessão para consultar catálogos de dados de diferentes proprietários.
- O ID do catálogo do Glue: com essa abordagem, você consulta diretamente o banco de dados. Esse método será mais conveniente se você for usar a sessão para consultar o catálogo de dados de somente um proprietário.

Usar a abordagem do separador de catálogos do AWS Glue

Ao editar as propriedades da sessão, adicione o seguinte:

```
{
    "spark.hadoop.aws.glue.catalog.separator": "/"
}
```

Ao executar uma consulta em uma célula, use uma sintaxe como a do exemplo a seguir. Observe que na cláusula `FROM`, o separador e o ID do catálogo e são necessários antes do nome do banco de dados.

```
df = spark.sql('SELECT requestip, uri, method, status FROM `999999999999/
mydatabase`.cloudfront_logs LIMIT 5')
df.show()
```

Usando a abordagem de ID de catálogo do AWS Glue

Ao editar as propriedades da sessão, insira a propriedade a seguir. Substitua *999999999999* pelo ID da conta do proprietário.

```
{
    "spark.hadoop.hive.metastore.glue.catalogid": "999999999999"
}
```

Ao executar uma consulta em uma célula, use uma sintaxe como a seguir. Observe que na cláusula `FROM`, o separador e ID do catálogo não são necessários antes do nome do banco de dados.

```
df = spark.sql('SELECT * FROM mydatabase.cloudfront_logs LIMIT 10')
df.show()
```

Recursos adicionais do

Acesso aos catálogos de dados do AWS Glue entre contas

Managing cross-account permissions using both AWS Glue and Lake Formation no Guia do desenvolvedor do AWS Lake Formation.

Configure o acesso entre contas para um AWS Glue Data Catalog compartilhado usando o Amazon Athena nos Padrões de orientação prescritiva da AWS.

Cotas de serviço do Amazon Athena para Apache Spark

As service quotas, também conhecidas como limites, correspondem ao número máximo de recursos ou operações de serviço que sua Conta da AWS pode usar. Para obter mais informações sobre as cotas de serviço para outros serviços AWS que você pode usar com o Amazon Athena para Spark, consulte Cotas de serviço AWS no Referência geral da Amazon Web Services.

> **Note**
>
> Novas Contas da AWS podem ter cotas iniciais mais baixas que podem ser aumentadas no decorrer do tempo. O Amazon Athena para Apache Spark monitora constantemente o uso em cada Região da AWS e aumenta automaticamente as cotas com base no uso. Se seus requisitos excederem os limites estabelecidos, entre em contato com o suporte ao cliente.

A tabela a seguir lista as cotas de serviço do Amazon Athena para o Apache Spark.

Nome	Padrão	Ajustável	Descrição
Concorrência de DPU do Apache Spark	160	Não	O número máximo de unidades de processamento de dados (DPUs) que você pode consumir

Nome	Padrão	Ajustável	Descrição
			simultaneamente para cálculos do Apache Spark para uma única conta na Região da AWS atual. Uma DPU é uma medida relativa do poder de processamento que consiste em uma capacidade computacional de 4 vCPUs e 16 GB de memória.
Simultaneidade de DPU da sessão Apache Spark	60	Não	O número máximo de DPUs que você pode consumir simultaneamente para um cálculo do Apache Spark em uma sessão.

APIs para cadernos do Athena

A lista a seguir contém links de referência para as ações de API disponíveis para cadernos do Athena. Para obter estruturas de dados e outras ações de API do Athena, consulte Amazon Athena API Reference (Referência para APIs do 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

Problemas conhecidos no Athena para Spark

Essa página documenta alguns dos problemas conhecidos no Athena para Apache Spark.

Exceção para argumento inválido ao criar uma tabela

Embora o Spark não permita que bancos de dados sejam criados com uma propriedade de local vazia, os bancos de dados no AWS Glue podem ter uma propriedade `LOCATION` vazia se forem criados de forma externa ao Spark.

Se você criar uma tabela e especificar um banco de dados do AWS Glue com um campo `LOCATION` vazio, poderá ocorrer uma exceção como a seguinte: IllegalArgumentException: Cannot create a path from an empty string. (IllegalArgumentException: não é possível criar um caminho usando uma string vazia).

Por exemplo, o comando a seguir gera uma exceção se o banco de dados padrão no AWS Glue contiver um campo `LOCATION` vazio:

```
spark.sql("create table testTable (firstName STRING)")
```

Solução sugerida A: use o AWS Glue para adicionar um local ao banco de dados que você está usando.

Para adicionar um local a um banco de dados do AWS Glue

1. Faça login no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. No painel de navegação, escolha Bancos de dados.

3. Na lista de bancos de dados, selecione o banco de dados que deseja editar.
4. Na página de detalhes do banco de dados, escolha Edit (Editar).
5. Na página Update a database (Atualizar um banco de dados), em Location (Local), insira um local do Amazon S3.
6. Escolha Update Database (Atualizar banco de dados).

Solução sugerida B: use um banco de dados diferente do AWS Glue que tenha um local válido e existente no Amazon S3. Por exemplo, se você tiver um banco de dados denominado `dbWithLocation`, use o comando `spark.sql("use dbWithLocation")` para alternar para esse banco de dados.

Solução sugerida C: ao usar o Spark SQL para criar a tabela, especifique um valor para `location`, como no exemplo a seguir.

```
spark.sql("create table testTable (firstName STRING)
       location 's3://DOC-EXAMPLE-BUCKET/'").
```

Solução sugerida D: se você especificou um local ao criar a tabela, mas o problema persistir, certifique-se de que o caminho do Amazon S3 fornecido tenha uma barra no final. Por exemplo, o comando a seguir gera uma exceção para o argumento inválido:

```
spark.sql("create table testTable (firstName STRING)
       location 's3://DOC-EXAMPLE-BUCKET'")
```

Para corrigir isso, adicione uma barra no final para o local (por exemplo, `'s3:// DOC-EXAMPLE-BUCKET/'`).

Banco de dados criado em um local do grupo de trabalho

Se você usar um comando como `spark.sql('create database db')` para criar um banco de dados e não especificar um local para ele, o Athena criará um subdiretório no local do seu grupo de trabalho e usará esse local para o banco de dados recém-criado.

Problemas com tabelas gerenciadas pelo Hive no banco de dados do AWS Glue padrão

Se a propriedade `Location` de seu banco de dados padrão no AWS Glue não estiver vazia e especificar um local válido no Amazon S3, e você usar o Athena para Spark para criar uma tabela

gerenciada pelo Hive em seu banco de dados do AWS Glue padrão, os dados serão gravados no local do Amazon S3 especificado no grupo de trabalho do Athena Spark, e não no local especificado pelo banco de dados do AWS Glue.

Esse problema ocorre devido à forma como o Apache Hive manipula o banco de dados padrão. O Apache Hive cria dados de tabela na localização raiz do warehouse Hive, que pode ser diferente da localização real do banco de dados padrão.

Quando você usa o Athena para Spark para criar uma tabela gerenciada pelo Hive no banco de dados padrão no AWS Glue, os metadados da tabela do AWS Glue podem apontar para dois locais diferentes. Isso pode causar um comportamento inesperado ao tentar uma operação `INSERT` ou `DROP TABLE`.

Estão são as etapas para reproduzir o problema:

1. No Athena para Spark, você usa um dos seguintes métodos para criar ou salvar uma tabela gerenciada pelo Hive:
   - Uma instrução SQL como `CREATE TABLE $tableName`
   - Um comando PySpark como `df.write.mode("overwrite").saveAsTable($tableName)` que não especifica a opção `path` na API Dataframe.

   Nesse momento, o console do AWS Glue pode mostrar uma localização incorreta no Amazon S3 para a tabela.
2. No Athena para Spark, use a instrução `DROP TABLE $table_name` para eliminar a tabela que você criou.
3. Após executar a instrução `DROP TABLE`, você percebe que os arquivos subjacentes no Amazon S3 ainda estão presentes.

Para resolver esse problema, execute um dos seguintes procedimentos:

Solução A: use um banco de dados do AWS Glue diferente ao criar tabelas gerenciadas pelo Hive.

Solução B: especifique um local vazio para o banco de dados padrão no AWS Glue. Em seguida, crie as tabelas gerenciadas no banco de dados padrão.

Incompatibilidade de formatos de arquivo CSV e JSON entre o Athena para Spark e o Athena SQL

Devido a um problema conhecido com o Spark de código aberto, ao criar uma tabela no Athena para Spark em dados CSV ou JSON, a tabela pode não ser legível no Athena SQL e vice-versa.

Por exemplo, é possível criar uma tabela no Athena para Spark das seguintes maneiras:

- Com esta seguinte sintaxe `USING csv`:

```
spark.sql('''CREATE EXTERNAL TABLE $tableName (
$colName1 $colType1,
$colName2 $colType2,
$colName3 $colType3)
USING csv
PARTITIONED BY ($colName1)
LOCATION $s3_location''')
```

- Com a seguinte sintaxe da API DataFrame:

```
df.write.format('csv').saveAsTable($table_name)
```

Devido ao problema conhecido com o Spark de código aberto, as consultas do Athena SQL nas tabelas resultantes podem não ser bem-sucedidas.

Solução sugerida: tente criar a tabela no Athena para Spark usando a sintaxe do Apache Hive. Para obter mais informações, consulte CREATE HIVEFORMAT TABLE na documentação do Apache Spark.

Solução de problemas do Athena para Spark

Use as informações a seguir para solucionar problemas que você pode ter ao usar cadernos e sessões no Athena.

Tópicos

- Solução de problemas de grupos de trabalho habilitados para Spark
- Uso da instrução EXPLAIN do Spark para solucionar problemas do Spark SQL
- Registro em log de eventos da aplicação Spark no Athena

- Uso do CloudTrail para solucionar problemas de chamadas de API de cadernos do Athena
- Como superar o limite de tamanho do bloco de código de 68 mil
- Solução de problemas de sessões
- Solução de problemas de tabelas
- Obter suporte

Solução de problemas de grupos de trabalho habilitados para Spark

Use as informações apresentadas a seguir para solucionar problemas de grupos de trabalho habilitados para Spark no Athena.

Sessão deixa de responder ao usar um perfil do IAM existente

Se você não criou um novo `AWSAthenaSparkExecutionRole` para o grupo de trabalho habilitado para Spark e, em vez disso, atualizou ou escolheu um perfil do IAM existente, sua sessão pode deixar de responder. Nesse caso, pode ser necessário adicionar as seguintes políticas de confiança e permissões ao perfil de execução do grupo de trabalho habilitado para Spark.

Adicione a política de confiança exemplificada a seguir. A política inclui uma verificação adjunta confundida para o perfil de execução. Substitua os valores de *111122223333*, *aws-region* e *workgroup-name* pelo ID da Conta da AWS, pela Região da AWS e pelo grupo de trabalho que você está usando.

```
{
    "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"
                }
```

```
            }
        }
    ]
}
```

Adicione uma política de permissões como a política padrão a seguir para os grupos de trabalho habilitados para cadernos. Modifique os locais reservados do Amazon S3 e os IDs da Conta da AWS para corresponder aos que você está usando. Substitua os valores de `DOC-EXAMPLE-BUCKET`, *`aws-region`*, *`111122223333`* e *`workgroup-name`* pelo bucket do Amazon S3, pela Região da AWS, pelo ID da Conta da AWS e pelo grupo de trabalho que você está usando.

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

Uso da instrução EXPLAIN do Spark para solucionar problemas do Spark SQL

É possível usar a instrução `EXPLAIN` do Spark com o Spark SQL para solucionar problemas relacionados ao código Spark. Os exemplos de código e de saída a seguir apresentam esse uso.

Example : instrução SELECT do Spark

```
spark.sql("select * from select_taxi_table").explain(True)
```

Saída

```
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 : estrutura de dados do Spark

O exemplo a seguir mostra como usar `EXPLAIN` com uma estrutura de dados do Spark.

```
taxi1_df=taxi_df.groupBy("VendorID", "passenger_count").count()
taxi1_df.explain("extended")
```

Saída

```
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>
```

Registro em log de eventos da aplicação Spark no Athena

O editor de cadernos do Athena permite o registro em log padrão do Jupyter, do Spark e do Python. É possível usar `df.show()` para exibir o conteúdo do PySpark DataFrame ou usar `print("Output")` para exibir os valores na saída da célula. As saídas `stdout`, `stderr` e `results` para seus cálculos serão gravadas no local do bucket de resultados da consulta no Amazon S3.

Registro em log de eventos da aplicação Spark no Amazon CloudWatch

As sessões do Athena também podem gravar logs no Amazon CloudWatch na conta que você está usando.

Noções básicas sobre fluxos de logs e grupos de logs

O CloudWatch organiza a atividade de log em fluxos de logs e grupos de logs.

Fluxos de logs: um fluxo de logs do CloudWatch corresponde a uma sequência de eventos de logs que compartilham a mesma origem. Cada origem separada de logs no CloudWatch Logs compõe um fluxo de logs separado.

Grupos de logs: no CloudWatch Logs, um grupo de logs corresponde a um grupo de fluxos de logs que compartilham as mesmas configurações de retenção, monitoramento e controle de acesso.

Não há limite para o número de streams de log que podem pertencer a um grupo de logs.

No Athena, quando você inicia uma sessão do caderno pela primeira vez, o Athena cria um grupo de logs no CloudWatch que usa o nome do seu grupo de trabalho habilitado para Spark, como no exemplo a seguir.

```
/aws-athena/workgroup-name
```

Esse grupo de logs recebe um fluxo de logs para cada executor em sua sessão, o que produz, no mínimo, um evento de logs. Um executor corresponde a menor unidade de computação que uma sessão do caderno pode solicitar do Athena. No CloudWatch, o nome do fluxo de logs começa com o ID da sessão e o ID do executor.

Para obter mais informações sobre os grupos de logs e os fluxos de logs do CloudWatch, consulte Trabalhar com grupos de logs e fluxos de logs no Guia do usuário do Amazon CloudWatch Logs.

Uso de objetos logger padrões no Athena para Spark

Em uma sessão do Athena para Spark, é possível usar os dois objetos logger de padrão global a seguir para gravar logs no Amazon CloudWatch:

- athena_user_logger: envia logs somente para o CloudWatch. Use esse objeto quando desejar registrar em log as informações de suas aplicações Spark diretamente no CloudWatch, como no exemplo a seguir.

  ```
  athena_user_logger.info("CloudWatch log line.")
  ```

  O exemplo grava um evento de logs no CloudWatch como o seguinte:

  ```
  AthenaForApacheSpark: 2022-01-01 12:00:00,000 INFO builtins: CloudWatch log line.
  ```

- athena_shared_logger: envia o mesmo log para o CloudWatch e para a AWS para fins de suporte. É possível usar esse objeto para compartilhar logs com as equipes de serviço da AWS para solução de problemas, como no exemplo a seguir.

  ```
  athena_shared_logger.info("Customer debug line.")
  var = [...some variable holding customer data...]
  athena_shared_logger.info(var)
  ```

  O exemplo registra em log a linha `debug` e o valor da variável `var` no CloudWatch Logs e envia uma cópia de cada linha para o AWS Support.

> **Note**
>
> Para sua privacidade, o código e os resultados dos cálculos não são compartilhados com a AWS. Certifique-se de que suas chamadas `athena_shared_logger` gravem somente as informações que você deseja tornar visíveis para o AWS Support.

Os loggers fornecidos gravam eventos por meio do Apache Log4j e herdam os níveis de registro em log dessa interface. Os valores de nível de log possíveis são `DEBUG`, `ERROR`, `FATAL`, `INFO` e `WARN` ou `WARNING`. É possível usar a função nomeada correspondente no logger para produzir esses valores.

> **Note**
>
> Não vincule novamente os nomes `athena_user_logger` ou `athena_shared_logger`. Fazer isso torna os objetos de registro em log incapazes de gravar no CloudWatch pelo restante da sessão.

Exemplo: registro em log de eventos de cadernos no CloudWatch

O procedimento a seguir mostra como registrar em log os eventos de cadernos do Athena no Amazon CloudWatch Logs.

Para registrar em log eventos de cadernos do Athena no Amazon CloudWatch Logs

1. Siga Conceitos básicos do Apache Spark no Amazon Athena para criar um grupo de trabalho habilitado para Spark no Athena com um nome exclusivo. Este tutorial usa o nome `athena-spark-example` para o grupo de trabalho.
2. Siga as etapas em Como criar seu próprio caderno para criar um caderno e iniciar uma nova sessão.
3. No editor de cadernos do Athena, em uma nova célula do caderno, digite o seguinte comando:

   ```
   athena_user_logger.info("Hello world.")
   ```
4. Execute a célula.
5. Recupere o ID da sessão atual seguindo um destes procedimentos:

- Visualize a saída da célula (por exemplo, `... session=72c24e73-2c24-8b22-14bd-443bdcd72de4`).
- Em uma nova célula, execute o comando mágico `%session_id`.

6. Salve o ID da sessão.
7. Com a mesma Conta da AWS que você está usando para executar a sessão do caderno, abra o console do CloudWatch em https://console.aws.amazon.com/cloudwatch/.
8. No painel de navegação do console do CloudWatch, escolha Log groups (Grupos de logs).
9. Na lista de grupos de logs, escolha o grupo de logs que tem o nome do seu grupo de trabalho do Athena habilitado para Spark, como no exemplo a seguir.

```
/aws-athena/athena-spark-example
```

A seção Log streams (Fluxos de logs) contém uma lista de um ou mais links de fluxo de logs para o grupo de trabalho. Cada nome de fluxo de logs contém o ID da sessão, o ID do executor e o UUID exclusivo separados por caracteres de barra.

Por exemplo, se o ID da sessão for `5ac22d11-9fd8-ded7-6542-0412133d3177` e o ID do executor for `f8c22d11-9fd8-ab13-8aba-c4100bfba7e2`, o nome do fluxo de logs será semelhante ao exemplo a seguir.

```
5ac22d11-9fd8-ded7-6542-0412133d3177/f8c22d11-9fd8-ab13-8aba-c4100bfba7e2/f012d7cb-cefd-40b1-90b9-67358f003d0b
```

10. Escolha o link do fluxo de logs para sua sessão.
11. Na página Log events (Eventos de logs), visualize a coluna Message (Mensagem).

O evento de logs para a célula executada será semelhante ao seguinte:

```
AthenaForApacheSpark: 2022-01-01 12:00:00,000 INFO builtins: Hello world.
```

12. Retorne ao editor de cadernos do Athena.
13. Em uma nova célula, insira o código a seguir. O código registrará em log uma variável no CloudWatch:

```
x = 6
athena_user_logger.warn(x)
```

14. Execute a célula.
15. Retorne à página Log events (Eventos de logs) do console do CloudWatch para obter um fluxo de logs semelhante.
16. O fluxo de logs agora vai conter uma entrada de evento de logs com uma mensagem como a seguinte:

```
AthenaForApacheSpark: 2022-01-01 12:00:00,000 WARN builtins: 6
```

Uso do CloudTrail para solucionar problemas de chamadas de API de cadernos do Athena

Para solucionar problemas de chamadas de API de cadernos, é possível examinar os logs do Athena CloudTrail para investigar anomalias ou descobrir ações iniciadas pelos usuários. Para obter informações detalhadas sobre como usar o CloudTrail com o Athena, consulte Registro de chamadas de API do Amazon Athena com o AWS CloudTrail.

Os exemplos a seguir demonstram as entradas de log do CloudTrail para APIs de cadernos do Athena:

- StartSession
- TerminateSession
- ImportNotebook
- UpdateNotebook
- StartCalculationExecution

StartSession

O exemplo a seguir mostra o log do CloudTrail para um evento StartSession do caderno.

```
{
    "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

O exemplo a seguir mostra o log do CloudTrail para um evento TerminateSession do caderno.

```
{
    "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

O exemplo a seguir mostra o log do CloudTrail para um evento ImportNotebook do caderno. Por segurança, alguns conteúdos foram ocultados.


```
{
    "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

O exemplo a seguir mostra o log do CloudTrail para um evento UpdateNotebook do caderno. Por segurança, alguns conteúdos foram ocultados.


```
{
    "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

O exemplo a seguir mostra o log do CloudTrail para um evento StartCalculationExecution do caderno. Por segurança, alguns conteúdos foram ocultados.

```
{
    "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": {
```

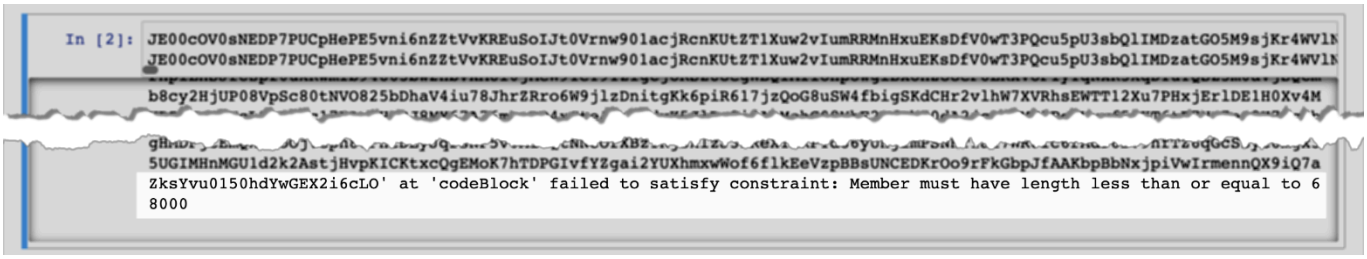
```
                "creationDate": "2022-10-14T16:48:06Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-10-14T16:52:37Z",
    "eventSource": "athena.amazonaws.com",
    "eventName": "StartCalculationExecution",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "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"
    }
}
```


Como superar o limite de tamanho do bloco de código de 68 mil

O Athena para Spark tem um limite de tamanho de bloco de código de cálculo conhecido de 68 mil caracteres. Ao executar um cálculo com um bloco de código acima desse limite, você poderá receber a seguinte mensagem de erro:

“…” em “codeBlock” não atende à restrição: o tamanho do membro deve ser menor ou igual a 68.000

A imagem a seguir mostra esse erro no editor de caderno do console do Athena.



O mesmo erro pode ocorrer quando você usa a AWS CLI para executar um cálculo que tenha um bloco de código grande, como no exemplo a seguir.

```
aws athena start-calculation-execution \
    --session-id "{SESSION_ID}" \
    --description "{SESSION_DESCRIPTION}" \
    --code-block "{LARGE_CODE_BLOCK}"
```

O comando apresenta a seguinte mensagem de erro:

{LARGE_CODE_BLOCK} em “codeBlock” não atende à restrição: o tamanho do membro deve ser menor ou igual a 68.000

Solução temporária

Para resolver esse problema, carregue o arquivo que contém a consulta ou o código do cálculo para o Amazon S3. Em seguida, use o boto3 para ler o arquivo e executar o SQL ou código.

Os exemplos a seguir pressupõem que você já carregou o arquivo que contém a consulta SQL ou o código Python no Amazon S3.

Exemplo de SQL

O código de exemplo a seguir lê o arquivo `large_sql_query.sql` em um bucket do Amazon S3 e executa a consulta grande que o arquivo contém.

```
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)
```

Exemplo do PySpark

O código de exemplo a seguir lê o arquivo `large_py_spark.py` no Amazon S3 e executa o bloco de código grande que está no arquivo.

```
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)
```

Solução de problemas de sessões

Use as informações apresentadas neste tópico para solucionar problemas de sessão.

Sessão em estado não íntegro

Se você receber a mensagem de erro: Session in unhealthy state. Please create a new session (Sessão em estado não íntegro. Crie uma nova sessão.), encerre a sessão existente e crie uma nova.

Não foi possível estabelecer uma conexão com o servidor do caderno

Ao abrir um caderno, é possível visualizar a mensagem de erro a seguir:

```
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.
```

Causa

Quando o Athena abre um caderno, ele cria uma sessão e se conecta ao caderno usando um URL previamente conectado para o caderno. A conexão com o caderno usa o protocolo WSS (WebSocket Secure).

O erro pode ocorrer pelos seguintes motivos:

- Um firewall local (por exemplo, um firewall que abrange toda a empresa) está bloqueando o tráfego WSS.
- O software proxy ou antivírus em seu computador local está bloqueando a conexão WSS.

Solução

Suponha que você tenha uma conexão WSS na região `us-east-1`, que é semelhante a seguir:

```
wss://94c2bcdf-66f9-4d17-9da6-7e7338060183.analytics-gateway.us-east-1.amazonaws.com/
api/kernels/33c78c82-b8d2-4631-bd22-1565dc6ec152/channels?session_id=
7f96a3a048ab4917b6376895ea8d7535
```

Para resolver o erro, use uma das seguintes estratégias.

- Use a sintaxe padrão de caráter universal para permitir a listagem do tráfego WSS na porta `443` entre as Regiões da AWS e as Contas da AWS.

  ```
  wss://*amazonaws.com
  ```

- Use a sintaxe padrão de caráter universal para permitir a listagem do tráfego WSS na porta `443` em uma Região da AWS e entre as Contas da AWS na Região da AWS que você especificar. O exemplo a seguir usa `us-east-1`.

  ```
  wss://*analytics-gateway.us-east-1.amazonaws.com
  ```

Solução de problemas de tabelas

Não é possível criar um caminho, ocorre um erro ao criar uma tabela

Mensagem de erro: IllegalArgumentException: Cannot create a path from an empty string (IllegalArgumentException: não é possível criar um caminho usando uma string vazia).

Causa: esse erro pode ocorrer quando você usa o Apache Spark no Athena para criar uma tabela em um banco de dados do AWS Glue, e o banco de dados tem uma propriedade `LOCATION` vazia.

Solução sugerida: para obter mais informações e soluções, consulte Exceção para argumento inválido ao criar uma tabela.

AccessDeniedException ao consultar tabelas do AWS Glue

Mensagem de erro: pyspark.sql.utils.AnalysisException: Unable to verify existence of default database: com.amazonaws.services.glue.model.AccessDeniedException: User: arn:aws:sts::*aws-account-id*:assumed-role/AWSAthenaSparkExecutionRole-*unique-identifier*/AthenaExecutor-*unique-identifier* is not authorized to perform: glue:GetDatabase on resource: arn:aws:glue:*aws-region*:*aws-account-id*:catalog because no identity-based policy allows the glue:GetDatabase action (Service: AWSGlue; Status Code: 400; Error Code: AccessDeniedException; Request ID: *request-id*; Proxy: null) (pyspark.sql.utils.AnalysisException: não é possível verificar a existência do banco de dados padrão: com.amazonaws.services.glue.model.AccessDeniedException: o usuário: arn:aws:sts::aws-account-id:assumed-role/AWSAthenaSparkExecutionRole-unique -identifier/AthenaExecutor-unique-identifier não está autorizado a executar: glue:GetDatabase no recurso: arn:aws:glue:aws-region:aws-account-id:catalog porque nenhuma política baseada em identidade permite a ação glue:GetDatabase (serviço: AWSGlue; código de status: 400; código de erro: AccessDeniedException; ID da solicitação: request-id; proxy: nulo).

Causa: o perfil de execução do grupo de trabalho habilitado para Spark não tem permissões para acessar os recursos do AWS Glue.

Solução sugerida: para resolver esse problema, conceda ao perfil de execução acesso aos recursos do AWS Glue e, em seguida, edite a política de bucket do Amazon S3 para conceder acesso ao perfil de execução.

O procedimento a seguir descreve essas etapas com mais detalhes.

Para conceder acesso aos recursos do AWS Glue ao perfil de execução

1. Abra o console do Athena em https://console.aws.amazon.com/athena/.
2. Se o painel de navegação do console não estiver visível, escolha o menu de expansão à esquerda.

3. No painel de navegação do console do Athena, escolha Workgroups (Grupos de trabalho).
4. Na página Workgroups (Grupos de trabalho), escolha o link do grupo de trabalho que você deseja visualizar.
5. Na página Overview Details (Detalhes gerais) para o grupo de trabalho, escolha o link Role ARN (ARN do perfil). O link abrirá o perfil de execução do Spark no console do IAM.
6. Na seção Permissions policies (Políticas de permissões), escolha o nome da política de perfil vinculada.
7. Escolha Edit policy (Editar política) e, em seguida, selecione JSON.
8. Adicione o acesso ao AWS Glue para o perfil. Normalmente, você adiciona permissões para as ações `glue:GetDatabase` e `glue:GetTable`. Para obter mais informações sobre como configurar perfis do IAM, consulte Adicionar e remover permissões de identidade do IAM no Guia do usuário do IAM.
9. Selecione Review policy (Revisar política) e, em seguida, escolha, Save changes (Salvar alterações).
10. Edite a política de bucket do Amazon S3 para conceder acesso ao perfil de execução. Observe que é necessário conceder ao perfil acesso tanto ao bucket quanto aos objetos do bucket. Para visualizar as etapas, consulte Adicionar uma política de bucket usando o console do Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

Obter suporte

Para obter assistência da AWS, escolha Support (Suporte), na Support Center (Central de suporte) no AWS Management Console. Para facilitar sua experiência, tenha em mãos as informações a seguir:

- ID da consulta do Athena
- ID da sessão
- ID do cálculo

Notas de release

Descreve os recursos, as melhorias e as correções de bugs do Amazon Athena por data de lançamento.

Tópicos

- Notas de lançamento do Athena para 2024
- Notas de lançamento do Athena para 2023
- Notas de release do Athena para 2022
- Notas de release do Athena para 2021
- Notas de release do Athena para 2020
- Notas de release do Athena para 2019
- Notas de release do Athena para 2018
- Notas de release do Athena para 2017

Notas de lançamento do Athena para 2024

26 de abril de 2024

Publicado em 26/04/2024

Athena lança o driver JDBC versão 3.2.0. Para obter mais informações sobre esta versão do driver JDBC, consulte Notas de versão do JDBC 3.x do Amazon Athena. Para baixar o driver JDBC 3.x, consulte Download do driver JDBC 3.x.

24 de abril de 2024

Publicado em 24/04/2024

Athena anuncia as correções e melhorias a seguir.

- Parquet: o Athena agora suporta leituras compatíveis com versões anteriores no Parquet para campos primitivos repetidos e sem anotações que não estão contidos em uma lista ou grupo de mapas. Essa alteração evita que resultados silenciosamente incorretos sejam retornados e melhora as mensagens de erro de incompatibilidades de esquema.

Para obter mais informações, consulte Support backwards compatible reads for unannotated repeated primitive fields in Parquet em GitHub.com.

- Iceberg OPTIMIZE: resolução de um problema com consultas `OPTIMIZE` que causava a perda de dados quando um filtro de chave sem partição era usado em uma cláusula `WHERE`. Para ter mais informações, consulte OPTIMIZE.

16 de abril de 2024

Publicado em 16/04/2024

Use o novo recurso de passagem de consultas federadas do Amazon Athena para executar consultas inteiras diretamente na fonte de dados subjacente. As consultas de passagem federadas ajudam você a aproveitar as exclusivas funções, linguagem de consulta e recursos de desempenho da fonte de dados original. Por exemplo, você pode executar consultas do Athena no DynamoDB usando a linguagem partiQL. As consultas de passagem federadas também são úteis quando você quiser executar consultas `SELECT` que agreguem, unam ou invoquem funções da sua fonte de dados que não estejam disponíveis no Athena. O uso de consultas de passagem pode reduzir a quantidade de dados processados pelo Athena e resultar em menores tempos de consulta.

Para ter mais informações, consulte Como realizar consultas de passagem federadas. Para atualizar com a versão mais recente os conectores que você já utiliza, consulte Atualizar um conector de fonte de dados.

10 de abril de 2024

Publicado em 10/04/2024

O Athena anuncia novos recursos e aperfeiçoamentos a seguir.

Driver ODBC 1.2.3.1000

Lançamento do driver ODBC 1.2.3.1000 para o Athena.

Problemas resolvidos:

- Problema de conexão do servidor proxy: quando um servidor proxy era usado sem o certificado raiz, o conector não conseguia estabelecer uma conexão.

Para obter mais informações e baixar o driver ODBC 1.x, notas de versão e documentação, consulte Driver ODBC 1.x do Athena.

Driver JDBC 2.1.5

Lançamento do driver JBDC 2.1.5 para o Athena.

Atualizações e aprimoramentos:

- Atualização do AWS Java SDK para usar a versão 1.12.687.
- Atualização das bibliotecas Jackson para usar a versão 2.16.0.
- Atualização das bibliotecas Logback para usar a versão 1.3.14.

Para obter mais informações e baixar o driver JDBC 2.x, notas de versão e documentação, consulte Driver JDBC 2.x do Athena.

8 de abril de 2024

Publicado em 08/04/2024

Athena anuncia o driver ODBC versão 2.0.3.0. Para obter mais informações, consulte as notas de versão do 2.0.3.0. Para baixar o driver ODBC v2, consulte Download do driver ODBC 2.x. Para obter informações sobre conexão, consulte o Amazon Athena ODBC 2.x.

15 de março de 2024

Publicado em 18/03/2024

Amazon Athena anuncia a disponibilidade do Athena SQL na região Oeste do Canadá (Calgary).

Para obter uma lista completa dos Serviços da AWS disponíveis em cada Região da AWS, consulte Serviços da AWS por região.

15 de fevereiro de 2024

Publicado em 15/02/2024

Athena lança o driver JDBC versão 3.1.0.

A versão 3.1.0 do driver JDBC do Amazon Athena acrescenta compatibilidade com a autenticação integrada do Windows por meio do Microsoft Active Directory Federation Services (AD FS) e

autenticação com base em formulário. A versão 3.1.0 também inclui outras pequenas melhorias e correções de problemas.

Para baixar o driver JDBC v3, consulte Download do driver JDBC 3.x.

31 de janeiro de 2024

Publicado em 31/01/2024

O Athena anuncia novos recursos e aperfeiçoamentos a seguir.

- Atualização do Hudi: agora, você pode usar o Athena SQL para consultar tabelas do Hudi 0.14.0. Para obter informações sobre como usar o Athena SQL para consultar tabelas do Hudi, consulte. Usar o Athena para consultar conjuntos de dados do Apache Hudi

Notas de lançamento do Athena para 2023

14 de dezembro de 2023

Publicado em 14/12/2023

Athena anuncia as correções e melhorias a seguir.

Athena lança a versão 2.1.3 do driver JDBC. O driver resolve os seguintes problemas:

- O registro em log foi aprimorado para evitar conflitos com o registro em log das aplicações Spring Boot e Gradle.
- Ao usar o método `executeBatch()` do JDBC para inserir registros, o driver inseriu somente um registro de forma incorreta. Como o Athena não oferece suporte à execução de consultas em lote, o driver passou a relatar um erro quando você usa `executeBatch()`. Para compensar a limitação, é possível enviar consultas únicas em um loop.

Para baixar o novo driver JDBC, as notas de lançamento e a documentação, consulte Driver JDBC 2.x do Athena.

9 de dezembro de 2023

Publicado em 9/12/2023

Lançamento do driver ODBC 1.2.1.1000 para o Athena.

Recursos e aprimoramentos:

- Suporte atualizado para o RStudio: o driver ODBC passou a oferecer suporte ao RStudio no macOS.
- Suporte para catálogo e esquema únicos: agora o conector pode retornar um catálogo e um esquema únicos. Para obter mais informações, consulte o guia de instalação e de configuração disponíveis para download.

Problemas resolvidos:

- Instruções preparadas: quando instruções preparadas com uma matriz de parâmetros que usa o esquema em colunas eram executadas, o conector retornava um resultado de consulta incorreto.
- Tamanho da coluna: quando a coluna do sistema `$file_modified_time` era selecionada, o conector retornava um tamanho de coluna incorreto.
- Função SQLPrepare: ao vincular parâmetros relacionados a `SQLPrepare` em consultas `SELECT`, o conector retornava um erro.

Para obter mais informações e baixar os novos drivers, as notas de lançamento e a documentação, consulte Driver ODBC 1.x do Athena.

7 de dezembro de 2023

Publicado em 7/12/2023

O Athena anuncia a versão 2.0.2.1 do driver ODBC. Para obter mais informações, consulte as notas de versão do 2.0.2.1. Para baixar o driver ODBC v2, consulte Download do driver ODBC 2.x. Para obter informações sobre conexão, consulte o Amazon Athena ODBC 2.x.

5 de dezembro de 2023

Publicado em 5/12/2023

Agora, é possível criar grupos de trabalho do Athena SQL que usam o modo de autenticação do AWS IAM Identity Center. Esses grupos de trabalho são compatíveis com o recurso de propagação de identidade confiável do Centro de Identidade do IAM. A propagação de identidade confiável

permite que as identidades sejam usadas em serviços de análise da AWS, como o Amazon Athena e o Amazon EMR Studio.

Para ter mais informações, consulte Uso de grupos de trabalho do Athena habilitados para o Centro de Identidade do IAM.

28 de novembro de 2023

Publicado em 28/11/2023

Agora, é possível consultar dados na classe de armazenamento Amazon S3 Express One Zone para obter resultados de consultas com rapidez. O S3 Express One Zone é uma classe de armazenamento com zona de disponibilidade única e alta performance desenvolvida com propósito específico de fornecer acesso consistente e abaixo de dez milissegundos a dados para os dados acessados com mais frequência e aplicações sensíveis à latência. Para começar a usar, mova os dados para o armazenamento S3 Express One Zone e catalogue os dados com o AWS Glue Data Catalog para obter uma experiência de consulta sem complicações no Athena.

Para ter mais informações, consulte Consulta de dados do S3 Express One Zone.

27 de novembro de 2023

Publicado em 27/11/2023

O Athena anuncia novos recursos e aperfeiçoamentos a seguir.

- Visualizações do Catálogo de Dados do Glue: as visualizações do Catálogo de Dados do Glue fornecem uma perspectiva única e comum sobre os serviços da AWS, como o Amazon Athena e o Amazon Redshift. Nas visualizações do Catálogo de Dados do Glue, as permissões de acesso são definidas pelo usuário que criou a visualização, e não pelo usuário que consulta a visualização. Essas visualizações fornecem maior controle sobre o acesso, ajudam a garantir registros completos, oferecem segurança aprimorada e podem impedir o acesso a tabelas subjacentes.

  Para ter mais informações, consulte Uso das visualizações do AWS Glue Data Catalog.
- Suporte ao CloudTrail Lake: passou a ser possível usar o Amazon Athena para analisar dados no AWS CloudTrail Lake. O AWS CloudTrail Lake corresponde a um data lake gerenciado para o CloudTrail que você pode usar para agregar, armazenar e analisar logs de atividades de forma imutável para investigações operacionais, de auditoria e de segurança. Para consultar os logs de

atividades do CloudTrail Lake usando o Athena, não é necessário mover dados ou desenvolver pipelines de processamento de dados separados. Nenhuma operação de ETL é exigida.

Para começar a usar, habilite a federação de dados no CloudTrail Lake. Quando você compartilha os metadados relativos ao armazenamento de dados de eventos do CloudTrail Lake com o AWS Glue Data Catalog, o CloudTrail cria os recursos necessários do AWS Glue Data Catalog e registra os dados com o AWS Lake Formation. No Lake Formation, é possível especificar os usuários e os perfis que podem usar o Athena para consultar o seu armazenamento de dados de eventos.

Para obter mais informações, consulte Enable Lake query federation no Guia do usuário do AWS CloudTrail.

17 de novembro de 2023

Publicado em 17/11/2023

O Athena anuncia novos recursos e aperfeiçoamentos a seguir.

Atributos

- Otimizador baseado em custos: o Athena anuncia a disponibilidade geral da otimização baseada em custos usando as estatísticas do AWS Glue. Para otimizar as consultas no Athena SQL, você pode solicitar que o Athena colete estatísticas no nível da tabela ou da coluna para as tabelas do AWS Glue. Se todas as tabelas em sua consulta tiverem estatísticas, o Athena usará essas estatísticas para examinar planos de execução alternativos e selecionar o que tiver a probabilidade de ser o mais rápido.

  Para ter mais informações, consulte Usar o otimizador baseado em custos.
- Integração com o Amazon EMR Studio: agora você pode usar o Athena em um Amazon EMR Studio sem precisar usar o console do Athena diretamente. Com a integração do Athena no Amazon EMR, você pode executar as seguintes tarefas:
  - Fazer consultas SQL do Athena
  - Visualizar resultados da consulta
  - Visualizar o histórico de consultas
  - Visualizar as consultas salvas
  - Fazer consultas parametrizadas
  - Visualizar bancos de dados, tabelas e visualizações de um catálogo de dados

Para obter mais informações, consulte Amazon EMR StudioIntegrações de AWS service (Serviço da AWS) ao Athena no tópico.

- Controle de acesso aninhado: o Athena anuncia que terá compatibilidade com o controle de acesso do Lake Formation para dados aninhados. No Lake Formation, você pode definir e aplicar filtros de dados em colunas aninhadas que tenham os tipos de dados `struct`. Você pode usar a filtragem de dados para restringir o acesso do usuário às subestruturas das colunas aninhadas. Para obter mais informações sobre como criar filtros para dados aninhados, consulte Creating a data filter no AWS Lake Formation Developer Guide.
- Métricas de uso da capacidade provisionada: o Athena anuncia novas métricas do CloudWatch para reservas de capacidade. Você pode usar as novas métricas para acompanhar o número de DPUs provisionadas e o número de DPUs sendo usadas por suas consultas. Quando as consultas terminam, você também pode visualizar o número de DPUs que elas consumiram.

  Para ter mais informações, consulte Monitorar consultas do Athena com métricas do CloudWatch.

Melhorias

- Alteração da mensagem de erro: a mensagem de erro `Insufficient Lake Formation permissions` agora diz `Table not found` ou `Schema not found`. Essa alteração foi feita para evitar que agentes mal-intencionados inferissem a existência de recursos de tabela ou banco de dados a partir da mensagem de erro.

16 de novembro de 2023

Publicado em 16/11/2023

O Athena lançou um novo driver JDBC que melhora a experiência de conexão, consulta e visualização de dados das aplicações SQL compatíveis de desenvolvimento e business intelligence. O novo driver é fácil de atualizar. O driver pode ler resultados de consultas diretamente no Amazon S3, disponibilizando-os em menos tempo.

Para ter mais informações, consulte Driver JDBC 3.x do Athena.

31 de outubro de 2023

Publicado em 31/10/2023

O Amazon Athena anuncia reservas de 1 hora para capacidade provisionada. A partir de hoje, você pode reservar e liberar a capacidade provisionada depois de uma hora. Essa alteração simplifica a otimização de custos de workloads cuja demanda muda com o tempo.

A capacidade provisionada é um atributo do Athena que oferece recursos de gerenciamento de workload que ajudam a priorizar, controlar e escalar as workloads interativas mais importantes. Você pode adicionar capacidade a qualquer momento para aumentar o número de consultas que poderão ser executadas simultaneamente, controlar quais workloads usarão a capacidade e compartilhar a capacidade entre as workloads.

Para ter mais informações, consulte Como gerenciar a capacidade de processamento de consulta. Para obter informações de preço, visite a página Preços do Amazon Athena.

25 de outubro de 2023

Publicado em 26/10/2023

Athena anuncia as correções e melhorias a seguir.

pacote do jackson-core: agora ocorrerá falha no texto JSON com um valor numérico com mais de 1.000 caracteres. Essa correção soluciona o problema de segurança sonatype-2022-6438.

17 de outubro de 2023

Publicado em 17/10/2023

O Athena anuncia o driver ODBC versão 2.0.2.0. Para obter mais informações, consulte as notas de versão do 2.0.2.0. Para baixar o driver ODBC v2, consulte Download do driver ODBC 2.x. Para obter informações sobre conexão, consulte o Amazon Athena ODBC 2.x.

26 de setembro de 2023

Publicado em 26/09/2023

O Athena anuncia novos recursos e aperfeiçoamentos a seguir.

- Suporte de leitura do Lake Formation para tabelas do Delta Lake. Para obter informações sobre o uso de tabelas do Delta Lake no Athena, consulte Consultar tabelas do Linux Foundation Delta Lake.

23 de agosto de 2023

Publicado em 23/08/2023

O Amazon Athena anuncia a disponibilidade do Athena SQL na região de Israel (Tel Aviv).

Para obter uma lista completa dos Serviços da AWS disponíveis em cada Região da AWS, consulte Serviços da AWS por região.

10 de agosto de 2023

Publicado em 10/08/2023

Athena anuncia as correções e melhorias a seguir.

Driver ODBC versão 2.0.1.1

O Athena anuncia o driver ODBC versão 2.0.1.1. Para obter mais informações, consulte as notas de versão do 2.0.1.1. Para baixar o driver ODBC v2, consulte Download do driver ODBC 2.x. Para obter informações sobre conexão, consulte o Amazon Athena ODBC 2.x.

Driver JDBC versão 2.1.1

Athena lança o driver JDBC versão 2.1.1. O driver resolve os seguintes problemas:

- Um erro que ocorria quando uma tabela era criada com uma instrução que continha uma expressão regular.
- Um problema que causava o parâmetro de conexão `ApplicationName` a ser aplicado incorretamente.

Para baixar o novo driver JDBC, as notas de lançamento e a documentação, consulte Conectar-se ao Amazon Athena com JDBC.

31 de julho de 2023

Publicado em 31/07/2023

O Amazon Athena anuncia a disponibilidade do Athena SQL em outras Regiões da AWS.

Esta versão expande a disponibilidade do Athena SQL para incluir Ásia-Pacífico (Hyderabad), Ásia-Pacífico (Melbourne), Europa (Espanha) e Europa (Zurique).

Para obter uma lista completa dos Serviços da AWS disponíveis em cada Região da AWS, consulte Serviços da AWS por região.

27 de julho de 2023

Publicado em 27/7/2023

Athena lança a versão 2023.30.1 do conector Google BigQuery. Essa versão do conector reduz o tempo de execução da consulta e adiciona suporte à consulta em endpoints privados do BigQuery.

Para obter informações sobre o conector do Google BigQuery, consulte Conector do Amazon Athena para o Google BigQuery. Para obter mais informações sobre como atualizar os conectores da fonte de dados existentes, consulte Atualizar um conector de fonte de dados.

24 de julho de 2023

Publicado em 24/7/2023

Athena anuncia as correções e melhorias a seguir.

- Consultas com uniões: melhorou o desempenho de determinadas consultas com uniões.
- Junções com comparações de tipos: corrigiu uma possível falha de consulta para instruções `JOIN` que incluíam uma comparação entre dois tipos diferentes.
- Subconsultas em colunas aninhadas: corrigiu um problema relacionado a falhas de consulta quando as subconsultas eram correlacionadas em colunas aninhadas.
- Visualizações do Iceberg: corrigiu um problema de compatibilidade com a precisão das colunas de timestamps nas visualizações do Apache Iceberg. As visualizações do Iceberg que têm colunas de timestamp agora podem ser lidas independentemente de terem sido criadas na versão 2 ou na versão 3 do mecanismo Athena.

20 de julho de 2023

Publicado em 20/7/2023

Athena lança o driver JDBC versão 2.1.0. O driver contém novos aprimoramentos e resolveu um problema.

Melhorias

As seguintes bibliotecas Jackson de análise JSON foram atualizadas:

- jackson-annotations 2.15.2 (anteriormente 2.14.0)
- jackson-core 2.15.2 (anteriormente 2.14.0)
- jackson-databind 2.15.2 (anteriormente 2.14.0)

Problemas resolvidos

- Foi corrigido um problema com a passagem de parâmetros de matriz quando a biblioteca sql2o foi usada.

Para obter mais informações e baixar os novos drivers, as notas de lançamento e a documentação, consulte Conectar-se ao Amazon Athena com JDBC.

13 de julho de 2023

Publicado em 09/19/2023

O Athena anuncia novos recursos e aperfeiçoamentos a seguir.

- EXPLAIN ANALYZE — Foi adicionado suporte para fila, análise, planejamento e tempo de execução à saída de `EXPLAIN ANALYZE`.
- EXPLAIN — a `EXPLAIN` saída agora mostra estatísticas quando a consulta contém agregações.
- Parquet Hive SerDe — Foi adicionada a `parquet.ignore.statistics` propriedade para permitir que as estatísticas de processamento sejam ignoradas ao ler dados do Parquet. Para ter mais informações, consulte Ignorando estatísticas do Parquet.

Para obter mais informações sobre `EXPLAIN` e `EXPLAIN ANALYZE`, consulte Usar EXPLAIN e EXPLAIN ANALYZE no Athena. Para obter mais informações sobre o Parquet Hive SerDe, consulte Parquet SerDe.

3 de julho de 2023

Publicado em 25/7/2023

A partir de 3 de julho de 2023, o Athena começou a ocultar as strings de de consulta dos logs do CloudTrail. A string de consulta agora tem um valor de `***OMITTED***`. Essa alteração foi feita para evitar a divulgação não intencional de nomes de tabelas ou valores de filtro que poderiam incluir informações sensíveis. Se você anteriormente dependia dos logs do CloudTrail para acesso total de strings de consulta, recomendamos usar a API `Athena::GetQueryExecution` e transmitir no valor de `responseElements.queryExecutionId` do log do CloudTrail. Para obter mais informações, consulte a ação GetQueryExecution na Amazon Athena API Reference.

30 de junho de 2023

Publicado em 30/6/2023

O editor de consultas do Athena agora é compatível com sugestões de código de digitação antecipada para proporcionar uma experiência de criação de consultas mais rápida. Agora você pode escrever consultas SQL com maior precisão e eficiência usando os seguintes atributos:

- Enquanto você digita, as sugestões são exibidas em tempo real para palavras-chave, variáveis locais, trechos e itens de catálogo.
- Quando você digita um nome de banco de dados ou nome de tabela seguido por um ponto, o editor exibe, de maneira conveniente, uma lista de tabelas ou colunas para escolher.
- Quando você passa o mouse sobre uma sugestão de trecho, uma sinopse exibe uma breve visão geral da sintaxe e do uso do trecho.
- Para melhorar a legibilidade do código, as palavras-chave e as respectivas regras de destaque também foram atualizadas para se alinharem à sintaxe mais recente do Trino e do Hive.

Esse recurso está habilitado por padrão. É possível habilitar ou desabilitar o atributo nas configurações de preferências do editor de código.

Para experimentar as sugestões de código de digitação antecipada no editor de consultas do Athena, acesse o console do Athena em https://console.aws.amazon.com/athena/.

29 de junho de 2023

Publicado em 29/6/2023

- O Athena anuncia o driver ODBC versão 2.0.1.0. Para obter mais informações, consulte as notas de versão do 2.0.1.0. Para baixar o driver ODBC v2, consulte Download do driver ODBC 2.x. Para obter informações sobre conexão, consulte o Amazon Athena ODBC 2.x.

- O Athena e seus atributos agora estão disponíveis na região do Oriente Médio (EAU). Para obter uma lista completa dos Serviços da AWS disponíveis em cada Região da AWS, consulte Serviços da AWS por região.

28 de junho de 2023

Publicado em 28/6/2023

Já é possível usar o Amazon Athena para consultar objetos restaurados das classes de armazenamento do Amazon S3 S3 Glacier Flexible Retrieval (antigo Glacier) e S3 Glacier Deep Archive. Configure esse recurso com base em tabelas. O atributo é compatível somente com tabelas do Apache Hive no mecanismo do Athena versão 3.

Para ter mais informações, consulte Consultar objetos restaurados do Amazon S3 Glacier.

12 de junho de 2023

Publicado em 12/6/2023

Athena anuncia as correções e melhorias a seguir.

- Carimbos de data e hora do Parquet Reader: a leitura de carimbos de data e hora como `bigint` (millissegundos) agora é compatível para o Parquet Reader. Esta atualização fornece paridade com o suporte no mecanismo do Athena versão 2.
- EXPLAIN ANALYZE: foi adicionado o tempo de leitura da entrada física às estatísticas da consulta e à saída de `EXPLAIN ANALYZE`. Para obter mais informações sobre o `EXPLAIN ANALYZE`, consulte Usar EXPLAIN e EXPLAIN ANALYZE no Athena.
- INSERT: melhoria na performance de consulta em tabelas gravadas com `INSERT`. Para obter mais informações sobre o `INSERT`, consulte INSERT INTO.
- Tabelas do Delta Lake: foi corrigido um problema com as tabelas `DROP TABLE` do Delta Lake que impedia que elas fossem totalmente excluídas quando sujeitas a modificações simultâneas.

8 de junho de 2023

Publicado em 8/6/2023

O Amazon Athena para Apache Spark anuncia os novos atributos a seguir.

- Suporte para bibliotecas e configurações personalizadas de Java: você já pode usar seus próprios pacotes e configurações personalizadas de Java para suas sessões do Apache Spark no Athena. Use as propriedades do Spark para especificar arquivos `.jar`, pacotes ou outras configurações personalizadas com o console do Athena, a AWS CLI ou a API do Athena. Para ter mais informações, consulte Adicionar arquivos JAR e configuração personalizada do Spark.
- Suporte para tabelas Apache Hudi, Apache Iceberg e Delta Lake: o Athena para Spark já é compatível com os formatos de tabela de armazenamento de data lake de código aberto Apache Iceberg, Apache Hudi e Linux Foundation Delta Lake. Para obter mais informações, consulte Usar formatos de tabela que não sejam do Hive no Amazon Athena para Apache Spark e os tópicos individuais de uso de tabelas Apache Iceberg, Apache Hudi e Linux Foundation Delta Lake no Athena para Spark.
- Criptografia compatível para o Apache Spark: no Athena para Spark, já é possível habilitar a criptografia em dados em trânsito entre os nós do Spark e em dados em repouso locais armazenados em disco pelo Spark. Para habilitar a criptografia do Spark, você pode usar o console do Athena, a AWS CLI ou a API do Athena. Para ter mais informações, consulte Habilitar a criptografia do Apache Spark.

Para obter mais informações sobre o Amazon Athena para Apache Spark, consulte Uso do Apache Spark no Amazon Athena.

2 de junho de 2023

Publicado em 02/06/2023

Agora você pode excluir reservas de capacidade no Athena e usar modelos do AWS CloudFormation para especificar as reservas de capacidade do Athena.

- Excluir reservas de capacidade: agora você pode excluir reservas de capacidade canceladas no Athena. Uma reserva deve ser cancelada para poder ser excluída. A exclusão de uma reserva de capacidade remove-a da sua conta imediatamente. A reserva excluída não pode mais ser referenciada nem por seu ARN. Para excluir uma reserva, você pode usar o console ou a API do Athena. Para obter mais informações, consulte Excluir uma reserva de capacidade no Guia do usuário do Amazon Athena e DeleteCapacityReservation na Referência de API do Amazon Athena.
- Usar modelos do AWS CloudFormation para reservas de capacidade: agora você pode usar modelos do AWS CloudFormation para especificar as reservas de capacidade do Athena

usando o recurso `AWS::Athena::CapacityReservation`. Para obter mais informações, consulte AWS::Athena::CapacityReservation no Guia do usuário do AWS CloudFormation.

Para obter mais informações sobre o uso de reservas de capacidade para provisionar capacidade no Athena, consulte Como gerenciar a capacidade de processamento de consulta.

25 de maio de 2023

Publicado em 25/05/2023

O Athena lançou atualizações de conectores de fonte de dados que melhoram a performance em consultas federadas. Novas otimizações de passagem direta e filtragem dinâmica permitem que mais operações sejam realizadas no banco de dados de origem em vez ocorrerem no Athena. Essas otimizações reduzem o runtime das consultas e a quantidade de dados examinados. Essas melhorias requerem o mecanismo Athena versão 3.

Os seguintes conectores foram atualizados:

- Azure Data Lake Storage
- Azure Synapse
- Cloudera Hive
- Cloudera Impala
- Db2
- DynamoDB
- Google BigQuery
- Hortonworks
- MySQL
- Oracle
- PostgreSQL
- Redshift
- SAP HANA
- Snowflake
- SQL Server
- Teradata

Para obter mais informações sobre como atualizar os conectores de fonte de dados, consulte Atualizar um conector de fonte de dados.

18 de maio de 2023

Publicado em 18/05/2023

Agora você pode usar o AWS PrivateLink para conexões de entrada IPv6 com o Amazon Athena.

O Amazon Athena ampliou a compatibilidade com conexões de entrada por endpoints Internet Protocol versão 6 (IPv6) para incluir o AWS PrivateLink. A partir de hoje, você pode se conectar ao Athena com segurança e privacidade usando o AWS PrivateLink da Amazon Virtual Private Cloud (Amazon VPC), além dos endpoints IPv6 públicos que estavam disponíveis anteriormente.

O rápido crescimento da Internet está esgotando a disponibilidade de endereços IPv4 (Protocolo de Internet versão 4). O IPv6 aumenta a quantidade de endereços disponíveis para que você não precise mais gerenciar espaços de endereço sobrepostos em suas VPCs. Com essa versão, agora você pode combinar os benefícios do endereçamento IPv6 com as vantagens de segurança e performance do AWS PrivateLink.

Para se conectar programaticamente a um serviço da AWS, você pode usar a AWS CLI ou o SDK da AWS para especificar um endpoint. Para obter mais informações sobre endpoints de serviço e endpoints de serviço do Athena, consulte endpoints de serviço da AWS e endpoints e cotas do Amazon Athena no Referência geral da Amazon Web Services.

15 de maio de 2023

Publicado em 15/05/2023

O Athena anuncia o lançamento dos conectores DataSourceV2 (DSV2) do Apache Spark para o DynamoDB, o CloudWatch Logs, o CloudWatch Metrics e o AWS CMDB. Use os novos conectores DSV2 para consultar essas fontes de dados usando o Spark. Os conectores DSV2 usam os mesmos parâmetros dos conectores federados Athena correspondentes. Os conectores DSV2 são executados diretamente nos operadores do Spark e não exigem que você implante uma função do Lambda para usá-los.

Para ter mais informações, consulte Conectores de fonte de dados do Athena para o Apache Spark.

10 de maio de 2023

Publicado em 10/05/2023

Lançamento do driver ODBC 1.1.20 para o Athena.

Recursos e aprimoramentos:

- Compatibilidade de endpoint Lake Formation com substituição.
- O plug-in de autenticação do ADFS tem um novo parâmetro para definir o valor de Terceira parte confiável (`LoginToRP`).
- Atualizações da biblioteca da AWS.

Correções de erros:

- Falha ao desalocar instrução previamente preparada quando ocorria uma falha no envio do método `SQLPrepare()`.
- Erro ao vincular parâmetros de instrução previamente preparada ao converter um tipo C em SQL.
- Falha no retorno de dados quando as consultas `EXPLAIN` e `EXPLAIN ANALYZE` usavam `SQLPrepare()` e `SQLExecute()`.

Para obter mais informações e baixar os novos drivers, as notas de lançamento e a documentação, consulte Conectar-se ao Amazon Athena com ODBC.

8 de maio de 2023

Publicado em 08/05/2023

Athena anuncia as correções e melhorias a seguir.

- Atualização da integração com o Hudi: o Athena atualizou sua integração com o Apache Hudi. Agora você pode usar o Athena para consultar as tabelas do Hudi 0.12.2, e as tabelas do Hudi agora são compatíveis com listagem de metadados do Hudi. Para obter informações, consulte Usar o Athena para consultar conjuntos de dados do Apache Hudi e Listagem de metadados do Hudi.
- Correção de conversão de timestamp: foi corrigido o tratamento de conversões de timestamp para um tipo de dados de menor precisão. Anteriormente, o mecanismo Athena versão 3 arredondava incorretamente o valor para o tipo de destino em vez de truncá-lo durante a conversão.

  Os exemplos a seguir ilustram o tratamento incorreto antes da correção.

  Exemplo 1: conversão de um timestamp em microssegundos para milissegundos

Dados de exemplo

```
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
```

A consulta a seguir tenta recuperar os timestamps que correspondem a um valor específico.

```
SELECT *
FROM table
WHERE timestamps.col = timestamp'2020-06-10 15:55:23.383'
```

A consulta retorna os resultados a seguir.

```
A, 2020-06-10 15:55:23.383
C, 2020-06-10 15:55:23.383
E, 2020-06-10 15:55:23.383
```

Antes da correção, o Athena não incluía os valores `2020-06-10 15:55:23.383945` ou `2020-06-10 15:55:23.383945278` porque eles eram arredondados para `2020-06-10 15:55:23.384`.

Exemplo 2: conversão de um timestamp em data

A consulta a seguir retornou um resultado incorreto.

```
SELECT date(timestamp '2020-12-31 23:59:59.999')
```

Resultado

`2021-01-01`

Antes da correção, o Athena arredondava o valor, adiantando, portanto, o dia. Esses valores agora são truncados em vez de arredondados.

28 de abril de 2023

Publicado em 28/4/2023

Já é possível usar reservas de capacidade no Amazon Athena para executar consultas SQL em capacidade de computação totalmente gerenciada.

A capacidade provisionada fornece recursos de gerenciamento de workload que ajudam a priorizar, controlar e escalar suas workloads interativas mais importantes. Você pode adicionar capacidade a qualquer momento para aumentar o número de consultas que poderão ser executadas simultaneamente, controlar quais workloads usarão a capacidade e compartilhar a capacidade entre as workloads.

Para ter mais informações, consulte Como gerenciar a capacidade de processamento de consulta. Para obter informações, acesse a página de preços do Amazon Athena.

17 de abril de 2023

Publicado em 17/4/2023

O Athena libera o driver JDBC versão 2.0.36. O driver contém novos recursos e resolveu um problema.

Novos atributos

- Já é possível usar identificadores personalizáveis de terceiros confiáveis com a autenticação AD FS.
- Já é possível adicionar o nome da aplicação que está usando o conector à string do agente do usuário.

Problemas resolvidos

- Foi corrigido um erro que ocorria com o uso de `getSchema()` para recuperar um esquema inexistente.

Para obter mais informações e baixar os novos drivers, as notas de lançamento e a documentação, consulte Conectar-se ao Amazon Athena com JDBC.

14 de abril de 2023

Publicado em 20/6/2023

Athena anuncia as correções e melhorias a seguir.

- Quando você converte uma string para carimbo de data e hora, é necessário um espaço entre o dia e a hora ou o fuso horário. Para ter mais informações, consulte Espaço necessário entre os valores de data e hora ao converter de string para carimbo de data e hora.
- Foi removida uma alteração importante na forma como a precisão do carimbo de data e hora era tratada. Para manter a consistência entre o mecanismo Athena versão 2 e o mecanismo Athena versão 3, a precisão do carimbo de data e hora foi padronizada para milissegundos em vez de microssegundos.
- O Athena agora impõe, de maneira consistente, o acesso ao bucket de saída da consulta ao executar consultas. Certifique-se de que todas as entidades principais do IAM que executam a ação StartQueryExecution tenham a permissão S3:GetBucketLocation no bucket de saída da consulta.

4 de abril de 2023

Publicado em 4/4/2023

Já é possível usar o Amazon Athena para criar e consultar visualizações em fontes de dados federadas. Use uma única visualização federada para consultar várias tabelas externas ou subconjuntos de dados. Isso simplifica o SQL necessário e oferece a flexibilidade de ofuscar fontes de dados de usuários finais que precisam usar SQL para consultar os dados.

Para obter mais informações, consulte Trabalhar com visualizações e Executar consultas federadas.

30 de março de 2023

Publicado em 30/3/2023

O Amazon Athena anuncia a disponibilidade do Amazon Athena para Apache Spark em outras Regiões da AWS.

Esta versão expandiu o Amazon Athena para Apache Spark e agora inclui Ásia-Pacífico (Mumbai), Ásia-Pacífico (Singapura), Ásia-Pacífico (Sydney) e Europa (Frankfurt).

Para obter mais informações sobre o Amazon Athena para Apache Spark, consulte Uso do Apache Spark no Amazon Athena.

28 de março de 2023

Publicado em 28/3/2023

Athena anuncia as correções e melhorias a seguir.

- Nas respostas às ações de API `GetQueryExecution` e `BatchGetQueryExecution` do Athena, o novo campo `subStatementType` mostra o tipo de consulta executada (por exemplo, `SELECT`, `INSERT`, `UNLOAD`, `CREATE_TABLE` ou `CREATE_TABLE_AS_SELECT`).
- Foi corrigido um bug no qual os arquivos de manifesto não eram criptografados corretamente para as operações de gravação do Apache Hive.
- O mecanismo Athena versão 3 agora manipula valores `NaN` e `Infinity` corretamente na função `approx_percentile`. A função `approx_percentile` retorna o percentil aproximado de um conjunto de dados na porcentagem dada.

  O mecanismo Athena versão 2 trata incorretamente `NaN` como um valor maior que `Infinity`. O mecanismo Athena versão 3 agora manipula `NaN` e `Infinity` de acordo com o tratamento desses valores em outras funções analíticas e estatísticas. Os pontos a seguir descrevem o novo comportamento com mais detalhes.

  - Se `NaN` estiver presente no conjunto de dados, o Athena retornará `NaN`.
  - Se `NaN` não estiver presente, mas `Infinity` estiver presente, o Athena tratará `Infinity` como um número muito grande.
  - Se vários valores `Infinity` estiverem presentes, o Athena os tratará como o mesmo número muito grande. Se necessário, o Athena produz `Infinity`.
  - Se um único conjunto de dados tiver tanto `Infinity` como `-Double.MAX_VALUE`, e um resultado percentil for `-Double.MAX_VALUE`, o Athena retornará `-Infinity`.
  - Se um único conjunto de dados tiver tanto `Infinity` como `Double.MAX_VALUE`, e um resultado percentil for `Double.MAX_VALUE`, o Athena retornará `Infinity`.
  - Para excluir `Infinity` e `NaN` de um cálculo, use a função `is_finite()`, como no exemplo a seguir.

    ```
    approx_percentile(x, 0.5) FILTER (WHERE is_finite(x))
    ```

27 de março de 2023

Publicado em 27/3/2023

Agora é possível especificar um nível mínimo de criptografia em grupos de trabalho de SQL do Athena no Amazon Athena. Esse recurso garante que os resultados de todas as consultas do grupo de trabalho de SQL do Athena sejam criptografados no nível de criptografia especificado ou acima. É possível escolher entre vários níveis de força de criptografia para proteger os dados. Para configurar o nível mínimo de criptografia desejado, use o console, a AWS CLI, a API ou o SDK do Athena.

O recurso de criptografia mínima não está disponível para grupos de trabalho habilitados para o Apache Spark. Para ter mais informações, consulte Configurar a criptografia mínima para um grupo de trabalho.

17 de março de 2023

Publicado em 17/3/2023

Athena anuncia as correções e melhorias a seguir.

- Foi corrigido um problema com o conector do Amazon Athena para DynamoDB que fazia com que as consultas falhassem com a mensagem de erro KeyConditionExpressions deve conter apenas uma condição por chave.

  Esse problema ocorre porque o mecanismo Athena versão 3 reconhece a oportunidade de propagar mais tipos de predicados do que o mecanismo Athena versão 2. No mecanismo Athena versão 3, cláusulas como `some_column LIKE 'someprefix%` são propagadas como predicados de filtro que aplicam um limite inferior e superior a determinada coluna. O mecanismo Athena versão 2 não propagava esses predicados. No mecanismo Athena versão 3, quando `some_column` é uma coluna de chave de classificação, o mecanismo propaga o predicado do filtro até o conector do DynamoDB. Em seguida, o predicado do filtro é propagado ainda para o serviço do DynamoDB. Como o DynamoDB não oferece suporte a mais de uma condição de filtro em uma chave de classificação, o DynamoDB retorna o erro.

  Para corrigir esse problema, atualize o conector do Amazon Athena para DynamoDB para a versão 2023.11.1. Para obter instruções sobre como atualizar o conector, consulte Atualizar um conector de fonte de dados.

8 de março de 2023

Publicado em 8/3/2023

Athena anuncia as correções e melhorias a seguir.

- Foi corrigido um problema com consultas federadas que fazia com que os valores dos predicados de carimbo de data e hora fossem enviados como microssegundos em vez de milissegundos.

15 de fevereiro de 2023

Publicado em 15/2/2023

Athena anuncia as correções e melhorias a seguir.

- Já é possível usar a criptografia do lado do cliente para criptografar dados no Amazon S3 para operações de gravação do Iceberg.
- Foi corrigido um problema que afetava a criptografia do lado do servidor no Amazon S3 em operações de gravação do Iceberg.

31 de janeiro de 2023

Publicado em 31/01/2023

Agora você pode usar o Amazon Athena para consultar dados no Google Cloud Storage. Como o Amazon S3, o Google Cloud Storage é um serviço gerenciado que armazena dados em buckets. Para executar consultas federadas interativas em seus dados externos, use o conector do Athena para Google Cloud Storage.

Para ter mais informações, consulte Conector Google Cloud Storage para Amazon Athena.

20 de janeiro de 2023

Publicado em 20/01/2023

Agora você pode ver a documentação expandida referente ao suporte à compressão do Athena. Foram adicionados tópicos adicionais referentes a Compressão de tabelas do Hive, Compressão de tabelas do Iceberg e Níveis de compressão ZSTD.

Para ter mais informações, consulte Suporte a compactação no Athena.

3 de janeiro de 2023

Publicado em 3/1/2023

O Athena anuncia as atualizações a seguir:

- Comandos adicionais para metastores do Hive: é possível usar o Athena para se conectar ao Apache Hive Metastore autogerenciado como um catálogo de metadados e dados de consulta armazenados no Amazon S3. Com esta versão, é possível usar `CREATE TABLE AS` (CTAS), `INSERT INTO` e 12 comandos adicionais de linguagem de definição de dados (DDL) para interagir com o Apache Hive Metastore. Você pode gerenciar os esquemas do Hive Metastore diretamente do Athena usando esse conjunto expandido de recursos SQL.

  Para ter mais informações, consulte Usar o conector de dados do Athena para metastore externo do Hive.
- Driver JDBC versão 2.0.35: o Athena realiza o lançamento do driver JDBC versão 2.0.35. O driver JDBC 2.0.35 contém as atualizações a seguir:
  - O driver passou a usar as bibliotecas a seguir para o analisador Jackson JSON.
    - jackson-annotations 2.14.0 (anteriormente 2.13.2)
    - jackson-core 2.14.0 (anteriormente 2.13.2)
    - jackson-databind 2.14.0 (anteriormente 2.13.2.2)
  - O suporte para o JDBC versão 4.1 foi descontinuado.

  Para obter mais informações e baixar do novo driver, das notas de lançamento e da documentação, consulte Conectar-se ao Amazon Athena com JDBC.

Notas de release do Athena para 2022

14 de dezembro de 2022

Publicado em 14/12/2022

Agora é possível usar o conector do Amazon Athena para Kafka para executar consultas SQL em dados de streaming. Por exemplo, você pode executar consultas analíticas em dados de streaming em tempo real no Amazon Managed Streaming for Apache Kafka (Amazon MSK) e associá-los a dados históricos em seu data lake no Amazon S3.

O conector do Amazon Athena para Kafka é compatível com consultas em diversos mecanismos de streaming. É possível usar o Athena para executar consultas SQL em clusters provisionados e com tecnologia sem servidor do Amazon MSK, em implantações autogerenciadas do Kafka e em streaming de dados na Confluent Cloud.

Para ter mais informações, consulte Conector MSK do Amazon Athena.

2 de dezembro de 2022

Publicado em 2/12/2022

O Athena realiza o lançamento do driver JDBC versão 2.0.34. O driver JDBC 2.0.34 inclui os novos recursos e problemas resolvidos apresentados a seguir:

- Reutilização dos resultados da consulta compatível: agora é possível reutilizar os resultados de consultas executadas anteriormente até um limite de tempo especificado, em vez de fazer com que o Athena recalcule os resultados sempre que a consulta for executada. Para obter mais informações, consulte o Guia de instalação e configuração, disponível na página de download do JDBC, e Reutilização dos resultados da consulta.
- Suporte para EC2InstanceMetadata: o driver JDBC passou a ser compatível com o método de autenticação EC2InstanceMetadata usando perfis de instância do IAM.
- Correção de exceção baseada em caracteres: uma exceção que ocorria com consultas contendo determinados caracteres de idioma foi corrigida.
- Correção de vulnerabilidade: uma vulnerabilidade relacionada às dependências da AWS empacotadas com o conector.

Para obter mais informações e baixar os novos drivers, as notas de lançamento e a documentação, consulte Conectar-se ao Amazon Athena com JDBC.

30 de novembro de 2022

Publicado em 30/11/2022

Agora é possível criar e executar, de forma interativa, aplicações Apache Spark e cadernos compatíveis com Jupyter no Athena. Execute análises de dados no Athena usando o Spark sem a necessidade de planejar, configurar ou gerenciar recursos. Envie o código Spark para processamento e receba os resultados de forma direta. Use a experiência simplificada de cadernos

no console do Amazon Athena para desenvolver aplicações do Apache Spark usando Python ou APIs para cadernos do Athena.

O Apache Spark no Amazon Athena corresponde a uma tecnologia sem servidor e oferece uma escalabilidade automática sob demanda que fornece computação instantânea para atender aos volumes de dados em constante mudança e aos requisitos de processamento.

Para ter mais informações, consulte Uso do Apache Spark no Amazon Athena.

18 de novembro de 2022

Publicado em 18/11/2022

Agora é possível usar o conector do Amazon Athena para IBM Db2 para consultar o Db2 do Athena. Por exemplo, você pode executar consultas analíticas em um data warehouse no Db2 e em um data lake no Amazon S3.

O conector Db2 do Amazon Athena apresenta diversas opções de configurações por meio de variáveis de ambiente do Lambda. Para obter informações sobre as opções de configuração, os parâmetros, as strings de conexão, a implantação e as limitações, consulte Conector IBM Db2 do Amazon Athena.

17 de novembro de 2022

Publicado em 17/11/2022

O suporte ao Apache Iceberg na versão 3 do mecanismo do Athena passou a oferecer os recursos de transação ACID aprimorados a seguir:

- Compatibilidade do ORC e Avro: crie tabelas do Iceberg usando os formatos de arquivo Apache Avro e Apache ORC baseados em linha e coluna. O suporte para esses formatos é adicional ao suporte existente para Parquet.
- MERGE INTO: use o comando `MERGE INTO` para mesclar dados em escala de forma eficiente. `MERGE INTO` combina as operações `INSERT`, `UPDATE` e `DELETE` em uma única transação. Isso reduz a sobrecarga de processamento em seu pipeline de dados e requer menos SQL para ser gravado. Para obter mais informações, consulte Atualizar dados nas tabelas Iceberg e MERGE INTO.
- Compatibilidade do CTAS e VIEW: use as instruções `CREATE TABLE AS SELECT` (CTAS) e `CREATE VIEW` com tabelas do Iceberg. Para obter mais informações, consulte CREATE TABLE AS e CREATE VIEW.

- Compatibilidade do VACUUM: é possível usar a instrução `VACUUM` para otimizar seu data lake ao excluir snapshots e dados que não são mais necessários. Você pode usar esse recurso para melhorar a performance de leitura e atender aos requisitos regulatórios, como o GDPR. Para obter mais informações, consulte Otimizar tabelas Iceberg e VACUUM.

Esses novos recursos requerem a versão 3 do mecanismo do Athena e estão disponíveis em todas as regiões em que o Athena é compatível. É possível usá-los com drivers, APIs ou com o console do Athena.

Para obter informações sobre como usar o Iceberg no Athena, consulte Usar tabelas do Apache Iceberg.

14 de novembro de 2022

Publicado em 14/11/2022

O Amazon Athena passou a ser compatível com endpoints do IPv6 para conexões de entrada que você pode usar para invocar funções do Athena usando o IPv6. É possível usar esse recurso para atender aos requisitos de conformidade do IPv6. Isso também elimina a necessidade de equipamentos de rede adicionais para tratar da conversão de endereços entre o IPv4 e o IPv6.

Para usar esse recurso, configure suas aplicações para usar os novos endpoints de pilha dupla do Athena, que são compatíveis com IPv4 e IPv6. Os endpoints de pilha dupla, usam o formato `athena.region.api.aws`. Por exemplo, o endpoint de pilha dupla na região Leste dos EUA (Norte da Virgínia) é `athena.us-east-1.api.aws`.

Quando você realiza uma solicitação para um endpoint de pilha dupla do Athena, o endpoint decide para um endereço IPv6 ou IPv4, dependendo do protocolo usado pela rede e pelo cliente. Para se conectar programaticamente a um serviço da AWS, você pode usar a AWS CLI ou o SDK da AWS para especificar um endpoint.

Para obter mais informações sobre endpoints de serviço, consulte Endpoints de serviço da AWS. Para saber mais sobre os endpoints de serviço do Athena, consulte Amazon Athena endpoints and quotas (Endpoints e cotas do Amazon Athena) na documentação da AWS.

É possível usar os novos endpoints de pilha dupla do Athena para conexões de entrada sem custos adicionais. Os endpoints de pilha dupla geralmente estão disponíveis em todas as Regiões da AWS.

11 de novembro de 2022

Publicado em 11/11/2022

Athena anuncia as correções e melhorias a seguir.

- Expansão do controle de acesso detalhado do Lake Formation: agora é possível usar políticas de controle de acesso detalhado do AWS Lake Formation nas consultas do Athena para dados armazenados em qualquer formato de arquivo ou tabela compatível. É possível usar o controle de acesso detalhado no Lake Formation para restringir o acesso aos dados nos resultados da consulta ao usar filtros de dados para obter segurança em nível de coluna, de linha e de célula. Os formatos de tabela compatíveis no Athena incluem Apache Iceberg, Apache Hudi e Apache Hive. A expansão do controle de acesso detalhado está disponível em todas as regiões compatíveis com o Athena. O suporte expandido para formatos de tabelas e arquivos requer a Mecanismo Athena versão 3, que oferece novos recursos e performance de consulta aprimorada, mas não altera a forma como você configura as políticas de controle de acesso detalhadas no Lake Formation.

  O uso dessa expansão de controle de acesso detalhado no Athena tem as seguintes considerações:

  - EXPLAIN: as informações de filtragem de linhas ou de células definidas no Lake Formation e as informações de estatísticas da consulta não são mostradas na saída do `EXPLAIN` e do `EXPLAIN ANALYZE`. Para obter informações sobre o `EXPLAIN` no Athena, consulte Usar EXPLAIN e EXPLAIN ANALYZE no Athena.
  - Metastores do Hive externos: as colunas ocultas do Apache Hive não podem ser usadas para filtragem de controle de acesso detalhado, e as tabelas de sistema ocultas do Apache Hive não tem suporte pelo controle de acesso detalhado. Para obter mais informações, consulte Considerações e limitações no tópico Usar o conector de dados do Athena para metastore externo do Hive.
  - Estatísticas de consulta: as informações referentes a contagem de linhas e ao tamanho dos dados de entrada e saída em nível de estágio não são mostradas nas estatísticas de consulta do Athena quando uma consulta tem filtros em nível de linha definidos no Lake Formation. Para obter informações sobre como visualizar estatísticas para consultas do Athena, consulte Visualizar estatísticas e detalhes de execução para consultas concluídas e GetQueryRuntimeStatistics.
  - Grupos de trabalho: os usuários do mesmo grupo de trabalho do Athena podem visualizar os dados que o controle de acesso detalhado do Lake Formation configurou para serem acessíveis ao grupo de trabalho. Para obter informações sobre como usar o Athena para consultar dados

registrados no Lake Formation, consulte Usar o Athena para consultar dados registrados com o AWS Lake Formation.

Para obter informações sobre como usar o controle de acesso detalhado no Lake Formation, consulte Gerenciar o controle de acesso detalhado usando o AWS Lake Formation no AWS Big Data Blog.

- Consulta federada do Athena: a consulta federada do Athena passou a preservar a capitalização original de nomes de campo em objetos `struct`. Anteriormente, os nomes dos campos `struct` eram automaticamente transformados em minúsculas.

8 de novembro de 2022

Publicado em 8/11/2022

Agora é possível usar o recurso de armazenamento em cache para reutilização dos resultados da consulta com a finalidade de acelerar consultas repetidas no Athena. Uma consulta repetida corresponde a uma consulta SQL idêntica a uma enviada recentemente que produz resultados semelhantes. Quando você tem necessidade de executar diversas consultas idênticas, o armazenamento em cache para reutilização de resultados pode diminuir o tempo necessário para produzir resultados. O armazenamento em cache para reutilização de resultados também reduz os custos ao reduzir o número de bytes verificados.

Para ter mais informações, consulte Reutilização dos resultados da consulta.

13 de outubro de 2022

Publicado em 13/10/2022

Athena anuncia o mecanismo Athena versão 3.

O Athena atualizou seu mecanismo de consulta SQL para incluir os recursos mais recentes do projeto de código aberto Trino. Além de oferecer suporte a todos os recursos do mecanismo Athena versão 2, o mecanismo Athena versão 3 inclui mais de 50 novas funções SQL, 30 novos recursos e mais de 90 melhorias na performance das consultas. Com o lançamento de hoje, o Athena também está disponibilizando uma abordagem de integração contínua para o gerenciamento de software de código aberto que melhora a prevalência com os projetos Trino e Presto, para que você tenha acesso mais rápido às melhorias da comunidade, integradas e ajustadas no mecanismo Athena.

Para ter mais informações, consulte Mecanismo Athena versão 3.

10 de outubro de 2022

Publicado em 10/10/2022

Athena lança o driver JDBC versão 2.0.33. O driver JDBC 2.0.33 inclui as seguintes alterações:

- Adição de nova versão do driver, versão do JDBC e propriedades de nome do plug-in à string user-agent na classe do provedor de credenciais.
- As mensagens de erro foram corrigidas e as informações necessárias foram adicionadas.
- Agora, as declarações preparadas são desalocadas se a conexão for encerrada ou se a execução da instrução preparada pelo Athena falhar.

Para obter mais informações e baixar os novos drivers, as notas de lançamento e a documentação, consulte Conectar-se ao Amazon Athena com JDBC.

23 de setembro de 2022

Publicado em 26/9/2022

Agora, o conector Amazon Athena Neptune oferece suporte à correspondência sem fazer distinção entre maiúsculas e minúsculas em nomes de colunas e tabelas.

- O conector da fonte de dados do Neptune é capaz de resolver nomes de colunas em tabelas do Neptune que usam letras maiúsculas e minúsculas, mesmo que os nomes das colunas estejam todos em minúsculas na tabela em AWS Glue. Para ativar esse comportamento, defina a variável de ambiente `enable_caseinsensitivematch` como `true` na função Lambda do conector Neptune.
- Como o AWS Glue só é compatível com nomes de tabelas em letras minúsculas, ao criar uma tabela AWS Glue para o Neptune, especifique o parâmetro `"glabel" = table_name` da tabela AWS Glue.

Para obter mais informações sobre o conector Neptune, consulte Conector do Amazon Athena para o Neptune.

13 de setembro de 2022

Publicado em 13/9/2022

Athena anuncia as correções e melhorias a seguir.

- Metastore externo do Hive: o Athena agora retorna `NULL` em vez de lançar uma exceção quando uma cláusula `WHERE` inclui uma partição que não exista em um metastore externo do Hive (EHMS). O novo comportamento corresponde ao do AWS Glue Data Catalog.
- Consultas parametrizadas: valores em consultas parametrizadas agora pode ser convertidos para o tipo de dados `DOUBLE`.
- Apache Iceberg: as operações de escrita em tabelas do Iceberg agora têm êxito quando Object Lock (Bloquear objetos) está habilitado em um bucket do Amazon S3.

31 de agosto de 2022

Publicado em 31/8/2022

O Amazon Athena anuncia a disponibilidade do Athena e de seus recursos na região Ásia-Pacífico (Jacarta).

Esta versão expande a disponibilidade do Athena nas regiões Ásia-Pacífico para incluir Ásia-Pacífico (Hong Kong), Ásia-Pacífico (Jacarta), Ásia-Pacífico (Mumbai), Ásia-Pacífico (Osaka), Ásia-Pacífico (Seul), Ásia-Pacífico (Singapura), Ásia-Pacífico (Sydney) e Ásia-Pacífico (Tóquio). Para ver uma lista completa dos Serviços da AWS disponíveis nessas e em outras regiões, consulte a Lista de serviços por Região da AWS.

23 de agosto de 2022

Publicado em 23/8/2022

A versão v2022.32.1 do SDK do Athena Query Federation contém estas alterações:

- Foi adicionado suporte ao conector de fonte de dados Amazon Athena Oracle para conexões baseadas em SSL para instâncias do Amazon RDS. O suporte é limitado ao protocolo Transport Layer Security (TLS) e à autenticação do servidor pelo cliente. Como não há suporte para autenticação mútua no Amazon RDS, a atualização não inclui suporte para autenticação mútua.

  Para ter mais informações, consulte Conector do Amazon Athena para Oracle.

3 de agosto de 2022

Publicado em 3/8/2022

Athena lança o driver JDBC versão 2.0.32. O driver JDBC 2.0.32 inclui as seguintes alterações:

- A cadeia de caracteres `User-Agent` enviada ao SDK do Athena foi estendida para conter a versão do driver, a versão da especificação JDBC e o nome do plugin de autenticação.
- Um `NullPointerException` que era lançado quando nenhum valor era fornecido para o parâmetro `CheckNonProxyHost` foi corrigido.
- Um problema com análise de `login_url` no plugin de autenticação BrowserSaml foi corrigido.
- Um problema de host do proxy que ocorria quando o parâmetro `UseProxyforIdp` era definido como `true` foi corrigido.

Para obter mais informações e baixar os novos drivers, as notas de lançamento e a documentação, consulte Conectar-se ao Amazon Athena com JDBC.

1º de agosto de 2022

Publicado em 01/08/2022

Athena anuncia melhorias no SDK do Athena Query Federation e em conectores de fonte de dados pré-criados do Athena. As melhorias incluem:

- Análise de structs: correção de um problema de análise `GlueFieldLexer` no SDK do Athena Query Federation que impedia que certos structs complicados exibissem todos os seus dados. Esse problema afetava os conectores criados no SDK do Athena Query Federation.
- Tabelas de AWS Glue: inclusão de suporte adicional para os tipos de colunas `set` e `decimal` em tabelas de AWS Glue.
- Conector DynamoDB: inclusão da capacidade de ignorar a capitalização em nomes de atributos do DynamoDB. Para obter mais informações, consulte `disable_projection_and_casing` na seção Parâmetros da página Conector do Amazon Athena para o DynamoDB.

Para obter mais informações, consulte a Versão v2022.30.2 do Athena Query Federation no GitHub

21 de julho de 2022

Publicado em 21/07/2022

Agora, é possível analisar e depurar consultas usando métricas de performance e ferramentas interativas de análise de consultas visuais no console do Athena. Os dados de performance e

os detalhes de execução de consultas podem ajudar você a identificar limitações em consultas, inspecionar os operadores e as estatísticas de cada estágio de uma consulta, acompanhar o volume de dados que fluem entre os estágios e validar o impacto de predicados de consultas. Agora é possível:

- Acesse o plano de execução distribuída e lógica da sua consulta com um único clique.
- Explore as operações em cada estágio antes que ele seja executado.
- Visualize a performance de consultas concluídas com métricas para o tempo gasto nos estágios de enfileiramento, planejamento e execução.
- Obtenha informações sobre o número de linhas e a quantidade de dados de origem processados e gerados pela consulta.
- Veja detalhes de execução granulares para suas consultas apresentados em contexto e formatados como um gráfico interativo.
- Use detalhes de execução precisos em nível de estágio para compreender o fluxo de dados na sua consulta.
- Analise dados de performance de consulta programaticamente usando novas APIs para obter estatísticas de runtime de consultas, também lançadas hoje.

Para saber como usar esses recursos nas suas consultas, assista ao tutorial em vídeo Optimize Amazon Athena Queries with New Query Analysis Tools no canal da AWS do YouTube.

Para acessar a documentação, consulte Visualizar planos de execução para consultas SQL e Visualizar estatísticas e detalhes de execução para consultas concluídas.

11 de julho de 2022

Publicado em 11/07/2022

Agora, é possível executar consultas parametrizadas diretamente do console ou da API do Athena sem preparar instruções SQL com antecedência.

Quando você executa consultas no console do Athena que têm parâmetros em forma de pontos de interrogação, a interface do usuário agora solicita que valores sejam inseridos diretamente para os parâmetros. Isso dispensa a necessidade de modificar valores literais no editor de consultas todas as vezes que você quiser executar a consulta.

Se você usar a API aprimorada de execução de consultas, agora poderá fornecer os parâmetros de execução e seus valores em uma única chamada.

Para obter mais informações, consulte Utilizar consultas parametrizadas neste guia do usuário e a postagem no blog de Big Data AWS Use Amazon Athena parameterized queries to provide data as a service.

8 de julho de 2022

Publicado em 08/07/2022

Athena anuncia as correções e melhorias a seguir.

- Correção de um problema com o tratamento de conversão de colunas `DATE` para endpoints do SageMaker (UDF) que estava causando falhas de consultas.

6 de junho de 2022

Publicado em 06/06/2022

Athena lança o driver JDBC versão 2.0.31. O driver JDBC 2.0.31 inclui as seguintes alterações:

- Problema de dependência log4j: implantação de solução para uma mensagem de erro Cannot find driver class (Não é possível localizar a classe do driver) causada por uma dependência log4j.

Para obter mais informações e baixar os novos drivers, as notas de lançamento e a documentação, consulte Conectar-se ao Amazon Athena com JDBC.

25 de maio de 2022

Publicado em 25/05/2022

Athena anuncia as correções e melhorias a seguir.

- Compatibilidade com Iceberg
  - Lançamento de compatibilidade para consultas entre regiões. Agora é possível consultar tabelas do Iceberg em uma Região da AWS diferente da Região da AWS que você está usando. Não há suporte para consultas entre regiões nas regiões da China.
  - Lançamento de compatibilidade com configuração de criptografia no lado do servidor. Agora você pode usar SSE-S3/SSE-KMS para criptografar dados de operações de gravação do Iceberg no Amazon S3.

Para obter mais informações sobre como usar o Apache Iceberg no Athena, consulte Usar tabelas do Apache Iceberg.

- Lançamento do driver JDBC 2.0.30

  O driver JDBC 2.0.30 para Athena oferece as seguintes melhorias:

  - Correção de um problema de corrida de dados que afetava instruções preparadas parametrizadas.
  - Correção de um problema de inicialização do aplicativo que ocorria em ambientes de compilação do Gradle.

  Para baixar o driver JDBC 2.0.30, ver as notas de release e a documentação, consulte Conectar-se ao Amazon Athena com JDBC.

6 de maio de 2022

Publicado em 06/05/2022

Lançamento dos drivers JDBC 2.0.29 e ODBC 1.1.17 para o Athena.

Esses drivers incluem as seguintes alterações:

- Atualização do processo de inicialização do navegador de plugins SAML.

Para obter mais informações sobre essas alterações e baixar os novos drivers, as notas de release e a documentação, consulte Conectar-se ao Amazon Athena com JDBC e Conectar-se ao Amazon Athena com ODBC.

22 de abril de 2022

Publicado em 22/04/2022

Athena anuncia as correções e melhorias a seguir.

- Correção de um problema no recurso de índices de partição e filtragem com o cache de partição que ocorria quando as seguintes condições eram atendidas:
  - A chave `partition_filtering.enabled` era definida como `true` nas propriedades da tabela do AWS Glue para uma tabela.
  - A mesma tabela era usada várias vezes com valores diferentes de filtro de partição.

21 de abril de 2022

Publicado em 21/04/2022

Agora você pode usar o Amazon Athena para executar consultas federadas em novas origens de dados, incluindo Google BigQuery, Azure Synapse e Snowflake. Os novos conectores de origem dos dados incluem:

- Azure Data Lake Storage (ADLS) Gen2
- Azure Synapse
- Cloudera Hive
- Cloudera Impala
- Google BigQuery
- Hortonworks
- Microsoft SQL Server
- Oracle
- SAP HANA (Express Edition)
- Snowflake
- Teradata

Para obter a lista completa das fontes de dados compatíveis com o Athena, consulte Conectores de fonte de dados disponíveis.

Para facilitar a navegação pelas origens disponíveis e se conectar a seus dados, agora você pode pesquisar, classificar e filtrar os conectores disponíveis em uma tela Data Sources (Fontes de dados) atualizada no console do Athena.

Para saber mais sobre como consultar fontes federadas, consulte Usar a consulta federada do Amazon Athena e Executar consultas federadas.

13 de abril de 2022

Publicado em 13/04/2022

Athena lança o driver JDBC versão 2.0.28. O driver JDBC 2.0.28 inclui as seguintes alterações:

- Suporte a JWT: agora o driver oferece suporte a tokens da Web JSON (JWT) para fins de autenticação. Para obter informações sobre o uso de JWT com o driver JDBC, consulte o guia de instalação e configuração, disponível para download na página do driver JDBC.
- Atualização das bibliotecas do Log4j: o driver JDBC agora usa as seguintes bibliotecas do Log4j:
  - Log4j-api 2.17.1 (anteriormente 2.17.0)
  - Log4j-core 2.17.1 (anteriormente 2.17.0)
  - Log4j-jcl 2.17.2
- Outras melhorias: o novo driver também inclui as seguintes melhorias e correções de erros:
  - O recurso de instruções preparadas do Athena agora está disponível usando o JDBC. Para obter informações sobre instruções preparadas, consulte Utilizar consultas parametrizadas.
  - A federação SAML para o driver JDBC do Athena agora funciona nas regiões da China.
  - Pequenas melhorias adicionais.

Para obter mais informações e baixar os novos drivers, as notas de lançamento e a documentação, consulte Conectar-se ao Amazon Athena com JDBC.

30 de março de 2022

Publicado em 30/03/2022

Athena anuncia as correções e melhorias a seguir.

- Consultas entre regiões: agora você pode usar o Athena para consultar dados localizados em um bucket do Amazon S3 entre Regiões da AWS, incluindo Ásia-Pacífico (Hong Kong), Oriente Médio (Bahrein), África (Cidade do Cabo) e Europa (Milão). Não há suporte para consultas entre regiões nas regiões da China.
  - Para obter uma lista de Regiões da AWS em que o Athena está disponível, consulte Amazon Athena endpoints and quotas (Endpoints e cotas do Amazon Athena).
  - Para obter informações sobre como habilitar uma Região da AWS que está desativada por padrão, consulte Enabling a Region (Habilitar uma região).
  - Para obter informações sobre consultas entre regiões, visite Realizar consultas entre regiões.

18 de março de 2022

Publicado em 18/03/2022

Athena anuncia as correções e melhorias a seguir.

- Filtragem dinâmica: a filtragem dinâmica foi melhorada para colunas de inteiros aplicando eficientemente o filtro a cada registro de uma tabela correspondente.
- Iceberg: corrigido um problema que causava falhas ao gravar arquivos Iceberg Parquet maiores que 2 GB.
- Saída descompactada: as instruções CREATE TABLE agora são compatíveis com a gravação de arquivos não compactados. Para gravar arquivos descompactados, use a seguinte sintaxe:
  - CREATE TABLE (arquivo de texto ou JSON): Em `TBLPROPERTIES`, especifique `write.compression = NONE`.
  - CREATE TABLE (Parquet): em `TBLPROPERTIES`, especifique `parquet.compression = UNCOMPRESSED`.
  - CREATE TABLE (ORC): em `TBLPROPERTIES`, especifique `orc.compress = NONE`.
- Compactação: corrigido um problema com inserções para tabelas de arquivos de texto que criavam arquivos compactados em um formato, mas usavam outra extensão de arquivo de formato de compactação quando métodos de compactação não padrão eram usados.
- Avro: corrigidos problemas que ocorriam ao ler decimais do tipo fixo dos arquivos Avro.

2 de março de 2022

Publicado em 02/03/2022

O Athena anuncia novos recursos e aperfeiçoamentos a seguir.

- Agora, você pode conceder ao proprietário do bucket do Amazon S3 acesso de controle total sobre os resultados das consultas quando ACLs estão habilitadas para o bucket de resultados de consultas. Para ter mais informações, consulte Especificar um local para resultados de consultas.
- Agora, é possível atualizar consultas nomeadas existentes. Para ter mais informações, consulte Usar consultas salvas.

23 de fevereiro de 2022

Publicado em 23/02/2022

Athena anuncia as seguintes correções e melhorias de performance.

- Melhorias de tratamento de memória para aumentar a performance e reduzir erros de memória.
- Agora, o Athena faz a leitura de colunas de timestamp ORC com informações de fuso horário armazenadas em rodapés de faixas e grava arquivos ORC com fuso horário (UTC) em rodapés. Isso afeta somente o comportamento de leituras de timestamp quando o arquivo ORC a ser lido foi criado em um ambiente de fuso horário não UTC.
- Correção de estimativas incorretas de tamanho de tabela de links simbólicos que resultavam em planos de consulta inferiores ao ideal.
- Exibições laterais explodidas agora podem ser consultadas no console do Athena a partir de origens de dados de metastore do Hive.
- Melhorias em mensagens de erro de leitura do Amazon S3 para incluir informações mais detalhadas sobre o Código de erro do Amazon S3.
- Correção de um problema que fazia com que arquivos de saída no formato ORC perdessem a compatibilidade com o Apache Hive 3.1.
- Correção de um problema que fazia com que nomes de tabelas com aspas falhassem em certas consultas DML e DDL.

15 de fevereiro de 2022

Publicado em 15/02/2022

Amazon Athena aumentou a cota de consultas DML ativas em todas as Regiões da AWS. Consultas ativas incluem consultas em execução e também consultas enfileiradas. Com essa alteração, agora é possível ter mais consultas DML em estado ativo do que antes.

Para obter informações sobre as cotas de serviço do Athena, consulte Service Quotas. Para as cotas de consultas na região onde você usa o Athena, consulte Endpoints e cotas do Amazon Athena na Referência geral da AWS.

Para monitorar o uso da cota, é possível usar métricas de uso do CloudWatch. O Athena publica a métrica `ActiveQueryCount` a seguir no namespace `AWS/Usage`. Para ter mais informações, consulte Monitorar as métricas de uso do Athena.

Depois de revisar o uso, será possível usar o console do Service Quotas para solicitar um aumento de cota. Se você já tiver solicitado um aumento de cota para sua conta, a cota solicitada ainda se aplicará se exceder a nova cota de consultas DML ativas padrão. Caso contrário, todas as contas usarão o novo padrão.

14 de fevereiro de 2022

Publicado em 14/02/2022

Este lançamento inclui o subcampo `ErrorType` para o objeto de resposta AthenaError na alçai de API GetQueryExecution do Athena.

Enquanto o campo `ErrorCategory` existente indica a origem geral de uma consulta com falha (sistema, usuário ou outro), o novo campo `ErrorType` fornece informações mais detalhadas sobre o erro ocorrido. Combine as informações de ambos os campos para obter insights sobre as causas da falha da consulta.

Para ter mais informações, consulte Catálogo de erros do Athena.

9 de fevereiro de 2022

Publicado em 09/02/2022

O console antigo do Athena não está mais disponível O novo console do Athena suporta todos os recursos do console anterior, mas com uma interface moderna e fácil de usar, e inclui novos recursos que melhoram a experiência de desenvolvimento de consultas, análise de dados e gerenciamento de uso. Para usar o novo console do Athena, acesse https://console.aws.amazon.com/athena/.

8 de fevereiro de 2022

Publicado em 08/02/2022

Proprietário do bucket esperado: como uma medida de segurança adicional, agora você tem a opção de especificar o ID da Conta da AWS que você espera ser a proprietária do bucket do local de saída dos resultados da consulta no Athena. Se o ID da conta do proprietário do bucket dos resultados da consulta não corresponder ao ID de conta que você especificar, as tentativas de enviar a saída para o bucket falharão com o erro de permissões do Amazon S3. Você pode fazer essa configuração no nível de cliente ou de grupo de trabalho.

Para ter mais informações, consulte Especificar um local para resultados de consultas.

28 de janeiro de 2022

Publicado em 28/01/2022

O Athena anuncia aprimoramentos a seguir nos recursos do mecanismo.

- {Apache Hudi}: agora as consultas de snapshots em tabelas Hudi Merge on Read (MoR) podem ler colunas de carimbo de data/hora com o tipo de dados `INT64`.
- Consultas UNION: melhoria de performance e redução de dados para determinadas consultas `UNION` que verificam a mesma tabela várias vezes.
- Consultas de disjunção: melhoria de performance para consultas que têm apenas valores de disjunção para cada coluna de partição no filtro.
- Melhorias na projeção de partições
  - Vários valores de disjunção agora são permitidos na condição de filtro para colunas do tipo `injected`. Para ter mais informações, consulte Tipo injected.
  - Melhoria de performance para colunas de tipos baseados em strings, como `CHAR` ou `VARCHAR` que têm apenas valores de disjunção no filtro.

13 de janeiro de 2022

Publicado em 13/01/2022

Lançamento dos drivers JDBC 2.0.27 e ODBC 1.1.15 para o Athena.

O driver JDBC 2.0.27 inclui as seguintes alterações:

- O driver foi atualizado para recuperar catálogos externos.
- O número da versão do driver estendido agora está incluído na string `user-agent` como parte da chamada de API do Athena.

O driver ODBC 1.1.15 inclui as seguintes alterações:

- Corrige um problema com as segundas chamadas para `SQLParamData()`.

Para obter mais informações sobre essas alterações e baixar os novos drivers, as notas de release e a documentação, consulte Conectar-se ao Amazon Athena com JDBC e Conectar-se ao Amazon Athena com ODBC.

Notas de release do Athena para 2021

26 de novembro de 2021

Publicado em 26/11/2021

O Athena anuncia a versão de pré-lançamento público das transações ACID do Athena, que adicionam operações de gravação, exclusão, atualização e viagem no tempo à SQL Data Manipulation Language (DML) do Athena. As transações ACID do Athena permitem que vários usuários simultâneos façam modificações confiáveis nos dados do Amazon S3 no nível da linha. Baseadas no formato de tabela Apache Iceberg, as transações ACID do Athena são compatíveis com outros serviços e mecanismos, como Amazon EMR e Apache Spark, que também suportam os formatos de tabela Iceberg.

As transações do ACID do Athena e a sintaxe SQL familiar simplificam as atualizações de dados empresariais e regulamentares. Por exemplo, para responder a uma solicitação de eliminação de dados, você pode executar uma operação `DELETE` em SQL. Para fazer correções manuais em registros, você pode usar uma única instrução `UPDATE`. Para recuperar dados que foram excluídos recentemente, você pode emitir consultas de viagem no tempo usando uma instrução `SELECT`. As transações do Athena estão disponíveis por meio do console do Athena, de operações de API e dos drivers ODBC e JDBC.

Para ter mais informações, consulte Usar transações ACID do Athena.

24 de novembro de 2021

Publicado em 24/11/2021

Athena anuncia suporte a leitura e gravação de dados nos formatos ORG, Parquet e textfile compactados como ZStandard. O Athena usa o nível 3 de compactação ZStandard ao gravar dados compactados como ZStandard.

Para obter mais informações sobre compactação de dados no Athena, consulte Suporte a compactação no Athena.

22 de novembro de 2021

Publicado em 22/11/2021

Agora é possível gerenciar fluxos de trabalho do AWS Step Functions no console do Amazon Athena, facilitando a criação de pipelines de processamento de dados escaláveis, a execução

de consultas com base em uma lógica de negócios personalizada, a automatização de tarefas administrativas e de alertas, e muito mais.

O Step Functions agora é integrado no console atualizado do Athena, e você pode usá-lo para visualizar um diagrama de fluxo de trabalho interativo de suas máquinas de estado que invocam o Athena. Para começar, selecione Workflows (Fluxos de trabalho) No painel de navegação esquerdo. Se você tiver máquinas de estado existentes com consultas do Athena, selecione uma máquina de estado para exibir um diagrama interativo do fluxo de trabalho. Se você nunca usou Step Functions antes, pode começar iniciando um projeto de amostra no console do Athena e personalizando-o para atender aos seus casos de uso.

Para obter mais informações, consulte Criar e orquestrar pipelines ETL usando o Amazon Athena e o AWS Step Functions ou consulte a documentação do Step Functions.

18 de novembro de 2021

Publicado em 18/11/2021

O Athena anuncia novos recursos e aperfeiçoamentos.

- Suporte a spill-to-disk para consultas de agregação que contêm `DISTINCT`, `ORDER BY` ou ambos, como no seguinte exemplo:

```
SELECT array_agg(orderstatus ORDER BY orderstatus)
FROM orders
GROUP BY orderpriority, custkey
```

- Problemas de tratamento de memória resolvidos para consultas que usam `DISTINCT`. Para evitar mensagens de erro como Consultar recursos esgotados nesse fator de escala quando você usas consultas `DISTINCT`, escolha colunas que tenham uma baixa cardinalidade para `DISTINCT` ou reduza o tamanho dos dados da consulta.
- Em consultas `SELECT COUNT(*)` que não especificam uma coluna, melhor performance e uso de memória, mantendo apenas a contagem, sem buffer de linhas.
- Introduzidas as funções de string a seguir.
  - `translate(source, from, to)`: retorna a string `source` com os caracteres encontrados na string `from` substituídos pelos caracteres correspondentes da string `to`. Se a string `from` contiver duplicatas, somente a primeira será usada. Se o caractere `source` não ocorrer na string `from`, o caractere `source` será copiado sem tradução. Se o índice do caractere

correspondente na string `from` for maior do que o comprimento da string `to`, o caractere será omitido da string resultante.

- `concat_ws(string0, array(varchar))`: retorna a concatenação de elementos na matriz usando `string0` como separador. Se `string0` for null, o valor retornado será nulo. Todos os valores null na matriz são ignorados.

- Corrigido um bug em que as consultas falhavam ao tentar acessar um subcampo ausente em um `struct`. As consultas agora retornam null para o subcampo ausente.
- Corrigido um problema de hashing inconsistente para o tipo de dados decimal.
- Correção de um problema que causava a exaustão de recursos quando havia muitas colunas em uma partição.

17 de novembro de 2021

Publicado em 17/11/2021

O Amazon Athena agora suporta indexação de partições para acelerar consultas em tabelas particionadas no AWS Glue Data Catalog.

Ao consultar tabelas particionadas, o Athena recupera e filtra as partições de tabela disponíveis para o subconjunto relevante para a sua consulta. À medida que novos dados e partições são adicionados, mais tempo é necessário para processar as partições e o runtime da consulta pode aumentar. Para otimizar o processamento de partições e melhorar a performance das consultas em tabelas altamente particionadas, o Athena agora suporta os índices de partição do AWS Glue.

Para ter mais informações, consulte Indexação e filtragem de partições do AWS Glue.

16 de novembro de 2021

Publicado em 16/11/2021

O novo console aprimorado do Amazon Athena agora está disponível para o público em geral nas regiões comerciais e GovCloud da AWS onde o Athena está disponível. O novo console do Athena suporta todos os recursos do console anterior, mas com uma interface moderna e fácil de usar, e inclui novos recursos que melhoram a experiência de desenvolvimento de consultas, análise de dados e gerenciamento de uso. Agora é possível:

- Reorganizar, navegar até ou fechar várias guias de consulta a partir de uma barra de guias de consulta redesenhada.

- Ler e editar consultas com mais facilidade com formatação SQL e texto aprimorada.
- Copiar os resultados da consulta para a área de transferência, além de baixar o conjunto completo de resultados.
- Classificar seu histórico de consultas, consultas salvas e grupos de trabalho, e escolher quais as colunas a serem exibidas ou ocultadas.
- Usar uma interface simplificada para configurar origens de dados e grupos de trabalho em menos cliques.
- Definir preferências para exibir resultados da consulta, histórico de consultas, quebra de linha e muito mais.
- Aumentar sua produtividade com atalhos de teclado novos e aprimorados e documentação de produto incorporada.

Com o anúncio de hoje, o console reformulado agora é o padrão. Para nos falar sobre sua experiência, escolha Feedback no canto inferior esquerdo do console.

Se desejar, você pode usar o console anterior fazendo login em seu Conta da AWS, escolhendo o Amazon Athena e desmarcando New Athena experience (Nova experiência do Athena) no painel de navegação à esquerda.

12 de novembro de 2021

Publicado em 12/11/2021

Agora você pode usar o Amazon Athena para executar consultas federadas em origens dos dados localizadas em um conta da AWS que não seja a sua. Até hoje, consultar esses dados exigia que a origem dos dados e seu conector usassem a mesma Conta da AWS que o usuário que consultou os dados.

Como administrador de dados, você pode habilitar consultas federadas entre contas compartilhando seu conector de dados com uma conta de analista de dados. Como analista de dados, você pode adicionar um conector de dados que um administrador de dados compartilhou com você à sua conta. As alterações de configuração no conector na conta de origem se aplicam automaticamente ao conector compartilhado.

Para obter informações sobre consultas federadas, consulte Habilitar consultas federadas entre contas. Para saber mais sobre como consultar fontes federadas, consulte Usar a consulta federada do Amazon Athena e Executar consultas federadas.

2 de novembro de 2021

Publicado em 02/11/2021

Agora é possível usar a instrução `EXPLAIN ANALYZE` no Athena para visualizar o plano de execução e o custo de cada operação das consultas SQL.

Para ter mais informações, consulte Usar EXPLAIN e EXPLAIN ANALYZE no Athena.

29 de outubro de 2021

Publicado em 29/10/2021

Athena lança os drivers JDBC 2.0.25 e ODBC 1.1.13, e anuncia recursos e aprimoramentos.

Drivers JDBC e ODBC

Lançamento dos drivers JDBC 2.0.25 e ODBC 1.1.13 para o Athena. Ambos os drivers oferecem suporte a autenticação multifator SAML do navegador, que pode ser configurada para funcionar com qualquer provedor de SAML 2.0.

O driver JDBC 2.0.25 inclui as seguintes alterações:

- Suporte a autenticação SAML do navegador. O driver inclui um plug-in SAML de navegador que pode ser configurado para funcionar com qualquer provedor de SAML 2.0.
- Suporte a chamadas de API do AWS Glue. Você pode usar o parâmetro `GlueEndpointOverride` para substituir o endpoint AWS Glue.
- Alterado o caminho da classe `com.simba.athena.amazonaws` para `com.amazonaws`.

O driver ODBC 1.1.13 inclui as seguintes alterações:

- Suporte a autenticação SAML do navegador. O driver inclui um plug-in SAML de navegador que pode ser configurado para funcionar com qualquer provedor de SAML 2.0. Para obter um exemplo de como usar o plugin SAML de navegador com o driver ODBC, consulte Configurar o Single Sign-On usando ODBC, SAML 2.0 e o provedor de identidade Okta.
- Agora você pode configurar a duração da sessão da função quando usa ADFS, Azure AD ou Browser Azure AD para autenticação.

Para obter mais informações sobre essas e outras alterações e baixar os novos drivers, as notas de release e a documentação, consulte Conectar-se ao Amazon Athena com JDBC e Conectar-se ao Amazon Athena com ODBC.

Recursos e melhorias

O Athena anuncia novos recursos e aperfeiçoamentos a seguir.

- Uma nova regra de otimização foi introduzida para evitar varreduras de tabelas duplicadas em determinados casos.

4 de outubro de 2021

Publicado em 04/10/2021

O Athena anuncia novos recursos e aperfeiçoamentos a seguir.

- SQL OFFSET: a cláusula SQL `OFFSET` agora é suportada nas instruções `SELECT`. Para ter mais informações, consulte SELECT.
- Métricas de uso do CloudWatch: o Athena agora publica a métrica `ActiveQueryCount` no namespace `AWS/Usage`. Para ter mais informações, consulte Monitorar as métricas de uso do Athena.
- Planejamento de consulta: corrigido um bug que, em casos raros, poderia causar o esgotamento dos tempos limites de planejamento de consulta.

16 de setembro de 2021

Publicado em 16/07/2021

O Athena anuncia os novos recursos e aperfeiçoamentos a seguir.

Atributos

- Ocorreu adição de suporte para especificar os arquivos de texto e a compactação JSON em CTAS usando a propriedade de tabela `write_compression`. Também é possível especificar a propriedade `write_compression` CTAS para os formatos Parquet e ORC. Para ter mais informações, consulte Propriedades da tabela CTAS.

- O formato de compactação BZIP2 passou a ser compatível com a gravação de arquivos de texto e arquivos JSON. Para obter mais informações sobre os formatos de compactação no Athena, consulte Suporte a compactação no Athena.

Melhorias

- Corrigido um bug no qual as informações de identidade não eram enviadas para a função UDF do Lambda.
- Corrigido um problema de pushdown de predicado com condições de filtro de disjunção.
- Corrigido um problema de hashing para tipos decimais.
- Corrigido um problema de coleta de estatísticas desnecessárias.
- Removida uma mensagem de erro inconsistente.
- Melhor performance de junção de transmissão aplicando redução dinâmica de partição no nó de processamento.
- Para consultas federadas:
  - Configuração alterada para reduzir a ocorrência de erros `CONSTRAINT_VIOLATION` em consultas federadas.

15 de setembro de 2021

Publicado em 15/09/2021

Agora você pode usar um console do Amazon Athena redesenhado (versão de demonstração). Um novo driver JDBC do Athena foi lançado.

Versão de demonstração do console do Athena

Agora é possível usar um console do Amazon Athena reformulado (versão de demonstração) em qualquer Região da AWS onde o Athena esteja disponível. O novo console suporta todos os recursos do console existente, mas em uma interface moderna e mais fácil de usar.

Para alternar para o novo console, faça login em seu Conta da AWS e escolha o Amazon Athena. Na barra de navegação do console da AWS, escolha Switch to the new console (Alternar para o novo console). Para voltar para o console padrão, desmarque New Athena experience (Nova experiência do Athena) no painel de navegação à esquerda.

Comece a usar o novo console hoje mesmo. Escolha Feedback no canto inferior esquerdo do console para nos falar sobre sua experiência.

Driver JDBC 2.0.24 do Athena

O Athena anuncia a disponibilidade do driver JDBC versão 2.0.24 para o Athena. Esta versão atualiza o suporte a proxy para todos os provedores de credenciais. O driver agora oferece suporte à autenticação de proxy para todos os hosts que não são suportados pela propriedade de conexão `NonProxyHosts`.

Para conveniência, esta versão inclui downloads do driver JDBC com e sem o SDK do AWS. Esta versão do driver JDBC permite que você tenha o AWS-SDK e o driver JDBC do Athena incorporados no projeto.

Para obter mais informações e baixar o novo driver, as notas de release e a documentação, consulte Conectar-se ao Amazon Athena com JDBC.

31 de agosto de 2021

Publicado em 31/08/2021

Athena anuncia as seguintes melhorias de recursos e correções de bugs.

- Melhorias na federação do Athena: o Athena incluiu suporte a tipos de mapas e aprimorou o suporte a tipos complexos como parte do Athena Query Federation SDK. Esta versão também inclui alguns aprimoramentos de memória e otimizações de performance.
- Novas categorias de erro: as categorias de erro `USER` e `SYSTEM` foram incluídas nas mensagens de erro. Essas categorias ajudam a distinguir entre os erros que você mesmo pode corrigir (`USER`) e os erros podem precisar da ajuda do suporte do Athena (`SYSTEM`).
- Mensagens de erro de consulta federada: as categorizações `USER_ERROR` foram atualizadas para erros relacionados à consulta federada.
- JOIN: os bugs relacionados a vazamento em disco (spill-to-disk) e os problemas de memória foram corrigidos para melhorar a performance e reduzir erros de memória nas operações `JOIN`.

12 de agosto de 2021

Publicado em 12/08/2021

Lançamento do driver ODBC 1.1.12 para Athena. Esta versão corrige problemas relacionados a `SQLPrepare()`, `SQLGetInfo()` e `EndpointOverride`.

Para baixar o novo driver, as notas de release e a documentação, consulte Conectar-se ao Amazon Athena com ODBC.

6 de agosto de 2021

Publicado em 06/08/2021

O Amazon Athena anuncia a disponibilidade do Athena e de seus recursos na região Ásia-Pacífico (Osaka).

Esta versão expande a disponibilidade do Athena nas regiões Ásia-Pacífico para incluir Ásia-Pacífico (Hong Kong), Ásia-Pacífico (Mumbai), Ásia-Pacífico (Osaka), Ásia-Pacífico (Seul), Ásia-Pacífico (Singapura), Ásia-Pacífico (Sydney) e Ásia-Pacífico (Tóquio). Para ver uma lista completa dos Serviços da AWS disponíveis nessas e em outras regiões, consulte a Lista de serviços por Região da AWS.

5 de agosto de 2021

Publicado em 05/08/2021

Você pode usar a instrução `UNLOAD` para gravar a saída de uma consulta `SELECT` nos formatos PARQUET, ORC, AVRO e JSON.

Para ter mais informações, consulte UNLOAD.

30 de julho de 2021

Publicado em 30/07/2021

Athena anuncia as seguintes melhorias de recursos e correções de bugs.

- Filtragem dinâmica e redução de partições: as melhorias aumentam a performance e reduzem a quantidade de dados verificados em determinadas consultas, como no exemplo a seguir.

  Este exemplo assume que `Table_B` é uma tabela não particionada com tamanhos de arquivos que somam menos de 20 MB. Para consultas como esta, menos dados são lidos em `Table_A` e a consulta é concluída com mais rapidez.

```
SELECT *
```

```
FROM Table_A
JOIN Table_B ON Table_A.date = Table_B.date
WHERE Table_B.column_A = "value"
```

- ORDER BY com LIMIT, DISTINCT com LIMIT: melhorias na performance de consultas que usam `ORDER BY` ou `DISTINCT` seguido de uma cláusula `LIMIT`.
- Arquivos do S3 Glacier Deep Archive: quando o Athena consulta uma tabela que contém uma mistura de arquivos do S3 Glacier Deep Archive e arquivos que não são do S3 Glacier, o Athena agora ignora os arquivos do S3 Glacier Deep Archive para você. Antes, era necessário mover manualmente esses arquivos do local da consulta para evitar falha. Se você deseja usar o Athena para consultar objetos no armazenamento do S3 Glacier Deep Archive, deve restaurá-los. Para obter mais informações, consulte Restaurar objetos arquivados no Manual do usuário do Amazon S3.
- Um bug foi corrigido em que arquivos vazios criados pela propriedade de tabela `bucketed_by` de CTAS não eram criptografados corretamente.

21 de julho de 2021

Publicado em 21/07/2021

Com o lançamento de julho de 2021 do Microsoft Power BI Desktop, você pode criar relatórios e painéis usando um conector de origem dos dados nativo para o Amazon Athena. O conector para o Amazon Athena está disponível como um conector padrão no Power BI, é compatível com DirectQuery e permite a análise de grandes conjuntos de dados e a atualização de conteúdo por meio do Power BI Gateway.

Como o conector usa o nome da origem dos dados ODBC existente (DSN) para se conectar e executar consultas no Athena, ele requer o driver ODBC do Athena. Para baixar o driver ODBC mais recente, consulte Conectar-se ao Amazon Athena com ODBC.

Para ter mais informações, consulte Usar o conector do Power BI para Amazon Athena.

16 de julho de 2021

Publicado em 16/07/2021

Amazon Athena atualizou sua integração com o Apache Hudi. O Hudi é um framework de gerenciamento de dados de código aberto que simplifica o processamento incremental de dados em data lakes do Amazon S3. A integração atualizada permite que você use o Athena para consultar

tabelas do Hudi 0.8.0 gerenciadas pelo Amazon EMR, Apache Spark, Apache Hive ou outros serviços compatíveis. Além disso, o Athena agora inclui dois recursos adicionais: consultas de snapshot em tabelas Merge-on-Read (MoR – Mesclar na leitura) e suporte para leitura em tabelas de bootstrap.

O Apache Hudi oferece processamento de dados no nível do registro que pode ajudar você a simplificar o desenvolvimento de pipelines de Change Data Capture (CDC – Captura de dados de alteração), cumprir as atualizações e exclusões orientadas pelo RGPD e gerenciar melhor os dados de streaming de sensores ou dispositivos que exigem inserção de dados e atualizações de eventos. A versão 0.8.0 facilita a migração de tabelas grandes do Parquet para o Hudi sem copiar os dados para que você possa consultá-las e analisá-las usando o Athena. Você pode usar o novo suporte do Athena para consultas de snapshot para obter visualizações quase em tempo real das suas atualizações de tabela de transmissão.

Para saber mais como usar o Hudi com o Athena, consulte Usar o Athena para consultar conjuntos de dados do Apache Hudi.

8 de julho de 2021

Publicado em 08/07/2021

Lançamento do driver ODBC 1.1.11 para Athena. O driver ODBC agora pode autenticar a conexão usando um JSON Web Token (JWT). No Linux, o valor padrão da propriedade Workgroup (Grupo de trabalho) foi definido como Primary (Primário).

Para obter mais informações e baixar o novo driver, as notas de release e a documentação, consulte Conectar-se ao Amazon Athena com ODBC.

1º de julho de 2021

Publicado em 01/07/2021

Em 1º de julho de 2021, o processamento especial dos grupos de trabalho de previsualização foi encerrado. Os grupos de trabalho `AmazonAthenaPreviewFunctionality` mantêm o nome, mas não têm mais um status especial. Você pode continuar usando os grupos de trabalho `AmazonAthenaPreviewFunctionality` para visualizar, modificar, organizar e executar consultas. No entanto, as consultas que usam recursos que antes estavam em previsualização agora estão sujeitas aos termos e condições de cobrança padrão do Athena. Para obter informações, consulte Preços do Amazon Athena.

23 de junho de 2021

Publicado em 23/06/2021

Lançamento dos drivers JDBC 2.0.23 e ODBC 1.1.10 para o Athena. Os dois drivers oferecem melhor performance de leitura e aceitam as instruções EXPLAIN e as consultas parametrizadas.

As instruções `EXPLAIN` mostram o plano de execução lógico ou distribuído de uma consulta SQL. As consultas parametrizadas permitem que a mesma consulta seja usada várias vezes com valores diferentes fornecidos no runtime.

A versão do JDBC também inclui suporte para os Serviços de Federação do Active Directory 2019 e uma opção de substituição de endpoint personalizada para AWS STS. A versão do ODBC corrige um problema com as credenciais de perfil do IAM.

Para obter mais informações e baixar os novos drivers, as notas de release e a documentação, consulte Conectar-se ao Amazon Athena com JDBC e Conectar-se ao Amazon Athena com ODBC.

12 de maio de 2021

Publicado em 12/05/2021

Agora você pode usar o Amazon Athena para registrar um catálogo do AWS Glue de uma conta diferente da sua. Depois de configurar as permissões do IAM necessárias para o AWS Glue, você poderá usar o Athena para executar consultas entre contas.

Para obter mais informações, consulte Registrar um AWS Glue Data Catalog de outra conta e Acesso aos catálogos de dados do AWS Glue entre contas.

10 de maio de 2021

Publicado em 10/05/2021

Lançamento do driver ODBC versão 1.1.9.1001 para o Athena. Esta versão corrige um problema com o tipo de autenticação `BrowserAzureAD` ao usar o Azure Active Directory (AD).

Para baixar os novos drivers, as notas de release e a documentação, consulte Conectar-se ao Amazon Athena com ODBC.

5 de maio de 2021

Publicado em 05/05/2021

Agora você pode usar o conector do Vertica para Amazon Athena em consultas federadas para consultar origens de dados do Vertica usando o Athena. Por exemplo, você pode executar consultas analíticas em um data warehouse no Vertica e um data lake no Amazon S3.

Para implantar o conector do Vertica para Athena, visite a página AthenaVerticaConnector no AWS Serverless Application Repository.

O conector do Vertica para Amazon Athena expõe várias opções de configuração por meio das variáveis de ambiente do Lambda. Para obter informações sobre as opções de configuração, os parâmetros, as strings de conexão, a implantação e as limitações, consulte Conector do Amazon Athena para o Vertica.

Para obter informações detalhadas sobre como usar o conector do Vertica, veja Consultar uma origem dos dados do Vertica no Amazon Athena usando o Athena Federated Query SDK (em inglês) no blog de big data da AWS.

30 de abril de 2021

Publicado em 30/04/2021

Lançamento dos drivers JDBC 2.0.21 e ODBC 1.1.9 para o Athena. As duas versões aceitam a autenticação SAML com o Azure Active Directory (AD) e a autenticação SAML com o PingFederate. A versão do JDBC também aceita consultas parametrizadas. Para obter informações sobre consultas parametrizadas no Athena, leia Utilizar consultas parametrizadas.

Para baixar os novos drivers, as notas de release e a documentação, consulte Conectar-se ao Amazon Athena com JDBC e Conectar-se ao Amazon Athena com ODBC.

29 de abril de 2021

Publicado em 29/04/2021

O Amazon Athena anuncia a disponibilidade do mecanismo do Athena versão 2 nas regiões China (Pequim) e China (Ningxia).

Para obter informações sobre o mecanismo do Athena versão 2, consulte Mecanismo do Athena versão 2.

26 de abril de 2021

Publicado em 26/04/2021

As funções de valor de janela no mecanismo do Athena versão 2 agora aceitam `IGNORE NULLS` e `RESPECT NULLS`.

Para obter mais informações, consulte Funções de valor na documentação do Presto.

21 de abril de 2021

Publicado em 21/04/2021

O Amazon Athena anuncia a disponibilidade do mecanismo do Athena versão 2 nas regiões Europa (Milão) e África (Cidade do Cabo).

Para obter informações sobre o mecanismo do Athena versão 2, consulte Mecanismo do Athena versão 2.

5 de abril de 2021

Publicado em 05/04/2021

Instrução EXPLAIN

Agora é possível usar a instrução `EXPLAIN` no Athena para visualizar o plano de execução das suas consultas SQL.

Para obter mais informações, consulte Usar EXPLAIN e EXPLAIN ANALYZE no Athena e Noções básicas sobre os resultados da instrução EXPLAIN do Athena.

Modelos de machine learning do SageMaker em consultas SQL

A inferência do modelo de machine learning com o Amazon SageMaker agora está em disponibilidade geral para o Amazon Athena. Use os modelos de machine learning em consultas SQL para simplificar tarefas complexas, como detecção de anomalias, análise de coorte de clientes e previsões de série temporal, invocando uma função em uma consulta SQL.

Para ter mais informações, consulte Usar Machine Learning (ML) com o Amazon Athena.

Funções definidas pelo usuário (UDFs)

As funções definidas pelo usuário (UDFs) agora estão em disponibilidade geral para o Athena. Use as UDFs para aproveitar as funções personalizadas que processam registros ou grupos de registros em uma única consulta SQL.

Para ter mais informações, consulte Executar consultas com funções definidas pelo usuário.

30 de março de 2021

Publicado em 30/03/2021

O Amazon Athena anuncia a disponibilidade do mecanismo do Athena versão 2 nas regiões Ásia-Pacífico (Hong Kong) e Oriente Médio (Bahrein).

Para obter informações sobre o mecanismo do Athena versão 2, consulte Mecanismo do Athena versão 2.

25 de março de 2021

Publicado em 25/03/2021

O Amazon Athena anuncia a disponibilidade do mecanismo do Athena versão 2 na região Europa (Estocolmo).

Para obter informações sobre o mecanismo do Athena versão 2, consulte Mecanismo do Athena versão 2.

5 de março de 2021

Publicado em 05/03/2021

O Amazon Athena anuncia a disponibilidade do mecanismo do Athena versão 2 nas regiões Canadá (Central), Europa (Frankfurt) e América do Sul (São Paulo).

Para obter informações sobre o mecanismo do Athena versão 2, consulte Mecanismo do Athena versão 2.

25 de fevereiro de 2021

Publicado em 25/02/2021

O Amazon Athena anuncia a disponibilidade geral do mecanismo do Athena versão 2 nas regiões Ásia-Pacífico (Seul), Ásia-Pacífico (Singapura), Ásia-Pacífico (Sydney), Europa (Londres) e Europa (Paris).

Para obter informações sobre o mecanismo do Athena versão 2, consulte Mecanismo do Athena versão 2.

Notas de release do Athena para 2020

16 de dezembro de 2020

Publicado em 16/12/2020

O Amazon Athena anuncia a disponibilidade do mecanismo do Athena versão 2, da consulta federada do Athena e do AWS PrivateLink em mais regiões.

Mecanismo do Athena versão 2 e consulta federada do Athena

O Amazon Athena anuncia a disponibilidade geral do mecanismo do Athena versão 2 e da consulta federada do Athena nas regiões Ásia-Pacífico (Mumbai), Ásia-Pacífico (Tóquio), Europa (Irlanda) e Oeste dos EUA (Norte da Califórnia). O mecanismo do Athena versão 2 e as consultas federadas já estão disponíveis nas regiões Leste dos EUA (Norte da Virgínia), Leste dos EUA (Ohio) e Oeste dos EUA (Oregon).

Para obter mais informações, consulte Mecanismo do Athena versão 2 e Usar a consulta federada do Amazon Athena.

AWS PrivateLink

O AWS PrivateLink para o Athena agora está disponível na região Europa (Estocolmo). Para obter informações sobre o AWS PrivateLink para o Athena, consulte Conectar-se ao Amazon Athena usando um endpoint da VPC de interface.

24 de novembro de 2020

Publicado em 24/11/2020

Lançamento dos drivers JDBC 2.0.16 e ODBC 1.1.6 para o Athena. Essas versões aceitam a Multifactor Authentication (MFA – Autenticação multifator) do Okta Verify no nível da conta. Você também pode usar a MFA do Okta para configurar a autenticação SMS e do Google Authenticator como fatores.

Para baixar os novos drivers, as notas de release e a documentação, consulte Conectar-se ao Amazon Athena com JDBC e Conectar-se ao Amazon Athena com ODBC.

11 de novembro de 2020

Publicado em 11/11/2020

O Amazon Athena anuncia a disponibilidade geral nas regiões Leste dos EUA (Norte da Virgínia), Leste dos EUA (Ohio) e Oeste dos EUA (Oregon) para o mecanismo do Athena versão 2 e as consultas federadas.

Mecanismo do Athena versão 2

O Amazon Athena anuncia a disponibilidade geral de uma nova versão do mecanismo de consulta, o mecanismo do Athena versão 2, nas regiões Leste dos EUA (Norte da Virgínia), Leste dos EUA (Ohio) e Oeste dos EUA (Oregon).

O mecanismo do Athena versão 2 inclui melhorias na performance e novas funcionalidades de recursos, como suporte à evolução de esquema para dados no formato Parquet, funções geoespaciais adicionais, suporte para leitura de esquema aninhado para reduzir custos e melhorias na performance das operações JOIN e AGGREGATE.

- Para obter informações sobre as melhorias, as alterações mais recentes e as correções de bugs, consulte Mecanismo do Athena versão 2.
- Para obter informações sobre como fazer upgrade, consulte Alterar versões do mecanismo do Athena.
- Para obter informações sobre teste de consultas, consulte Testar as consultas antes do upgrade da versão do mecanismo.

Consultas SQL federadas

Agora é possível usar a consulta federada do Athena nas regiões Leste dos EUA (Norte da Virgínia), Leste dos EUA (Ohio) e Oeste dos EUA (Oregon) sem usar o grupo de trabalho `AmazonAthenaPreviewFunctionality`.

Use consultas SQL federadas para executar consultas SQL em fontes de dados relacionais, não relacionais, de objeto e personalizadas. Graças às consultas federadas, é possível enviar uma única consulta SQL que verifica dados de várias fontes em execução on-premises ou hospedadas na nuvem.

A execução de análises em dados distribuídos entre aplicativos pode ser complexa e demorada pelos seguintes motivos:

- Os dados necessários para realizar análises costumam estar distribuídos em armazenamentos de dados relacionais, documentais, gráficos, na memória, de busca, de objetos, com séries temporais, com valores-chave e livros contábeis.

- Para analisar dados de todas essas fontes, são criados pipelines complexos para extrair, transformar e carregar dados em um data warehouse, para que eles possam ser consultados.
- Para acessar dados de várias fontes, é necessário aprender novas linguagens de programação e construções de acesso de dados.

As consultas SQL federadas no Athena eliminam essa complexidade porque permitem que os usuários consultem os dados no local onde quer que eles residam. Os analistas podem usar as construções SQL conhecidas para fazer `JOIN` dos dados em várias origens para análises rápidas e armazenar os resultados no Amazon S3 para uso futuro.

Conectores de fontes de dados

Para processar as consultas federadas, o Athena usa seus conectores de origem dos dados executados no AWS Lambda. Os conectores predefinidos de código aberto a seguir foram escritos e testados pelo Athena. Use-os para executar consultas SQL no Athena em suas origens de dados correspondentes.

- CloudWatch
- Métricas do CloudWatch
- DocumentDB
- DynamoDB
- OpenSearch
- HBase
- Neptune
- Redis
- Timestream
- TPC Benchmark DS (TPC-DS)

Conectores de fontes de dados personalizados

Com o uso do Athena Query Federation SDK, os desenvolvedores podem criar conectores para qualquer origem dos dados para que o Athena possa executar consultas SQL nela. O conector do Athena Query Federation oferece mais benefícios de consultas federadas do que os conectores disponíveis da AWS. Como os conectores são executados no AWS Lambda, não é necessário gerenciar a infraestrutura nem planejar o dimensionamento para atender às demandas máximas.

Próximos Passos

- Para saber mais sobre o recurso de consulta federada, acesse Usar a consulta federada do Amazon Athena.
- Para começar a usar um conector existente, consulte Implantar um conector e conectar a uma fonte de dados.
- Para saber como criar o próprio conector de origem dos dados usando o Athena Query Federation SDK, consulte Exemplo de conector do Athena no GitHub.

22 de outubro de 2020

Publicado em 22/10/2020

Agora é possível chamar o Athena com o AWS Step Functions. O AWS Step Functions pode controlar determinados Serviços da AWS diretamente no Amazon States Language. Você pode usar o Step Functions com o Athena para iniciar e interromper a execução da consulta, acessar os resultados da consulta, executar consultas de dados específicas ou agendadas e recuperar os resultados de data lakes no Amazon S3.

Para obter mais informações, consulte Chamar o Athena com o Step Functions no Guia do desenvolvedor do AWS Step Functions.

29 de julho de 2020

Publicado em 29/07/2020

Lançamento do driver JDBC versão 2.0.13. Essa versão permite usar vários catálogos de dados registrados no Athena, o serviço Okta para autenticação e conexões com endpoints da VPC.

Para baixar e usar a nova versão do driver, consulte Conectar-se ao Amazon Athena com JDBC.

9 de julho de 2020

Publicado em 09/07/2020

O Amazon Athena inclui suporte para consultar conjuntos de dados compactados do Hudi e o recurso `AWS::Athena::DataCatalog` do AWS CloudFormation para criar, atualizar ou excluir catálogos de dados registrados no Athena.

Consultar conjuntos de dados do Apache Hudi

O Apache Hudi é um framework de gerenciamento de dados de código aberto que simplifica o processamento incremental de dados. O Amazon Athena agora permite a consulta da visualização otimizada para leitura de um conjunto de dados do Apache Hudi em seu data lake baseado no Amazon S3.

Para ter mais informações, consulte Usar o Athena para consultar conjuntos de dados do Apache Hudi.

Recurso de catálogo de dados do AWS CloudFormation

Para usar o recurso de consulta federada do Amazon Athena para consultar qualquer origem dos dados, você deve primeiro registrar seu catálogo de dados no Athena. Agora é possível usar o recurso `AWS::Athena::DataCatalog` do AWS CloudFormation para criar, atualizar ou excluir catálogos de dados registrados no Athena.

Para obter mais informações, consulte AWS::Athena::DataCatalog no Manual do usuário do AWS CloudFormation.

1 de junho de 2020

Publicado em 01-06-2020

Usar o metastore do Apache Hive como um metacatálogo com o Amazon Athena

Agora é possível conectar o Athena a um ou mais metastores do Apache Hive, além do AWS Glue Data Catalog com o Athena.

Para se conectar a um metastore do Hive auto-hospedado, você precisa de um conector de metastore do Athena Hive. O Athena inclui um conector de implementação de referência que você pode usar. O conector é executado na sua conta como uma função do AWS Lambda.

Para ter mais informações, consulte Usar o conector de dados do Athena para metastore externo do Hive.

21 de maio de 2020

Publicado em 21/05/2020

O Amazon Athena oferece suporte para projeção de partições. Use a projeção de partições para acelerar o processamento de consultas de tabelas altamente particionadas e automatizar o gerenciamento de partições. Para ter mais informações, consulte Projeção de partições com o Amazon Athena.

1 de abril de 2020

Publicado em 01-04-2020

Além da região Leste dos EUA (Norte da Virgínia), os recursos de consulta federada do Amazon Athena, funções definidas pelo usuário (UDFs), inferência de machine learning e metastore externo do Hive agora estão disponíveis em versão de demonstração nas regiões Ásia-Pacífico (Mumbai), Europa (Irlanda) e Oeste dos EUA (Oregon).

11 de março de 2020

Publicado em 11-03-2020

O Amazon Athena agora publica os eventos do Amazon EventBridge para as transições de estado das consultas. Quando uma consulta faz a transição entre estados, por exemplo, de Em execução para um estado terminal, como Com êxito ou Cancelada, o Athena publica um evento de alteração de estado da consulta no EventBridge. O evento contém informações sobre a transição do estado da consulta. Para ter mais informações, consulte Monitorar consultas com eventos do Amazon EventBridge.

6 de março de 2020

Publicado em 06-03-2020

Agora você pode criar e atualizar grupos de trabalho do Amazon Athena usando o recurso `AWS::Athena::WorkGroup` do AWS CloudFormation. Para obter mais informações, consulte AWS::Athena::WorkGroup no Guia do usuário do AWS CloudFormation.

Notas de release do Athena para 2019

26 de novembro de 2019

Publicado em 17-12-2019

O Amazon Athena inclui suporte para executar consultas SQL em origens de dados relacionais, não relacionais, de objeto e personalizadas, invocar modelos de machine learning em consultas SQL e funções definidas pelo usuário (UDFs) (previsualização), usar o metastore do Apache Hive como um catálogo de metadados com o Amazon Athena (previsualização) e mais quatro métricas relacionadas a consulta.

Consultas SQL federadas

Use consultas SQL federadas para executar consultas SQL em fontes de dados relacionais, não relacionais, de objeto e personalizadas.

Agora é possível usar a consulta federada do Athena para verificar os dados armazenados em origens de dados relacionais, não relacionais, de objeto e personalizadas. Graças às consultas federadas, é possível enviar uma única consulta SQL que verifica dados de várias fontes em execução on-premises ou hospedadas na nuvem.

A execução de análises em dados distribuídos entre aplicativos pode ser complexa e demorada pelos seguintes motivos:

- Os dados necessários para realizar análises costumam estar distribuídos em armazenamentos de dados relacionais, documentais, gráficos, na memória, de busca, de objetos, com séries temporais, com valores-chave e livros contábeis.
- Para analisar dados de todas essas fontes, são criados pipelines complexos para extrair, transformar e carregar dados em um data warehouse, para que eles possam ser consultados.
- Para acessar dados de várias fontes, é necessário aprender novas linguagens de programação e construções de acesso de dados.

As consultas SQL federadas no Athena eliminam essa complexidade porque permitem que os usuários consultem os dados no local onde quer que eles residam. Os analistas podem usar as construções SQL conhecidas para fazer `JOIN` dos dados em várias origens para análises rápidas e armazenar os resultados no Amazon S3 para uso futuro.

Conectores de fontes de dados

O Athena processa as consultas federadas usando seus conectores de origem dos dados executados em AWS Lambda. Use esses conectores de origem dos dados de código aberto para executar consultas SQL federadas no Athena no Amazon DynamoDB, no Apache HBase, no Amazon Document DB, no Amazon CloudWatch, nas métricas do Amazon CloudWatch e nos bancos

de dados relacionais compatíveis com JDBC, como MySQL e PostgreSQL, com a licença do Apache 2.0.

Conectores de fontes de dados personalizados

Com o uso do Athena Query Federation SDK, os desenvolvedores podem criar conectores para qualquer origem dos dados para que o Athena possa executar consultas SQL nela. O conector do Athena Query Federation oferece mais benefícios de consultas federadas do que os conectores disponíveis da AWS. Como os conectores são executados no AWS Lambda, não é necessário gerenciar a infraestrutura nem planejar o dimensionamento para atender às demandas máximas.

Disponibilidade de visualização

A consulta federada do Athena está disponível em previsualização na região Leste dos EUA (Norte da Virgínia).

Próximos Passos

- Para iniciar sua previsualização, siga as instruções nas Perguntas frequentes sobre recursos em previsualização do Athena.
- Para saber mais sobre o recurso de consulta federada, veja Usar a consulta federada do Amazon Athena (previsualização).
- Para começar a usar um conector existente, consulte Implantar um conector e conectar a uma fonte de dados.
- Para saber como criar o próprio conector de origem dos dados usando o Athena Query Federation SDK, consulte Exemplo de conector do Athena no GitHub.

Chamar modelos de machine learning em consultas SQL

Agora é possível invocar modelos de machine learning para inferência diretamente das suas consultas do Athena. A capacidade de usar modelos de machine learning em consultas SQL simplifica tarefas complexas como detecção de anomalias, análise de coorte de clientes e previsões de vendas, para que sejam tão simples quanto invocar uma função em uma consulta SQL.

Modelos de ML

É possível usar mais de dez algoritmos de machine learning incorporados disponíveis no Amazon SageMaker, treinar seus próprios modelos ou encontrar e assinar pacotes de modelos no AWS

Marketplace e implantá-los nos serviços de hospedagem do Amazon SageMaker. Não há necessidade de configurações adicionais. Você pode invocar esses modelos de ML em suas consultas SQL usando o console, as APIs e o driver JDBC em pré-visualização do Athena.

Disponibilidade de visualização

A funcionalidade de ML do Athena está disponível hoje em pré-visualização na região Leste dos EUA (Norte da Virgínia).

Próximos Passos

- Para iniciar sua previsualização, siga as instruções nas Perguntas frequentes sobre recursos em previsualização do Athena.
- Para saber mais sobre o recurso de machine learning, consulte Usar machine learning (ML) com Amazon Athena (previsualização).

Funções definidas pelo usuário (UDFs) (visualização)

Agora você pode escrever funções escalares personalizadas e chamá-las em consultas do Athena. É possível escrever as UDFs em Java usando o Athena Query Federation SDK. Quando uma UDF é usada em uma consulta SQL enviada para o Athena, ela é chamada e executada no AWS Lambda. UDFs podem ser usados em cláusulas `SELECT` e `FILTER` de uma consulta SQL. E possível chamar vários UDFs na mesma consulta.

Disponibilidade de visualização

A funcionalidade de UDF do Athena está disponível no modo de previsualização na região Leste dos EUA (Norte da Virgínia).

Próximos Passos

- Para iniciar sua previsualização, siga as instruções nas Perguntas frequentes sobre recursos em previsualização do Athena.
- Para saber mais, consulte Consultar com funções definidas pelo usuário (visualização).
- Para ver exemplos de implementação de UDF, consulte Conector de UDF do Amazon Athena no GitHub.
- Para saber como escrever as próprias funções usando o Athena Query Federation SDK, consulte Criar e implantar uma UDF usando o Lambda.

Usar o metastore do Apache Hive como um metacatálogo com o Amazon Athena (previsualização)

Agora é possível conectar o Athena a um ou mais metastores do Apache Hive, além do AWS Glue Data Catalog com o Athena.

Conector de metastore

Para se conectar a um metastore do Hive auto-hospedado, você precisa de um conector de metastore do Athena Hive. O Athena inclui um conector de implementação de referência que você pode usar. O conector é executado na sua conta como uma função do AWS Lambda. Para obter mais informações, consulte Usar o conector de dados do Athena para metastore externo do Hive (previsualização).

Disponibilidade de visualização

O recurso de metastore do Hive está disponível no modo de previsualização na região Leste dos EUA (Norte da Virgínia).

Próximos Passos

- Para iniciar sua previsualização, siga as instruções nas Perguntas frequentes sobre recursos em previsualização do Athena.
- Para saber mais sobre esse recurso, acesse Usar o conector de dados do Athena para metastore externo do Hive (previsualização).

Novas métricas relacionadas à consulta

O Athena agora publica métricas de consulta adicionais que podem ajudar você a saber como está a performance do Amazon Athena. O Athena publica métricas relacionadas à consulta no Amazon CloudWatch. Nesta versão, o Athena publica as seguintes métricas de consulta adicionais:

- Query Planning Time (Tempo de planejamento da consulta): o tempo necessário para planejar a consulta. Isso inclui o tempo gasto recuperando partições de tabela da fonte de dados.
- Query Queuing Time (Tempo de fila da consulta): o tempo que a consulta ficou na fila aguardando recursos.
- Service Processing Time (Tempo de processamento do serviço): o tempo que levou para gravar os resultados após a conclusão do processamento do mecanismo de consulta.
- Rntime totale: o tempo necessário para o Athena executar a consulta.

Para consumir essas novas métricas de consulta, é possível criar painéis personalizados, definir alarmes e acionadores baseados em métricas no CloudWatch ou usar painéis já preenchidos diretamente no console do Athena.

Próximos Passos

Para obter mais informações, leia Monitorar consultas do Athena com as métricas do CloudWatch.

12 de novembro de 2019

Publicado em 17-12-2019

O Amazon Athena já está disponível na região Oriente Médio (Bahrein).

8 de novembro de 2019

Publicado em 17-12-2019

O Amazon Athena já está disponível nas regiões Oeste dos EUA (Norte da Califórnia) e Europa (Paris).

8 de outubro de 2019

Publicado em 17-12-2019

O Amazon Athena agora permite que você se conecte diretamente ao Athena por um endpoint da VPC de interface em sua Virtual Private Cloud (VPC). Com esse recurso, é possível enviar suas consultas ao Athena com segurança, sem a necessidade de um gateway da Internet na VPC.

Para criar um endpoint da VPC de interface para se conectar ao Athena, é possível usar o AWS Management Console ou a AWS Command Line Interface (AWS CLI). Para obter informações sobre como criar um endpoint de interface, consulte Criar um endpoint de interface.

Ao usar um endpoint da VPC de interface, a comunicação entre a VPC e as APIs do Athena fica protegida e permanece dentro da rede da AWS. Não há custos adicionais do Athena para usar esse recurso. São aplicáveis taxas de VPC endpoint de interface.

Para saber mais sobre esse recurso, consulte Conectar-se ao Amazon Athena usando um endpoint da VPC de interface.

19 de setembro de 2019

Publicado em 17-12-2019

O Amazon Athena inclui suporte para inserir novos dados em uma tabela existente usando a instrução `INSERT INTO`. É possível inserir linhas em uma tabela de destino com base em uma instrução de consulta `SELECT` executada em uma tabela de origem ou em um conjunto de valores informados como parte da instrução de consulta. Os formatos de dados compatíveis incluem arquivos Avro, JSON, ORC, Parquet e de texto.

As instruções `INSERT INTO` também podem ajudar a simplificar processos de ETL. Por exemplo, é possível usar `INSERT INTO` em uma única consulta para selecionar dados de uma tabela de origem no formato JSON e gravá-los em uma tabela de destino no formato Parquet.

As instruções `INSERT INTO` são cobradas conforme o número de bytes que são verificados na fase `SELECT`, de modo similar à cobrança das consultas `SELECT` do Athena. Para obter mais informações, consulte Preços do Amazon Athena.

Para obter mais informações sobre como usar `INSERT INTO`, incluindo formatos compatíveis, SerDes e exemplos, consulte INSERT INTO no Manual do usuário do Athena.

12 de setembro de 2019

Publicado em 17-12-2019

Agora o Amazon Athena está disponível na região da Ásia-Pacífico (Hong Kong).

16 de agosto de 2019

Publicado em 17-12-2019

O Amazon Athena inclui suporte para consulta de dados em buckets de pagamento a cargo do solicitante do Amazon S3.

Quando um bucket do Amazon S3 é configurado como pagamento a cargo do solicitante, o solicitante, não o proprietário do bucket, paga pelos custos de solicitação e de transferência de dados do Amazon S3. No Athena, os administradores de grupo de trabalho agora podem definir as configurações do grupo de trabalho para permitir que seus membros consultem os buckets de pagamento a cargo do solicitante do S3.

Para obter informações sobre como definir a configuração de pagamento a cargo do solicitante em seu grupo de trabalho, consulte Criar um grupo de trabalho no Manual do usuário do Amazon Athena. Para obter mais informações, consulte Buckets de pagamento a cargo do solicitante no Guia do desenvolvedor do Amazon Simple Storage Service.

9 de agosto de 2019

Publicado em 17-12-2019

O Amazon Athena agora permite a aplicação de políticas do AWS Lake Formation para controle de acesso refinado a bancos de dados, tabelas e colunas novos ou existentes definidos no AWS Glue Data Catalog para dados armazenados no Amazon S3.

Você pode usar esse recurso nas seguintes Regiões da AWS: Leste dos EUA (Ohio), Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico (Tóquio) e Europa (Irlanda). Não há cobranças adicionais por esse recurso.

Para obter mais informações sobre o uso desse recurso, consulte Usar o Athena para consultar dados registrados com o AWS Lake Formation. Para obter mais informações sobre o AWS Lake Formation, consulte AWS Lake Formation.

26 de junho de 2019

O Amazon Athena já está disponível na região Europa (Estocolmo). Para obter uma lista de regiões e endpoints compatíveis, consulte Regiões da AWS e endpoints.

24 de maio de 2019

Publicado em 24-05-2019

O Amazon Athena já está disponível nas regiões AWS GovCloud (EUA-Leste) e AWS GovCloud (EUA-Oeste). Para obter uma lista de regiões e endpoints compatíveis, consulte Regiões da AWS e endpoints.

05 de março de 2019

Publicado em 05/03/2019

O Amazon Athena já está disponível na região Canadá (Central). Para obter uma lista de regiões e endpoints compatíveis, consulte Regiões da AWS e endpoints. A nova versão do driver ODBC foi lançada com suporte a grupos de trabalho do Athena. Para obter mais informações, consulte as Notas de release do driver ODBC.

Para fazer download do driver ODBC versão 1.0.5 e da respectiva documentação, consulte Conectar-se ao Amazon Athena com ODBC. Para obter mais informações sobre essa versão, consulte as Notas de release do driver ODBC.

Para usar grupos de trabalho com o driver ODBC, defina a nova propriedade de conexão, `Workgroup`, na string de conexão, conforme mostrado no exemplo a seguir:

```
Driver=Simba Athena ODBC
 Driver;AwsRegion=[Region];S3OutputLocation=[S3Path];AuthenticationType=IAM
 Credentials;UID=[YourAccessKey];PWD=[YourSecretKey];Workgroup=[WorkgroupName]
```

Para obter mais informações, procure "workgroup" no ODBC Driver Installation and Configuration Guide versão 1.0.5. Não há alterações para a string de conexão do driver ODBC ao usar tags nos grupos de trabalho. Para usar tags, atualize para a versão mais recente do driver ODBC, que é a versão atual.

Essa versão do driver permite que você use as ações de grupo de trabalho da API do Athena para criar e gerenciar grupos de trabalho e as ações de etiqueta da API do Athena para adicionar, listar ou remover etiquetas dos grupos de trabalho. Antes de começar, você deve ter permissões no nível do recurso no IAM para as ações nos grupos de trabalho e nas etiquetas.

Para obter mais informações, consulte:

- Usar grupos de trabalho para executar consultas e Exemplo de políticas de grupo de trabalho.
- Etiquetar recursos do Athena e Políticas de controle de acesso do IAM baseadas em etiquetas.

Se você usa o driver JDBC ou o AWS SDK, faça upgrade para a versão mais recente do driver e do SDK, ambos já incluem suporte para grupos de trabalho e etiquetas no Athena. Para ter mais informações, consulte Conectar-se ao Amazon Athena com JDBC.

22 de fevereiro de 2019

Publicado em 22/02/2019

Suporte incluído para etiquetas de grupos de trabalho no Amazon Athena. Uma tag consiste em uma chave e um valor, ambos definidos por você. Quando você marca um grupo de trabalho, atribui metadados personalizados a ele. É possível adicionar tags a grupos de trabalho para ajudar a categorizá-los, usando as práticas recomendadas de marcação com tags da AWS. Você pode usar tags para restringir o acesso a grupos de trabalho e rastrear custos. Por exemplo, crie um grupo de trabalho para cada centro de custo. Ao adicionar etiquetas a esses grupos de trabalho, você pode acompanhar os gastos com o Athena para cada centro de custo. Para obter mais informações, consulte Usar tags para faturamento no Guia do usuário do AWS Billing and Cost Management;.

Você pode trabalhar com etiquetas usando o console do Athena ou as operações de API. Para ter mais informações, consulte Etiquetar recursos do Athena.

No console do Athena, você pode adicionar uma ou mais etiquetas a cada um dos grupos de trabalho e pesquisá-las. Os grupos de trabalho são um recurso controlado pelo IAM no Athena. No IAM, você pode restringir quem pode adicionar, remover ou listar etiquetas nos grupos de trabalho que você cria. Você também pode usar a operação de API `CreateWorkGroup` que tenha o parâmetro de tag opcional para adicionar uma ou mais tags ao grupo de trabalho. Para adicionar, remover ou listar tags, use `TagResource`, `UntagResource` e `ListTagsForResource`. Para ter mais informações, consulte Usar operações de etiquetas.

Para permitir que os usuários adicionem etiquetas ao criar grupos de trabalho, você deve conceder a cada usuário permissões do IAM para as ações da API `TagResource` e `CreateWorkGroup`. Para ter mais informações e exemplos, consulte Políticas de controle de acesso do IAM baseadas em etiquetas.

Não há alterações para o driver JDBC ao usar tags nos grupos de trabalho. Se você criar grupos de trabalho e usar o driver JDBC ou o AWS SDK, faça upgrade para a versão mais recente do driver e do SDK. Para ter mais informações, consulte Conectar-se ao Amazon Athena com JDBC.

18 de fevereiro de 2019

Publicado em 18/02/2019

Adição da capacidade de controlar os custos de consulta ao executar consultas nos grupos de trabalho. Para ter mais informações, consulte Usar grupos de trabalho para controlar o acesso a consultas e os custos. Melhoria no JSON OpenX SerDe usado no Athena, foi corrigido um problema em que o Athena não ignorava os objetos que fizeram a transição para a classe de armazenamento do `GLACIER` e foram incluídos exemplos de consulta de logs do Network Load Balancer.

Foram feitas as seguintes alterações:

- Adição de suporte a grupos de trabalho. Uso de grupos de trabalho para separar usuários, equipes, aplicativos ou cargas de trabalho e para definir limites de quantidade de dados que cada consulta ou todo o grupo de trabalho pode processar. Como os grupos de trabalho atuam como recursos do IAM, você pode usar permissões no nível do recurso para controlar o acesso a um determinado grupo de trabalho. Você também pode visualizar as métricas relacionadas à consulta no Amazon CloudWatch, controlar os custos das consultas configurando limites para a quantidade de dados verificada, criar limites e acionar ações, como alarmes do Amazon SNS, quando esses

limites são violados. Para obter mais informações, consulte Usar grupos de trabalho para executar consultas e Controlar custos e monitorar consultas com métricas e eventos do CloudWatch.

Os grupos de trabalho são um recurso do IAM. Para obter uma lista completa de recursos, condições e ações relacionados a grupo de trabalho no IAM, consulte Ações, recursos e chaves de condição do Amazon Athena na Referência de autorização do serviço. Antes de criar grupos de trabalho, não esqueça de usar as políticas de grupo de trabalho do IAM e a Política gerenciada pela AWS: AmazonAthenaFullAccess.

Você pode começar a usar grupos de trabalho no console com as operações da API do grupo de trabalho ou com o driver JDBC. Para um procedimento de alto nível, consulte Configurar grupos de trabalho. Para baixar o driver JDBC com suporte ao grupo de trabalho, consulte Conectar-se ao Amazon Athena com JDBC.

Se você usar grupos de trabalho com o driver JDBC, defina o nome do grupo de trabalho na string de conexão usando o parâmetro de configuração `Workgroup`, como no exemplo a seguir:

```
jdbc:awsathena://AwsRegion=<AWSREGION>;UID=<ACCESSKEY>;
PWD=<SECRETKEY>;S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/<athena-
output>-<AWSREGION>/;
Workgroup=<WORKGROUPNAME>;
```

Não há alterações na forma como você executa instruções SQL ou faz chamadas de API JDBC para o driver. O driver passa o nome do grupo de trabalho para o Athena.

Para obter informações sobre as diferenças introduzidas com os grupos de trabalho, consulte APIs de grupos de trabalho do Athena e Resolver problemas nos grupos de trabalho.

- Melhoria no JSON OpenX SerDe usado no Athena. As melhorias incluem, entre outros:
  - Suporte à propriedade `ConvertDotsInJsonKeysToUnderscores`. Quando definido como `TRUE`, permite que o SerDe substitua por sublinhados os pontos nos nomes de chaves. Por exemplo, se o conjunto de dados JSON tem uma chave chamada `"a.b"`, você pode usar essa propriedade para definir o nome da coluna como `"a_b"` no Athena. O padrão é `FALSE`. Por padrão, o Athena não permite pontos nos nomes de coluna.
  - Suporte à propriedade `case.insensitive`. Por padrão, o Athena exige que todas as chaves do conjunto de dados JSON estejam em letras minúsculas. O uso de `WITH SERDE PROPERTIES ("case.insensitive"= FALSE;)` permite usar nomes de chaves com diferenciação de maiúsculas e minúsculas nos seus dados. O padrão é `TRUE`. Quando definido como `TRUE`, o SerDe converte todas as colunas em maiúsculas para minúsculas.

Para ter mais informações, consulte OpenX JSON SerDe.

- Um problema foi corrigido em que o Athena retornava mensagens de erro `"access denied"` quando processava objetos do Amazon S3 arquivados no Glacier pelas políticas de ciclo de vida do Amazon S3. Como resultado da correção desse problema, o Athena ignora os objetos que fizeram a transição para a classe de armazenamento do `GLACIER`. O Athena não permite a consulta de dados na classe de armazenamento do `GLACIER`.

  Para obter mais informações, consulte the section called “Requisitos para tabelas no Athena e dados no Amazon S3” e Transição para a classe de armazenamento do GLACIER (arquivamento de objetos) no Guia do usuário do Amazon Simple Storage Service.
- Adição de exemplos para consultar logs de acesso do Network Load Balancer que recebem informações sobre as solicitações de Transport Layer Security (TLS). Para ter mais informações, consulte the section called “Network Load Balancer”.

Notas de release do Athena para 2018

20 de novembro de 2018

Publicado em 20/11/2018

As novas versões dos drivers JDBC e ODBC foram lançadas com suporte para acesso federado à API do Athena com o AD FS e o SAML 2.0 (Security Assertion Markup Language 2.0). Para obter mais detalhes, consulte as Notas de release do driver JDBC e Notas de release do driver ODBC.

Com essa versão, o acesso federado ao Athena é permitido para o Active Directory Federation Service (AD FS 3.0). O acesso é estabelecido por meio das versões dos drivers JDBC ou ODBC que oferecem suporte ao SAML 2.0. Para obter informações sobre como configurar o acesso federado à API do Athena, consulte the section called “Permitir acesso federado à API do Athena”.

Para fazer download do driver JDBC versão 2.0.6 e da respectiva documentação, consulte Conectar-se ao Amazon Athena com JDBC. Para obter mais informações sobre essa versão, consulte as Notas de release do driver JDBC.

Para fazer download do driver ODBC versão 1.0.4 e da respectiva documentação, consulte Conectar-se ao Amazon Athena com ODBC. Para obter mais informações sobre essa versão, consulte as Notas de release do driver ODBC.

Para obter mais informações sobre o suporte ao SAML 2.0 na AWS, consulte Sobre a federação do SAML 2.0 no Manual do usuário do IAM.

15 de outubro de 2018

Publicado em 15/10/2018

Se você tiver atualizado para o AWS Glue Data Catalog, haverá dois novos recursos que fornecem suporte para:

- Criptografia de metadados do Catálogo de dados. Se você criptografar os metadados no Catálogo de dados, deverá adicionar políticas específicas ao Athena. Para obter mais informações, consulte Acesso a metadados criptografados no AWS Glue Data Catalog.
- Permissões refinadas para acessar recursos no AWS Glue Data Catalog. Agora você pode definir políticas baseadas em identidade (IAM) que restringem ou permitem o acesso a bancos de dados e tabelas específicos do Catálogo de dados usados no Athena. Para ter mais informações, consulte Acesso granular a bancos de dados e tabelas no AWS Glue Data Catalog.

> **Note**
>
> Os dados residem nos buckets do Amazon S3 e o acesso a eles é controlado pelas Acesso ao Amazon S3. Para acessar os dados em bancos de dados e tabelas, continue usando as políticas de controle de acesso aos buckets do Amazon S3 que armazenam os dados.

10 de outubro de 2018

Publicado em 10/10/2018

O Athena oferece suporte a `CREATE TABLE AS SELECT`, que cria uma tabela com base no resultado de uma instrução de consulta `SELECT`. Para obter detalhes, consulte Criar uma tabela a partir de resultados da consulta (CTAS).

Antes de criar consultas CTAS, é importante conhecer melhor o comportamento delas na documentação do Athena. Ela inclui informações sobre o local para salvar os resultados das consultas no Amazon S3, a lista de formatos compatíveis para armazenar resultados das consultas CTAS, o número de partições que é possível criar e os formatos de compactação permitidos. Para ter mais informações, consulte Considerações e limitações de consultas CTAS.

Use consultas CTAS para:

- Crie uma tabela a partir de resultados da consulta em uma etapa.
- Crie consultas CTAS no console do Athena, usando Exemplos. Para obter informações sobre a sintaxe, consulte CREATE TABLE AS.
- Transforme resultados da consulta em outros formatos de armazenamento, como PARQUET, ORC, AVRO, JSON e TEXTFILE. Para obter mais informações, consulte Considerações e limitações de consultas CTAS e Formatos de armazenamento colunar.

6 de setembro de 2018

Publicado em 06/09/2018

Lançada a nova versão do driver ODBC (versão 1.0.3). A nova versão do driver ODBC transmite resultados por padrão, em vez de paginação por meio deles, permitindo que as ferramentas de business intelligence recuperem grandes conjuntos de dados com mais rapidez. Essa versão também inclui melhorias, correções de erros e uma documentação atualizada para "Usar SSL com um servidor de proxy". Para obter detalhes, consulte as Notas de release do driver.

Para fazer download do driver ODBC versão 1.0.3 e da respectiva documentação, consulte Conectar-se ao Amazon Athena com ODBC.

O recurso de streaming de resultados está disponível com essa nova versão do driver ODBC. Também está disponível com o driver JDBC. Para obter informações sobre o streaming de resultados, consulte o Guia de instalação e configuração do driver ODBC e pesquise por UseResultsetStreaming.

O driver ODBC versão 1.0.3 é um substituto pronto para a versão anterior do driver. Recomendamos que você migre para o driver atual.

> **Important**
>
> Para usar o driver ODBC versão 1.0.3, siga estes requisitos:
>
> - Mantenha a porta 444 aberta para tráfego de saída.
> - Adicione a ação da política `athena:GetQueryResultsStream` à lista de políticas do Athena. Essa ação de política não é exposta diretamente com a API e só é usada com os

drivers ODBC e JDBC, como parte do suporte ao streaming de resultados. Para visualizar um exemplo de política, consulte Política gerenciada AWS: AWSQuicksightAthenaAccess.

23 de agosto de 2018

Publicado em 23/08/2018

Adicionado suporte para estes recursos relacionados à DDL e corrigidos vários erros, como a seguir:

- Adicionado suporte para os tipos de dados `BINARY` e `DATE` para dados no Parquet, e para os tipos de dados `DATE` e `TIMESTAMP` para dados no Avro.
- Adicionamos suporte para `INT` e `DOUBLE` em consultas DDL. `INTEGER` é um alias para `INT`, e `DOUBLE PRECISION` é um alias para `DOUBLE`.
- Melhor performance de consultas `DROP TABLE` e `DROP DATABASE`.
- Foi removida a criação de objeto `_$folder$` no Amazon S3 quando um bucket de dados está vazio.
- Corrigido um problema em que `ALTER TABLE ADD PARTITION` gerou um erro quando nenhum valor de partição foi fornecido.
- Corrigido um problema em que `DROP TABLE` ignorou o nome do banco de dados ao verificar partições após o nome qualificado ter sido especificado na instrução.

Para saber mais sobre os tipos de dados compatíveis com o Athena, consulte Tipos de dados no Amazon Athena.

Para obter informações sobre os mapeamentos de tipos de dados permitidos entre os tipos no Athena, o driver JDBC e os tipos de dados do Java, consulte a seção "Tipos de dados" no Guia de instalação e configuração do driver JDBC.

16 de agosto de 2018

Publicado em 16/08/2018

Lançado o driver JDBC versão 2.0.5. A nova versão do driver JDBC transmite resultados por padrão, em vez de paginação por meio deles, permitindo que as ferramentas de business intelligence recuperem grandes conjuntos de dados com mais rapidez. Em comparação com a versão anterior do driver JDBC, houve as seguintes melhorias de performance:

- Aproximadamente, o dobro da performance ao obter menos que 10.000 linhas.
- Aproximadamente, cinco a seis vezes de aumento na performance ao obter mais que 10.000 linhas.

O recurso de streaming de resultados está disponível apenas com o driver JDBC. Não está disponível com o driver ODBC. Você não pode usá-lo com a API do Athena. Para obter informações sobre o streaming de resultados, consulte o Guia de instalação e configuração do driver JDBC e pesquise por UseResultsetStreaming.

Para fazer download do driver JDBC versão 2.0.5 e da respectiva documentação, consulte Conectar-se ao Amazon Athena com JDBC.

O driver JDBC versão 2.0.5 é um substituto pronto para a versão anterior do driver (2.0.2). Para garantir que você possa usar o driver JDBC versão 2.0.5, adicione a ação da política `athena:GetQueryResultsStream` à lista de políticas do Athena. Essa ação de política não é exposta diretamente com a API e só é usada com o driver JDBC, como parte do suporte ao streaming de resultados. Para visualizar um exemplo de política, consulte Política gerenciada AWS: AWSQuicksightAthenaAccess. Para obter mais informações sobre como migrar da versão 2.0.2 para a versão 2.0.5 do driver, consulte o Guia de migração do driver JDBC.

Se estiver migrando de um driver 1.x para um driver 2.x, você precisará migrar suas configurações existentes para a nova configuração. É altamente recomendável migrar para a versão atual do driver. Para obter mais informações, consulte e o Guia de migração do driver JDBC.

7 de agosto de 2018

Publicado em 07/08/2018

Agora você pode armazenar logs de fluxo do Amazon Virtual Private Cloud diretamente no Amazon S3 em um formato GZIP que você pode consultar no Athena. Para obter informações, consulte Consultar os logs de fluxo do Amazon VPC e Logs de fluxo da Amazon VPC agora podem ser entregues para o S3.

5 de junho de 2018

Publicado em 05/06/2018

Tópicos

- Suporte para exibições

- Melhorias e atualizações em mensagens de erro
- Correções de bugs

Suporte para exibições

Adicionado suporte para exibições. Agora você pode usar CREATE VIEW, DESCRIBE VIEW, DROP VIEW, SHOW CREATE VIEW e SHOW VIEWS no Athena. A consulta que define a exibição é executada sempre que você faz referência à exibição em uma consulta. Para ter mais informações, consulte Trabalhar com visualizações.

Melhorias e atualizações em mensagens de erro

- Foi incluída uma biblioteca GSON 2.8.0 no CloudTrail SerDe para resolver um problema com o CloudTrail SerDe e habilitar a análise de strings JSON.
- Validação aprimorada do esquema de partição no Athena para Parquet e, em alguns casos, para ORC, permitindo a reclassificação de colunas. Isso permite que o Athena processe melhor as alterações feitas na evolução do esquema ao longo do tempo e as tabelas adicionadas pelo Crawler do AWS Glue. Para ter mais informações, consulte Tratamento de atualizações do esquema.
- Foi adicionado suporte para análise de `SHOW VIEWS`.
- Feitas as seguintes melhorias nas mensagens de erro mais comuns:
  - Foi substituída uma mensagem de Internal Error (Erro interno) por uma mensagem descritiva do erro quando um SerDe não consegue analisar a coluna em uma consulta do Athena. Antes, o Athena emitia um erro interno em caso de erros de análise. A nova mensagem de erro lê: " HIVE_BAD_DATA: Erro ao analisar o valor de campo para o campo 0: java.lang. A string não pode ser convertida para org.openx.data.jsonserde.json.JSONObject".
  - Mensagens de erro aprimoradas sobre permissões insuficientes, adicionando mais detalhes.

Correções de bugs

Corrigidos os seguintes bugs:

- Corrigido um problema que permite a conversão interna dos tipos de dados `REAL` em `FLOAT`. Isso melhora a integração com o crawler do AWS Glue que retorna tipos de dados `FLOAT`.
- Foi corrigido um problema em que o Athena não convertia `DECIMAL` do AVRO (um tipo lógico) em um tipo `DECIMAL`.

- Foi corrigido um problema em que o Athena não retornava resultados das consultas em dados do Parquet com cláusulas `WHERE` que faziam referência a valores no tipo de dados `TIMESTAMP`.

17 de maio de 2018

Publicado em 17/05/2018

A cota de simultaneidade de consultas no Athena foi aumentada de cinco para vinte. Isso significa que você pode enviar e executar até 20 consultas `DDL` e 20 consultas `SELECT` de cada vez. Observe que as cotas de simultaneidade são separadas para as consultas `DDL` e `SELECT`.

As cotas de simultaneidade no Athena são definidas como o número de consultas que podem ser enviadas ao mesmo tempo para o serviço. Você pode enviar até 20 consultas do mesmo tipo (`DDL` ou `SELECT`) por vez. Se você enviar uma consulta que exceda a cota de consultas simultâneas, a API do Athena exibirá uma mensagem de erro.

Depois que você envia suas consultas para o Athena, ele as processa atribuindo recursos com base na carga de serviço geral e na quantidade de solicitações recebidas. Nós monitoramos e fazemos ajustes continuamente no serviço para que suas consultas sejam processadas o mais rápido possível.

Para ter mais informações, consulte Service Quotas. Esta é uma cota ajustável. Você pode usar o console do Service Quotas para solicitar um aumento de cotas para consultas simultâneas.

19 de abril de 2018

Publicado em 19/04/2018

Lançada a nova versão do driver JDBC (versão 2.0.2) com suporte para retorno de dados de `ResultSet` como um tipo de dados Array, melhorias e correções de bugs. Para obter detalhes, consulte as Notas de release do driver.

Para obter informações sobre como fazer download do novo driver JDBC versão 2.0.2 e respectiva documentação, consulte Conectar-se ao Amazon Athena com JDBC.

A versão mais recente do driver JDBC é a 2.0.2. Se estiver migrando de um driver 1.x para um driver 2.x, você precisará migrar suas configurações existentes para a nova configuração. É altamente recomendável que você migre para o driver atual.

Para obter informações sobre as alterações introduzidas na nova versão do driver, as diferenças de versão e os exemplos, consulte o Guia de migração do driver JDBC.

6 de abril de 2018

Publicado em 06/04/2018

Use o preenchimento automático para digitar consultas no console do Athena.

15 de março de 2018

Publicado em 15/03/2018

Recurso incluído para criação automática de tabelas do Athena para arquivos de log do CloudTrail diretamente no console do CloudTrail. Para ter mais informações, consulte Usar o console do CloudTrail para criar uma tabela do Athena com logs do CloudTrail .

2 de fevereiro de 2018

Publicado em 12/02/2018

Adicionada a capacidade de descarregar com segurança dados intermediários no disco para consultas com uso intensivo de memória que empregam a cláusula `GROUP BY`. Isso melhora a confiabilidade dessas consultas, evitando erros do tipo "Query resource exhausted" (Recurso de consulta esgotado).

19 de janeiro de 2018

Publicado em 19/01/2018

O Athena usa o Presto, um mecanismo de consulta distribuído de código aberto, para executar consultas.

Com o Athena, não há versões a serem gerenciadas. Fizemos upgrade transparente do mecanismo subjacente no Athena para uma versão baseada no Presto 0.172. Não é necessária nenhuma ação de sua parte.

Com o upgrade, você pode usar as funções e os operadores do Presto 0.172, incluindo as expressões do Lambda para Presto 0.172 no Athena.

Entre as principais atualizações desta versão, inclusive correções de contribuição da comunidade, estão:

- Suporte para ignorar cabeçalhos. Você pode usar a propriedade `skip.header.line.count` ao definir tabelas para permitir que o Athena ignore os cabeçalhos. Esse recurso é permitido para consultas que usam o LazySimpleSerDe e o SerDe do OpenCSV, e não o SerDes do Grok ou Regex.
- Suporte para o tipo de dados `CHAR(n)` em funções `STRING`. O intervalo de `CHAR(n)` é `[1.255]`, e o intervalo de `VARCHAR(n)` é `[1,65535]`.
- Suporte para subconsultas correlacionadas.
- Suporte para expressões e funções do Lambda do Presto.
- Performance aprimorada do tipo `DECIMAL` e dos operadores.
- Suporte para agregações filtradas, como `SELECT sum(col_name) FILTER`, em que `id > 0`.
- Predicados para os tipos de dados `DECIMAL`, `TINYINT`, `SMALLINT` e `REAL`.
- Suporte para predicados de comparação quantificados: `ALL`, `ANY` e `SOME`.
- Funções adicionadas: `arrays_overlap()`, `array_except()`, `levenshtein_distance()`, `codepoint()`, `skewness()`, `kurtosis()`, e `typeof()`.
- Adicionada uma variante da função `from_unixtime()` que utiliza um argumento de fuso horário.
- Adicionadas as funções de agregação `bitwise_and_agg()` e `bitwise_or_agg()`.
- As funções `xxhash64()` e `to_big_endian_64()` foram adicionadas.
- Suporte adicional para aspas duplas ou barras invertidas de escape usando uma barra invertida com um sublinhado de caminho JSON nas funções `json_extract()` e `json_extract_scalar()`. Isso altera a semântica de qualquer invocação usando uma barra invertida, porque as barras invertidas já foram tratados como caracteres normais.

Para obter mais informações sobre as funções e os operadores, consulte Consultas, funções e operadores em DML neste guia e Functions and operators (Funções e operadores) na documentação do Presto.

O Athena não permite todos os recursos do Presto. Para obter mais informações, consulte Limitações.

Notas de release do Athena para 2017

13 de novembro de 2017

Publicado em 13/11/2017

Suporte adicional para conectar o Athena ao driver ODBC. Para ter mais informações, consulte Conectar-se ao Amazon Athena com ODBC.

1 de novembro de 2017

Publicado em 01/11/2017

Adicionado suporte para consultar dados geoespaciais e para regiões Ásia-Pacífico (Seul), Ásia-Pacífico (Mumbai) e UE (Londres). Para obter informações, consulte Consultar dados geoespaciais e Regiões da AWS e Endpoints.

19 de outubro de 2017

Publicado em 19/10/2017

Adicionado suporte para UE (Frankfurt). Para obter uma lista de regiões e endpoints compatíveis, consulte Regiões da AWS e Endpoints.

3 de outubro de 2017

Publicado em 03/10/2017

Crie consultas nomeadas do Athena com o AWS CloudFormation. Para obter mais informações, consulte AWS::Athena::NamedQuery no Guia do usuário do AWS CloudFormation.

25 de setembro de 2017

Publicado em 25/09/2017

Suporte incluído para a região Ásia-Pacífico (Sydney). Para obter uma lista de regiões e endpoints compatíveis, consulte Regiões da AWS e Endpoints.

14 de agosto de 2017

Publicado em 14/08/2017

Integração incluída com o AWS Glue Data Catalog e um assistente de migração para atualizar do catálogo de dados gerenciados do Athena para o AWS Glue Data Catalog. Para ter mais informações, consulte Integração com AWS Glue.

4 de agosto de 2017

Publicado em 04/08/2017

Adicionado suporte para Grok SerDe, que oferece um padrão de correspondência mais simples para registros em arquivos de texto desestruturados, como logs. Para ter mais informações, consulte Grok SerDe. Adicionados atalhos de teclado para percorrer o histórico de consultas usando o console (CTRL + ⇧/⇩ usando o Windows, CMD + ⇧/⇩ usando o Mac).

22 de junho de 2017

Publicado em 22/06/2017

Suporte incluído para as regiões Ásia-Pacífico (Tóquio) e Ásia-Pacífico (Singapura). Para obter uma lista de regiões e endpoints compatíveis, consulte Regiões da AWS e Endpoints.

8 de junho de 2017

Publicado em 08/06/2017

Suporte incluído para a região Europa (Irlanda). Para obter mais informações, consulte Regiões da AWS e endpoints.

19 de maio de 2017

Publicado em 19/05/2017

Foram incluídos uma API do Amazon Athena e suporte à AWS CLI no Athena; o driver JDBC foi atualizado para a versão 1.1.0; vários problemas foram corrigidos.

- O Amazon Athena permite a programação do aplicativo para o Athena. Para obter mais informações, consulte Referência de API do Amazon Athena. Os AWS SDKs mais recentes são compatíveis com a API do Athena. Para links da documentação e downloads, consulte a seção SDKs em Ferramentas da Amazon Web Services.
- A AWS CLI inclui novos comandos para o Athena. Para obter mais informações, consulte a Referência de API do Amazon Athena.
- Um novo driver JDBC 1.1.0 está disponível, o que oferece suporte à nova API do Athena, bem como aos recursos e às correções de bugs mais recentes. Faça download do driver em https://downloads.athena.us-east-1.amazonaws.com/drivers/AthenaJDBC41-1.1.0.jar. Recomendamos atualizar para a versão mais recente do driver JDBC do Athena. No entanto, você ainda pode usar

a versão do driver anterior. As versões anteriores do driver não oferecem suporte à API do Athena. Para ter mais informações, consulte Conectar-se ao Amazon Athena com JDBC.

- Ações específicas em declarações de política em versões anteriores do Athena foram substituídas. Se atualizar a versão do driver JDBC 1.1.0 e tiver políticas do IAM gerenciadas por clientes ou em linha anexadas a usuários do JDBC, você deverá atualizar as políticas do IAM. Por outro lado, as versões anteriores do driver JDBC não oferecem suporte à API do Athena. Dessa maneira, você só pode especificar ações obsoletas em políticas anexadas a usuários JDBC de uma versão anterior. Por esse motivo, você não precisa atualizar políticas do IAM gerenciadas pelo cliente ou em linha.
- Essas ações específicas da política foram usadas no Athena antes do lançamento da API do Athena. Use essas ações obsoletas em políticas apenas com drivers JDBC anteriores a essa versão 1.1.0. Se estiver atualizando o driver JDBC, substitua as declarações de políticas que permitem ou negam ações obsoletas por ações da API apropriada conforme listadas ou ocorrerão erros:

Ação específica da política obsoleta	Ação da API do Athena correspondente
`athena:RunQuery`	`athena:StartQueryExecution`
`athena:CancelQueryExecution`	`athena:StopQueryExecution`
`athena:GetQueryExecutions`	`athena:ListQueryExecutions`

Melhorias

- Aumentado o limite de tamanho da string de consulta para 256 KB.

Correções de bugs

- Corrigido um problema que fazia os resultados da consulta terem aparência malformada durante a rolagem pelos resultados no console.
- Corrigido um problema em que uma string de caractere `\u0000` em arquivos de dados do Amazon S3 causaria erros.

- Corrigido um problema que fazia as solicitações para cancelar uma consulta feita pelo driver JDBC falharem.
- Foi corrigido um problema que fazia com que o SerDe do AWS CloudTrail falhasse com dados do Amazon S3 na região Leste dos EUA (Ohio).
- Corrigido um problema que fez com que `DROP TABLE` falhasse em uma tabela particionada.

4 de abril de 2017

Publicado em 04/04/2017

Adicionado suporte adicional para criptografia de dados do Amazon S3 e liberada atualização do driver JDBC (versão 1.0.1) com suporte à criptografia, melhorias e correções de bugs.

Atributos

- Adicionados os seguintes recursos de criptografia:
  - Suporte para consultar dados criptografados no Amazon S3.
  - Suporte para criptografar resultados da consulta do Athena.
- Uma nova versão do driver oferece suporte a novos recursos de criptografia, adiciona melhorias e corrige problemas.
- Adicionada a possibilidade de adicionar, substituir e alterar colunas usando `ALTER TABLE`. Para obter mais informações, consulte Alter Column na documentação do Hive.
- Adicionado suporte para consultar dados compactados por LZO.

Para ter mais informações, consulte Criptografia inativa.

Melhorias

- Melhor performance da consulta JDBC com melhorias no tamanho da página, retornando 1.000 linhas, em vez de 100.
- Adicionada capacidade para cancelar uma consulta usando a interface do driver JDBC.
- Adicionada capacidade para especificar opções de JDBC no URL de conexão JDBC. Consulte Conectar-se ao Amazon Athena com JDBC para obter o driver JDBC mais atual.
- Adicionada configuração de PROXY ao driver, que já pode ser definida usando ClientConfiguration no AWS SDK for Java.

Correções de bugs

Corrigidos os seguintes bugs:

- Erros de controle de utilização ocorreriam quando várias consultas foram emitidas usando a interface do driver JDBC.
- O driver JDBC era interrompido durante a projeção de um tipo de dados decimal.
- O driver JDBC retornaria cada tipo de dados como uma string, independentemente de como o tipo de dados foi definido na tabela. Por exemplo, selecionar uma coluna definida como um tipo de dados `INT` usando `resultSet.GetObject()` retornaria um tipo de dados `STRING`, em vez de `INT`.
- O driver JDBC verificaria credenciais no momento em que uma conexão foi estabelecida, e não no momento em que uma consulta seria executada.
- As consultas feitas por meio do driver JDBC falhariam quando um esquema fosse especificado com o URL.

24 de março de 2017

Publicado em 24/03/2017

Foi adicionado o SerDe do AWS CloudTrail, a performance foi melhorada, os problemas de partição foram corrigidos.

Atributos

- Ocorreu a adição do SerDe para o AWS CloudTrail, que desde então foi substituído pelo Hive JSON SerDe para leitura de logs do CloudTrail. Para obter informações sobre a consulta de logs do CloudTrail, consulte Consultar os logs do AWS CloudTrail.

Melhorias

- Melhorada a performance durante o exame de um grande número de partições.
- Melhorada a performance na operação `MSCK Repair Table`.
- Recurso incluído para consultar dados do Amazon S3 armazenados em regiões diferentes da região principal. Taxas de transferência de dados entre regiões padrão do Amazon S3 se aplicam, além de cobranças do Athena padrão.

Correções de bugs

- Corrigido um bug em que um erro "tabela não encontrada" poderá ocorrer se nenhuma partição for carregada.
- Corrigido um bug para evitar lançar uma exceção com consultas `ALTER TABLE ADD PARTITION IF NOT EXISTS`.
- Corrigido um bug em `DROP PARTITIONS`.

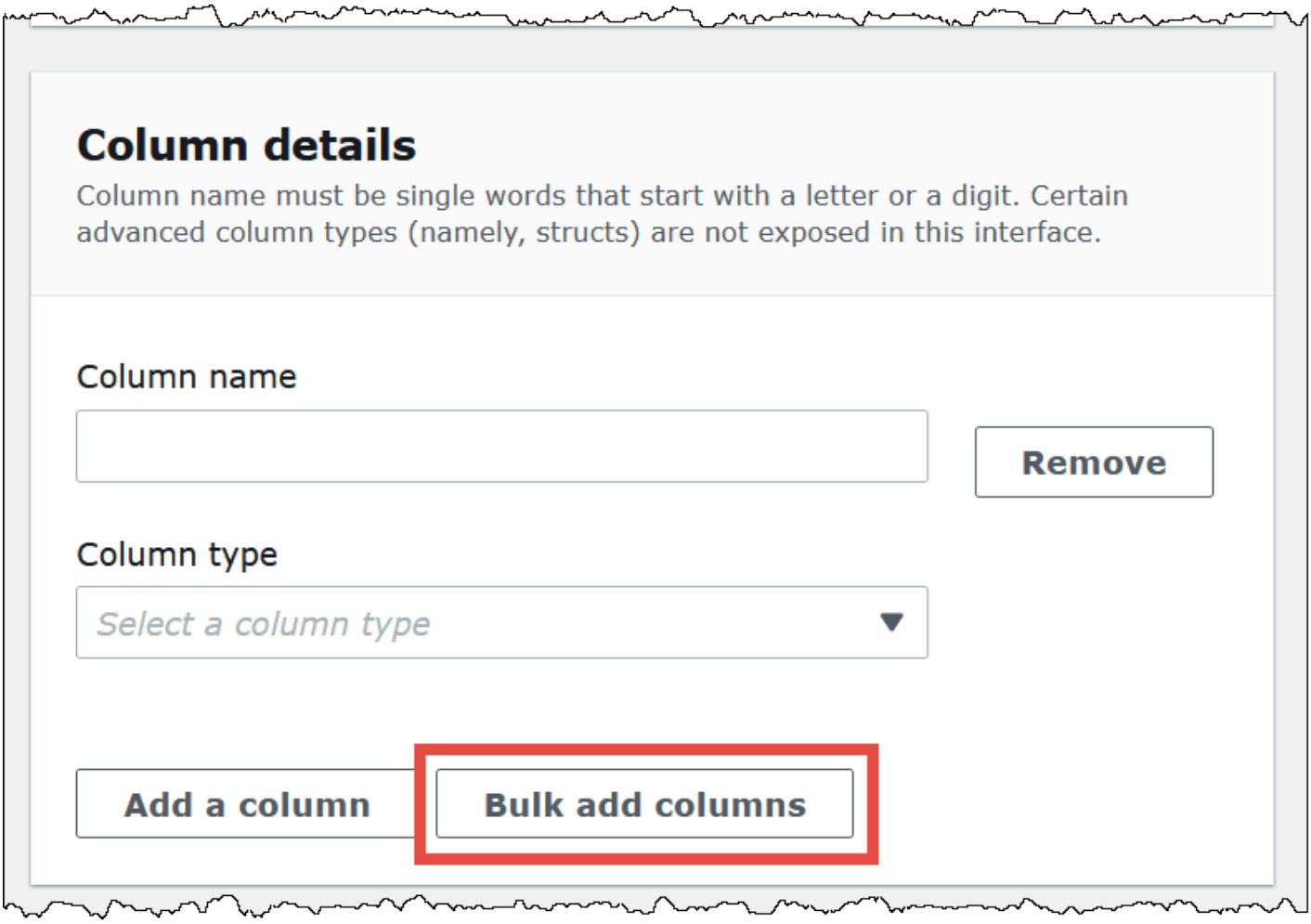
20 de fevereiro de 2017

Publicado em 20/02/2017

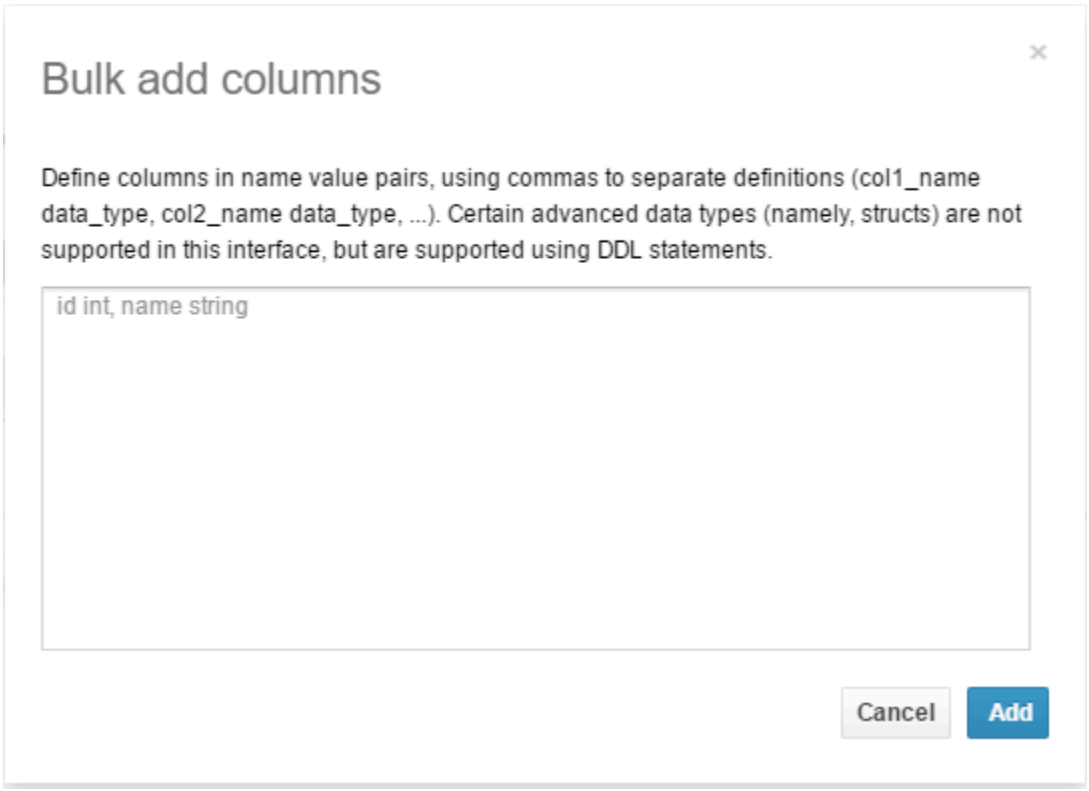
Suporte incluído para AvroSerDe e OpenCSVSerDe, região Leste dos EUA (Ohio) e edição em massa de colunas no assistente do console. Melhorada a performance em tabelas Parquet grandes.

Atributos

- Introduzido suporte para novo SerDes:
  - Avro SerDe
  - OpenCSVSerDe para processar CSV
- Lançamento da região Leste dos EUA (Ohio) (us-east-2) launch. Você já pode executar consultas nessa região.
- Agora é possível usar o formulário Create Table From S3 bucket data (Criar tabela a partir de dados do bucket do S3) para definir o esquema da tabela em massa. No editor de consultas, escolha Create (Criar), S3 bucket data (Dados do bucket do S3) e, em seguida, escolha Bulk add columns (Adicionar colunas em massa) na seção Column details (Detalhes da coluna).



Digite os pares de nome e valor na caixa de texto e escolha Add.



Melhorias

- Melhorada a performance em tabelas Parquet grandes.

Histórico do documento

Última atualização da documentação: 28 de maio de 2024.

Atualizamos a documentação com frequência para abordar seus comentários. A tabela a seguir descreve as inclusões importantes feitas na documentação do Amazon Athena. Nem todas as atualizações estão representadas.

Alteração	Descrição	Data de lançamento
Atualização da política gerenciada `AmazonAthenaFullAccess` .	As permissões `datazone:ListDomains` , `datazone:ListProjects` e `datazone:ListAccountEnvironments` foram adicionadas à política gerenciada AmazonAthenaFullAccess. As ações adicionadas permitem que os usuários do Athena trabalhem com domínios, projetos e ambientes do Amazon DataZone. Para ter mais informações, consulte Usar o Amazon DataZone no Athena.	3 de janeiro de 2024
Atualização da política gerenciada `AmazonAthenaFullAccess` .	As permissões `glue:StartColumnStatisticsTaskRun` , `glue:GetColumnStatisticsTaskRun` e `glue:GetColumnStatisticsTaskRuns` foram adicionadas à política gerenciada AmazonAthenaFullAccess. As ações adicionadas permitem que o Athena chame o AWS Glue para recuperar as estatísticas do atributo otimizador baseado em custos. Para ter mais informações, consulte Usar o otimizador baseado em custos.	3 de janeiro de 2024
Adição de documentação para grupos de trabalho do Athena habilitados para o Centro de Identidade do IAM.	É possível criar grupos de trabalho do Athena SQL que usam o modo de autenticação do Centro de Identidade do IAM. Esses grupos de trabalho são compatíveis com o uso da mesma identidade em serviços da AWS, como o Amazon Athena e o Amazon EMR Studio. Para ter mais informações, consulte Uso de grupos de trabalho	5 de dezembro de 2023

Alteração	Descrição	Data de lançamento
	do Athena habilitados para o Centro de Identidade do IAM.	
Adição de documentação para consultas de dados do S3 Express One Zone.	É possível usar o Athena para consultar dados na classe de armazenamento Amazon S3 Express One Zone. Para ter mais informações, consulte Consulta de dados do S3 Express One Zone.	28 de novembro de 2023
Adição de documentação para visualizações do Catálogo de Dados do Glue.	É possível usar as visualizações do Catálogo de Dados do Glue para fornecer uma perspectiva única e comum sobre os serviços da AWS, como o Amazon Athena e o Amazon Redshift. Para ter mais informações, consulte Uso das visualizações do AWS Glue Data Catalog.	27 de novembro de 2023
Adição de documentação para o recurso do otimizador baseado em custos.	É possível usar as estatísticas do AWS Glue para otimizar as consultas no Athena SQL. Para ter mais informações, consulte Usar o otimizador baseado em custos.	17 de novembro de 2023
Documentação adicionada para o driver JDBC 3.x do Athena	Você pode usar o driver JDBC 3.x do Athena para ler resultados de consultas diretamente no Amazon S3. O driver JDBC 3.x é compatível com quase todos os métodos de autenticação que são compatíveis com o driver JDBC 2.x. Para ter mais informações, consulte Driver JDBC 3.x do Athena.	16 de novembro de 2023
Adicionada documentação sobre o uso do DataZone no Athena.	Você pode usar o DataZone para simplificar sua experiência em todos os serviços de análise da AWS, como o Athena, o AWS Glue e o Lake Formation. Para ter mais informações, consulte Usar o Amazon DataZone no Athena.	4 de outubro de 2023

Alteração	Descrição	Data de lançamento
Foi adicionada documentação para reservas de capacidade.	Já é possível usar reservas de capacidade no Amazon Athena para executar consultas SQL em capacidade de computação totalmente gerenciada. Para ter mais informações, consulte Como gerenciar a capacidade de processamento de consulta.	28 de abril de 2023
Foi adicionada documentação sobre consulta de visualizações federadas.	Já é possível criar e consultar visualizações em fontes de dados federadas no Athena. Para ter mais informaçõ es, consulte Consultar visualizações federadas.	4 de abril de 2023
Foi adicionad a documenta ção sobre como evitar controle de utilização no Amazon S3.	Para ter mais informações, consulte Como prevenir o controle de utilização do Amazon S3.	24 de março de 2023
Atualização da política gerenciad a `AmazonAth enaFullAc cess` .	`pricing:GetProducts` adicionado à política gerenciada AmazonAthenaFullAccess. A ação adicionad a fornece acesso ao AWS Billing and Cost Managemen t. Para obter mais informações, consulte GetProduc ts na Referência de APIs do AWS Billing and Cost Management.	25 de janeiro de 2023
Documenta ção expandida referente ao suporte à compressão do Athena.	Tópicos individuais adicionados para Compressão de tabelas do Hive, Compressão de tabelas do Iceberg e Níveis de compressão ZSTD. Para ter mais informações, consulte Suporte a compactação no Athena.	20 de janeiro de 2023

Alteração	Descrição	Data de lançamento
Documentação adicionada ao Amazon Athena para Apache Spark.	Agora é possível criar e executar, de forma interativa, aplicações Apache Spark e cadernos compatíveis com Jupyter no Amazon Athena. Para ter mais informações, consulte Uso do Apache Spark no Amazon Athena.	30 de novembro de 2022
Documentação adicionada para o conector IBM Db2 do Athena.	É possível usar o conector do Amazon Athena para IBM Db2 para consultar o Db2 do Athena. Para ter mais informações, consulte Conector IBM Db2 do Amazon Athena.	18 de novembro de 2022
Documentação adicionada para reutilização do resultado da consulta.	Ao executar novamente uma consulta no Athena, agora é possível optar por reutilizar o último resultado de consulta armazenado, se desejar. Isso pode aumentar a performance e reduzir os custos, em termos de números, para os bytes verificados. Para ter mais informações, consulte Reutilização dos resultados da consulta.	8 de novembro de 2022
Documentação atualizada para os logs do CloudTrail.	O DDL `CREATE TABLE` para a consulta de logs do CloudTrail foi atualizado para usar o JSON SerDe em vez do CloudTrail SerDe. Para ter mais informações, consulte Consultar os logs do AWS CloudTrail.	3 de novembro de 2022
Adição de documentação sobre o mecanismo Athena versão 3.	Para obter mais informações sobre o mecanismo Athena versão 3, consulte Mecanismo Athena versão 3.	13 de outubro de 2022

Alteração	Descrição	Data de lançamento
Tutorial adicionado sobre como configurar o SSO para ODBC usando o plugin Okta.	Configure o driver ODBC do Amazon Athena e o plugin Okta para obter o recurso de logon único (SSO) usando o provedor de identidade Okta. Para ter mais informações, consulte Configurar o SSO para ODBC usando o plugin Okta e o Okta Identity Provider.	23 de agosto de 2022
Documentação incluída para visualizar planos de consulta e estatísticas no console do Athena.	É possível utilizar o editor de consultas do Athena para ver representações gráficas de como suas consultas serão executadas, bem como gráficos, detalhes e estatísticas de como as consultas concluídas foram executadas. Para obter mais informações, consulte Visualizar planos de execução para consultas SQL e Visualizar estatísticas e detalhes de execução para consultas concluídas.	21 de julho de 2022
Documentação adicionada sobre a consulta de visualizações do Apache Hive em metastores externos do Hive.	Você pode usar o Athena para consultar visualizações do Apache criadas em metastores externos do Hive. Algumas funções do Hive não são compatíveis ou exigem tratamento especial. Para ter mais informações, consulte Usar visualizações do Hive.	22 de abril de 2022
Inclusão da documentação para consultas salvas.	Você pode usar o recurso de consultas salvas no Athena para salvar, recuperar, editar e renomear consultas. Para mais informações, consulte Usar consultas salvas neste guia e UpdateNamedQuery na Referência de APIs do Amazon Athena.	28 de fevereiro de 2022

Alteração	Descrição	Data de lançamento
Documentação de demonstração adicionada referente ao suporte ao Apache Iceberg.	O Athena oferece suporte a consultas de leitura, viagem no tempo e gravação para tabelas Apache Iceberg que usam o formato Apache Parquet para dados e o catálogo do AWS Glue para metastore. Para ter mais informações, consulte Usar tabelas do Apache Iceberg.	26 de novembro de 2021
Documentação adicionada referente a consultas federadas entre contas.	Você pode usar o recurso de consulta federada entre contas para consultar origens de dados em outra conta. Para obter informações sobre como configurar permissões para ativar esse recurso, consulte Habilitar consultas federadas entre contas.	12 de novembro de 2021
Documentação incluída sobre a instrução `UNLOAD` no Athena.	Uso da instrução `UNLOAD` para gravar os resultados das consultas de uma instrução `SELECT` nos formatos Apache Parquet, ORC, Apache Avro e JSON. Para ter mais informações, consulte UNLOAD.	5 de agosto de 2021
Documentação incluída sobre o recurso da instrução `EXPLAIN` no Athena.	Para obter mais informações, consulte Usar EXPLAIN e EXPLAIN ANALYZE no Athena e Noções básicas sobre os resultados da instrução EXPLAIN do Athena.	5 de abril de 2021
Páginas adicionadas sobre solução de problemas e ajuste de performance no Athena.	Para obter mais informações, consulte Solução de problemas no Athena e Ajuste de performance no Athena.	30 de dezembro de 2020

Alteração	Descrição	Data de lançamento
Documentação incluída sobre o versionamento do mecanismo Athena e o mecanismo Athena versão 2.	Para ter mais informações, consulte Versionamento do mecanismo do Athena.	11 de novembro de 2020
Documentação sobre consulta federada atualizada para a versão de disponibilidade geral.	Para obter mais informações, consulte Usar a consulta federada do Amazon Athena e Usar o Athena com chaves de contexto CalledVia.	11 de novembro de 2020
Documentação incluída sobre como usar o driver JDBC com o Lake Formation para acesso federado ao Athena.	Para obter mais informações, consulte Usar o Lake Formation e os drivers JDBC e ODBC do Athena para acesso federado ao Athena e Tutorial: Configurar acesso federado para usuários do Okta ao Athena usando Lake Formation e JDBC.	25 de setembro de 2020
Documentação incluída sobre o conector de dados OpenSearch do Amazon Athena.	Para ter mais informações, consulte Conector do Amazon Athena para o OpenSearch.	21 de julho de 2020
Documentação incluída sobre consulta de conjuntos de dados do Hudi.	Para ter mais informações, consulte Usar o Athena para consultar conjuntos de dados do Apache Hudi.	9 de julho de 2020

Alteração	Descrição	Data de lançamento
Documentação incluída sobre consulta de logs do servidor da Web do Apache e do IIS armazenados no Amazon S3.	Para obter mais informações, consulte Consultar logs do Apache armazenados no Amazon S3 e Consultar logs do Internet Information Server (IIS) armazenados no Amazon S3.	8 de julho de 2020
Documentação incluída sobre o lançamento geral do conector de dados do Athena para metastore externo do Hive.	Para ter mais informações, consulte Usar o conector de dados do Athena para metastore externo do Hive.	1 de junho de 2020
Adição de documentação para marcação de recursos de catálogo de dados.	Para ter mais informações, consulte Etiquetar recursos do Athena.	1 de junho de 2020
Adição da documentação sobre projeção de partições.	Para ter mais informações, consulte Projeção de partições com o Amazon Athena.	21 de maio de 2020
Exemplos de código Java atualizados para o Athena.	Para ter mais informações, consulte Exemplos de código.	11 de maio de 2020

Alteração	Descrição	Data de lançamento
Tópico adicionado sobre como consultar as descobertas do Amazon GuardDuty.	Para ter mais informações, consulte Consultar descobertas do Amazon GuardDuty.	19 de março de 2020
Tópico adicionado sobre como usar o CloudWatch Events para monitorar transições de estado de consulta do Athena.	Para ter mais informações, consulte Monitorar consultas com eventos do Amazon EventBridge.	11 de março de 2020
Tópico adicionado sobre consulta de logs de fluxo do AWS Global Accelerator com o Athena.	Para ter mais informações, consulte Consultar logs de fluxo do AWS Global Accelerator.	6 de fevereiro de 2020

Alteração	Descrição	Data de lançamento
• Adição da documentação sobre o uso de CTAS com INSERT INTO para adicionar dados de uma origem não particionada a um destino particionado. • Adição de links para download da versão de pré-visualização 1.1.0 do driver ODBC para Athena. • Correção da descrição para regex `SHOW DATABASES LIKE`. • Correção da sintaxe `partitioned_by` no tópico de CTA. • Outras pequenas correções.	As atualizações da documentação incluem, entre outros, os seguintes tópicos: • Usar CTAS e INSERT INTO para ETL e análise de dados • Conectar-se ao Amazon Athena com ODBC (Os recursos de previsualização 1.1.0 agora estão incluídos no driver ODBC 1.1.2.) • SHOW DATABASES • CREATE TABLE AS	4 de fevereiro de 2020

Alteração	Descrição	Data de lançamento
Adição da documentação sobre como usar CTAS com INSERT INTO para adicionar dados de uma origem particionada a um destino particion ado.	Para ter mais informações, consulte Usar CTAS e INSERT INTO para resolver o limite de 100 partições.	22 de janeiro de 2020
Atualização de informações de local dos resultado s de consulta.	O Athena não cria mais um local de resultados da consulta “padrão”. Para ter mais informações, consulte Especificar um local para resultados de consultas.	20 de janeiro de 2020
Adição de um tópico sobre como consultar o AWS Glue Data Catalog. Foram atualizadas as informações sobre cotas de serviço (antes chamadas de “limites do serviço”) no Athena.	Para obter mais informações, consulte os tópicos a seguir. • Consulta no AWS Glue Data Catalog • Service Quotas	17 de janeiro de 2020

Alteração	Descrição	Data de lançamento
Correção do tópico sobre OpenCSVSerDe para destacar que o tipo `TIMESTAMP` deve ser especificado no formato numérico UNIX.	Para ter mais informações, consulte OpenCSVSerDe para processar CSV.	15 de janeiro de 2020
Tópico de segurança atualizado sobre criptografia para destacar que o Athena não permite chaves assimétricas.	O Athena permite apenas chaves simétricas para leitura e gravação de dados. Para ter mais informações, consulte Opções de criptografia permitidas do Amazon S3.	8 de janeiro de 2020
Foram adicionadas informações sobre acesso entre contas a buckets do Amazon S3 criptografados com uma chave AWS KMS personalizada.	Para ter mais informações, consulte Acesso entre contas a um bucket criptografado com uma chave personalizada do AWS KMS.	13 de dezembro de 2019

Alteração	Descrição	Data de lançamento
Adição da documentação para consultas federadas, metastores externos do Hive, machine learning e funções definidas pelo usuário. Novas métricas do CloudWatch adicionadas.	Para obter mais informações, consulte os tópicos a seguir. • Usar a consulta federada do Amazon Athena • Conectores de fonte de dados disponíveis • Usar o conector de dados do Athena para metastore externo do Hive • Usar Machine Learning (ML) com o Amazon Athena • Executar consultas com funções definidas pelo usuário • Lista de métricas e dimensões do CloudWatch para o Athena	26 de novembro de 2019
Adição da seção para o novo comando INSERT INTO e atualização das informações de localização do resultado da consulta para oferecer suporte a arquivos manifesto dos dados.	Para obter mais informações, consulte INSERT INTO e Trabalhar com resultados de consultas, consultas recentes e arquivos de saída.	18 de setembro de 2019

Alteração	Descrição	Data de lançamento
Adição da seção de suporte a endpoints da VPC de interface (PrivateLink). Atualização dos drivers JDBC. Foram atualizadas as informações sobre os logs de fluxo da VPC aprimorados.	Para obter mais informações, consulte Conectar-se ao Amazon Athena usando um endpoint da VPC de interface, Consultar os logs de fluxo do Amazon VPC e Conectar-se ao Amazon Athena com JDBC.	11 de setembro de 2019
Adição da seção sobre integração ao AWS Lake Formation.	Para ter mais informações, consulte Usar o Athena para consultar dados registrados com o AWS Lake Formation.	26 de junho de 2019
Atualização da seção Segurança para consistência com outros serviços da AWS.	Para ter mais informações, consulte Segurança do Amazon Athena.	26 de junho de 2019
Adição da seção sobre a consulta de logs do AWS WAF.	Para ter mais informações, consulte Consultar os logs do AWS WAF.	31 de maio de 2019

Alteração	Descrição	Data de lançamento
A nova versão do driver ODBC foi lançada com suporte a grupos de trabalho do Athena.	Para fazer download do driver ODBC versão 1.0.5 e da respectiva documentação, consulte Conectar-se ao Amazon Athena com ODBC. Não há alterações para a string de conexão do driver ODBC ao usar tags nos grupos de trabalho. Para usar tags, atualize para a versão mais recente do driver ODBC, que é a versão atual. Essa versão do driver permite que você use as ações de grupo de trabalho da API do Athena para criar e gerenciar grupos de trabalho e as ações de etiqueta da API do Athena para adicionar, listar ou remover etiquetas dos grupos de trabalho. Antes de começar, você deve ter permissões no nível do recurso no IAM para as ações nos grupos de trabalho e nas etiquetas.	5 de março de 2019
Suporte incluído para etiquetas de grupos de trabalho no Amazon Athena.	Uma tag consiste em uma chave e um valor, ambos definidos por você. Quando você marca um grupo de trabalho, atribui metadados personalizados a ele. Por exemplo, crie um grupo de trabalho para cada centro de custo. Ao adicionar etiquetas a esses grupos de trabalho, você pode acompanhar os gastos com o Athena para cada centro de custo. Para obter mais informações, consulte Usar etiquetas para faturamento no Guia do usuário do AWS Billing and Cost Management.	22 de fevereiro de 2019

Alteração	Descrição	Data de lançamento
Melhoria no JSON OpenX SerDe usado no Athena.	As melhorias incluem, entre outros: • Suporte à propriedade `ConvertDotsInJsonKeysToUnderscores` . Quando definido como `TRUE`, permite que o SerDe substitua por sublinhados os pontos nos nomes de chaves. Por exemplo, se o conjunto de dados JSON tem uma chave chamada `"a.b"`, você pode usar essa propriedade para definir o nome da coluna como `"a_b"` no Athena. O padrão é `FALSE`. Por padrão, o Athena não permite pontos nos nomes de coluna. • Suporte à propriedade `case.insensitive` . Por padrão, o Athena exige que todas as chaves do conjunto de dados JSON estejam em letras minúsculas. O uso de `WITH SERDE PROPERTIES ("case.insensitive"= FALSE;)` permite usar nomes de chaves com diferenciação de maiúsculas e minúsculas nos seus dados. O padrão é `TRUE`. Quando definido como `TRUE`, o SerDe converte todas as colunas em maiúsculas para minúsculas. Para ter mais informações, consulte OpenX JSON SerDe.	18 de fevereiro de 2019

Alteração	Descrição	Data de lançamento
Adição de suporte a grupos de trabalho.	Uso de grupos de trabalho para separar usuários, equipes, aplicativos ou cargas de trabalho e para definir limites de quantidade de dados que cada consulta ou todo o grupo de trabalho pode processar. Como os grupos de trabalho atuam como recursos do IAM, você pode usar permissões no nível do recurso para controlar o acesso a um determinado grupo de trabalho. Você também pode visualizar as métricas relacionadas à consulta no Amazon CloudWatch, controlar os custos das consultas configurando limites para a quantidade de dados verificada, criar limites e acionar ações, como alarmes do Amazon SNS, quando esses limites são violados. Para obter mais informações, consulte Usar grupos de trabalho para executar consultas e Controlar custos e monitorar consultas com métricas e eventos do CloudWatch.	18 de fevereiro de 2019
Suporte incluído para análise de logs do Network Load Balancer.	Exemplo adicionado de consultas do Athena para análise de logs do Network Load Balancer. Esses logs recebem informações detalhadas sobre as solicitações Transport Layer Security (TLS) enviadas ao Network Load Balancer. Você pode usar esses logs de acesso para analisar padrões de tráfego e solucionar problemas . Para ter mais informações, consulte the section called “Network Load Balancer”.	24 de janeiro de 2019

Alteração	Descrição	Data de lançamento
As novas versões dos drivers JDBC e ODBC foram lançadas com suporte para acesso federado à API do Athena com o AD FS e o SAML 2.0 (Security Assertion Markup Language 2.0).	Com esse lançamento dos drivers, o acesso federado ao Athena é permitido no Active Directory Federation Service (AD FS 3.0). O acesso é estabelecido por meio das versões dos drivers JDBC ou ODBC que oferecem suporte ao SAML 2.0. Para obter informações sobre como configurar o acesso federado à API do Athena, consulte the section called “Permitir acesso federado à API do Athena”.	10 de novembro de 2018
Suporte incluído para controle de acesso granular a bancos de dados e tabelas no Athena. Políticas também foram adicionadas ao Athena para permitir a criptografia de metadados de banco de dados e tabela no Catálogo de dados.	Suporte incluído para criação de políticas baseadas em identidade (IAM) que oferecem controle de acesso granular a recursos no AWS Glue Data Catalog, como bancos de dados e tabelas usados no Athena. Você também pode criptografar os metadados de banco de dados e tabela no Catálogo de dados adicionando políticas específicas ao Athena. Para obter detalhes, consulte Acesso granular a bancos de dados e tabelas no AWS Glue Data Catalog.	15 de outubro de 2018

Alteração	Descrição	Data de lançamento
Adicionado suporte para instruções `CREATE TABLE AS SELECT`. Outras melhorias foram feitas na documentação.	Adicionado suporte para instruções `CREATE TABLE AS SELECT`. Consulte Criar uma tabela a partir de resultados de consultas (CTAS), Considerações e limitações de consultas CTAS e Exemplos de consultas CTAS.	10 de outubro de 2018
Liberado o driver ODBC versão 1.0.3 com suporte para streaming de resultados em vez de obtê-los em páginas. Outras melhorias foram feitas na documentação.	O driver ODBC versão 1.0.3 oferece suporte ao streaming de resultados e também inclui melhorias, correções de erros e uma documentação atualizada para "Usar o SSL com um servidor de proxy". Para fazer download do driver ODBC versão 1.0.3 e da respectiva documentação, consulte Conectar-se ao Amazon Athena com ODBC.	6 de setembro de 2018
Liberado o driver JDBC versão 2.0.5 com suporte para padrão para streaming de resultados em vez de buscá-los em páginas. Outras melhorias foram feitas na documentação.	Liberado o driver JDBC 2.0.5 com suporte para padrão para streaming de resultados em vez de buscá-los em páginas. Para ter mais informações, consulte Conectar-se ao Amazon Athena com JDBC.	16 de agosto de 2018

Alteração	Descrição	Data de lançamento
Documentação atualizada sobre consulta de logs de fluxo do Amazon Virtual Private Cloud, que podem ser armazenados diretamente no Amazon S3 no formato GZIP. Atualizados os exemplos para consultar logs ALB.	Documentação atualizada sobre consulta de logs de fluxo do Amazon Virtual Private Cloud, que podem ser armazenados diretamente no Amazon S3 no formato GZIP. Para ter mais informações, consulte Consultar os logs de fluxo do Amazon VPC. Atualizados os exemplos para consultar logs ALB. Para ter mais informações, consulte Consultar logs do Application Load Balancer.	7 de agosto de 2018
Adicionado suporte para exibições. Foram adicionad as orientações para manuseios do esquema para vários formatos de armazenamento físico de dados.	Adicionado suporte para exibições. Para ter mais informações, consulte Trabalhar com visualizações. Este guia foi atualizado com orientação sobre o manuseio de atualizações do esquema para vários formatos de armazenamento físico de dados. Para ter mais informações, consulte Tratamento de atualizações do esquema.	5 de junho de 2018
Limites maiores de simultaneidade de consultas padrão de 5 para 20.	Você pode enviar e executar até 20 consultas DDL e 20 consultas SELECT de cada vez. Para ter mais informaçõ es, consulte Service Quotas.	17 de maio de 2018

Alteração	Descrição	Data de lançamento
Adicionadas as abas de consulta e a capacidade de configurar o preenchimento automático no Query Editor.	Adicionadas as abas de consulta e a capacidade de configurar o preenchimento automático no Query Editor. Para ter mais informações, consulte Conceitos básicos.	8 de maio de 2018
Lançado o driver JDBC versão 2.0.2.	Lançada a nova versão do driver JDBC (versão 2.0.2). Para ter mais informações, consulte Conectar-se ao Amazon Athena com JDBC.	19 de abril de 2018
Preenchimento automático incluído para digitação de consultas no console do Athena.	Preenchimento automático incluído para digitação de consultas no console do Athena.	6 de abril de 2018
Recurso incluído para criar tabelas do Athena para arquivos de log do CloudTrail diretamente no console do CloudTrail.	Recurso incluído para criação automática de tabelas do Athena para arquivos de log do CloudTrail diretamente no console do CloudTrail. Para ter mais informações, consulte Usar o console do CloudTrail para criar uma tabela do Athena com logs do CloudTrail .	15 de março de 2018

Alteração	Descrição	Data de lançamento
Inclusão de suporte para descarga segura de dados intermediários em disco para consultas com `GROUP BY`.	Adicionada a capacidade de descarregar com segurança dados intermediários no disco para consultas com uso intensivo de memória que empregam a cláusula `GROUP BY`. Isso melhora a confiabilidade dessas consultas, evitando erros do tipo "Query resource exhausted" (Recurso de consulta esgotado). Para obter mais informações, consulte a nota de release de 2 de fevereiro de 2018.	2 de fevereiro de 2018
Inclusão de suporte ao Presto versão 0.172.	Mecanismo subjacente no Amazon Athena atualizado para uma versão baseada no Presto 0.172. Para obter mais informações, consulte a nota de release de 19 de janeiro de 2018.	19 de janeiro de 2018
Inclusão de suporte ao Driver ODBC.	Suporte adicional para conectar o Athena ao driver ODBC. Para obter informações, consulte Conectar-se ao Amazon Athena com ODBC.	13 de novembro de 2017
Suporte incluído para as regiões Ásia-Pacífico (Seul), Ásia-Pacífico (Mumbai) e Europa (Londres). Inclusão de suporte para consultar dados geoespaciais.	Suporte incluído para consultar dados geoespaciais e para as regiões Ásia-Pacífico (Seul), Ásia-Pacífico (Mumbai) e Europa (Londres). Para obter mais informações, consulte Consultar dados geospaciais e a página sobre Regiões da AWS e endpoints.	1 de novembro de 2017
Suporte incluído para Europa (Frankfurt).	Suporte incluído para Europa (Frankfurt). Para obter a lista de regiões compatíveis, consulte a página sobre Regiões da AWS e endpoints.	19 de outubro de 2017

Alteração	Descrição	Data de lançamento
Suporte incluído para consultas nomeadas do Athena com AWS CloudFormation.	Suporte incluído para criação de consultas nomeadas do Athena com AWS CloudFormation. Para obter mais informações, consulte AWS::Athena::NamedQuery no Guia do usuário do AWS CloudFormation.	3 de outubro de 2017
Suporte incluído para a região Ásia-Pacífico (Sydney).	Suporte incluído para a região Ásia-Pacífico (Sydney). Para obter a lista de regiões compatíveis, consulte a página sobre Regiões da AWS e endpoints.	25 de setembro de 2017
Uma seção foi adicionada a este guia sobre a consulta de logs de AWS service (Serviço da AWS) e de diferentes tipos de dados, incluindo mapas, matrizes, dados aninhados e dados contendo JSON.	Exemplos adicionados para Consulta de logs do AWS service (Serviço da AWS) e sobre a consulta de diferentes tipos de dados no Athena. Para ter mais informações, consulte Executar consultas SQL usando o Amazon Athena.	5 de setembro de 2017
O suporte adicionado para AWS Glue Data Catalog.	Integração incluída com o AWS Glue Data Catalog e um assistente de migração para atualizar do catálogo de dados gerenciados do Athena para o AWS Glue Data Catalog. Para obter mais informações, consulte Integração com AWS Glue e AWS Glue.	14 de agosto de 2017

Alteração	Descrição	Data de lançamento
Inclusão de suporte ao Grok SerDe.	Adicionado suporte para Grok SerDe, que oferece um padrão de correspondência mais simples para registros em arquivos de texto desestruturados, como logs. Para obter mais informações, consulte Grok SerDe. Atalhos de teclado adicionados para percorrer o histórico de consultas usando o console.	4 de agosto de 2017
Suporte incluído para a região Ásia-Pacífico (Tóquio).	Suporte incluído para as regiões Ásia-Pacífico (Tóquio) e Ásia-Pacífico (Singapura). Para obter a lista de regiões compatíveis, consulte a página sobre Regiões da AWS e endpoints.	22 de junho de 2017
Suporte incluído para a região Europa (Irlanda).	Suporte incluído para a região Europa (Irlanda). Para obter mais informações, consulte a página sobre Regiões da AWS e endpoints.	8 de junho de 2017
Uma API do Amazon Athena e o suporte para AWS CLI foram incluídos.	Uma API do Amazon Athena e o suporte para AWS CLI foram incluídos no Athena. Atualização do driver JDBC para a versão 1.1.0.	19 de maio de 2017
Suporte incluído para criptografia de dados do Amazon S3.	Suporte incluído para criptografia de dados do Amazon S3 e lançamento de uma atualização do driver JDBC (versão 1.0.1) com suporte à criptografia, melhorias e correções de bugs. Para ter mais informações, consulte Criptografia inativa.	4 de abril de 2017

Alteração	Descrição	Data de lançamento
Adicionado o SerDe do AWS CloudTrail.	Adicionado o SerDe do AWS CloudTrail, melhor performance, corrigidos problemas de partição. • O AWS CloudTrail SerDe foi substituído pelo Hive JSON SerDe para leitura de logs do CloudTrail. Para obter informações sobre a consulta de logs do CloudTrail, consulte Consultar os logs do AWS CloudTrail. • Melhorada a performance durante o exame de um grande número de partições. • Melhorada a performance na operação `MSCK Repair Table`. • Adicionada capacidade de consultar dados do Amazon S3 armazenados em regiões diferentes da região principal. Taxas de transferência de dados entre regiões padrão do Amazon S3 se aplicam, além de cobranças do Athena padrão.	24 de março de 2017
Suporte incluído para a região Leste dos EUA (Ohio).	Suporte incluído para Avro SerDe e OpenCSVSerDe para processar CSV, Leste dos EUA (Ohio) e edição em massa de colunas no assistente do console. Melhorada a performance em tabelas Parquet grandes.	20 de fevereiro de 2017
	O lançamento inicial do Manual do usuário do Amazon Athena.	Novembro de 2016

Glossário da AWS

Para obter a terminologia mais recente da AWS, consulte o AWSglossário na Glossário da AWSReferência.