用户指南

Amazon Athena

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

Amazon Athena: 用户指南

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

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Table of Contents

什么是 Amazon Athena？

Amazon Athena 是一种交互式查询服务，让您能够轻松使用标准 SQL 直接分析 Amazon Simple Storage Service (Amazon S3) 中的数据。只需在 AWS Management Console 中执行几项操作，即可将 Athena 指向 Amazon S3 中存储的数据，并开始使用标准 SQL 运行临时查询，然后在几秒钟内获得结果。

有关更多信息，请参阅 开始使用。

Amazon Athena 还可使用 Apache Spark 以交互方式轻松运行数据分析，无需规划、配置或管理资源。在 Athena 上运行 Apache Spark 应用程序时，您需要提交 Spark 代码以进行处理并直接接收结果。使用 Amazon Athena 控制台中简化的笔记本体验，以通过 Python 或 Athena 笔记本 API 开发 Apache Spark 应用程序。

有关更多信息，请参阅 Amazon Athena 上的 Apache Spark 入门。

Athena SQL 和 Amazon Athena 上的 Apache Spark 无服务器，因此您无需设置或管理任何基础设施，只需为运行的查询付费。Athena 可自动扩展（并行执行查询），因此，即使在数据集很大、查询很复杂的情况下也能很快获得结果。

主题

应在何时使用 Athena？

Amazon Athena 等查询服务、Amazon Redshift 等数据仓库以及 Amazon EMR 等复杂数据处理框架都能满足不同的需求和使用案例。以下指南可帮助您根据自己的需求选择一项或多项服务。

Amazon Athena

Athena 可帮助您分析在 Amazon S3 中存储的非结构化、半结构化和结构化数据。示例包括 CSV、JSON 或列式数据格式，如 Apache Parquet 和 Apache ORC。您可以使用 ANSI SQL 通过 Athena 运行临时查询，而无需将数据聚合或加载到 Athena 中。

Athena 与 Amazon QuickSight 集成，轻松实现数据可视化。您可以使用 Athena 生成报表，或借助商业智能工具或 SQL 客户端浏览数据（通过 JDBC 或 ODBC 驱动程序进行连接）。有关更多信息，请参阅《Amazon QuickSight 用户指南》中的什么是 Amazon QuickSight和 通过 ODBC 和 JDBC 驱动程序连接到 Amazon Athena。

Athena 与 AWS Glue Data Catalog 集成，后者为您在 Amazon S3 中的数据提供了持久元数据存储。这使您可以根据在您的整个 Amazon Web Services 账户中可用并与 AWS Glue 的 ETL 和数据发现功能集成在一起的中央元数据存储来创建表和在 Athena 中查询数据。有关更多信息，请参阅《AWS Glue 开发人员指南》中的与 AWS Glue 集成和 AWS Glue 是什么。

Amazon Athena 让您能够轻松地直接在 Amazon S3 中对数据运行交互式查询，而无需进行数据格式化或管理基础设施。例如，假设您需要对 Web 日志运行快速查询，以排查站点上的性能问题，则可以选择 Athena。Athena 的入门十分简单快速：只需为数据定义一个表，即可使用标准的 SQL 开始查询。

如果要对 Amazon S3 上的数据运行交互式的临时 SQL 查询，而无需管理任何基础设施或集群，则应使用 Amazon Athena。Amazon Athena 提供了在 Amazon S3 中运行临时数据查询的便捷方法，无需设置或管理任何服务器。

有关 Athena 利用或者与之集成的 AWS 服务 服务列表，请参阅 the section called “AWS 服务 与 Athena 的集成”。

Amazon EMR

与本地部署相比，Amazon EMR 让您能够简单且经济高效地运行高度分布式处理框架，例如 Hadoop、Spark 和 Presto。Amazon EMR 非常灵活 – 您可以运行自定义应用程序和代码，并且可以定义特定的计算、内存、存储和应用程序参数来优化分析要求。

除了运行 SQL 查询之外，Amazon EMR 还可以为应用程序运行各种横向扩展数据处理任务，例如机器学习、图表分析、数据转换、数据流式传输，以及几乎任何可以编码的内容。如果您使用自定义代码来处理和分析采用最新大数据处理框架（如 Spark、Hadoop、Presto 或 Hbase）的极大数据集，则应使用 Amazon EMR。Amazon EMR 让您能够完全控制集群的配置以及在集群上安装的软件。

您可以使用 Amazon Athena 来查询您使用 Amazon EMR 处理的数据。Amazon Athena 支持许多与 Amazon EMR 相同的数据格式。Athena 的数据目录与 Hive 元数据仓兼容。如果您使用 EMR 并且已经拥有 Hive 元数据仓，则可以在 Amazon Athena 上运行 DDL 语句并立即查询数据，而不会影响您的 Amazon EMR 任务。

Amazon Redshift

当您需要将来自许多不同源（例如库存系统、财务系统和零售销售系统）的数据汇集到通用格式并长时间存储时，诸如 Amazon Redshift 之类的数据仓库是理想选择。如果您需要根据历史数据构建复杂的业务报告，则建议选择诸如 Amazon Redshift 之类的数据仓库。Amazon Redshift 中的查询引擎经过优化，尤其善于运行对大量超大数据库表进行交集运算的复杂查询。如果您需要对高度结构化的数据运行查询，并且要对大量的超大表进行交集运算，则建议选择 Amazon Redshift。

有关何时使用 Athena 的更多信息，请参阅以下资源：

- 入门资源中心中有关 AWS 分析服务决策指南
- Amazon Athena 常见问题中何时使用 Athena 而不是其他大数据服务
- Amazon Athena 概览
- Amazon Athena 功能
- Amazon Athena 常见问题
- Amazon Athena 博客文章

AWS 服务 与 Athena 的集成

您可以使用 Athena 查询本节中所列 AWS 服务 中的数据。要查看每个服务支持的区域，请参阅 Amazon Web Services 一般参考 中的区域和端点。

AWS 服务 与 Athena 集成

- 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
- Simple Storage Service (Amazon S3) 清单
- AWS Step Functions

- AWS Systems Manager 清单
- Amazon Virtual Private Cloud

请参阅下面几节，了解有关每个整合的信息。

AWS CloudFormation

容量预留

参考主题：《AWS CloudFormation 用户指南》中的 AWS::Athena::CapacityReservation

使用提供的名称和请求的数据处理单元数量指定容量预留。有关更多信息，请参阅《Amazon Athena 用户指南》中的 管理查询处理容量 和 Amazon Athena API 参考中的 CreateCapacityReservation。

数据目录

参考主题：《AWS CloudFormation 用户指南》中的 AWS::Athena::DataCatalog

指定 Athena 数据目录，包括名称、描述、类型、参数和标签。有关更多信息，请参阅《Amazon Athena 用户指南》中的 了解表、数据库和数据目录 和 Amazon Athena API 参考中的 CreateDataCatalog。

命名查询

参考主题：《AWS CloudFormation 用户指南》中的 AWS::Athena::NamedQuery

通过 AWS CloudFormation 指定命名查询，然后在 Athena 中运行它们。命名查询允许您将查询名称映射到查询，然后从 Athena 控制台将其作为保存的查询运行。有关更多信息，请参阅《Amazon Athena 用户指南》中的 使用已保存的查询 和 Amazon Athena API 参考中的 CreateNamedQuery。

预准备语句

参考主题：《AWS CloudFormation 用户指南》中的 AWS::Athena::PreparedStatement

指定一个用于 Athena 中的 SQL 查询的预编译语句。预准备语句包含参数占位符，其值在执行时提供。有关更多信息，请参阅《Amazon Athena 用户指南》中的 使用参数化查询 和 Amazon Athena API 参考中的 CreatePreparedStatement。

工作组

参考主题：《AWS CloudFormation 用户指南》中的 AWS::Athena::WorkGroup

使用 AWS CloudFormation 指定 Athena 工作组。使用 Athena 工作组可将您或您的组的查询，与同一账户中的其他查询隔离开来。有关更多信息，请参阅《Amazon Athena 用户指南》中的使用工作组控制查询访问和成本 和 Amazon Athena API 参考中的 CreateWorkGroup。

Amazon CloudFront

参考主题：查询 Amazon CloudFront 日志

使用 Athena 查询 Amazon CloudFront 日志。有关使用 CloudFront 的更多信息，请参阅《Amazon CloudFront 开发人员指南》。

AWS CloudTrail

参考主题：查询 AWS CloudTrail 日志

将 Athena 与 CloudTrail 日志结合使用是加强对 AWS 服务活动进行分析的强有力方法。例如，您可以使用查询来确定趋势，并根据属性（如源 IP 地址或用户）进一步隔离活动。您可以自动创建表，用于直接从 CloudTrail 控制台查询日志，并将这些表用于在 Athena 中运行查询。有关更多信息，请参阅 使用 CloudTrail 控制台为 CloudTrail 日志创建 Athena 表 。

Amazon DataZone

参考主题：在 Athena 中使用 Amazon DataZone

使用 Amazon DataZone 跨组织边界大规模共享、搜索和发现数据。DataZone 简化了您在 Athena、AWS Glue 和 AWS Lake Formation 等 AWS 分析服务中的体验。如果您在不同的数据来源中有大量数据，则可使用 Amazon DataZone 根据人员、数据和工具分组构建业务用例。

在 Athena 中，您可以使用查询编辑器来访问和查询 DataZone 环境。有关更多信息，请参阅 在 Athena 中使用 Amazon DataZone。

Elastic Load Balancing

参考主题：查询 Application Load Balancer 日志

通过查询 Application Load Balancer 日志，您可以查看进出 Elastic Load Balancing 实例和后端应用程序的流量来源、延迟和传输字节。有关更多信息，请参阅 查询 Application Load Balancer 日志。

参考主题：查询经典负载均衡器日志

查询经典负载均衡器日志，分析和了解传入和传出 Elastic Load Balancing 实例和后端应用程序的流量模式。您可以查看流量来源、延迟和传输字节。有关更多信息，请参阅为 ELB 日志创建表。

Amazon EMR Studio

参考主题：在 EMR Studio 中使用 Amazon Athena SQL 编辑器

您可以在 EMR Studio 中使用 Athena 来开发和运行交互式查询。这样的话，您可以通过 Spark、Scala 和其他工作负载所用的相同 Amazon EMR 接口，使用 EMR Studio 在 Athena 上进行 SQL 分析。利用 EMR Studio 中的 Athena 集成，您可以执行以下任务：

- 执行 Athena SQL 查询
- 查看查询结果
- 查看查询历史记录
- 查看保存的查询
- 执行参数化查询
- 查看数据目录的数据库、表和视图

Amazon EMR Studio 不提供以下 Athena 功能：

- 管理功能，例如创建或更新 Athena 工作组、数据来源或容量预留
- Athena for Spark 或 Spark 笔记本
- DataZone 集成
- Step Functions

在所有可以使用 EMR Studio 和 Athena 的 AWS 区域 中，都可以使用与 Athena 的 EMR Studio 集成。有关在 EMR Studio 中使用 Athena 的更多信息，请参阅《Amazon EMR 管理指南》中的 Use the Amazon Athena SQL editor in EMR Studio。

AWS Glue Data Catalog

参考主题：与 AWS Glue 集成

Athena 与 AWS Glue Data Catalog 集成，后者为您在 Amazon S3 中的数据提供了持久元数据存储。这使您可以根据在您的整个 Amazon Web Services 账户中可用并与 AWS Glue 的 ETL 和数据发现功能集成在一起的中央元数据存储来创建表和在 Athena 中查询数据。有关更多信息，请参阅与 AWS Glue 集成 和《AWS Glue 开发人员指南》中的什么是 AWS Glue？

AWS Identity and Access Management (IAM)

参考主题：Amazon Athena 的操作

您可在 IAM 权限策略中使用 Athena API 操作。有关更多信息，请参阅 Amazon Athena 的操作和 Athena 中的 Identity and Access Management。

Amazon QuickSight

参考主题：通过 ODBC 和 JDBC 驱动程序连接到 Amazon Athena

Athena 与 Amazon QuickSight 集成，轻松实现数据可视化。您可以使用 Athena 生成报表，或借助商业智能工具或 SQL 客户端浏览数据（通过 JDBC 或 ODBC 驱动程序进行连接）。有关 Amazon QuickSight 的更多信息，请参阅《Amazon QuickSight 用户指南》中的什么是 Amazon QuickSight。有关将 JDBC 和 ODBC 驱动程序与 Athena 一起使用的信息，请参阅通过 ODBC 和 JDBC 驱动程序连接到 Amazon Athena。

Simple Storage Service (Amazon S3) 清单

参考主题：《Amazon Simple Storage Service 用户指南》中的使用 Athena 查询清单

您可以使用 Amazon Athena 通过标准 SQL 来查询 Amazon S3 清单。出于业务、合规性和法规要求，您可以使用 Amazon S3 清单来审计和报告对象的复制和加密状态。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 Amazon S3 清单。

AWS Step Functions

参考主题：《AWS Step Functions 开发人员指南》中的使用 Step 函数调用 Athena

使用 AWS Step Functions 调用 Athena。AWS Step Functions 可以使用 Amazon States Language 直接控制 AWS 服务 选择。您可以将 Step 函数与 Athena 结合使用，以启动和停止查询执行、获取查询结果、运行临时或计划数据查询，以及从 Amazon S3 中的数据湖检索结果。Step Functions 角色必须有权使用 Athena。有关更多信息，请参见 AWS Step Functions 开发人员指南。

视频：使用 AWS Step Functions 编排 Amazon Athena 查询

以下视频说明了如何使用 Amazon Athena 和 AWS Step Functions 运行定期计划的 Athena 查询并生成相应的报告。

使用 AWS Step Functions 编排 Amazon Athena 查询

有关使用 Step 函数和 Amazon EventBridge 编排 AWS Glue DataBrew、Athena 和 Amazon QuickSight 的示例，请参阅 AWS 大数据博客中的使用 AWS Step Functions 编排 AWS Glue DataBrew 任务和 Amazon Athena 查询。

AWS Systems Manager 清单

参考主题：《AWS Systems Manager 用户指南》中的查询多个区域和账户的清单数据

AWS Systems Manager Inventory 与 Amazon Athena 集成，可帮助您从多个 AWS 区域 和账户中查询清单数据。有关更多信息，请参阅 AWS Systems Manager 用户指南。

Amazon Virtual Private Cloud

参考主题：查询 Amazon VPC 流日志

Amazon Virtual Private Cloud 流日志捕获有关在 VPC 中传入和传出网络接口的 IP 流量的信息。查询 Athena 中的日志，调查网络流量模式，并识别 Amazon VPC 网络中的威胁和风险。有关 Amazon VPC 的更多信息，请参阅《Amazon VPC 用户指南》。

设置

如果您已注册 Amazon Web Services，则可以立即开始使用 Amazon Athena。如果您尚未注册 AWS 或需要入门帮助，请确保完成以下任务。

注册 AWS 账户

如果您还没有 AWS 账户，请完成以下步骤来创建一个。

注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册 AWS 账户时，系统将会创建一个 AWS 账户根用户。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

注册过程完成后，AWS 会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册 AWS 账户 后，请保护好您的 AWS 账户根用户，启用 AWS IAM Identity Center，并创建一个管理用户，以避免使用根用户执行日常任务。

保护您的 AWS 账户根用户

1. 选择根用户并输入您的 AWS 账户电子邮件地址，以账户拥有者身份登录 AWS Management Console。在下一页上，输入您的密码。

要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的以根用户身份登录。

2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅《IAM 用户指南》中的为 AWS 账户根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关如何使用 IAM Identity Center 目录 作为身份源的教程，请参阅《AWS IAM Identity Center 用户指南》中的使用默认的 IAM Identity Center 目录 配置用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  要获取使用 IAM Identity Center 用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的 登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。

2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

授权以编程方式访问

如果用户需要在 AWS Management Console 之外与 AWS 交互，则需要编程式访问权限。授予编程式访问权限的方法取决于访问 AWS 的用户类型。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
人力身份 （在 IAM Identity Center 中管理的用户）	使用临时凭证签署向 AWS CLI、AWS SDK 或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关 AWS CLI 的更多信息，请参阅《AWS Command Line Interface 用户指南》中的配置 AWS CLI 以使用 AWS IAM Identity Center。 • 有关 AWS SDK、工具和 AWS API 的更多信息，请参阅《AWS SDK 和工具参考指南》中的 IAM Identity Center 身份验证。
IAM	使用临时凭证签署向 AWS CLI、AWS SDK 或 AWS API 发出的编程请求。	按照《IAM 用户指南》中将临时凭证用于 AWS 资源中的说明进行操作。
IAM	（不推荐使用） 使用长期凭证签署向 AWS CLI、AWS SDK 或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关 AWS CLI 的更多信息，请参阅《AWS Command Line Interface 用户指南》中的使用 IAM 用户凭证进行身份验证。 • 有关 AWS SDK 和工具的更多信息，请参阅《AWS SDK 和工具参考指南》中的使用长期凭证进行身份验证。 • 有关 AWS API 的更多信息，请参阅《IAM 用户指

哪个用户需要编程式访问权限？	目的	方式
		南》中的管理 IAM 用户的访问密钥。

为 Athena 附加托管式策略

Athena 托管策略授予 Athena 功能使用权限。您可以将这些托管策略附加到一个或多个 IAM 角色，用户可以担任这些角色以便使用 Athena。

IAM 角色是可在账户中创建的一种具有特定权限的 IAM 身份。IAM 角色类似于 IAM 用户，因为它是一个 AWS 身份，具有确定其在 AWS 中可执行和不可执行的操作的权限策略。但是，角色旨在让需要它的任何人代入，而不是唯一地与某个人员关联。此外，角色没有关联的标准长期凭证（如密码或访问密钥）。相反，当您代入角色时，它会为您提供角色会话的临时安全凭证。

有关角色的更多信息，请参阅《IAM 用户指南》中的 IAM 角色和创建 IAM 角色。

要创建向 Athena 授予访问权限的角色，您需要将 Athena 托管策略附加到该角色。Athena 有两个托管策略：`AmazonAthenaFullAccess` 和 `AWSQuicksightAthenaAccess`。这些策略向 Athena 授予查询 Amazon S3 的权限以及代表您将您的查询结果写入单独存储桶的权限。要查看适用于 Athena 的这些策略的内容，请参阅 Amazon Athena 的 AWS 托管策略。

有关向角色附加 Athena 托管策略的步骤，请遵循《IAM 用户指南》中的添加 IAM 身份权限（控制台）进行操作，并将 `AmazonAthenaFullAccess` 和 `AWSQuicksightAthenaAccess` 托管策略添加到您创建的角色。

> **Note**
>
> 您可能需要额外的权限才能访问 Amazon S3 中的底层数据集。如果您不是账户拥有者或以其他方式限制了对某个存储桶的访问权限，请与存储桶拥有者联系以使用基于资源的存储桶策略授予访问权限，或与您的账户管理员联系以使用基于角色的策略授予访问权限。有关更多信息，请参阅 对 Amazon S3 的访问权限。如果数据集或 Athena 查询结果是加密的，您可能需要额外的权限。有关更多信息，请参阅 静态加密。

访问 Athena

您可使用 AWS Management Console、JDBC 或 ODBC 连接、Athena API、Athena CLI、AWS 开发工具包或 AWS Tools for Windows PowerShell 访问 Athena。

- 要开始将 Athena SQL 与控制台结合使用，请参阅 开始使用。
- 要开始创建与 Jupyter 兼容的笔记本和使用 Python 的 Apache Spark 应用程序，请参阅 在 Amazon Athena 中使用 Apache Spark。
- 要了解如何使用 JDBC 或 ODBC 驱动程序，请参阅通过 JDBC 连接到 Amazon Athena 和通过 ODBC 连接到 Amazon Athena。
- 要使用 Athena API，请参阅 Amazon Athena API 参考。
- 要使用 CLI，请安装 AWS CLI，然后从命令行键入 `aws athena help` 以查看可用的命令。有关可用命令的信息，请参阅 Amazon Athena 命令行参考。
- 要使用 AWS SDK for Java 2.x，请参阅 AWS SDK for Java 2.x API 参考 的 Athena 部分，Github.com 上的 Athena Java V2 examples（Athena Java V2 示例），以及 AWS SDK for Java 2.x 开发人员指南。
- 要使用 AWS SDK for .NET，请参阅 AWS SDK for .NET API 参考中的 `Amazon.Athena` 命名空间，Github 上的 .NET Athena 示例，以及《AWS SDK for .NET 开发人员指南》。
- 要使用 AWS Tools for Windows PowerShell，请参阅 AWS Tools for PowerShell - Amazon Athena cmdlet 引用，AWS Tools for PowerShell 门户页面和《AWS Tools for Windows PowerShell 用户指南》。
- 有关可以以编程方式连接到的 Athena 服务端点的信息，请参阅 Amazon Web Services 一般参考 中的 Amazon Athena 端点和限额。

使用 Athena SQL

您可以使用 Athena SQL，借助 AWS Glue Data Catalog（外部 Hive 元存储）或者通过各种预构建连接器对其他数据来源进行联合查询，从而在 Amazon S3 中就地查询数据。

您也可以：

- 使用 Athena 的 JDBC 和 ODBC 驱动程序 连接到商业智能工具和其他应用程序。
- 查询 AWS 服务日志。
- 查询 Apache Iceberg 表，包括时间旅行查询和 Apache Hudi 数据集。
- 查询地理空间数据。
- 使用来自 Amazon SageMaker 的机器学习推理进行查询。
- 使用您自己的用户定义函数进行查询。
- 使用分区投影可加快对高度分区表的查询处理，并自动执行分区管理。

主题

了解表、数据库和数据目录

在 Athena 中，目录、数据库和表是为底层源数据定义架构的元数据定义的容器。

Athena 使用以下术语来指代数据对象的层次结构：

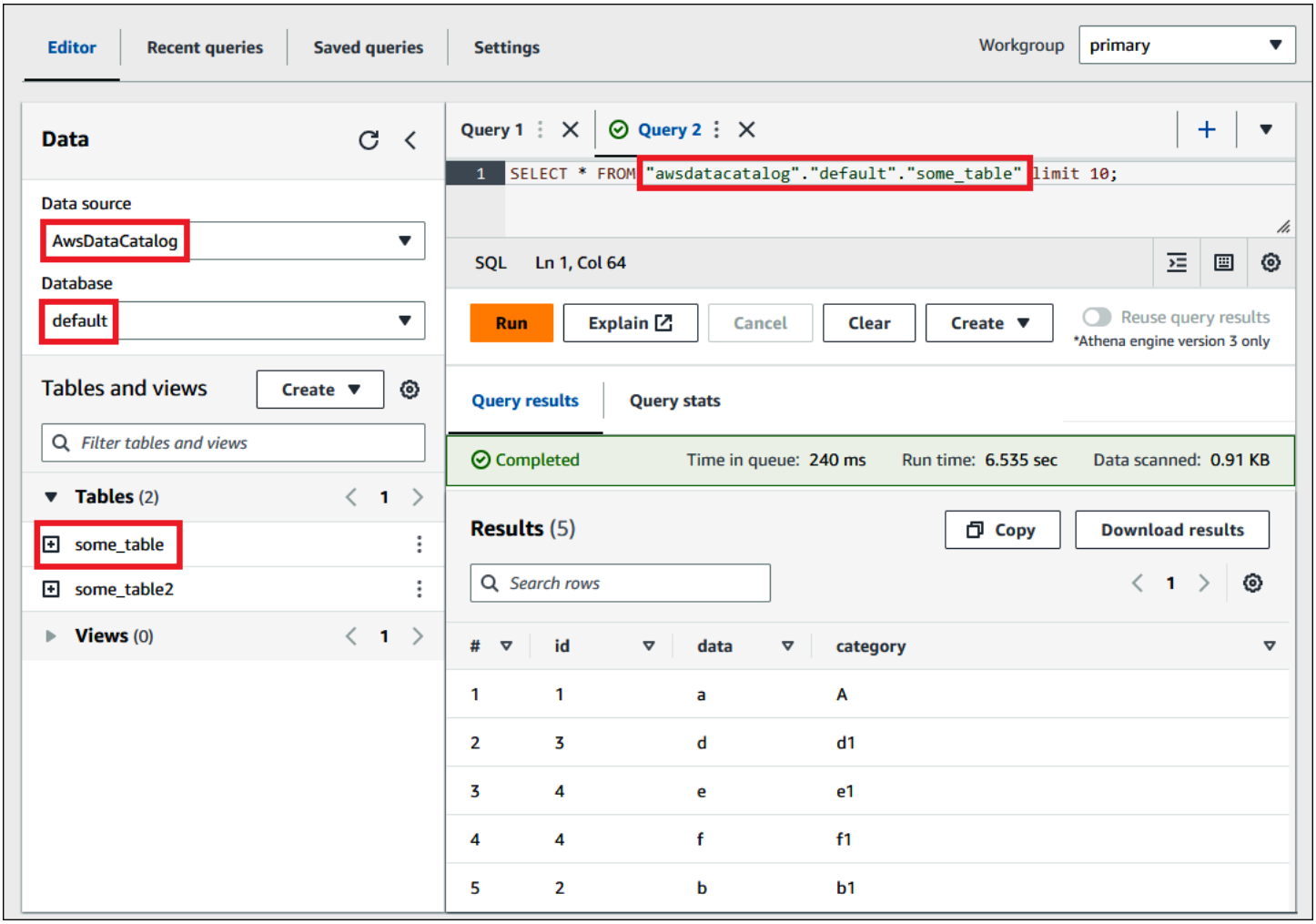
- 数据来源 - 一组数据库
- 数据库 - 一组表
- 表 - 按一组行或列组织的数据

有时，这些对象也会用替代但等效的名称来指代，如下所示：

- 数据来源有时也称为目录。
- 数据库有时也称为架构。

> **Note**
>
> 在您与 Athena 一起使用的联合数据来源中，此术语可能有所不同。有关更多信息，请参阅 Athena 和联合表名限定词。

Athena 控制台中的以下示例查询使用 `awsdatacatalog` 数据来源、数据库 `default` 和表 `some_table`。



对于每个数据集，Athena 中都需要存在一个表。表中的元数据让 Athena 知道数据在 Amazon S3 中的位置，并指定数据的结构，例如列名称、数据类型和表的名称。数据库是表的逻辑分组，并且只保存数据集的元数据和架构信息。

对于您要查询的每个数据集，Athena 必须有一个底层表，以便用来获取和返回查询结果。因此，在查询数据之前，必须在 Athena 中注册表。当您自动或手动创建表时，会发生注册。

您可以使用 AWS Glue 爬网程序自动创建表。有关 AWS Glue 和爬网程序的更多信息，请参阅与 AWS Glue 集成。当 AWS Glue 创建表时，它会将其注册到自己的 AWS Glue Data Catalog 中。Athena 使用 AWS Glue Data Catalog 来存储和检索此元数据，以便当您运行查询来分析底层数据集时使用它。

无论采用何种方式创建表，表的创建过程都会向 Athena 注册数据集。此注册发生在 AWS Glue Data Catalog 中，并允许 Athena 对数据运行查询。在 Athena 查询编辑器中，使用标签 `AwsDataCatalog` 引用此目录（或数据来源）。

创建表后，您可以使用 SQL SELECT 语句来查询该表，包括获取源数据的特定文件位置。您的查询结果存储在 Amazon S3 中指定的查询结果位置。

您可通过 Amazon Web Services 账户访问到 AWS Glue Data Catalog。其他 AWS 服务 可以共享 AWS Glue Data Catalog，因此，您可以使用 Athena 查看在整个企业内创建的数据库和表，反之亦然。

- 要手动创建表，请执行以下操作：
  - 使用 Athena 控制台运行创建表向导。
  - 使用 Athena 控制台在查询编辑器中编写 Hive DDL 语句。
  - 使用 Athena API 或 CLI，通过 DDL 语句运行 SQL 查询。
  - 使用 Athena JDBC 或 ODBC 驱动程序。

当您手动创建表和数据库时，Athena 会在后台使用 HiveQL 数据定义语言 (DDL) 语句（例如`CREATE TABLE`、`CREATE DATABASE` 和 `DROP TABLE`）以在 AWS Glue Data Catalog 中创建表和数据库。

要开始使用，您可以使用 Athena 控制台中的教程，也可以参阅 Athena 文档中的分步指南。

- 要在 Athena 控制台中使用本教程，请选择控制台右上角的信息图标，然后选择教程选项卡。
- 有关在 Athena 查询编辑器中创建表和编写查询的分步教程，请参阅 开始使用。

开始使用

本教程将指导您完成使用 Amazon Athena 来查询数据。您将根据 Amazon Simple Storage Service 中存储的示例数据创建表、查询表，并检查查询的结果。

本教程使用的是实时资源，因此，您需要为您运行的查询付费。您无需为本教程使用的位置中的示例数据付费，但如果您将自己的数据文件上载到 Amazon S3，则会收取费用。

先决条件

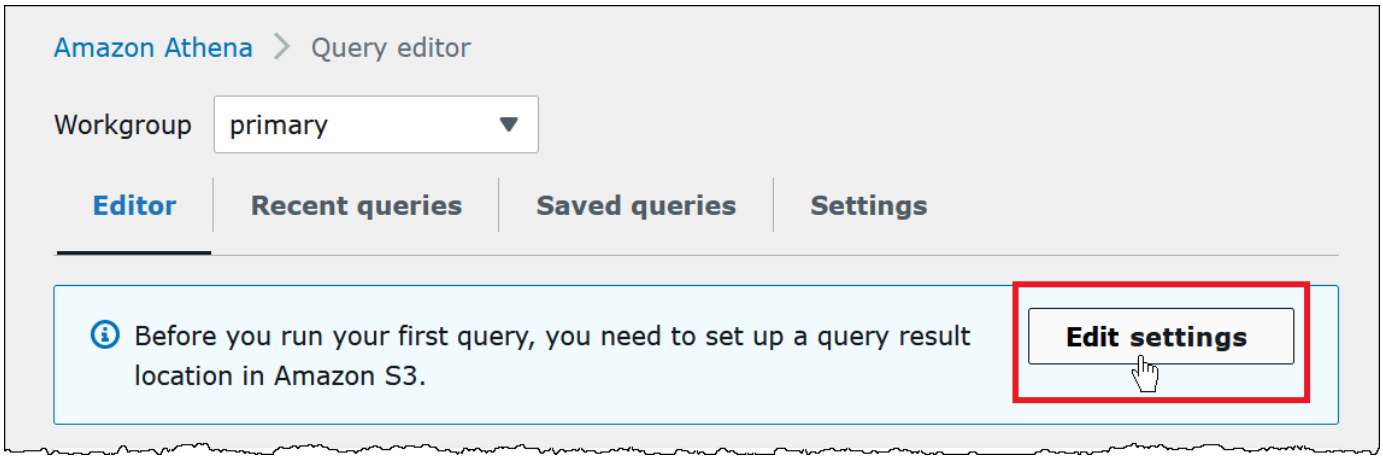
- 如果您尚未完成此操作，注册 AWS 账户。
- 使用与您在 Athena 中使用的相同 AWS 区域（例如，美国西部（俄勒冈州））和账户，按照相应的步骤在 Amazon S3 中创建存储桶以保存您的 Athena 查询结果。您需要将此存储桶配置为查询输出位置。

步骤 1：创建数据库

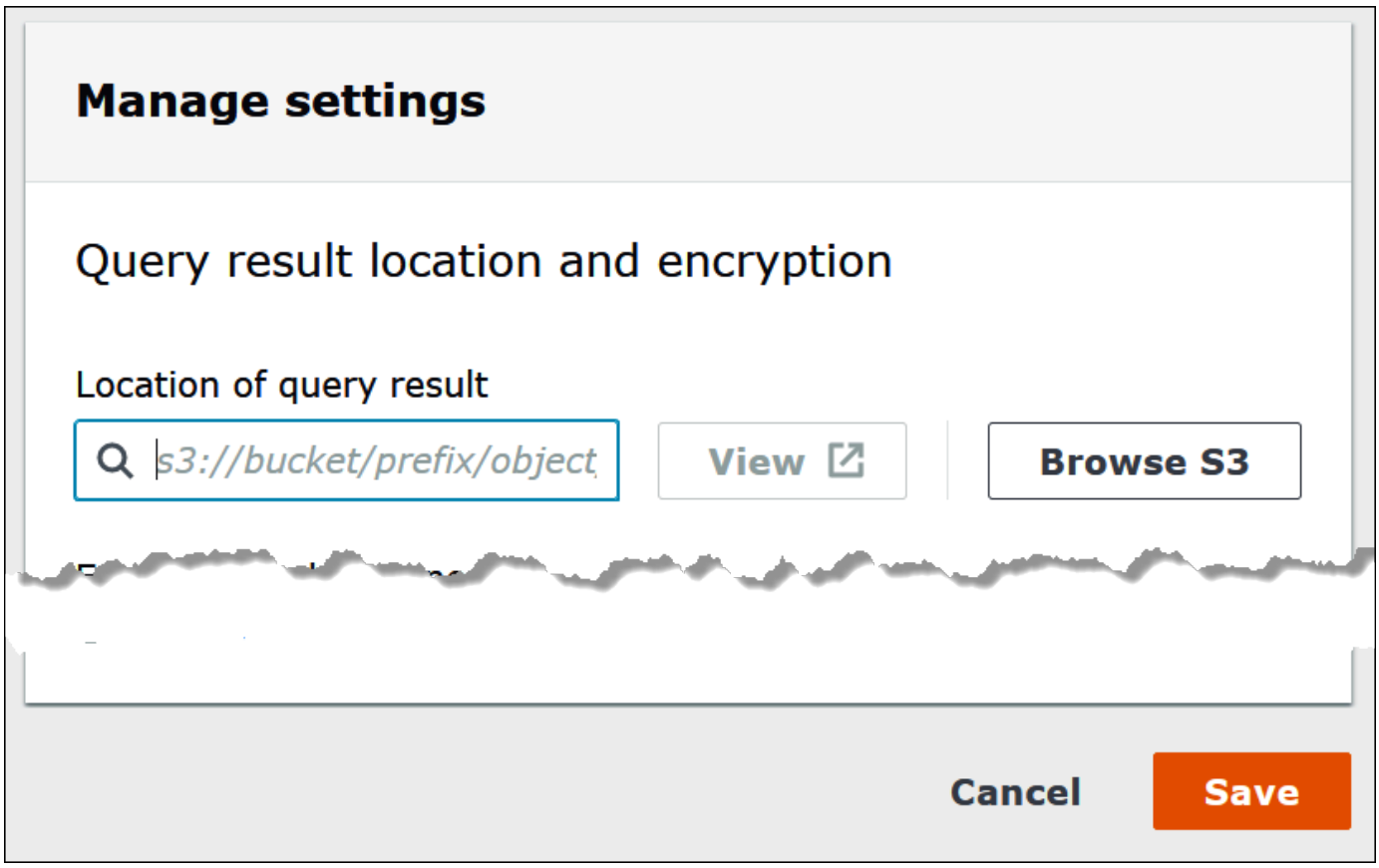
首先，您需要在 Athena 中创建一个数据库。

创建 Athena 数据库

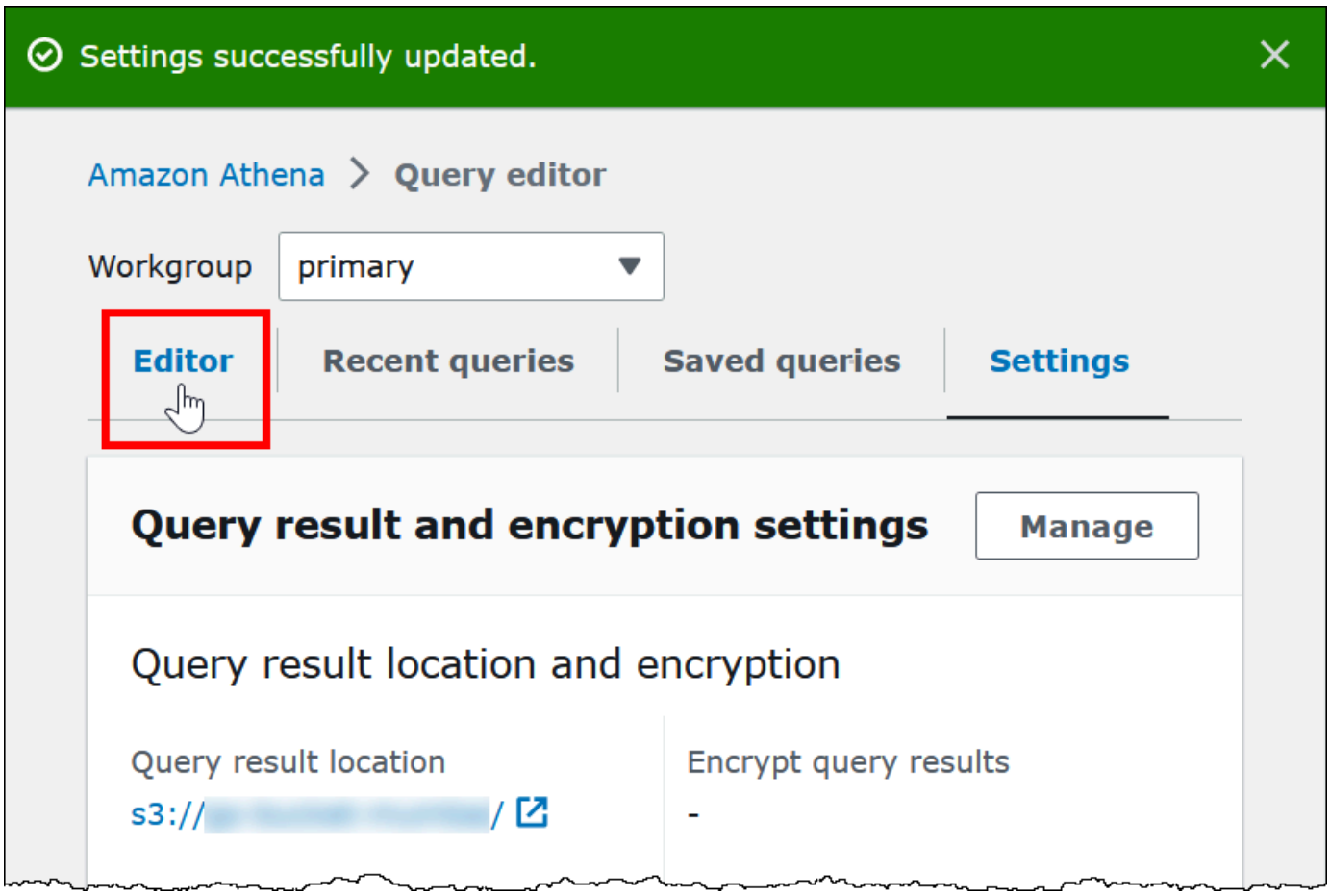
1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果这是您首次在目前的 AWS 区域 中访问 Athena 控制台，选择 Explore the query editor（浏览查询编辑器）以打开查询编辑器。否则，Athena 会在查询编辑器中打开。
3. 选择 Edit Settings（编辑设置）以在 Amazon S3 中设置查询结果位置。



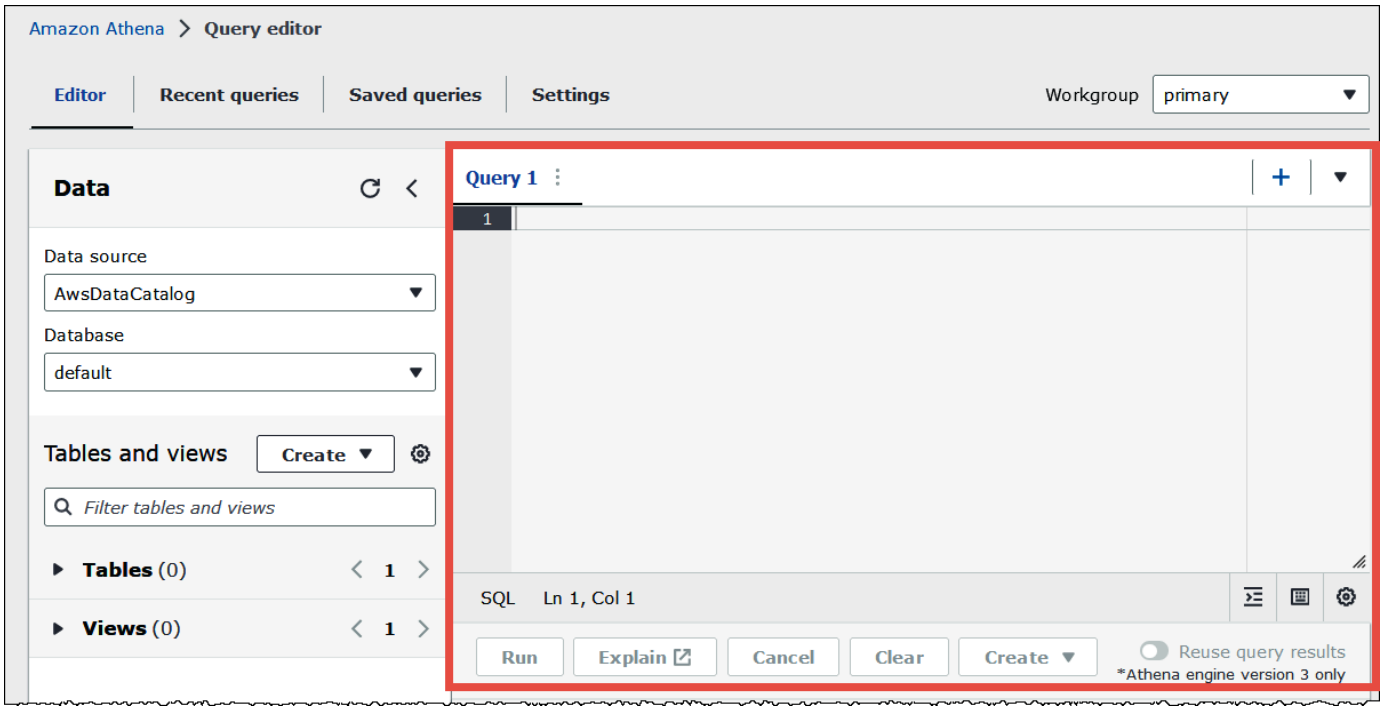
4. 对于 Manage settings（管理设置），执行以下操作之一：
   - 在 Location of query result（查询结果位置）文本框中，输入您在 Amazon S3 中为查询结果创建的存储桶路径。在路径前添加前缀 `s3://`。
   - 选择 Browse S3（浏览 S3），选择您为当前区域创建的 Amazon S3 存储桶，然后选择 Choose（选择）。



5. 选择保存。
6. 选择 Editor（编辑器）以切换到查询编辑器。



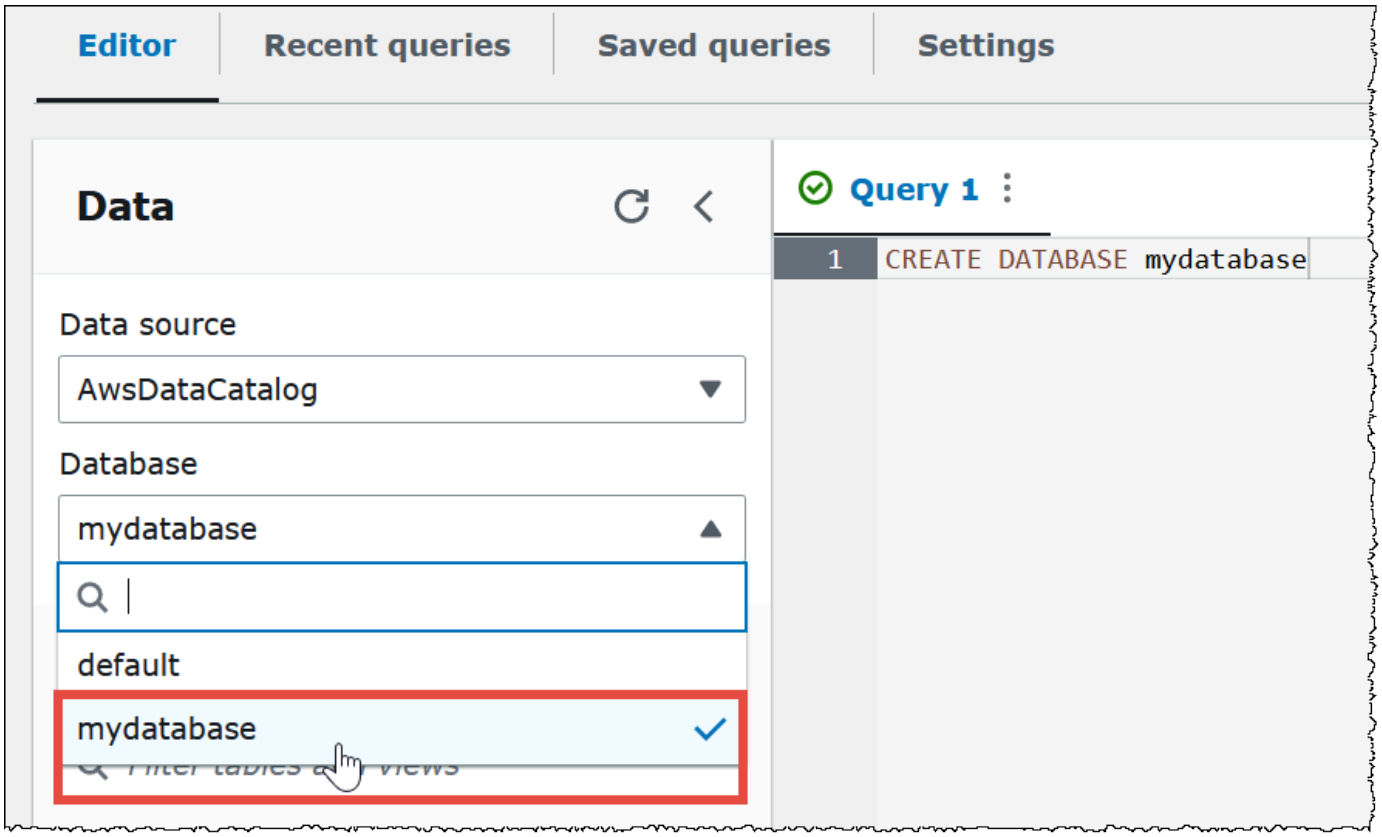
7. 在导航窗格的右侧，您可以使用 Athena 查询编辑器输入和运行查询和语句。



8. 要创建一个名为 mydatabase 的数据库，请输入以下 CREATE DATABASE 语句。

```
CREATE DATABASE mydatabase
```

9. 选择 Run（运行）或者按 **Ctrl+ENTER**。
10. 在左侧的 Database（数据库）列表中，选择 mydatabase 使其成为当前数据库。



步骤 2：创建表

现在，您有了数据库，可以为其创建一个 Athena 表。您创建的表将基于位置 s3://athena-examples-*myregion*/cloudfront/plaintext/ 中的示例 Amazon CloudFront 日志数据，其中 *myregion* 是您当前的 AWS 区域。

示例日志数据采用制表符分隔值(TSV)格式，这意味着将制表符用作分隔符来分隔字段。数据类似于以下示例。为了便于阅读，摘录中的制表符已转换为空格，并缩短了最后一个字段。

```
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[...]
```

要让 Athena 能够读取此数据，可以创建如下所示的简单 CREATE EXTERNAL TABLE 语句。创建表的语句定义映射到数据的列，指定如何分隔数据，并指定包含示例数据的 Amazon S3 位置。请注意，由于 Athena 要扫描文件夹中的所有文件，LOCATION 子句会指定 Amazon S3 文件夹位置，而不是特定文件。

暂勿使用此示例，因为其有一个重要限制，稍后将对此进行解释。

```
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/';
```

此示例创建了一个名为 cloudfront_logs 的表，并为每个字段指定名称和数据类型。这些字段将成为表中的列。由于 date 是一个预留关键字，因此使用反引号(`)字符进行转义。ROW FORMAT DELIMITED 意味着 Athena 将使用一个名为 LazySimpleSerDe 的默认库来完成解析数据的实际工作。该示例还指定字段以制表符分隔(FIELDS TERMINATED BY '\t')，并且文件中的每条记录以换行符(LINES TERMINATED BY '\n)结束。最后，LOCATION 子句指定要读取的实际数据所在的 Amazon S3 中的路径。

如果您有自己的选项卡或逗号分隔数据，只要字段不包含嵌套信息，就可以使用如上述示例所示的 CREATE TABLE 语句。不过，如果有类似 ClientInfo 这样的列，其中包含使用不同分隔符的嵌套信息，则需要采用不同的方法。

从 ClientInfo 字段中提取数据

查看示例数据，以下为最终字段 ClientInfo 的完整示例：

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

正如您所看到的，此字段是多值的。由于上述所示的示例 CREATE TABLE 语句将制表符指定为字段分隔符，所以无法将 ClientInfo 字段中的单独组件分解为单独的列。因此，需要一个新的 CREATE TABLE 语句。

要根据 ClientInfo 字段内的值创建列，可以使用包含正则表达式组的正则表达式（regex）。您指定的正则表达式组将成为单独的表列。要在您的 CREATE TABLE 语句中使用正则表达式，请使用类似下面的语法。此语法指示 Athena 使用 Regex SerDe 库和您指定的正则表达式。

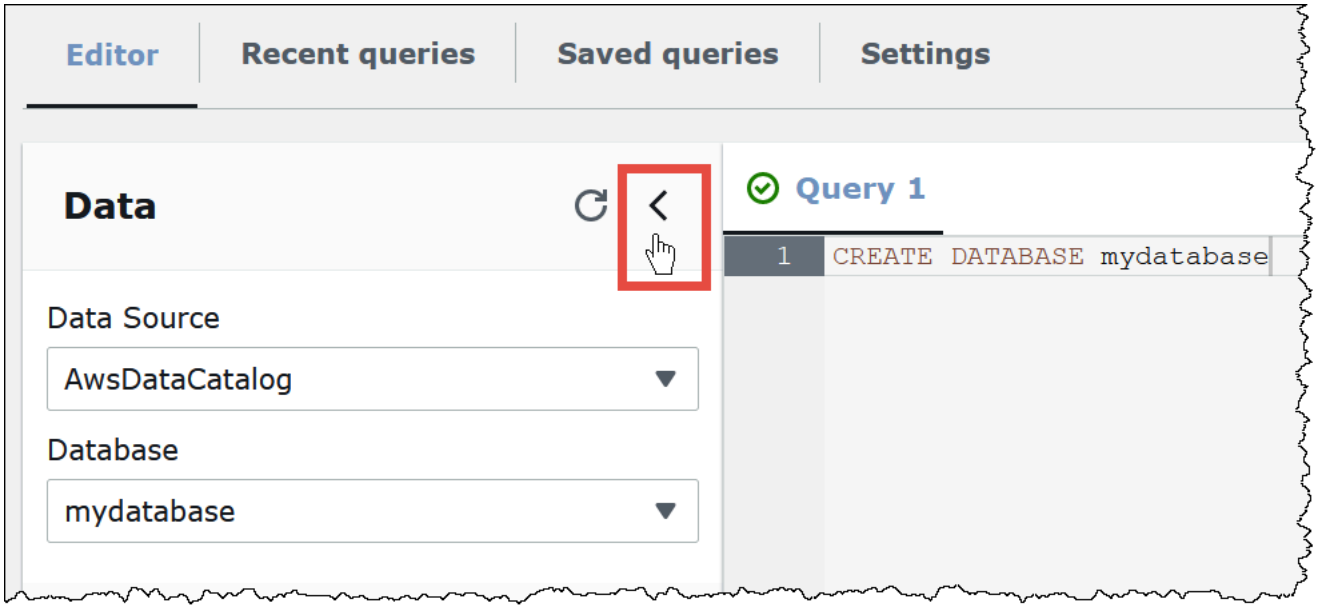
```
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
  WITH SERDEPROPERTIES ("input.regex" = "regular_expression")
```

正则表达式可用于根据复杂的 CSV 或 TSV 数据创建表，但可能难以编写和维护。幸运的是，还有其他库可以用于 JSON、Parquet 和 ORC 等格式。有关更多信息，请参阅 支持的 SerDes 和数据格式。

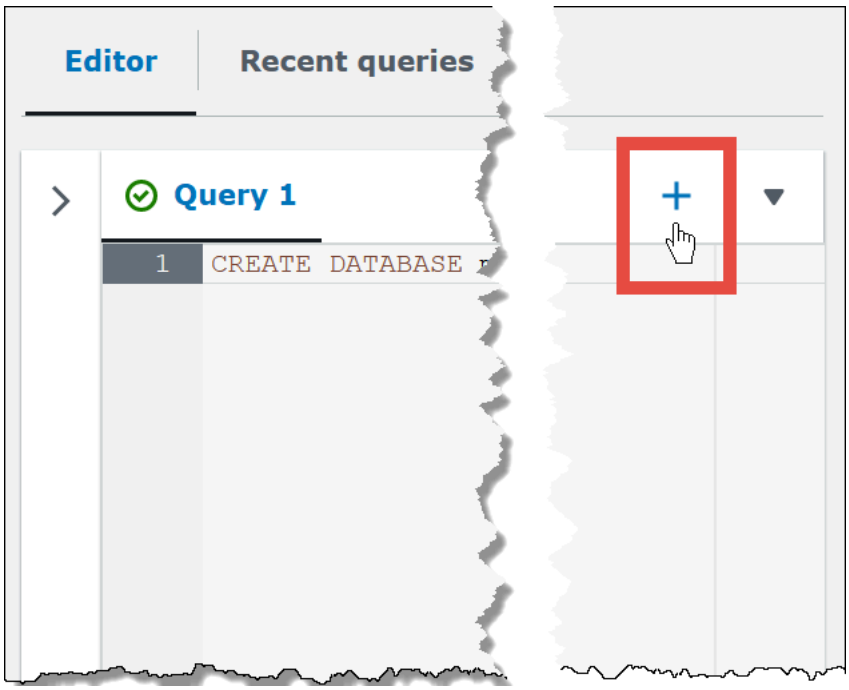
现在您已准备就绪，可在 Athena 查询编辑器中创建表。已为您准备好 CREATE TABLE 语句和正则表达式。

在 Athena 中创建表

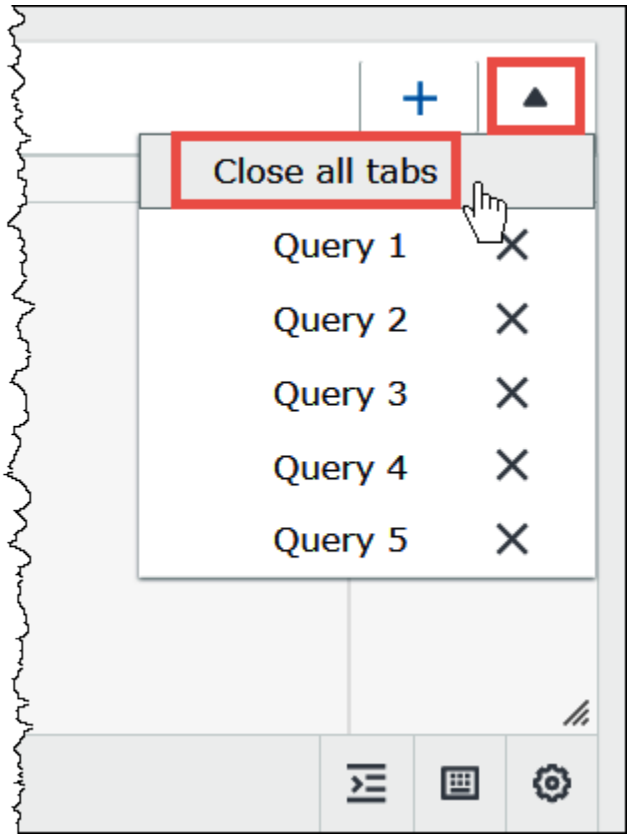
1. 在导航窗格中，对于 Database（数据库），请确保选择了 mydatabase。
2. 要在查询编辑器中扩大空间，您可以选择箭头图标来折叠导航窗格。



3. 要创建新查询的选项卡，请在查询编辑器中选择加号 (+)。您最多可以同时打开十个查询选项卡。



4. 要关闭一个或多个查询选项卡，请选择加号旁边的箭头。要一次关闭所有选项卡，请选择箭头，然后选择 Close all tabs（关闭所有选项卡）。



5. 在查询窗格中，输入以下 CREATE EXTERNAL TABLE 语句。正则表达式将操作系统、浏览器和浏览器版本信息从日志数据中的 ClientInfo 字段划分出来。

```
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. 在 LOCATION 语句中，将 *myregion* 替换为您当前使用的 AWS 区域（例如，us-west-1）。
7. 选择运行。

   表 cloudfront_logs 已创建，并显示在 mydatabase 数据库的 Tables(表)列表中。

步骤 3：查询数据

现在，您已根据 Amazon S3 中的数据在 Athena 中创建了 cloudfront_logs 表，您可以针对该表运行 SQL 查询，并在 Athena 中查看结果。有关在 Athena 中使用 SQL 的更多信息，请参阅 Athena 的 SQL 参考。

运行查询

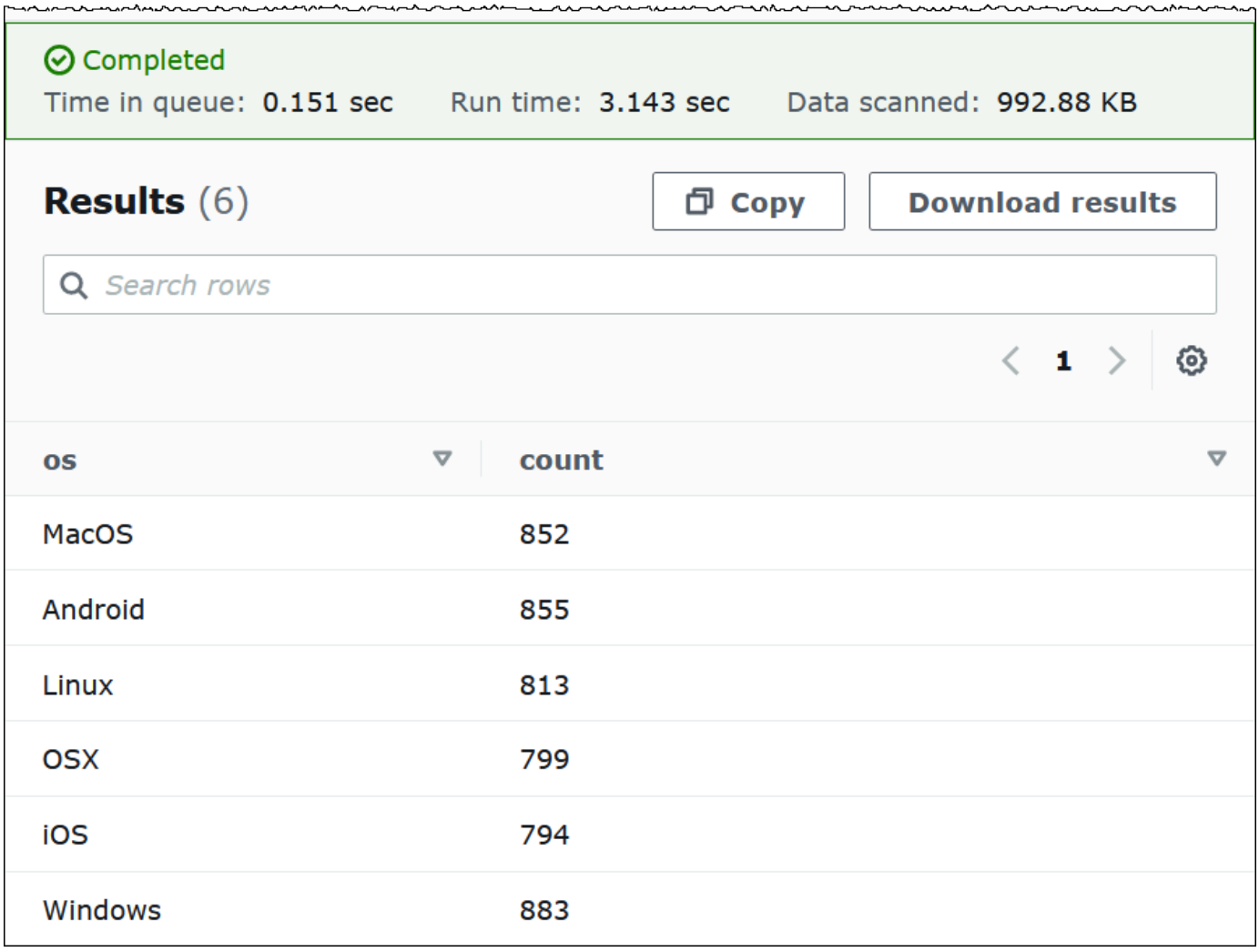
1. 选择加号(+)以打开新的查询选项卡，然后在查询窗格中输入以下 SQL 语句。

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

```
GROUP BY os
```

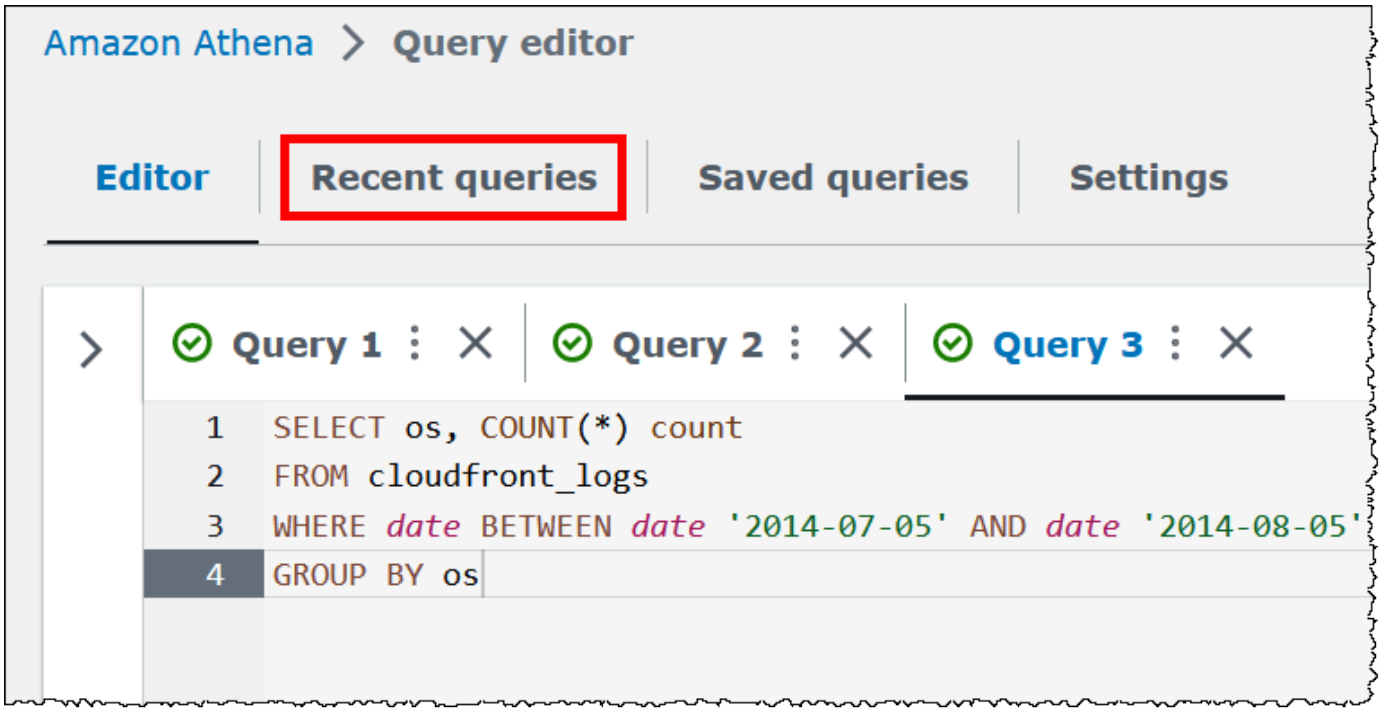
2. 选择运行。

结果与以下内容类似：

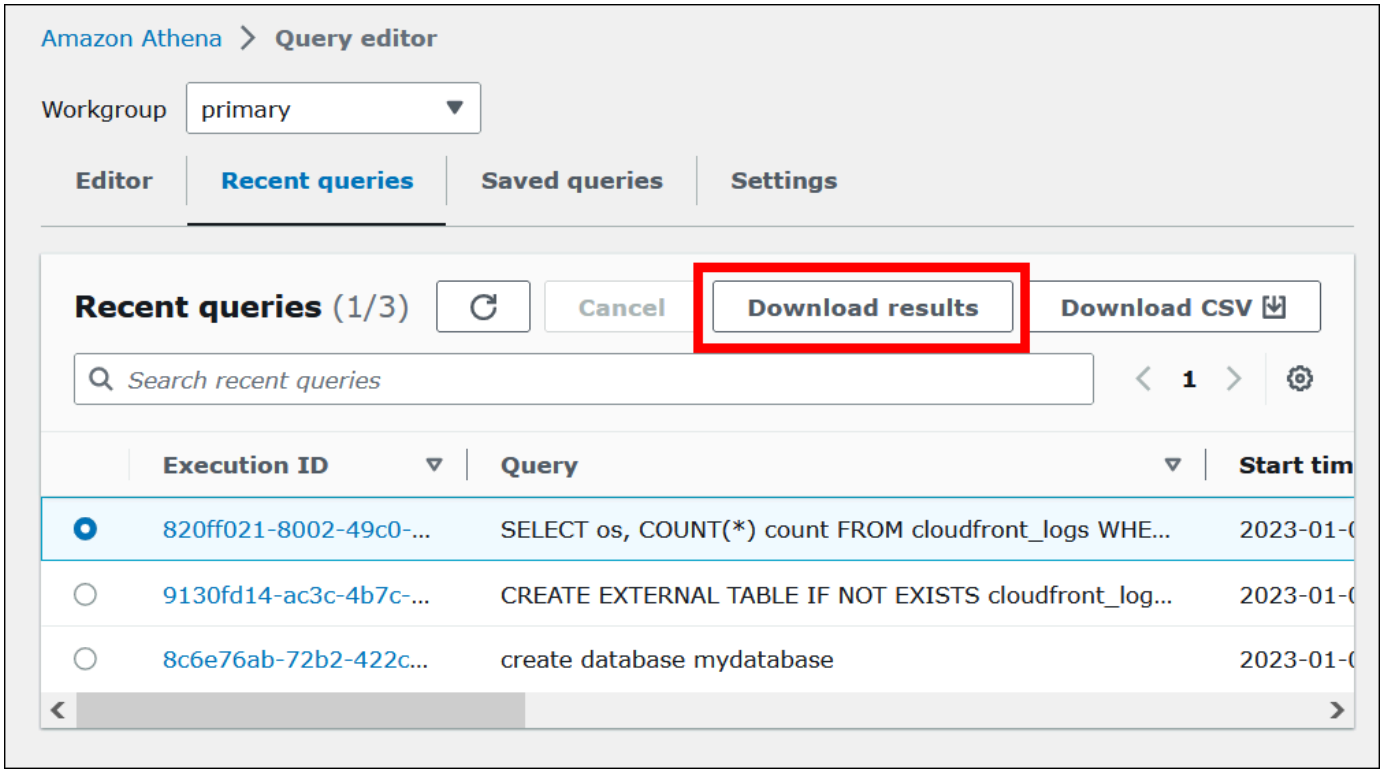


3. 要将查询结果保存到 `.csv` 文件，选择 Download results（下载结果）。

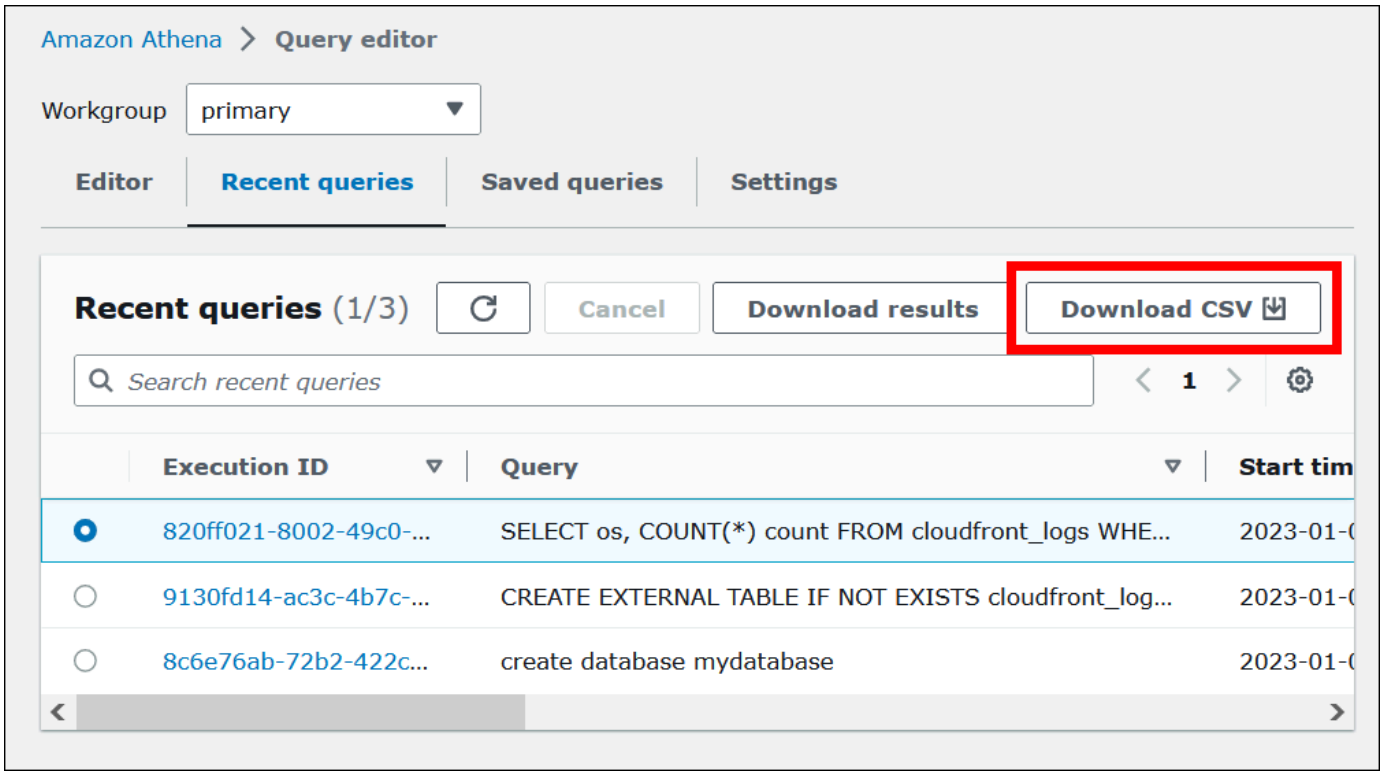
4. 要查看或运行之前的查询，选择 Recent queries（最近的查询）选项卡。



5. 要从 Recent queries（最近的查询）选项卡下载以前查询的结果，请选择查询，然后选择 Download results（下载结果）。查询保留 45 天。



6. 要将一个或多个最近的 SQL 查询字符串下载到 CSV 文件，请选择 Download CSV（下载 CSV）。



有关更多信息，请参阅 使用查询结果、最近查询和输出文件。

保存查询

您可以使用某个名称来保存您在查询编辑器中创建或编辑的查询。Athena 会将这些查询存储在 Saved queries（已保存的查询）选项卡中。您可以使用 Saved queries（已保存的查询）选项卡来回调、运行、重命名或删除已保存的查询。有关更多信息，请参阅 使用已保存的查询。

键盘快捷键和提前输入建议

Athena 查询编辑器提供了许多键盘快捷键，可用于运行查询、格式化查询、行操作以及查找和替换等操作。有关更多信息和快捷键完整列表，请参阅 AWS 大数据博客中的通过在 Amazon Athena 查询编辑器中使用键盘快捷键提高工作效率。

Athena 查询编辑器支持提前输入代码建议，以实现更快的查询创作体验。为了帮助您以更高的准确性和更高的效率编写 SQL 查询，该编辑器提供了以下功能：

- 在您输入时，将实时显示关键字、局部变量、片段和目录项的建议。

- 当您在数据库名称或表名称后面输入一个点时，该编辑器会显示一系列的表或列，方便您从中进行选择。
- 当您将光标悬停在片段建议上时，摘要会显示该片段的语法和用法的简要概述。
- 为了提高代码的可读性，还更新了关键字及其突出显示规则，使其与 Trino 和 Hive 的最新语法保持一致。

该功能已默认启用。要启用或禁用该功能，使用该查询编辑器窗口右下角的代码编辑器首选项（齿轮图标）。

连接到其他数据来源

本教程使用 Amazon S3 中 CSV 格式的数据来源。有关将 Athena 与 AWS Glue 结合使用的信息，请参阅 使用 AWS Glue 连接到 Simple Storage Service（Amazon S3）中的数据来源。您可以使用 ODBC 和 JDBC 驱动程序、外部 Hive 元存储和 Athena 数据来源连接器，将 Athena 连接到各种数据来源。有关更多信息，请参阅 连接到数据源。

连接到数据源

您可以使用 Amazon Athena 查询数据集中的不同位置和以不同格式存储的数据。此数据集可能采用 CSV、JSON、Avro、Parquet 或其他格式。

您在 Athena 中用于运行查询的表格和数据库基于元数据。元数据是与数据集中的底层数据有关的数据。元数据描述数据集的方式称为架构。例如，表名称、表中的列名称以及每列的数据类型都是架构，保存为元数据，用于描述底层数据集。在 Athena 中，我们将用于整理元数据的系统称为数据目录或元数据仓。数据集和用于描述它的数据目录的组合称为数据源。

元数据与底层数据集的关系取决于您所用的数据源类型。MySQL、PostgreSQL 和 SQL Server 等关系数据源将元数据与数据集紧密集成。在这些系统中，在写入数据时通常会写入元数据。其他数据源（如使用 Hive 构建的数据源）允许您在读取数据集时即时定义元数据。数据集可以采用多种格式，例如 CSV、JSON、Parquet 或 Avro。

Athena 原生支持 AWS Glue Data Catalog。AWS Glue Data Catalog 是在其他数据集和数据源（如 Amazon S3、Amazon Redshift 和 Amazon DynamoDB）之上构建的数据目录。您还可以使用各种连接器将 Athena 连接到其他数据源。

主题

- 与 AWS Glue 集成
- 将 Athena 数据连接器用于外部 Hive 元数据仓

- 使用 Amazon Athena 联合查询
- 用于访问数据目录的 IAM policy
- 管理数据来源
- 在 Athena 中使用 Amazon DataZone

与 AWS Glue 集成

AWS Glue 是一项完全托管式 ETL（提取、转换和加载）AWS 服务。其关键能力之一是对数据进行分析和分类。您可以使用 AWS Glue 爬网程序自动从 Amazon S3 中的数据推断数据库和表架构，并将关联的元数据存储在 AWS Glue Data Catalog。

Athena 使用 AWS Glue Data Catalog 在您的亚马逊云科技账户中存储和检索 Amazon S3 数据的表元数据。通过表元数据，Athena 查询引擎可以了解如何查找、读取和处理您要查询的数据。

要在 AWS Glue Data Catalog 中创建数据库和表架构，您可以在 Athena 中对数据源运行 AWS Glue 爬网程序，或者您可以直接在 Athena 查询编辑器中运行数据定义语言 (DDL) 查询。然后，使用您创建的数据库和表架构，您可以使用 Athena 中的数据操作 (DML) 查询来查询数据。

您可以从不属于您自己的账户中注册 AWS Glue Data Catalog。在您为 AWS Glue 配置所需的 IAM 权限之后，就可以使用 Athena 运行跨账户查询。有关更多信息，请参阅 授予 AWS Glue 数据目录跨账户访问权限。

有关 AWS Glue Data Catalog 的更多信息，请参阅《AWS Glue 开发人员指南》中的 AWS Glue 中的数据目录和爬网程序。

需单独支付 AWS Glue 的费用。有关更多信息，请参阅AWS Glue 定价。

主题

- 使用 AWS Glue 连接到 Simple Storage Service（Amazon S3）中的数据来源
- 从另一个账户注册 AWS Glue Data Catalog
- 将 Athena 与 AWS Glue 结合使用时的最佳实践
- 使用 AWS CLI 重新创建 AWS Glue 数据库及其表

使用 AWS Glue 连接到 Simple Storage Service（Amazon S3）中的数据来源

Athena 可以使用 AWS Glue Data Catalog 连接到 Amazon S3 中存储的数据，以便存储表和列名等元数据。建立连接后，数据库、表和视图将显示在 Athena 的查询编辑器中。

要定义供 AWS Glue 使用的架构信息，您可以设置 AWS Glue 爬网程序来自动检索信息，也可以手动添加表并输入架构信息。

创建 AWS Glue 爬网程序

您可以通过在 Athena 控制台中启动，然后以集成方式使用 AWS Glue 控制台来创建爬网程序。创建爬网程序时，您可以在 Amazon S3 中指定要爬取的数据位置。

在 AWS Glue 中从 Athena 控制台开始创建爬网程序

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 在查询编辑器中，选择 Tables and views（表和视图）旁的 Create（创建），然后选择 AWS Glue crawler（爬网程序）。
3. 在 AWS Glue 控制台的 Add crawler (添加爬网程序) 页面上，按照步骤创建爬网程序。有关更多信息，请参阅本指南中的使用 AWS Glue 爬网程序和《AWS Glue 开发人员指南》中的填充 AWS Glue Data Catalog。

> **Note**
>
> Athena 不承认您为 AWS Glue 爬网程序指定的排除模式。例如，如果您有一个 Amazon S3 存储桶，其中包含 `.csv` 和 `.json` 文件，并且您从爬网程序中排除了 `.json` 文件时，Athena 会查询两组文件。要避免这种情况，请将要排除的文件放置在其他位置。

使用表单添加表

以下过程演示了如何使用 Athena 控制台通过根据 S3 存储桶数据创建表表单添加表。

使用表单添加表并输入架构信息

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 在查询编辑器中，选择 Tables and views（表和视图）旁边的 Create（创建），然后选择 S3 bucket data（S3 存储桶数据）。
3. 在根据 S3 存储桶数据创建表表单中，对于 Table name（表名称），输入表的名称。
4. 对于 Database configuration（数据库配置），选择现有数据库或新建一个数据库。
5. 对于 Location of Input Data Set（输入数据集的位置），在 Amazon S3 中指定包含要处理的数据集的文件夹的路径。请勿在路径中包含文件的名称。Athena 会扫描指定文件夹中的所有文件。如果您的数据已经分区（例如，

s3://DOC-EXAMPLE-BUCKET/logs/year=2004/month=12/day=11/），则仅输入基本路径（例如，s3://DOC-EXAMPLE-BUCKET/logs/）。

6. 对于 Data Format（数据格式），请选择以下选项：

   - 对于Table type（表格类型），选择 Apache Hive、Apache Iceberg 或 Delta Lake。Athena 的默认表类型是 Apache Hive。有关在 Athena 中查询 Apache Iceberg 表的信息，请参阅使用 Apache Iceberg 表。有关在 Athena 中使用 Delta Lake 表的信息，请参阅 查询 Linux Foundation Delta Lake 表。
   - 对于 File format（文件格式），选择您的数据将采用的文件或日志格式。
     - 对于 Text File with Custom Delimiters (带自定义分隔符的文本文件) 选项，请指定 Field terminator (字段终止符)（即列分隔符）。您也可以指定用于标记数组类型结束的 Collection terminator（集合终止符）或用于标记映射数据类型结束的 Collection terminator（集合终止符）。
   - SerDe 库 – SerDe（串行器解串器）库用于解析特定的数据格式，以便 Athena 可以为其创建表。对于大多数格式，系统会为您选择默认 SerDe 库。对于以下格式，请根据您的需要选择库：
     - Apache Web 日志 – 选择 RegexSerDe 或 GrokSerDe 库。对于 RegexSerDe，请在 Regex definition（正则表达式定义）框中提供一个正则表达式。对于 GrokSerDe，为 `input.format` SerDe 属性提供一个已命名正则表达式系列。已命名正则表达式比正则表达式更便于阅读和维护。有关更多信息，请参阅 查询存储在 Amazon S3 中的 Apache 日志。
     - CSV – 如果以逗号分隔的数据不包含用双引号括起的值或者使用 `java.sql.Timestamp` 格式，请选择 LazySimpleSerDe。如果数据包含引号或为使用 UNIX 数字格式的 `TIMESTAMP`（例如，`1564610311`）时，请选择 OpenCSVSerDe。有关更多信息，请参阅用于 CSV、TSV 和自定义分隔文件的 LazySimpleSerDe 和用于处理 CSV 的 OpenCSVSerDe。
     - JSON – 选择 OpenX 或 Hive JSON SerDe 库。这两种格式都要求每个 JSON 文档均位于单行文本中，并且不使用换行字符分隔字段。OpenX SerDe 提供了一些额外的属性。有关这些属性的更多信息，请参阅OpenX JSON SerDe。有关 Hive SerDe 的多信息，请参阅 Hive JSON SerDe。

   有关在 Athena 中使用 SerDe 库的更多信息，请参阅 支持的 SerDes 和数据格式。

7. 对于 SerDe properties（SerDe 属性），请根据您使用的 SerDe 库以及您的需求添加、编辑或移除属性及值。

   - 要添加 SerDe 属性，请选择Add SerDe property（添加 SerDe 属性）。

- 在 Name（名称）字段中，输入属性的名称。
- 在 Value（值）字段中，输入属性的值。
- 要移除 SerDe 属性，请选择 Remove（移除）。

8. 对于 Table properties（表属性），请根据您的需要选择或编辑表属性。

   - 对于 Write compression（写入压缩），选择一个压缩选项。写入压缩选项以及可用压缩选项的可用性取决于数据格式。有关更多信息，请参阅 Athena 压缩支持。
   - 对于 Encryption（加密），如果基础数据已在 Amazon S3 中加密，则选择 Encrypted data set（加密数据集）。此选项在 `CREATE TABLE` 语句中会将 `has_encrypted_data` 表属性设置为 true。

9. 对于 Column details（列详细信息），输入要添加到表中的列名称和数据类型。

   - 要添加更多列（一次添加一列），请选择 Add a column (添加列)。
   - 要快速添加更多列，请选择 Bulk add columns (批量添加列)。在文本框中，输入以逗号分隔的列列表，格式为 *`column_name data_type`*, *`column_name data_type`*[,...]，然后选择 Add（添加）。

10. （可选）对于 Partition details（分区详情），添加一个或多个列名称和数据类型。分区会根据列值将相关数据组合在一起，这可帮助减少每次查询扫描的数据量。有关分区的信息，请参阅 在 Athena 中对数据进行分区。

11. （可选）对于 Bucketing（分桶），您可以指定包含要组合在一起的行的一列或多列，然后将这些行放入多个桶中。这让您能够在指定分桶列值时仅查询要读取的桶。

    - 对于 Buckets（桶），选择一个或多个具有大量唯一值（例如，主键）且经常用于筛选查询中数据的列。
    - 对于 Number of buckets（桶数量），输入可优化文件大小的数字。有关更多信息，请参阅 AWS 大数据博客中的 Top 10 Performance Tuning Tips for Amazon Athena（Amazon Athena 的十大性能优化技巧）。
    - 要指定分桶列，`CREATE TABLE` 语句将使用以下语法：

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

      > **Note**
      >
      > Bucketing（分桶）选项不适用于 Iceberg 表类型。

12. Preview table query（预览表查询）方框显示您在表单中输入的信息所生成的 CREATE TABLE 语句。无法直接编辑预览语句。要更改语句，请修改预览部分以上的表单字段，或在查询编辑器中直接创建语句，而不使用表单。
13. 选择 Create table（创建表），在查询编辑器中运行生成的语句并创建表。

从另一个账户注册 AWS Glue Data Catalog

您可以使用 Athena 的跨账户 AWS Glue 目录功能来从您自己的账户以外的账户注册 AWS Glue 目录。在您为 AWS Glue 配置所需的 IAM 权限并将目录注册为 Athena `DataCatalog` 资源后，您可以使用 Athena 运行跨账户查询。有关配置所需权限的信息，请参阅 授予 AWS Glue 数据目录跨账户访问权限。

以下过程说明如何使用 Athena 控制台将 AWS Glue Data Catalog 在不属于您的 Amazon Web Services 账户中配置为数据源。

要从另一个账户注册 AWS Glue Data Catalog

1. 请按 授予 AWS Glue 数据目录跨账户访问权限 中的步骤操作，以确保您有权查询其他账户中的数据目录。
2. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
3. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。

4. 选择 Data Source（数据源）。
5. 选择右上角的 Connect data source（连接数据来源）。

6. 在 Choose a data source（选择数据来源）页面上，对于Data sources（数据来源），选择 S3 - AWS Glue Data Catalog，然后选择 Next（下一步）。
7. 在输入数据来源详细信息页面的 AWS Glue Data Catalog 部分中，对于选择 AWS Glue Data Catalog，选择 AWS Glue Data Catalog 中的其他账户。
8. 对于 Data source details（数据源详细信息），提供以下信息：
   - Data source name（数据源名称）– 输入要在 SQL 查询中使用的名称，以引用其他账户中的数据目录。
   - Description（描述）–（可选）输入其他账户中数据目录的描述。
   - Catalog ID（目录编号）– 输入数据目录所属账户的 12 位 Amazon Web Services 账户 ID。Amazon Web Services 账户 ID 即是目录 ID。
9. （可选）对于 Tags（标签），输入要与数据源关联的键-值对。有关标签的更多信息，请参阅 为 Athena 资源添加标签。
10. 选择下一步。
11. 在 Review and create（审核和创建）页面中，检查数据来源的详细信息，然后选择 Create data source（创建数据来源）。Data source details（数据来源详细信息）页面列出了所注册数据目录的数据库和标签。
12. 选择 Data Source（数据源）。您注册的数据目录在 Data source name（数据来源名称）列中列出。
13. 要查看或编辑有关数据目录的信息，请选择该目录，然后选择Actions（操作）、Edit（编辑）。
14. 要删除新数据目录，请选择该目录，然后选择 Actions（操作）、Delete（删除）。

有关更多信息，请参阅 AWS 大数据博客中的使用 Amazon Athena 查询跨账户 AWS Glue Data Catalog。

将 Athena 与 AWS Glue 结合使用时的最佳实践

在将 Athena 与 AWS Glue Data Catalog 配合使用时，可使用 AWS Glue 创建要在 Athena 中查询的数据库和表（架构），也可以使用 Athena 创建架构，然后将其用于 AWS Glue 和相关服务。本主题提供了在使用上述任一方法时的注意事项和最佳实践。

在后台，Athena 使用 Trino 来处理 DML 语句，使用 Hive 来处理创建和修改架构的 DDL 语句。借助这些技术，有几个可遵循的惯例，以使 Athena 和 AWS Glue 很好地协作。

本主题内容

数据库、表和列名称

当您在 AWS Glue 中创建架构以便在 Athena 中进行查询时，请考虑以下事项：

- AWS Glue 中可接受的数据库名称、表名以及列名的字符必须是 UTF-8 字符串。字符串长度不得少于 1 个字节，也不得超过 255 个字节。其中可以使用的字符包括空格，并由以下单行字符串模式定义字符：

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

- 目前，AWS Glue 正则表达式模式允许在名称的开头添加前导空格。但这些前导空格可能难以检测且创建后可能导致可用性问题，所以请避免创建带有前导空格的对象名称。
- 如果您使用 AWS::Glue::Database AWS CloudFormation 模板创建 AWS Glue 数据库而未指定数据库名称，AWS Glue 会自动以与 Athena 不兼容的 *resource_name-random_string* 格式生成数据库名称。
- 您可以使用 AWS Glue Catalog Manager 重命名列，但不重命名表名称或数据库名称。要解决此限制，必须使用旧数据库的定义来创建具有新名称的数据库。然后，使用旧数据库中的表定义在新数据

库中重新创建表。为此，您可以使用 AWS CLI 或 AWS Glue 软件开发工具包。要查看步骤，请参阅使用 AWS CLI 重新创建 AWS Glue 数据库及其表。

有关 AWS Glue 中数据库和表的更多信息，请参阅《AWS Glue Developer Guide》中的 Databases 和 Tables。

使用 AWS Glue 爬网程序

AWS Glue 爬网程序可帮助发现数据集的架构，并在 AWS Glue Data Catalog 中将其注册为表。爬网程序会遍历您的数据并确定架构。此外，爬网程序还可以检测并注册分区。有关更多信息，请参阅《AWS Glue 开发人员指南》中的定义爬网程序。可以从 Athena 查询成功抓取的数据中的表。

> **Note**
>
> Athena 不承认您为 AWS Glue 爬网程序指定的排除模式。例如，如果您有一个 Amazon S3 存储桶，其中包含 `.csv` 和 `.json` 文件，并且您从爬网程序中排除了 `.json` 文件时，Athena 会查询两组文件。要避免这种情况，请将要排除的文件放置在其他位置。

安排爬网程序以保持 AWS Glue Data Catalog 和 Simple Storage Service (Amazon S3) 同步

AWS Glue 爬网程序可以设置为按计划或按需运行。有关更多信息，请参阅《AWS Glue 开发人员指南》中的基于时间的任务和爬网程序安排。

如果您的数据在固定时间到达分区表，则可以设置 AWS Glue 爬网程序按计划运行以检测和更新表分区。这样就不需要运行耗时长且昂贵的 `MSCK REPAIR` 命令或手动运行 `ALTER TABLE ADD PARTITION` 命令。有关更多信息，请参阅《AWS Glue 开发人员指南》中的表分区。

将多个数据源和爬网程序结合使用

当 AWS Glue 爬网程序扫描 Amazon S3 并检测到多个目录时，它会使用启发法来确定表的根在目录结构中的位置，以及表的分区所在的目录。在某些情况下，如果在两个或更多目录中检测到的架构相似，则爬网程序可能将它们视为分区而不是单独的表。一种帮助爬网程序发现单个表的方法是将每个表的根目录添加为爬网程序的数据存储。

以下示例是 Amazon S3 中的分区：

```
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
```

如果 `table1` 和 `table2` 的架构类似，并且单个数据源在 AWS Glue 中设置为 `s3://DOC-EXAMPLE-BUCKET/folder1/`，则爬网程序可能创建一个具有两个分区列的表：一个分区列包含 `table1` 和 `table2`，另一个分区列包含 `partition1` 到 `partition5`。

要让 AWS Glue 爬网程序创建两个单独的表，请将爬网程序设置为具有两个数据源 `s3://DOC-EXAMPLE-BUCKET/folder1/table1/` 和 `s3://DOC-EXAMPLE-BUCKET/folder1/table2`，如以下过程所示。

向 AWS Glue 中的现有爬网程序中添加 S3 数据存储

1. 登录 AWS Management Console，然后打开 AWS Glue 控制台，网址为：https://console.aws.amazon.com/glue/。
2. 在导航窗格中，选择 爬网程序。
3. 选择指向爬网程序的链接，然后选择 Edit（编辑）。
4. 对于步骤 2：选择数据源和分类器，选择 Edit（编辑）。
5. 对于 Data sources（数据来源），选择 Add a data source（添加数据来源）。
6. 在 Add data source（添加数据来源）对话框中，对于 S3 path（S3 路径），选择 Browse（浏览）。
7. 选择要使用的计划，然后选择 Choose（选择）。

   您添加的数据来源显示在 Data sources（数据来源）列表中。
8. 选择下一步。
9. 在 Configure security settings（配置安全设置）页面中，创建或选择爬网程序的 IAM 角色，然后选择 Next（下一步）。
10. 确保 S3 路径以斜杠结尾，然后选择 Add an S3 data source（添加 S3 数据来源）。
11. 在 Set output and scheduling（设置输出和计划）页面中，对于 Output configuration（输出配置），选择目标数据库。
12. 选择下一步。
13. 在 Review and update（审核和更新）页面中，查看您所做的选择。要编辑步骤，请选择 Edit（编辑）。
14. 选择更新。

同步分区架构以避免“HIVE_PARTITION_SCHEMA_MISMATCH”

对 AWS Glue Data Catalog 中每个具有分区列的表，架构都在表级别并且针对表中的每个单独分区存储。分区的架构由 AWS Glue 爬网程序根据它在分区中读取的数据样本进行填充。有关更多信息，请参阅 将多个数据源和爬网程序结合使用。

当 Athena 运行查询时，它会验证表的架构和查询所需的任何分区的架构。验证会将列数据类型按顺序进行比较，确保它们对于重叠的列匹配。这可防止意外的操作，例如在表的中间添加或删除列。如果 Athena 检测到分区的架构与表的架构不同，则 Athena 可能无法处理查询，会因 `HIVE_PARTITION_SCHEMA_MISMATCH` 而失败。

可通过几种方式解决此问题。首先，如果意外添加了数据，您可以删除导致架构差异的数据文件，删除该分区，然后重新爬取数据。其次，您可以删除单个分区，然后在 Athena 中运行 `MSCK REPAIR`，以使用表的架构重新创建分区。只有当您确信应用的架构将继续正确读取数据时，此第二个选项才有效。

更新表元数据

在爬取之后，AWS Glue 爬网程序会自动分配某些表元数据，以帮助它与其他外部技术（如 Apache Hive、Presto 和 Spark）兼容。有时，爬网程序可能会错误地分配元数据属性。在使用 Athena 查询表之前，手动更正 AWS Glue 中的属性。有关更多信息，请参阅《AWS Glue 开发人员指南》中的查看和编辑表详细信息。

当 CSV 文件将每个数据字段都用引号引起来，使 `serializationLib` 属性错误时，AWS Glue 可能会错误分配元数据。有关更多信息，请参阅 用引号引起来的 CSV 数据。

使用 CSV 文件

CSV 文件有时会将每个列所适用的数据值用引号引起来，并且 CSV 文件中可能包含标题值，而这不是要分析的数据的一部分。当您使用 AWS Glue 从这些文件创建架构时，请遵循本部分中的指导。

用引号引起来的 CSV 数据

您可能有一个 CSV 文件，该文件包含在双引号中的数据字段，如下例所示：

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

要在 Athena 中对使用具有引号值的 CSV 文件创建的表运行查询，必须修改 AWS Glue 中的表属性以使用 OpenCSVSerDe。有关 OpenCSV SerDe 的更多信息，请参阅 用于处理 CSV 的 OpenCSVSerDe。

要在 AWS Glue 控制台中编辑表属性

1. 在 AWS Glue 控制台中，选择导航窗格中的 Tables（表）。
2. 选择您要编辑的表的链接，然后依次选择 Action（操作）、Edit table（编辑表）。
3. 在 Edit table（编辑表）页面上，进行以下更改：
   - 对于 Serialization lib（序列化库），输入 `org.apache.hadoop.hive.serde2.OpenCSVSerde`。
   - 对于 Serde parameters（SerDe 参数），为键 `escapeChar`、`quoteChar` 和 `separatorChar` 输入以下值：
     - 对于 `escapeChar`，输入一个反斜杠 (****)。
     - 对于 `quoteChar`，输入一个双引号 (**"**)。
     - 对于 `separatorChar`，输入一个逗号 (**,**)。
4. 选择保存。

有关更多信息，请参阅《AWS Glue 开发人员指南》中的查看和编辑表详细信息。

以编程方式更新 AWS Glue 表属性

您可以使用 AWS Glue UpdateTable API 操作或更新表 CLI 命令修改 `SerDeInfo` 数据块，如以下示例 JSON 所示。

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

具有标题的 CSV 文件

当您在 Athena 中使用 `CREATE TABLE` 语句定义一个表时，可以使用 `skip.header.line.count` 表属性以忽略 CSV 数据中的标题，如下例所示。

```
...
```

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

或者，您可以事先删除 CSV 标题，以便不将标题信息包含在 Athena 查询结果中。实现此操作的一种方法是使用 AWS Glue 任务，它执行提取、转换和加载 (ETL) 工作。您可以使用 PySpark Python 方言的扩展语言在 AWS Glue 中编写脚本。有关更多信息，请参阅《AWS Glue 开发人员指南》中的在 AWS Glue 中编写任务。

以下示例显示 AWS Glue 脚本中的一个函数，它使用 `from_options` 写出动态帧，并将 `writeHeader` 格式选项设置为 false，从而删除标题信息：

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

AWS Glue 分区索引和筛选

在 Athena 查询分区表时，其会检索可用的表分区并筛选出与查询相关的子集。随着新数据和分区的添加，处理分区需要更多时间，查询运行时间可能会增加。如果您的表包含随着时间的推移而增长的大量分区，请考虑使用 AWS Glue 分区索引和筛选。使用分区索引，Athena 可以优化分区处理并提升高度分区表的查询性能。在表的属性中设置分区筛选包括两个步骤：

1. 在 AWS Glue 中创建分区索引。
2. 为表启用分区筛选。

创建分区索引

有关在 AWS Glue 中创建分区索引的步骤，请参阅《AWS Glue 开发人员指南》中的使用分区索引。有关 AWS Glue 中对分区索引的限制，请参阅该页面上的关于分区索引部分。

启用分区筛选

要为表启用分区筛选，必须在 AWS Glue 中设置新的表属性。有关如何在 AWS Glue 中设置表属性的步骤，请参阅设置分区投影页面。当您在 AWS Glue 中编辑表详细信息时，将以下键值对添加到 Table properties（表属性）部分：

- 对于 Key（键），请添加 `partition_filtering.enabled`
- 对于 Value（值），请添加 `true`

您可以随时通过将 `partition_filtering.enabled` 值设置为 `false` 来对此表禁用分区筛选。

完成上述步骤后，您可以返回 Athena 控制台查询数据。

有关使用分区索引和筛选的更多信息，请参阅 AWS 大数据博客中的使用 AWS Glue Data Catalog 分区索引提高 Amazon Athena 查询性能。

使用地理空间数据

AWS Glue 不内在支持已知文本 (WKT)、已知二进制 (WKB) 或其他 PostGIS 数据类型。AWS Glue 分类器解析地理空间数据并使用相应格式支持的数据类型对其进行分类，例如用于 CSV 的 `varchar`。与其他 AWS Glue 表一样，您可能需要更新从地理空间数据创建的表的属性，以允许 Athena 按原样解析这些数据类型。有关更多信息，请参阅 使用 AWS Glue 爬网程序 和 使用 CSV 文件。Athena 可能无法按原样解析 AWS Glue 表中的某些地理空间数据类型。有关在 Athena 中使用地理空间数据的更多信息，请参阅查询地理空间数据。

将 AWS Glue ETL 任务与 Athena 结合使用

AWS Glue 任务执行 ETL 操作。AWS Glue 任务运行一个从源中提取数据、转换数据并将其加载到目标中的脚本。有关更多信息，请参阅《AWS Glue 开发人员指南》中的在 AWS Glue 中编写任务。

将 Athena 用于 AWS Glue ETL 任务来创建表

您在 Athena 中创建的表必须添加有名为 `classification` 的表属性，该属性标识数据的格式。这使 AWS Glue 能够将这些表用于 ETL 任务。分类值可以是 `avro`、`csv`、`json`、`orc`、`parquet` 或 `xml`。下面是 Athena 中的示例 `CREATE TABLE` 语句：

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

如果在创建表时未添加表属性，则可以使用 AWS Glue 控制台添加它。

使用 AWS Glue 控制台添加分类表属性

1. 登录 AWS Management Console，然后打开 AWS Glue 控制台，网址为：https://console.aws.amazon.com/glue/。

2. 在控制台导航窗格中。选择 Tables（表）。
3. 选择您要编辑的表的链接，然后依次选择 Action（操作）、Edit table（编辑表）。
4. 向下滚动到 Table properties（表属性）部分。
5. 选择 添加。
6. 对于键，输入 **`classification`**。
7. 对于 Value（值），输入数据类型（例如 **`json`**)。
8. 选择保存。

   在 Table details（表详细信息）部分中，您输入的数据类型显示在表的 Classification（分类）字段中。

有关更多信息，请参阅《AWS Glue 开发人员指南》中的使用表。

使用 ETL 任务优化查询性能

AWS Glue 任务可帮助您将数据转换为一种可优化 Athena 中的查询性能的格式。数据格式会极大影响 Athena 中的查询性能和查询成本。

我们建议使用 Parquet 和 ORC 数据格式。AWS Glue 支持写入到这两种数据格式，从而使您可以更方便快捷地将数据转换为对 Athena 最佳的格式。有关这些格式的更多信息以及提高性能的其他方法，请参阅 Top 10 performance tuning tips for Amazon Athena。

当转换为 ORC 时将 SMALLINT 和 TINYINT 数据类型转换为 INT

要减少 Athena 无法读取 AWS Glue ETL 任务生成的 `SMALLINT` 和 `TINYINT` 数据类型的可能性，请在使用向导或为 ETL 编写脚本时，将 `SMALLINT` 和 `TINYINT` 转换为 `INT`。

自动执行 AWS Glue ETL 任务

您可以将 AWS Glue ETL 任务配置为基于触发器自动运行。当来自外部 AWS 的数据被以次优格式推送到 Amazon S3 存储桶，以用于在 Athena 中查询时，此功能非常适用。有关更多信息，请参阅《AWS Glue 开发人员指南》中的触发 AWS Glue 任务。

使用 AWS CLI 重新创建 AWS Glue 数据库及其表

无法直接重命名 AWS Glue 数据库，但可以复制其定义，修改定义，然后使用该定义重新创建具有其他名称的数据库。同样，您可以复制旧数据库中的表定义，修改定义，然后使用修改后的定义在新数据库中重新创建表。

> **Note**
>
> 所提供的方法不会复制表分区。

适用于 Windows 的以下过程假设 AWS CLI 已针对 JSON 输出进行配置。要更改 AWS CLI 中的默认输出格式，运行 `aws configure`。

使用 AWS CLI 复制 AWS Glue 数据库

1. 在命令提示符处，运行以下 AWS CLI 命令检索要复制的 AWS Glue 数据库的定义。

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

   有关 `get-database` 命令的更多信息，请参阅 get-database。

2. 将 JSON 输出保存到桌面上以新数据库名称命名的文件中（例如，*new_database_name*.json）。
3. 在文本编辑器中打开 *new_database_name*.json 文件。
4. 在 JSON 文件中，执行以下步骤：

   a. 移除文件末尾的外部 { "Database": 条目和相应的右大括号 }。

   b. 将 Name 条目更改为新的数据库名称。

   c. 删除 CatalogId 字段。

5. 保存该文件。
6. 在命令提示符处，运行以下 AWS CLI 命令使用修改后的数据库定义文件创建具有新名称的数据库。

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

   有关 `create-database` 命令的更多信息，请参阅 create-database。有关从文件中加载 AWS CLI 参数的更多信息，请参阅《AWS Command Line Interface 用户指南》中的从文件中加载 AWS CLI 参数。

7. 要验证是否已在 AWS Glue 中创建新数据库，请运行以下命令：

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

现在，您可以随时获取要复制到新数据库的表的定义，修改定义，然后使用修改后的定义在新数据库中重新创建表。此过程不会更改表名称。

使用 AWS CLI 复制 AWS Glue 表

1. 在命令提示符处，运行以下 AWS CLI 命令。

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

   有关 `get-table` 命令的更多信息，请参阅 get-table。
2. 将 JSON 输出保存到 Windows 桌面上以表名称命名的文件中（例如，*table_name*.json）。
3. 在文本编辑器中打开该文件。
4. 在 JSON 文件中，移除文件末尾的外部 `{"Table":` 条目和相应的右大括号 `}`。
5. 在 JSON 文件中，移除以下条目及其值：

   - `DatabaseName` - 此条目不是必需项，因为 `create-table` CLI 命令使用 `--database-name` 参数。
   - `CreateTime`
   - `UpdateTime`
   - `CreatedBy`
   - `IsRegisteredWithLakeFormation`
   - `CatalogId`
   - `VersionId`
6. 保存表定义文件。
7. 在命令提示符处，运行以下 AWS CLI 命令在新数据库中重新创建表：

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

   有关 `create-table` 命令的更多信息，请参阅 create-table。

   现在，该表出现在 AWS Glue 的新数据库中，可以从 Athena 进行查询。
8. 重复上述步骤，将每个附加表复制到 AWS Glue 的新数据库中。

将 Athena 数据连接器用于外部 Hive 元数据仓

您可以使用外部 Hive 元存储的 Amazon Athena 数据连接器来查询 Amazon S3 中使用 Apache Hive 元存储的数据集。无需将元数据迁移到 AWS Glue Data Catalog。在 Athena 管理控制台中，配置 Lambda 函数以与私有 VPC 中的 Hive 元存储进行通信，然后将它连接到元存储。从 Lambda 到 Hive 元存储的连接由私有 Amazon VPC 通道提供保护，并且不使用公共 Internet。您可以提供您自己的 Lambda 函数代码，也可以使用外部 Hive 元存储的 Athena 数据连接器的默认实施。

主题

功能概览

利用外部 Hive 元存储的 Athena 数据连接器，您可以执行以下任务：

- 使用 Athena 控制台注册自定义目录并使用这些目录运行查询。
- 为不同的外部 Hive 元存储定义 Lambda 函数，并在 Athena 查询中联接它们。
- 在同一 Athena 查询中使用 AWS Glue Data Catalog 和您的外部 Hive 元数据仓。
- 在查询执行上下文中指定目录作为当前默认目录。这样一来，便无需在查询中将目录名称置于数据库名称的前面。您可以使用 *database*.*table*，而不是使用语法 *catalog*.*database*.*table*。
- 使用各种工具运行引用外部 Hive 元存储的查询。您可以使用 Athena 控制台、AWS CLI、AWS 软件开发工具包以及更新的 Athena JDBC 和 ODBC 驱动程序。更新后的驱动程序支持自定义目录。

API 支持

适用于外部 Hive 元数据仓的 Athena 数据连接器包含对目录注册 API 操作和元数据 API 操作的支持。

- 目录注册 – 为外部 Hive 元数据仓和联合数据源注册自定义目录。
- 元数据 – 使用元数据 API 为 AWS Glue 和任何注册到 Athena 目录提供数据库和表信息。
- Athena JAVA 软件开发工具包客户端 – 在更新后的 Athena JAVA 软件开发工具包客户端上的 `StartQueryExecution` 操作中使用目录注册 API、元数据 API 并支持目录。

参考实现

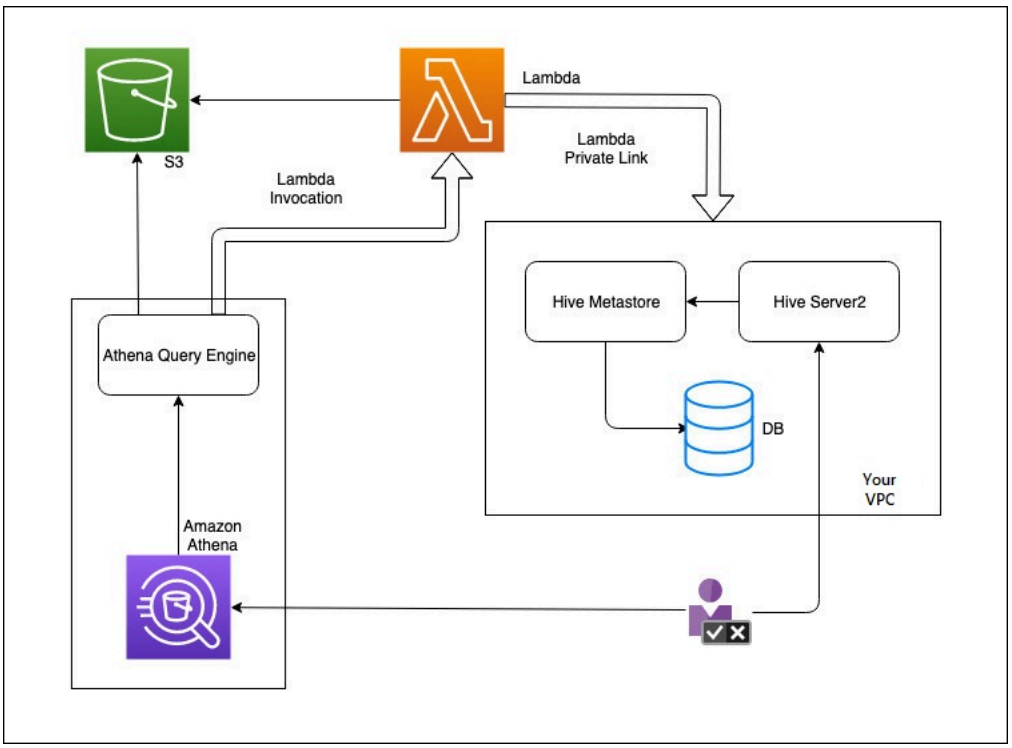
Athena 提供连接到外部 Hive 元存储的 Lambda 函数的参考实施。参考实现在 GitHub 上作为 Athena Hive 元存储的开源项目提供。

参考实施在 AWS Serverless Application Repository (SAR) 中作为以下两种 AWS SAM 应用程序提供。您可以在 SAR 中使用这两种应用程序之一来创建您自己的 Lambda 函数。

- **`AthenaHiveMetastoreFunction`** – Uber Lambda 函数 `.jar` 文件。“uber”JAR（也称为 fat JAR 或具有依赖关系的 JAR）是 `.jar` 文件，该文件在单个文件中同时包含 Java 程序及其依赖关系。
- **`AthenaHiveMetastoreFunctionWithLayer`** – Lambda 层和 thin Lambda 函数 `.jar` 文件。

工作流

下图说明了 Athena 如何与外部 Hive 元数据仓进行交互。



在此工作流程中，您的数据库连接的 Hive 元存储位于您的 VPC 中。您使用 Hive Server2 通过 Hive CLI 管理 Hive 元存储。

使用来自 Athena 的外部 Hive 元数据仓的工作流程包括以下步骤。

1. 创建一个 Lambda 函数以将 Athena 连接到 VPC 中的 Hive 元数据仓。
2. 为 Hive 元存储注册唯一目录名称，并在账户中注册相应的函数名称。
3. 当您运行使用目录名称的 Athena DML 或 DDL 查询时，Athena 查询引擎将调用与目录名称关联的 Lambda 函数名称。
4. 使用 AWS PrivateLink，Lambda 函数与 VPC 中的外部 Hive 元数据仓进行通信，并接收对元数据请求的响应。Athena 使用外部 Hive 元数据仓中的元数据，就像使用默认 AWS Glue Data Catalog 中的元数据一样。

注意事项和限制

在为外部 Hive 元数据仓使用 Athena 数据连接器时，请考虑以下几点：

- 您可以使用 CTAS 在外部 Hive 元存储上创建表。
- 您可以使用 INSERT INTO 将数据插入到外部 Hive 元存储中。
- 对外部 Hive 元数据仓的 DDL 支持仅限于以下语句。

  - ALTER DATABASE SET DBPROPERTIES
  - ALTER TABLE ADD COLUMNS
  - ALTER TABLE ADD PARTITION
  - ALTER TABLE DROP PARTITION
  - ALTER TABLE RENAME PARTITION
  - ALTER TABLE REPLACE COLUMNS
  - ALTER TABLE SET LOCATION
  - ALTER TABLE SET TBLPROPERTIES
  - CREATE DATABASE
  - CREATE TABLE
  - CREATE TABLE AS
  - DESCRIBE TABLE
  - DROP DATABASE
  - DROP TABLE
  - SHOW COLUMNS
  - SHOW CREATE TABLE
  - SHOW PARTITIONS
  - SHOW SCHEMAS
  - SHOW TABLES
  - SHOW TBLPROPERTIES
- 您可以拥有的已注册目录的最大数量为 1000。
- 无法对 Hive 元存储进行 Kerberos 身份验证。
- 若要将 JDBC 驱动程序与外部 Hive 元数据仓或联合查询结合使用，请在 JDBC 连接字符串中包含 `MetadataRetrievalMethod=ProxyAPI`。有关 JDBC 驱动程序的信息，请参阅 通过 JDBC 连接到 Amazon Athena。
- Hive 隐藏列 `$path`、`$bucket`、`$file_size`、`$file_modified_time`、`$partition`、`$row_id` 不能用于精细访问控制筛选。
- Hive 的隐藏系统表，例如 *`example_table`*`$partitions` 或 *`example_table`*`$properties`，不支持精细访问控制。

权限

预构建和自定义数据连接器可能需要访问以下资源才能正常工作。检查您使用的连接器的信息，以确保您已正确配置 VPC。有关在 Athena 中运行查询和创建数据源连接器所需的 IAM 权限的信息，请参阅允许访问适用于外部配置 Hive 元存储的 Athena 数据连接器 和 允许 Lambda 函数访问外部 Hive 元存储。

- Amazon S3 – 除了将查询结果写入 Amazon S3 中的 Athena 查询结果位置之外，数据连接器还会写入到 Amazon S3 中的溢出存储桶。此 Amazon S3 位置的连接和权限是必需的。有关更多信息，请参阅本主题后面的Amazon S3 中的溢出位置。
- Athena – 需要访问权限才能检查查询状态并防止过度扫描。
- AWS Glue – 如果您的连接器将 AWS Glue 用于补充元数据或主元数据，则需要访问权限。
- AWS Key Management Service
- 策略 – 除了 AWS 托管策略：AmazonAthenaFullAccess，Hive 元数据仓、Athena Query Federation 和 UDF 需要策略。有关更多信息，请参阅 Athena 中的 Identity and Access Management。

Amazon S3 中的溢出位置

由于对 Lambda 函数响应大小的限制，则大于阈值的响应会溢出到您在创建 Lambda 函数时指定的 Amazon S3 位置。Athena 直接从 Amazon S3 读取这些响应。

> **Note**
>
> Athena 不会删除 Amazon S3 上的响应文件。我们建议您设置保留策略以自动删除响应文件。

将 Athena 连接到 Apache Hive 元存储

要将 Athena 连接到 Apache Hive 元数据仓，您必须创建和配置 Lambda 函数。对于基本实现，您可以从 Athena 管理控制台执行所有必需步骤。

> **Note**
>
> 以下过程要求您具有为 Lambda 函数创建自定义 IAM 角色的权限。如果您没有权限创建自定义角色，则可以使用 Athena 参考实现以单独创建 Lambda 函数，然后使用 AWS Lambda 控制

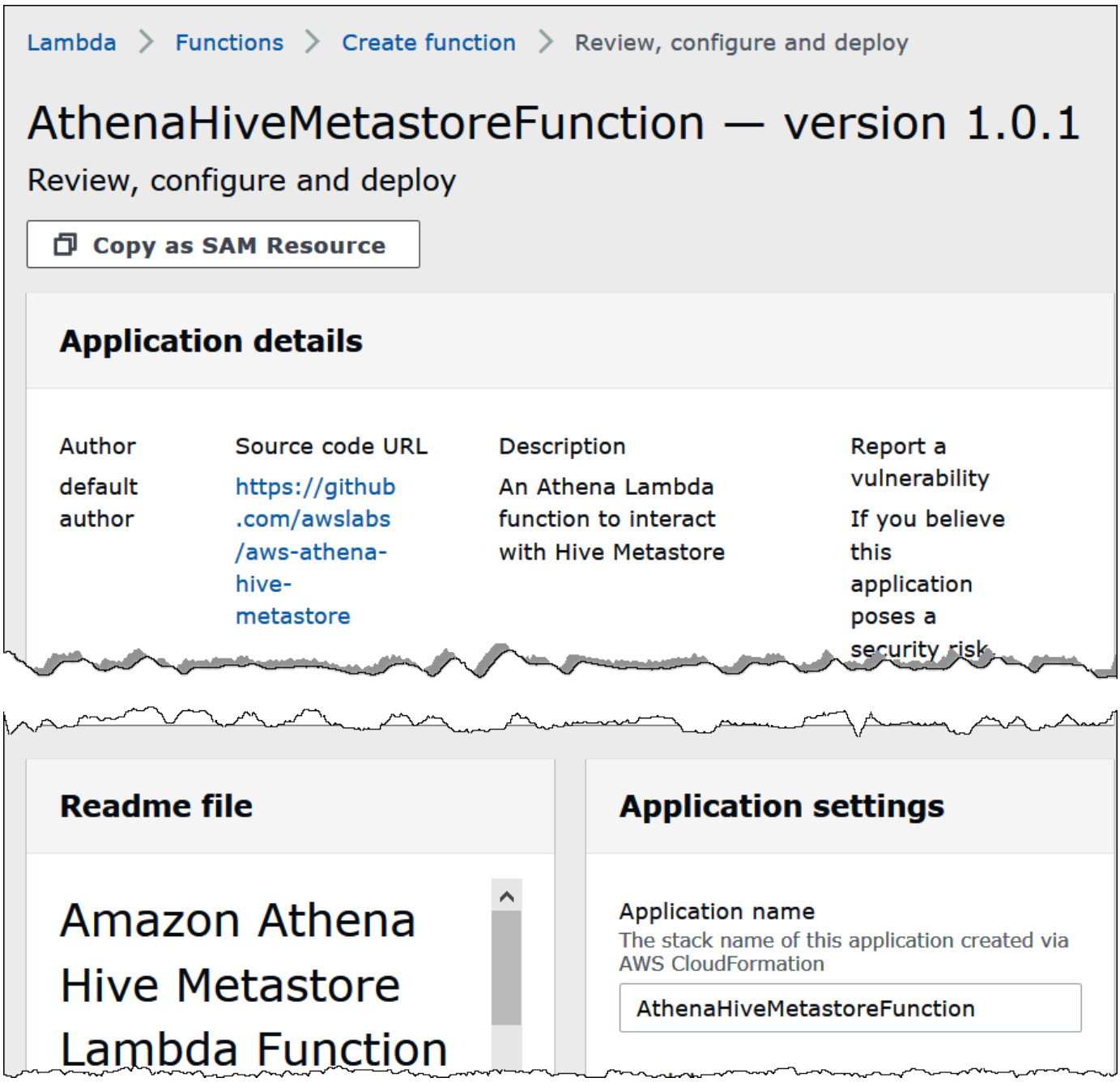
台，为函数选择现有 IAM 角色。有关更多信息，请参阅 使用现有 IAM 执行角色将 Athena 连接到 Hive 元存储。

要将 Athena 连接到 Hive 元数据仓

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。

3. 选择 Data Source（数据源）。
4. 在控制台右上角，选择 Connect data source（连接数据源）。
5. 在 Choose a data sources（选择数据源）页面中，对于 Data source（数据源），请选择 S3 - Apache Hive metastore（S3 - Apache Hive 元数据仓）。
6. 选择下一步。
7. 在 Data source details（数据源详细信息）部分，对于 Data source name（数据源名称），输入从 Athena 查询数据源时想要在 SQL 语句中使用的名称。名称最多可以包括 127 个字符，并且在您的账户中必须是唯一的。它在创建后即无法更改。有效字符包括 a-z、A-Z、0-9、_（下划线）、@（at 符号）和 -（连字符）。名称 `awsdatacatalog`、`hive`、`jmx` 和 `system` 是 Athena 预留的名称，无法用于数据源名称。
8. 对于 Lambda function（Lambda 函数），选择 Create Lambda function（创建 Lambda 函数），然后选择 Create a new Lambda function in AWS Lambda（在 中新建 Lambda 函数）

   AthenaHiveMetastoreFunction 页面将会在 AWS Lambda 控制台中打开。该页面包括连接器的详细信息。

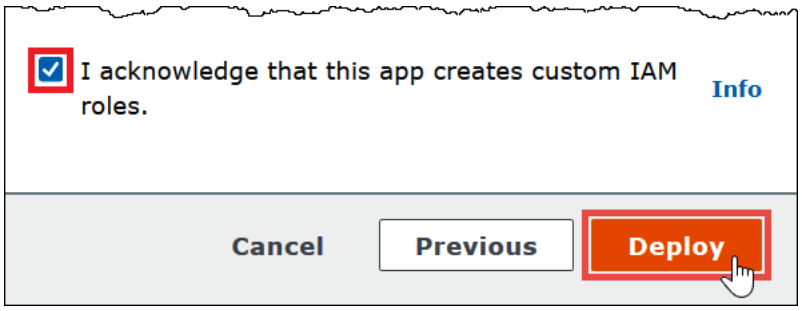


9. 在 Application settings（应用程序设置）中，输入 Lambda 函数的参数。

   - LambdaFuncName – 提供函数的名称。例如，myHiveMetastore。
   - SpillLocation – 在此账户中指定 Amazon S3 位置，以便在 Lambda 函数响应大小超过 4 MB 时保存溢出元数据。
   - HMSUris – 输入您的 Hive 元数据仓主机的名称，该主机在端口 9083 使用 Thrift 协议。使用语法 `thrift://<host_name>:9083`。

- LambdaMemory – 指定从 128 MB 到 3008 MB 的值。Lambda 函数分配与您配置的内存量成比例的 CPU 周期。默认值为 1024。
- LambdaTimeout – 指定允许的最大 Lambda 调用运行时（以秒为单位），从 1 到 900（900 秒为 15 分钟）。默认值为 300 秒（5 分钟）。
- VpcSecurityGroupIds – 为 Hive 元数据仓输入一个逗号分隔的 VPC 安全组 ID 列表。
- VpcSubnetIds – 为 Hive 元数据仓输入一个逗号分隔的 VPC 子网 ID 列表。

10. 选择 I acknowledge that this app creates custom IAM roles（我确认此应用程序创建自定义 IAM 角色），然后选择 Deploy（部署）。



部署完成后，函数将显示在 Lambda 应用程序列表中。现在 Hive 元数据仓功能已部署到您的账户，您可以配置 Athena 配置以使用此功能。

11. 返回至 Athena 控制台中的 Enter data source details（输入数据源详细信息）页面。
12. 在 Lambda function（Lambda 函数）部分，选择 Lambda 函数搜索框旁的刷新图标。刷新可用函数列表会导致新创建的函数出现在列表中。
13. 在 Lambda 控制台上选择刚创建的函数名称。将显示 Lambda 函数的 ARN。
14. （可选）对于 Tags（标签），添加要与此数据源关联的键值对。有关标签的更多信息，请参阅 为 Athena 资源添加标签。
15. 选择下一步。
16. 在 Review and create（审核和创建）页面中，查看数据源的详细信息，然后选择 Create data source（创建数据源）。
17. 数据源此页面的 Data source details（数据源详细信息）部分显示了有关新连接器的信息。

    您现在可以使用指定的 Data source name（数据源名称）以在 Athena 的 SQL 查询中引用 Hive 元数据仓。在 SQL 查询中，使用以下示例语法，并将 `hms-catalog-1` 替换为您之前指定的目录名称。

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

18. 有关查看、编辑或删除您创建的数据源的信息，请参阅 管理数据来源。

使用 AWS Serverless Application Repository 部署 Hive 数据来源连接器

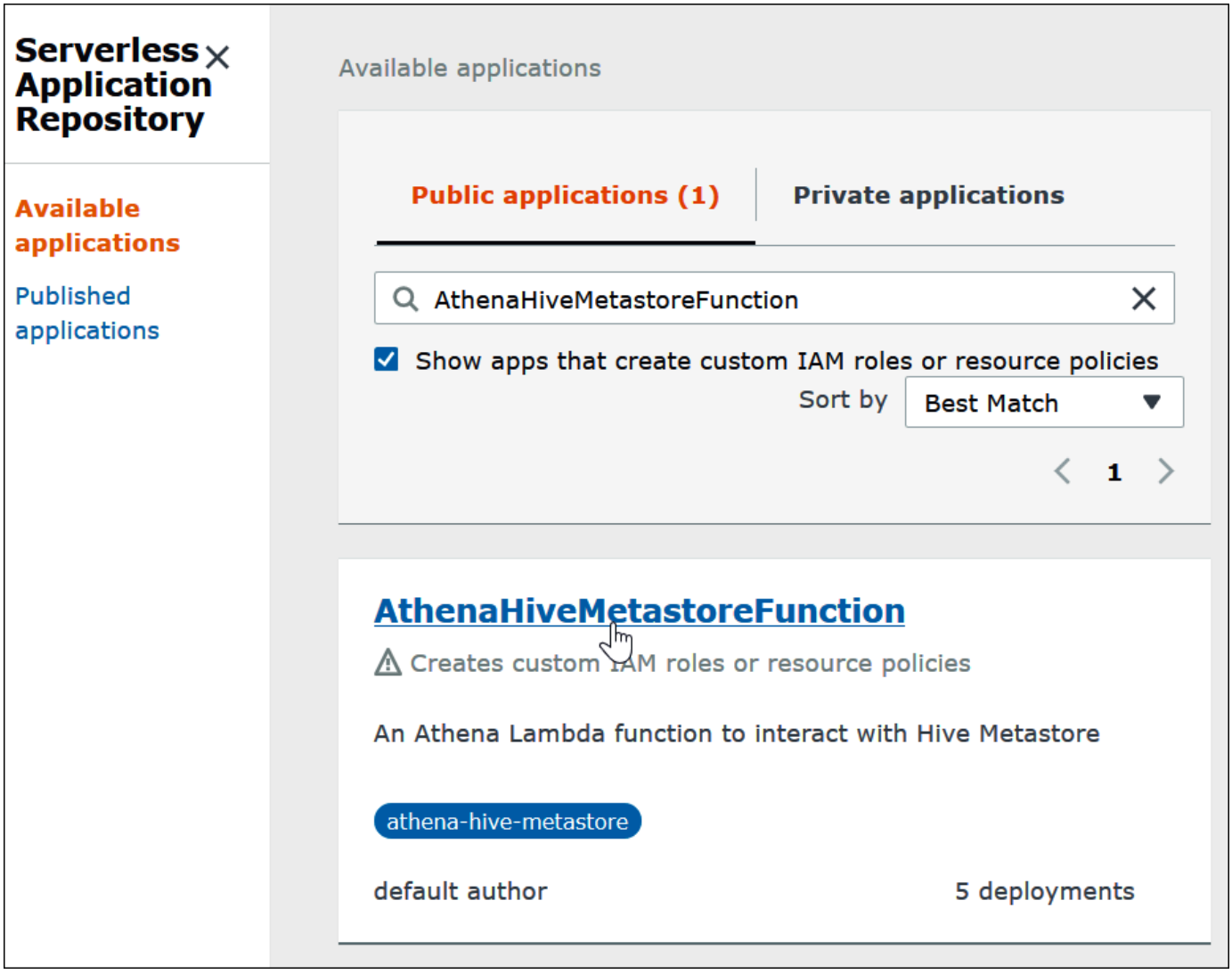
要部署适用于 Hive 的 Athena 数据来源连接器，您可以使用 AWS Serverless Application Repository，而不是从 Athena 控制台开始。使用 AWS Serverless Application Repository 找到要使用的连接器，提供连接器所需的参数，然后将连接器部署到您的账户。部署连接器后，您可以使用 Athena 控制台向 Athena 提供数据来源。

使用 AWS Serverless Application Repository 将适用于 Hive 的数据来源连接器部署到您的账户

1. 登录 AWS Management Console并打开 Serverless App Repository (无服务器应用程序存储库)。
2. 在导航窗格中，选择 Available applications (可用应用程序)。
3. 选择 Show apps that create custom IAM roles or resource policies（显示创建自定义 IAM 角色或资源策略的应用程序）选项。
4. 在搜索框中，输入 **`Hive`**。显示的连接器包括以下两个：

   - AthenaHiveMetastoreFunction – Uber Lambda 函数 `.jar` 文件。
   - AthenaHiveMetastoreFunctionWithLayer – Lambda 层和 thin Lambda 函数 `.jar` 文件。

   这两个应用程序具有相同的功能，仅在实施方面存在差异。您可以使用二者之一创建将 Athena 连接到 Hive 元数据仓的 Lambda 函数。
5. 选择要使用的连接器名称。本教程使用 AthenaHiveMetastoreFunction。



6. 在 Application settings（应用程序设置）中，输入 Lambda 函数的参数。

   - LambdaFuncName – 提供函数的名称。例如，myHiveMetastore。
   - SpillLocation – 在此账户中指定 Amazon S3 位置，以便在 Lambda 函数响应大小超过 4 MB 时保存溢出元数据。
   - HMSUris – 输入您的 Hive 元数据仓主机的名称，该主机在端口 9083 使用 Thrift 协议。使用语法 `thrift://<host_name>:9083`。
   - LambdaMemory – 指定从 128 MB 到 3008 MB 的值。Lambda 函数分配与您配置的内存量成比例的 CPU 周期。默认值为 1024。
   - LambdaTimeout – 指定允许的最大 Lambda 调用运行时（以秒为单位），从 1 到 900（900 秒为 15 分钟）。默认值为 300 秒（5 分钟）。
   - VpcSecurityGroupIds – 为 Hive 元数据仓输入一个逗号分隔的 VPC 安全组 ID 列表。

- VpcSubnetIds – 为 Hive 元数据仓输入一个逗号分隔的 VPC 子网 ID 列表。

7. 在 Application details（应用程序详细信息）页面的右下角，选择 I acknowledge that this app creates custom IAM roles（我确认此应用程序创建自定义 IAM 角色），然后选择 Deploy（部署）。

此时，您可以将 Athena 配置为使用 Lambda 函数连接到 Hive 元数据仓。要查看步骤，请参阅 配置 Athena 使用部署的 Hive 元存储连接器。

使用现有 IAM 执行角色将 Athena 连接到 Hive 元存储

要借助使用现有 IAM 角色的 Lambda 函数将外部 Hive 元数据仓连接到 Athena，您可以将 Athena 的 Athena 连接器引用实现用于外部 Hive 元数据仓。

这三个主要步骤如下所示：

1. 克隆和构建：克隆 Athena 引用实现并构建包含 Lambda 函数代码的 JAR 文件。
2. AWS Lambda 控制台 – 在 AWS Lambda 控制台，创建 Lambda 函数，为其分配现有 IAM 执行角色，然后上载您生成的函数代码。
3. Amazon Athena 控制台 – 在 Amazon Athena 控制台中，创建一个数据来源名称，您可以使用该名称在 Athena 查询中引用外部 Hive 元存储。

如果您已拥有创建自定义 IAM 角色的权限，则可以使用更简单的工作流程，该工作流程使用 Athena 控制台和 AWS Serverless Application Repository 创建和配置 Lambda 函数。有关更多信息，请参阅 将 Athena 连接到 Apache Hive 元存储。

先决条件

- Git 必须安装在您的系统上。
- 您必须先行安装 Apache Maven。
- 您有一个 IAM 执行角色，可以将其分配给 Lambda 函数。有关更多信息，请参阅 允许 Lambda 函数访问外部 Hive 元存储。

克隆并构建 Lambda 函数

Athena 参考实现的函数代码是一个位于 GitHub 上的 Maven 项目，具体地址为 awslabs/aws-athena-hive-metastore。有关项目的详细信息，请参阅 GitHub 上相应的自述文件或本文档中的 参考实现 主题。

克隆和构建 Lambda 函数代码

1. 输入以下命令以克隆 Athena 引用实现：

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

2. 运行以下命令为 Lambda 函数构建 `.jar` 文件：

```
mvn clean install
```

项目构建成功后，以下 `.jar` 文件将在项目的目标文件夹中创建：

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

在下一部分中，您将使用 AWS Lambda 控制台将此文件上载到您的亚马逊云科技账户。

在 AWS Lambda 控制台中创建和配置 Lambda 函数

在本节中，您将使用 AWS Lambda 控制台创建一个使用现有 IAM 执行角色的函数。为函数配置 VPC 后，您可以上载函数代码并配置该函数的环境变量。

创建 Lambda 函数

在此步骤中，您将在 AWS Lambda 控制台中创建一个使用现有 IAM 角色的函数。

要创建使用现有 IAM 角色的 Lambda 函数

1. 登录到 AWS Management Console，然后通过以下网址打开 AWS Lambda 控制台：https://console.aws.amazon.com/lambda/。
2. 在导航窗格中，选择函数。
3. 选择 Create function (创建函数)。
4. 选择从头开始创作。
5. 在 Function name（函数名称）中，输入 Lambda 函数的名称（例如 **`EHMSBasedLambda`**）。
6. 对于 Runtime（运行时），请选择 Java 8。
7. 在 Permissions（权限）下，展开 Change default execution role（更改默认执行角色）。
8. 对于 Execution role (执行角色)，选择 Use an existing role (使用现有角色)。
9. 对于 Existing role（现有角色），请选择 Lambda 函数将用于 Athena 的 IAM 执行角色（此示例使用名为 `AthenaLambdaExecutionRole` 的角色)。

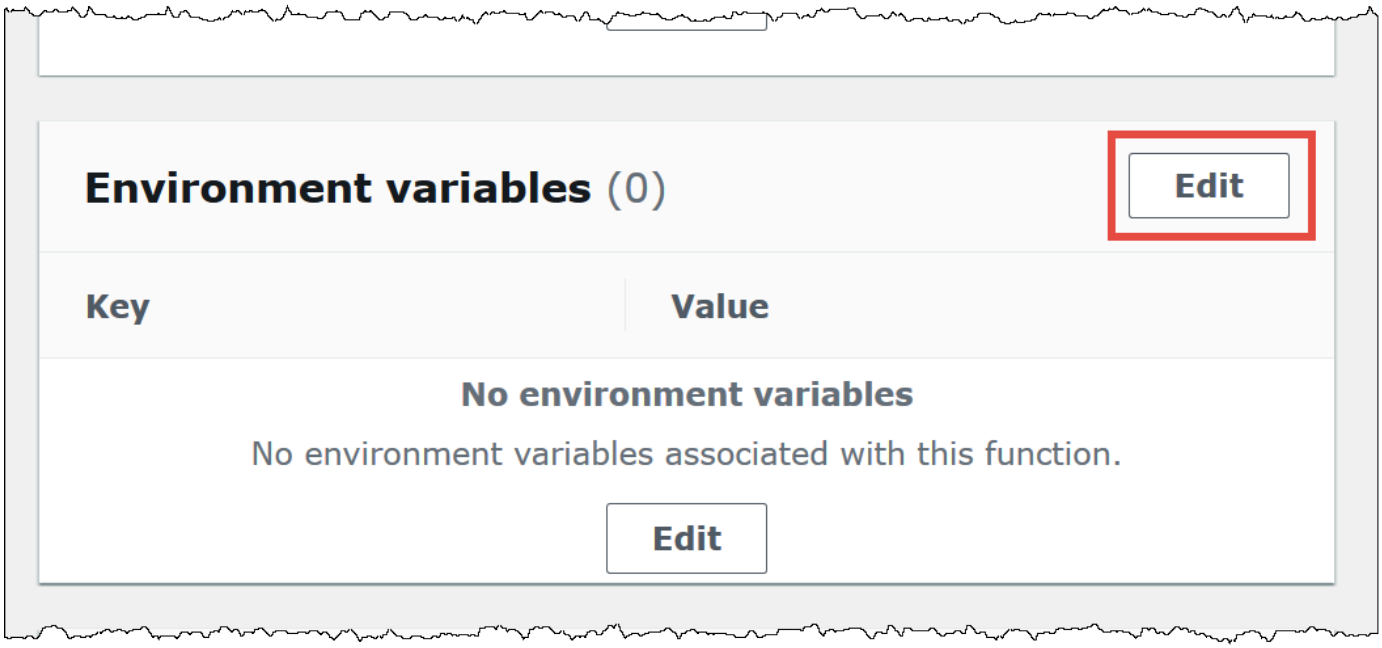
10. 展开 Advanced settings（高级设置）。
11. 选择 Enable Network（启用网络）。
12. 对于 VPC，请选择您的函数将有权访问的 VPC。
13. 对于 Subnets（子网），选择 Lambda 要使用的 VPC 子网。
14. 对于 Security groups（安全组），选择 Lambda 要使用的 VPC 安全组。
15. 选择创建函数。AWS Lambda 控制台，然后打开函数的配置页面并开始创建函数。

上载代码并配置 Lambda 函数

当控制台通知您已成功创建函数时，您就可以上载函数代码并配置其环境变量。

上载 Lambda 函数代码并配置其环境变量

1. 在 Lambda 控制台中，确保您位于指定函数页面的 Code（代码）选项卡上。
2. 对于 Code source（代码源），选择 Upload from（上传格式），然后选择 .zip or .jar file（.zip 或 .jar 文件）。
3. 上载您之前生成的 `hms-lambda-func-1.0-SNAPSHOT-withdep.jar` 文件。
4. 在 Lambda 函数页面上，选择配置选项卡。
5. 从左侧窗格中选择 Environment variables（环境变量）。
6. 在 Environment variables (环境变量) 部分中，选择 Edit (编辑)。

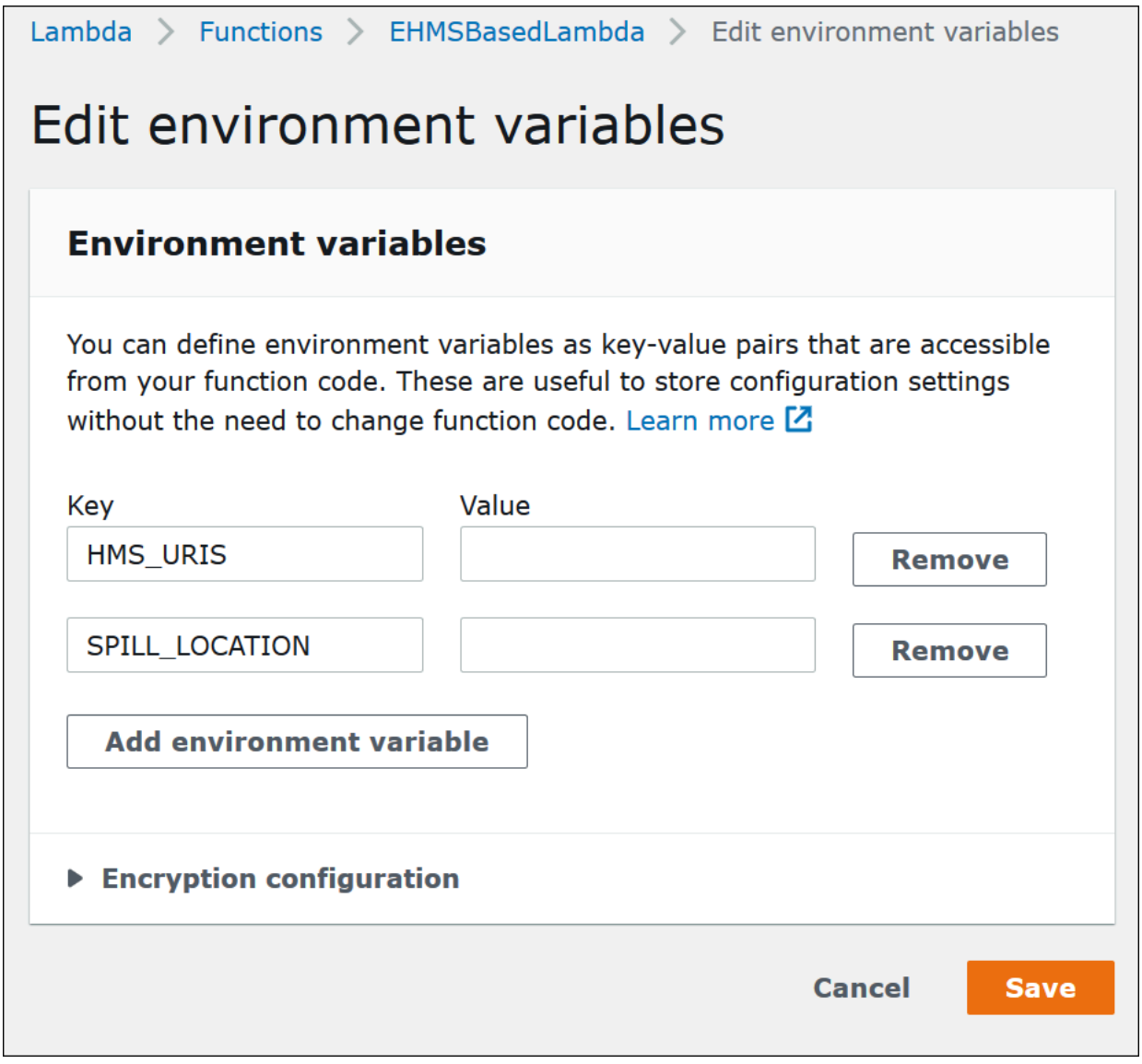


7. 在 Edit environment variables（编辑环境变量）页面上，使用 Add environment variable（添加环境变量）选项，添加以下环境变量键和值：

   - HMS_URIS – 使用以下语法，输入 Hive 元数据仓主机的 URI，该主机在端口 9083 使用 Thrift 协议。

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

   - SPILL_LOCATION – 在亚马逊云科技账户中指定 Amazon S3 位置，以便在 Lambda 函数响应大小超过 4 MB 时保存溢出元数据。



8. 选择保存。

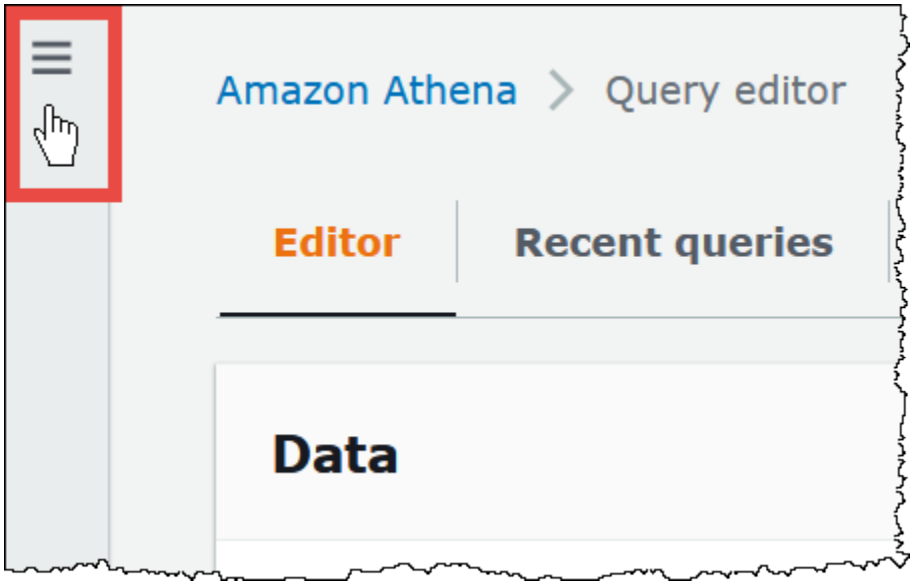
此时，您已准备好将 Athena 配置为使用 Lambda 函数连接到 Hive 元数据仓。要查看步骤，请参阅 配置 Athena 使用部署的 Hive 元存储连接器。

配置 Athena 使用部署的 Hive 元存储连接器

将 `AthenaHiveMetastoreFunction` 等 Lambda 数据源连接器部署到账户后，您可以配置 Athena 以进行使用。为此，请创建数据源名称，该名称会引用外部 Hive 元数据仓以在 Athena 查询中使用。

使用现有 Lambda 函数将 Athena 连接到 Hive 元数据仓

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。



3. 选择 Data Source（数据源）。
4. 在 Data sources（数据源）页面上，选择 Create data source（创建数据源）。
5. 在 Choose a data sources（选择数据源）页面中，对于 Data source（数据源），请选择 S3 - Apache Hive metastore（S3 - Apache Hive 元数据仓）。
6. 选择下一步。
7. 在 Data source details（数据源详细信息）部分中，对于 Data source name（数据源名称），请输入从 Athena 查询数据源时要在 SQL 语句中使用的名称（例如 `MyHiveMetastore`）。名称最多可以包括 127 个字符，并且在您的账户中必须是唯一的。它在创建后即无法更改。有效字符包括 a-z、A-Z、0-9、_（下划线）、@（at 符号）和 -（连字符）。名称 `awsdatacatalog`、`hive`、`jmx` 和 `system` 是 Athena 预留的名称，无法用于数据源名称。
8. 在 Connection details（连接详细信息）部分中，请使用 Select or enter a Lambda function（选择或输入 Lambda 函数）框以选择刚创建的函数名称。将显示 Lambda 函数的 ARN。

9. （可选）对于 Tags（标签），添加要与此数据源关联的键值对。有关标签的更多信息，请参阅 为 Athena 资源添加标签。
10. 选择下一步。
11. 在 Review and create（审核和创建）页面中，查看数据源的详细信息，然后选择 Create data source（创建数据源）。
12. 数据源此页面的 Data source details（数据源详细信息）部分显示了有关新连接器的信息。

    您现在可以使用指定的 Data source name（数据源名称）以在 Athena 的 SQL 查询中引用 Hive 元数据仓。

    在 SQL 查询中，使用以下示例语法，并将 `ehms-catalog` 替换为您之前指定的数据源名称。

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

13. 要查看、编辑或删除您创建的数据源，请参阅管理数据来源。

在外部 Hive 元存储查询中使用默认数据源

在外部 Hive 元存储上运行 DML 和 DDL 查询时，如果在查询编辑器中选择了目录名称，则可以通过省略该名称来简化查询语法。此功能将受到一些限制。

DML 语句

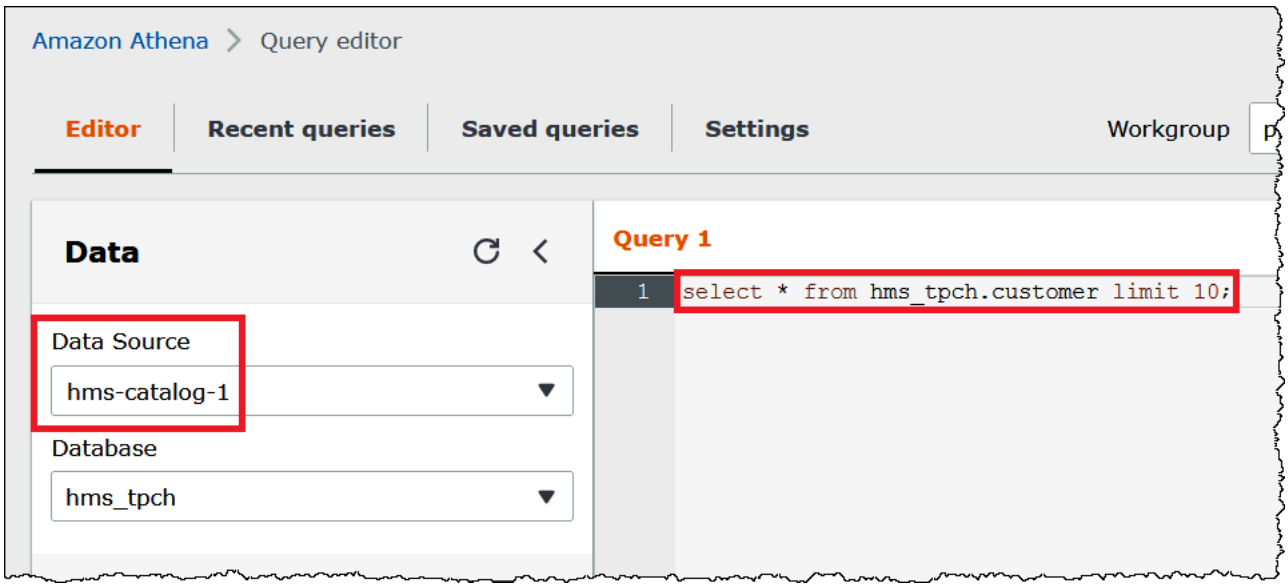
使用已注册目录运行查询

1. 您可以使用语法 `[[data_source_name].database_name].table_name` 将数据源名称置于数据库之前，如以下示例所示。

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

2. 如果已在查询编辑器中选定要使用的数据源，您可以在查询中省略数据源名称，如以下示例所示。

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



3. 在查询中使用多个数据源时，只能省略默认数据源名称，并且必须为任何非默认数据源指定全名。

   例如，假设在查询编辑器中将 AwsDataCatalog 选作默认数据源。以下查询摘录中的 FROM 语句完全限定了前两个数据源名称，但省略了第三个数据源的名称，因为它位于 AWS Glue 数据目录中。

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

DDL 语句

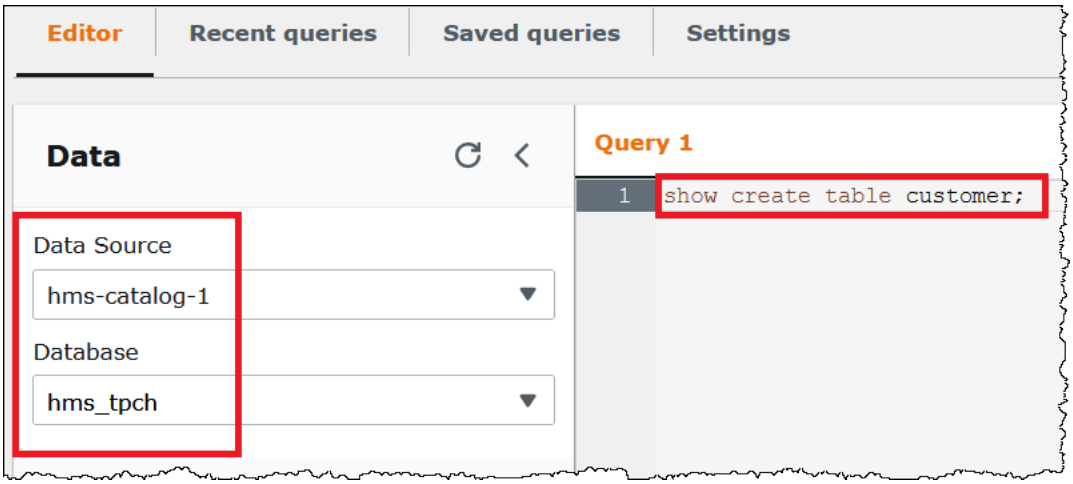
以下 Athena DDL 语句支持目录名称前缀。其他 DDL 语句中的目录名称前缀会导致语法错误。

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

与 DML 语句一样，在查询编辑器中选择了数据源和数据库时，可以省略查询中的数据源和数据库前缀。

在以下示例中，在查询编辑器中选择了 hms-catalog-1 数据源和 hms_tpch 数据库。即使查询本身中忽略 hms-catalog-1 前缀和 hms_tpch 数据库名称，show create table customer 语句也会成功。



在 JDBC 连接字符串中指定默认数据源

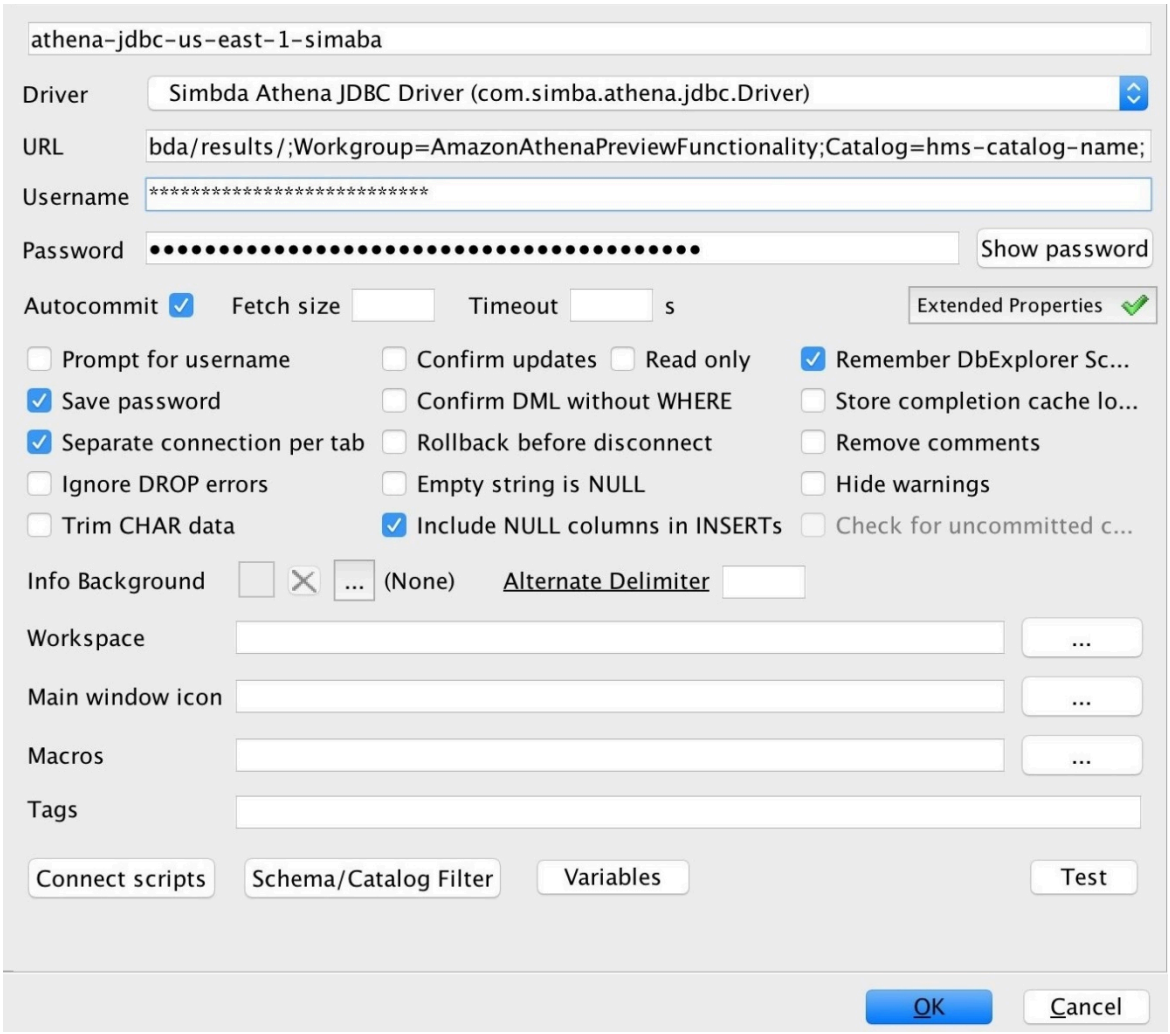
当您使用 Athena JDBC 驱动程序将 Athena 连接到外部 Hive 元存储时，可以使用 Catalog 参数在 SQL 编辑器（如 SQL Workbench）中的连接字符串中指定默认数据源名称。

> **Note**
>
> 要下载最新版本的 Athena JDBC 驱动程序，请参阅将 Athena 与 JDBC 驱动程序一起使用。

以下连接字符串指定默认数据源 *hms-catalog-name*。

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

下图显示了 SQL Workbench 中配置的示例 JDBC 连接 URL。



使用 Hive 视图

您可以使用 Athena 查询外部 Apache Hive 元存储中的现有视图。Athena 会在运行时实时转换视图，无需更改原始视图或存储翻译。

例如，假设您的 Hive 视图使用了 Athena 中不支持的语法，比如 `LATERAL VIEW explode()`：

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

Athena 将 Hive 视图查询字符串转换为 Athena 可以运行的语句，如下所示：

```
SELECT team, score
FROM matches
```

```
CROSS JOIN UNNEST(scores) AS m (score)
```

有关将外部 Hive 元存储连接到 Athena 的信息，请参阅 将 Athena 数据连接器用于外部 Hive 元数据仓。

注意事项和限制

在从 Athena 查询 Hive 视图时，请考虑以下几点：

- Athena 不支持创建 Hive 视图。您可以在外部 Hive 元存储中创建 Hive 视图，之后可以从 Athena 查询该视图。
- Athena Hive 视图不支持自定义 UDF。
- 由于 Athena 控制台中的已知问题，Hive 视图显示在表列表下而不是视图列表下。
- 尽管转换过程会自动执行，但 Hive 视图不支持某些 Hive 函数或需要特殊处理。有关更多信息，请参阅以下 章节。

Hive 函数支持限制

此部分重点介绍了 Athena Hive 视图不支持的或需要特殊处理的 Hive 函数。目前，Athena 主要支持 Hive 2.2.0 的函数，因此不支持只在更高版本（例如 Hive 4.0.0）中可用的函数。有关 Hive 函数的完整列表，请参阅 Hive 语言手册 UDF。

聚合函数

需要特殊处理的聚合函数

以下 Hive 视图的聚合函数需要特殊处理。

- Avg – 请使用 `avg(CAST(i AS DOUBLE))`，而非 `avg(INT i)`。

不支持的聚合函数

Athena 的 Hive 视图不支持以下 Hive 聚合函数。

```
covar_pop
histogram_numeric
ntile
percentile
percentile_approx
```

Athena 的 Hive 视图不支持 `regr_count`、`regr_r2` 和 `regr_sxx` 等回归函数。

不支持的日期函数

Athena 的 Hive 视图不支持以下 Hive 日期函数。

```
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)
```

不支持的掩码函数

Athena 的 Hive 视图不支持 `mask()` 和 `mask_first_n()` 等 Hive 掩码函数。

其他函数

需要特殊处理的其他函数

Hive 视图的以下其他函数需要特殊处理。

- md5 – Athena 支持 `md5(binary)`，但不支持 `md5(varchar)`。
- Explode – 在以下语法中使用时，Athena 支持 `explode`：

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

- Posplode：在以下语法中使用时，Athena 支持 `posexplode`：

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

  在 `(pos, val)` 输出中，Athena 将 `pos` 视为 `BIGINT`。因此，您可能需要将 `pos` 转换为 `BIGINT`，防止提供过时视图。以下示例对此方法进行了说明。

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

不支持的其他函数

Athena 的 Hive 视图不支持以下 Hive 函数。

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

运算符

需要特殊处理的运算符

Hive 视图的以下运算符需要特殊处理。

- Mod 运算符 (%)：因为 DOUBLE 类型隐式转换为 DECIMAL(x,y)，以下语法可能导致出现 View is stale（视图已过时）错误消息：

  ```
  a_double % 1.0 AS column
  ```

  使用 CAST 可解决此问题，如以下示例所示。

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

- 除法运算符 (/)：在 Hive 中，int 除以 int 会生成 double。在 Athena 中，同样的操作会产生截断 int。

不支持的运算符

Athena 的 Hive 视图不支持以下运算符。

~A – 按位 NOT

A ^ b – 按位 XOR

A & b – 按位 AND

A | b – 按位 OR

A <=> b – 返回与非空操作数的等于 (=) 运算符相同的结果。如果两者都为 NULL，则返回 TRUE；如果其中一个为 NULL，则返回 FALSE。

字符串函数

需要特殊处理的字符串函数

Hive 视图的以下 Hive 字符串函数需要特殊处理。

- chr(bigint|double a) – Hive 允许负参数，但 Athena 不允许。
- instr(string str, string substr) – 因为 `instr` 函数的 Athena 映射返回 `BIGINT` 而非 `INT`，所以使用以下语法：

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

  如果不执行此步骤，视图将被视作过时。
- length(string a) – 因为 `length` 函数的 Athena 映射返回 `BIGINT` 而非 `INT`，所以使用以下函数，避免视图被视作过时：

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

不支持的字符串函数

Athena 的 Hive 视图不支持以下 Hive 字符串函数。

```
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)
```

不支持的 xPath 函数

Athena 的 Hive 视图不支持 `xpath`、`xpath_short` 和 `xpath_int` 等 Hive XPath 函数。

故障排除

在 Athena 中使用 Hive 视图时，可能遇到以下问题：

- 视图 ***<view name>*** 已过时 – 此消息通常表示 Hive 视图与 Athena 视图存在类型不匹配问题。如果 Hive LanguageManual UDF 以及 Presto 函数和运算符文档中的相同函数具有不同签名，请尝试强制转换不匹配的数据类型。
- 函数未注册：Athena 目前不支持此函数。有关更多信息，请参阅本文档前面所述的信息。

将 AWS CLI 与 Hive 元存储结合使用

您可以使用 `aws athena` CLI 命令管理与 Athena 结合使用的 Hive 元数据仓数据目录。在定义一个或多个要与 Athena 结合使用的目录后，可以在 `aws athena` DDL 和 DML 命令中引用这些目录。

使用 AWS CLI 管理 Hive 元存储目录

注册目录：Create-data-catalog

要注册数据目录，请使用 `create-data-catalog` 命令。使用 `name` 参数指定要用于目录的名称。将 Lambda 函数的 ARN 传递到 `parameters` 参数的 `metadata-function` 选项。要为新目录创建标签，请将 `tags` 参数与一个或多个用空格分隔的 `Key=`*`key`*`,Value=`*`value`* 参数对结合使用。

以下示例注册名为 `hms-catalog-1` 的 Hive 元存储目录。该命令设置了格式以便于阅读。

```
$ 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
```

显示目录详细信息：Get-data-catalog

要显示目录的详细信息，请将目录的名称传递到 `get-data-catalog` 命令，如以下示例所示。

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

以下示例结果采用 JSON 格式。

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

列出已注册的目录：List-data-catalogs

要列出已注册的目录，请使用 `list-data-catalogs` 命令并选择性地指定一个区域，如以下示例所示。列出的目录始终包含 AWS Glue。

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

以下示例结果采用 JSON 格式。

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

更新目录：Update-data-catalog

要更新数据目录，请使用 update-data-catalog 命令，如以下示例所示。该命令设置了格式以便于阅读。

```
$ 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
```

删除目录：Delete-data-catalog

要删除数据目录，请使用 delete-data-catalog 命令，如以下示例所示。

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

显示数据库详细信息：Get-database

要显示数据库的详细信息，请将目录名称和数据库名称传递到 get-database 命令，如以下示例所示。

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

以下示例结果采用 JSON 格式。

```
{
```

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

在目录中列出数据库：List-databases

要在目录中列出数据库，请使用 `list-databases` 命令并选择性地指定一个区域，如以下示例所示。

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

以下示例结果采用 JSON 格式。

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

显示表详细信息：Get-table-metadata

要显示表的元数据（包括列名和数据类型），请将目录名称、数据库名称和表名称传递到 get-table-metadata 命令，如以下示例所示。

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

以下示例结果采用 JSON 格式。

```
{
    "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"
```

```
            }
        }
}
```

显示数据库中所有表的元数据：List-table-metadata

要显示数据库中所有表的元数据，请将目录名称和数据库名称传递到 `list-table-metadata` 命令。`list-table-metadata` 命令与 `get-table-metadata` 命令类似，只是未指定表名。要限制结果的数量，您可以使用 `--max-results` 选项，如以下示例所示。

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

以下示例结果采用 JSON 格式。

```
{
    "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
```

```
}
```

运行 DDL 和 DML 语句

在使用 AWS CLI 运行 DDL 和 DML 语句时，您可以通过两种方式之一传递 Hive 元存储目录的名称：

- 直接传入支持它的语句。
- 传递到 `--query-execution-context` `Catalog` 参数。

DDL 语句

以下示例将目录名称作为 `show create table` DDL 语句的一部分直接传入。该命令设置了格式以便于阅读。

```
$ 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"
```

以下示例 DDL `show create table` 语句使用 `--query-execution-context` 的 `Catalog` 参数传递 Hive 元存储目录名称 `hms-catalog-1`。该命令设置了格式以便于阅读。

```
$ 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"
```

DML 语句

以下示例 DML `select` 语句将目录名称直接传入查询。该命令设置了格式以便于阅读。

```
$ 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"
```

以下示例 DML `select` 语句使用 `--query-execution-context` 的 `Catalog` 参数传入 Hive 元存储目录名称 `hms-catalog-1`。该命令设置了格式以便于阅读。

```
$ 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"
```

参考实现

Athena 为 GitHub.com（网址为 https://github.com/awslabs/aws-athena-hive-metastore）上的外部 Hive 元数据仓提供其连接器的参考实施。

参考实施是一个具有以下模块的 Apache Maven 项目：

- **hms-service-api** – 包含 Lambda 函数和 Athena 服务客户端之间的 API 操作。这些 API 操作是在 HiveMetaStoreService 接口中定义的。由于这是一个服务合同，因此，您不应更改此模块中的任何内容。
- **hms-lambda-handler** – 一组默认 Lambda 处理程序，用于处理所有 Hive 元数据仓 API 调用。类 MetadataHandler 是用于所有 API 调用的调度程序。您无需更改此程序包。
- **hms-lambda-func** – 一个具有以下组件的示例 Lambda 函数。
  - **HiveMetaStoreLambdaFunc** – 一个扩展 MetadataHandler 的示例 Lambda 函数。
  - **ThriftHiveMetaStoreClient** – 一个与 Hive 元存储进行通信的 Thrift 客户端。此客户端是针对 Hive 2.3.0 编写的。如果您使用其他 Hive 版本，则可能需要更新此类以确保响应对象可兼容。
  - **ThriftHiveMetaStoreClientFactory** – 控制 Lambda 函数的行为。例如，您可以通过覆盖 getHandlerProvider() 方法来提供自己的一组处理程序提供程序。
  - hms.properties – 配置 Lambda 函数。在大多数情况下，只需更新以下两个属性。
    - hive.metastore.uris – 格式为 thrift://*<host_name>*:9083 的 Hive 元存储 URI。
    - hive.metastore.response.spill.location：当响应对象的大小超过给定阈值（例如 4 MB）时，用于存储响应对象的 Amazon S3 位置。阈值是在属性 hive.metastore.response.spill.threshold 中定义的。建议不要更改默认值。

  > **Note**
  >
  > 这两个属性可以由 Lambda 环境变量 HMS_URIS 和 SPILL_LOCATION 覆盖。当您希望将 Lambda 函数用于不同的 Hive 元数据仓或溢出位置时，请使用这些变量而不是重新编译该函数的源代码。

- **hms-lambda-layer** – 一个 Maven 程序集项目，用于将 hms-service-api、hms-lambda-handler 及其依赖关系放入一个 .zip 文件中。此 .zip 文件将注册为 Lambda 层以供多个 Lambda 函数使用。

- **`hms-lambda-rnp`** – 记录来自 Lambda 函数的响应，然后将其用于重播响应。您可以使用此模型来模拟 Lambda 响应以进行测试。

自行构建构件

在大多数使用案例中，无需修改参考实施。但是，如有必要，您可以修改源代码、自行构建构件并将其上载到 Amazon S3 位置。

在构建构件之前，请更新 `hms-lambda-func` 模块中 `hms.properties` 文件中的属性 `hive.metastore.uris` 和 `hive.metastore.response.spill.location`。

要构建构件，您必须已安装 Apache Maven 并运行命令 `mvn install`。这将在模块 `hms-lambda-layer` 中的名为 `target` 的输出文件夹中生成层 `.zip` 文件，并在模块 `hms-lambd-func` 中生成 Lambda 函数 `.jar` 文件。

使用 Amazon Athena 联合查询

如果您在 Amazon S3 以外的源中拥有数据，则可以使用 Athena Federated Query 来查询数据或构建从多个数据源提取数据并将其存储在 Amazon S3 中的管道。通过 Athena Federated Query，您可以对存储在关系数据源、非关系数据源、对象数据源和自定义数据源中的数据运行 SQL 查询。

Athena 使用在 AWS Lambda 上运行的数据源连接器来运行联合查询。数据源连接器是一段代码，可以在目标数据源和 Athena 之间进行转换。您可以将连接器视为 Athena 查询引擎的扩展。使用 Apache 2.0 许可证的 Amazon CloudWatch Logs、Amazon DynamoDB、Amazon DocumentDB 和 Amazon RDS 等数据源以及 MySQL 和 PostgreSQL 等兼容 JDBC 的关系数据源中存在预构建的 Athena 数据源连接器。您还可以使用 Athena Query Federation 软件开发工具包编写自定义连接器。要选择、配置数据源连接器并将其部署到您的账户，可以使用 Athena 和 Lambda 控制台或 AWS Serverless Application Repository。部署数据源连接器后，连接器将与可以在 SQL 查询中指定的目录相关联。您可以使用单个查询来组合来自多个目录的 SQL 语句和跨多个数据源。

针对某个数据源提交查询时，Athena 调用相应的连接器以识别需要读取的表部分、管理并行度以及下移筛选器谓词。根据提交查询的用户，连接器可以提供或限制对特定数据元素的访问。连接器使用 Apache Arrow 作为查询中请求的数据的返回格式，这使得连接器能够以 C、C++、Java、Python 和 Rust 等语言实现。由于连接器在 Lambda 中处理，因此它们可用于访问来自云中的任何数据源或可从 Lambda 访问的本地数据源的数据。

要编写自己的数据源连接器，您可以使用 Athena Query Federation 软件开发工具包自定义 Amazon Athena 提供和维护的预构建连接器之一。您可以修改来自 GitHub 存储库的源代码的副本，然后使用连接器发布工具创建您自己的 AWS Serverless Application Repository 软件包。

> **Note**
>
> 第三方开发人员可能已经使用 Athena Query Federation 软件开发工具包来写入数据源连接器。有关这些数据源连接器的支持或许可问题，请与您的连接器提供商联系。这些连接器不受 AWS 的测试或支持。

有关 Athena 编写和测试的数据源连接器列表，请参阅 可用数据来源连接器。

有关编写自己的数据源连接器的信息，请参阅 GitHub 上的 Example Athena connector（示例 Athena 连接器）。

注意事项和限制

- 引擎版本 – Athena Federated Query 仅在 Athena 引擎版本 2 及更高版本中受支持。有关 Athena 引擎版本的更多信息，请参阅 Athena 引擎版本控制。
- 视图 - 您可以创建和查询联合数据来源的视图。联合视图存储在 AWS Glue 中，而不是底层数据来源中。有关更多信息，请参阅 查询联合视图。
- 写入操作 – 写入操作不受支持，例如 INSERT INTO。尝试这样做可能会导致出现错误消息 This operation is currently not supported for external catalogs（外部目录目前不支持此操作）。
- 定价 – 有关定价信息，请参阅 Amazon Athena 定价。

  JDBC 驱动程序 – 若要将 JDBC 驱动程序与联合查询或外部 Hive 元数据仓结合使用，请在 JDBC 连接字符串中加入 `MetadataRetrievalMethod=ProxyAPI`。有关 JDBC 驱动程序的信息，请参阅 通过 JDBC 连接到 Amazon Athena。
- Secrets Manager – 要将 Athena Federated Query 功能与 AWS Secrets Manager 结合使用，您必须为 Secrets Manager 配置 Amazon VPC 私有端点。有关更多信息，请参阅《AWS Secrets Manager 用户指南》中的 创建 Secrets Manager VPC 私有端点。

数据源连接器可能需要访问以下资源才能正常工作。如果您使用预构建的连接器，请检查连接器的信息，以确保您已正确配置 VPC。此外，请确保运行查询和创建连接器的 IAM 委托人具有执行所需操作的权限。有关更多信息，请参阅 允许 Athena 联合查询的 IAM 权限策略示例。

- Amazon S3 – 除了将查询结果写入 Amazon S3 中的 Athena 查询结果位置之外，数据连接器还会写入到 Amazon S3 中的溢出存储桶。此 Amazon S3 位置的连接和权限是必需的。
- Athena – 数据源需要连接到 Athena，反之亦然，以便检查查询状态并防止过扫描。

- AWS Glue Data Catalog – 如果连接器使用 Data Catalog 来获取补充元数据或主元数据，则需要连接和权限。

视频

观看以下视频，了解有关使用 Athena Federated Query 的更多信息。

视频：在 Amazon QuickSight 中分析 Amazon Athena 中的联合查询结果

以下视频说明了如何在 Amazon QuickSight 中分析 Athena 联合查询的结果。

在 Amazon QuickSight 中分析 Amazon Athena 中的联合查询结果

视频：Gaming Analytics Pipeline

以下视频演示了如何部署可扩展的无服务器数据管道，以便使用 Amazon Athena 联合查询从游戏和服务中获取、存储和分析遥测数据。

Gaming Analytics Pipeline

可用数据来源连接器

本节列出了可用于查询 Amazon S3 外部的各种数据来源的预构建 Athena 数据来源连接器。要在 Athena 查询中使用连接器，请对其进行配置并部署到您的账户。

注意事项和限制

- 一些预构建连接器需要您先创建 VPC 和安全组，才能使用该连接器。有关创建 VPC 的信息，请参阅 为数据源连接器创建 VPC。
- 若要将 Athena Federated Query 功能与 AWS Secrets Manager 结合使用，您必须为 Secrets Manager 配置 Amazon VPC 私有端点。有关更多信息，请参阅《AWS Secrets Manager 用户指南》中的 创建 Secrets Manager VPC 私有端点。
- 对于不支持谓词下推的连接器，执行包含谓词的查询花费的时间明显更长。对于小型数据集，扫描的数据很少，因而查询平均需要大约 2 分钟。但是，对于大型数据集，许多查询可能会超时。
- 一些联合数据来源使用术语来指代与 Athena 不同的数据对象。有关更多信息，请参阅 Athena 和联合表名限定词。
- 对于在您列出表时不支持分页的连接器，如果您的数据库中有许多表和元数据，Web 服务可能会超时。以下连接器为列出表提供分页支持：

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

其他信息

- 有关部署 Athena 数据来源连接器的信息，请参阅 部署数据来源连接器。
- 有关使用 Athena 数据来源连接器的查询的信息，请参阅 编写联合查询。
- 有关 Athena 数据来源连接器的完整信息，请参阅 GitHub 上的 Available connectors（可用连接器）。

Athena 数据来源连接器

- Amazon Athena Azure Data Lake Storage（ADLS）Gen2 连接器
- Amazon Athena Azure Synapse 连接器
- Amazon Athena Cloudera Hive 连接器
- Amazon Athena Cloudera Impala 连接器
- Amazon Athena CloudWatch 连接器
- Amazon Athena CloudWatch 指标连接器
- Amazon Athena AWS CMDB 连接器
- Amazon Athena IBM Db2 连接器
- Amazon Athena IBM Db2 AS/400（Db2 iSeries）连接器
- Amazon Athena DocumentDB 连接器
- Amazon Athena DynamoDB 连接器
- Amazon Athena Google BigQuery 连接器
- Amazon Athena Google Cloud Storage 连接器
- Amazon Athena HBase 连接器

- Amazon Athena HortonWorks 连接器
- Amazon Athena Apache Kafka 连接器
- Amazon Athena MSK 连接器
- Amazon Athena MySQL 连接器
- Amazon Athena Neptune 连接器
- Amazon Athena OpenSearch 连接器
- Amazon Athena Oracle 连接器
- Amazon Athena PostgreSQL 连接器
- Amazon Athena Redis 连接器
- Amazon Athena Redshift 连接器
- Amazon Athena SAP HANA 连接器
- Amazon Athena Snowflake 连接器
- Amazon Athena Microsoft SQL Server 连接器
- Amazon Athena Teradata 连接器
- Amazon Athena Timestream 连接器
- Amazon Athena TPC 基准 DS（TPC-DS）连接器
- Amazon Athena Vertica 连接器

> **Note**
>
> AthenaJdbcConnector（最新版本 2022.4.1）已弃用。请改用特定于数据库的连接器，例如 MySQL、Redshift 或 PostgreSQL 的连接器。

Amazon Athena Azure Data Lake Storage（ADLS）Gen2 连接器

适用于 Azure Data Lake Storage（ADLS）Gen2 的 Amazon Athena 连接器使 Amazon Athena 能够对存储在 ADLS 上的数据运行 SQL 查询。Athena 无法直接访问数据湖中存储的文件。

- 工作流程 - 连接器实施 JDBC 接口，以使用 `com.microsoft.sqlserver.jdbc.SQLServerDriver` 驱动程序。连接器将查询传递给 Azure Synapse 引擎，以访问数据湖。

- 数据处理和 S3 - 通常，Lambda 连接器直接查询数据，无需传输到 Amazon S3。但是，如果 Lambda 函数返回的数据超过 Lambda 限值，该数据将写入指定 Amazon S3 溢出存储桶，这样 Athena 就可以读取多余的数据。
- AAD 身份验证 - AAD 可用作 Azure Synapse 连接器的身份验证方法。要使用 AAD，连接器使用的 JDBC 连接字符串必须包含 URL 参数 `authentication=ActiveDirectoryServicePrincipal`、`AADSecurePrincipalId` 和 `AADSecurePrincipalSecret`。这些参数可以直接传入，也可以通过 Secrets Manager 传入。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

限制

- 不支持写入 DDL 操作。
- 在多路复用器设置中，溢出桶和前缀在所有数据库实例之间共享。
- 任何相关的 Lambda 限制。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。
- 必须将筛选条件中的日期和时间戳数据类型转换为适当的数据类型。

术语

以下术语与 Azure Data Lake Storage Gen2 连接器有关。

- 数据库实例 — 部署在本地、Amazon EC2 或 Amazon RDS 上的任何数据库实例。
- 处理程序 — 访问您数据库实例的 Lambda 处理程序。处理程序可以用于元数据或数据记录。
- 元数据处理程序 — 从您的数据库实例中检索元数据的 Lambda 处理程序。
- 记录处理程序 — 从您的数据库实例中检索数据记录的 Lambda 处理程序。
- 复合处理程序 — 从您的数据库实例中检索元数据和数据记录的 Lambda 处理程序。
- 属性或参数 — 处理程序用来提取数据库信息的数据库属性。您可以将这些属性配置为 Lambda 环境变量。
- 连接字符串 — 用于建立数据库实例连接的文本字符串。

- 目录 — 向 Athena 注册的非 AWS Glue 目录，是 connection_string 属性的必要前缀。
- 多路复用处理程序 — 可以接受和使用多个数据库连接的 Lambda 处理程序。

参数

使用本节中的 Lambda 环境变量配置 Azure Data Lake Storage Gen2 连接器。

连接字符串

使用以下格式的 JDBC 连接字符串连接到数据库实例。

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

使用多路复用处理程序

您可以使用多路复用器通过单个 Lambda 函数连接到多个数据库实例。按目录名称来路由请求。在 Lambda 中使用以下类。

处理程序	类
复合处理程序	DataLakeGen2MuxCompositeHandler
元数据处理程序	DataLakeGen2MuxMetadataHandler
记录处理程序	DataLakeGen2MuxRecordHandler

多路复用处理程序参数

参数	描述
$*catalog*_connection_string	必需。数据库实例连接字符串。将 Athena 中使用的目录的名称作为环境变量前缀。例如，如果向 Athena 注册的目录是 mydatalakegentwocatalog ，则环境变量名称是 mydatalakegentwocatalog_connection_string 。
default	必需。默认连接字符串。目录为 lambda:${ *AWS_LAMBDA_FUNCTION_NAME* } 时使用此字符串。

以下示例属性适用于支持两个数据库实例的 DataLakeGen2 MUX Lambda 函数：`datalakegentwo1`（默认）和 `datalakegentwo2`。

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

提供凭证

要在 JDBC 连接字符串中为数据库提供用户名和密码，可以使用连接字符串属性或 AWS Secrets Manager。

- 连接字符串 — 可以将用户名和密码指定为 JDBC 连接字符串中的属性。

  > **Important**
  >
  > 作为安全最佳实践，请勿在您的环境变量或连接字符串中使用硬编码凭证。有关将硬编码密钥移至 AWS Secrets Manager 的信息，请参阅《AWS Secrets Manager 用户指南》中的将硬编码密钥移至 AWS Secrets Manager。

- AWS Secrets Manager - 要将 Athena 联合查询功能与 AWS Secrets Manager 配合使用，连接到您的 Lambda 函数的 VPC 应该拥有互联网访问权限或者 VPC 端点，以连接到 Secrets Manager。

  您可以将 AWS Secrets Manager 中的密钥名称放入您的 JDBC 连接字符串中。连接器将该密钥名称替换为来自 Secrets Manager 的 `username` 和 `password` 值。

对于 Amazon RDS 数据库实例，将紧密集成这种支持。如果您使用 Amazon RDS，我们强烈建议您使用 AWS Secrets Manager 和凭证轮换。如果您的数据库不使用 Amazon RDS，请按以下格式将凭证存储为 JSON：

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

带有密钥名称的示例连接字符串

以下字符串带有密钥名称 ${secret1_name}。

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

该连接器使用该密钥名称来检索密钥，并提供用户名和密码，如以下示例所示。

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

使用单个连接处理程序

您可以使用以下单一连接元数据和记录处理程序连接到单一 Azure Data Lake Storage Gen2 实例。

处理程序类型	类
复合处理程序	DataLakeGen2CompositeHandler
元数据处理程序	DataLakeGen2MetadataHandler
记录处理程序	DataLakeGen2RecordHandler

单个连接处理程序参数

参数	描述
default	必需。默认连接字符串。

单个连接处理程序支持一个数据库实例，并且必须提供 default 连接字符串参数。将忽略所有其他连接字符串。

以下示例属性适用于 Lambda 函数支持的单一 Azure Data Lake Storage Gen2 实例。

属性	值
default	datalakegentwo://jdbc:sqlserver://*hostname*:*port*;databaseName=;${ *secret_name* }

溢出参数

Lambda 开发工具包可以将数据溢出到 Amazon S3。由同一 Lambda 函数访问的所有数据库实例都会溢出到同一位置。

参数	描述
spill_bucket	必需。溢出桶名称。
spill_prefix	必需。溢出桶密钥前缀。
spill_put_request_headers	（可选）用于溢出的 Amazon S3 putObject 请求的请求标头和值的 JSON 编码映射（例如，{"x-amz-server-side-encryption" : "AES256"} ）。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject 。

数据类型支持

下表显示了适用于 ADLS Gen2 和 Arrow 的相应数据类型。

ADLS Gen2	Arrow
bit	TINYINT
tinyint	SMALLINT
smallint	SMALLINT

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

分区和拆分

Azure Data Lake Storage Gen2 使用与 Hadoop 兼容的 Gen2 Blob 存储来存储数据文件。这些文件中的数据将从 Azure Synapse 引擎查询。Azure Synapse 引擎将存储在文件系统中的 Gen2 数据视为外

部表。根据数据类型实施分区。如果已在 Gen2 存储系统中对数据进行了分区和分发，则该连接器将以单个拆分的形式检索数据。

Performance

当同时运行多个查询时，Azure Data Lake Storage Gen2 连接器的查询性能会变慢，并且会受到节流。

Azure Data Lake Storage Gen2 连接器可执行谓词下推，以减少查询扫描的数据量。简单谓词和复杂表达式将下推到连接器，以减少扫描的数据量并缩短查询执行的运行时间。

Predicates

谓词是 SQL 查询的 WHERE 子句中的表达式，其评估结果为布尔值并根据多个条件筛选行。Athena Azure Data Lake Storage Gen2 连接器可以组合这些表达式并将其直接推送到 Azure Data Lake Storage Gen2，以增强功能并减少扫描的数据量。

以下 Athena Azure Data Lake Storage Gen2 连接器运算符支持谓词下推：

- 布尔值：AND、OR、NOT
- 相等：EQUAL、NOT_EQUAL、LESS_THAN、LESS_THAN_OR_EQUAL、GREATER_THAN、GREATER_
- 算术：ADD、SUBTRACT、MULTIPLY、DIVIDE、MODULUS、NEGATE
- 其他：LIKE_PATTERN、IN

组合下推示例

要增强查询功能，组合下推类型，如以下示例所示：

```
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%');
```

传递查询

Azure Data Lake Storage Gen2 连接器支持传递查询。传递查询使用表函数将完整查询下推到数据来源来执行查询。

要在 Azure Data Lake Storage Gen2 中执行传递查询，可以使用以下语法：

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

以下示例查询将查询下推到 Azure Data Lake Storage Gen2 中的数据来源。该查询选择了 `customer` 表中的所有列，将结果限制为 10。

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

许可证信息

使用此连接器，即表示您确认包含第三方组件（这些组件的列表可在此连接器的 pom.xml 文件中找到），并同意 GitHub.com 上的 LICENSE.txt 文件中提供的相应第三方许可证中的条款。

其他 资源

有关最新 JDBC 驱动程序版本信息，请参阅 GitHub.com 上适用于 Azure Data Lake Storage Gen2 连接器的 pom.xml 文件。

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena Azure Synapse 连接器

适用于 Azure Synapse Analytics 的 Amazon Athena 连接器使 Amazon Athena 能够使用 JDBC 在您的 Azure Synapse 数据库上运行 SQL 查询。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

限制

- 不支持写入 DDL 操作。

- 在多路复用器设置中，溢出桶和前缀在所有数据库实例之间共享。
- 任何相关的 Lambda 限制。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。
- 在筛选条件中，必须将 Date 和 Timestamp 数据类型转换为适当的数据类型。
- 要搜索类型 Real 和 Float 的负值，请使用 <= 或 >= 运算符。
- 不支持 binary、varbinary、image 和 rowversion 数据类型。

术语

以下术语与 Synapse 连接器有关。

- 数据库实例 — 部署在本地、Amazon EC2 或 Amazon RDS 上的任何数据库实例。
- 处理程序 — 访问您数据库实例的 Lambda 处理程序。处理程序可以用于元数据或数据记录。
- 元数据处理程序 — 从您的数据库实例中检索元数据的 Lambda 处理程序。
- 记录处理程序 — 从您的数据库实例中检索数据记录的 Lambda 处理程序。
- 复合处理程序 — 从您的数据库实例中检索元数据和数据记录的 Lambda 处理程序。
- 属性或参数 — 处理程序用来提取数据库信息的数据库属性。您可以将这些属性配置为 Lambda 环境变量。
- 连接字符串 — 用于建立数据库实例连接的文本字符串。
- 目录 — 向 Athena 注册的非 AWS Glue 目录，是 connection_string 属性的必要前缀。
- 多路复用处理程序 — 可以接受和使用多个数据库连接的 Lambda 处理程序。

参数

使用本节中的 Lambda 环境变量来配置 Synapse 连接器。

连接字符串

使用以下格式的 JDBC 连接字符串连接到数据库实例。

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

使用多路复用处理程序

您可以使用多路复用器通过单个 Lambda 函数连接到多个数据库实例。按目录名称来路由请求。在 Lambda 中使用以下类。

处理程序	类
复合处理程序	SynapseMuxCompositeHandler
元数据处理程序	SynapseMuxMetadataHandler
记录处理程序	SynapseMuxRecordHandler

多路复用处理程序参数

参数	描述
$*catalog*_connection_string	必需。数据库实例连接字符串。将 Athena 中使用的目录的名称作为环境变量前缀。例如，如果向 Athena 注册的目录是 mysynapsecatalog ，则环境变量名称是 mysynapsecatalog_connection_string 。
default	必需。默认连接字符串。目录为 lambda:${ *AWS_LAMBDA_FUNCTION_NAME* } 时使用此字符串。

以下示例属性适用于支持两个数据库实例的 Synapse MUX Lambda 函数：synapse1（默认）和 synapse2。

属性	值
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_c	synapse://jdbc:synapse://synapse2.hostname:port;databaseName= *<database_name>* ;${*secret2_name* }

属性	值
onnection _string	

提供凭证

要在 JDBC 连接字符串中为数据库提供用户名和密码，可以使用连接字符串属性或 AWS Secrets Manager。

- 连接字符串 — 可以将用户名和密码指定为 JDBC 连接字符串中的属性。

  > **Important**
  >
  > 作为安全最佳实践，请勿在您的环境变量或连接字符串中使用硬编码凭证。有关将硬编码密钥移至 AWS Secrets Manager 的信息，请参阅《AWS Secrets Manager 用户指南》中的将硬编码密钥移至 AWS Secrets Manager。

- AWS Secrets Manager - 要将 Athena 联合查询功能与 AWS Secrets Manager 配合使用，连接到您的 Lambda 函数的 VPC 应该拥有互联网访问权限或者 VPC 端点，以连接到 Secrets Manager。

  您可以将 AWS Secrets Manager 中的密钥名称放入您的 JDBC 连接字符串中。连接器将该密钥名称替换为来自 Secrets Manager 的 `username` 和 `password` 值。

  对于 Amazon RDS 数据库实例，将紧密集成这种支持。如果您使用 Amazon RDS，我们强烈建议您使用 AWS Secrets Manager 和凭证轮换。如果您的数据库不使用 Amazon RDS，请按以下格式将凭证存储为 JSON：

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

带有密钥名称的示例连接字符串

以下字符串带有密钥名称 ${secret_name}。

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

该连接器使用该密钥名称来检索密钥，并提供用户名和密码，如以下示例所示。

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

使用单个连接处理程序

您可以使用以下单个连接元数据和记录处理程序连接到单个 Synapse 实例。

处理程序类型	类
复合处理程序	SynapseCompositeHandler
元数据处理程序	SynapseMetadataHandler
记录处理程序	SynapseRecordHandler

单个连接处理程序参数

参数	描述
default	必需。默认连接字符串。

单个连接处理程序支持一个数据库实例，并且必须提供 default 连接字符串参数。将忽略所有其他连接字符串。

以下示例属性适用于 Lambda 函数支持的单个 Synapse 实例。

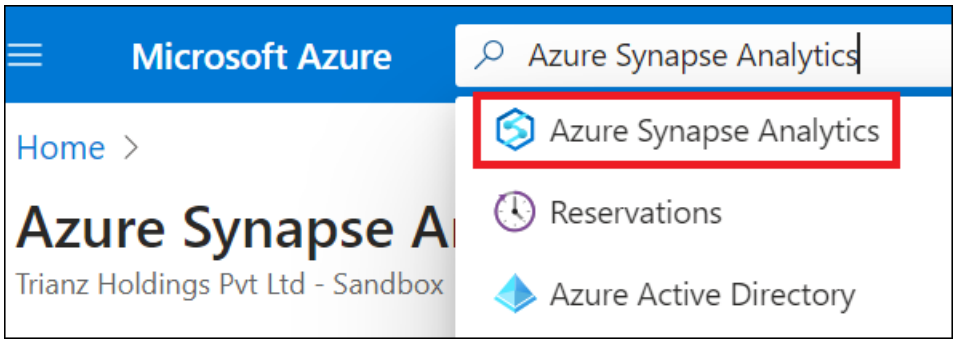
属性	值
defa	synapse://jdbc:sqlserver://hostname:port;databaseName= <database_name> ;${secret_name }

配置 Active Directory 身份验证

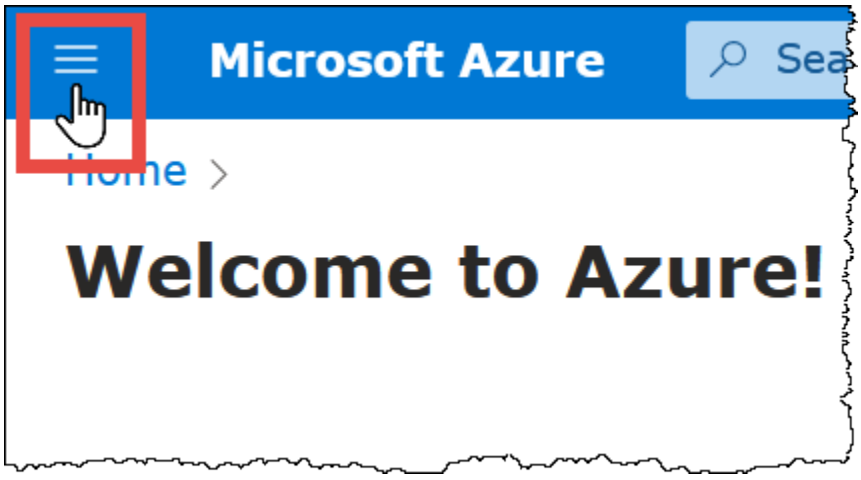
Amazon Athena Azure Synapse 连接器支持 Microsoft Active Directory 身份验证。在开始之前，必须在 Microsoft Azure 门户中配置管理用户，然后使用 AWS Secrets Manager 创建密钥。

设置 Active Directory 管理用户

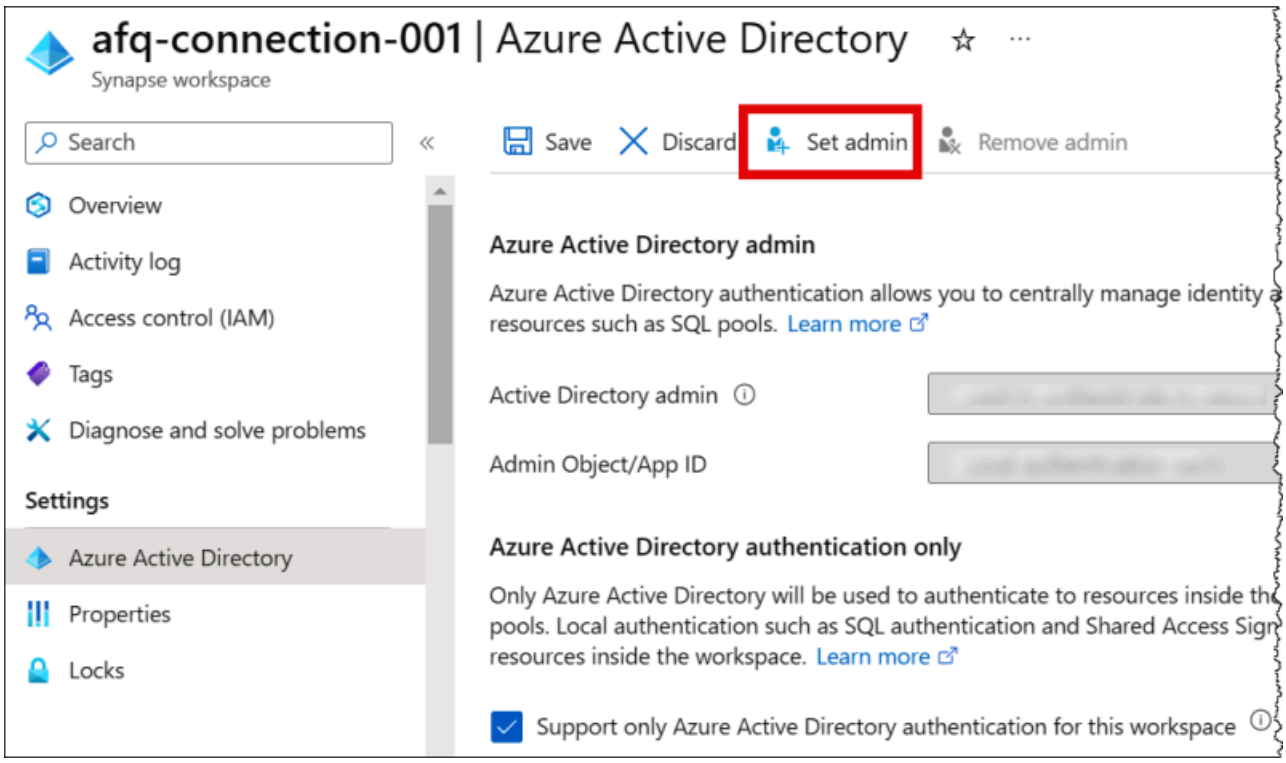
1. 使用具有管理权限的帐户，登录 Microsoft Azure 门户，网址为 https://portal.azure.com/。
2. 在搜索框中，输入 Azure Synapse Analytics，然后选择 Azure Synapse Analytics。



3. 打开左侧菜单。



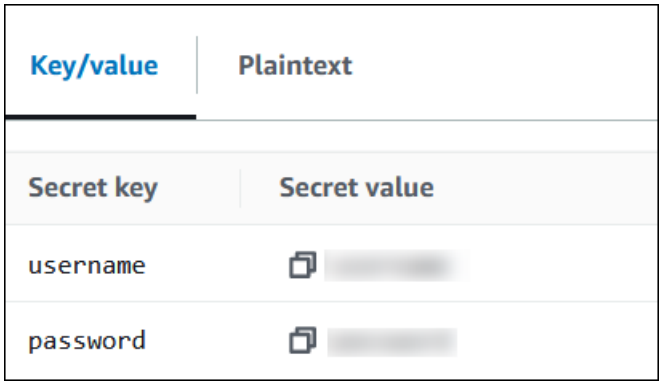
4. 在导航窗格中，选择 Azure Active Directory。
5. 在设置管理员选项卡中，将 Active Directory 管理员设置为新用户或现有用户。



6. 在 AWS Secrets Manager 中，存储管理员用户名和密码凭证。有关在 Secrets Manager 中创建密钥的信息，请参阅创建 AWS Secrets Manager 密钥。

在 Secrets Manager 中查看密钥

1. 打开 Secrets Manager 控制台，网址为 https://console.aws.amazon.com/secretsmanager/。
2. 在导航窗格中，选择 Secrets（密钥）。
3. 在 Secrets（密钥）页面，选择密钥链接。
4. 在密钥的详细信息页面上，选择 Retrieve secret value（检索密钥值）。



修改连接字符串

要为连接器启用 Active Directory 身份验证，使用以下语法修改连接字符串：

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

使用 ActiveDirectoryServicePrincipal

Amazon Athena Azure Synapse 连接器还支持 `ActiveDirectoryServicePrincipal`。要启用此项，如下所示修改连接字符串。

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

对于 `secret_name`，将应用程序或客户端 ID 指定为用户名，并在密码中指定服务主体身份的密钥。

溢出参数

Lambda 开发工具包可以将数据溢出到 Amazon S3。由同一 Lambda 函数访问的所有数据库实例都会溢出到同一位置。

参数	描述
`spill_bucket`	必需。溢出桶名称。
`spill_prefix`	必需。溢出桶密钥前缀。
`spill_put_request_headers`	（可选）用于溢出的 Amazon S3 `putObject` 请求的请求标头和值的 JSON 编码映射（例如，`{"x-amz-server-side-encryption" : "AES256"}`)。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。

数据类型支持

下表显示了适用于 Synapse 和 Apache Arrow 的相应数据类型。

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)
time	VARCHAR
datetimeoffset	Date(MILLISECOND)
char[n]	VARCHAR
varchar[n/max]	VARCHAR

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

分区和拆分

分区由类型为 `varchar` 的单个分区列表示。Synapse 支持范围分区，因此分区通过从 Synapse 元数据表中提取分区列和分区范围来实现。这些范围值用于创建拆分。

Performance

选择列的子集会显著减少查询运行时。由于并发，连接器显示出严重的节流。

Athena Synapse 连接器可执行谓词下推，以减少查询扫描的数据量。简单谓词和复杂表达式将下推到连接器，以减少扫描的数据量并缩短查询执行的运行时间。

Predicates

谓词是 SQL 查询的 `WHERE` 子句中的表达式，其评估结果为布尔值并根据多个条件筛选行。Athena Synapse 连接器可以组合这些表达式并将其直接推送到 Synapse，以增强功能并减少扫描的数据量。

以下 Athena Synapse 连接器运算符支持谓词下推：

- 布尔值：AND、OR、NOT
- 相等：EQUAL、NOT_EQUAL、LESS_THAN、LESS_THAN_OR_EQUAL、GREATER_THAN、GREATER_
- 算术：ADD、SUBTRACT、MULTIPLY、DIVIDE、MODULUS、NEGATE
- 其他：LIKE_PATTERN、IN

组合下推示例

要增强查询功能，组合下推类型，如以下示例所示：

```
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%');
```

传递查询

Synapse 连接器支持传递查询。传递查询使用表函数将完整查询下推到数据来源来执行查询。

要在 Synapse 中执行传递查询，可以使用以下语法：

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

以下示例查询将查询下推到 Synapse 中的数据来源。该查询选择了 customer 表中的所有列，将结果限制为 10。

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

许可证信息

使用此连接器，即表示您确认包含第三方组件（这些组件的列表可在此连接器的 pom.xml 文件中找到），并同意 GitHub.com 上的 LICENSE.txt 文件中提供的相应第三方许可证中的条款。

其他 资源

- 有关介绍如何使用 Amazon QuickSight 和 Amazon Athena 联合查询对存储在 Microsoft Azure Synapse 数据库中的数据构建控制面板和可视化的文章，请参阅 AWS 大数据博客中的使用 Amazon QuickSight、Amazon Athena 联合查询和 Microsoft Azure Synapse 执行多云分析。
- 有关最新 JDBC 驱动程序版本信息，请参见 GitHub.com 上适用于 Synapse 连接器的 pom.xml 文件。
- 有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena Cloudera Hive 连接器

适用于 Cloudera Hive 的 Amazon Athena 连接器使 Athena 能够对 Cloudera Hive Hadoop 分配运行 SQL 查询。该连接器会将您的 Athena SQL 查询转换为其等效的 HiveQL 语法。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。
- 使用此连接器前，请先设置 VPC 和安全组。有关更多信息，请参阅 为数据源连接器创建 VPC。

限制

- 不支持写入 DDL 操作。
- 在多路复用器设置中，溢出桶和前缀在所有数据库实例之间共享。
- 任何相关的 Lambda 限制。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。

术语

以下术语与 Cloudera Hive 连接器有关。

- 数据库实例 - 部署在本地、Amazon EC2 或 Amazon RDS 上的任何数据库实例。
- 处理程序 — 访问您数据库实例的 Lambda 处理程序。处理程序可以用于元数据或数据记录。
- 元数据处理程序 — 从您的数据库实例中检索元数据的 Lambda 处理程序。
- 记录处理程序 — 从您的数据库实例中检索数据记录的 Lambda 处理程序。
- 复合处理程序 — 从您的数据库实例中检索元数据和数据记录的 Lambda 处理程序。
- 属性或参数 — 处理程序用来提取数据库信息的数据库属性。您可以将这些属性配置为 Lambda 环境变量。
- 连接字符串 — 用于建立数据库实例连接的文本字符串。
- 目录 — 向 Athena 注册的非 AWS Glue 目录，是 `connection_string` 属性的必要前缀。
- 多路复用处理程序 — 可以接受和使用多个数据库连接的 Lambda 处理程序。

参数

使用本节中的 Lambda 环境变量来配置 Cloudera Hive 连接器。

连接字符串

使用以下格式的 JDBC 连接字符串连接到数据库实例。

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

使用多路复用处理程序

您可以使用多路复用器通过单个 Lambda 函数连接到多个数据库实例。按目录名称来路由请求。在 Lambda 中使用以下类。

处理程序	类
复合处理程序	HiveMuxCompositeHandler
元数据处理程序	HiveMuxMetadataHandler
记录处理程序	HiveMuxRecordHandler

多路复用处理程序参数

参数	描述
$*catalog*_connection_string	必需。数据库实例连接字符串。将 Athena 中使用的目录的名称作为环境变量前缀。例如，如果向 Athena 注册的目录是 myhivecatalog ，则环境变量名称是 myhivecatalog_connection_string 。
default	必需。默认连接字符串。目录为 lambda:${ *AWS_LAMBDA_FUNCTION_NAME* } 时使用此字符串。

以下示例属性适用于支持两个数据库实例的 Hive MUX Lambda 函数：hive1（默认）和 hive2。

属性	值
default	hive://jdbc:hive2://hive1:10000/default?${Test/RDS/hive1}
hive2_catalog1_connection_string	hive://jdbc:hive2://hive1:10000/default?${Test/RDS/hive1}

属性	值
hive2_catalog2_connection_string	hive://jdbc:hive2://hive2:10000/default?UID=sample&PWD=sample

提供凭证

要在 JDBC 连接字符串中为数据库提供用户名和密码，可以使用连接字符串属性或 AWS Secrets Manager。

- 连接字符串 — 可以将用户名和密码指定为 JDBC 连接字符串中的属性。

  > **Important**
  >
  > 作为安全最佳实践，请勿在您的环境变量或连接字符串中使用硬编码凭证。有关将硬编码密钥移至 AWS Secrets Manager 的信息，请参阅《AWS Secrets Manager 用户指南》中的将硬编码密钥移至 AWS Secrets Manager。

- AWS Secrets Manager - 要将 Athena 联合查询功能与 AWS Secrets Manager 配合使用，连接到您的 Lambda 函数的 VPC 应该拥有互联网访问权限或者 VPC 端点，以连接到 Secrets Manager。

  您可以将 AWS Secrets Manager 中的密钥名称放入您的 JDBC 连接字符串中。连接器将该密钥名称替换为来自 Secrets Manager 的 `username` 和 `password` 值。

  对于 Amazon RDS 数据库实例，将紧密集成这种支持。如果您使用 Amazon RDS，我们强烈建议您使用 AWS Secrets Manager 和凭证轮换。如果您的数据库不使用 Amazon RDS，请按以下格式将凭证存储为 JSON：

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

带有密钥名称的示例连接字符串

以下字符串带有密钥名称 `${Test/RDS/hive1}`。

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

该连接器使用该密钥名称来检索密钥，并提供用户名和密码，如以下示例所示。

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

目前，Cloudera Hive 连接器可以识别 UID 和 PWD JDBC 属性。

使用单个连接处理程序

您可以使用以下单个连接元数据和记录处理程序连接到单个 Cloudera Hive 实例。

处理程序类型	类
复合处理程序	HiveCompositeHandler
元数据处理程序	HiveMetadataHandler
记录处理程序	HiveRecordHandler

单个连接处理程序参数

参数	描述
default	必需。默认连接字符串。

单个连接处理程序支持一个数据库实例，并且必须提供 default 连接字符串参数。将忽略所有其他连接字符串。

以下示例属性适用于 Lambda 函数支持的单个 Cloudera Hive 实例。

属性	值
默认值	hive://jdbc:hive2://hive1:10000/default?secret=${Test/RDS/hive1}

溢出参数

Lambda 开发工具包可以将数据溢出到 Amazon S3。由同一 Lambda 函数访问的所有数据库实例都会溢出到同一位置。

参数	描述
spill_bucket	必需。溢出桶名称。
spill_prefix	必需。溢出桶密钥前缀。
spill_put_request_headers	（可选）用于溢出的 Amazon S3 putObject 请求的请求标头和值的 JSON 编码映射（例如，{"x-amz-server-side-encryption" : "AES256"})。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。

数据类型支持

下表显示了适用于 JDBC、Cloudera Hive 和 Arrow 的相应数据类型。

JDBC	Cloudera Hive	Arrow
布尔值	布尔值	位
整数	TINYINT	Tiny
短型	SMALLINT	Smallint
整数	INT	Int
长整型	BIGINT	Bigint
float	float4	Float4
Double	float8	Float8
Date	date	DateDay
Timestamp	时间戳	DateMilli
String	VARCHAR	Varchar
字节	bytes	Varbinary

JDBC	Cloudera Hive	Arrow
BigDecimal	十进制	十进制
ARRAY	不适用（见注释）	列出

> **Note**
>
> 目前，Cloudera Hive 不支持聚合类型 ARRAY、MAP、STRUCT 或 UNIONTYPE。聚合类型的列被视为 SQL 中的 VARCHAR 列。

分区和拆分

分区用于确定如何为该连接器生成拆分。Athena 将构建一个 `varchar` 类型的合成列，它将展示表的分区方案，以帮助该连接器生成拆分。该连接器不会修改实际的表定义。

Performance

Cloudera Hive 支持静态分区。Athena Cloudera Hive 连接器可从这些分区并行检索数据。如果您想查询具有均匀分区分布的非常大的数据集，强烈建议使用静态分区。Cloudera Hive 连接器能够灵活地应对并发造成的节流。

Athena Cloudera Hive 连接器执行谓词下推，以减少查询扫描的数据量。`LIMIT` 子句、简单谓词和复杂表达式将下推到连接器，以减少扫描数据量并缩短查询执行的运行时间。

LIMIT 子句

`LIMIT N` 语句用于减少查询扫描的数据量。`LIMIT N` 下推时，连接器仅向 Athena 返回 `N` 行。

Predicates

谓词是 SQL 查询的 `WHERE` 子句中的表达式，其评估结果为布尔值并根据多个条件筛选行。Athena Cloudera Hive 连接器可以组合这些表达式并将其直接推送到 Cloudera Hive，以增强功能并减少扫描的数据量。

以下 Athena Cloudera Hive 连接器运算符支持谓词下推：

- 布尔值：AND、OR、NOT

- 相
  等：EQUAL、NOT_EQUAL、LESS_THAN、LESS_THAN_OR_EQUAL、GREATER_THAN、GREATER_
- 算术：ADD、SUBTRACT、MULTIPLY、DIVIDE、MODULUS、NEGATE
- 其他：LIKE_PATTERN、IN

组合下推示例

要增强查询功能，组合下推类型，如以下示例所示：

```
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;
```

传递查询

Cloudera Hive 连接器支持传递查询。传递查询使用表函数将完整查询下推到数据来源来执行查询。

要在 Cloudera Hive 中执行传递查询，可以使用以下语法：

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

以下示例查询将查询下推到 Cloudera Hive 中的数据来源。该查询选择了 `customer` 表中的所有列，将结果限制为 10。

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

许可证信息

使用此连接器，即表示您确认包含第三方组件（这些组件的列表可在此连接器的 pom.xml 文件中找到），并同意 GitHub.com 上的 LICENSE.txt 文件中提供的相应第三方许可证中的条款。

其他 资源

有关最新 JDBC 驱动程序版本信息，请参阅 GitHub.com 上适用于 Cloudera Hive 连接器的 pom.xml 文件。

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena Cloudera Impala 连接器

Amazon Athena Cloudera Impala 连接器可让 Athena 在 Cloudera Impala 发行版上运行 SQL 查询。该连接器可将 Athena SQL 查询转换为等效的 Impala 语法。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。
- 使用此连接器前，请先设置 VPC 和安全组。有关更多信息，请参阅 为数据源连接器创建 VPC。

限制

- 不支持写入 DDL 操作。
- 在多路复用器设置中，溢出桶和前缀在所有数据库实例之间共享。
- 任何相关的 Lambda 限制。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。

术语

以下术语与 Cloudera Impala 连接器有关。

- 数据库实例 - 部署在本地、Amazon EC2 或 Amazon RDS 上的任何数据库实例。
- 处理程序 — 访问您数据库实例的 Lambda 处理程序。处理程序可以用于元数据或数据记录。
- 元数据处理程序 — 从您的数据库实例中检索元数据的 Lambda 处理程序。
- 记录处理程序 — 从您的数据库实例中检索数据记录的 Lambda 处理程序。
- 复合处理程序 — 从您的数据库实例中检索元数据和数据记录的 Lambda 处理程序。
- 属性或参数 — 处理程序用来提取数据库信息的数据库属性。您可以将这些属性配置为 Lambda 环境变量。

- 连接字符串 — 用于建立数据库实例连接的文本字符串。
- 目录 — 向 Athena 注册的非 AWS Glue 目录，是 connection_string 属性的必要前缀。
- 多路复用处理程序 — 可以接受和使用多个数据库连接的 Lambda 处理程序。

参数

使用本节中的 Lambda 环境变量来配置 Cloudera Impala 连接器。

连接字符串

使用以下格式的 JDBC 连接字符串连接到 Impala 集群。

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

使用多路复用处理程序

您可以使用多路复用器通过单个 Lambda 函数连接到多个数据库实例。按目录名称来路由请求。在 Lambda 中使用以下类。

处理程序	类
复合处理程序	ImpalaMuxCompositeHandler
元数据处理程序	ImpalaMuxMetadataHandler
记录处理程序	ImpalaMuxRecordHandler

多路复用处理程序参数

参数	描述
$*catalog*_connection_string	必需。Athena 目录的 Impala 集群连接字符串。将 Athena 中使用的目录的名称作为环境变量前缀。例如，如果向 Athena 注册的目录是 myimpalacatalog ，则环境变量名称是 myimpalacatalog_connection_string 。
default	必需。默认连接字符串。目录为 lambda:${ *AWS_LAMBDA_FUNCTION_NAME* } 时使用此字符串。

以下示例属性适用于支持两个数据库实例的 Impala MUX Lambda 函数：`impala1`（默认）和 `impala2`。

属性	值
`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`

提供凭证

要在 JDBC 连接字符串中为数据库提供用户名和密码，可以使用连接字符串属性或 AWS Secrets Manager。

- 连接字符串 — 可以将用户名和密码指定为 JDBC 连接字符串中的属性。

  > **Important**
  >
  > 作为安全最佳实践，请勿在您的环境变量或连接字符串中使用硬编码凭证。有关将硬编码密钥移至 AWS Secrets Manager 的信息，请参阅《AWS Secrets Manager 用户指南》中的将硬编码密钥移至 AWS Secrets Manager。

- AWS Secrets Manager - 要将 Athena 联合查询功能与 AWS Secrets Manager 配合使用，连接到您的 Lambda 函数的 VPC 应该拥有互联网访问权限或者 VPC 端点，以连接到 Secrets Manager。

  您可以将 AWS Secrets Manager 中的密钥名称放入您的 JDBC 连接字符串中。连接器将该密钥名称替换为来自 Secrets Manager 的 `username` 和 `password` 值。

  对于 Amazon RDS 数据库实例，将紧密集成这种支持。如果您使用 Amazon RDS，我们强烈建议您使用 AWS Secrets Manager 和凭证轮换。如果您的数据库不使用 Amazon RDS，请按以下格式将凭证存储为 JSON：

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

带有密钥名称的示例连接字符串

以下字符串带有密钥名称 ${Test/impala1host}。

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

该连接器使用该密钥名称来检索密钥，并提供用户名和密码，如以下示例所示。

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

目前，Cloudera Impala 可以识别 UID 和 PWD JDBC 属性。

使用单个连接处理程序

您可以使用以下单个连接元数据和记录处理程序连接到单个 Cloudera Impala 实例。

处理程序类型	类
复合处理程序	ImpalaCompositeHandler
元数据处理程序	ImpalaMetadataHandler
记录处理程序	ImpalaRecordHandler

单个连接处理程序参数

参数	描述
default	必需。默认连接字符串。

单个连接处理程序支持一个数据库实例，并且必须提供 default 连接字符串参数。将忽略所有其他连接字符串。

以下示例属性适用于 Lambda 函数支持的单个 Cloudera Impala 实例。

属性	值
default	impala://jdbc:impala://Impala1host:21050/?secret=${Test/impala1host}

溢出参数

Lambda 开发工具包可以将数据溢出到 Amazon S3。由同一 Lambda 函数访问的所有数据库实例都会溢出到同一位置。

参数	描述
spill_bucket	必需。溢出桶名称。
spill_prefix	必需。溢出桶密钥前缀。
spill_put_request_headers	（可选）用于溢出的 Amazon S3 putObject 请求的请求标头和值的 JSON 编码映射（例如，{"x-amz-server-side-encryption" : "AES256"} ）。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。

数据类型支持

下表显示了适用于 JDBC、Cloudera Impala 和 Arrow 的相应数据类型。

JDBC	Cloudera Impala	Arrow
布尔值	布尔值	位
整数	TINYINT	Tiny
短型	SMALLINT	Smallint
整数	INT	Int
长整型	BIGINT	Bigint

JDBC	Cloudera Impala	Arrow
float	float4	Float4
Double	float8	Float8
Date	date	DateDay
Timestamp	时间戳	DateMilli
String	VARCHAR	Varchar
字节	bytes	Varbinary
BigDecimal	十进制	十进制
ARRAY	不适用（见注释）	列出

> **Note**
>
> 目前，Cloudera Impala 不支持聚合类型 ARRAY、MAP、STRUCT 或 UNIONTYPE。聚合类型的列被视为 SQL 中的 VARCHAR 列。

分区和拆分

分区用于确定如何为该连接器生成拆分。Athena 将构建一个 varchar 类型的合成列，它将展示表的分区方案，以帮助该连接器生成拆分。该连接器不会修改实际的表定义。

Performance

Cloudera Impala 支持静态分区。Athena Cloudera Impala 连接器可从这些分区并行检索数据。如果您想查询具有均匀分区分布的非常大的数据集，强烈建议使用静态分区。Cloudera Impala 连接器能够灵活地应对并发造成的节流。

Athena Cloudera Impala 连接器执行谓词下推，以减少查询扫描的数据量。LIMIT 子句、简单谓词和复杂表达式将下推到连接器，以减少扫描数据量并缩短查询执行的运行时间。

LIMIT 子句

LIMIT N 语句用于减少查询扫描的数据量。LIMIT N 下推时，连接器仅向 Athena 返回 N 行。

Predicates

谓词是 SQL 查询的 WHERE 子句中的表达式，其评估结果为布尔值并根据多个条件筛选行。Athena Cloudera Impala 连接器可以组合这些表达式并将其直接推送到 Cloudera Impala，以增强功能并减少扫描的数据量。

以下 Athena Cloudera Impala 连接器运算符支持谓词下推：

- 布尔值：AND、OR、NOT
- 相等：EQUAL、NOT_EQUAL、LESS_THAN、LESS_THAN_OR_EQUAL、GREATER_THAN、GREATER_
- 算术：ADD、SUBTRACT、MULTIPLY、DIVIDE、MODULUS、NEGATE
- 其他：LIKE_PATTERN、IN

组合下推示例

要增强查询功能，组合下推类型，如以下示例所示：

```
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;
```

传递查询

Cloudera Impala 连接器支持传递查询。传递查询使用表函数将完整查询下推到数据来源来执行查询。

要在 Cloudera Impala 中执行传递查询，可以使用以下语法：

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

以下示例查询将查询下推到 Cloudera Impala 中的数据来源。该查询选择了 customer 表中的所有列，将结果限制为 10。

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

许可证信息

使用此连接器，即表示您确认包含第三方组件（这些组件的列表可在此连接器的 pom.xml 文件中找到），并同意 GitHub.com 上的 LICENSE.txt 文件中提供的相应第三方许可证中的条款。

其他 资源

有关最新 JDBC 驱动程序版本信息，请参阅 GitHub.com 上适用于 Cloudera Impala 连接器的 pom.xml文件。

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena CloudWatch 连接器

Amazon Athena CloudWatch 连接器使 Amazon Athena 可以与 CloudWatch 通信，以便您可以使用 SQL 查询日志数据。

连接器将您的 LogGroup 映射为架构，并将每个 LogStream 映射为表。连接器还映射一个特殊的 `all_log_streams` 视图，其中包含 LogGroup 中的所有 LogStream。此视图使您能够一次性查询 LogGroup 中的所有日志，而不是单个地搜索每个 LogStream。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

参数

使用本节中的 Lambda 环境变量来配置 CloudWatch 连接器。

- spill_bucket - 为超出 Lambda 函数限制的数据指定 Amazon S3 存储桶。
- spill_prefix -（可选）默认为指定 `spill_bucket`（称为 `athena-federation-spill`）中的子文件夹。我们建议您在此位置配置 Amazon S3 存储生命周期，以删除早于预定天数或小时数的溢出内容。

- spill_put_request_headers — （可选）用于溢出的 Amazon S3 `putObject` 请求的请求标头和值的 JSON 编码映射（例如 `{"x-amz-server-side-encryption" : "AES256"}`）。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。
- kms_key_id - （可选）默认情况下，将使用经过 AES-GCM 身份验证的加密模式和随机生成的密钥对溢出到 Amazon S3 的任何数据进行加密。要让您的 Lambda 函数使用 KMS 生成的更强的加密密钥（如 `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`），您可以指定 KMS 密钥 ID。
- disable_spill_encryption - （可选）当设置为 `True` 时，将禁用溢出加密。默认值为 `False`，此时将使用 AES-GCM 对溢出到 S3 的数据使用进行加密 - 使用随机生成的密钥，或者使用 KMS 生成密钥。禁用溢出加密可以提高性能，尤其是当您的溢出位置使用服务器端加密时。

该连接器还支持 AIMD 拥塞控制，以通过 Amazon Athena Query Federation SDK `ThrottlingInvoker`构造处理来自 CloudWatch 的节流事件。您可以通过设置以下任何可选环境变量来调整默认的节流行为：

- throttle_initial_delay_ms - 在第一个拥塞事件之后应用的初始调用延迟。默认为 10 毫秒。
- throttle_max_delay_ms - 调用之间的最大延迟时间。您可以通过将其分成 1000 毫秒来推导 TPS。默认为 1000 毫秒。
- throttle_decrease_factor – Athena 降低调用速率的因子。默认值为 0.5。
- throttle_increase_ms – Athena 减少调用延迟的速率。默认为 10 毫秒。

数据库和表

Athena CloudWatch 连接器将您的 LogGroup 映射为架构（即数据库），并将每个 LogStream 映射为表。连接器还映射一个特殊的 `all_log_streams` 视图，其中包含 LogGroup 中的所有 LogStream。此视图使您能够一次性查询 LogGroup 中的所有日志，而不是单个地搜索每个 LogStream。

Athena CloudWatch 连接器映射的每个表都有以下架构。此架构与 CloudWatch 日志提供的字段相匹配。

- log_stream - `VARCHAR`，包含该行来自的 LogStream 的名称。
- 时间 - `INT64`，包含生成日志行时的纪元时间。
- 消息 - `VARCHAR`，包含日志消息。

示例

以下示例演示了如何对指定的 LogStream 执行 `SELECT` 查询。

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

以下示例演示了如何使用 `all_log_streams` 视图对指定 LogGroup 中的所有 LogStream 执行查询。

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

所需权限

要获取有关此连接器所需 IAM policy 的完整详细信息，请查看 athena-cloudwatch.yaml 文件的 `Policies` 部分。以下列表汇总了所需的权限。

- Amazon S3 写入权限 – 连接器需要对 Amazon S3 中的位置具有写入权限，以溢出大型查询的结果。
- Athena GetQueryExecution – 当上游 Athena 查询终止时，该连接器将使用此权限快速失败。
- CloudWatch 日志读写 - 连接器使用此权限读取您的日志数据并写入其诊断日志。

Performance

Athena CloudWatch 连接器尝试通过并行扫描查询所需的日志流来优化针对 CloudWatch 的查询。在特定时间段筛选条件下，谓词下推既在 Lambda 函数内执行，也在 CloudWatch 日志中执行。

为了获得最佳性能，日志组名称和日志流名称仅限使用小写。使用混合大小写会使连接器执行不区分大小写的搜索，这种搜索的计算密集度更高。

传递查询

CloudWatch 连接器支持使用 CloudWatch Logs Insights 查询语法的传递查询。有关 CloudWatch Logs Insights 的更多信息，请参阅《Amazon CloudWatch Logs 用户指南》中的使用 CloudWatch Logs Insights 分析日志数据。

要使用 Cloudera 创建传递查询，请使用以下语法：

```
SELECT * FROM TABLE(
        system.query(
            STARTTIME => 'start_time',
```

```
            ENDTIME => 'end_time',
            QUERYSTRING => 'query_string',
            LOGGROUPNAMES => 'log_group-names',
            LIMIT => 'max_number_of_results'
        ))
```

以下示例 CloudWatch 传递查询会筛选不等于 1000 时的 duration 字段。

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

许可证信息

Amazon Athena CloudWatch 连接器项目已获得 Apache-2.0 许可证授权。

其他 资源

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena CloudWatch 指标连接器

借助 Amazon Athena CloudWatch 指标连接器，Amazon Athena 可以使用 SQL 查询 CloudWatch 指标数据。

有关从 Athena 本身向 CloudWatch 发布查询指标的信息，请参阅 使用 CloudWatch 指标和事件控制成本和监控查询。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

参数

使用本节中的 Lambda 环境变量来配置 CloudWatch 指标连接器。

- spill_bucket - 为超出 Lambda 函数限制的数据指定 Amazon S3 存储桶。
- spill_prefix - （可选）默认为指定 `spill_bucket`（称为 `athena-federation-spill`）中的子文件夹。我们建议您在此位置配置 Amazon S3 存储生命周期，以删除早于预定天数或小时数的溢出内容。
- spill_put_request_headers —（可选）用于溢出的 Amazon S3 `putObject` 请求的请求标头和值的 JSON 编码映射（例如 `{"x-amz-server-side-encryption" : "AES256"}`）。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。
- kms_key_id - （可选）默认情况下，将使用经过 AES-GCM 身份验证的加密模式和随机生成的密钥对溢出到 Amazon S3 的任何数据进行加密。要让您的 Lambda 函数使用 KMS 生成的更强的加密密钥（如 `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`），您可以指定 KMS 密钥 ID。
- disable_spill_encryption - （可选）当设置为 `True` 时，将禁用溢出加密。默认值为 `False`，此时将使用 AES-GCM 对溢出到 S3 的数据使用进行加密 - 使用随机生成的密钥，或者使用 KMS 生成密钥。禁用溢出加密可以提高性能，尤其是当您的溢出位置使用服务器端加密时。

该连接器还支持 AIMD 拥塞控制，以通过 Amazon Athena Query Federation SDK `ThrottlingInvoker`构造处理来自 CloudWatch 的节流事件。您可以通过设置以下任何可选环境变量来调整默认的节流行为：

- throttle_initial_delay_ms - 在第一个拥塞事件之后应用的初始调用延迟。默认为 10 毫秒。
- throttle_max_delay_ms - 调用之间的最大延迟时间。您可以通过将其分成 1000 毫秒来推导 TPS。默认为 1000 毫秒。
- throttle_decrease_factor – Athena 降低调用速率的因子。默认值为 0.5。
- throttle_increase_ms – Athena 减少调用延迟的速率。默认为 10 毫秒。

数据库和表

Athena CloudWatch 指标连接器将您的命名空间、维度、指标和指标值映射到名为 `default` 的单个架构中的两个表中。

指标表

`metrics` 表包含由命名空间、集合和名称的组合唯一定义的可用指标。`metrics` 表包含以下列。

- 命名空间 - 包含命名空间的 `VARCHAR`。
- metric_name - 包含指标名称的 `VARCHAR`。

- 维度 - LISTSTRUCT 对象，由 dim_name （VARCHAR）和 dim_value （VARCHAR）组成。
- 统计数据 - 可用于该指标的 LISTVARCH 统计数据（例如，p90、AVERAGE...）。

metric_samples 表

metric_samples 表包含 metrics 表中每个指标的可用指标示例。metric_samples 表包含以下列。

- 命名空间 - 包含命名空间的 VARCHAR。
- metric_name - 包含指标名称的 VARCHAR。
- 维度 - STRUCT 对象的 LIST，由 dim_name （VARCHAR）和 dim_value （VARCHAR）组成。
- dim_name - 可用于轻松筛选单个维度名称的 VARCHAR 便捷字段。
- dim_value - 可用于轻松筛选单个维度值的 VARCHAR 便捷字段。
- 周期 - 表示指标“周期”的 INT 字段，以秒为单位（例如，60 秒指标）。
- 时间戳 - 表示指标样本纪元时间的 BIGINT 字段（以秒为单位）。
- 值 - 包含样本值的 FLOAT8 字段。
- 统计数据 - 包含样本统计类型的 VARCHAR（例如，AVERAGE 或 p90)。

所需权限

要获取有关此连接器所需 IAM policy 的完整详细信息，请查看 athena-cloudwatch-metrics.yaml 文件的 Policies 部分。以下列表汇总了所需的权限。

- Amazon S3 写入权限 – 连接器需要对 Amazon S3 中的位置具有写入权限，以溢出大型查询的结果。
- Athena GetQueryExecution – 当上游 Athena 查询终止时，该连接器将使用此权限快速失败。
- CloudWatch 指标只读 - 连接器使用此权限来查询您的指标数据。
- CloudWatch 日志写入 - 连接器使用此权限写入其诊断日志。

Performance

Athena CloudWatch 指标连接器尝试通过并行扫描查询所需的日志流来优化针对 CloudWatch 指标的查询。在特定时间段、指标、命名空间和维度筛选条件下，谓词下推既在 Lambda 函数内执行，也在 CloudWatch 日志中执行。

许可证信息

Amazon Athena CloudWatch 指标连接器项目已根据 Apache-2.0 许可证获得许可。

其他 资源

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena AWS CMDB 连接器

使用 Amazon Athena AWS CMDB 连接器使 Athena 可以与各种 AWS 服务通信，以便您可以使用 SQL 查询这些服务。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

参数

使用本节中的 Lambda 环境变量来配置 AWS CMDB 连接器。

- spill_bucket - 为超出 Lambda 函数限制的数据指定 Amazon S3 存储桶。
- spill_prefix -（可选）默认为指定 `spill_bucket`（称为 `athena-federation-spill`）中的子文件夹。我们建议您在此位置配置 Amazon S3 存储生命周期，以删除早于预定天数或小时数的溢出内容。
- spill_put_request_headers —（可选）用于溢出的 Amazon S3 `putObject` 请求的请求标头和值的 JSON 编码映射（例如 `{"x-amz-server-side-encryption" : "AES256"}`）。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。
- kms_key_id -（可选）默认情况下，将使用经过 AES-GCM 身份验证的加密模式和随机生成的密钥对溢出到 Amazon S3 的任何数据进行加密。要让您的 Lambda 函数使用 KMS 生成的更强的加密密钥（如 `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`），您可以指定 KMS 密钥 ID。
- disable_spill_encryption -（可选）当设置为 `True` 时，将禁用溢出加密。默认值为 `False`，此时将使用 AES-GCM 对溢出到 S3 的数据使用进行加密 - 使用随机生成的密钥，或者使用 KMS 生成密钥。禁用溢出加密可以提高性能，尤其是当您的溢出位置使用服务器端加密时。
- default_ec2_image_owner –（可选）设置后，控制筛选 Amazon 机器映像 (AMI) 的默认 Amazon EC2 映像所有者。如果您未设置此值，并且您对 EC2 映像表的查询不包含所有者筛选条件，则您的结果将包括所有公有映像。

数据库和表

Athena AWS CMDB 连接器使以下数据库和表可用于查询您的 AWS 资源清单。有关每个表中可用列的更多信息，请使用 Athena 控制台或 API 运行 `DESCRIBE database.table` 语句。

- ec2 - 该数据库包含 Amazon EC2 相关资源，包括以下内容。
- ebs_volumes - 包含您的 Amazon EBS 卷的详细信息。
- ec2_instances - 包含您的 EC2 实例的详细信息。
- ec2_images - 包含您的 EC2 实例映像的详细信息。
- routing_tables - 包含您的 VPC 路由表的详细信息。
- security_groups - 包含您的安全组的详细信息。
- 子网 - 包含您的 VPC 子网的详细信息。
- vpcs - 包含您的 VPC 的详细信息。
- emr - 该数据库包含 Amazon EMR 相关资源，包括以下内容。
- emr_clusters - 包含您的 EMR 集群的详细信息。
- rds - 该数据库包含 Amazon RDS 相关资源，包括以下内容。
- rds_instances - 包含您的 RDS 实例的详细信息。
- s3 - 该数据库包含 RDS 相关资源，包括以下内容。
- 桶 - 包含您的 Amazon S3 存储桶的详细信息。
- 对象 - 包含您的 Amazon S3 对象（不包括其内容）的详细信息。

所需权限

要获取有关此连接器所需 IAM policy 的完整详细信息，请查看 athena-aws-cmdb.yaml 文件的 `Policies` 部分。以下列表汇总了所需的权限。

- Amazon S3 写入权限 – 连接器需要对 Amazon S3 中的位置具有写入权限，以溢出大型查询的结果。
- Athena GetQueryExecution – 当上游 Athena 查询终止时，该连接器将使用此权限快速失败。
- S3 清单 - 连接器使用此权限列出您的 Amazon S3 存储桶和对象。
- EC2 描述 - 连接器使用此权限来描述资源，例如您的 Amazon EC2 实例、安全组、VPC 和 Amazon EBS 卷。
- EMR 描述/列表 - 连接器使用此权限来描述您的 EMR 集群。
- RDS 描述 - 连接器使用此权限来描述您的 RDS 实例。

Performance

目前，Athena AWS CMDB 连接器不支持并行扫描。谓词下推在 Lambda 函数中执行。在可能的情况下，部分谓词会被推送到正在查询的服务。例如，查询特定 Amazon EC2 实例的详细信息会调用具有特定实例 ID 的 EC2 API 来运行目标描述操作。

许可证信息

Amazon Athena AWS CMDB 连接器项目已获得 Apache-2.0 许可证授权。

其他 资源

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena IBM Db2 连接器

使用适用于 Db2 的 Amazon Athena 连接器，Amazon Athena 能够使用 JDBC 对 IBM Db2 数据库运行 SQL 查询。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。
- 使用此连接器前，请先设置 VPC 和安全组。有关更多信息，请参阅 为数据源连接器创建 VPC。

限制

- 不支持写入 DDL 操作。
- 在多路复用器设置中，溢出桶和前缀在所有数据库实例之间共享。

- 任何相关的 Lambda 限制。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。
- 必须将筛选条件中的日期和时间戳数据类型转换为适当的数据类型。

术语

以下术语与 Db2 连接器有关。

- 数据库实例 — 部署在本地、Amazon EC2 或 Amazon RDS 上的任何数据库实例。
- 处理程序 — 访问您数据库实例的 Lambda 处理程序。处理程序可以用于元数据或数据记录。
- 元数据处理程序 — 从您的数据库实例中检索元数据的 Lambda 处理程序。
- 记录处理程序 — 从您的数据库实例中检索数据记录的 Lambda 处理程序。
- 复合处理程序 — 从您的数据库实例中检索元数据和数据记录的 Lambda 处理程序。
- 属性或参数 — 处理程序用来提取数据库信息的数据库属性。您可以将这些属性配置为 Lambda 环境变量。
- 连接字符串 — 用于建立数据库实例连接的文本字符串。
- 目录 — 向 Athena 注册的非 AWS Glue 目录，是 `connection_string` 属性的必要前缀。
- 多路复用处理程序 — 可以接受和使用多个数据库连接的 Lambda 处理程序。

参数

使用本节中的 Lambda 环境变量来配置 Db2 连接器。

连接字符串

使用以下格式的 JDBC 连接字符串连接到数据库实例。

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

使用多路复用处理程序

您可以使用多路复用器通过单个 Lambda 函数连接到多个数据库实例。按目录名称来路由请求。在 Lambda 中使用以下类。

处理程序	类
复合处理程序	`Db2MuxCompositeHandler`

处理程序	类
元数据处理程序	Db2MuxMetadataHandler
记录处理程序	Db2MuxRecordHandler

多路复用处理程序参数

参数	描述
$*catalog*_connection_string	必需。数据库实例连接字符串。将 Athena 中使用的目录的名称作为环境变量前缀。例如，如果向 Athena 注册的目录是 mydbtwocatalog ，则环境变量名称是 mydbtwocatalog_connection_string 。
default	必需。默认连接字符串。目录为 lambda:${ *AWS_LAMBDA_FUNCTION_NAME* } 时使用此字符串。

以下示例属性适用于支持两个数据库实例的 Db2 MUX Lambda 函数：dbtwo1（默认）和 dbtwo2。

属性	值
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* }

提供凭证

要在 JDBC 连接字符串中为数据库提供用户名和密码，可以使用连接字符串属性或 AWS Secrets Manager。

- 连接字符串 — 可以将用户名和密码指定为 JDBC 连接字符串中的属性。

> **Important**
>
> 作为安全最佳实践，请勿在您的环境变量或连接字符串中使用硬编码凭证。有关将硬编码密钥移至 AWS Secrets Manager 的信息，请参阅《AWS Secrets Manager 用户指南》中的将硬编码密钥移至 AWS Secrets Manager。

- AWS Secrets Manager - 要将 Athena 联合查询功能与 AWS Secrets Manager 配合使用，连接到您的 Lambda 函数的 VPC 应该拥有互联网访问权限或者 VPC 端点，以连接到 Secrets Manager。

  您可以将 AWS Secrets Manager 中的密钥名称放入您的 JDBC 连接字符串中。连接器将该密钥名称替换为来自 Secrets Manager 的 `username` 和 `password` 值。

  对于 Amazon RDS 数据库实例，将紧密集成这种支持。如果您使用 Amazon RDS，我们强烈建议您使用 AWS Secrets Manager 和凭证轮换。如果您的数据库不使用 Amazon RDS，请按以下格式将凭证存储为 JSON：

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

带有密钥名称的示例连接字符串

以下字符串带有密钥名称 `${secret_name}`。

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

该连接器使用该密钥名称来检索密钥，并提供用户名和密码，如以下示例所示。

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

使用单个连接处理程序

您可以使用以下单个连接元数据和记录处理程序连接到单个 Db2 实例。

处理程序类型	类
复合处理程序	`Db2CompositeHandler`
元数据处理程序	`Db2MetadataHandler`

处理程序类型	类
记录处理程序	`Db2RecordHandler`

单个连接处理程序参数

参数	描述
`default`	必需。默认连接字符串。

单个连接处理程序支持一个数据库实例，并且必须提供 `default` 连接字符串参数。将忽略所有其他连接字符串。

以下示例属性适用于 Lambda 函数支持的单个 Db2 实例。

属性	值
`default`	dbtwo://jdbc:db2://hostname:port/*`database_name`* :${secret_name}

溢出参数

Lambda 开发工具包可以将数据溢出到 Amazon S3。由同一 Lambda 函数访问的所有数据库实例都会溢出到同一位置。

参数	描述
`spill_bucket`	必需。溢出桶名称。
`spill_prefix`	必需。溢出桶密钥前缀。
`spill_put_request_headers`	（可选）用于溢出的 Amazon S3 `putObject` 请求的请求标头和值的 JSON 编码映射（例如，`{"x-amz-server-side-encryption" : "AES256"}`)。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。

数据类型支持

下表显示了适用于 JDBC 和 Arrow 的相应数据类型。

Db2	Arrow
CHAR	VARCHAR
VARCHAR	VARCHAR
DATE	DATEDAY
TIME	VARCHAR
TIMESTAMP	DATEMILLI
DATETIME	DATEMILLI
BOOLEAN	BOOL
SMALLINT	SMALLINT
INTEGER	INT
BIGINT	BIGINT
DECIMAL	DECIMAL
REAL	FLOAT8
DOUBLE	FLOAT8
DECFLOAT	FLOAT8

分区和拆分

分区由一个或多个类型为 `varchar` 的分区列表示。Db2 连接器使用以下组织架构创建分区。

- 按哈希值分发
- 按范围分区
- 按维度组织

连接器从一个或多个 Db2 元数据表中检索分区详细信息，例如分区数和列名称。拆分根据指定的分区数而创建。

Performance

Athena Db2 连接器执行谓词下推，以减少查询扫描的数据量。LIMIT 子句、简单谓词和复杂表达式将下推到连接器，以减少扫描数据量并缩短查询执行的运行时间。

LIMIT 子句

LIMIT N 语句用于减少查询扫描的数据量。LIMIT N 下推时，连接器仅向 Athena 返回 N 行。

Predicates

谓词是 SQL 查询的 WHERE 子句中的表达式，其评估结果为布尔值并根据多个条件筛选行。Athena Db2 连接器可以组合这些表达式并将其直接推送到 Db2，以增强功能并减少扫描的数据量。

以下 Athena Db2 连接器运算符支持谓词下推：

- 布尔值：AND、OR、NOT
- 相等：EQUAL、NOT_EQUAL、LESS_THAN、LESS_THAN_OR_EQUAL、GREATER_THAN、GREATER_
- 算术：ADD、SUBTRACT、MULTIPLY、DIVIDE、MODULUS、NEGATE
- 其他：LIKE_PATTERN、IN

组合下推示例

要增强查询功能，组合下推类型，如以下示例所示：

```
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;
```

传递查询

Db2 连接器支持传递查询。传递查询使用表函数将完整查询下推到数据来源来执行查询。

要在 Db2 中执行传递查询，可以使用以下语法：

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

以下示例查询将查询下推到 Db2 中的数据来源。该查询选择了 customer 表中的所有列，将结果限制为 10。

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

许可证信息

使用此连接器，即表示您确认包含第三方组件（这些组件的列表可在此连接器的 pom.xml 文件中找到），并同意 GitHub.com 上的 LICENSE.txt 文件中提供的相应第三方许可证中的条款。

其他 资源

有关最新 JDBC 驱动程序版本的信息，请参阅 GitHub.com 上适用于 Db2 连接器的 pom.xml 文件。

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena IBM Db2 AS/400（Db2 iSeries）连接器

使用适用于 Db2 AS/400 的 Amazon Athena 连接器，Amazon Athena 能够使用 JDBC 在 IBM Db2 AS/400（Db2 iSeries）数据库上运行 SQL 查询。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。
- 使用此连接器前，请先设置 VPC 和安全组。有关更多信息，请参阅 为数据源连接器创建 VPC。

限制

- 不支持写入 DDL 操作。
- 在多路复用器设置中，溢出桶和前缀在所有数据库实例之间共享。

- 任何相关的 Lambda 限制。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。
- 必须将筛选条件中的日期和时间戳数据类型转换为适当的数据类型。

术语

以下术语与 Db2 AS/400 连接器有关。

- 数据库实例 — 部署在本地、Amazon EC2 或 Amazon RDS 上的任何数据库实例。
- 处理程序 — 访问您数据库实例的 Lambda 处理程序。处理程序可以用于元数据或数据记录。
- 元数据处理程序 — 从您的数据库实例中检索元数据的 Lambda 处理程序。
- 记录处理程序 — 从您的数据库实例中检索数据记录的 Lambda 处理程序。
- 复合处理程序 — 从您的数据库实例中检索元数据和数据记录的 Lambda 处理程序。
- 属性或参数 — 处理程序用来提取数据库信息的数据库属性。您可以将这些属性配置为 Lambda 环境变量。
- 连接字符串 — 用于建立数据库实例连接的文本字符串。
- 目录 — 向 Athena 注册的非 AWS Glue 目录，是 `connection_string` 属性的必要前缀。
- 多路复用处理程序 — 可以接受和使用多个数据库连接的 Lambda 处理程序。

参数

使用本节中的 Lambda 环境变量来配置 Db2 AS/400 连接器。

连接字符串

使用以下格式的 JDBC 连接字符串连接到数据库实例。

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

使用多路复用处理程序

您可以使用多路复用器通过单个 Lambda 函数连接到多个数据库实例。按目录名称来路由请求。在 Lambda 中使用以下类。

处理程序	类
复合处理程序	`Db2MuxCompositeHandler`

处理程序	类
元数据处理程序	Db2MuxMetadataHandler
记录处理程序	Db2MuxRecordHandler

多路复用处理程序参数

参数	描述
$*catalog*_connection_string	必需。数据库实例连接字符串。将 Athena 中使用的目录的名称作为环境变量前缀。例如，如果向 Athena 注册的目录是 mydb2as400catalog ，则环境变量名称是 mydb2as400catalog_connection_string 。
default	必需。默认连接字符串。目录为 lambda:${ *AWS_LAMBDA_FUNCTION_NAME* } 时使用此字符串。

以下示例属性适用于支持两个数据库实例的 Db2 MUX Lambda 函数：db2as4001（默认）和 db2as4002。

属性	值
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>* ;

提供凭证

要在 JDBC 连接字符串中为数据库提供用户名和密码，可以使用连接字符串属性或 AWS Secrets Manager。

- 连接字符串 — 可以将用户名和密码指定为 JDBC 连接字符串中的属性。

  > **Important**
  >
  > 作为安全最佳实践，请勿在您的环境变量或连接字符串中使用硬编码凭证。有关将硬编码密钥移至 AWS Secrets Manager 的信息，请参阅《AWS Secrets Manager 用户指南》中的将硬编码密钥移至 AWS Secrets Manager。

- AWS Secrets Manager - 要将 Athena 联合查询功能与 AWS Secrets Manager 配合使用，连接到您的 Lambda 函数的 VPC 应该拥有互联网访问权限或者 VPC 端点，以连接到 Secrets Manager。

  您可以将 AWS Secrets Manager 中的密钥名称放入您的 JDBC 连接字符串中。连接器将该密钥名称替换为来自 Secrets Manager 的 `username` 和 `password` 值。

  对于 Amazon RDS 数据库实例，将紧密集成这种支持。如果您使用 Amazon RDS，我们强烈建议您使用 AWS Secrets Manager 和凭证轮换。如果您的数据库不使用 Amazon RDS，请按以下格式将凭证存储为 JSON：

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

带有密钥名称的示例连接字符串

以下字符串带有密钥名称 `${secret_name}`。

```
db2as400://jdbc:as400://<ip_address>;<properties>;:${<secret_name>};
```

该连接器使用该密钥名称来检索密钥，并提供用户名和密码，如以下示例所示。

```
db2as400://jdbc:as400://<ip_address>;user=<username>;password=<password>;<properties>;
```

使用单个连接处理程序

您可以使用以下单个连接元数据和记录处理程序来连接到单个 Db2 AS/400实例。

处理程序类型	类
复合处理程序	Db2CompositeHandler
元数据处理程序	Db2MetadataHandler
记录处理程序	Db2RecordHandler

单个连接处理程序参数

参数	描述
default	必需。默认连接字符串。

单个连接处理程序支持一个数据库实例，并且必须提供 default 连接字符串参数。将忽略所有其他连接字符串。

以下示例属性适用于 Lambda 函数支持的单个 Db2 AS/400 实例。

属性	值
default	db2as400://jdbc:as400:// *<ip_address>* ;*<properties>* ;: ${*<secret_name>* };

溢出参数

Lambda 开发工具包可以将数据溢出到 Amazon S3。由同一 Lambda 函数访问的所有数据库实例都会溢出到同一位置。

参数	描述
spill_bucket	必需。溢出桶名称。
spill_prefix	必需。溢出桶密钥前缀。
spill_put_request_ headers	（可选）用于溢出的 Amazon S3 putObject 请求的请求标头和值的 JSON 编码映射（例如，{"x-amz-server-side-

参数	描述
	encryption" : "AES256"})。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject 。

数据类型支持

下表显示了适用于 JDBC 和 Apache Arrow 的相应数据类型。

Db2 AS/400	Arrow
CHAR	VARCHAR
VARCHAR	VARCHAR
DATE	DATEDAY
TIME	VARCHAR
TIMESTAMF	DATEMILLI
DATETIME	DATEMILLI
BOOLEAN	BOOL
SMALLINT	SMALLINT
INTEGER	INT
BIGINT	BIGINT
DECIMAL	DECIMAL
REAL	FLOAT8
DOUBLE	FLOAT8
DECFLOAT	FLOAT8

分区和拆分

分区由一个或多个类型为 `varchar` 的分区列表示。Db2 AS/400 连接器使用以下组织架构创建分区。

- 按哈希值分发
- 按范围分区
- 按维度组织

该连接器从一个或多个 Db2 AS/400 元数据表中检索分区详细信息，例如分区数和列名称。拆分根据指定的分区数而创建。

Performance

要提高性能，请使用谓词“下推”从 Athena 进行查询，如下例所示。

```
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'
```

传递查询

Db2 AS/400 连接器支持传递查询。传递查询使用表函数将完整查询下推到数据来源来执行查询。

要在 Db2 AS/400 中执行传递查询，可以使用以下语法：

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

以下示例查询将查询下推到 Db2 AS/400 中的数据来源。该查询选择了 `customer` 表中的所有列，将结果限制为 10。

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

```
        ))
```

许可证信息

使用此连接器，即表示您确认包含第三方组件（这些组件的列表可在此连接器的 pom.xml 文件中找到），并同意 GitHub.com 上的 LICENSE.txt 文件中提供的相应第三方许可证中的条款。

其他 资源

有关最新 JDBC 驱动程序版本的信息，请参阅 GitHub.com 上适用于 Db2 AS/400 连接器的 pom.xml 文件。

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena DocumentDB 连接器

Amazon Athena DocumentDB 连接器使 Athena 可以与您的 DocumentDB 实例通信，以便您可以使用 SQL 查询 DocumentDB 数据。该连接器还可与任何与 MongoDB 兼容的终端节点配合使用。

与传统的关系数据存储不同，Amazon DocumentDB 集合没有集架构。DocumentDB 没有元数据存储。DocumentDB 集合中的每个条目均可具有不同字段和数据类型。

DocumentDB 连接器支持两种生成表架构信息的机制：基本架构推理和 AWS Glue Data Catalog 元数据。

默认设置为架构推理。此选项将扫描您的集合中的少量文档，形成所有字段的并集，并强制使用非重叠数据类型的字段。此选项适用于条目大多为统一的集合。

对于具有更多数据类型的集合，该连接器支持从 AWS Glue Data Catalog 检索元数据。如果该连接器发现与您的 DocumentDB 数据库和集合名称相匹配的 AWS Glue 数据库和表，它将从相应的 AWS Glue 表中获取其架构信息。在您创建 AWS Glue 表时，我们建议您将其设置为您可能想从 DocumentDB 集合访问的所有字段的超集。

如果您在账户中启用了 Lake Formation，则您在 AWS Serverless Application Repository 中部署的 Athena 联合身份 Lambda 连接器的 IAM 角色必须在 Lake Formation 中具有 AWS Glue Data Catalog 的读取权限。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

参数

使用本节中的 Lambda 环境变量来配置 DocumentDB 连接器。

- spill_bucket - 为超出 Lambda 函数限制的数据指定 Amazon S3 存储桶。
- spill_prefix - （可选）默认为指定 `spill_bucket`（称为 `athena-federation-spill`）中的子文件夹。我们建议您在此位置配置 Amazon S3 存储生命周期，以删除早于预定天数或小时数的溢出内容。
- spill_put_request_headers —（可选）用于溢出的 Amazon S3 `putObject` 请求的请求标头和值的 JSON 编码映射（例如 `{"x-amz-server-side-encryption" : "AES256"}`）。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。
- kms_key_id - （可选）默认情况下，将使用经过 AES-GCM 身份验证的加密模式和随机生成的密钥对溢出到 Amazon S3 的任何数据进行加密。要让您的 Lambda 函数使用 KMS 生成的更强的加密密钥（如 `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`），您可以指定 KMS 密钥 ID。
- disable_spill_encryption - （可选）当设置为 `True` 时，将禁用溢出加密。默认值为 `False`，此时将使用 AES-GCM 对溢出到 S3 的数据使用进行加密 - 使用随机生成的密钥，或者使用 KMS 生成密钥。禁用溢出加密可以提高性能，尤其是当您的溢出位置使用服务器端加密时。
- disable_glue - （可选）如果存在且设置为 true，则该连接器不会尝试从 AWS Glue 检索补充元数据。
- glue_catalog – （可选）使用此选项指定跨账户 AWS Glue 目录。默认情况下，该连接器将尝试从其自己的 AWS Glue 账户中获取元数据。
- default_docdb – 如果存在，则指定在不存在特定于目录的环境变量时要使用的 DocumentDB 连接字符串。
- disable_projection_and_casing – （可选）禁用投影和大小写。在您需要查询使用区分大小写的列名称的 Amazon DocumentDB 表时使用。`disable_projection_and_casing` 参数使用以下值来指定大小写和列映射的行为：
  - false – 这是默认设置。投影已启用，连接器要求所有列名称为小写。
  - true – 禁用投影和区分大小写。在使用 `disable_projection_and_casing` 参数时，请记住以下几点：
    - 使用该参数可能会导致更高的带宽使用量。此外，如果 Lambda 函数与数据来源处于不同的 AWS 区域，则由于带宽使用量较高，会产生更高的标准 AWS 跨区域传输成本。有关跨区域传输成本的更多信息，请参阅 AWS 合作伙伴网络博客中的服务器和无服务器架构的 AWS 数据传输费用。
    - 由于传输的字节数越多，并且字节数越大，需要反序列化时间越长，因此整体延迟可能会增加。

- enable_case_insensitive_match – (可选) 如果是 `true`，则对 Amazon DocumentDB 中的架构和表名执行搜索，搜索不区分大小写。默认为 `false`。如果查询包含大写的架构名或表名，则使用该参数。

指定连接字符串

您可以提供一个或多个属性，用于定义与该连接器配合使用的 DocumentDB 实例的 DocumentDB 连接详细信息。为此，请设置一个与您要在 Athena 中使用的目录名称相对应的 Lambda 环境变量。例如，假设您要使用以下查询来查询 Athena 中的两个不同 DocumentDB 实例：

```
SELECT * FROM "docdb_instance_1".database.table
```

```
SELECT * FROM "docdb_instance_2".database.table
```

您必须向 Lambda 函数中添加以下两个环境变量，然后才能使用这两个 SQL 语句：`docdb_instance_1` 和 `docdb_instance_2`。每个环境变量的值均应为以下格式的 DocumentDB 连接字符串：

```
mongodb://:@:/?ssl=true&ssl_ca_certs=rds-combined-ca-bundle.pem&replicaSet=rs0
```

使用密钥

您可以选择将 AWS Secrets Manager 用于您的连接字符串详细信息的部分值或全部值。要将 Athena 联合查询功能与 Secrets Manager 配合使用，连接到您的 Lambda 函数的 VPC 应该拥有互联网访问权限或者 VPC 端点，以连接到 Secrets Manager。

如果您使用语法 `${my_secret}` 将来自 Secrets Manager 的密钥的名称放入连接字符串，该连接器会将 `${my_secret}` 替换为来自 Secrets Manager 的纯文本值。密钥应存储为具有值 *<username>*:*<password>* 的纯文本密钥。存储为 {username:*<username>*,password:*<password>*} 的密钥将无法正确传递到连接字符串。

密钥也可以完全用于整个连接字符串，并且可以在密钥中定义用户名和密码。

例如，假设您将 `docdb_instance_1` 的 Lambda 环境变量设置为以下值：

```
mongodb://${docdb_instance_1_creds}@myhostname.com:123/?ssl=true&ssl_ca_certs=rds-
combined-ca-bundle.pem&replicaSet=rs0
```

Athena 查询联合软件开发工具包 (SDK) 将自动尝试从 Secrets Manager 检索名为的密钥 `docdb_instance_1_creds` 的密钥，然后注入该值来替换 `${docdb_instance_1_creds}`。`${ }` 字符组合所包含的该连接字符串的任何部分将被解释为来自 Secrets Manager 的密钥。如果您指定了该连接器在 Secrets Manager 中无法找到的密钥名称，则该连接器不会替换该文本。

在 AWS Glue 中设置数据库和表

由于该连接器的内置架构推理功能扫描有限数量的文档，并且仅支持一部分数据类型，因此您可能需要改为将 AWS Glue 用于元数据。

要使 AWS Glue 表可与 Amazon DocumentDB 配合使用，您必须拥有 AWS Glue 数据库和表，适用于您要为其提供补充元数据的 DocumentDB 数据库和集合。

将 AWS Glue 表用于补充元数据

1. 当您在 AWS Glue 控制台中编辑表和数据库时，可以添加以下表属性。

   - docdb-metadata-flag – 此属性将向 DocumentDB 连接器指明该连接器可将该表用于补充元数据。只要表属性的列表中存在 `docdb-metadata-flag` 属性，您就可以为 `docdb-metadata-flag` 提供任何值。

2. （可选）添加 sourceTable 表属性。此属性用于定义 Amazon DocumentDB 中的源表名称。如果 AWS Glue 表命名规则阻止您创建与 Amazon DocumentDB 表同名的 AWS Glue 表，请使用此属性。例如，AWS Glue 表名称中不允许大写字母，但 Amazon DocumentDB 表名称中允许使用大写字母。

3. （可选）添加 columnMapping 表属性。此属性用于定义列名映射。如果 AWS Glue 列命名规则阻止您创建列名称与 Amazon DocumentDB 表相同的 AWS Glue 表，请使用此属性。这可能很有用，因为 Amazon DocumentDB 列名称中允许使用大写字母，但 AWS Glue 列名称中不允许使用。

   `columnMapping` 属性值应为采用 `col1=Col1,col2=Col2` 格式的一组映射。

   > **Note**
   >
   > 列映射仅适用于顶级列名称，不适用于嵌套字段。

   在添加 AWS Glue `columnMapping` 表属性后，您可以移除 `disable_projection_and_casing` Lambda 环境变量。

4. 请确保使用适合 AWS Glue 的数据类型，如本文档中所列。

数据类型支持

本节列出了 DocumentDB 连接器用于架构推断的数据类型，以及使用 AWS Glue 元数据时的数据类型。

架构推理数据类型

DocumentDB 连接器的架构推断功能会尝试将值推断为属于以下数据类型之一。该表显示了适用于 Amazon DocumentDB、Java 和 Apache Arrow 的相应数据类型。

Apache Arrow	Java 或 DocDB
VARCHAR	String
INT	整数
BIGINT	长整型
BIT	布尔值
FLOAT4	浮点型
FLOAT8	Double
TIMESTAMPSEC	Date
VARCHAR	ObjectId
LIST	列出
STRUCT	文档

AWS Glue 数据类型

如果将 AWS Glue 用于补充元数据，则可配置以下数据类型。该表显示了适用于 AWS Glue 和 Apache Arrow 的相应数据类型。

AWS Glue	Apache Arrow
int	INT
bigint	BIGINT
double	FLOAT8
float	FLOAT4
布尔值	BIT
binary	VARBINARY
字符串	VARCHAR
列出	LIST
Struct	STRUCT

所需权限

有关此连接器所需 IAM policy 的完整详细信息，请查看 athena-docdb.yaml 文件的 `Policies` 部分。以下列表汇总了所需的权限。

- Amazon S3 写入权限 – 连接器需要对 Amazon S3 中的位置具有写入权限，以溢出大型查询的结果。
- Athena GetQueryExecution – 当上游 Athena 查询终止时，该连接器将使用此权限快速失败。
- AWS Glue Data Catalog – DocumentDB 连接器需要针对 AWS Glue Data Catalog 的只读访问权限，以获取架构信息。
- CloudWatch Logs – 该连接器需要针对 CloudWatch Logs 的访问权限，以存储日志。
- AWS Secrets Manager 读取权限 - 如果您选择在 Secrets Manager 中存储 DocumentDB 端点详细信息，则必须授予该连接器针对这些秘密的访问权限。
- VPC 访问权限 - 该连接器需要能够连接和断开您的 VPC 接口，以便它能连接到 VPC 并与您的 DocumentDB 实例通信。

Performance

Athena Amazon DocumentDB 连接器当前不支持并行扫描，但会尝试下推谓词作为其 DocumentDB 查询的组成部分，并且针对 DocumentDB 集合索引的谓词将显著减少扫描的数据。

Lambda 函数执行投影下推，以减少查询扫描的数据。但是，选择列的子集有时会导致更长的查询执行运行时。`LIMIT` 子句可减少扫描的数据量，但如果未提供谓词，则预期要使用包含 `LIMIT` 子句的 `SELECT` 查询，来扫描至少 16 MB 的数据。

传递查询

Athena Amazon DocumentDB 连接器支持传递查询，并且以 NoSQL 为基础。有关查询 Amazon DocumentDB 的信息，请参阅《Amazon DocumentDB Developer Guide》中的 Querying。

要在 Amazon DocumentDB 中执行传递查询，请使用以下语法：

```
SELECT * FROM TABLE(
        system.query(
            database => 'database_name',
            collection => 'collection_name',
            filter => '{query_syntax}'
        ))
```

以下示例对 TPCDS 集合中的 `example` 数据库进行了查询，筛选出书名为《Bill of Rights》的所有书籍。

```
SELECT * FROM TABLE(
        system.query(
            database => 'example',
            collection => 'tpcds',
            filter => '{title: "Bill of Rights"}'
        ))
```

其他 资源

- 有关使用 Amazon Athena 联合查询将 MongoDB 数据库连接到 Amazon QuickSight 以构建控制面板和可视化的文章，请参阅 AWS 大数据博客中的使用 Amazon Athena 联合查询从 Amazon QuickSight 可视化 MongoDB 数据。
- 有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena DynamoDB 连接器

Amazon Athena DynamoDB 连接器使 Amazon Athena 可以与 DynamoDB 通信，以便您可以使用 SQL 查询表。写入操作不受支持，例如 INSERT INTO。

如果您在账户中启用了 Lake Formation，则您在 AWS Serverless Application Repository 中部署的 Athena 联合身份 Lambda 连接器的 IAM 角色必须在 Lake Formation 中具有 AWS Glue Data Catalog 的读取权限。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

参数

使用本节中的 Lambda 环境变量来配置 DynamoDB 连接器。

- spill_bucket - 为超出 Lambda 函数限制的数据指定 Amazon S3 存储桶。
- spill_prefix - （可选）默认为指定 `spill_bucket`（称为 `athena-federation-spill`）中的子文件夹。我们建议您在此位置配置 Amazon S3 存储生命周期，以删除早于预定天数或小时数的溢出内容。
- spill_put_request_headers —（可选）用于溢出的 Amazon S3 `putObject` 请求的请求标头和值的 JSON 编码映射（例如 `{"x-amz-server-side-encryption" : "AES256"}`）。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。
- kms_key_id - （可选）默认情况下，将使用经过 AES-GCM 身份验证的加密模式和随机生成的密钥对溢出到 Amazon S3 的任何数据进行加密。要让您的 Lambda 函数使用 KMS 生成的更强的加密密钥（如 `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`），您可以指定 KMS 密钥 ID。
- disable_spill_encryption - （可选）当设置为 `True` 时，将禁用溢出加密。默认值为 `False`，此时将使用 AES-GCM 对溢出到 S3 的数据使用进行加密 - 使用随机生成的密钥，或者使用 KMS 生成密钥。禁用溢出加密可以提高性能，尤其是当您的溢出位置使用服务器端加密时。
- disable_glue - （可选）如果存在且设置为 true，则该连接器不会尝试从 AWS Glue 检索补充元数据。
- glue_catalog – （可选）使用此选项指定跨账户 AWS Glue 目录。默认情况下，该连接器将尝试从其自己的 AWS Glue 账户中获取元数据。

- disable_projection_and_casing – （可选）禁用投影和大小写。如果您要查询列名中有大小写的 DynamoDB 表，并且不想在您的 AWS Glue 表上指定 `columnMapping` 属性，请使用此参数。

  `disable_projection_and_casing` 参数使用以下值来指定大小写和列映射的行为：

  - auto – 当检测到以前不支持的类型，并且未在表上设置列名称映射时，禁用投影和大小写。这是默认设置。
  - always – 无条件禁用投影和大小写。当您的 DynamoDB 列名称中有大小写，但您不想指定任何列名称映射时，此参数很有用。

  在使用 `disable_projection_and_casing` 参数时，请记住以下几点：

  - 使用该参数可能会导致更高的带宽使用量。此外，如果 Lambda 函数与数据来源处于不同的 AWS 区域，则由于带宽使用量较高，会产生更高的标准 AWS 跨区域传输成本。有关跨区域传输成本的更多信息，请参阅 AWS 合作伙伴网络博客中的服务器和无服务器架构的 AWS 数据传输费用。
  - 由于传输的字节数越多，并且字节数越大，需要反序列化时间越长，因此整体延迟可能会增加。

在 AWS Glue 中设置数据库和表

由于该连接器的内置架构推理功能有限，因此您可能需要将 AWS Glue 用于元数据。为此，您必须在 AWS Glue 中有一个数据库和表。要将其与 DynamoDB 一起使用，您必须编辑其属性。

在 AWS Glue 控制台中编辑表属性

1. 登录 AWS Management Console，然后打开 AWS Glue 控制台，网址为：https://console.aws.amazon.com/glue/。
2. 选择 Databases（数据库）选项卡。

   在 Databases（数据库）页面上，您可以编辑现有的数据库，也可以选择 Add database（添加数据库）来创建数据库。
3. 在数据库列表中，选择要编辑的数据库的链接。
4. 选择编辑。
5. 在 Update a database（更新数据库）页面上，对于 Location（位置），添加字符串 **`dynamo-db-flag`**。此关键字指示数据库包含 Athena DynamoDB 连接器用于补充元数据的表，并且是除 `default` 以外其他 AWS Glue 数据库所需要的。`dynamo-db-flag` 属性非常适合用于筛选出具有多个数据库的账户中的数据库。

要在 AWS Glue 控制台中编辑表属性

1. 登录 AWS Management Console，然后打开 AWS Glue 控制台，网址为：https://console.aws.amazon.com/glue/。
2. 选择 Tables (表) 选项卡。

   在表选项卡上编辑现有表。有关手动添加表或使用爬网程序添加表的信息，请参阅《AWS Glue 开发人员指南》中的在 AWS Glue 控制台上使用表。
3. 在表的列表中，选择要编辑的表的链接。
4. 依次选择 Actions（操作）、Edit table（编辑表）。
5. 在 Edit table（编辑表）页面的 Table properties（表属性）部分，根据需要添加以下表属性。如果您使用 AWS Glue DynamoDB 爬网程序，将自动设置这些属性。

   - dynamodb – 向 Athena DynamoDB 连接器指明可将该表用于补充元数据的字符串。在表属性中名为 classification（分类）的字段下输入字符串 dynamodb（完全匹配）。

     > **Note**
     >
     > 设置表属性页面是 AWS Glue 控制台中的表创建过程的一部分，内含带分类字段的数据格式部分。您无法在此处输入或选择 dynamodb。但在创建表之后，请按照步骤编辑表，并在表属性部分中以键值对的形式输入 `classification` 和 dynamodb。

   - sourceTable – 可选的表属性，用于定义 DynamoDB 中的源表名称。如果 AWS Glue 表命名规则阻止您创建与 DynamoDB 表同名的 AWS Glue 表，请使用此属性。例如，AWS Glue 表名称中不允许大写字母，但允 DynamoDB 表名称中允许大写字母。
   - columnMapping – 可选的表属性，用于定义列名称映射。如果 AWS Glue 列命名规则阻止您创建与 DynamoDB 表具有相同列名称的 AWS Glue 表，请使用此属性。例如，AWS Glue 列名称中不允许大写字母，但允 DynamoDB 列名称中允许大写字母。该属性值的格式应为 col1=Col1,col2=Col2。请注意，列映射仅适用于顶级列名，不适用于嵌套字段。
   - defaultTimeZone – 可选的表属性，应用于没有明确时区的 `date` 或 `datetime` 值。设置此值是一种很好的做法，可以避免数据来源默认时区与 Athena 会话时区之间存在差异。
   - datetimeFormatMapping – 可选的表属性，用于指定要在解析 AWS Glue `date` 或 `timestamp` 数据类型的列中的数据时使用的 `date` 或 `datetime` 格式。如果未指定此属性，则该连接器将尝试推断 ISO-8601 格式。如果该连接器无法推断 `date` 或 `datetime` 格式或无法解析原始字符串，则将在结果中省略该值。

datetimeFormatMapping 值应该采用 col1=someformat1,col2=someformat2 格式。下面是一些示例格式：

```
yyyyMMdd'T'HHmmss
ddMMyyyy'T'HH:mm:ss
```

如果您的列包含没有时区的 date 或 datetime 值，并且您想使用 WHERE 子句中的列，请为该列设置 datetimeFormatMapping 属性。

6. 如果您手动定义列，请确保使用适当的数据类型。如果您使用了爬网程序，请验证该爬网程序发现的列和类型。

所需权限

有关此连接器所需 IAM policy 的完整详细信息，请查看 athena-dynamodb.yaml 文件的 Policies 部分。以下列表汇总了所需的权限。

- Amazon S3 写入权限 – 连接器需要对 Amazon S3 中的位置具有写入权限，以溢出大型查询的结果。
- Athena GetQueryExecution – 当上游 Athena 查询终止时，该连接器将使用此权限快速失败。
- AWS Glue Data Catalog – DynamoDB 连接器需要针对 AWS Glue Data Catalog 的只读访问权限，以获取架构信息。
- CloudWatch Logs – 该连接器需要针对 CloudWatch Logs 的访问权限，以存储日志。
- DynamoDB 读取权限 – 该连接器使用 DescribeTable、ListSchemas、ListTables、Query 和 Scan API 操作。

Performance

Athena DynamoDB 连接器支持并行扫描，并尝试下推谓词作为其 DynamoDB 查询的组成部分。具有 X 个不同值的哈希键谓词将导致 X 个指向 DynamoDB 的查询调用。所有其他谓词场景都会导致 Y 次扫描调用，其中 Y 是根据您的表的大小及其预调配吞吐量以启发方式确定的。但是，选择一部分列有时会导致查询执行的运行时间延长。

LIMIT 子句和简单谓词将下推，这可以减少扫描的数据量，从而缩短查询执行的运行时间。

LIMIT 子句

LIMIT N 语句用于减少查询扫描的数据量。LIMIT N 下推时，连接器仅向 Athena 返回 N 行。

Predicates

谓词是 SQL 查询的 WHERE 子句中的表达式，其评估结果为布尔值并根据多个条件筛选行。为了增强功能并减少扫描的数据量，Athena DynamoDB 连接器可以组合这些表达式并将其直接推送到 DynamoDB。

以下 Athena DynamoDB 连接器运算符支持谓词下推：

- 布尔值：AND
- 相等：EQUAL、NOT_EQUAL、LESS_THAN、LESS_THAN_OR_EQUAL、GREATER_THAN、GREATER_

组合下推示例

要增强查询功能，组合下推类型，如以下示例所示：

```
SELECT *
FROM my_table
WHERE col_a > 10 and col_b < 10
LIMIT 10
```

有关使用谓词下推来提高联合查询（包括 DynamoDB）性能的文章，请参阅 AWS 大数据博客中的在 Amazon Athena 中使用谓词下推改善联合查询。

传递查询

DynamoDB 连接器支持传递查询，且使用 PartiQL 语法。不支持 DynamoDB GetItem API 操作。有关使用 PartiQL 查询 DynamoDB 的信息，请参阅《Amazon DynamoDB Developer Guide》中的 PartiQL select statements for DynamoDB。

要在 DynamoDB 中执行传递查询，请使用以下语法：

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

以下 DynamoDB 传递查询示例使用了 PartiQL 来返回 DateWatched 属性晚于 2022 年 12 月 24 日的 Fire TV Stick 设备列表。

```
SELECT * FROM TABLE(
        system.query(
           query => 'SELECT Devices
                       FROM WatchList
                       WHERE Devices.FireStick.DateWatched[0] > '12/24/22''
        ))
```

故障排除

排序键列上的多个筛选器

错误消息：keyConditionExpressions 每个键仅限包含一个条件

原因：在 Athena 引擎版本 3 中，如果查询在 DynamoDB 排序键列上同时具有下限和上限筛选器，可能会出现此问题。由于 DynamoDB 在排序键上不支持多个筛选条件，因此当连接器尝试下推同时应用这两个条件的查询时，会引发错误。

解决方案：将连接器更新为 2023.11.1 或更高版本。有关更新连接器的说明，请参阅 更新数据来源连接器。

成本

该连接器的使用成本取决于所使用的底层 AWS 资源。因为使用扫描的查询可能会消耗大量读取容量单位 (RCU)，请认真考虑有关 Amazon DynamoDB 定价的信息。

其他 资源

- 有关使用 Amazon Athena DynamoDB 连接器的介绍，请参阅 AWS 规范性指导模式指南中的使用 Athena 访问、查询和连接 Amazon DynamoDB 表。
- 有关将 Amazon Athena DynamoDB 连接器与 Amazon DynamoDB、Athena 和 Amazon QuickSight 一起用于创建简单监管控制面板的文章，请参阅 AWS 大数据博客中的使用 Amazon Athena 联合查询查询跨账户 Amazon DynamoDB 表。
- 有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena Google BigQuery 连接器

适用于 Google BigQuery 的 Amazon Athena 连接器使 Amazon Athena 能够对您的 Google BigQuery 数据运行 SQL 查询。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

限制

- Lambda 函数的最大超时值为 15 分钟。每次拆分都会在 BigQuery 上执行一次查询，并且必须有足够的时间来存储结果，以便 Athena 读取。如果 Lambda 函数超时，查询将失败。
- Google BigQuery 区分大小写。连接器尝试更正数据集名称和表名的大小写，但不对项目 ID 进行任何大小写更正。这很有必要，因为 Athena 所有元数据均采用小写形式。这些更正对 Google BigQuery 进行了许多额外的调用。
- 不支持 Binary 数据类型。
- 由于 Google BigQuery 的并发和配额限制，连接器可能会遇到 Google 配额限制问题。为了避免这些问题，请尽可能多地向 Google BigQuery 施加限制。有关 BigQuery 配额的信息，请参阅 Google BigQuery 文档中的 配额和限制。

参数

使用本节中的 Lambda 环境变量来配置 Google BigQuery 连接器。

- spill_bucket - 为超出 Lambda 函数限制的数据指定 Amazon S3 存储桶。
- spill_prefix - （可选）默认为指定 `spill_bucket`（称为 `athena-federation-spill`）中的子文件夹。我们建议您在此位置配置 Amazon S3 存储生命周期，以删除早于预定天数或小时数的溢出内容。
- spill_put_request_headers —（可选）用于溢出的 Amazon S3 `putObject` 请求的请求标头和值的 JSON 编码映射（例如 `{"x-amz-server-side-encryption" : "AES256"}`）。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。
- kms_key_id - （可选）默认情况下，将使用经过 AES-GCM 身份验证的加密模式和随机生成的密钥对溢出到 Amazon S3 的任何数据进行加密。要让您的 Lambda 函数使用 KMS 生成的更强的加密密钥（如 `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`），您可以指定 KMS 密钥 ID。
- disable_spill_encryption - （可选）当设置为 `True` 时，将禁用溢出加密。默认值为 `False`，此时将使用 AES-GCM 对溢出到 S3 的数据使用进行加密 - 使用随机生成的密钥，或者使用 KMS 生成密钥。禁用溢出加密可以提高性能，尤其是当您的溢出位置使用服务器端加密时。

- gcp_project_id - 项目 ID（不是项目名称），包含连接器应从中读取的数据集（例如 `semiotic-primer-1234567`)。
- secret_manager_gcp_creds_name - AWS Secrets Manager 中的密钥名称，包含 JSON 格式的 BigQuery 凭证（例如 `GoogleCloudPlatformCredentials`)。
- big_query_endpoint – （可选）BigQuery 私有端点的 URL。如果您想通过私有端点访问 BigQuery，使用此参数。

拆分与视图

由于 BigQuery 连接器使用 BigQuery 存储读取 API 来查询表，而 BigQuery 存储 API 不支持视图，所以该连接器使用具有单个视图拆分功能的 BigQuery 客户端。

Performance

为了查询表，BigQuery 连接器使用了 BigQuery 存储读取 API，该 API 使用基于 RPC 的协议，支持快速访问 BigQuery 托管存储。有关 BigQuery 存储读取 API 的更多信息，请参阅 Google Cloud 文档中的使用 BigQuery 存储读取 API 读取表格数据。

选择列的子集可以显著减少查询运行时及扫描的数据。随着并发增加，连接器容易出现查询失败，并且通常是慢速连接器。

Athena Google BigQuery 连接器执行谓词下推，以减少查询扫描的数据。`LIMIT` 子句、`ORDER BY` 子句、简单谓词和复杂表达式将下推到连接器，以减少扫描数据量并缩短查询执行的运行时间。

LIMIT 子句

`LIMIT N` 语句用于减少查询扫描的数据量。`LIMIT N` 下推时，连接器仅向 Athena 返回 `N` 行。

前 N 个查询

前 `N` 个查询用于指定结果集的顺序以及返回的行数限值。您可以使用此类查询确定数据集的前 `N` 个最大值或前 `N` 个最小值。前 `N` 个查询下推时，连接器仅向 Athena 返回 `N` 个已排序行。

Predicates

谓词是 SQL 查询的 `WHERE` 子句中的表达式，其评估结果为布尔值并根据多个条件筛选行。Athena Google BigQuery 连接器可以组合这些表达式并将其直接推送到 Google BigQuery，以增强功能并减少扫描数据量。

以下 Athena Google BigQuery 连接器运算符支持谓词下推：

- 布尔值：AND、OR、NOT
- 相
  等：EQUAL、NOT_EQUAL、LESS_THAN、LESS_THAN_OR_EQUAL、GREATER_THAN、GREATER_
- 算术：ADD、SUBTRACT、MULTIPLY、DIVIDE、MODULUS、NEGATE
- 其他：LIKE_PATTERN、IN

组合下推示例

要增强查询功能，组合下推类型，如以下示例所示：

```
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;
```

传递查询

Google BigQuery 连接器支持传递查询。传递查询使用表函数将完整查询下推到数据来源来执行查询。

要在 Google BigQuery 中执行传递查询，可以使用以下语法：

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

以下示例查询将查询下推到 Google BigQuery 中的数据来源。该查询选择了 customer 表中的所有列，将结果限制为 10。

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

许可证信息

Amazon Athena Google BigQuery 连接器项目已获得 Apache-2.0 许可证授权。

使用此连接器，即表示您确认包含第三方组件（这些组件的列表可在此连接器的 pom.xml 文件中找到），并同意 GitHub.com 上的 LICENSE.txt 文件中提供的相应第三方许可证中的条款。

其他 资源

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena Google Cloud Storage 连接器

借助 Amazon Athena Google Cloud Storage 连接器，Amazon Athena 可以对存储在 Google Cloud Storage（GCS）存储桶中的 Parquet 和 CSV 文件运行查询。将一个或多个 Parquet 或 CSV 文件分组到 GCS 存储桶中的未分区或已分区文件夹中后，即可以在 AWS Glue 数据库表中进行组织。

如果您在账户中启用了 Lake Formation，则您在 AWS Serverless Application Repository 中部署的 Athena 联合身份 Lambda 连接器的 IAM 角色必须在 Lake Formation 中具有 AWS Glue Data Catalog 的读取权限。

先决条件

- 根据您在 Google Cloud Storage 中的存储桶和文件夹设置对应的 AWS Glue 数据库和表。有关详细步骤，请参阅本文档后面的 在 AWS Glue 中设置数据库和表 部分。
- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

限制

- 不支持写入 DDL 操作。
- 任何相关的 Lambda 限制。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。
- 目前，此连接器仅支持分区列的 VARCHAR 类型（AWS Glue 表 Schema 中的 `string` 或 `varchar`）。当您在 Athena 中查询其他分区字段类型时会引发错误。

术语

以下术语与 GCS 连接器有关。

- 处理程序 – 访问您的 GCS 存储桶的 Lambda 处理程序。处理程序可以用于元数据或数据记录。
- 元数据处理程序 – 从您的 GCS 存储桶中检索元数据的 Lambda 处理程序。

- 记录处理程序 – 从您的 GCS 存储桶中检索数据记录的 Lambda 处理程序。
- 复合处理程序 – 从您的 GCS 存储桶中检索元数据和数据记录的 Lambda 处理程序。

支持的文件类型

GCS 连接器支持 Parquet 和 CSV 文件类型。

> **Note**
>
> 注意不要将 CSV 和 Parquet 文件放在同一 GCS 存储桶或路径中。这样做可能会导致尝试以 CSV 格式读取 Parquet 文件，或相反操作，从而出现运行时错误。

参数

使用这一部分中的 Lambda 环境变量来配置 GCS 连接器。

- spill_bucket - 为超出 Lambda 函数限制的数据指定 Amazon S3 存储桶。
- spill_prefix - （可选）默认为指定 `spill_bucket`（称为 `athena-federation-spill`）中的子文件夹。我们建议您在此位置配置 Amazon S3 存储生命周期，以删除早于预定天数或小时数的溢出内容。
- spill_put_request_headers —（可选）用于溢出的 Amazon S3 `putObject` 请求的请求标头和值的 JSON 编码映射（例如 `{"x-amz-server-side-encryption" : "AES256"}`）。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。
- kms_key_id - （可选）默认情况下，将使用经过 AES-GCM 身份验证的加密模式和随机生成的密钥对溢出到 Amazon S3 的任何数据进行加密。要让您的 Lambda 函数使用 KMS 生成的更强的加密密钥（如 `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`），您可以指定 KMS 密钥 ID。
- disable_spill_encryption - （可选）当设置为 `True` 时，将禁用溢出加密。默认值为 `False`，此时将使用 AES-GCM 对溢出到 S3 的数据使用进行加密 - 使用随机生成的密钥，或者使用 KMS 生成密钥。禁用溢出加密可以提高性能，尤其是当您的溢出位置使用服务器端加密时。
- secret_manager_gcp_creds_name - AWS Secrets Manager 中的密钥名称，包含 JSON 格式的 GCS 凭证（例如 `GoogleCloudPlatformCredentials`）。

在 AWS Glue 中设置数据库和表

由于 GCS 连接器的内置 Schema 推理能力有限，因此我们建议您使用 AWS Glue 来处理元数据。下面的过程演示了如何在 AWS Glue 中创建您可以从 Athena 访问的数据库和表。

在 AWS Glue 中创建数据库

您可以通过 AWS Glue 控制台创建可以与 GCS 连接器一起使用的数据库。

在 AWS Glue 创建数据库

1. 登录 AWS Management Console，然后打开 AWS Glue 控制台，网址为：https://console.aws.amazon.com/glue/。
2. 从导航窗格中选择 Databases（数据库）。
3. 选择 Add database（添加数据库）。
4. 对于 Name（名称），输入您要用于 GCS 连接器的数据库名称。
5. 对于 Location（位置），指定 `s3://google-cloud-storage-flag.`。此位置将告诉 GCS 连接器 AWS Glue 数据库包含要在 Athena 中查询的 GCS 数据的表。此连接器会识别 Athena 中具有此标志的数据库，并且会忽略没有此标志的数据库。
6. 选择创建数据库。

在 AWS Glue 中创建表

现在您可以为该数据库创建一个表。创建用于 GCS 连接器的 AWS Glue 表时，您必须指定额外的元数据。

在 AWS Glue 控制台中创建表

1. 在 AWS Glue 控制台中，从导航窗格中选择 Tables（表）。
2. 在 Tables（表）页面上，选择 Add table（添加表）。
3. 在 Set table properties（设置表属性）页面上，输入以下信息。

   - Name（名称）– 输入表的唯一名称。
   - Database（数据库）– 选择您为 GCS 连接器创建的 AWS Glue 数据库。
   - Include path（包含路径）– 对于 Data store（数据存储）部分中的 Include path（包含路径），输入前缀为 `gs://` 的 GCS URI 位置（例如 `gs://gcs_table/data/`）。如果您有一个或多个分区文件夹，请不要将其包含在路径中。

     > **Note**
     >
     > 当您输入非 `s3://` 表路径时，AWS Glue 控制台会显示错误。您可以忽略该错误。该表将会成功创建。

- Data format（数据格式）–对于 Classification（分类），选择 CSV 或 Parquet。

4. 选择下一步。
5. 在 Choose or define schema（选择或定义 Schema）页面上，强烈建议定义表 Schema，但这不是强制性的。如果您未定义 Schema，GCS 连接器会尝试为您推断 Schema。

   请执行以下操作之一：

   - 如果您希望 GCS 连接器尝试为您推断 Schema，请选择 Next（下一步），然后选择 Create（创建）。
   - 要自己定义 Schema，请执行下一部分中的步骤。

在 AWS Glue 中定义表 Schema

在 AWS Glue 中定义表 Schema 需要更多步骤，但可以更好地控制表创建过程。

在 AWS Glue 中定义表 Schema

1. 在 Choose or define schema（选择或定义 Schema）页面上，选择 Add（添加）。
2. 在 Add schema entry（添加 Schema 条目）对话框中提供列名称和数据类型。
3. 要将该列指定为分区列，请选中 Set as partition key（设为分区键）选项。
4. 选择 Save（保存）以保存更改。
5. 选择 Add（添加）以添加其他列。
6. 添加完列后，选择 Next（下一步）。
7. 在 Review and create（检查并创建）页面上检查表，然后选择 Create（创建）。
8. 如果您的 Schema 包含分区信息，请按照下一部分中的步骤，在 AWS Glue 中为表的属性添加分区模式。

在 AWS Glue 中将分区模式添加到表属性

如果您的 GCS 存储桶有分区，则必须在 AWS Glue 中将分区模式添加到表属性中。

在 AWS Glue 中将分区信息添加到表属性

1. 在您在 AWS Glue 中创建的表格的详细信息页面上，选择 Actions（操作），然后选择 Edit table（编辑表）。
2. 在 Edit table（编辑表）页面上，向下滚动到 Table properties（表属性）部分。

3. 选择 Add（添加）以添加分区键。
4. 对于键，输入 **partition.pattern**。此键将会定义文件夹路径模式。
5. 对于 Value（值），输入文件夹路径模式（例如 **StateName=${statename}/ZipCode=${zipcode}/**），其中用 **${}** 括起来的 **statename** 和 **zipcode** 是分区列名称。GCS 连接器支持 Hive 和非 Hive 分区 Schema。
6. 完成后，选择 Save。
7. 要查看刚才创建的表属性，请选择 Advanced properties（高级属性）选项卡。

然后，您可以导航到 Athena 控制台。您可以在 Athena 中查询您在 AWS Glue 中创建的数据库和表。

数据类型支持

下表显示了支持的 CSV 和 Parquet 数据类型。

CSV

数据性质	推断的数据类型
数据看起来像一个数字	BIGINT
数据看起来像一个字符串	VARCHAR
数据看起来像一个浮点数（浮点、双精度或十进制）	DOUBLE
数据看起来像一个日期	Timestamp
数据包含 true/false 值	BOOL

Parquet

PARQUET	Athena（箭头）
BINARY	VARCHAR
BOOLEAN	BOOL

PARQUET	Athena（箭头）
DOUBLE	DOUBLE
ENUM	VARCHAR
FIXED_LEN _BYTE_ARRAY	DECIMAL
FLOAT	FLOAT（32 位）
INT32	1. INT32 2. DATEDAY（当 Parquet 列的逻辑类型为 DATE 时）
INT64	1. INT64 2. TIMESTAMP（当 Parquet 列的逻辑类型为 TIMESTAMP 时）
INT96	Timestamp
MAP	MAP
STRUCT	STRUCT
LIST	LIST

所需权限

有关此连接器所需 IAM policy 的完整详细信息，请查看 athena-gcs.yaml 文件的 `Policies` 部分。以下列表汇总了所需的权限。

- Amazon S3 写入权限 – 连接器需要对 Amazon S3 中的位置具有写入权限，以溢出大型查询的结果。
- Athena GetQueryExecution – 当上游 Athena 查询终止时，该连接器将使用此权限快速失败。
- AWS Glue Data Catalog – GCS 连接器需要 AWS Glue Data Catalog 的只读访问权限，以获取 Schema 信息。
- CloudWatch Logs – 该连接器需要针对 CloudWatch Logs 的访问权限，以存储日志。

Performance

当表 Schema 包含分区字段并且 `partition.pattern` 表属性配置正确时，可以在查询的 `WHERE` 子句中包含该分区字段。对于此类查询，GCS 连接器使用分区列来优化 GCS 文件夹路径，避免扫描 GCS 文件夹中不需要的文件。

对于 Parquet 数据集，选择列的子集可减少要扫描的数据量。这通常可在应用列投影时缩短查询执行时间。

CSV 数据集不支持列投影，并且不会减少扫描的数据量。

`LIMIT` 子句可减少扫描的数据量，但如果未提供谓词，则预期要使用包含 `LIMIT` 子句的 `SELECT` 查询，扫描至少 16MB 的数据。无论是否应用 `LIMIT` 子句，GCS 连接器为较大数据集扫描的数据量都高于较小的数据集。例如，查询 `SELECT * LIMIT 10000` 为较大基础数据集扫描的数据高于较小的基础数据集。

许可证信息

使用此连接器，即表示您确认包含第三方组件（这些组件的列表可在此连接器的 pom.xml 文件中找到），并同意 GitHub.com 上的 LICENSE.txt 文件中提供的相应第三方许可证中的条款。

其他 资源

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena HBase 连接器

Amazon Athena HBase 连接器使 Amazon Athena 可以与 Apache HBase 实例通信，以便您可以使用 SQL 查询 HBase 数据。

与传统的关系数据存储不同，HBase 集合没有集架构。HBase 没有元数据存储。HBase 集合中的每个条目均可具有不同字段和数据类型。

HBase 连接器支持两种生成表架构信息的机制：基本架构推理和 AWS Glue Data Catalog 元数据。

默认设置为架构推理。此选项将扫描您的集合中的少量文档，形成所有字段的并集，并强制使用非重叠数据类型的字段。此选项适用于条目大多为统一的集合。

对于具有更多数据类型的集合，该连接器支持从 AWS Glue Data Catalog 检索元数据。如果该连接器发现与您的 HBase 命名空间和集合名称相匹配的 AWS Glue 数据库和表，它将从相应的 AWS Glue 表

中获取其架构信息。在您创建 AWS Glue 表时，我们建议您将其设置为您可能想从 HBase 集合访问的所有字段的超集。

如果您在账户中启用了 Lake Formation，则您在 AWS Serverless Application Repository 中部署的 Athena 联合身份 Lambda 连接器的 IAM 角色必须在 Lake Formation 中具有 AWS Glue Data Catalog 的读取权限。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

参数

使用本节中的 Lambda 环境变量来配置 HBase 连接器。

- spill_bucket - 为超出 Lambda 函数限制的数据指定 Amazon S3 存储桶。
- spill_prefix - （可选）默认为指定 `spill_bucket`（称为 `athena-federation-spill`）中的子文件夹。我们建议您在此位置配置 Amazon S3 存储生命周期，以删除早于预定天数或小时数的溢出内容。
- spill_put_request_headers — （可选）用于溢出的 Amazon S3 `putObject` 请求的请求标头和值的 JSON 编码映射（例如 `{"x-amz-server-side-encryption" : "AES256"}`）。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。
- kms_key_id - （可选）默认情况下，将使用经过 AES-GCM 身份验证的加密模式和随机生成的密钥对溢出到 Amazon S3 的任何数据进行加密。要让您的 Lambda 函数使用 KMS 生成的更强的加密密钥（如 `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`），您可以指定 KMS 密钥 ID。
- disable_spill_encryption - （可选）当设置为 `True` 时，将禁用溢出加密。默认值为 `False`，此时将使用 AES-GCM 对溢出到 S3 的数据使用进行加密 - 使用随机生成的密钥，或者使用 KMS 生成密钥。禁用溢出加密可以提高性能，尤其是当您的溢出位置使用服务器端加密时。
- disable_glue - （可选）如果存在且设置为 true，则该连接器不会尝试从 AWS Glue 检索补充元数据。
- glue_catalog – （可选）使用此选项指定跨账户 AWS Glue 目录。默认情况下，该连接器将尝试从其自己的 AWS Glue 账户中获取元数据。
- default_hbase – 如果存在，则指定在不存在特定于目录的环境变量时要使用的 HBase 连接字符串。

指定连接字符串

您可以提供一个或多个属性，用于定义与该连接器配合使用的 HBase 实例的 HBase 连接详细信息。为此，请设置一个与您要在 Athena 中使用的目录名称相对应的 Lambda 环境变量。例如，假设您要使用以下查询来查询 Athena 中的两个不同 HBase 实例：

```
SELECT * FROM "hbase_instance_1".database.table
```

```
SELECT * FROM "hbase_instance_2".database.table
```

您必须向 Lambda 函数中添加以下两个环境变量，然后才能使用这两个 SQL 语句：`hbase_instance_1` 和 `hbase_instance_2`。每个环境变量的值均应为以下格式的 HBase 连接字符串：

```
master_hostname:hbase_port:zookeeper_port
```

使用密钥

您可以选择将 AWS Secrets Manager 用于您的连接字符串详细信息的部分值或全部值。要将 Athena 联合查询功能与 Secrets Manager 配合使用，连接到您的 Lambda 函数的 VPC 应该拥有互联网访问权限或者 VPC 端点，以连接到 Secrets Manager。

如果您使用语法 `${my_secret}` 将来自 Secrets Manager 的密钥的名称放入您的连接字符串中，该连接器会将该密钥名称替换为来自 Secrets Manager 的您的用户名和密码值。

例如，假设您将 `hbase_instance_1` 的 Lambda 环境变量设置为以下值：

```
${hbase_host_1}:${hbase_master_port_1}:${hbase_zookeeper_port_1}
```

Athena 查询联合软件开发工具包 (SDK) 将自动尝试从 Secrets Manager 检索名为的密钥 `hbase_instance_1_creds` 的密钥，然后注入该值来替换 `${hbase_instance_1_creds}`。`${ }` 字符组合所包含的该连接字符串的任何部分将被解释为来自 Secrets Manager 的密钥。如果您指定了该连接器在 Secrets Manager 中无法找到的密钥名称，则该连接器不会替换该文本。

在 AWS Glue 中设置数据库和表

该连接器的内置架构推理仅支持在 HBase 中序列化为字符串的值（例如，`String.valueOf(int)`）。由于该连接器的内置架构推理功能有限，因此您可能需要改为将

AWS Glue 用于元数据。要使 AWS Glue 表可与 HBase 配合使用，您必须拥有名称与 HBase 命名空间相匹配的 AWS Glue 数据库和表，以及您要为其提供补充元数据的表。使用 HBase 列系列命名约定是可选的，但不是必需的。

将 AWS Glue 表用于补充元数据

1. 当您在 AWS Glue 控制台中编辑表和数据库时，可以添加以下表属性：

   - hbase-metadata-flag – 此属性将向 HBase 连接器指明该连接器可将该表用于补充元数据。只要表属性的列表中存在 `hbase-metadata-flag` 属性，您就可以为 `hbase-metadata-flag` 提供任何值。
   - hbase-native-storage-flag – 可以使用此标记切换该连接器支持的两种值序列化模式。默认情况下，当此字段不存在时，该连接器将假定所有值都以字符串形式存储在 HBase 中。因此，它将尝试解析来自 HBase 的字符串形式的数据类型（如 `INT`、`BIGINT` 和 `DOUBLE`）。如果使用 AWS Glue 中的表上的任何值设置此字段，该连接器将切换到“本机”存储模式，并尝试使用以下函数以字节形式读取 `INT`、`BIGINT`、`BIT` 和 `DOUBLE`：

     ```
     ByteBuffer.wrap(value).getInt()
     ByteBuffer.wrap(value).getLong()
     ByteBuffer.wrap(value).get()
     ByteBuffer.wrap(value).getDouble()
     ```

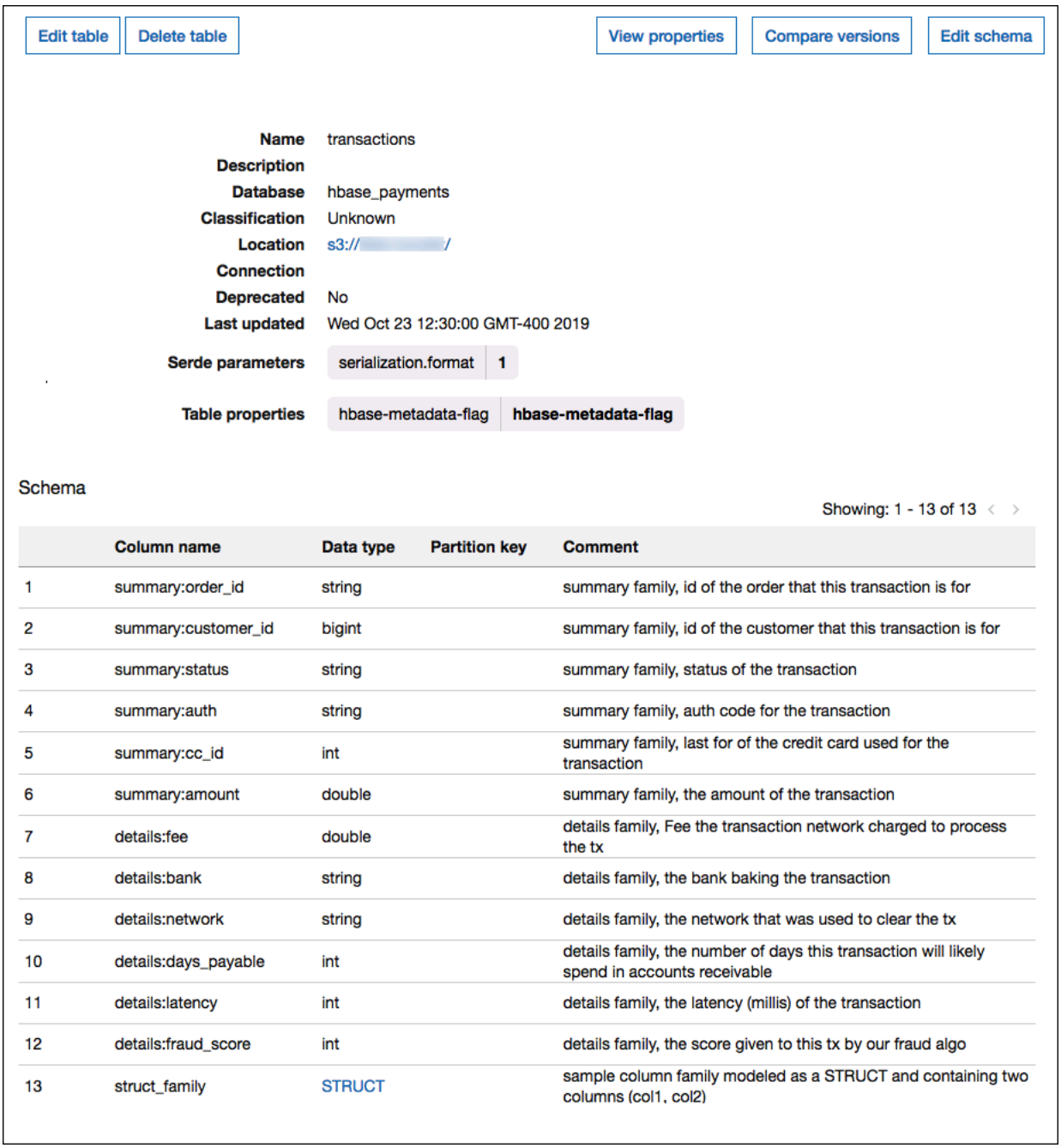
2. 请确保使用适合 AWS Glue 的数据类型，如本文档中所列。

为列系列建模

Athena HBase 连接器支持两种为 HBase 列系列建模的方法：完全限定（扁平化）命名（如 `family:column`），或使用 `STRUCT` 对象。

在 `STRUCT` 模型中，`STRUCT` 字段的名称应与列系列相匹配，并且 `STRUCT` 的子级应与该系列的列名称相匹配。但是，由于复杂类型（如 `STRUCT`）尚不完全支持谓词下推和列式读取，因此目前不建议使用 `STRUCT`。

下图显示了使用两种方法的组合在 AWS Glue 中配置的表。



数据类型支持

该连接器将以基本字节类型的形式检索所有 HBase 值。然后，根据您在 AWS Glue Data Catalog 中定义表的方式，它会将值映射到下表中的一种 Apache Arrow 数据类型。

AWS Glue 数据类型	Apache Arrow 数据类型
int	INT

AWS Glue 数据类型	Apache Arrow 数据类型
bigint	BIGINT
double	FLOAT8
float	FLOAT4
布尔值	BIT
binary	VARBINARY
字符串	VARCHAR

> **Note**
>
> 如果您不使用 AWS Glue 补充您的元数据，则该连接器的架构推理将仅使用数据类型 `BIGINT`、`FLOAT8` 和 `VARCHAR`。

所需权限

有关此连接器所需 IAM policy 的完整详细信息，请查看 athena-hbase.yaml 文件的 `Policies` 部分。以下列表汇总了所需的权限。

- Amazon S3 写入权限 – 连接器需要对 Amazon S3 中的位置具有写入权限，以溢出大型查询的结果。
- Athena GetQueryExecution – 当上游 Athena 查询终止时，该连接器将使用此权限快速失败。
- AWS Glue Data Catalog – HBase 连接器需要针对 AWS Glue Data Catalog 的只读访问权限，以获取架构信息。
- CloudWatch Logs – 该连接器需要针对 CloudWatch Logs 的访问权限，以存储日志。
- AWS Secrets Manager 读取权限 - 如果您选择在 Secrets Manager 中存储 HBase 端点详细信息，则必须授予该连接器针对这些秘密的访问权限。
- VPC 访问权限 - 该连接器需要能够连接和断开您的 VPC 接口，以便它能连接到 VPC 并与您的 HBase 实例通信。

Performance

Athena HBase 连接器将尝试通过并行读取每个区域服务器，来并行处理针对您的 HBase 实例的查询。Athena HBase 连接器可执行谓词下推，以减少查询扫描的数据量。

Lambda 函数还执行投影下推，以减少查询扫描的数据。但是，选择列的子集有时会导致更长的查询执行运行时。`LIMIT` 子句可减少扫描的数据量，但如果未提供谓词，则预期要使用包含 `LIMIT` 子句的 `SELECT` 查询，来扫描至少 16 MB 的数据。

HBase 容易出现查询失败和不同的查询执行时间。您可能需要多次重试查询才能成功。HBase 连接器能够灵活地应对并发造成的节流。

传递查询

HBase 连接器支持传递查询，并且以 NoSQL 为基础。有关查询 Apache HBase 的信息，请参阅 Apache 文档中的 Querying HBase。

要在 HBase 中执行传递查询，请使用以下语法：

```
SELECT * FROM TABLE(
        system.query(
            database => 'database_name',
            collection => 'collection_name',
            filter => '{query_syntax}'
        ))
```

以下 HBase 传递查询示例对 `default` 数据库 `employee` 集合中年龄为 24 或 30 岁的员工进行了筛选。

```
SELECT * FROM TABLE(
        system.query(
            DATABASE => 'default',
            COLLECTION => 'employee',
            FILTER => 'SingleColumnValueFilter(''personaldata'', ''age'', =,
 ''binary:30'')' ||
                      ' OR SingleColumnValueFilter(''personaldata'', ''age'', =,
 ''binary:24'')'
        ))
```

许可证信息

Amazon Athena HBase 连接器项目已经根据 Apache-2.0 许可证获得许可。

其他 资源

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena HortonWorks 连接器

适用于 Hortonworks 的 Amazon Athena 连接器使 Amazon Athena 能够在 Cloudera Hortonworks 数据平台上运行 SQL 查询。该连接器会将您的 Athena SQL 查询转换为其等效的 HiveQL 语法。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

限制

- 不支持写入 DDL 操作。
- 在多路复用器设置中，溢出桶和前缀在所有数据库实例之间共享。
- 任何相关的 Lambda 限制。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。

术语

以下术语与 Hortonworks Hive 连接器有关。

- 数据库实例 - 部署在本地、Amazon EC2 或 Amazon RDS 上的任何数据库实例。
- 处理程序 — 访问您数据库实例的 Lambda 处理程序。处理程序可以用于元数据或数据记录。
- 元数据处理程序 — 从您的数据库实例中检索元数据的 Lambda 处理程序。
- 记录处理程序 — 从您的数据库实例中检索数据记录的 Lambda 处理程序。
- 复合处理程序 — 从您的数据库实例中检索元数据和数据记录的 Lambda 处理程序。
- 属性或参数 — 处理程序用来提取数据库信息的数据库属性。您可以将这些属性配置为 Lambda 环境变量。
- 连接字符串 — 用于建立数据库实例连接的文本字符串。
- 目录 — 向 Athena 注册的非 AWS Glue 目录，是 `connection_string` 属性的必要前缀。
- 多路复用处理程序 — 可以接受和使用多个数据库连接的 Lambda 处理程序。

参数

使用本节中的 Lambda 环境变量来配置 Hortonworks Hive 连接器。

连接字符串

使用以下格式的 JDBC 连接字符串连接到数据库实例。

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

使用多路复用处理程序

您可以使用多路复用器通过单个 Lambda 函数连接到多个数据库实例。按目录名称来路由请求。在 Lambda 中使用以下类。

处理程序	类
复合处理程序	HiveMuxCompositeHandler
元数据处理程序	HiveMuxMetadataHandler
记录处理程序	HiveMuxRecordHandler

多路复用处理程序参数

参数	描述
$*catalog*_connection_string	必需。数据库实例连接字符串。将 Athena 中使用的目录的名称作为环境变量前缀。例如，如果向 Athena 注册的目录是 myhivecatalog ，则环境变量名称是 myhivecatalog_connection_string 。
default	必需。默认连接字符串。目录为 lambda:${ *AWS_LAMBDA_FUNCTION_NAME* } 时使用此字符串。

以下示例属性适用于支持两个数据库实例的 Hive MUX Lambda 函数：hive1（默认）和 hive2。

属性	值
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

提供凭证

要在 JDBC 连接字符串中为数据库提供用户名和密码，可以使用连接字符串属性或 AWS Secrets Manager。

- 连接字符串 — 可以将用户名和密码指定为 JDBC 连接字符串中的属性。

  > **Important**
  >
  > 作为安全最佳实践，请勿在您的环境变量或连接字符串中使用硬编码凭证。有关将硬编码密钥移至 AWS Secrets Manager 的信息，请参阅《AWS Secrets Manager 用户指南》中的将硬编码密钥移至 AWS Secrets Manager。

- AWS Secrets Manager - 要将 Athena 联合查询功能与 AWS Secrets Manager 配合使用，连接到您的 Lambda 函数的 VPC 应该拥有互联网访问权限或者 VPC 端点，以连接到 Secrets Manager。

  您可以将 AWS Secrets Manager 中的密钥名称放入您的 JDBC 连接字符串中。连接器将该密钥名称替换为来自 Secrets Manager 的 `username` 和 `password` 值。

  对于 Amazon RDS 数据库实例，将紧密集成这种支持。如果您使用 Amazon RDS，我们强烈建议您使用 AWS Secrets Manager 和凭证轮换。如果您的数据库不使用 Amazon RDS，请按以下格式将凭证存储为 JSON：

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

带有密钥名称的示例连接字符串

以下字符串带有密钥名称 ${Test/RDS/hive1host}。

```
hive://jdbc:hive2://hive1host:10000/default?...&${Test/RDS/hive1host}&...
```

该连接器使用该密钥名称来检索密钥，并提供用户名和密码，如以下示例所示。

```
hive://jdbc:hive2://hive1host:10000/default?...&UID=sample2&PWD=sample2&...
```

目前，Hortonworks Hive 连接器可以识别 UID 和 PWD JDBC 属性。

使用单个连接处理程序

您可以使用以下单个连接元数据和记录处理程序连接到单个 Hortonworks Hive 实例。

处理程序类型	类
复合处理程序	HiveCompositeHandler
元数据处理程序	HiveMetadataHandler
记录处理程序	HiveRecordHandler

单个连接处理程序参数

参数	描述
default	必需。默认连接字符串。

单个连接处理程序支持一个数据库实例，并且必须提供 default 连接字符串参数。将忽略所有其他连接字符串。

以下示例属性适用于 Lambda 函数支持的单个 Hortonworks Hive 实例。

属性	值
default	hive://jdbc:hive2://hive1host:10000/default?secret=${Test/RDS/hive1host}

溢出参数

Lambda 开发工具包可以将数据溢出到 Amazon S3。由同一 Lambda 函数访问的所有数据库实例都会溢出到同一位置。

参数	描述
`spill_bucket`	必需。溢出桶名称。
`spill_prefix`	必需。溢出桶密钥前缀。
`spill_put_request_headers`	（可选）用于溢出的 Amazon S3 `putObject` 请求的请求标头和值的 JSON 编码映射（例如，`{"x-amz-server-side-encryption" : "AES256"}`)。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。

数据类型支持

下表显示了适用于 JDBC、Hortonworks Hive 和 Arrow 的相应数据类型。

JDBC	Hortonworks Hive	Arrow
布尔值	布尔值	位
整数	TINYINT	Tiny
短型	SMALLINT	Smallint
整数	INT	Int
长整型	BIGINT	Bigint
float	float4	Float4
Double	float8	Float8
Date	date	DateDay
Timestamp	时间戳	DateMilli

JDBC	Hortonworks Hive	Arrow
String	VARCHAR	Varchar
字节	bytes	Varbinary
BigDecimal	十进制	十进制
ARRAY	不适用（见注释）	列出

> **Note**
>
> 目前，Hortonworks Hive 不支持聚合类型 ARRAY、MAP、STRUCT 或 UNIONTYPE。聚合类型的列被视为 SQL 中的 VARCHAR 列。

分区和拆分

分区用于确定如何为该连接器生成拆分。Athena 将构建一个 varchar 类型的合成列，它将展示表的分区方案，以帮助该连接器生成拆分。该连接器不会修改实际的表定义。

Performance

Hortonworks Hive 支持静态分区。Athena Hortonworks Hive 连接器可从这些分区并行检索数据。如果您想查询具有均匀分区分布的非常大的数据集，强烈建议使用静态分区。选择列的子集可以显著减少查询运行时及扫描的数据。Hortonworks Hive 连接器能够灵活地应对并发造成的节流。

Athena Hortonworks Hive 连接器执行谓词下推，以减少查询扫描的数据量。LIMIT 子句、简单谓词和复杂表达式将下推到连接器，以减少扫描数据量并缩短查询执行的运行时间。

LIMIT 子句

LIMIT N 语句用于减少查询扫描的数据量。LIMIT N 下推时，连接器仅向 Athena 返回 N 行。

Predicates

谓词是 SQL 查询的 WHERE 子句中的表达式，其评估结果为布尔值并根据多个条件筛选行。Athena Hortonworks Hive 连接器可以组合这些表达式并将其直接推送到 Hortonworks Hive，以增强功能并减少扫描的数据量。

以下 Athena Hortonworks Hive 连接器运算符支持谓词下推：

- 布尔值：AND、OR、NOT
- 相
  等：EQUAL、NOT_EQUAL、LESS_THAN、LESS_THAN_OR_EQUAL、GREATER_THAN、GREATER_
- 算术：ADD、SUBTRACT、MULTIPLY、DIVIDE、MODULUS、NEGATE
- 其他：LIKE_PATTERN、IN

组合下推示例

要增强查询功能，组合下推类型，如以下示例所示：

```
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;
```

传递查询

Hortonworks Hive 连接器支持传递查询。传递查询使用表函数将完整查询下推到数据来源来执行查询。

要在 Hortonworks Hive 中执行传递查询，可以使用以下语法：

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

以下示例查询将查询下推到 Hortonworks Hive 中的数据来源。该查询选择了 customer 表中的所有列，将结果限制为 10。

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

许可证信息

使用此连接器，即表示您确认包含第三方组件（这些组件的列表可在此连接器的 pom.xml 文件中找到），并同意 GitHub.com 上的 LICENSE.txt 文件中提供的相应第三方许可证中的条款。

其他 资源

有关最新 JDBC 驱动程序版本信息，请参阅 GitHub.com 上适用于 Hortonworks Hive 连接器的 pom.xml 文件。

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena Apache Kafka 连接器

可通过适用于 Apache Kafka 的 Amazon Athena 连接器支持 Amazon Athena 对 Apace Kafka 主题运行 SQL 查询。使用此连接器在 Athena 中以表的形式查看 Apache Kafka 主题，以行的形式查看消息。

先决条件

可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

限制

- 不支持写入 DDL 操作。
- 任何相关的 Lambda 限制。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。
- 必须将筛选条件中的日期和时间戳数据类型转换为适当的数据类型。
- CSV 文件类型不支持日期和时间戳数据类型，它们被视为 VARCHAR 值。
- 不支持映射到嵌套 JSON 字段。连接器仅映射顶级字段。
- 连接器不支持复杂类型。复杂类型解释为字符串。
- 要提取或处理复杂的 JSON 值，请使用 Athena 中可用的 JSON 相关函数。有关更多信息，请参阅从 JSON 中提取数据。
- 连接器不支持对 Kafka 消息元数据的访问。

术语

- 元数据处理程序 — 从您的数据库实例中检索元数据的 Lambda 处理程序。

- 记录处理程序 — 从您的数据库实例中检索数据记录的 Lambda 处理程序。
- 复合处理程序 — 从您的数据库实例中检索元数据和数据记录的 Lambda 处理程序。
- Kafka 端点 – 文本字符串，用于建立与 Kafka 实例的连接。

集群兼容性

Kafka 连接器可用于以下集群类型。

- 独立 Kafka - 与 Kafka 直接连接（经过或未经身份验证）。
- Confluent - 与 Confluent Kafka 直接连接。有关将 Athena 与 Confluent Kafka 数据配合使用的信息，请参阅 AWS 商业智能博客中的使用 Amazon Athena 在 Amazon QuickSight 中可视化 Confluent 数据。

连接到 Confluent

要连接到 Confluent，需要执行以下步骤：

1. 从 Confluent 生成 API 密钥。
2. 将 Confluent API 密钥的用户名和密码存储到 AWS Secrets Manager 中。
3. 在 Kafka 连接器中提供 `secrets_manager_secret` 环境变量的密钥名称。
4. 按照本文档的 设置 Kafka 连接器 节中的步骤执行操作。

支持的身份验证方法

连接器支持以下身份验证方法。

- SSL
- SASL/SCRAM
- SASL/PLAIN
- SASL/PLAINTEXT
- NO_AUTH
- 自行管理的 Kafka 和 Confluent 平台 – SSL、SASL/SCRAM、SASL/PLAINTEXT、NO_AUTH
- 自行管理的 Kafka 和 Confluent Cloud - SASL/PLAIN

有关更多信息，请参阅 为 Athena Kafka 连接器配置身份验证。

支持的输入数据格式

连接器支持以下输入数据格式。

- JSON
- CSV

参数

使用本节中介绍的 Lambda 环境变量配置 Athena Kafka 连接器。

- auth_type – 指定集群的身份验证类型。连接器支持以下身份验证类型：
  - NO_AUTH - 直接连接到 Kafka（例如，连接到部署在 EC2 实例上的 Kafka 集群，但不使用身份验证）。
  - SASL_SSL_PLAIN – 此方法使用 `SASL_SSL` 安全协议和 `PLAIN` SASL 机制。有关更多信息，请参阅 Apache Kafka 文档中的 SASL 配置。
  - SASL_PLAINTEXT_PLAIN – 此方法使用 `SASL_PLAINTEXT` 安全协议和 `PLAIN` SASL 机制。有关更多信息，请参阅 Apache Kafka 文档中的 SASL 配置。
  - SASL_SSL_SCRAM_SHA512 - 您可以使用此身份验证类型控制对 Apache Kafka 集群的访问权限。此方法将用户名和密码存储在 AWS Secrets Manager 中。密钥必须与 Kafka 集群相关。有关更多信息，请参阅 Apache Kafka 文档中的使用 SASL/SCRAM 进行身份验证。
  - SASL_PLAINTEXT_SCRAM_SHA512 - 此方法使用 `SASL_PLAINTEXT` 安全协议和 `SCRAM_SHA512` SASL 机制。此方法使用存储在 AWS Secrets Manager 中的用户名和密码。有关更多信息，请参阅 Apache Kafka 文档中的 SASL 配置一节。
  - SSL - SSL 身份验证使用密钥存储和信任存储文件来连接 Apache Kafka 集群。您必须生成信任存储和密钥存储文件，将其上传到 Amazon S3 存储桶，并在部署连接器时提供对 Amazon S3 的引用。密钥存储、信任存储和 SSL 密钥存储在 AWS Secrets Manager 中。部署连接器时需要提供 AWS 私有密钥。有关更多信息，请参阅 Apache Kafka 文档中的使用 SSL 进行加密和身份验证。

    有关更多信息，请参阅 为 Athena Kafka 连接器配置身份验证。
- certificates_s3_reference – 包含证书（密钥存储和信任存储文件）的 Amazon S3 位置。
- disable_spill_encryption -（可选）当设置为 `True` 时，将禁用溢出加密。默认值为 `False`，此时将使用 AES-GCM 对溢出到 S3 的数据使用进行加密 - 使用随机生成的密钥，或者使用 KMS 生成密钥。禁用溢出加密可以提高性能，尤其是当您的溢出位置使用服务器端加密时。
- kafka_endpoint – 提供给 Kafka 的端点详细信息。
- secrets_manager_secret – 保存凭证的 AWS 密钥名称。

- 溢出参数 – Lambda 函数将不适合内存的数据临时存储（“溢出”）到 Amazon S3。由同一 Lambda 函数访问的所有数据库实例都会溢出到同一位置。使用下表中的参数指定溢出位置。

参数	描述
`spill_bucket`	必需。Amazon S3 存储桶的名称，Lambda 函数可以在该存储桶中溢出数据。
`spill_prefix`	必需。溢出存储桶中的前缀，Lambda 函数可以在该存储桶中溢出数据。
`spill_put_request_headers`	（可选）用于溢出的 Amazon S3 `putObject` 请求的请求标头和值的 JSON 编码映射（例如，`{"x-amz-server-side-encryption" : "AES256"}` ）。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。

- 子网 ID - 与 Lambda 函数可用于访问数据来源的子网对应的一个或多个子网 ID。
  - 公有 Kafka 集群或标准 Confluent Cloud 集群 - 将连接器与具有 NAT 网关的私有子网关联。
  - 具有私有连接的 Confluent Cloud 集群 - 将连接器与具有通往 Confluent Cloud 集群的路由的私有子网关联。
    - 对于 AWS 中转网关，子网必须位于连接到 Confluent Cloud 使用的相同中转网关的 VPC 中。
    - 对于 VPC 对等，子网必须位于与 Confluent Cloud VPC 对等的 VPC 中。
    - 对于 AWS PrivateLink，子网必须位于具有通往 VPC 端点的路由的 VPC 中，该端点连接到 Confluent Cloud。

> **Note**
>
> 如果您将连接器部署到 VPC 中以访问私有资源，并且还想连接到 Confluent 等可公开访问的服务，则必须将连接器关联到某个具有 NAT 网关的私有子网。有关更多信息，请参阅《Amazon VPC 用户指南》中的 NAT 网关。

数据类型支持

下表显示了 Kafka 和 Apache Arrow 支持的相应数据类型。

Kafka	Arrow
CHAR	VARCHAR
VARCHAR	VARCHAR
TIMESTAMP	MILLISECOND
DATE	DAY
BOOLEAN	BOOL
SMALLINT	SMALLINT
INTEGER	INT
BIGINT	BIGINT
DECIMAL	FLOAT8
DOUBLE	FLOAT8

分区和拆分

Kafka 主题分为多个分区。每个分区会进行排序。分区中的每条消息都有一个增量 ID，称为偏移。每个 Kafka 分区进一步分为多个分区，以进行并行处理。数据在 Kafka 集群中配置的保留期内可用。

最佳实践

最佳做法是在查询 Athena 时使用谓词下推，如下例所示。

```
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'
```

设置 Kafka 连接器

在使用连接器之前，您必须设置 Apache Kafka 集群，使用 AWS Glue 架构注册表定义架构，并为连接器配置身份验证。

使用 AWS Glue 架构注册表时，请注意以下几点：

- 确保 AWS Glue 架构注册表的 Description（描述）字段中的文本包含 `{AthenaFederationKafka}` 字符串。此标记字符串是与 Amazon Athena Kafka 连接器一起使用的 AWS Glue 注册表中的必需项。
- 为了获得最佳性能，数据库名称和表名称仅限使用小写。使用混合大小写会使连接器执行不区分大小写的搜索，这种搜索的计算密集度更高。

设置 Apache Kafka 环境和 AWS Glue 架构注册表

1. 设置 Apache Kafka 环境。
2. 将 JSON 格式的 Kafka 主题描述文件（即其架构）上传到 AWS Glue 架构注册表。有关更多信息，请参阅《AWS Glue 开发人员指南》中的与 AWS Glue 架构注册表集成。有关详细示例，请参阅以下章节。

AWS Glue 架构注册表的架构示例

将架构上传到 AWS Glue 架构注册表时，请使用本节中的示例格式。

JSON 类型架构示例

在以下示例中，要在 AWS Glue 架构注册表中创建的架构指定 `json` 作为 `dataFormat` 的值，并将 `datatypejson` 用于 `topicName`。

> **Note**
>
> `topicName` 的值应使用与 Kafka 中主题名称相同的大小写。

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

CSV 类型架构示例

在以下示例中，要在 AWS Glue 架构注册表中创建的架构指定 `csv` 作为 `dataFormat` 的值，并将 `datatypecsvbulk` 用于 `topicName`。`topicName` 的值应使用与 Kafka 中主题名称相同的大小写。

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

为 Athena Kafka 连接器配置身份验证

您可以使用多种方法对 Apache Kafka 集群进行身份验证，包括 SSL、SASL/SCRAM、SASL/PLAIN 和 SASL/PLAINTEXT。

下表显示了连接器的身份验证类型以及每种连接器的安全协议和 SASL 机制。有关更多信息，请参阅 Apache Kafka 文档中的安全性一节。

auth_type	security.protocol	sasl.mechanism	集群类型兼容性
SASL_SSL_PLAIN	SASL_SSL	PLAIN	• 自行管理的 Kafka • Confluent 平台 • Confluent Cloud
SASL_PLAINTEXT_PLAIN	SASL_PLAINTEXT	PLAIN	• 自行管理的 Kafka • Confluent 平台
SASL_SSL_SCRAM_SHA512	SASL_SSL	SCRAM-SHA-512	• 自行管理的 Kafka • Confluent 平台
SASL_PLAINTEXT_SCRAM_SHA512	SASL_PLAINTEXT	SCRAM-SHA-512	• 自行管理的 Kafka • Confluent 平台

auth_type	security.protocol	sasl.mechanism	集群类型兼容性
SSL	SSL	不适用	• 自行管理的 Kafka • Confluent 平台

SSL

如果集群经过 SSL 身份验证，则您必须生成信任存储和密钥存储文件，并将其上传到 Amazon S3 存储桶。部署连接器时，必须提供此 Amazon S3 参考资料。密钥存储、信任存储和 SSL 密钥存储在 AWS Secrets Manager 中。部署连接器时需要提供 AWS 密钥。

有关在 Secrets Manager 中创建密钥的信息，请参阅创建 AWS Secrets Manager 密钥。

要使用此身份验证类型，请按下表所示设置环境变量。

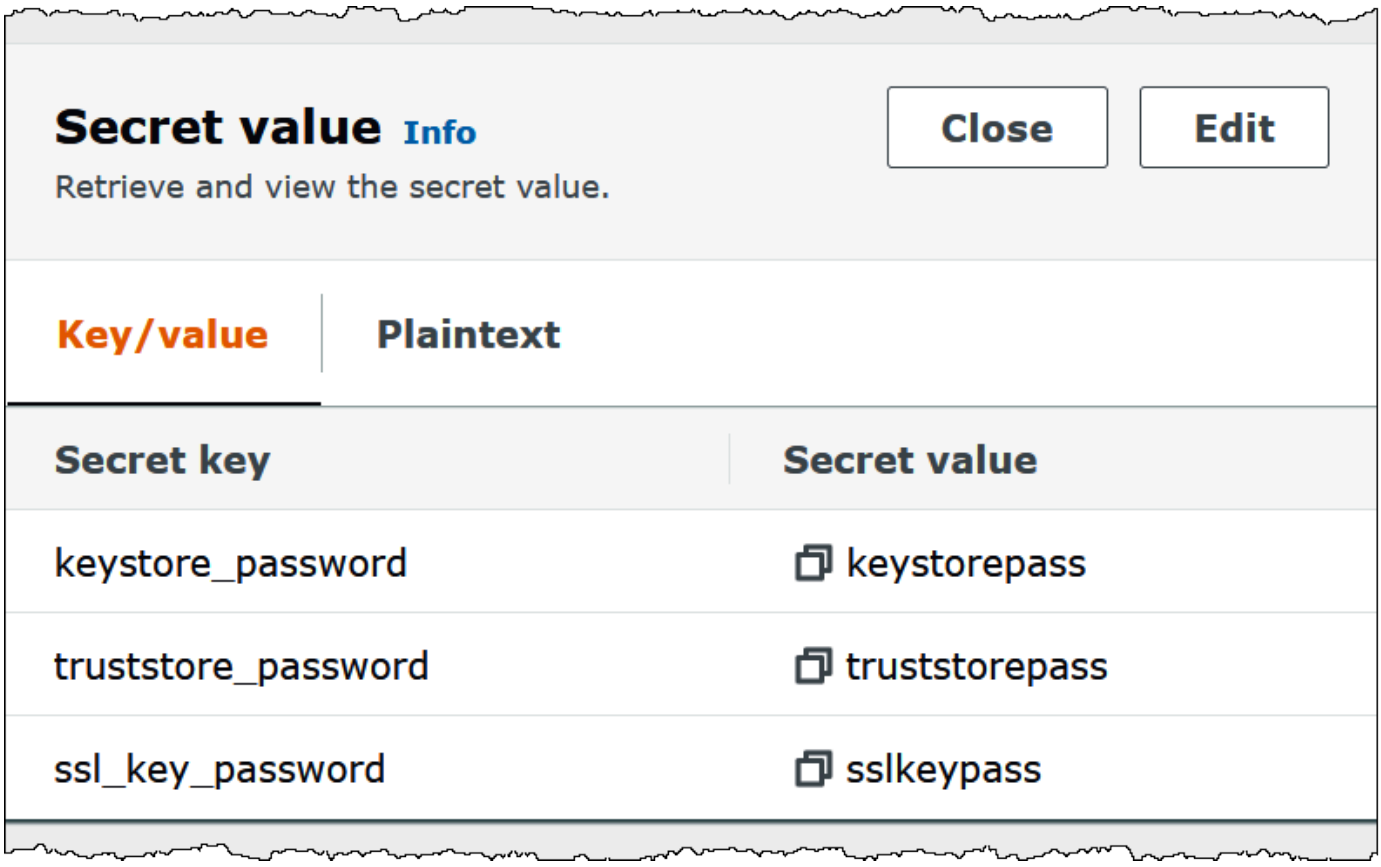
参数	值
auth_type	SSL
certificates_s3_reference	包含证书的 Amazon S3 位置。
secrets_manager_secret	您的 AWS 密钥名称。

在 Secrets Manager 中创建密钥后，您可以在 Secrets Manager 控制台中查看该密钥。

在 Secrets Manager 中查看密钥

1. 打开 Secrets Manager 控制台，网址为 https://console.aws.amazon.com/secretsmanager/。
2. 在导航窗格中，选择 Secrets（密钥）。
3. 在 Secrets（密钥）页面，选择密钥链接。
4. 在密钥的详细信息页面上，选择 Retrieve secret value（检索密钥值）。

   下图显示了示例密钥，其中包含三个键值对：keystore_password、truststore_password 和 ssl_key_password。



有关在 Kafka 中使用 SSL 的更多信息，请参阅 Apache Kafka 文档中的使用 SSL 进行加密和身份验证。

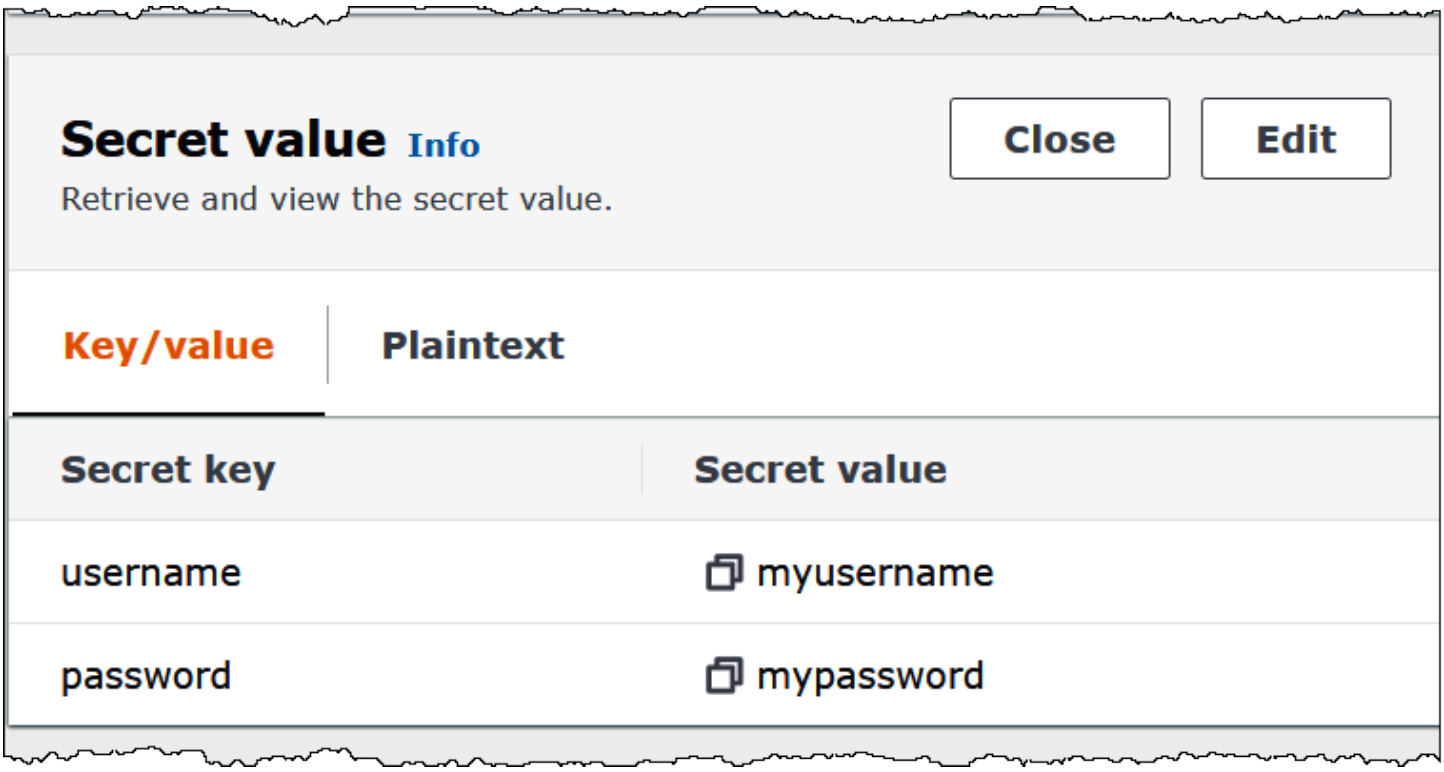
SASL/SCRAM

如果您的集群使用 SCRAM 身份验证，请在部署连接器时提供与集群关联的 Secrets Manager 密钥。用户的 AWS 凭证（密钥和访问密钥）用于与集群进行身份验证。

按下表所示设置环境变量。

参数	值
`auth_type`	`SASL_SSL_SCRAM_SHA512`
`secrets_manager_secret`	您的 AWS 密钥名称。

下图显示了 Secrets Manager 控制台中的示例密钥，其中包含两个键值对：一个用于 `username`，另一个用于 `password`。



有关在 Kafka 中使用 SASL/SCRAM 的更多信息，请参阅 Apache Kafka 文档中的使用 SASL/SCRAM 进行身份验证。

许可证信息

使用此连接器，即表示您确认包含第三方组件（这些组件的列表可在此连接器的 pom.xml 文件中找到），并同意 GitHub.com 上的 LICENSE.txt 文件中提供的相应第三方许可证中的条款。

其他 资源

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena MSK 连接器

使用适用于 Amazon MSK 的 Amazon Athena 连接器，Amazon Athena 能够对 Apace Kafka 主题运行 SQL 查询。使用此连接器在 Athena 中以表的形式查看 Apache Kafka 主题，以行的形式查看消息。有关更多信息，请参阅 AWS 大数据博客中的 Analyze real-time streaming data in Amazon MSK with Amazon Athena（使用 Amazon Athena 分析 Amazon MSK 中的实时流数据）。

先决条件

可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

限制

- 不支持写入 DDL 操作。
- 任何相关的 Lambda 限制。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。
- 必须将筛选条件中的日期和时间戳数据类型转换为适当的数据类型。
- CSV 文件类型不支持日期和时间戳数据类型，它们被视为 VARCHAR 值。
- 不支持映射到嵌套 JSON 字段。连接器仅映射顶级字段。
- 连接器不支持复杂类型。复杂类型解释为字符串。
- 要提取或处理复杂的 JSON 值，请使用 Athena 中可用的 JSON 相关函数。有关更多信息，请参阅从 JSON 中提取数据。
- 连接器不支持对 Kafka 消息元数据的访问。

术语

- 元数据处理程序 — 从您的数据库实例中检索元数据的 Lambda 处理程序。
- 记录处理程序 — 从您的数据库实例中检索数据记录的 Lambda 处理程序。
- 复合处理程序 — 从您的数据库实例中检索元数据和数据记录的 Lambda 处理程序。
- Kafka 端点 – 文本字符串，用于建立与 Kafka 实例的连接。

集群兼容性

MSK 连接器可用于以下集群类型。

- MSK 预置集群 – 您可以手动指定、监控和扩展集群容量。
- MSK 无服务器集群 – 提供随应用程序 I/O 扩展而自动扩展的按需容量。
- 独立 Kafka - 与 Kafka 直接连接（经过或未经身份验证）。

支持的身份验证方法

连接器支持以下身份验证方法。

- SASL/IAM
- SSL
- SASL/SCRAM
- SASL/PLAIN
- SASL/PLAINTEXT
- NO_AUTH

有关更多信息，请参阅 为 Athena MSK 连接器配置身份验证。

支持的输入数据格式

连接器支持以下输入数据格式。

- JSON
- CSV

参数

使用本节中介绍的 Lambda 环境变量来配置 Athena MSK 连接器。

- auth_type – 指定集群的身份验证类型。连接器支持以下身份验证类型：
  - NO_AUTH – 无需身份验证即可直接连接到 Kafka（例如，连接到部署在 EC2 实例上的 Kafka 集群，但不使用身份验证）。
  - SASL_SSL_PLAIN – 此方法使用 `SASL_SSL` 安全协议和 `PLAIN` SASL 机制。
  - SASL_PLAINTEXT_PLAIN – 此方法使用 `SASL_PLAINTEXT` 安全协议和 `PLAIN` SASL 机制。

    > **Note**
    >
    > Apache Kafka 支持 `SASL_SSL_PLAIN` 和 `SASL_PLAINTEXT_PLAIN` 身份验证类型，但 Amazon MSK 不支持。

  - SASL_SSL_AWS_MSK_IAM – Amazon MSK 的 IAM 访问控制使您能够处理 MSK 集群的身份验证和授权。您用户的 AWS 凭证（密钥和访问密钥）用于与集群连接。有关更多信息，请参阅

《Amazon Managed Streaming for Apache Kafka 开发人员指南》中的 IAM access control（IAM 访问控制）。

- SASL_SSL_SCRAM_SHA512 – 您可以使用这种身份验证类型来控制对 Amazon MSK 集群的访问权限。此方法将用户名和密码存储在 AWS Secrets Manager 上。密钥必须与 Amazon MSK 集群相关。有关更多信息，请参阅《Amazon Managed Streaming for Apache Kafka 开发人员指南》中的为 Amazon MSK 集群设置 SASL/SCRAM 身份验证。
- SSL – SSL 身份验证使用密钥存储和信任存储文件来连接 Amazon MSK 集群。您必须生成信任存储和密钥存储文件，将其上传到 Amazon S3 存储桶，并在部署连接器时提供对 Amazon S3 的引用。密钥存储、信任存储和 SSL 密钥存储在 AWS Secrets Manager 中。部署连接器时需要提供 AWS 私有密钥。有关更多信息，请参阅《Amazon Managed Streaming for Apache Kafka 开发人员指南》中的 Mutual TLS authentication（双向 TLS 身份验证）。

  有关更多信息，请参阅 为 Athena MSK 连接器配置身份验证。

- certificates_s3_reference – 包含证书（密钥存储和信任存储文件）的 Amazon S3 位置。
- disable_spill_encryption -（可选）当设置为 `True` 时，将禁用溢出加密。默认值为 `False`，此时将使用 AES-GCM 对溢出到 S3 的数据使用进行加密 - 使用随机生成的密钥，或者使用 KMS 生成密钥。禁用溢出加密可以提高性能，尤其是当您的溢出位置使用服务器端加密时。
- kafka_endpoint – 提供给 Kafka 的端点详细信息。例如，对于 Amazon MSK 集群，您可以为该集群提供引导 URL。
- secrets_manager_secret – 保存凭证的 AWS 密钥名称。IAM 身份验证不需要此参数。
- 溢出参数 – Lambda 函数将不适合内存的数据临时存储（“溢出”）到 Amazon S3。由同一 Lambda 函数访问的所有数据库实例都会溢出到同一位置。使用下表中的参数指定溢出位置。

参数	描述
`spill_bucket`	必需。Amazon S3 存储桶的名称，Lambda 函数可以在该存储桶中溢出数据。
`spill_prefix`	必需。溢出存储桶中的前缀，Lambda 函数可以在该存储桶中溢出数据。
`spill_put_request_headers`	（可选）用于溢出的 Amazon S3 `putObject` 请求的请求标头和值的 JSON 编码映射（例如，`{"x-amz-server-side-encryption" : "AES256"}`）。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。

数据类型支持

下表显示了 Kafka 和 Apache Arrow 支持的相应数据类型。

Kafka	Arrow
CHAR	VARCHAR
VARCHAR	VARCHAR
TIMESTAMP	MILLISECOND
DATE	DAY
BOOLEAN	BOOL
SMALLINT	SMALLINT
INTEGER	INT
BIGINT	BIGINT
DECIMAL	FLOAT8
DOUBLE	FLOAT8

分区和拆分

Kafka 主题分为多个分区。每个分区会进行排序。分区中的每条消息都有一个增量 ID，称为偏移。每个 Kafka 分区进一步分为多个分区，以进行并行处理。数据在 Kafka 集群中配置的保留期内可用。

最佳实践

最佳做法是在查询 Athena 时使用谓词下推，如下例所示。

```
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'
```

设置 MSK 连接器

在使用连接器之前，您必须设置 Amazon MSK 集群，使用 AWS Glue 架构注册表定义架构，并为连接器配置身份验证。

> **Note**
>
> 如果您将连接器部署到 VPC 中以访问私有资源，并且还想连接到 Confluent 等可公开访问的服务，则必须将连接器关联到某个具有 NAT 网关的私有子网。有关更多信息，请参阅《Amazon VPC 用户指南》中的 NAT 网关。

使用 AWS Glue 架构注册表时，请注意以下几点：

- 确保 AWS Glue 架构注册表的 Description（描述）字段中的文本包含 `{AthenaFederationMSK}` 字符串。此标记字符串是与 Amazon Athena MSK 连接器一起使用的 AWS Glue 注册表中的必填字段。
- 为了获得最佳性能，数据库名称和表名称仅限使用小写。使用混合大小写会使连接器执行不区分大小写的搜索，这种搜索的计算密集度更高。

设置 Amazon MSK 环境和 AWS Glue 架构注册表

1. 设置 Amazon MSK 环境。有关信息和步骤，请参阅《Amazon Managed Streaming for Apache Kafka 开发人员指南》中的 Setting up Amazon MSK（设置 Amazon MSK）和 Getting started using Amazon MSK（开始使用 Amazon MSK）。
2. 将 JSON 格式的 Kafka 主题描述文件（即其架构）上传到 AWS Glue 架构注册表。有关更多信息，请参阅《AWS Glue 开发人员指南》中的与 AWS Glue 架构注册表集成。有关详细示例，请参阅以下章节。

AWS Glue 架构注册表的架构示例

将架构上传到 AWS Glue 架构注册表时，请使用本节中的示例格式。

JSON 类型架构示例

在以下示例中，要在 AWS Glue 架构注册表中创建的架构指定 `json` 作为 `dataFormat` 的值，并将 `datatypejson` 用于 `topicName`。

> **Note**
>
> topicName 的值应使用与 Kafka 中主题名称相同的大小写。

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

CSV 类型架构示例

在以下示例中，要在 AWS Glue 架构注册表中创建的架构指定 `csv` 作为 `dataFormat` 的值，并将 `datatypecsvbulk` 用于 `topicName`。`topicName` 的值应使用与 Kafka 中主题名称相同的大小写。

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

为 Athena MSK 连接器配置身份验证

您可以使用多种方法对 Amazon MSK 集群进行身份验证，包括 IAM、SSL、SCRAM 和独立的 Kafka。

下表显示了连接器的身份验证类型以及每种连接器的安全协议和 SASL 机制。有关更多信息，请参阅《Amazon Managed Streaming for Apache Kafka 开发人员指南》中的 Authentication and authorization for Apache Kafka APIs（Apache Kafka API 的身份验证和授权）。

auth_type	security.protocol	sasl.mechanism
SASL_SSL_PLAIN	SASL_SSL	PLAIN

auth_type	security.protocol	sasl.mechanism
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	不适用

> **Note**
>
> Apache Kafka 支持 SASL_SSL_PLAIN 和 SASL_PLAINTEXT_PLAIN 身份验证类型，但 Amazon MSK 不支持。

SASL/IAM

如果集群使用 IAM 身份验证，则在设置集群时必须为用户配置 IAM policy。有关更多信息，请参阅《Amazon Managed Streaming for Apache Kafka 开发人员指南》中的 IAM access control（IAM 访问控制）。

要使用此身份验证类型，请将连接器的 auth_type Lambda 环境变量设置为 SASL_SSL_AWS_MSK_IAM。

SSL

如果集群经过 SSL 身份验证，则您必须生成信任存储和密钥存储文件，并将其上传到 Amazon S3 存储桶。部署连接器时，必须提供此 Amazon S3 参考资料。密钥存储、信任存储和 SSL 密钥存储在 AWS Secrets Manager 中。部署连接器时需要提供 AWS 密钥。

有关在 Secrets Manager 中创建密钥的信息，请参阅创建 AWS Secrets Manager 密钥。

要使用此身份验证类型，请按下表所示设置环境变量。

参数	值
auth_type	SSL

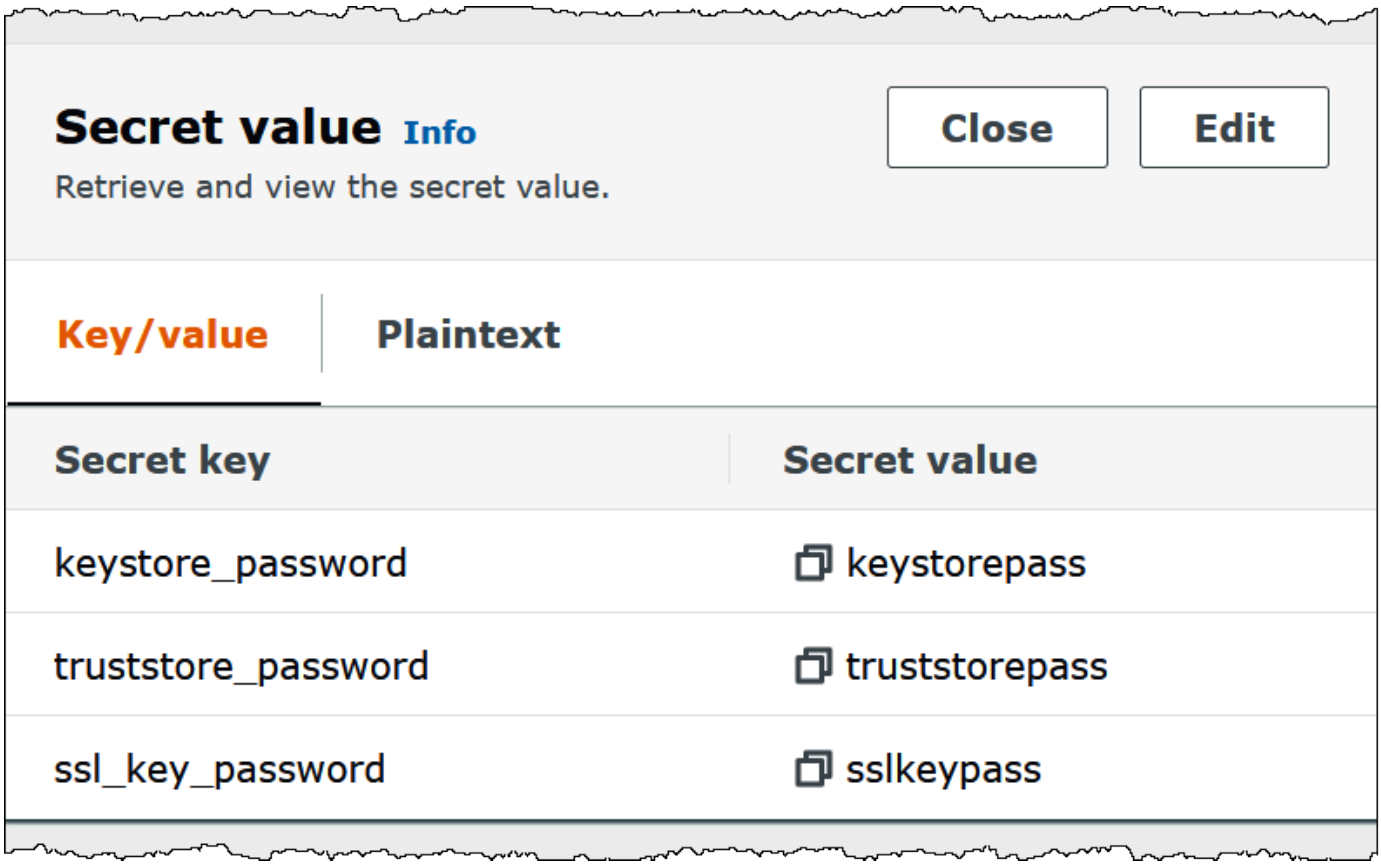
参数	值
certificates_s3_reference	包含证书的 Amazon S3 位置。
secrets_manager_secret	您的 AWS 密钥名称。

在 Secrets Manager 中创建密钥后，您可以在 Secrets Manager 控制台中查看该密钥。

在 Secrets Manager 中查看密钥

1. 打开 Secrets Manager 控制台，网址为 https://console.aws.amazon.com/secretsmanager/。
2. 在导航窗格中，选择 Secrets（密钥）。
3. 在 Secrets（密钥）页面，选择密钥链接。
4. 在密钥的详细信息页面上，选择 Retrieve secret value（检索密钥值）。

   下图显示了示例密钥，其中包含三个键值对：keystore_password、truststore_password 和 ssl_key_password。



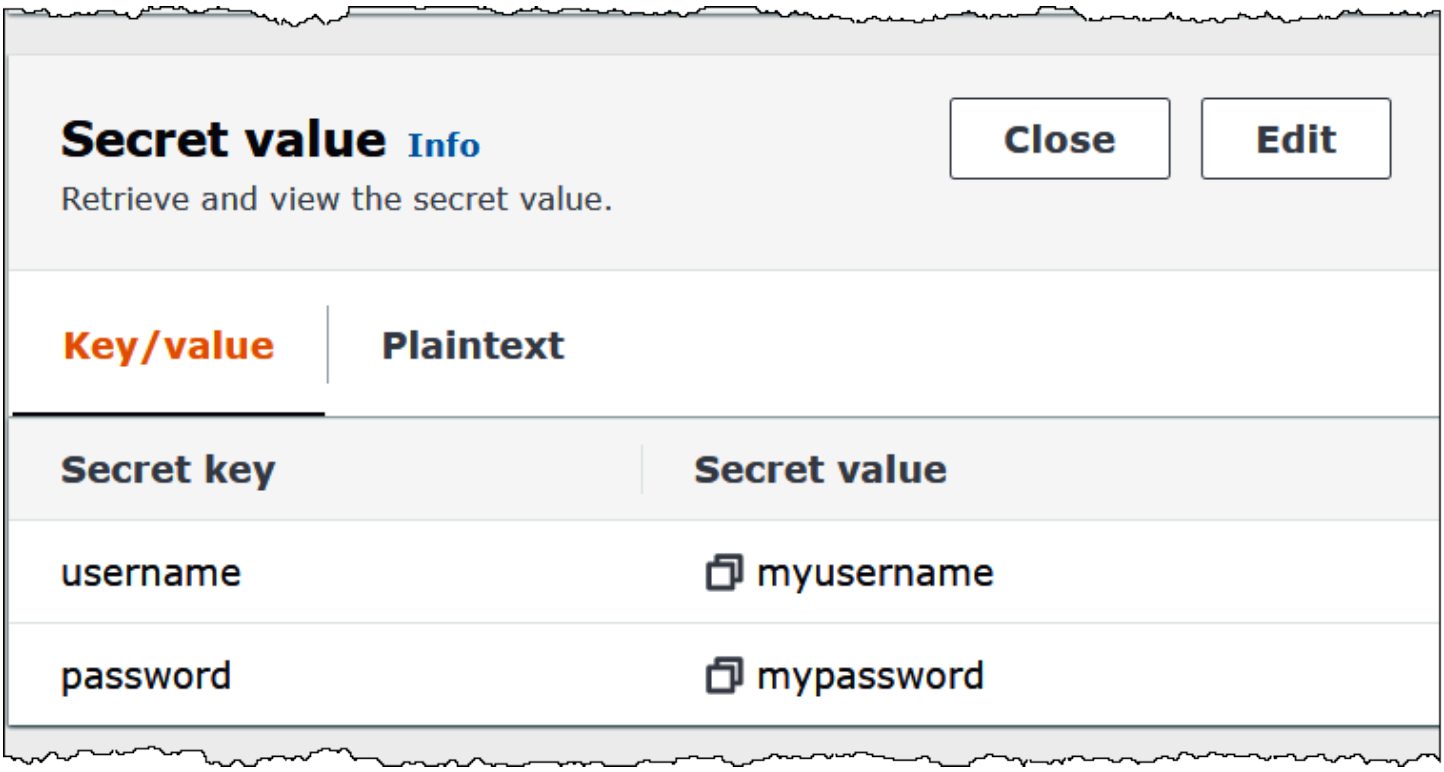
SASL/SCRAM

如果您的集群使用 SCRAM 身份验证，请在部署连接器时提供与集群关联的 Secrets Manager 密钥。用户的 AWS 凭证（密钥和访问密钥）用于与集群进行身份验证。

按下表所示设置环境变量。

参数	值
`auth_type`	`SASL_SSL_SCRAM_SHA512`
`secrets_manager_secret`	您的 AWS 密钥名称。

下图显示了 Secrets Manager 控制台中的示例密钥，其中包含两个键值对：一个用于 `username`，另一个用于 `password`。



许可证信息

使用此连接器，即表示您确认包含第三方组件（这些组件的列表可在此连接器的 pom.xml 文件中找到），并同意 GitHub.com 上的 LICENSE.txt 文件中提供的相应第三方许可证中的条款。

其他 资源

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena MySQL 连接器

Amazon Athena Lambda MySQL 连接器使 Amazon Athena 可以访问 MySQL 数据库。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。
- 使用此连接器前，请先设置 VPC 和安全组。有关更多信息，请参阅 为数据源连接器创建 VPC。

限制

- 不支持写入 DDL 操作。

- 在多路复用器设置中，溢出桶和前缀在所有数据库实例之间共享。
- 任何相关的 Lambda 限制。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。
- 由于 Athena 会将查询转换为小写，所以 MySQL 表名称必须使用小写。例如，针对名为 `myTable` 的表进行的 Athena 查询将失败。

术语

以下术语与 MySQL 连接器有关。

- 数据库实例 - 部署在本地、Amazon EC2 或 Amazon RDS 上的任何数据库实例。
- 处理程序 — 访问您数据库实例的 Lambda 处理程序。处理程序可以用于元数据或数据记录。
- 元数据处理程序 — 从您的数据库实例中检索元数据的 Lambda 处理程序。
- 记录处理程序 — 从您的数据库实例中检索数据记录的 Lambda 处理程序。
- 复合处理程序 — 从您的数据库实例中检索元数据和数据记录的 Lambda 处理程序。
- 属性或参数 — 处理程序用来提取数据库信息的数据库属性。您可以将这些属性配置为 Lambda 环境变量。
- 连接字符串 — 用于建立数据库实例连接的文本字符串。
- 目录 — 向 Athena 注册的非 AWS Glue 目录，是 `connection_string` 属性的必要前缀。
- 多路复用处理程序 — 可以接受和使用多个数据库连接的 Lambda 处理程序。

参数

使用本节中的 Lambda 环境变量来配置 MySQL 连接器。

连接字符串

使用以下格式的 JDBC 连接字符串连接到数据库实例。

```
mysql://${jdbc_connection_string}
```

> **Note**
>
> 如果您在对 MySQL 表执行 `SELECT` 查询时收到错误 java.sql.SQLException: Zero date value prohibited（java.sql.SQLException：禁止零日期值），请将以下参数添加到您的连接字符串中：

```
zeroDateTimeBehavior=convertToNull
```

有关更多信息，请参阅 GitHub.com 上的尝试从 MySQL 表中进行选择时出现“Zero date value prohibited”（禁止零日期值）错误。

使用多路复用处理程序

您可以使用多路复用器通过单个 Lambda 函数连接到多个数据库实例。按目录名称来路由请求。在 Lambda 中使用以下类。

处理程序	类
复合处理程序	MySqlMuxCompositeHandler
元数据处理程序	MySqlMuxMetadataHandler
记录处理程序	MySqlMuxRecordHandler

多路复用处理程序参数

参数	描述
$*catalog*_connection_string	必需。数据库实例连接字符串。将 Athena 中使用的目录的名称作为环境变量前缀。例如，如果向 Athena 注册的目录是 mymysqlcatalog ，则环境变量名称是 mymysqlcatalog_connection_string 。
default	必需。默认连接字符串。当目录为 lambda:${ *AWS_LAMBDA_FUNCTION_NAME* } 时使用此字符串。

以下示例属性适用于支持两个数据库实例的 MySql MUX Lambda 函数：mysql1（默认值）和 mysql2。

属性	值
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

提供凭证

要在 JDBC 连接字符串中为数据库提供用户名和密码，可以使用连接字符串属性或 AWS Secrets Manager。

- 连接字符串 — 可以将用户名和密码指定为 JDBC 连接字符串中的属性。

  > **Important**
  >
  > 作为安全最佳实践，请勿在您的环境变量或连接字符串中使用硬编码凭证。有关将硬编码密钥移至 AWS Secrets Manager 的信息，请参阅《AWS Secrets Manager 用户指南》中的将硬编码密钥移至 AWS Secrets Manager。

- AWS Secrets Manager - 要将 Athena 联合查询功能与 AWS Secrets Manager 配合使用，连接到您的 Lambda 函数的 VPC 应该拥有互联网访问权限或者 VPC 端点，以连接到 Secrets Manager。

  您可以将 AWS Secrets Manager 中的密钥名称放入您的 JDBC 连接字符串中。连接器将该密钥名称替换为来自 Secrets Manager 的 `username` 和 `password` 值。

  对于 Amazon RDS 数据库实例，将紧密集成这种支持。如果您使用 Amazon RDS，我们强烈建议您使用 AWS Secrets Manager 和凭证轮换。如果您的数据库不使用 Amazon RDS，请按以下格式将凭证存储为 JSON：

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

带有密钥名称的示例连接字符串

以下字符串带有密钥名称 ${Test/RDS/MySql1}。

```
mysql://jdbc:mysql://mysql1.host:3306/default?...&${Test/RDS/MySql1}&...
```

该连接器使用该密钥名称来检索密钥，并提供用户名和密码，如以下示例所示。

```
mysql://jdbc:mysql://mysql1host:3306/default?...&user=sample2&password=sample2&...
```

目前，MySQL 连接器可以识别 user 和 password JDBC 属性。

使用单个连接处理程序

您可以使用以下单个连接元数据和记录处理程序连接到单个 MySQL 实例。

处理程序类型	类
复合处理程序	MySqlCompositeHandler
元数据处理程序	MySqlMetadataHandler
记录处理程序	MySqlRecordHandler

单个连接处理程序参数

参数	描述
default	必需。默认连接字符串。

单个连接处理程序支持一个数据库实例，并且必须提供 default 连接字符串参数。将忽略所有其他连接字符串。

以下示例属性适用于 Lambda 函数支持的单个 MySQL 实例。

属性	值
default	mysql://mysql1.host:3306/default?secret=Test/RDS/ MySql1

溢出参数

Lambda 开发工具包可以将数据溢出到 Amazon S3。由同一 Lambda 函数访问的所有数据库实例都会溢出到同一位置。

参数	描述
`spill_bucket`	必需。溢出桶名称。
`spill_prefix`	必需。溢出桶密钥前缀。
`spill_put_request_headers`	（可选）用于溢出的 Amazon S3 `putObject` 请求的请求标头和值的 JSON 编码映射（例如，`{"x-amz-server-side-encryption" : "AES256"}`)。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。

数据类型支持

下表显示了适用于 JDBC 和 Arrow 的相应数据类型。

JDBC	Arrow
布尔值	位
整数	Tiny
短型	Smallint
整数	Int
长整型	Bigint
float	Float4
Double	Float8
Date	DateDay
Timestamp	DateMilli

JDBC	Arrow
String	Varchar
字节	Varbinary
BigDecimal	十进制
ARRAY	列出

分区和拆分

分区用于确定如何为该连接器生成拆分。Athena 将构建一个 `varchar` 类型的合成列，它将展示表的分区方案，以帮助该连接器生成拆分。该连接器不会修改实际的表定义。

Performance

MySQL 支持本机分区。Athena MySQL 连接器可从这些分区并行检索数据。如果您想查询具有统一分区分布的超大型数据集，强烈建议使用本机分区。

Athena MySQL 连接器执行谓词下推，以减少查询扫描的数据量。`LIMIT` 子句、简单谓词和复杂表达式将下推到连接器，以减少扫描数据量并缩短查询执行的运行时间。

LIMIT 子句

`LIMIT N` 语句用于减少查询扫描的数据量。`LIMIT N` 下推时，连接器仅向 Athena 返回 `N` 行。

Predicates

谓词是 SQL 查询的 `WHERE` 子句中的表达式，其评估结果为布尔值并根据多个条件筛选行。Athena MySQL 连接器可以组合这些表达式并将其直接推送到 MySQL，以增强功能并减少扫描的数据量。

以下 Athena MySQL 连接器运算符支持谓词下推：

- 布尔值：AND、OR、NOT
- 相等：EQUAL、NOT_EQUAL、LESS_THAN、LESS_THAN_OR_EQUAL、GREATER_THAN、GREATER_
- 算术：ADD、SUBTRACT、MULTIPLY、DIVIDE、MODULUS、NEGATE
- 其他：LIKE_PATTERN、IN

组合下推示例

要增强查询功能，组合下推类型，如以下示例所示：

```
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;
```

有关使用谓词下推来提高联合查询（包括 MySQL）性能的文章，请参阅 AWS 大数据博客中的在 Amazon Athena 中使用谓词下推改善联合查询。

传递查询

MySQL 连接器支持传递查询。传递查询使用表函数将完整查询下推到数据来源来执行查询。

要在 MySQL 中执行传递查询，可以使用以下语法：

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

以下示例查询将查询下推到 MySQL 中的数据来源。该查询选择了 `customer` 表中的所有列，将结果限制为 10。

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

许可证信息

使用此连接器，即表示您确认包含第三方组件（这些组件的列表可在此连接器的 pom.xml 文件中找到），并同意 GitHub.com 上的 LICENSE.txt 文件中提供的相应第三方许可证中的条款。

其他 资源

有关最新 JDBC 驱动程序版本信息，请参阅 GitHub.com 上适用于 MySQL 连接器的 pom.xml 文件。

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena Neptune 连接器

Amazon Neptune 是一项快速、可靠且完全托管式的图数据库服务，可帮助您轻松构建和运行适用于高度互连数据集的应用程序。Neptune 的核心是一个专门打造的高性能图数据库引擎，它经过优化，可存储数十亿个关系并能以毫秒级延迟进行图形查询。有关更多信息，请参阅《Neptune 用户指南》。

Amazon Athena Neptune 连接器使 Athena 能够与您的 Neptune 图数据库实例进行通信，从而使您的 Neptune 图数据可以通过 SQL 查询访问。

如果您在账户中启用了 Lake Formation，则您在 AWS Serverless Application Repository 中部署的 Athena 联合身份 Lambda 连接器的 IAM 角色必须在 Lake Formation 中具有 AWS Glue Data Catalog 的读取权限。

先决条件

使用 Neptune 连接器需要以下三个步骤。

- 设置 Neptune 集群
- 设置 AWS Glue Data Catalog
- 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。有关特定于部署 Neptune 连接器的更多详细信息，请参阅 GitHub.com 上的部署 Amazon Athena Neptune 连接器。

限制

目前，Neptune 连接器具有以下限制。

- 不支持投影列，包括主键（ID）。

设置 Neptune 集群

如果您现在没有 Amazon Neptune 集群，或者您要使用的这种集群中没有属性图数据集，则您必须设置一个。

确保您在托管 Neptune 集群的 VPC 中拥有互联网网关和 NAT 网关。Neptune 连接器 Lambda 函数使用的私有子网应该具有通过此 NAT 网关连接到互联网的路由。Neptune 连接器 Lambda 函数使用 NAT 网关与 AWS Glue 通信。

有关设置新 Neptune 集群并使用示例数据集加载它的说明，请参阅 GitHub.com 上的 Neptune 集群设置示例。

设置 AWS Glue Data Catalog

与传统的关系数据存储不同，Neptune 图形数据库 (DB) 节点和边缘不使用集架构。每个条目均可具有不同字段和数据类型。但是，由于 Neptune 连接器从 AWS Glue Data Catalog 检索元数据，您必须使用所需架构创建一个包含多个表的 AWS Glue 数据库。在创建 AWS Glue 数据库和表之后，该连接器可以填充表（可用于来自 Athena 的查询）的列表。

启用不区分大小写的列匹配

要使用正确的大小写来解析 Neptune 表中的列名称，即使列名称在 AWS Glue 全部为小写，也可以将 Neptune 连接器配置为不区分大小写的匹配。

要启用此功能，将 Neptune 连接器 Lambda 函数环境变量 `enable_caseinsensitivematch` 设置为 `true`。

为大小写的表名称指定 AWS Glue glabel 表参数

由于 AWS Glue 仅支持小写表名称，因此在为 Neptune 创建 AWS Glue 表并且 Neptune 表名称包含大小写时，需要指定 `glabel` AWS Glue 表参数。

在 AWS Glue 表定义中，包含 `glabel` 参数并将其值设置为采用原始大小写的表名称。这样可以确保在 AWS Glue 与 Neptune 表进行交互时保留正确的大小写。以下示例将 `glabel` 的值设置为表名称 `Airport`。

```
glabel = Airport
```

Table properties (3)

Key	Value
separatorChar	,
componenttype	vertex
glabel	Airport

有关设置 AWS Glue Data Catalog 以与 Neptune 配合使用的更多信息，请参阅 GitHub.com 上的设置 AWS Glue 目录。

Performance

Athena Neptune 连接器可执行谓词下推，以减少查询扫描的数据量。但是，使用主键结果的谓词会导致查询失败。`LIMIT` 子句会减少扫描的数据量，但如果未提供谓词，则预期要使用包含 `LIMIT` 子句的 `SELECT` 查询，来扫描至少 16 MB 的数据。Neptune 连接器能够灵活地应对并发造成的节流。

传递查询

Neptune 连接器支持传递查询。可以使用此功能在属性图上运行 Gremlin 查询，也可以对 RDF 数据运行 SPARQL 查询。

要使用 Neptune 创建传递查询，请使用以下语法：

```
SELECT * FROM TABLE(
        system.query(
            DATABASE => 'database_name',
            COLLECTION => 'collection_name',
            QUERY => 'query_string'
        ))
```

以下示例 Neptune 传递查询会筛选带有代码 ATL 的机场。

```
SELECT * FROM TABLE(
        system.query(
            DATABASE => 'graph-database',
            COLLECTION => 'airport',
            QUERY => 'g.V().has(\"airport\", \"code\", \"ATL\").valueMap()',
        ))
```

其他 资源

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena OpenSearch 连接器

OpenSearch Service

Amazon Athena OpenSearch 连接器使 Amazon Athena 能够与您的 OpenSearch 实例进行通信，以便您可以使用 SQL 查询您的 OpenSearch 数据。

> **Note**
>
> 由于存在已知问题，OpenSearch 连接器无法与 VPC 一起使用。

如果您在账户中启用了 Lake Formation，则您在 AWS Serverless Application Repository 中部署的 Athena 联合身份 Lambda 连接器的 IAM 角色必须在 Lake Formation 中具有 AWS Glue Data Catalog 的读取权限。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

术语

以下术语与 OpenSearch 连接器有关。

- 域 - 此连接器与 OpenSearch 实例的端点关联的名称。域也用作数据库名称。对于在 Amazon OpenSearch Service 中定义的 OpenSearch 实例，域是可自动发现的。对于其他实例，您必须提供域名和端点之间的映射。
- 索引 - 在 OpenSearch 实例中定义的数据库表。
- 映射 - 如果索引是数据库表，则映射是其架构（即其字段和属性的定义）。

  此连接器同时支持从 OpenSearch 实例和从 AWS Glue Data Catalog 检索元数据。如果该连接器找到与您的 OpenSearch 域名和索引名称相匹配的 AWS Glue 数据库和表，该连接器会尝试使用它们进行架构定义。我们建议您创建 AWS Glue 表，这样该表就是 OpenSearch 索引中所有字段的超集。
- 文档 - 数据库表中的记录。
- 数据流 - 基于时间的数据，由多个后备索引组成。有关更多信息，请参阅 OpenSearch 文档中的数据流和《Amazon OpenSearch Service 开发人员指南》中的数据流入门。

> **Note**
>
> 由于数据流索引在内部创建并通过开放搜索进行管理，因此连接器会从第一个可用索引中选择架构映射。因此，我们强烈建议将 AWS Glue 表设置为补充元数据源。有关更多信息，请参阅 在 AWS Glue 中设置数据库和表。

参数

使用本节中的 Lambda 环境变量来配置 OpenSearch 连接器。

- spill_bucket - 为超出 Lambda 函数限制的数据指定 Amazon S3 存储桶。

- spill_prefix -（可选）默认为指定 `spill_bucket`（称为 `athena-federation-spill`）中的子文件夹。我们建议您在此位置配置 Amazon S3 存储生命周期，以删除早于预定天数或小时数的溢出内容。
- spill_put_request_headers —（可选）用于溢出的 Amazon S3 `putObject` 请求的请求标头和值的 JSON 编码映射（例如 `{"x-amz-server-side-encryption" : "AES256"}`）。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。
- kms_key_id -（可选）默认情况下，将使用经过 AES-GCM 身份验证的加密模式和随机生成的密钥对溢出到 Amazon S3 的任何数据进行加密。要让您的 Lambda 函数使用 KMS 生成的更强的加密密钥（如 `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`），您可以指定 KMS 密钥 ID。
- disable_spill_encryption -（可选）当设置为 `True` 时，将禁用溢出加密。默认值为 `False`，此时将使用 AES-GCM 对溢出到 S3 的数据使用进行加密 - 使用随机生成的密钥，或者使用 KMS 生成密钥。禁用溢出加密可以提高性能，尤其是当您的溢出位置使用服务器端加密时。
- disable_glue -（可选）如果存在且设置为 true，则该连接器不会尝试从 AWS Glue 检索补充元数据。
- query_timeout_cluster – 生成并行扫描时使用的集群运行状况查询的超时时间（以秒为单位）。
- query_timeout_search – 用于从索引检索文档的搜索查询的超时时间（以秒为单位）。
- auto_discover_endpoint – 布尔型。默认为 `true`。当您使用 Amazon OpenSearch Service 并将此参数设置为 true 时，该连接器可以通过在 OpenSearch Service 上调用适当的描述或列出 API 操作来自动发现您的域和端点。对于任何其他类型的 OpenSearch 实例（例如，自托管型），您必须在 `domain_mapping` 变量中指定相关联的域端点。如果 `auto_discover_endpoint=true`，该连接器将使用 AWS 凭证对 OpenSearch Service 进行身份验证。否则，该连接器将通过 `domain_mapping` 变量从 AWS Secrets Manager 检索用户名和密码凭证。
- domain_mapping – 仅在将 `auto_discover_endpoint` 设置为 false 并定义域名与其关联端点之间的映射时使用。`domain_mapping` 变量可以容纳以下格式的多个 OpenSearch 端点：

```
domain1=endpoint1,domain2=endpoint2,domain3=endpoint3,...
```

  为了对 OpenSearch 端点进行身份验证，该连接器支持使用 `${SecretName}:` 格式以及从 AWS Secrets Manager 检索的用户名和密码注入的替换字符串。表达式末尾的冒号 (:) 用作与端点的其余部分的分隔符。

> **Important**
>
> 作为安全最佳实践，请勿在您的环境变量或连接字符串中使用硬编码凭证。有关将硬编码密钥移至 AWS Secrets Manager 的信息，请参阅《AWS Secrets Manager 用户指南》中的将硬编码密钥移至 AWS Secrets Manager。

以下示例使用 opensearch-creds 密钥。

```
movies=https://${opensearch-creds}:search-movies-ne...qu.us-east-1.es.amazonaws.com
```

在运行时，${opensearch-creds} 以用户名和密码的形式呈现，如以下示例所示。

```
movies=https://myusername@mypassword:search-movies-ne...qu.us-east-1.es.amazonaws.com
```

在 domain_mapping 参数中，每个域-端点对均可使用不同的密钥。密钥本身必须以 *user_name*@*password* 格式指定。虽然密码可能包含嵌入式 @ 标志，但第一个 @ 将用作来自 *user_name* 的分隔符。

还要注意，此连接器使用逗号 (,) 和等号 (=) 作为域-端点对的分隔符。因此，您不应在存储的密钥中的任何位置使用它们。

在 AWS Glue 中设置数据库和表

该连接器使用 AWS Glue 或 OpenSearch 获取元数据信息。您可以设置一个 AWS Glue 表，作为补充元数据定义源。要启用此功能，请定义与您要补充的源的域和索引相匹配的 AWS Glue 数据库和表。该连接器还可以通过检索指定索引的映射，来利用存储在 OpenSearch 实例中的元数据定义。

在 OpenSearch 中为数组定义元数据

OpenSearch 没有专用的数组数据类型。任何字段均可包含零个或多个值，只要它们属于相同数据类型即可。如果您要使用 OpenSearch 作为元数据定义源，则必须为与 Athena 配合使用的所有索引（用于将被视为列表或数组的字段）定义 _meta 属性。如果您未能完成此步骤，则查询将仅返回列表字段中的第一个元素。当您指定 _meta 属性时，字段名称应该完全符合嵌套 JSON 结构（例如，address.street，其中 street 是 address 结构内的嵌套字段）。

以下示例在 movies 表中定义了 actor 和 genre 列表。

```
PUT movies/_mapping
{
  "_meta": {
    "actor": "list",
    "genre": "list"
  }
}
```

数据类型

OpenSearch 连接器可从 AWS Glue 或 OpenSearch 实例提取元数据定义。该连接器使用下表中的映射将这些定义转换为 Apache Arrow 数据类型，包括下节中提到的要点。

OpenSearch	Apache Arrow	AWS Glue
text、keyword、binary	VARCHAR	字符串
long	BIGINT	bigint
scaled_float	BIGINT	SCALED_FLOAT(...)
整数	INT	int
short	SMALLINT	smallint
字节	TINYINT	tinyint
double	FLOAT8	double
float、half_float	FLOAT4	float
布尔值	BIT	布尔值
date、date_nanos	DATEMILLI	时间戳
JSON 结构	STRUCT	STRUCT
_meta（有关信息，请参阅 在 OpenSearch 中为数组定义元数据 一节。）	LIST	ARRAY

有关数据类型的注释

- 目前，该连接器仅支持上表中列出的 OpenSearch 和 AWS Glue 数据类型。
- `scaled_float` 是按固定双精度缩放系数缩放的浮点数，在 Apache Arrow 中以 `BIGINT` 形式表示。例如，在缩放系数为 100 的情况下，0.756 将四舍五入为 76。
- 要在 AWS Glue 中定义 `scaled_float`，您必须选择 `array` 列类型，并使用 SCALED_FLOAT(*scaling_factor*) 格式声明字段。

  以下示例有效：

  ```
  SCALED_FLOAT(10.51)
  SCALED_FLOAT(100)
  SCALED_FLOAT(100.0)
  ```

  以下示例无效：

  ```
  SCALED_FLOAT(10.)
  SCALED_FLOAT(.5)
  ```

- 从 `date_nanos` 转换为 `DATEMILLI` 时，纳秒将四舍五入到最接近的毫秒。`date` 和 `date_nanos` 的有效值包括但不限于以下格式：

  ```
  "2020-05-18T10:15:30.123456789"
  "2020-05-15T06:50:01.123Z"
  "2020-05-15T06:49:30.123-05:00"
  1589525370001 (epoch milliseconds)
  ```

- OpenSearch `binary` 是使用 `Base64` 编码的二进制值的字符串表示形式，将被转换为 `VARCHAR`。

运行 SQL 查询

以下是您可与此连接器配合使用的 DDL 查询的示例。在这些示例中，*function_name* 对应于您的 Lambda 函数的名称，*domain* 是您要查询的域的名称，*index* 是您的索引的名称。

```
SHOW DATABASES in `lambda:function_name`
```

```
SHOW TABLES in `lambda:function_name`.domain
```

```
DESCRIBE `lambda:function_name`.domain.index
```

Performance

Athena OpenSearch 连接器支持基于分片的并行扫描。该连接器使用从 OpenSearch 实例检索的集群运行状况信息来生成多个文档搜索查询请求。将为每个分片拆分这些请求并同时运行。

作为其文档搜索查询的组成部分，该连接器还将下推谓词。以下示例查询和谓词显示了该连接器如何使用谓词下推。

查询

```
SELECT * FROM "lambda:elasticsearch".movies.movies
WHERE year >= 1955 AND year <= 1962 OR year = 1996
```

谓词

```
(_exists_:year) AND year:([1955 TO 1962] OR 1996)
```

传递查询

OpenSearch 连接器支持传递查询，使用 Query DSL 语言。有关使用 Query DSL 进行查询的更多信息，请参阅 Elasticsearch 文档中的 Query DSL 或 OpenSearch 文档中的 Query DSL。

要在 OpenSearch 连接器中执行传递查询，请使用以下语法：

```
SELECT * FROM TABLE(
        system.query(
            schema => 'schema_name',
            index => 'index_name',
            query => "{query_string}"
        ))
```

以下 OpenSearch 示例传递查询，对 employee 架构 default 索引中处于活跃就业状态的员工进行了筛选。

```
SELECT * FROM TABLE(
        system.query(
            schema => 'default',
```

```
            index => 'employee',
            query => "{ ''bool'':{''filter'':{''term'':{''status'': ''active''}}}}"
        ))
```

其他 资源

- 有关使用 Amazon Athena OpenSearch 连接器在单一查询中查询 Amazon OpenSearch Service 和 Amazon S3 中的数据的文章，请参阅 AWS 大数据博客中的使用 Amazon Athena 的 SQL 查询 Amazon OpenSearch Service 中的数据。
- 有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena Oracle 连接器

适用于 Oracle 的 Amazon Athena 连接器使 Amazon Athena 能够对存储在 Oracle（在本地或在 Amazon EC2 或 Amazon RDS 上运行）中的数据运行 SQL 查询。您还可以使用该连接器查询存储在 Oracle Exadata 上的数据。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

限制

- 不支持写入 DDL 操作。
- 在多路复用器设置中，溢出桶和前缀在所有数据库实例之间共享。
- 任何相关的 Lambda 限制。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。

术语

以下术语与 Oracle 连接器有关。

- 数据库实例 - 部署在本地、Amazon EC2 或 Amazon RDS 上的任何数据库实例。
- 处理程序 — 访问您数据库实例的 Lambda 处理程序。处理程序可以用于元数据或数据记录。
- 元数据处理程序 — 从您的数据库实例中检索元数据的 Lambda 处理程序。

- 记录处理程序 — 从您的数据库实例中检索数据记录的 Lambda 处理程序。
- 复合处理程序 — 从您的数据库实例中检索元数据和数据记录的 Lambda 处理程序。
- 属性或参数 — 处理程序用来提取数据库信息的数据库属性。您可以将这些属性配置为 Lambda 环境变量。
- 连接字符串 — 用于建立数据库实例连接的文本字符串。
- 目录 — 向 Athena 注册的非 AWS Glue 目录，是 `connection_string` 属性的必要前缀。
- 多路复用处理程序 — 可以接受和使用多个数据库连接的 Lambda 处理程序。

参数

使用本节中的 Lambda 环境变量来配置 Oracle 连接器。

连接字符串

使用以下格式的 JDBC 连接字符串连接到数据库实例。

```
oracle://${jdbc_connection_string}
```

> **Note**
>
> 如果您的密码包含特殊字符（例如，`some.password`），则在将密码传递给连接字符串时将密码包含在双引号中（例如，`"some.password"`）。否则，可能会导致指定的 Oracle URL 无效错误。

使用多路复用处理程序

您可以使用多路复用器通过单个 Lambda 函数连接到多个数据库实例。按目录名称来路由请求。在 Lambda 中使用以下类。

处理程序	类
复合处理程序	`OracleMuxCompositeHandler`
元数据处理程序	`OracleMuxMetadataHandler`
记录处理程序	`OracleMuxRecordHandler`

多路复用处理程序参数

参数	描述
$*catalog*_connection_string	必需。数据库实例连接字符串。将 Athena 中使用的目录的名称作为环境变量前缀。例如，如果向 Athena 注册的目录是 myoraclecatalog ，则环境变量名称是 myoraclecatalog_connection_string 。
default	必需。默认连接字符串。当目录为 lambda:${ *AWS_LAMBDA_FUNCTION_NAME* } 时使用此字符串。

以下示例属性适用于支持两个数据库实例的 Oracle MUX Lambda 函数：oracle1（默认值）和 oracle2。

属性	值
default	oracle://jdbc:oracle:thin:${Test/RDS/Oracle1}@//oracle1.hostname:port/servicename
oracle_catalog1_connection_string	oracle://jdbc:oracle:thin:${Test/RDS/Oracle1}@//oracle1.hostname:port/servicename
oracle_catalog2_connection_string	oracle://jdbc:oracle:thin:${Test/RDS/Oracle2}@//oracle2.hostname:port/servicename

提供凭证

要在 JDBC 连接字符串中为数据库提供用户名和密码，可以使用连接字符串属性或 AWS Secrets Manager。

- 连接字符串 — 可以将用户名和密码指定为 JDBC 连接字符串中的属性。

> **Important**
>
> 作为安全最佳实践，请勿在您的环境变量或连接字符串中使用硬编码凭证。有关将硬编码密钥移至 AWS Secrets Manager 的信息，请参阅《AWS Secrets Manager 用户指南》中的将硬编码密钥移至 AWS Secrets Manager。

- AWS Secrets Manager - 要将 Athena 联合查询功能与 AWS Secrets Manager 配合使用，连接到您的 Lambda 函数的 VPC 应该拥有互联网访问权限或者 VPC 端点，以连接到 Secrets Manager。

  您可以将 AWS Secrets Manager 中的密钥名称放入您的 JDBC 连接字符串中。连接器将该密钥名称替换为来自 Secrets Manager 的 `username` 和 `password` 值。

  对于 Amazon RDS 数据库实例，将紧密集成这种支持。如果您使用 Amazon RDS，我们强烈建议您使用 AWS Secrets Manager 和凭证轮换。如果您的数据库不使用 Amazon RDS，请按以下格式将凭证存储为 JSON：

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

> **Note**
>
> 如果您的密码包含特殊字符（例如，`some.password`），则在将密码存储在 Secrets Manager 中时将密码包含在双引号中（例如，`"some.password"`）。否则，可能会导致指定的 Oracle URL 无效错误。

带有密钥名称的示例连接字符串

以下字符串带有密钥名称 `${Test/RDS/Oracle}`。

```
oracle://jdbc:oracle:thin:${Test/RDS/Oracle}@//hostname:port/servicename
```

该连接器使用该密钥名称来检索密钥，并提供用户名和密码，如以下示例所示。

```
oracle://jdbc:oracle:thin:username/password@//hostname:port/servicename
```

目前，Oracle 连接器可以识别 `UID` 和 `PWD` JDBC 属性。

使用单个连接处理程序

您可以使用以下单个连接元数据和记录处理程序连接到单个 Oracle 实例。

处理程序类型	类
复合处理程序	`OracleCompositeHandler`
元数据处理程序	`OracleMetadataHandler`
记录处理程序	`OracleRecordHandler`

单个连接处理程序参数

参数	描述
`default`	必需。默认连接字符串。
`IsFIPSEnabled`	可选。在启用 FIPS 模式时设置为 `true`。默认为 `false`。

单个连接处理程序支持一个数据库实例，并且必须提供 `default` 连接字符串参数。将忽略所有其他连接字符串。

该连接器支持基于 SSL 的与 Amazon RDS 实例的连接。支持仅限于传输层安全性协议（TLS）以及客户端对服务器的身份验证。在 Amazon RDS 中不支持相互身份验证。下表中的第二行显示了使用 SSL 的语法。

以下示例属性适用于 Lambda 函数支持的单个 Oracle 实例。

属性	值
`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=)))`

溢出参数

Lambda 开发工具包可以将数据溢出到 Amazon S3。由同一 Lambda 函数访问的所有数据库实例都会溢出到同一位置。

参数	描述
spill_bucket	必需。溢出桶名称。
spill_prefix	必需。溢出桶密钥前缀。
spill_put_request_headers	（可选）用于溢出的 Amazon S3 putObject 请求的请求标头和值的 JSON 编码映射（例如，{"x-amz-server-side-encryption" : "AES256"})。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。

数据类型支持

下表显示了适用于 JDBC、Oracle 和 Arrow 的相应数据类型。

JDBC	Oracle	Arrow
布尔值	布尔值	位
整数	不适用	Tiny
短型	smallint	Smallint
整数	整数	Int
长整型	bigint	Bigint
float	float4	Float4
Double	float8	Float8
Date	date	DateDay
Timestamp	时间戳	DateMilli

JDBC	Oracle	Arrow
String	文本	Varchar
字节	bytes	Varbinary
BigDecimal	numeric(p,s)	十进制
ARRAY	不适用（见注释）	列出

分区和拆分

分区用于确定如何为该连接器生成拆分。Athena 将构建一个 `varchar` 类型的合成列，它将展示表的分区方案，以帮助该连接器生成拆分。该连接器不会修改实际的表定义。

Performance

Oracle 支持本机分区。Athena Oracle 连接器可从这些分区并行检索数据。如果您想查询具有统一分区分布的超大型数据集，强烈建议使用本机分区。选择列的子集可以显著减少查询运行时及扫描的数据。Oracle 连接器能够灵活地应对并发造成的节流。但是，查询运行时往往很长。

Athena Oracle 连接器可执行谓词下推，以减少查询扫描的数据量。简单谓词和复杂表达式将下推到连接器，以减少扫描的数据量并缩短查询执行的运行时间。

Predicates

谓词是 SQL 查询的 `WHERE` 子句中的表达式，其评估结果为布尔值并根据多个条件筛选行。Athena Oracle 连接器可以组合这些表达式并将其直接推送到 Oracle，以增强功能并减少扫描的数据量。

以下 Athena Oracle 连接器运算符支持谓词下推：

- 布尔值：AND、OR、NOT
- 相等：EQUAL、NOT_EQUAL、LESS_THAN、LESS_THAN_OR_EQUAL、GREATER_THAN、GREATER_
- 算术：ADD、SUBTRACT、MULTIPLY、DIVIDE、NEGATE
- 其他：LIKE_PATTERN、IN

组合下推示例

要增强查询功能，组合下推类型，如以下示例所示：

```
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%');
```

传递查询

Oracle 连接器支持传递查询。传递查询使用表函数将完整查询下推到数据来源来执行查询。

要在 Oracle 中执行传递查询，可以使用以下语法：

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

以下示例查询将查询下推到 Oracle 中的数据来源。该查询会选择 customer 表中的所有列。

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

许可证信息

使用此连接器，即表示您确认包含第三方组件（这些组件的列表可在此连接器的 pom.xml 文件中找到），并同意 GitHub.com 上的 LICENSE.txt 文件中提供的相应第三方许可证中的条款。

其他 资源

有关最新 JDBC 驱动程序版本信息，请参阅 GitHub.com 上适用于 Oracle 连接器的 pom.xml 文件。

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena PostgreSQL 连接器

Amazon Athena PostgreSQL 连接器使 Athena 可以访问 PostgreSQL 数据库。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

限制

- 不支持写入 DDL 操作。
- 在多路复用器设置中，溢出桶和前缀在所有数据库实例之间共享。
- 任何相关的 Lambda 限制。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。

术语

以下术语和概念与 PostgreSQL 连接器有关。

- 数据库实例 - 部署在本地、Amazon EC2 或 Amazon RDS 上的任何数据库实例。
- 处理程序 — 访问您数据库实例的 Lambda 处理程序。处理程序可以用于元数据或数据记录。
- 元数据处理程序 — 从您的数据库实例中检索元数据的 Lambda 处理程序。
- 记录处理程序 — 从您的数据库实例中检索数据记录的 Lambda 处理程序。
- 复合处理程序 — 从您的数据库实例中检索元数据和数据记录的 Lambda 处理程序。
- 属性或参数 — 处理程序用来提取数据库信息的数据库属性。您可以将这些属性配置为 Lambda 环境变量。
- 连接字符串 — 用于建立数据库实例连接的文本字符串。
- 目录 — 向 Athena 注册的非 AWS Glue 目录，是 `connection_string` 属性的必要前缀。
- 多路复用处理程序 — 可以接受和使用多个数据库连接的 Lambda 处理程序。

参数

使用本节中的 Lambda 环境变量来配置 PostgreSQL 连接器。

连接字符串

使用以下格式的 JDBC 连接字符串连接到数据库实例。

```
postgres://${jdbc_connection_string}
```

使用多路复用处理程序

您可以使用多路复用器通过单个 Lambda 函数连接到多个数据库实例。按目录名称来路由请求。在 Lambda 中使用以下类。

处理程序	类
复合处理程序	PostGreSqlMuxCompositeHandler
元数据处理程序	PostGreSqlMuxMetadataHandler
记录处理程序	PostGreSqlMuxRecordHandler

多路复用处理程序参数

参数	描述
$*catalog*_connection_string	必需。数据库实例连接字符串。将 Athena 中使用的目录的名称作为环境变量前缀。例如，如果向 Athena 注册的目录是 mypostgrescatalog ，则环境变量名称是 mypostgrescatalog_connection_string 。
default	必需。默认连接字符串。当目录为 lambda:${ *AWS_LAMBDA_FUNCTION_NAME* } 时使用此字符串。

以下示例属性适用于支持两个数据库实例的 PostGreSql MUX Lambda 函数：postgres1（默认）和 postgres2。

属性	值
default	postgres://jdbc:postgresql://postgres1.host:5432/default?${Test/RDS/PostGres1}
postgres_catalog1_	postgres://jdbc:postgresql://postgres1.host:5432/default?${Test/RDS/PostGres1}

属性	值
connection_string	
postgres_catalog2_connection_string	postgres://jdbc:postgresql://postgres2.host:5432/default?user=sample&password=sample

提供凭证

要在 JDBC 连接字符串中为数据库提供用户名和密码，可以使用连接字符串属性或 AWS Secrets Manager。

- 连接字符串 — 可以将用户名和密码指定为 JDBC 连接字符串中的属性。

  > **Important**
  >
  > 作为安全最佳实践，请勿在您的环境变量或连接字符串中使用硬编码凭证。有关将硬编码密钥移至 AWS Secrets Manager 的信息，请参阅《AWS Secrets Manager 用户指南》中的将硬编码密钥移至 AWS Secrets Manager。

- AWS Secrets Manager - 要将 Athena 联合查询功能与 AWS Secrets Manager 配合使用，连接到您的 Lambda 函数的 VPC 应该拥有互联网访问权限或者 VPC 端点，以连接到 Secrets Manager。

  您可以将 AWS Secrets Manager 中的密钥名称放入您的 JDBC 连接字符串中。连接器将该密钥名称替换为来自 Secrets Manager 的 `username` 和 `password` 值。

  对于 Amazon RDS 数据库实例，将紧密集成这种支持。如果您使用 Amazon RDS，我们强烈建议您使用 AWS Secrets Manager 和凭证轮换。如果您的数据库不使用 Amazon RDS，请按以下格式将凭证存储为 JSON：

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

带有密钥名称的示例连接字符串

以下字符串带有密钥名称 `${Test/RDS/PostGres1}`。

```
postgres://jdbc:postgresql://postgres1.host:5432/default?...&${Test/RDS/PostGres1}&...
```

该连接器使用该密钥名称来检索秘密，并提供用户名和密码，如以下示例所示。

```
postgres://jdbc:postgresql://postgres1.host:5432/
default?...&user=sample2&password=sample2&...
```

目前，PostgreSQL 连接器可以识别 `user` 和 `password` JDBC 属性。

启用 SSL

要在 PostgreSQL 连接中支持 SSL，请在连接字符串中附加以下内容：

```
&sslmode=verify-ca&sslfactory=org.postgresql.ssl.DefaultJavaSSLFactory
```

示例

以下示例连接字符串不使用 SSL。

```
postgres://jdbc:postgresql://example-asdf-aurora-postgres-endpoint:5432/asdf?
user=someuser&password=somepassword
```

要启用 SSL，如下所示修改字符串。

```
postgres://jdbc:postgresql://example-asdf-aurora-postgres-
endpoint:5432/asdf?user=someuser&password=somepassword&sslmode=verify-
ca&sslfactory=org.postgresql.ssl.DefaultJavaSSLFactory
```

使用单个连接处理程序

您可以使用以下单个连接元数据和记录处理程序连接到单个 PostgreSQL 实例。

处理程序类型	类
复合处理程序	`PostGreSqlCompositeHandler`
元数据处理程序	`PostGreSqlMetadataHandler`
记录处理程序	`PostGreSqlRecordHandler`

单个连接处理程序参数

参数	描述
default	必需。默认连接字符串。

单个连接处理程序支持一个数据库实例，并且必须提供 default 连接字符串参数。将忽略所有其他连接字符串。

以下示例属性适用于 Lambda 函数支持的单个 PostgreSQL 实例。

属性	值
default	postgres://jdbc:postgresql://postgres1.host:5432/default?secret=${Test/RDS/PostgreSQL1}

溢出参数

Lambda 开发工具包可以将数据溢出到 Amazon S3。由同一 Lambda 函数访问的所有数据库实例都会溢出到同一位置。

参数	描述
spill_bucket	必需。溢出桶名称。
spill_prefix	必需。溢出桶密钥前缀。
spill_put_request_headers	（可选）用于溢出的 Amazon S3 putObject 请求的请求标头和值的 JSON 编码映射（例如，{"x-amz-server-side-encryption" : "AES256"})。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。

数据类型支持

下表显示了适用于 JDBC、PostGreSQL 和 Arrow 的相应数据类型。

JDBC	PostGreSQL	Arrow
布尔值	布尔值	位
整数	不适用	Tiny
短型	smallint	Smallint
整数	整数	Int
长整型	bigint	Bigint
float	float4	Float4
Double	float8	Float8
Date	date	DateDay
Timestamp	时间戳	DateMilli
String	文本	Varchar
字节	bytes	Varbinary
BigDecimal	numeric(p,s)	十进制
ARRAY	不适用（见注释）	列出

> **Note**
>
> PostgreSQL 连接器支持该 ARRAY 类型，但具有以下限制：不支持多维数组（`<data_type>[][]` 或嵌套数组）。包含不受支持的 ARRAY 数据类型的列将被转换为字符串元素数组 (`array<varchar>`)。

分区和拆分

分区用于确定如何为该连接器生成拆分。Athena 将构建一个 `varchar` 类型的合成列，它将展示表的分区方案，以帮助该连接器生成拆分。该连接器不会修改实际的表定义。

Performance

PostgreSQL 支持本机分区。Athena PostgreSQL 连接器可从这些分区并行检索数据。如果您想查询具有统一分区分布的超大型数据集，强烈建议使用本机分区。

Athena PostgreSQL 连接器执行谓词下推，以减少查询扫描的数据量。LIMIT 子句、简单谓词和复杂表达式将下推到连接器，以减少扫描数据量并缩短查询执行的运行时间。但是，选择一部分列有时会导致查询执行的运行时间延长。

LIMIT 子句

LIMIT N 语句用于减少查询扫描的数据量。LIMIT N 下推时，连接器仅向 Athena 返回 N 行。

Predicates

谓词是 SQL 查询的 WHERE 子句中的表达式，其评估结果为布尔值并根据多个条件筛选行。Athena PostgreSQL 连接器可以组合这些表达式并将其直接推送到 PostgreSQL，以增强功能并减少扫描的数据量。

以下 Athena PostgreSQL 连接器运算符支持谓词下推：

- 布尔值：AND、OR、NOT
- 相等：EQUAL、NOT_EQUAL、LESS_THAN、LESS_THAN_OR_EQUAL、GREATER_THAN、GREATER_
- 算术：ADD、SUBTRACT、MULTIPLY、DIVIDE、MODULUS、NEGATE
- 其他：LIKE_PATTERN、IN

组合下推示例

要增强查询功能，组合下推类型，如以下示例所示：

```
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;
```

传递查询

PostgreSQL 连接器支持传递查询。传递查询使用表函数将完整查询下推到数据来源来执行查询。

要在 PostgreSQL 中执行传递查询，可以使用以下语法：

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

以下示例查询将查询下推到 PostgreSQL 中的数据来源。该查询选择了 customer 表中的所有列，将结果限制为 10。

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

其他 资源

有关最新 JDBC 驱动程序版本信息，请参阅 GitHub.com 上适用于 PostgreSQL 连接器的 pom.xml 文件。

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena Redis 连接器

Amazon Redis 连接器使 Amazon Athena 可以与 Redis 实例通信，以便您可以使用 SQL 查询 Redis 数据。您可以使用 AWS Glue Data Catalog 将 Redis 键值对映射到虚拟表。

与传统的关系数据存储不同，Redis 没有表或列的概念。相反，Redis 提供了键值访问模式，其中，键本质上是一个 string，值是 string、z-set 或 hmap。

您可以使用 AWS Glue Data Catalog 创建架构和配置虚拟表。特殊的表属性告诉 Athena Redis 连接器如何将 Redis 键和值映射到表中。有关更多信息，请参阅本文后面的在 AWS Glue 中设置数据库和表。

如果您在账户中启用了 Lake Formation，则您在 AWS Serverless Application Repository 中部署的 Athena 联合身份 Lambda 连接器的 IAM 角色必须在 Lake Formation 中具有 AWS Glue Data Catalog 的读取权限。

Amazon Athena Redis 连接器支持适用于 Redis 的 Amazon MemoryDB 和 Amazon ElastiCache for Redis。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。
- 使用此连接器前，请先设置 VPC 和安全组。有关更多信息，请参阅 为数据源连接器创建 VPC。

参数

使用本节中的 Lambda 环境变量配置 Redis 连接器。

- spill_bucket - 为超出 Lambda 函数限制的数据指定 Amazon S3 存储桶。
- spill_prefix - （可选）默认为指定 `spill_bucket`（称为 `athena-federation-spill`）中的子文件夹。我们建议您在此位置配置 Amazon S3 存储生命周期，以删除早于预定天数或小时数的溢出内容。
- spill_put_request_headers —（可选）用于溢出的 Amazon S3 `putObject` 请求的请求标头和值的 JSON 编码映射（例如 `{"x-amz-server-side-encryption" : "AES256"}`）。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。
- kms_key_id - （可选）默认情况下，将使用经过 AES-GCM 身份验证的加密模式和随机生成的密钥对溢出到 Amazon S3 的任何数据进行加密。要让您的 Lambda 函数使用 KMS 生成的更强的加密密钥（如 `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`），您可以指定 KMS 密钥 ID。
- disable_spill_encryption - （可选）当设置为 `True` 时，将禁用溢出加密。默认值为 `False`，此时将使用 AES-GCM 对溢出到 S3 的数据使用进行加密 - 使用随机生成的密钥，或者使用 KMS 生成密钥。禁用溢出加密可以提高性能，尤其是当您的溢出位置使用服务器端加密时。
- glue_catalog –（可选）使用此选项指定跨账户 AWS Glue 目录。默认情况下，该连接器将尝试从其自己的 AWS Glue 账户中获取元数据。

在 AWS Glue 中设置数据库和表

要将 AWS Glue 表与 Redis 配合使用，可以在表上设置以下表属性：`redis-endpoint`、`redis-value-type`，以及 `redis-keys-zset` 或 `redis-key-prefix`。

此外，包含 Redis 表的任何 AWS Glue 数据库必须在数据库的 URI 属性中有 `redis-db-flag`。要设置 `redis-db-flag` URI 属性，请使用 AWS Glue 控制台编辑数据库。

以下列表描述了表属性。

- redis-endpoint — （必选）包含此表数据的 Redis 服务器的`###:##:##`（例如 `athena-federation-demo.cache.amazonaws.com:6379`）。或者，您可以通过将 ${*Secret_Name*} 用作表属性值，来将端点或部分端点存储在 AWS Secrets Manager。

> **Note**
>
> 要将 Athena 联合查询功能与 AWS Secrets Manager 配合使用，连接到您的 Lambda 函数的 VPC 应该拥有互联网访问权限或者 VPC 端点，以连接到 Secrets Manager。

- redis-keys-zset — （必选，如果未使用 `redis-key-prefix`）以逗号分隔的密钥列表，值为 zset（例如 `active-orders,pending-orders`）。zset 中的每个值都被视为密钥（表的一部分）。必须设置 `redis-keys-zset` 属性或 `redis-key-prefix` 属性。
- redis-key-prefix — （必选，如果未使用 `redis-keys-zset`）以逗号分隔的键前缀列表，用于扫描表中的值（例如 `accounts-*,acct-`）。必须设置 `redis-key-prefix` 属性或 `redis-keys-zset` 属性。
- redis-value-type — （必选）定义由 `redis-key-prefix` 或 `redis-keys-zset` 定义的键的值如何映射到您的表。文本映射到单个列。zset 也映射到单个列，但每个键可以存储许多行。哈希使每个键成为包含多个列的行（例如，哈希、文本或 zset）。
- redis-ssl-flag — （可选）`True` 时，创建一个使用 SSL/TLS 的 Redis 连接。默认为 `False`。
- redis-cluster-flag — （可选）`True` 时，启用对集群 Redis 实例的支持。默认为 `False`。
- redis-db-number — （可选）仅适用于独立的非集群实例。）将此数字（例如 1、2 或 3）设置为从非默认 Redis 数据库读取。默认为 Redis 逻辑数据库 0。此数字不是指 Athena 或 AWS Glue 中的数据库，而是指 Redis 逻辑数据库。有关更多信息，请参阅 Redis 文档中的 SELECT 索引。

数据类型

Redis 连接器支持以下数据类型。不支持 Redis 流。

- String
- 哈希
- 排序集（ZSet）

所有 Redis 值均作为 `string` 数据类型检索。然后，根据您的表在 AWS Glue Data Catalog 中的定义方式，将这些值转换为以下 Apache Arrow 数据类型之一。

AWS Glue 数据类型	Apache Arrow 数据类型
int	INT
字符串	VARCHAR
bigint	BIGINT
double	FLOAT8
float	FLOAT4
smallint	SMALLINT
tinyint	TINYINT
布尔值	BIT
binary	VARBINARY

所需权限

有关此连接器所需 IAM policy 的完整详细信息，请查看 athena-redis.yaml 文件的 `Policies` 部分。以下列表汇总了所需的权限。

- Amazon S3 写入权限 – 连接器需要对 Amazon S3 中的位置具有写入权限，以溢出大型查询的结果。
- Athena GetQueryExecution – 当上游 Athena 查询终止时，该连接器将使用此权限快速失败。
- AWS Glue Data Catalog — Redis 连接器需要针对 AWS Glue Data Catalog 的只读访问权限，以获取架构信息。
- CloudWatch Logs – 该连接器需要针对 CloudWatch Logs 的访问权限，以存储日志。
- AWS Secrets Manager读取访问权限 — 如果您选择在 Secrets Manager 中存储 Redis 端点详细信息，则必须授予连接器访问这些密钥的权限。
- VPC 访问 — 连接器需要能够连接和分离您 VPC 的接口，以便连接到 VPC 并与您的 Redis 实例通信。

Performance

Athena Redis 连接器尝试根据您定义的表类型（例如，zset 密钥或前缀密钥）针对您的 Redis 实例并行查询。

Athena Redis 连接器可执行谓词下推，以减少查询扫描的数据量。但是，包含谓词的主键结果查询会超时失败。`LIMIT` 子句会减少扫描的数据量，但如果未提供谓词，则预期要使用包含 `LIMIT` 子句的 `SELECT` 查询，来扫描至少 16 MB 的数据。Redis 连接器能够灵活地应对并发造成的节流。

许可证信息

Amazon Athena Redis 连接器项目已根据 Apache-2.0 许可证获得许可。

其他 资源

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena Redshift 连接器

Amazon Athena Redshift 连接器使 Amazon Athena 能够访问您的 Amazon Redshift 和 Amazon Redshight Serverless 数据库，包括 Redshift Serverless 视图。您可以使用本页所述的 JDBC 连接字符串配置设置连接到任一服务。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

限制

- 不支持写入 DDL 操作。
- 在多路复用器设置中，溢出桶和前缀在所有数据库实例之间共享。
- 任何相关的 Lambda 限制。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。
- 由于 Redshift 不支持外部分区，因此每次都会检索查询指定的所有数据。

术语

以下术语与 Redshift 连接器有关。

- 数据库实例 — 部署在本地、Amazon EC2 或 Amazon RDS 上的任何数据库实例。
- 处理程序 — 访问您数据库实例的 Lambda 处理程序。处理程序可以用于元数据或数据记录。
- 元数据处理程序 — 从您的数据库实例中检索元数据的 Lambda 处理程序。
- 记录处理程序 — 从您的数据库实例中检索数据记录的 Lambda 处理程序。
- 复合处理程序 — 从您的数据库实例中检索元数据和数据记录的 Lambda 处理程序。
- 属性或参数 — 处理程序用来提取数据库信息的数据库属性。您可以将这些属性配置为 Lambda 环境变量。
- 连接字符串 — 用于建立数据库实例连接的文本字符串。
- 目录 — 向 Athena 注册的非 AWS Glue 目录，是 `connection_string` 属性的必要前缀。
- 多路复用处理程序 — 可以接受和使用多个数据库连接的 Lambda 处理程序。

参数

使用本节中的 Lambda 环境变量来配置 Redshift 连接器。

连接字符串

使用以下格式的 JDBC 连接字符串连接到数据库实例。

```
redshift://${jdbc_connection_string}
```

使用多路复用处理程序

您可以使用多路复用器通过单个 Lambda 函数连接到多个数据库实例。按目录名称来路由请求。在 Lambda 中使用以下类。

处理程序	类
复合处理程序	`RedshiftMuxCompositeHandler`
元数据处理程序	`RedshiftMuxMetadataHandler`
记录处理程序	`RedshiftMuxRecordHandler`

多路复用处理程序参数

参数	描述
$*catalog*_connection_string	必需。数据库实例连接字符串。将 Athena 中使用的目录的名称作为环境变量前缀。例如，如果向 Athena 注册的目录是 myredshiftcatalog ，则环境变量名称是 myredshiftcatalog_connection_string 。
default	必需。默认连接字符串。目录为 lambda:${ *AWS_LAMBDA_FUNCTION_NAME* } 时使用此字符串。

以下示例属性适用于支持两个数据库实例的 Redshift MUX Lambda 函数：redshift1（默认）和 redshift2。

属性	值
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

提供凭证

要在 JDBC 连接字符串中为数据库提供用户名和密码，可以使用连接字符串属性或 AWS Secrets Manager。

- 连接字符串 — 可以将用户名和密码指定为 JDBC 连接字符串中的属性。

> **Important**
>
> 作为安全最佳实践，请勿在您的环境变量或连接字符串中使用硬编码凭证。有关将硬编码密钥移至 AWS Secrets Manager 的信息，请参阅《AWS Secrets Manager 用户指南》中的将硬编码密钥移至 AWS Secrets Manager。

- AWS Secrets Manager - 要将 Athena 联合查询功能与 AWS Secrets Manager 配合使用，连接到您的 Lambda 函数的 VPC 应该拥有互联网访问权限或者 VPC 端点，以连接到 Secrets Manager。

  您可以将 AWS Secrets Manager 中的密钥名称放入您的 JDBC 连接字符串中。连接器将该密钥名称替换为来自 Secrets Manager 的 `username` 和 `password` 值。

  对于 Amazon RDS 数据库实例，将紧密集成这种支持。如果您使用 Amazon RDS，我们强烈建议您使用 AWS Secrets Manager 和凭证轮换。如果您的数据库不使用 Amazon RDS，请按以下格式将凭证存储为 JSON：

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

带有密钥名称的示例连接字符串

以下字符串带有密钥名称 ${Test/RDS/Redshift1}。

```
redshift://jdbc:redshift://redshift1.host:3306/default?...&${Test/RDS/Redshift1}&...
```

该连接器使用该密钥名称来检索密钥，并提供用户名和密码，如以下示例所示。

```
redshift://jdbc:redshift://redshift1.host:3306/
default?...&user=sample2&password=sample2&...
```

目前，Redshift 连接器可以识别 `user` 和 `password` JDBC 属性。

溢出参数

Lambda 开发工具包可以将数据溢出到 Amazon S3。由同一 Lambda 函数访问的所有数据库实例都会溢出到同一位置。

参数	描述
spill_bucket	必需。溢出桶名称。
spill_prefix	必需。溢出桶密钥前缀。
spill_put_request_headers	（可选）用于溢出的 Amazon S3 putObject 请求的请求标头和值的 JSON 编码映射（例如，{"x-amz-server-side-encryption" : "AES256"})。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。

数据类型支持

下表显示了适用于 JDBC 和 Apache Arrow 的相应数据类型。

JDBC	Arrow
布尔值	位
整数	Tiny
短型	Smallint
整数	Int
长整型	Bigint
float	Float4
Double	Float8
Date	DateDay
Timestamp	DateMilli
String	Varchar
字节	Varbinary

JDBC	Arrow
BigDecimal	十进制
ARRAY	列出

分区和拆分

Redshift 不支持外部分区。有关性能相关问题的信息，请参阅 Performance。

Performance

Athena Redshift 连接器执行谓词下推，以减少查询扫描的数据。LIMIT 子句、ORDER BY 子句、简单谓词和复杂表达式将下推到连接器，以减少扫描数据量并缩短查询执行的运行时间。但是，选择一部分列有时会导致查询执行的运行时间延长。当您同时运行多个查询时，Amazon Redshift 特别容易受到查询执行速度减慢的影响。

LIMIT 子句

LIMIT N 语句用于减少查询扫描的数据量。LIMIT N 下推时，连接器仅向 Athena 返回 N 行。

前 N 个查询

前 N 个查询用于指定结果集的顺序以及返回的行数限值。您可以使用此类查询确定数据集的前 N 个最大值或前 N 个最小值。前 N 个查询下推时，连接器仅向 Athena 返回 N 个已排序行。

Predicates

谓词是 SQL 查询的 WHERE 子句中的表达式，其评估结果为布尔值并根据多个条件筛选行。Athena Redshift 连接器可以组合这些表达式并将其直接推送到 Redshift，以增强功能并减少扫描的数据量。

以下 Athena Redshift 连接器运算符支持谓词下推：

- 布尔值：AND、OR、NOT
- 相等：EQUAL、NOT_EQUAL、LESS_THAN、LESS_THAN_OR_EQUAL、GREATER_THAN、GREATER_
- 算术：ADD、SUBTRACT、MULTIPLY、DIVIDE、MODULUS、NEGATE
- 其他：LIKE_PATTERN、IN

组合下推示例

要增强查询功能，组合下推类型，如以下示例所示：

```
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;
```

有关使用谓词下推来提高联合查询（包括 Amazon Redshift）性能的文章，请参阅 AWS 大数据博客中的在 Amazon Athena 中使用谓词下推改善联合查询。

传递查询

Redshift 连接器支持传递查询。传递查询使用表函数将完整查询下推到数据来源来执行查询。

要在 Redshift 中执行传递查询，可以使用以下语法：

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

以下示例查询将查询下推到 Redshift 中的数据来源。该查询选择了 `customer` 表中的所有列，将结果限制为 10。

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

其他 资源

有关最新 JDBC 驱动程序版本信息，请参见 GitHub.com 上适用于 Redshift 连接器的 pom.xml 文件。

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena SAP HANA 连接器

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

限制

- 不支持写入 DDL 操作。
- 在多路复用器设置中，溢出桶和前缀在所有数据库实例之间共享。
- 任何相关的 Lambda 限制。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。
- 在 SAP HANA 中，当对象名称存储在 SAP HANA 数据库中时，它们会转换为大写形式。但是，由于引号中的名称区分大小写，因此两个表能分别以小写和大写形式具有相同的名称（例如 `EMPLOYEE` 和 `employee`）。

  在 Athena 联合查询中，架构表名称以小写形式提供给 Lambda 函数。要解决此问题，您可以提供 `@schemaCase` 查询提示，用于从名称区分大小写的表中检索数据。以下是两个带有查询提示的示例查询。

```
SELECT *
FROM "lambda:saphanaconnector".SYSTEM."MY_TABLE@schemaCase=upper&tableCase=upper"
```

```
SELECT *
FROM "lambda:saphanaconnector".SYSTEM."MY_TABLE@schemaCase=upper&tableCase=lower"
```

术语

以下术语与 SAP HANA 连接器有关。

- 数据库实例 — 部署在本地、Amazon EC2 或 Amazon RDS 上的任何数据库实例。
- 处理程序 — 访问您数据库实例的 Lambda 处理程序。处理程序可以用于元数据或数据记录。
- 元数据处理程序 — 从您的数据库实例中检索元数据的 Lambda 处理程序。

- 记录处理程序 — 从您的数据库实例中检索数据记录的 Lambda 处理程序。
- 复合处理程序 — 从您的数据库实例中检索元数据和数据记录的 Lambda 处理程序。
- 属性或参数 — 处理程序用来提取数据库信息的数据库属性。您可以将这些属性配置为 Lambda 环境变量。
- 连接字符串 — 用于建立数据库实例连接的文本字符串。
- 目录 — 向 Athena 注册的非 AWS Glue 目录，是 `connection_string` 属性的必要前缀。
- 多路复用处理程序 — 可以接受和使用多个数据库连接的 Lambda 处理程序。

参数

使用本节中的 Lambda 环境变量来配置 SAP HANA 连接器。

连接字符串

使用以下格式的 JDBC 连接字符串连接到数据库实例。

```
saphana://${jdbc_connection_string}
```

使用多路复用处理程序

您可以使用多路复用器通过单个 Lambda 函数连接到多个数据库实例。按目录名称来路由请求。在 Lambda 中使用以下类。

处理程序	类
复合处理程序	`SaphanaMuxCompositeHandler`
元数据处理程序	`SaphanaMuxMetadataHandler`
记录处理程序	`SaphanaMuxRecordHandler`

多路复用处理程序参数

参数	描述
`$catalog_connection_string`	必需。数据库实例连接字符串。将 Athena 中使用的目录的名称作为环境变量前缀。例如，如果向 Athena 注册的目录是 `mysaphana`

参数	描述
	`catalog` ，则环境变量名称是 `mysaphanacatalog_connection_string` 。
`default`	必需。默认连接字符串。目录为 `lambda:${` *AWS_LAMBDA_FUNCTION_NAME* `}` 时使用此字符串。

以下示例属性适用于支持两个数据库实例的 Saphana MUX Lambda 函数：`saphana1`（默认）和 `saphana2`。

属性	值
`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`

提供凭证

要在 JDBC 连接字符串中为数据库提供用户名和密码，可以使用连接字符串属性或 AWS Secrets Manager。

- 连接字符串 — 可以将用户名和密码指定为 JDBC 连接字符串中的属性。

> **Important**
>
> 作为安全最佳实践，请勿在您的环境变量或连接字符串中使用硬编码凭证。有关将硬编码密钥移至 AWS Secrets Manager 的信息，请参阅《AWS Secrets Manager 用户指南》中的将硬编码密钥移至 AWS Secrets Manager。

- AWS Secrets Manager - 要将 Athena 联合查询功能与 AWS Secrets Manager 配合使用，连接到您的 Lambda 函数的 VPC 应该拥有互联网访问权限或者 VPC 端点，以连接到 Secrets Manager。

  您可以将 AWS Secrets Manager 中的密钥名称放入您的 JDBC 连接字符串中。连接器将该密钥名称替换为来自 Secrets Manager 的 `username` 和 `password` 值。

  对于 Amazon RDS 数据库实例，将紧密集成这种支持。如果您使用 Amazon RDS，我们强烈建议您使用 AWS Secrets Manager 和凭证轮换。如果您的数据库不使用 Amazon RDS，请按以下格式将凭证存储为 JSON：

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

带有密钥名称的示例连接字符串

以下字符串带有密钥名称 `${Test/RDS/Saphana1}`。

```
saphana://jdbc:sap://saphana1.host:port/?${Test/RDS/Saphana1}&...
```

该连接器使用该密钥名称来检索密钥，并提供用户名和密码，如以下示例所示。

```
saphana://jdbc:sap://saphana1.host:port/?user=sample2&password=sample2&...
```

目前，SAP HANA 连接器可以识别 `user` 和 `password` JDBC 属性。

使用单个连接处理程序

您可以使用以下单个连接元数据和记录处理程序连接到单个 SAP HANA 实例。

处理程序类型	类
复合处理程序	`SaphanaCompositeHandler`
元数据处理程序	`SaphanaMetadataHandler`
记录处理程序	`SaphanaRecordHandler`

单个连接处理程序参数

参数	描述
default	必需。默认连接字符串。

单个连接处理程序支持一个数据库实例，并且必须提供 default 连接字符串参数。将忽略所有其他连接字符串。

以下示例属性适用于 Lambda 函数支持的单个 SAP HANA 实例。

属性	值
default	saphana://jdbc:sap://saphana1.host:port/?secret=Test/RDS/Saphana1

溢出参数

Lambda 开发工具包可以将数据溢出到 Amazon S3。由同一 Lambda 函数访问的所有数据库实例都会溢出到同一位置。

参数	描述
spill_bucket	必需。溢出桶名称。
spill_prefix	必需。溢出桶密钥前缀。
spill_put_request_headers	（可选）用于溢出的 Amazon S3 putObject 请求的请求标头和值的 JSON 编码映射（例如，{"x-amz-server-side-encryption" : "AES256"})。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。

数据类型支持

下表显示了适用于 JDBC 和 Apache Arrow 的相应数据类型。

JDBC	Arrow
布尔值	位
整数	Tiny
短型	Smallint
整数	Int
长整型	Bigint
float	Float4
Double	Float8
Date	DateDay
Timestamp	DateMilli
String	Varchar
字节	Varbinary
BigDecimal	十进制
ARRAY	列出

数据类型转换

除了 JDBC 到 Arrow 的转换外，连接器还执行某些其他转换，以使 SAP HANA 源和 Athena 数据类型兼容。这些转换有助于确保成功执行查询。下表显示了这些转换。

源数据类型 (SAP HANA)	转换后的数据类型 (Athena)
DECIMAL	BIGINT
INTEGER	INT
DATE	DATEDAY

源数据类型 (SAP HANA)	转换后的数据类型 (Athena)
TIMESTAMP	DATEMILLI

所有其他不支持的数据类型都转换为 VARCHAR。

分区和拆分

分区由类型为 Integer 的单个分区列表示。该列包含在 SAP HANA 表上定义的分区的分区名称。对于没有分区名称的表，返回 *，相当于单个分区。分区等同于拆分。

名称	Type	描述
PART_ID	整数	SAP HANA 中的命名分区。

Performance

SAP HANA 支持本机分区。Athena SAP HANA 连接器可从这些分区并行检索数据。如果您想查询具有统一分区分布的超大型数据集，强烈建议使用本机分区。选择列的子集可以显著减少查询运行时及扫描的数据。由于并发，连接器显示出严重的节流，有时还会出现查询失败。

Athena SAP HANA 连接器执行谓词下推，以减少查询扫描的数据量。LIMIT 子句、简单谓词和复杂表达式将下推到连接器，以减少扫描数据量并缩短查询执行的运行时间。

LIMIT 子句

LIMIT N 语句用于减少查询扫描的数据量。LIMIT N 下推时，连接器仅向 Athena 返回 N 行。

Predicates

谓词是 SQL 查询的 WHERE 子句中的表达式，其评估结果为布尔值并根据多个条件筛选行。Athena SAP HANA 连接器可以组合这些表达式并将其直接推送到 SAP HANA，以增强功能并减少扫描的数据量。

以下 Athena SAP HANA 连接器运算符支持谓词下推：

- 布尔值：AND、OR、NOT
- 相
  等：EQUAL、NOT_EQUAL、LESS_THAN、LESS_THAN_OR_EQUAL、GREATER_THAN、GREATER_

- 算术：ADD、SUBTRACT、MULTIPLY、DIVIDE、MODULUS、NEGATE
- 其他：LIKE_PATTERN、IN

组合下推示例

要增强查询功能，组合下推类型，如以下示例所示：

```
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;
```

传递查询

SAP HANA 连接器支持传递查询。传递查询使用表函数将完整查询下推到数据来源来执行查询。

要在 SAP HANA 中执行传递查询，可以使用以下语法：

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

以下示例查询将查询下推到 SAP HANA 中的数据来源。该查询选择了 `customer` 表中的所有列，将结果限制为 10。

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

许可证信息

使用此连接器，即表示您确认包含第三方组件（这些组件的列表可在此连接器的 pom.xml 文件中找到），并同意 GitHub.com 上的 LICENSE.txt 文件中提供的相应第三方许可证中的条款。

其他 资源

有关最新 JDBC 驱动程序版本信息，请参见 GitHub.com 上适用于 SAP HANA 连接器的 pom.xml 文件。

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena Snowflake 连接器

适用于 Snowflake 的 Amazon Athena 连接器使 Amazon Athena 能够使用 JDBC 对存储在 Snowflake SQL 数据库 中的数据或 RDS 实例运行 SQL 查询。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

限制

- 不支持写入 DDL 操作。
- 在多路复用器设置中，溢出桶和前缀在所有数据库实例之间共享。
- 任何相关的 Lambda 限制。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。
- 目前，不支持单个拆分的 Snowflake 视图。
- 在 Snowflake 中，由于对象名称区分大小写，因此两个表能分别以小写和大写形式具有相同的名称（例如 EMPLOYEE 和 employee)。在 Athena 联合查询中，架构表名称以小写形式提供给 Lambda 函数。要解决此问题，您可以提供 @schemaCase 查询提示，用于从名称区分大小写的表中检索数据。以下是两个带有查询提示的示例查询。

```
SELECT *
FROM "lambda:snowflakeconnector".SYSTEM."MY_TABLE@schemaCase=upper&tableCase=upper"
```

```
SELECT *
FROM "lambda:snowflakeconnector".SYSTEM."MY_TABLE@schemaCase=upper&tableCase=lower"
```

术语

以下术语与 Snowflake 连接器有关。

- 数据库实例 — 部署在本地、Amazon EC2 或 Amazon RDS 上的任何数据库实例。
- 处理程序 — 访问您数据库实例的 Lambda 处理程序。处理程序可以用于元数据或数据记录。
- 元数据处理程序 — 从您的数据库实例中检索元数据的 Lambda 处理程序。
- 记录处理程序 — 从您的数据库实例中检索数据记录的 Lambda 处理程序。
- 复合处理程序 — 从您的数据库实例中检索元数据和数据记录的 Lambda 处理程序。
- 属性或参数 — 处理程序用来提取数据库信息的数据库属性。您可以将这些属性配置为 Lambda 环境变量。
- 连接字符串 — 用于建立数据库实例连接的文本字符串。
- 目录 — 向 Athena 注册的非 AWS Glue 目录，是 `connection_string` 属性的必要前缀。
- 多路复用处理程序 — 可以接受和使用多个数据库连接的 Lambda 处理程序。

参数

使用本节中的 Lambda 环境变量来配置 Snowflake 连接器。

连接字符串

使用以下格式的 JDBC 连接字符串连接到数据库实例。

```
snowflake://${jdbc_connection_string}
```

使用多路复用处理程序

您可以使用多路复用器通过单个 Lambda 函数连接到多个数据库实例。按目录名称来路由请求。在 Lambda 中使用以下类。

处理程序	类
复合处理程序	`SnowflakeMuxCompositeHandler`
元数据处理程序	`SnowflakeMuxMetadataHandler`
记录处理程序	`SnowflakeMuxRecordHandler`

多路复用处理程序参数

参数	描述
$*catalog*_connection_string	必需。数据库实例连接字符串。将 Athena 中使用的目录的名称作为环境变量前缀。例如，如果向 Athena 注册的目录是 mysnowflakecatalog ，则环境变量名称是 mysnowflakecatalog_connection_string 。
default	必需。默认连接字符串。目录为 lambda:${ *AWS_LAMBDA_FUNCTION_NAME* } 时使用此字符串。

以下示例属性适用于支持两个数据库实例的 Snowflake MUX Lambda 函数：snowflake1（默认）和 snowflake2。

属性	值
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

提供凭证

要在 JDBC 连接字符串中为数据库提供用户名和密码，可以使用连接字符串属性或 AWS Secrets Manager。

- 连接字符串 — 可以将用户名和密码指定为 JDBC 连接字符串中的属性。

> **Important**
>
> 作为安全最佳实践，请勿在您的环境变量或连接字符串中使用硬编码凭证。有关将硬编码密钥移至 AWS Secrets Manager 的信息，请参阅《AWS Secrets Manager 用户指南》中的将硬编码密钥移至 AWS Secrets Manager。

- AWS Secrets Manager - 要将 Athena 联合查询功能与 AWS Secrets Manager 配合使用，连接到您的 Lambda 函数的 VPC 应该拥有互联网访问权限或者 VPC 端点，以连接到 Secrets Manager。

  您可以将 AWS Secrets Manager 中的密钥名称放入您的 JDBC 连接字符串中。连接器将该密钥名称替换为来自 Secrets Manager 的 `username` 和 `password` 值。

  对于 Amazon RDS 数据库实例，将紧密集成这种支持。如果您使用 Amazon RDS，我们强烈建议您使用 AWS Secrets Manager 和凭证轮换。如果您的数据库不使用 Amazon RDS，请按以下格式将凭证存储为 JSON：

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

带有密钥名称的示例连接字符串

以下字符串带有密钥名称 `${Test/RDS/Snowflake1}`。

```
snowflake://jdbc:snowflake://snowflake1.host:port/?
warehouse=warehousename&db=db1&schema=schema1${Test/RDS/Snowflake1}&...
```

该连接器使用该密钥名称来检索密钥，并提供用户名和密码，如以下示例所示。

```
snowflake://jdbc:snowflake://snowflake1.host:port/
warehouse=warehousename&db=db1&schema=schema1&user=sample2&password=sample2&...
```

目前，Snowflake 可以识别 `user` 和 `password` JDBC 属性。还接受以下格式的用户名和密码：*###/##*（不含密钥 `user` 或 `password`）。

使用单个连接处理程序

您可以使用以下单个连接元数据和记录处理程序连接到单个 Snowflake 实例。

处理程序类型	类
复合处理程序	SnowflakeCompositeHandler
元数据处理程序	SnowflakeMetadataHandler
记录处理程序	SnowflakeRecordHandler

单个连接处理程序参数

参数	描述
default	必需。默认连接字符串。

单个连接处理程序支持一个数据库实例，并且必须提供 default 连接字符串参数。将忽略所有其他连接字符串。

以下示例属性适用于 Lambda 函数支持的单个 Snowflake 实例。

属性	值
default	snowflake://jdbc:snowflake://snowflake1.host:port/?secret=Test/RDS/Snowflake1

溢出参数

Lambda 开发工具包可以将数据溢出到 Amazon S3。由同一 Lambda 函数访问的所有数据库实例都会溢出到同一位置。

参数	描述
spill_bucket	必需。溢出桶名称。
spill_prefix	必需。溢出桶密钥前缀。
spill_put_request_headers	（可选）用于溢出的 Amazon S3 putObject 请求的请求标头和值的 JSON 编码映射（例如，{"x-amz-server-side-

参数	描述
	encryption" : "AES256"})。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject 。

数据类型支持

下表显示了适用于 JDBC 和 Apache Arrow 的相应数据类型。

JDBC	Arrow
布尔值	位
整数	Tiny
短型	Smallint
整数	Int
长整型	Bigint
float	Float4
Double	Float8
Date	DateDay
Timestamp	DateMilli
String	Varchar
字节	Varbinary
BigDecimal	十进制
ARRAY	列出

数据类型转换

除了 JDBC 到 Arrow 的转换外，连接器还执行某些其他转换，以使 Snowflake 源和 Athena 数据类型兼容。这些转换有助于确保成功执行查询。下表显示了这些转换。

源数据类型 (Snowflake)	转换后的数据类型 (Athena)
TIMESTAMP	TIMESTAMPMILLI
DATE	TIMESTAMPMILLI
INTEGER	INT
DECIMAL	BIGINT
TIMESTAMP_NTZ	TIMESTAMPMILLI

所有其他不支持的数据类型都转换为 VARCHAR。

分区和拆分

分区用于确定如何为该连接器生成拆分。Athena 将构建一个 `varchar` 类型的合成列，它将展示表的分区方案，以帮助该连接器生成拆分。该连接器不会修改实际的表定义。

若要创建此合成列和分区，Athena 需要定义主键。但是，由于 Snowflake 不强制执行主键约束，您必须自己强制执行唯一性。不这样做会导致 Athena 默认为单次拆分。

Performance

为了获得最佳性能，请尽可能在查询中使用筛选条件。此外，我们强烈建议使用本机分区来检索具有均匀分区分布的大型数据集。选择列的子集可以显著减少查询运行时及扫描的数据。Snowflake 连接器能够灵活地应对并发造成的节流。

Athena Snowflake 连接器执行谓词下推，以减少查询扫描的数据量。`LIMIT` 子句、简单谓词和复杂表达式将下推到连接器，以减少扫描数据量并缩短查询执行的运行时间。

LIMIT 子句

`LIMIT N` 语句用于减少查询扫描的数据量。`LIMIT N` 下推时，连接器仅向 Athena 返回 `N` 行。

Predicates

谓词是 SQL 查询的 WHERE 子句中的表达式，其评估结果为布尔值并根据多个条件筛选行。Athena Snowflake 连接器可以组合这些表达式并将其直接推送到 Snowflake，以增强功能并减少扫描的数据量。

以下 Athena Snowflake 连接器运算符支持谓词下推：

- 布尔值：AND、OR、NOT
- 相等：EQUAL、NOT_EQUAL、LESS_THAN、LESS_THAN_OR_EQUAL、GREATER_THAN、GREATER_
- 算术：ADD、SUBTRACT、MULTIPLY、DIVIDE、MODULUS、NEGATE
- 其他：LIKE_PATTERN、IN

组合下推示例

要增强查询功能，组合下推类型，如以下示例所示：

```
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;
```

传递查询

Snowflake 连接器支持传递查询。传递查询使用表函数将完整查询下推到数据来源来执行查询。

要在 Snowflake 中执行传递查询，可以使用以下语法：

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

以下示例查询将查询下推到 Snowflake 中的数据来源。该查询选择了 customer 表中的所有列，将结果限制为 10。

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

许可证信息

使用此连接器，即表示您确认包含第三方组件（这些组件的列表可在此连接器的 pom.xml 文件中找到），并同意 GitHub.com 上的 LICENSE.txt 文件中提供的相应第三方许可证中的条款。

其他 资源

有关最新 JDBC 驱动程序版本信息，请参见 GitHub.com 上适用于 Snowflake 连接器的 pom.xml 文件。

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena Microsoft SQL Server 连接器

适用于 Microsoft SQL Server 的 Amazon Athena 连接器使 Amazon Athena 能够使用 JDBC 对存储在 Microsoft SQL Server 中的数据运行 SQL 查询。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

限制

- 不支持写入 DDL 操作。
- 在多路复用器设置中，溢出桶和前缀在所有数据库实例之间共享。
- 任何相关的 Lambda 限制。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。
- 在筛选条件中，必须将 `Date` 和 `Timestamp` 数据类型转换为适当的数据类型。
- 要搜索类型 `Real` 和 `Float` 的负值，请使用 <= 或 >= 运算符。
- 不支持 `binary`、`varbinary`、`image` 和 `rowversion` 数据类型。

术语

以下术语与 SQL Server 连接器有关。

- 数据库实例 — 部署在本地、Amazon EC2 或 Amazon RDS 上的任何数据库实例。
- 处理程序 — 访问您数据库实例的 Lambda 处理程序。处理程序可以用于元数据或数据记录。
- 元数据处理程序 — 从您的数据库实例中检索元数据的 Lambda 处理程序。
- 记录处理程序 — 从您的数据库实例中检索数据记录的 Lambda 处理程序。
- 复合处理程序 — 从您的数据库实例中检索元数据和数据记录的 Lambda 处理程序。
- 属性或参数 — 处理程序用来提取数据库信息的数据库属性。您可以将这些属性配置为 Lambda 环境变量。
- 连接字符串 — 用于建立数据库实例连接的文本字符串。
- 目录 — 向 Athena 注册的非 AWS Glue 目录，是 `connection_string` 属性的必要前缀。
- 多路复用处理程序 — 可以接受和使用多个数据库连接的 Lambda 处理程序。

参数

使用本节中的 Lambda 环境变量来配置 SQL Server 连接器。

连接字符串

使用以下格式的 JDBC 连接字符串连接到数据库实例。

```
sqlserver://${jdbc_connection_string}
```

使用多路复用处理程序

您可以使用多路复用器通过单个 Lambda 函数连接到多个数据库实例。按目录名称来路由请求。在 Lambda 中使用以下类。

处理程序	类
复合处理程序	`SqlServerMuxCompositeHandler`
元数据处理程序	`SqlServerMuxMetadataHandler`
记录处理程序	`SqlServerMuxRecordHandler`

多路复用处理程序参数

参数	描述
$*catalog*_connection_string	必需。数据库实例连接字符串。将 Athena 中使用的目录的名称作为环境变量前缀。例如，如果向 Athena 注册的目录是 mysqlservercatalog ，则环境变量名称是 mysqlservercatalog_connection_string 。
default	必需。默认连接字符串。目录为 lambda:${ *AWS_LAMBDA_FUNCTION_NAME* } 时使用此字符串。

以下示例属性适用于支持两个数据库实例的 SQLServer MUX Lambda 函数：sqlserver1（默认）和 sqlserver2。

属性	值
default	sqlserver://jdbc:sqlserver://sqlserver1. *hostname*:*port*;databaseName= *<database_name>* ;${*secret1_name* }
sqlserver_catalog1_connection_string	sqlserver://jdbc:sqlserver://sqlserver1. *hostname*:*port*;databaseName= *<database_name>* ;${*secret1_name* }
sqlserver_catalog2_connection_string	sqlserver://jdbc:sqlserver://sqlserver2. *hostname*:*port*;databaseName= *<database_name>* ;${*secret2_name* }

提供凭证

要在 JDBC 连接字符串中为数据库提供用户名和密码，可以使用连接字符串属性或 AWS Secrets Manager。

- 连接字符串 — 可以将用户名和密码指定为 JDBC 连接字符串中的属性。

> **Important**
>
> 作为安全最佳实践，请勿在您的环境变量或连接字符串中使用硬编码凭证。有关将硬编码密钥移至 AWS Secrets Manager 的信息，请参阅《AWS Secrets Manager 用户指南》中的将硬编码密钥移至 AWS Secrets Manager。

- AWS Secrets Manager - 要将 Athena 联合查询功能与 AWS Secrets Manager 配合使用，连接到您的 Lambda 函数的 VPC 应该拥有互联网访问权限或者 VPC 端点，以连接到 Secrets Manager。

  您可以将 AWS Secrets Manager 中的密钥名称放入您的 JDBC 连接字符串中。连接器将该密钥名称替换为来自 Secrets Manager 的 `username` 和 `password` 值。

  对于 Amazon RDS 数据库实例，将紧密集成这种支持。如果您使用 Amazon RDS，我们强烈建议您使用 AWS Secrets Manager 和凭证轮换。如果您的数据库不使用 Amazon RDS，请按以下格式将凭证存储为 JSON：

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

带有密钥名称的示例连接字符串

以下字符串带有密钥名称 `${secret_name}`。

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

该连接器使用该密钥名称来检索密钥，并提供用户名和密码，如以下示例所示。

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

使用单个连接处理程序

您可以使用以下单连接元数据和记录处理程序连接到单个 SQL Server 实例。

处理程序类型	类
复合处理程序	`SqlServerCompositeHandler`

处理程序类型	类
元数据处理程序	SqlServerMetadataHandler
记录处理程序	SqlServerRecordHandler

单个连接处理程序参数

参数	描述
default	必需。默认连接字符串。

单个连接处理程序支持一个数据库实例，并且必须提供 default 连接字符串参数。将忽略所有其他连接字符串。

以下示例属性适用于 Lambda 函数支持的单个 SQL Server 实例。

属性	值
default	sqlserver://jdbc:sqlserver:// *hostname*:*port*;databaseName= *<database_name>* ;${*secret_name* }

溢出参数

Lambda 开发工具包可以将数据溢出到 Amazon S3。由同一 Lambda 函数访问的所有数据库实例都会溢出到同一位置。

参数	描述
spill_bucket	必需。溢出桶名称。
spill_prefix	必需。溢出桶密钥前缀。
spill_put_request_headers	（可选）用于溢出的 Amazon S3 putObject 请求的请求标头和值的 JSON 编码映射（例如，{"x-amz-server-side-encryption" : "AES256"})。有关其他可能的标头，请参

参数	描述
	阅《Amazon Simple Storage Service API 参考》中的 PutObject。

数据类型支持

下表显示了适用于 SQL Server 和 Apache Arrow 的相应数据类型。

SQL Server	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)

SQL Server	Arrow
date	Date(DAY)
time	VARCHAR
datetimeoffset	Date(MILLISECOND)
char[n]	VARCHAR
varchar[n/max]	VARCHAR
nchar[n]	VARCHAR
nvarchar[n/max]	VARCHAR
文本	VARCHAR
ntext	VARCHAR

分区和拆分

分区由类型为 `varchar` 的单个分区列表示。如果是 SQL Server 连接器，则分区函数决定如何在表上应用分区。分区函数和列名称信息可以从 SQL Server 元数据表中检索。然后，自定义查询会获取分区。拆分是根据收到的不同分区的数量而创建。

Performance

选择列的子集可以显著减少查询运行时及扫描的数据。SQL Server 连接器能够灵活地应对并发造成的节流。

Athena SQL Server 连接器可执行谓词下推，以减少查询扫描的数据量。简单谓词和复杂表达式将下推到连接器，以减少扫描的数据量并缩短查询执行的运行时间。

Predicates

谓词是 SQL 查询的 `WHERE` 子句中的表达式，其评估结果为布尔值并根据多个条件筛选行。Athena SQL Server 连接器可以组合这些表达式并将其直接推送到 SQL Server，以增强功能并减少扫描的数据量。

以下 Athena SQL Server 连接器运算符支持谓词下推：

- 布尔值：AND、OR、NOT
- 相
  等：EQUAL、NOT_EQUAL、LESS_THAN、LESS_THAN_OR_EQUAL、GREATER_THAN、GREATER_
- 算术：ADD、SUBTRACT、MULTIPLY、DIVIDE、MODULUS、NEGATE
- 其他：LIKE_PATTERN、IN

组合下推示例

要增强查询功能，组合下推类型，如以下示例所示：

```
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%');
```

传递查询

SQL Server 连接器支持传递查询。传递查询使用表函数将完整查询下推到数据来源来执行查询。

要在 SQL Server 中执行传递查询，可以使用以下语法：

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

以下示例查询将查询下推到 SQL Server 中的数据来源。该查询选择了 customer 表中的所有列，将结果限制为 10。

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

许可证信息

使用此连接器，即表示您确认包含第三方组件（这些组件的列表可在此连接器的 pom.xml 文件中找到），并同意 GitHub.com 上的 LICENSE.txt 文件中提供的相应第三方许可证中的条款。

其他 资源

有关最新 JDBC 驱动程序版本信息，请参见 GitHub.com 上适用于 SQL Server 连接器的 pom.xml 文件。

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena Teradata 连接器

适用于 Teradata 的 Amazon Athena 连接器使 Athena 能够对存储在 Teradata 数据库中的数据运行 SQL 查询。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

限制

- 不支持写入 DDL 操作。
- 在多路复用器设置中，溢出桶和前缀在所有数据库实例之间共享。
- 任何相关的 Lambda 限制。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。

术语

以下术语与 Teradata 连接器有关。

- 数据库实例 — 部署在本地、Amazon EC2 或 Amazon RDS 上的任何数据库实例。
- 处理程序 — 访问您数据库实例的 Lambda 处理程序。处理程序可以用于元数据或数据记录。
- 元数据处理程序 — 从您的数据库实例中检索元数据的 Lambda 处理程序。
- 记录处理程序 — 从您的数据库实例中检索数据记录的 Lambda 处理程序。
- 复合处理程序 — 从您的数据库实例中检索元数据和数据记录的 Lambda 处理程序。
- 属性或参数 — 处理程序用来提取数据库信息的数据库属性。您可以将这些属性配置为 Lambda 环境变量。
- 连接字符串 — 用于建立数据库实例连接的文本字符串。

- 目录 — 向 Athena 注册的非 AWS Glue 目录，是 `connection_string` 属性的必要前缀。
- 多路复用处理程序 — 可以接受和使用多个数据库连接的 Lambda 处理程序。

Lambda 层先决条件

要将 Teradata 连接器与 Athena 一起使用，您必须创建一个包含 Teradata JDBC 驱动程序的 Lambda 层。Lambda 层是包含某个 Lambda 函数的其他代码的 `.zip` 文件归档。将 Teradata 连接器部署到您的账户时，您将指定该层的 ARN。这会将带有 Teradata JDBC 驱动程序的 Lambda 层连接到 Teradata 连接器，以便您可以将其与 Athena 一起使用。

有关 Lambda 层的更多信息，请参阅《AWS Lambda 开发人员指南》中的 创建和共享 Lambda 层。

为 Teradata 连接器创建 Lambda 层

1. 浏览到 Teradata JDBC 驱动程序下载页面 https://downloads.teradata.com/download/connectivity/jdbc-driver。
2. 下载 Teradata JDBC 驱动程序。该网站要求您创建一个账户并接受许可协议才能下载文件。
3. 从您下载的归档文件提取 `terajdbc4.jar` 文件。
4. 创建以下文件夹结构并将 `.jar` 文件放在其中。

   ```
   java\lib\terajdbc4.jar
   ```
5. 在整个文件夹结构中创建一个 `.zip` 文件，其中包含 `terajdbc4.jar` 文件。
6. 登录到 AWS Management Console，然后通过以下网址打开 AWS Lambda 控制台：https://console.aws.amazon.com/lambda/。
7. 在导航窗格中选择 Layers（层），然后选择 Create layer（创建层）。
8. 对于 Name（名称），输入层的名称（例如，`TeradataJava11LambdaLayer`）。
9. 确保 Upload a .zip file（上传 .zip 文件）选项已选中。
10. 选择 Upload（上传），然后上传包含 Teradata JDBC 驱动程序的压缩文件夹。
11. 选择创建。
12. 在层的详细信息页面上，选择页面顶部的剪贴板图标复制层 ARN。
13. 保存该 ARN 以供参考。

参数

使用本节中的 Lambda 环境变量来配置 Teradata 连接器。

连接字符串

使用以下格式的 JDBC 连接字符串连接到数据库实例。

```
teradata://${jdbc_connection_string}
```

使用多路复用处理程序

您可以使用多路复用器通过单个 Lambda 函数连接到多个数据库实例。按目录名称来路由请求。在 Lambda 中使用以下类。

处理程序	类
复合处理程序	TeradataMuxCompositeHandler
元数据处理程序	TeradataMuxMetadataHandler
记录处理程序	TeradataMuxRecordHandler

多路复用处理程序参数

参数	描述
$*catalog*_connection_string	必需。数据库实例连接字符串。将 Athena 中使用的目录的名称作为环境变量前缀。例如，如果向 Athena 注册的目录是 myteradatacatalog ，则环境变量名称是 myteradatacatalog_connection_string 。
default	必需。默认连接字符串。目录为 lambda:${ *AWS_LAMBDA_FUNCTION_NAME* } 时使用此字符串。

以下示例属性适用于支持两个数据库实例的 Teradata MUX Lambda 函数：teradata1（默认）和 teradata2。

属性	值
default	teradata://jdbc:teradata://teradata2.host/TMODE=ANSI,C

属性	值
	HARSET=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

提供凭证

要在 JDBC 连接字符串中为数据库提供用户名和密码，可以使用连接字符串属性或 AWS Secrets Manager。

- 连接字符串 — 可以将用户名和密码指定为 JDBC 连接字符串中的属性。

  > **Important**
  >
  > 作为安全最佳实践，请勿在您的环境变量或连接字符串中使用硬编码凭证。有关将硬编码密钥移至 AWS Secrets Manager 的信息，请参阅《AWS Secrets Manager 用户指南》中的将硬编码密钥移至 AWS Secrets Manager。

- AWS Secrets Manager - 要将 Athena 联合查询功能与 AWS Secrets Manager 配合使用，连接到您的 Lambda 函数的 VPC 应该拥有互联网访问权限或者 VPC 端点，以连接到 Secrets Manager。

  您可以将 AWS Secrets Manager 中的密钥名称放入您的 JDBC 连接字符串中。连接器将该密钥名称替换为来自 Secrets Manager 的 `username` 和 `password` 值。

  对于 Amazon RDS 数据库实例，将紧密集成这种支持。如果您使用 Amazon RDS，我们强烈建议您使用 AWS Secrets Manager 和凭证轮换。如果您的数据库不使用 Amazon RDS，请按以下格式将凭证存储为 JSON：

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

带有密钥名称的示例连接字符串

以下字符串带有密钥名称 ${Test/RDS/Teradata1}。

```
teradata://jdbc:teradata1.host/TMODE=ANSI,CHARSET=UTF8,DATABASE=TEST,${Test/RDS/
Teradata1}&...
```

该连接器使用该密钥名称来检索密钥，并提供用户名和密码，如以下示例所示。

```
teradata://jdbc:teradata://teradata1.host/
TMODE=ANSI,CHARSET=UTF8,DATABASE=TEST,...&user=sample2&password=sample2&...
```

目前，Teradata 可以识别 user 和 password JDBC 属性。还接受以下格式的用户名和密码：###/###（不含密钥 user 或 password）。

使用单个连接处理程序

您可以使用以下单个连接元数据和记录处理程序连接到单个 Teradata 实例。

处理程序类型	类
复合处理程序	TeradataCompositeHandler
元数据处理程序	TeradataMetadataHandler
记录处理程序	TeradataRecordHandler

单个连接处理程序参数

参数	描述
default	必需。默认连接字符串。

单个连接处理程序支持一个数据库实例，并且必须提供 default 连接字符串参数。将忽略所有其他连接字符串。

以下示例属性适用于 Lambda 函数支持的单个 Teradata 实例。

属性	值
default	teradata://jdbc:teradata://teradata1.host/TMODE=ANSI,CHARSET=UTF8,DATABASE=TEST,secret=Test/RDS/Teradata1

溢出参数

Lambda 开发工具包可以将数据溢出到 Amazon S3。由同一 Lambda 函数访问的所有数据库实例都会溢出到同一位置。

参数	描述
spill_bucket	必需。溢出桶名称。
spill_prefix	必需。溢出桶密钥前缀。
spill_put_request_headers	（可选）用于溢出的 Amazon S3 putObject 请求的请求标头和值的 JSON 编码映射（例如，{"x-amz-server-side-encryption" : "AES256"})。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。

数据类型支持

下表显示了适用于 JDBC 和 Apache Arrow 的相应数据类型。

JDBC	Arrow
布尔值	位
整数	Tiny
短型	Smallint

JDBC	Arrow
整数	Int
长整型	Bigint
float	Float4
Double	Float8
Date	DateDay
Timestamp	DateMilli
String	Varchar
字节	Varbinary
BigDecimal	十进制
ARRAY	列出

分区和拆分

分区由类型为 `Integer` 的单个分区列表示。该列包含在 Teradata 表中定义的分区的分区名称。对于没有分区名称的表，返回 *，相当于单个分区。分区等同于拆分。

名称	Type	描述
分区	整数	Teradata 中的命名分区。

Performance

Teradata 支持本机分区。Athena Teradata 连接器可从这些分区并行检索数据。如果您想查询具有统一分区分布的超大型数据集，强烈建议使用本机分区。选择列的子集会显著减少查询运行时。由于并发，连接器显示出一定的节流。

Athena Teradata 连接器可执行谓词下推，以减少查询扫描的数据量。简单谓词和复杂表达式将下推到连接器，以减少扫描的数据量并缩短查询执行的运行时间。

Predicates

谓词是 SQL 查询的 WHERE 子句中的表达式，其评估结果为布尔值并根据多个条件筛选行。Athena Teradata 连接器可以组合这些表达式并将其直接推送到 Teradata，以增强功能并减少扫描的数据量。

以下 Athena Teradata 连接器运算符支持谓词下推：

- 布尔值：AND、OR、NOT
- 相等：EQUAL、NOT_EQUAL、LESS_THAN、LESS_THAN_OR_EQUAL、GREATER_THAN、GREATER_
- 算术：ADD、SUBTRACT、MULTIPLY、DIVIDE、MODULUS、NEGATE
- 其他：LIKE_PATTERN、IN

组合下推示例

要增强查询功能，组合下推类型，如以下示例所示：

```
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%');
```

传递查询

Teradata 连接器支持传递查询。传递查询使用表函数将完整查询下推到数据来源来执行查询。

要在 Teradata 中执行传递查询，可以使用以下语法：

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

以下示例查询将查询下推到 Teradata 中的数据来源。该查询选择了 customer 表中的所有列，将结果限制为 10。

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

```
        ))
```

许可证信息

使用此连接器，即表示您确认包含第三方组件（这些组件的列表可在此连接器的 pom.xml 文件中找到），并同意 GitHub.com 上的 LICENSE.txt 文件中提供的相应第三方许可证中的条款。

其他 资源

有关最新 JDBC 驱动程序版本信息，请参见 GitHub.com 上适用于 Teradata 连接器的 pom.xml文件。

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena Timestream 连接器

Amazon Athena Timestream 连接器使 Amazon Athena 能够与 Amazon Timestream 进行通信，从而使您的时间序列数据可以通过 Amazon Athena 访问。您可以选择使用 AWS Glue Data Catalog 作为元数据的补充来源。

Amazon Timestream 是一个快速、可扩展、完全托管式、专门构建的时间序列数据库，每天可轻松存储和分析数万亿个时间序列数据点。Timestream 通过根据用户定义的策略将最新数据保存在内存中，并将历史数据移动到成本优化的存储层，从而节省您在管理时间序列数据生命周期方面的时间和成本。

如果您在账户中启用了 Lake Formation，则您在 AWS Serverless Application Repository 中部署的 Athena 联合身份 Lambda 连接器的 IAM 角色必须在 Lake Formation 中具有 AWS Glue Data Catalog 的读取权限。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

参数

使用本节中的 Lambda 环境变量来配置 Timestream 连接器。

- spill_bucket - 为超出 Lambda 函数限制的数据指定 Amazon S3 存储桶。
- spill_prefix -（可选）默认为指定 `spill_bucket`（称为 `athena-federation-spill`）中的子文件夹。我们建议您在此位置配置 Amazon S3 存储生命周期，以删除早于预定天数或小时数的溢出内容。

- spill_put_request_headers — （可选）用于溢出的 Amazon S3 `putObject` 请求的请求标头和值的 JSON 编码映射（例如 `{"x-amz-server-side-encryption" : "AES256"}`）。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。
- kms_key_id - （可选）默认情况下，将使用经过 AES-GCM 身份验证的加密模式和随机生成的密钥对溢出到 Amazon S3 的任何数据进行加密。要让您的 Lambda 函数使用 KMS 生成的更强的加密密钥（如 `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`），您可以指定 KMS 密钥 ID。
- disable_spill_encryption - （可选）当设置为 `True` 时，将禁用溢出加密。默认值为 `False`，此时将使用 AES-GCM 对溢出到 S3 的数据使用进行加密 - 使用随机生成的密钥，或者使用 KMS 生成密钥。禁用溢出加密可以提高性能，尤其是当您的溢出位置使用服务器端加密时。
- glue_catalog – （可选）使用此选项指定跨账户 AWS Glue 目录。默认情况下，该连接器将尝试从其自己的 AWS Glue 账户中获取元数据。

在 AWS Glue 中设置数据库和表

您可以选择使用 AWS Glue Data Catalog 作为元数据的补充来源。要使 AWS Glue 表与 Timestream 配合使用，必须具备 AWS Glue 数据库和表，且它们的名称应与要提供补充元数据的 Timestream 数据库和表相匹配。

> **Note**
>
> 为了获得最佳性能，数据库名称和表名称仅限使用小写。使用混合大小写会使连接器执行不区分大小写的搜索，这种搜索的计算密集度更高。

要配置 AWS Glue 表与 Timestream 配合使用，您必须在 AWS Glue 中设置其表属性。

将 AWS Glue 表用于补充元数据

1. 在 AWS Glue 控制台中编辑表，添加以下表属性：
   - timestream-metadata-flag — 此属性向 Timestream 连接器指明该连接器可将该表用于补充元数据。只要表属性的列表中存在 `timestream-metadata-flag` 属性，您就可以为 `timestream-metadata-flag` 提供任何值。
   - _view_template — 使用 AWS Glue 作为补充元数据时，您可以使用此表属性并指定任何 Timestream SQL 作为视图。Athena Timestream 连接器使用来自视图的 SQL 和来自 Athena 的 SQL 来运行查询。如果您想使用 Athena 中不具备的 Timestream SQL 功能，这将非常有用。

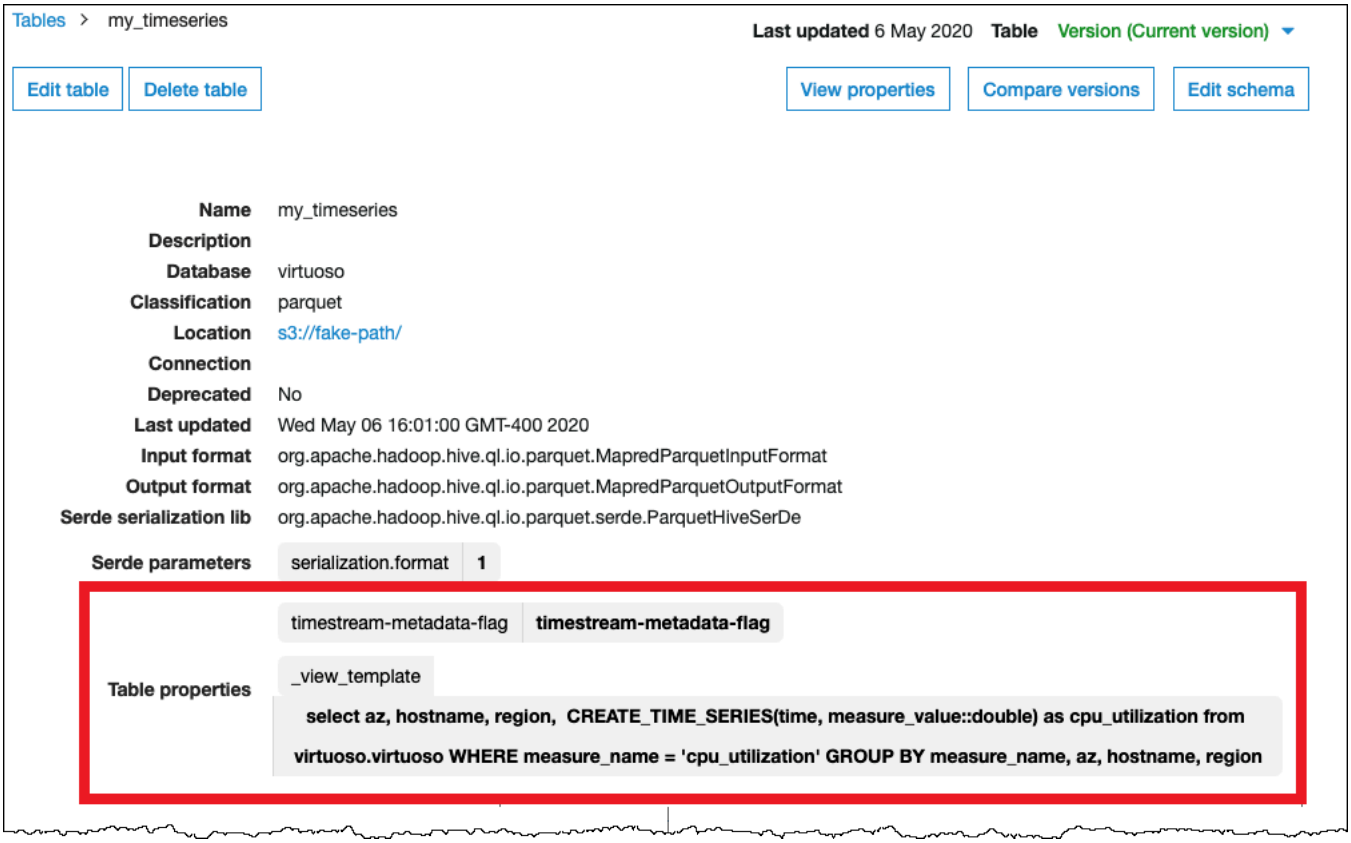
2. 请确保使用适合 AWS Glue 的数据类型，如本文档中所列。

数据类型

目前，Timestream 连接器仅支持 Timestream 中可用数据类型的子集，尤其是：标量值 `varchar`、`double` 和 `timestamp`。

要查询 `timeseries` 数据类型，您必须在使用 Timestream `CREATE_TIME_SERIES` 函数的 AWS Glue 表属性中配置视图。您还需要为使用语法 `ARRAY<STRUCT<time:timestamp,measure_value::double:double>>` 作为任何时间序列列的类型的视图提供架构。务必将 `double` 替换为适合您表的标量类型。

下图显示了为设置时间序列视图而配置的 AWS Glue 表属性的示例。



所需权限

要获取有关此连接器所需 IAM policy 的完整详细信息，请查看 athena-timestream.yaml 文件的 `Policies` 部分。以下列表汇总了所需的权限。

- Amazon S3 写入权限 – 连接器需要对 Amazon S3 中的位置具有写入权限，以溢出大型查询的结果。

- Athena GetQueryExecution – 当上游 Athena 查询终止时，该连接器将使用此权限快速失败。
- AWS Glue Data Catalog — Timestream 连接器需要针对 AWS Glue Data Catalog 的只读访问权限，以获取架构信息。
- CloudWatch Logs – 该连接器需要针对 CloudWatch Logs 的访问权限，以存储日志。
- Timestream 访问权限 — 用于运行 Timestream 查询。

Performance

建议您使用 `LIMIT` 子句将返回的数据（非扫描的数据）限制在 256 MB 以内，以确保交互式查询性能良好。

Athena Timestream 连接器执行谓词下推，以减少查询扫描的数据。`LIMIT` 子句可减少扫描的数据量，但如果未提供谓词，则预期要使用包含 `LIMIT` 子句的 `SELECT` 查询，来扫描至少 16MB 的数据。选择列的子集可以显著减少查询运行时及扫描的数据。Timestream 连接器能够灵活地应对并发造成的节流。

传递查询

Timestream 连接器支持传递查询。传递查询使用表函数将完整查询下推到数据来源来执行查询。

要在 Timestream 中执行传递查询，可以使用以下语法：

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

以下示例查询将查询下推到 Timestream 中的数据来源。该查询选择了 `customer` 表中的所有列，将结果限制为 10。

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

许可证信息

Amazon Athena Timestream 连接器项目已根据 Apache-2.0 许可证获得许可。

其他 资源

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena TPC 基准 DS（TPC-DS）连接器

Amazon Athena TPC-DS 连接器使 Amazon Athena 可以与随机生成的 TPC 基准 DS 数据源通信，用于 Athena Federation 的基准测试和功能测试。Athena TPC-DS 连接器以四种比例因子之一生成符合 TPC-DS 标准的数据库。我们不建议将此连接器用作基于 Amazon S3 的数据湖性能测试的替代方法。

先决条件

- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。

参数

使用本节中的 Lambda 环境变量来配置 TPC-DS 连接器。

- spill_bucket - 为超出 Lambda 函数限制的数据指定 Amazon S3 存储桶。
- spill_prefix -（可选）默认为指定 `spill_bucket`（称为 `athena-federation-spill`）中的子文件夹。我们建议您在此位置配置 Amazon S3 存储生命周期，以删除早于预定天数或小时数的溢出内容。
- spill_put_request_headers —（可选）用于溢出的 Amazon S3 `putObject` 请求的请求标头和值的 JSON 编码映射（例如 `{"x-amz-server-side-encryption" : "AES256"}`）。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。
- kms_key_id -（可选）默认情况下，将使用经过 AES-GCM 身份验证的加密模式和随机生成的密钥对溢出到 Amazon S3 的任何数据进行加密。要让您的 Lambda 函数使用 KMS 生成的更强的加密密钥（如 `a7e63k4b-8loc-40db-a2a1-4d0en2cd8331`），您可以指定 KMS 密钥 ID。
- disable_spill_encryption -（可选）当设置为 `True` 时，将禁用溢出加密。默认值为 `False`，此时将使用 AES-GCM 对溢出到 S3 的数据使用进行加密 - 使用随机生成的密钥，或者使用 KMS 生成密钥。禁用溢出加密可以提高性能，尤其是当您的溢出位置使用服务器端加密时。

测试数据库和表

Athena TPC-DS 连接器以四种比例因子之一（`tpcds1`、`tpcds10`、`tpcds100`、`tpcds250` 或 `tpcds1000`）生成符合 TPC-DS 标准的数据库。

表格摘要

有关测试数据表和列的完整列表，请运行 SHOW TABLES 或 DESCRIBE TABLE 查询。为方便起见，提供以下表格摘要。

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. 清单
13. item
14. promotion
15. reason
16. ship_mode
17. 存储
18. store_returns
19. store_sales
20. time_dim
21. warehouse
22. web_page
23. web_return
24. web_sales
25. web_site

有关与此生成的架构和数据兼容的 TPC-DS 查询，请参阅 GitHub 上的 athena-tpcds/src/main/resources/queries/ 目录。

示例查询

以下 SELECT 查询示例查询特定县的客户人口统计数据的 tpcds 目录。

```
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
```

所需权限

有关此连接器所需 IAM policy 的完整详细信息，请查看 athena-tpcds..yaml 文件的 `Policies` 部分。以下列表汇总了所需的权限。

- Amazon S3 写入权限 – 连接器需要对 Amazon S3 中的位置具有写入权限，以溢出大型查询的结果。
- Athena GetQueryExecution – 当上游 Athena 查询终止时，该连接器将使用此权限快速失败。

Performance

Athena TPC-DS 连接器尝试根据您选择的比例因子对查询并行化处理。谓词下推在 Lambda 函数中执行。

许可证信息

Amazon Athena TPC-DS 连接器项目已根据 Apache-2.0 许可证获得许可。

其他 资源

有关此连接器的更多信息，请访问 GitHub.com 上的相应站点。

Amazon Athena Vertica 连接器

Vertica 是一个列存数据库平台，可以部署在云中或在支持 EB 级数据仓库的本地部署。您可以在联合查询中使用 Amazon Athena Vertica 连接器来查询来自 Athena 的 Vertica 数据来源。例如，您可以通过 Vertica 上的数据仓库和 Amazon S3 中的数据湖运行分析查询。

先决条件

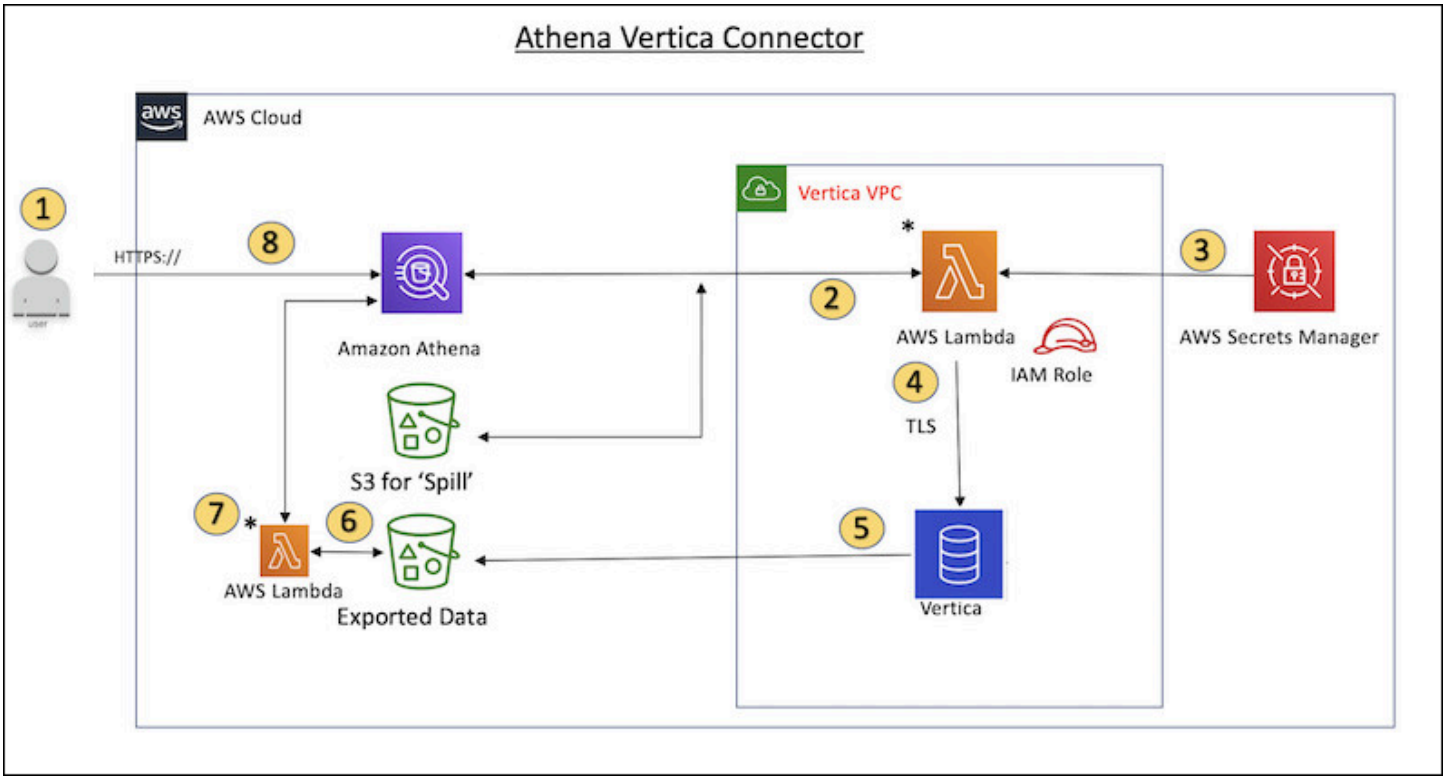
- 可以使用 Athena 控制台或 AWS Serverless Application Repository 将该连接器部署到您的 AWS 账户。有关更多信息，请参阅部署数据来源连接器或使用 AWS Serverless Application Repository 部署数据源连接器。
- 使用此连接器前，请先设置 VPC 和安全组。有关更多信息，请参阅 为数据源连接器创建 VPC。

限制

- 由于 Athena Vertica 连接器使用 Amazon S3 Select 从 Amazon S3 读取 Parquet 文件，因此连接器的性能可能很慢。查询大型表时，我们建议您使用创建表为（选择...）查询和 SQL 谓词。
- 目前，由于 Athena 联合查询中存在一个已知问题，连接器会让 Vertica 将查询表的所有列导出到 Amazon S3，但在 Athena 控制台的结果中只能看到被查询的列。
- 不支持写入 DDL 操作。
- 任何相关的 Lambda 限制。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。

工作流

以下图表显示了使用 Vertica 连接器运行查询的工作流程。



1. 针对 Vertica 中的一个或多个表发出 SQL 查询。
2. 连接器解析 SQL 查询，以通过 JDBC 连接将相关部分发送到 Vertica。
3. 连接字符串使用存储在 AWS Secrets Manager 中的用户名和密码，以获得 Vertica 的访问权限。
4. 连接器使用 Vertica EXPORT 命令包装 SQL 查询，如以下示例所示。

```
EXPORT TO PARQUET (directory = 's3://DOC-EXAMPLE-BUCKET/folder_name',
   Compression='Snappy', fileSizeMB=64) OVER() as
SELECT
PATH_ID,
...
SOURCE_ITEMIZED,
SOURCE_OVERRIDE
FROM DELETED_OBJECT_SCHEMA.FORM_USAGE_DATA
WHERE PATH_ID <= 5;
```

5. Vertica 处理 SQL 查询并将结果集发送到 Amazon S3 存储桶。为了提高吞吐量，Vertica 使用 EXPORT 选项并行处理多个 Parquet 文件的写入操作。
6. Athena 扫描 Amazon S3 存储桶，以确定要为结果集读取的文件数。
7. Athena 对 Lambda 函数进行多次调用，并使用 Amazon S3 Select 从结果集中读取 Parquet 文件。多次调用使 Athena 能够并行读取 Amazon S3 文件，实现高达每秒 100GB 的吞吐量。

8. Athena 使用数据湖扫描的数据处理从 Vertica 返回的数据并返回结果。

术语

以下术语与 Vertica 连接器有关。

- 数据库实例 — 部署在 Amazon EC2 上的 Vertica 数据库的任何实例。
- 处理程序 — 访问您数据库实例的 Lambda 处理程序。处理程序可以用于元数据或数据记录。
- 元数据处理程序 — 从您的数据库实例中检索元数据的 Lambda 处理程序。
- 记录处理程序 — 从您的数据库实例中检索数据记录的 Lambda 处理程序。
- 复合处理程序 — 从您的数据库实例中检索元数据和数据记录的 Lambda 处理程序。
- 属性或参数 — 处理程序用来提取数据库信息的数据库属性。您可以将这些属性配置为 Lambda 环境变量。
- 连接字符串 — 用于建立数据库实例连接的文本字符串。
- 目录 — 向 Athena 注册的非 AWS Glue 目录，是 `connection_string` 属性的必要前缀。

参数

Amazon Athena Vertica 连接器通过 Lambda 环境变量显示了配置选项。您可以使用以下 Lambda 环境变量来配置连接器。

- AthenaCatalogName — Lambda 函数名称
- ExportBucket — 将 Vertica 查询结果导出到的 Amazon S3 存储桶。
- SpillBucket — 此函数可以溢出数据的 Amazon S3 存储桶的名称。
- SpillPrefix — 此函数可以溢出数据的 `SpillBucket` 位置的前缀。
- SecurityGroupIds — 与应该应用于 Lambda 函数的安全组对应的一个或多个 ID（例如 `sg1`、`sg2` 或 `sg3`）。
- SubnetIds — 与 Lambda 函数可用于访问您数据来源的子网对应的一个或多个子网 ID（例如 `subnet1` 或 `subnet2`）。
- SecretNameOrPrefix — 此函数有权访问的 Secrets Manager 中一组密钥的名称或前缀（例如 `vertica-*`）
- VerticaConnectionString — 如果未定义特定于目录的连接，则默认使用的 Vertica 连接详细信息。该字符串可以选择使用 AWS Secrets Manager 语法（例如 `${secret_name}`）。
- VPC ID — 要附加到 Lambda 函数的 VPC ID。

连接字符串

使用以下格式的 JDBC 连接字符串连接到数据库实例。

```
vertica://jdbc:vertica://host_name:port/database?user=vertica-
username&password=vertica-password
```

使用单个连接处理程序

您可以使用以下单个连接元数据和记录处理程序连接到单个 Vertica 实例。

处理程序类型	类
复合处理程序	`VerticaCompositeHandler`
元数据处理程序	`VerticaMetadataHandler`
记录处理程序	`VerticaRecordHandler`

单个连接处理程序参数

参数	描述
`default`	必需。默认连接字符串。

单个连接处理程序支持一个数据库实例，并且必须提供 `default` 连接字符串参数。将忽略所有其他连接字符串。

提供凭证

要在 JDBC 连接字符串中为数据库提供用户名和密码，可以使用连接字符串属性或 AWS Secrets Manager。

- 连接字符串 — 可以将用户名和密码指定为 JDBC 连接字符串中的属性。

> **Important**
>
> 作为安全最佳实践，请勿在您的环境变量或连接字符串中使用硬编码凭证。有关将硬编码密钥移至 AWS Secrets Manager 的信息，请参阅《AWS Secrets Manager 用户指南》中的将硬编码密钥移至 AWS Secrets Manager。

- AWS Secrets Manager - 要将 Athena 联合查询功能与 AWS Secrets Manager 配合使用，连接到您的 Lambda 函数的 VPC 应该拥有互联网访问权限或者 VPC 端点，以连接到 Secrets Manager。

  您可以将 AWS Secrets Manager 中的密钥名称放入您的 JDBC 连接字符串中。连接器将该密钥名称替换为来自 Secrets Manager 的 `username` 和 `password` 值。

  对于 Amazon RDS 数据库实例，将紧密集成这种支持。如果您使用 Amazon RDS，我们强烈建议您使用 AWS Secrets Manager 和凭证轮换。如果您的数据库不使用 Amazon RDS，请按以下格式将凭证存储为 JSON：

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

带有密钥名称的示例连接字符串

以下字符串带有密钥名称 `${vertica-username}` 和 `${vertica-password}`。

```
vertica://jdbc:vertica://host_name:port/database?user=${vertica-username}&password=
${vertica-password}
```

该连接器使用该密钥名称来检索密钥，并提供用户名和密码，如以下示例所示。

```
vertica://jdbc:vertica://host_name:port/database?user=sample-user&password=sample-
password
```

目前，Vertica 连接器可以识别 `vertica-username` 和 `vertica-password` JDBC 属性。

溢出参数

Lambda 开发工具包可以将数据溢出到 Amazon S3。由同一 Lambda 函数访问的所有数据库实例都会溢出到同一位置。

参数	描述
spill_bucket	必需。溢出桶名称。
spill_prefix	必需。溢出桶密钥前缀。
spill_put_request_headers	（可选）用于溢出的 Amazon S3 putObject 请求的请求标头和值的 JSON 编码映射（例如，{"x-amz-server-side-encryption" : "AES256"})。有关其他可能的标头，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。

数据类型支持

下表显示了 Vertica 连接器支持的数据类型。

布尔值
BigInt
短型
整数
长整型
浮点型
Double
Date
Varchar
字节
BigDecimal
作为 Varchar 的时间戳

Performance

Lambda 函数执行投影下推，以减少查询扫描的数据。LIMIT 子句会减少扫描的数据量，但如果未提供谓词，则预期要使用包含 LIMIT 子句的 SELECT 查询，来扫描至少 16 MB 的数据。Vertica 连接器能够灵活地应对并发造成的节流。

传递查询

Vertica 连接器支持传递查询。传递查询使用表函数将完整查询下推到数据来源来执行查询。

要在 Vertica 中执行传递查询，可以使用以下语法：

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

以下示例查询将查询下推到 Vertica 中的数据来源。该查询选择了 customer 表中的所有列，将结果限制为 10。

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

许可证信息

使用此连接器，即表示您确认包含第三方组件（这些组件的列表可在此连接器的 pom.xml 文件中找到），并同意 GitHub.com 上的 LICENSE.txt 文件中提供的相应第三方许可证中的条款。

其他 资源

有关最新 JDBC 驱动程序版本信息，请参见 GitHub.com 上适用于 Vertica 连接器的 pom.xml 文件。

有关此连接器的更多信息，请参阅 GitHub.com 上的相应站点，以及AWS 大数据博客中的文章“使用 Athena 联合查询 SDK 在 Amazon Athena 中查询 Vertica 数据来源”。

部署数据来源连接器

准备创建联合查询是一个分为两部分的过程：部署 Lambda 函数数据源连接器和将 Lambda 函数连接到数据源。在此过程中，您可以为 Lambda 函数指定一个名称，您可以稍后在 Athena 控制台中选择该名称，并为连接器指定一个可以在 SQL 查询中引用的名称。

> **Note**
>
> 若要将 Athena Federated Query 功能与 AWS Secrets Manager 结合使用，您必须为 Secrets Manager 配置 Amazon VPC 私有端点。有关更多信息，请参阅《AWS Secrets Manager 用户指南》中的 创建 Secrets Manager VPC 私有端点。

主题

使用 Athena 控制台

要选择、命名和部署数据源连接器，请在集成过程中使用 Athena 和 Lambda 控制台。

部署数据源连接器

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。

3. 在导航窗格中，选择 Data sources（数据来源）。

4. 在 Data sources（数据源）页面上，选择 Create data source（创建数据源）。
5. 对于 Choose Data source（选择数据源），请选择想要 Athena 查询的数据源，同时考虑以下准则：

   - 选择与数据源对应的联合查询选项。Athena 具有预构建的数据源连接器，您可以为 MySQL、Amazon DocumentDB 和 PostgreSQL 等源配置这些连接器。
   - 如果您想查询 Amazon S3 中的数据，但您没有使用 Apache Hive 元数据仓或本页面上的其他联合查询数据源选项之一，请选择 S3 - AWS Glue Data Catalog。Athena 使用 AWS Glue Data Catalog 在 Amazon S3 中存储数据源的元数据和架构信息。这是默认的（非联合）选项。有关更多信息，请参阅 使用 AWS Glue 连接到 Simple Storage Service（Amazon S3）中的数据来源。
   - 选择 S3 - Apache Hive metastore（S3 - Apache Hive 元数据仓），查询在 Amazon S3 中使用 Apache Hive 元数据仓的数据集。有关此选项的更多信息，请参阅 将 Athena 连接到 Apache Hive 元存储。
   - 如果您想创建自己的数据源连接器，以便与 Athena 一起使用，请选择 Custom or shared connector（自定义或共享连接器）。有关编写数据源连接器的信息，请参阅 使用 Athena Query Federation SDK 编写数据来源连接器。

   本教程选择 Amazon CloudWatch Logs 作为联合数据源。
6. 选择下一步。
7. 在 Enter data source details（输入数据源详细信息）页面中，对于 Data source name（数据源名称），请输入从 Athena 查询数据源时要在 SQL 语句中使用的名称（例如 `CloudWatchLogs`）。名称最多可以包括 127 个字符，并且在您的账户中必须是唯一的。它在创建后即无法更改。有效字符包括 a-z、A-Z、0-9、_（下划线）、@（at 符号）和 -（连字符）。名称 `awsdatacatalog`、`hive`、`jmx` 和 `system` 是 Athena 预留的名称，无法用于数据源名称。
8. 对于 Lambda function（Lambda 函数），请选择 Create Lambda function（创建 Lambda 函数）。您选择的连接器的函数页将在 AWS Lambda 控制台中打开。该页面包括连接器的详细信息。
9. 在 Application settings（应用程序设置）项下，请仔细阅读每个应用程序设置的说明，然后输入符合您要求的值。

   您看到的应用程序设置因数据源的连接器而异。所需的最低设置包括：

   - AthenaCatalogName – 指明其目标数据源的小写 Lambda 函数名称，例如 `cloudwatchlogs`。

- SpillBucket – 您的账户中用于存储超出 Lambda 函数响应大小限制的数据的 Amazon S3 存储桶。

> **Note**
>
> 溢出的数据不会在后续的执行中重复使用，并且可以在 12 小时后安全地删除。Athena 不会为您删除这些数据。要管理这些对象，请考虑添加一个将从 Amazon S3 溢出存储桶中删除旧数据的对象生命周期策略。有关更多信息，请参阅《Amazon S3 用户指南》中的对象生命周期管理。

10. 选中 I acknowledge that this app creates custom IAM roles and resource policies（我确认此应用程序创建自定义 IAM 角色和资源策略）。有关更多信息，请选择 Info (信息) 链接。
11. 选择部署。部署完成后，Lambda 函数将显示在 Lambda 控制台中的 Resource（资源）部分。

连接到数据来源

将数据源连接器部署到您的账户后，可以将 Athena 连接到此数据源连接器。

使用已部署到账户的连接器将 Athena 连接到数据源

1. 返回至 Athena 控制台中的 Enter data source details（输入数据源详细信息）页面。
2. 在 Connection details（连接详细信息）部分中，选择 Select or enter a Lambda function（选择或输入 Lambda 函数）搜索框旁的刷新图标。
3. 在 Lambda 控制台上选择刚创建的函数名称。将显示 Lambda 函数的 ARN。
4. （可选）对于 Tags（标签），添加要与此数据源关联的键值对。有关标签的更多信息，请参阅 为 Athena 资源添加标签。
5. 选择下一步。
6. 在 Review and create（审核和创建）页面中，查看数据源的详细信息，然后选择 Create data source（创建数据源）。
7. 数据源此页面的 Data source details（数据源详细信息）部分显示了有关新连接器的信息。现在，您可以在 Athena 查询中使用连接器。

   有关在查询中使用数据连接器的信息，请参阅 编写联合查询。

使用 AWS Serverless Application Repository 部署数据源连接器

要部署数据源连接器，可以使用 AWS Serverless Application Repository，而非从 Athena 控制台开始。使用 AWS Serverless Application Repository 找到要使用的连接器，提供连接器所需的参数，然后将连接器部署到您的账户。部署连接器后，您可以使用 Athena 控制台向 Athena 提供数据来源。

将连接器部署到您的账户

使用 AWS Serverless Application Repository 将数据源连接器部署到您的账户

1. 登录 AWS Management Console并打开 Serverless App Repository (无服务器应用程序存储库)。
2. 在导航窗格中，选择 Available applications (可用应用程序)。
3. 选择 Show apps that create custom IAM roles or resource policies（显示创建自定义 IAM 角色或资源策略的应用程序）选项。
4. 在搜索框中，键入连接器的名称。有关预构建的 Athena 数据连接器的列表，请参阅 可用数据来源连接器。
5. 选择连接器的名称。选择连接器后会打开 AWS Lambda 控制台中 Lambda 函数的 Application details（应用程序详细信息）页面。
6. 在详细信息页面的右侧，为 Application settings（应用程序设置）填写必填信息。最少的必填设置包括以下项。有关适用于 Athena 所构建的数据连接器的其余可配置选项的信息，请参阅 GitHub 上相应的 Available connectors（可用连接器）主题。

   - AthenaCatalogName – 指明其目标数据源的小写 Lambda 函数名称，例如 `cloudwatchlogs`。
   - SpillBucket – 在您的账户指定一个 Amazon S3 存储桶，用于接收来自超出 Lambda 函数响应大小限制的任何大型响应有效负载的数据。
7. 选中 I acknowledge that this app creates custom IAM roles and resource policies（我确认此应用程序创建自定义 IAM 角色和资源策略）。有关更多信息，请选择 Info (信息) 链接。
8. 在 Application settings（应用程序设置）部分的右下角，选择 Deploy（部署）。部署完成后，Lambda 函数将显示在 Lambda 控制台中的 Resource（资源）部分。

使连接器在 Athena 中可用

现在您可以使用 Athena 控制台使数据源连接器对 Athena 可用。

使数据源连接器对 Athena 可用

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。

2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。

3. 在导航窗格中，选择 Data sources（数据来源）。
4. 在 Data sources（数据源）页面上，选择 Create data source（创建数据源）。
5. 对于 Choose a data source（选择数据源），选择您在 AWS Serverless Application Repository 中创建了连接器的数据源。本教程使用 Amazon CloudWatch Logs 作为联合数据源。
6. 选择下一步。
7. 在 Enter data source details（输入数据源详细信息）页面中，对于 Data source name（数据源名称），请输入从 Athena 查询数据源时要在 SQL 语句中使用的名称（例如 `CloudWatchLogs`）。名称最多可以包括 127 个字符，并且在您的账户中必须是唯一的。它在创建后即无法更改。有效字符包括 a-z、A-Z、0-9、_（下划线）、@（at 符号）和 -（连字符）。名称 `awsdatacatalog`、`hive`、`jmx` 和 `system` 是 Athena 预留的名称，无法用于数据源名称。
8. 在 Connection details（连接详细信息）部分中，请使用 Select or enter a Lambda function（选择或输入 Lambda 函数）框以选择刚创建的函数名称。将显示 Lambda 函数的 ARN。
9. （可选）对于 Tags（标签），添加要与此数据源关联的键值对。有关标签的更多信息，请参阅 为 Athena 资源添加标签。
10. 选择下一步。
11. 在 Review and create（审核和创建）页面中，查看数据源的详细信息，然后选择 Create data source（创建数据源）。
12. 数据源此页面的 Data source details（数据源详细信息）部分显示了有关新连接器的信息。现在，您可以在 Athena 查询中使用连接器。

    有关在查询中使用数据连接器的信息，请参阅 编写联合查询。

为数据源连接器创建 VPC

部分 Athena 数据源连接器需要用到 VPC 和安全组。本主题介绍如何创建具有子网的 VPC，以及如何为 VPC 创建安全组。在此过程中，您要检索自己创建的 VPC、子网和安全组的 ID。您需要这些 ID 来配置连接器，以便与 Athena 搭配使用。

为 Athena 数据源连接器创建 VPC

1. 通过 https://console.aws.amazon.com/vpc/ 登录到AWS Management Console并打开 Amazon VPC 控制台。
2. 在导航窗格中，确保选中 New VPC Experience（新 VPC 体验）。
3. 选择创建 VPC。
4. 在 VPC Settings（VPC 设置）的 Resources to create（要创建的资源）下，选择 VPC and more（VPC 及更多）。
5. 在 Auto-generate（自动生成）中输入一个值，该值将用于为 VPC 中的所有资源生成名称标签。
6. 选择创建 VPC。
7. 选择 View VPC（查看 VPC）。
8. 在 Details（详细信息）部分的 VPC ID 中，复制 VPC ID 供之后参考。

现在，您可以检索刚才创建的 VPC 的子网 ID。

检索 VPC 子网 ID

1. 在 VPC 控制台的导航窗格中，选择 Subnets（子网）。
2. 选择与您创建的子网对应的名称。
3. 在 Details（详细信息）部分的 Subnet ID（子网 ID）中，复制子网 ID 供之后参考。

接着为 VPC 创建安全组。

为 VPC 创建安全组

1. 在 VPC 控制台的导航窗格中，依次选择 Security（安全）、Security Groups（安全组）。
2. 选择创建安全组。
3. 在 Create Security Group（创建安全组）页面上，输入以下信息：
   - 在 Security group name（安全组名称）中，为安全组输入名称。

- 在 Description（描述）中，为安全组输入描述。该字段为必填。
- 在 VPC 中，输入您为数据源连接器创建的 VPC 的 VPC ID。
- 在 Inbound rules（入站规则）和 Outbound rules（出站规则）中，分别添加所需的入站和出站规则。

4. 选择创建安全组。
5. 在安全组的 Details（详细信息）页面中，复制 Security group ID（安全组 ID）供之后参考。

启用跨账户联合查询

联合查询允许您使用部署在 AWS Lambda 上的数据源连接器查询 Amazon S3 以外的数据源。跨账户联合查询功能允许 Lambda 函数和要查询的数据源位于不同的账户中。

作为数据管理员，您可以通过与数据分析人员的账户共享数据连接器来启用跨账户联合查询。或者作为数据分析人员，可以通过将数据管理员提供的共享 Lambda ARN 添加到您的账户中来启用跨账户联合查询。对原始账户中的连接器进行配置更改时，更新的配置将自动应用于其他用户账户中该连接器的共享实例。

注意事项和限制

- 跨账户联合查询功能适用于非 Hive 元数据仓数据连接器，该连接器使用基于 Lambda 的数据源。
- 此功能不适用于 AWS Glue Data Catalog 数据源类型。有关跨账户访问 AWS Glue Data Catalog 的信息，请参阅 授予 AWS Glue 数据目录跨账户访问权限。
- 如果来自连接器的 Lambda 函数的响应超过 6MB 的 Lambda 响应大小限制，Athena 会自动对响应进行加密、批处理并溢出到您配置的 Amazon S3 存储桶。运行 Athena 查询的实体必须有权访问溢出位置，Athena 才能读取溢出的数据。我们建议您设置 Amazon S3 生命周期策略，以从溢出位置删除对象，因为查询完成后不需要数据。
- 不支持跨 AWS 区域 使用联合查询。

所需的权限

- 若要让数据管理员账户 A 与数据分析人员账户 B 共享 Lambda 函数，账户 B 需要 Lambda 调用函数和溢出存储桶访问权限。因此，账户 A 应该将基于资源的策略添加到 Lambda 函数并添加主体访问 Amazon S3 中其溢出存储桶的权限。

  1. 以下策略向账户 A 中的 Lambda 函数授予账户 B 调用 Lambda 函数的权限。

```
{
```

```
  "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. 以下策略允许溢出存储桶访问账户 B 中的主体。

```
{
    "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. 如果 Lambda 函数使用 AWS KMS 密钥而不是联合开发工具包提供的默认加密来加密溢出存储桶，则账户 A 中的 AWS KMS 密钥策略必须向账户 B 中的用户授予访问权限，如以下示例所示。

```
{
    "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.
 }
```

- 若要让账户 A 与账户 B 共享其连接器，账户 B 必须创建一个名为 `AthenaCrossAccountCreate-account-A-id` 的角色，账户 A 通过调用 AWS Security Token Service AssumeRole API 操作担任该角色。

  应在账户 B 中创建允许以下 `CreateDataCatalog` 操作的策略，并添加到账户 B 为账户 A 创建的 `AthenaCrossAccountCreate-account-A-id` 角色中。

```
{
 "Effect": "Allow",
 "Action": "athena:CreateDataCatalog",
 "Resource": "arn:aws:athena:*:account-B-id:datacatalog/*"
}
```

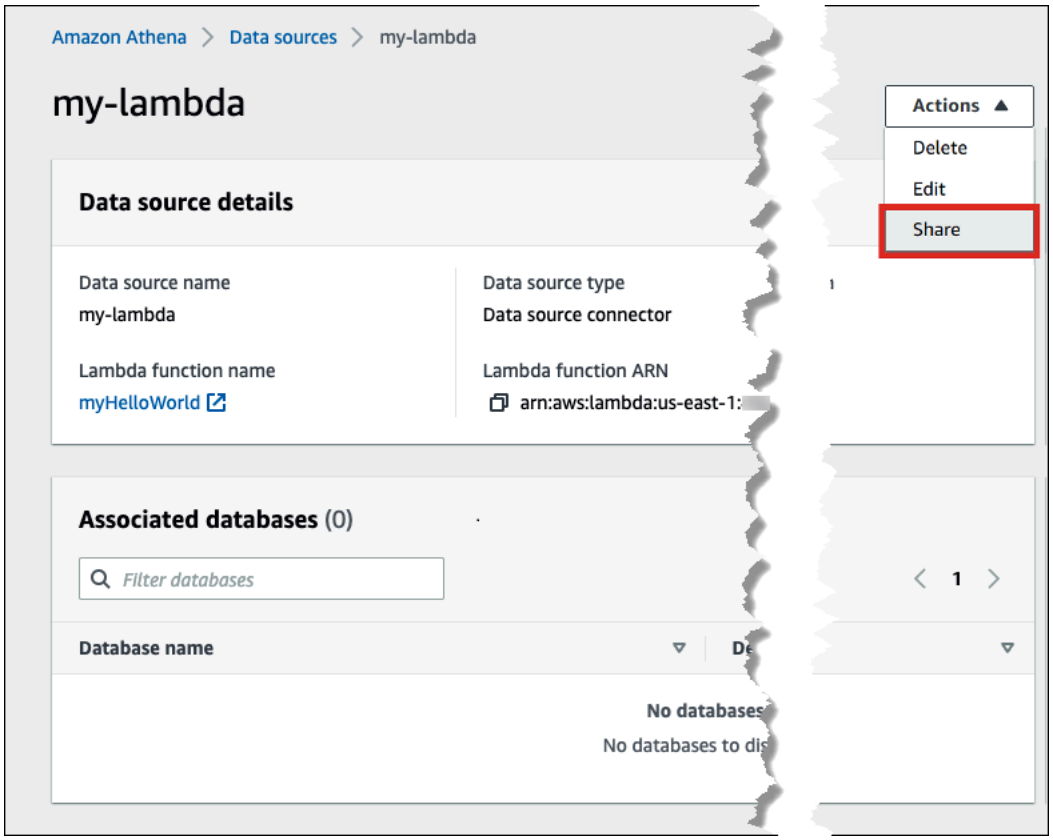
与账户 B 共享账户 A 中的数据源

获得相应权限后，您可以使用 Athena 控制台中的 Data sources（数据源）页面与其他账户（账户 B）共享您账户（账户 A）中的数据连接器。账户 A 保留对连接器的完全控制权和所有权。账户 A 对连接器进行配置更改时，已更新的配置将应用于账户 B 中的共享连接器。

与账户 B 共享账户 A 中的 Lambda 数据源

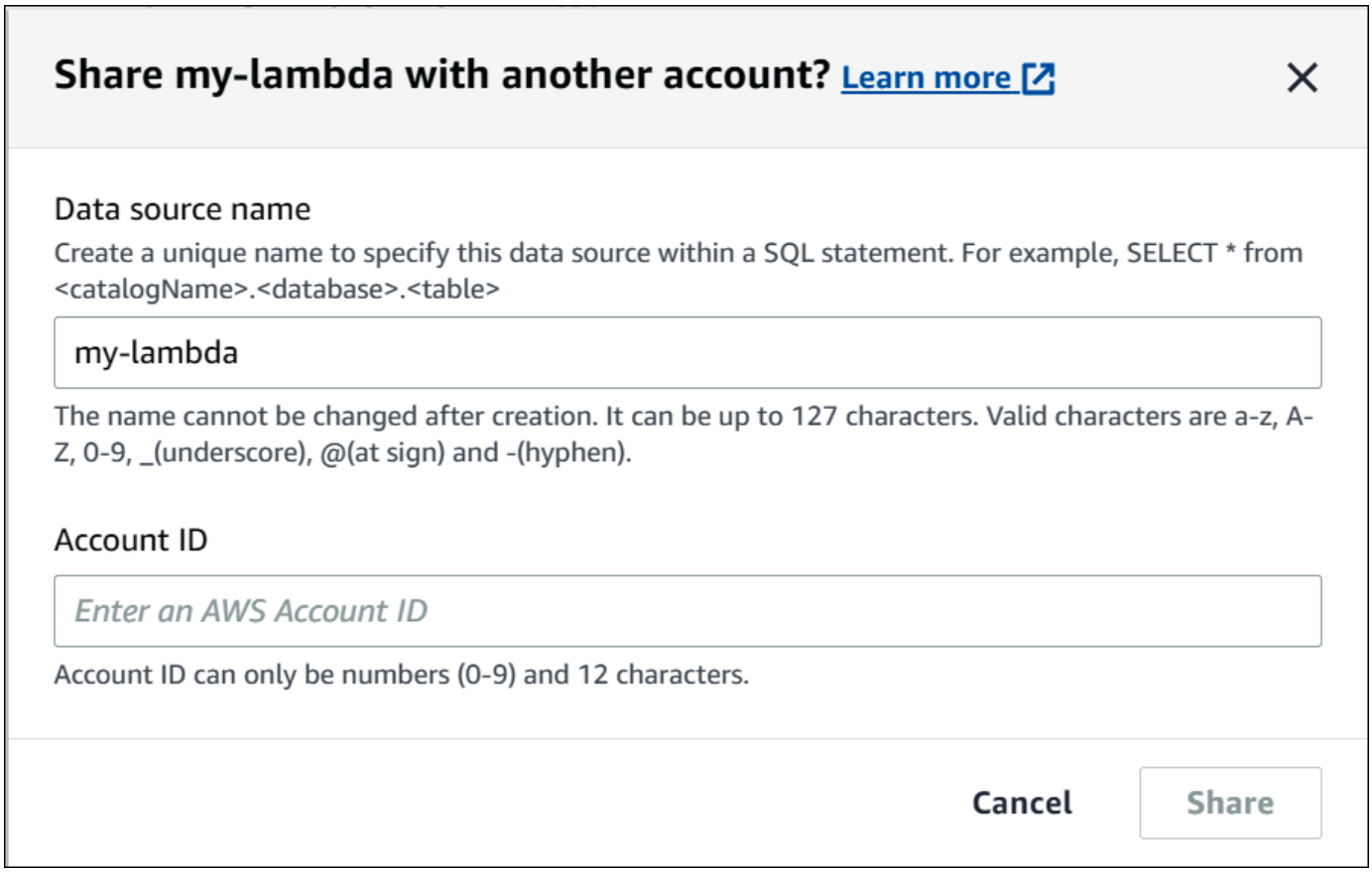
1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。

3. 选择 Data Source（数据源）。
4. 在 Data sources（数据源）页面上，选择要共享的连接器的链接。
5. 在 Lambda 数据源的详细信息页面上，选择右上角的 Share（共享）选项。



6. 在 Share ***Lambda-name*** with another account（与其他账户共享 Lambda-name）对话框中，输入所需信息。

- 在 Data source name（数据源名称）中，输入您要在另一个账户中显示的复制数据源名称。
- 在 Account ID（账户 ID）中，输入要与其共享数据源的账户 ID（在本例中为账户 B）。



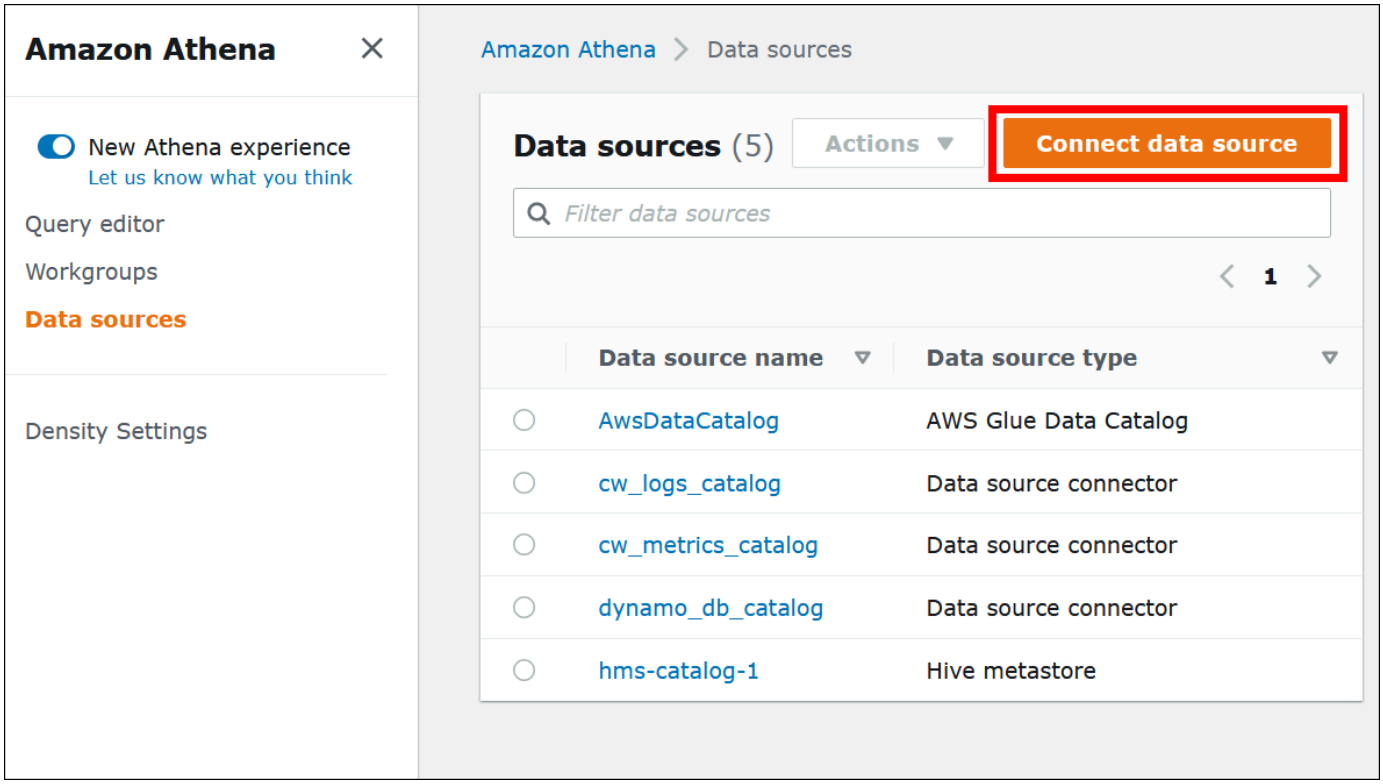
7. 选择共享。您指定的共享数据连接器在账户 B 中创建。账户 A 中连接器的配置更改将应用于账户 B 中的连接器。

将共享数据源从账户 A 添加到账户 B

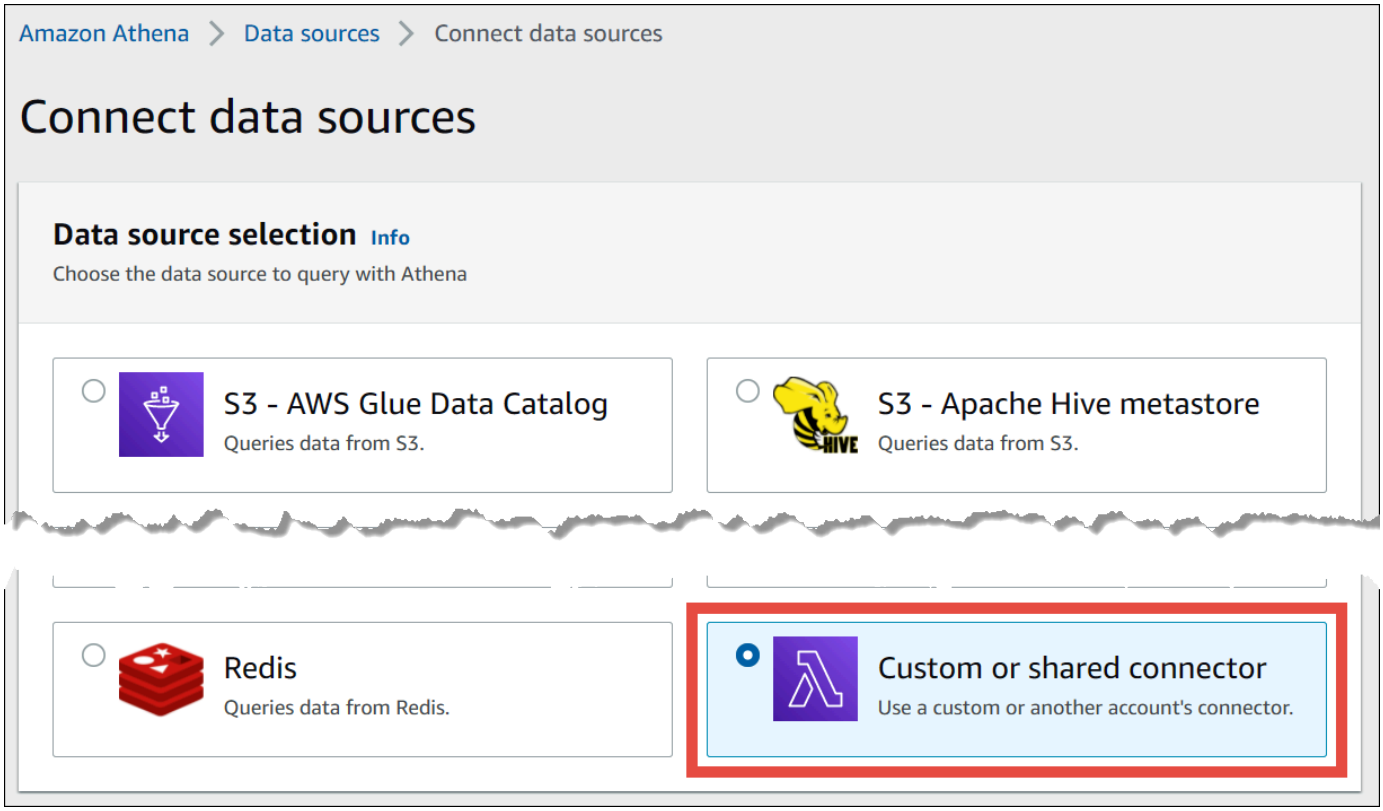
作为数据分析人员，您可能会从数据管理员那里获得连接器的 ARN 以添加到您的账户中。您可以使用 Athena 控制台的 Data sources（数据源）页面，将管理员提供的 Lambda ARN 添加到您的账户中。

将共享数据连接器的 Lambda ARN 添加到您的账户

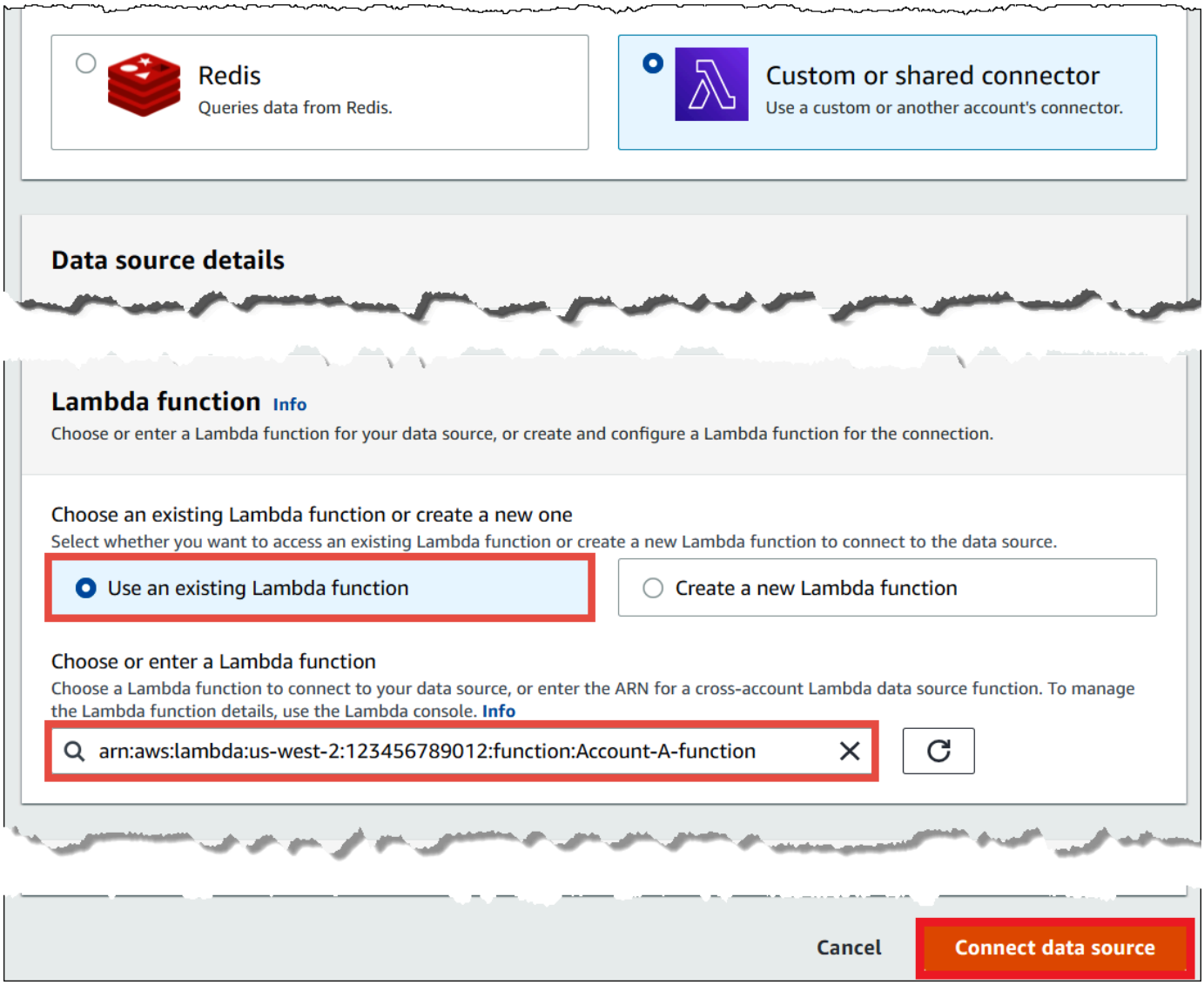
1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果您正在使用新的控制台并且导航窗格不可见，请选择左侧的扩展菜单。
3. 选择 Data Source（数据源）。
4. 在 Data sources（数据源）页面上，选择 Connect data source（连接数据源）。



5. 选择 Custom or shared connector（自定义或共享连接器）。



6. 在 Lambda function（Lambda 函数）部分中，请确保已选择 Use an existing Lambda function（使用现有 Lambda 函数）选项。



7. 在 Choose or enter a Lambda function（选择或输入 Lambda 函数）中，输入账户 A 的 Lambda ARN。
8. 选择 Connect data source (连接数据源)。

故障排除

如果您收到一条错误消息，指出账户 A 没有在账户 B 中担任角色的权限，请确保正确拼写在账户 B 中创建的角色名称，并且附加了适当的策略。

更新数据来源连接器

Athena 建议您定期将所用的数据来源连接器更新为最新版本，以利用新功能和增强功能。首先，必须找到最新的版本号。

查找 Athena Query Federation 最新版本

Athena 数据来源连接器的最新版本号对应于 Athena Query Federation 最新版本。在某些情况下，GitHub 版本可能比 AWS Serverless Application Repository（SAR）上提供的版本稍微更新一些。

查找 Athena Query Federation 最新版本号

1. 访问 GitHub URL https://github.com/awslabs/aws-athena-query-federation/releases/latest。
2. 请注意，主页标题中的版本号格式如下：

   Athena Query Federation 版本 v *year*.*week_of_year*.*iteration_of_week*

   例如，Athena Query Federation 版本 v2023.8.3 的版本号为 2023.8.3。

查找并记下资源名称

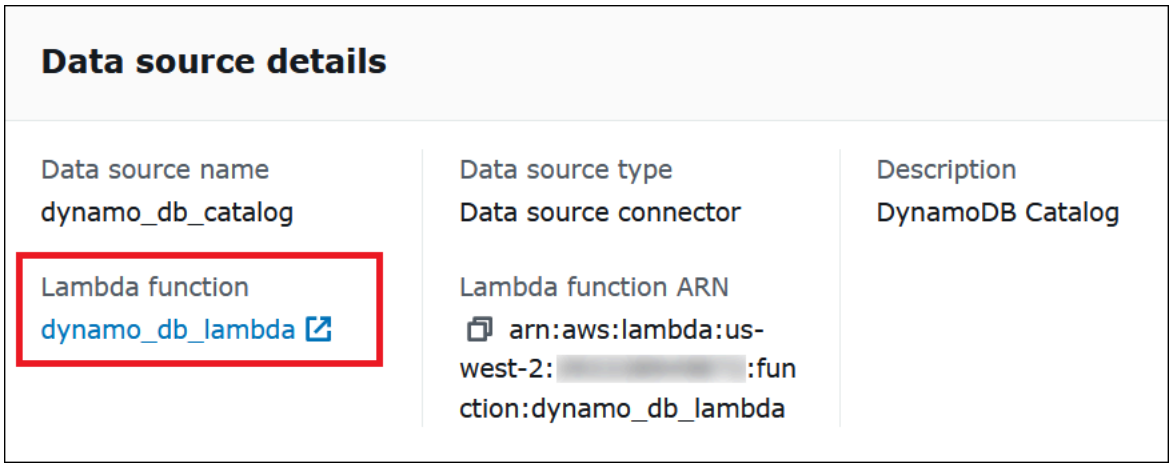
在准备升级时，必须找到并记下以下信息：

1. 连接器的 Lambda 函数名称。
2. Lambda 函数环境变量。
3. Lambda 应用程序名称，用于管理连接器的 Lambda 函数。

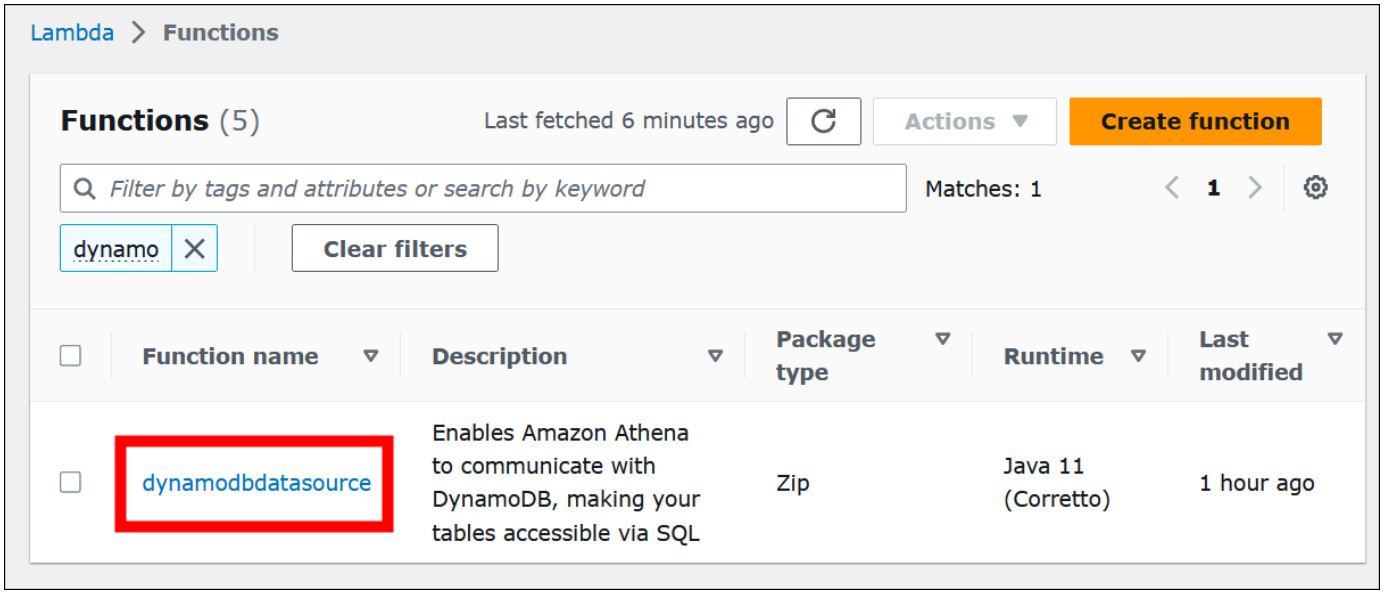
在 Athena 控制台中查找资源名称

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。

3. 在导航窗格中，选择 Data sources（数据来源）。
4. 在数据来源名称列中，选择指向连接器数据来源的链接。
5. 在数据来源详细信息部分的 Lambda 函数下，选择指向您的 Lambda 函数的链接。



6. 在函数页面的函数名称列中，记下连接器的函数名称。



7. 选择函数名称链接。
8. 在函数概述部分下，选择配置选项卡。
9. 在左侧窗格中，选择环境变量。
10. 在环境变量部分中，记下键及其对应的值。
11. 滚动到页面顶部。
12. 在消息此函数属于应用程序中。单击此处对其进行管理，选择单击此处链接。
13. 在 serverlessrepo-***your_application_name*** 页面中，记下不含 serverlessrepo 的应用程序名称。例如，如果应用程序名称为 serverlessrepo-DynamoDbTestApp，则您的应用程序名称为 DynamoDbTestApp。
14. 留在应用程序的 Lambda 控制台页面上，然后继续执行查找您正在使用的连接器的版本中的步骤。

查找您正在使用的连接器的版本

按照以下步骤查找您正在使用的连接器的版本

查找您正在使用的连接器的版本

1. 在 Lambda 应用程序的 Lambda 控制台页面中，选择部署选项卡。
2. 在部署选项卡中，展开 SAM 模板。
3. 搜索 CodeUri。
4. 在 CodeUri 下的键字段中，查找以下字符串：

```
applications-connector_name-
versions-year.week_of_year.iteration_of_week/hash_number
```

以下示例显示了适用于 CloudWatch 连接器的字符串：

```
applications-AthenaCloudwatchConnector-versions-2021.42.1/15151159...
```

5. 记录 *year.week_of_year.iteration_of_week* 的值（例如，2021.42.1）。这是您的连接器的版本。

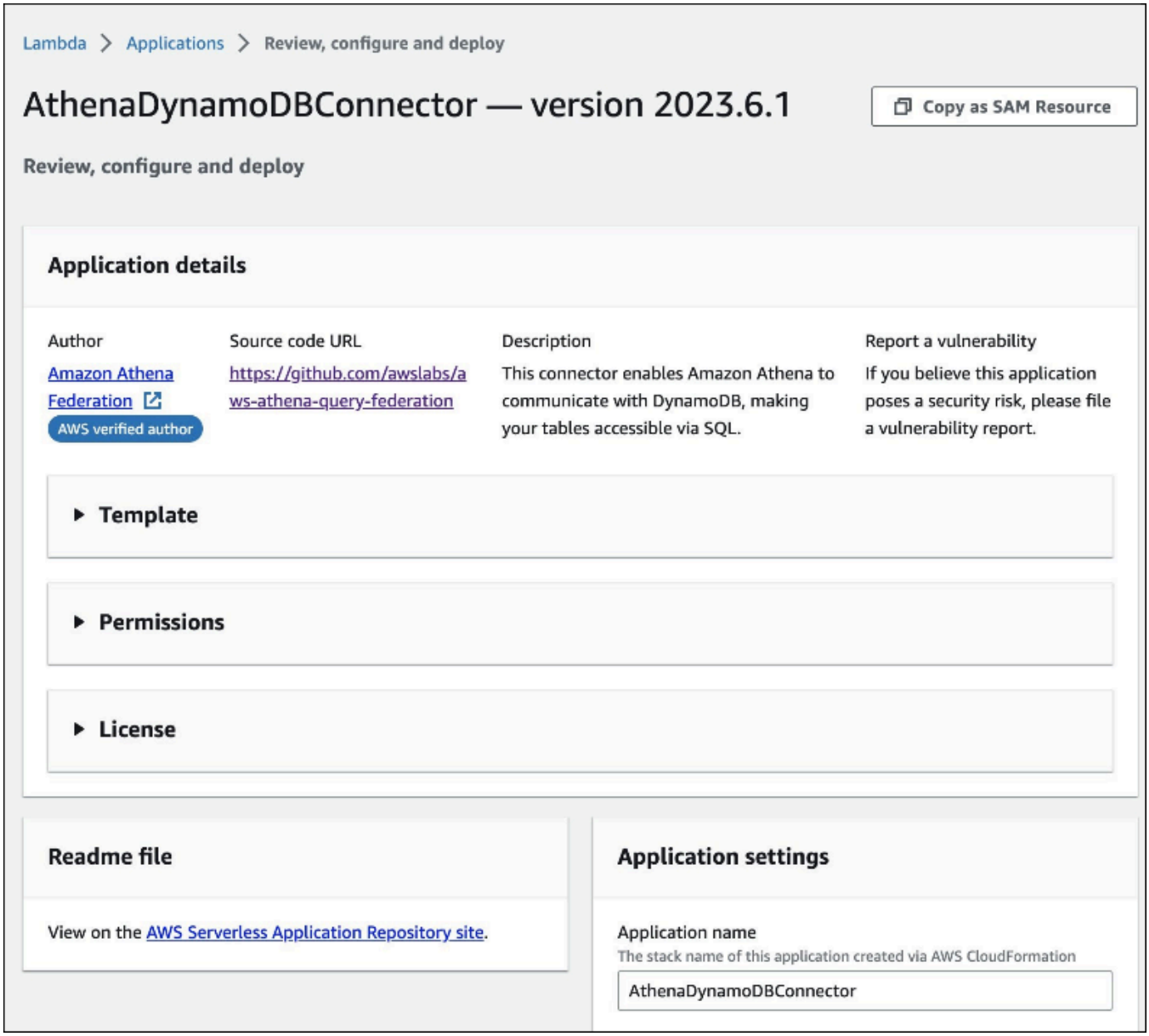
部署连接器的新版本

按照以下步骤部署连接器的新版本。

部署连接器的新版本

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。

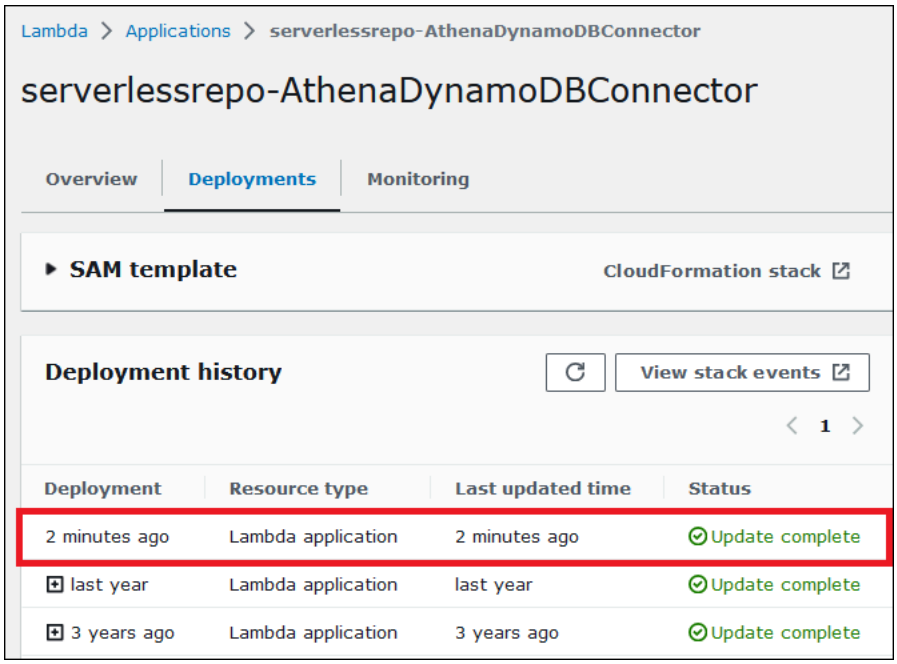
3. 在导航窗格中，选择 Data sources（数据来源）。
4. 在 Data sources（数据源）页面上，选择 Create data source（创建数据源）。
5. 选择要升级的数据来源，然后选择下一步。
6. 在连接详细信息部分中，选择创建 Lambda 函数。这将打开 Lambda 控制台，可在其中部署更新后的应用程序。



7. 由于您实际上并不是在新建数据来源，因此可以关闭 Athena 控制台选项卡。
8. 在连接器的 Lambda 控制台页面中，执行以下步骤：
   a. 确保已从应用程序名称中移除了 serverlessrepo- 前缀，然后将应用程序名称复制到应用程序名称字段。
   b. 将您的 Lambda 函数名称复制到 AthenaCatalogName 字段。在某些连接器中，此字段名为 LambdaFunctionName。
   c. 将您记录的环境变量复制到其相应的字段中。
9. 选择选项我确认此应用程序创建自定义 IAM 角色和资源策略，然后选择部署。

10. 要验证您的应用程序是否已更新，选择部署选项卡。

    部署历史记录部分显示您的更新已完成。



11. 要确认新版本号，您可以像以前一样展开 SAM 模板，找到 CodeUri，然后在键字段中检查连接器版本号。

现在，您可以使用更新后的连接器创建 Athena 联合查询。

编写联合查询

配置一个或多个数据连接器并将其部署到您的账户之后，您可以在 Athena 查询中进行使用。

查询单个数据源

本节中的示例假定您已配置 Amazon Athena CloudWatch 连接器 并将其部署到您的账户。使用其他连接器时，使用相同的方法进行查询。

创建使用 CloudWatch 连接器的 Athena 查询

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 在 Athena 查询编辑器中，创建在 FROM 子句中使用以下语法的 SQL 查询。

```
MyCloudwatchCatalog.database_name.table_name
```

示例

以下示例使用 Athena CloudWatch 连接器连接到 `/var/ecommerce-engine/order-processor` CloudWatch Logs 日志组中的 `all_log_streams` 视图。`all_log_streams` 视图是日志组中所有日志流的视图。示例查询将返回的行数限制为 100。

```
SELECT *
FROM "MyCloudwatchCatalog"."/var/ecommerce-engine/order-processor".all_log_streams
LIMIT 100;
```

以下示例解析与上一示例相同的视图中的信息。该示例提取顺序 ID 和日志级别，并筛选出所有 `INFO` 级别的消息。

```
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'
```

查询多个数据源

举一个更复杂的例子，假设一家电子商务公司使用以下数据来源存储与客户购买相关的数据：

- Amazon RDS for MySQL，用于存储产品目录数据
- Amazon DocumentDB，用于存储客户账户数据，例如电子邮件和配送地址
- Amazon DynamoDB，用于存储订单发货和追踪数据

假设该电子商务应用程序的数据分析师了解到，某些地区的送货时间受到当地天气状况的影响。分析师希望了解延迟送货的订单数量、受影响客户所在位置以及受影响最大的产品。分析师没有单独调查信息来源，而是使用 Athena 将数据联接到单个联合查询中。

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
```

查询联合视图

查询联合源时，可以使用视图对底层数据来源进行模糊处理，或者隐藏来自查询数据的其他分析师的复杂联接。

注意事项和限制

- 联合视图需要 Athena 引擎版本 3。
- 联合视图存储在 AWS Glue 中，而不是底层数据来源中。
- 使用联合目录创建的视图必须使用完全限定名称语法，如以下示例所示：

  ```
  "ddbcatalog"."default"."customers"
  ```

- 在联合源上运行查询的用户必须有权查询联合源。
- 联合视图需要 `athena:GetDataCatalog` 权限。有关更多信息，请参阅 [允许 Athena 联合查询的 IAM 权限策略示例]。

示例

以下示例针对存储在联合数据来源中的数据创建了一个名为 `customers` 的视图。

Example

```
CREATE VIEW customers AS
SELECT *
FROM my_federated_source.default.table
```

以下示例查询显示了一个引用 `customers` 视图而不是底层联合数据来源的查询。

Example

```
SELECT id, SUM(order_amount)
```

```
FROM customers
GROUP by 1
ORDER by 2 DESC
LIMIT 50
```

以下示例创建了一个名为 order_summary 的视图，该视图合并了来自联合数据来源和 Amazon S3 数据来源的数据。在已在 Athena 中创建的联合源中，视图使用 person 和 profile 表。在 Amazon S3 中，视图使用 purchase 和 payment 表。为了引用 Amazon S3，语句使用了关键字 awsdatacatalog。请注意，联合数据来源使用完全限定名称语法 *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
```

其他 资源

- 有关与其原始源分离且适用于多用户模型中的按需分析的联合视图示例，请参阅 AWS 大数据博客中的使用 Amazon Athena 和联合视图扩展数据网格。
- 有关在 Athena 中使用视图的更多信息，请参阅 使用视图。

运行联合传递查询

在 Athena 中，可以使用数据来源本身的查询语言对联合数据来源运行查询，并将完整查询下推到数据来源来执行查询。这些查询称为传递查询。要运行传递查询，可以在 Athena 查询中使用表函数。您可以将要在数据来源上运行的传递查询包含在表函数的其中一个参数中。传递查询会返回一个表，该表可以使用 Athena SQL 进行分析。

支持的连接器

以下 Athena 数据来源连接器支持传递查询。

- Azure Data Lake 存储
- 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

注意事项和限制

在 Athena 中使用传递查询时，请注意以下几点：

- 只有 Athena SELECT 语句或读取操作才支持查询传递。
- 必须在外部查询（即调用表函数的查询）目录的上下文中运行传递查询。
- 查询性能可能因数据来源的配置而异。
- 视图不支持传递查询。

语法

常规 Athena 查询传递语法如下所示：

```
SELECT * FROM TABLE(system.function_name(arg1 => 'arg1Value'[, arg2 =>
 'arg2Value', ...]))
```

对于大多数数据来源，第一个也是唯一的参数为 query，后跟箭头运算符 => 和查询字符串。

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

为简单起见，可选的命名参数 query 和箭头运算符 => 均可省略。

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

如果数据来源需要的不仅仅是查询字符串，请按照数据来源所需的顺序使用命名参数。例如，表达式 *arg1* => '*arg1Value*' 包含第一个参数及其值。*arg1* 名称与数据来源相关，可能因连接器而异。

```
SELECT * FROM TABLE(
        system.query(
            arg1 => 'arg1Value',
            arg2 => 'arg2Value',
            arg3 => 'arg3Value'
        ));
```

有关用于特定连接器的确切语法的信息，请参阅相应连接器的页面。

引号用法

必须用单引号将参数值（包含传递的查询字符串）括起来，如下例所示：

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

如果使用双引号将查询字符串括起来，查询会失败。以下查询失败，显示错误消息：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"))
```

要转义单引号，请在原始单引号中添加单引号（例如由 terry's_group 变为 terry''s_group）。

示例

以下示例查询将查询下推到数据来源。该查询选择了 customer 表中的所有列，将结果限制为 10。

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

以下语句运行相同的查询，但删除了可选的命名参数 query 和箭头运算符 =>。

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

Athena 和联合表名限定词

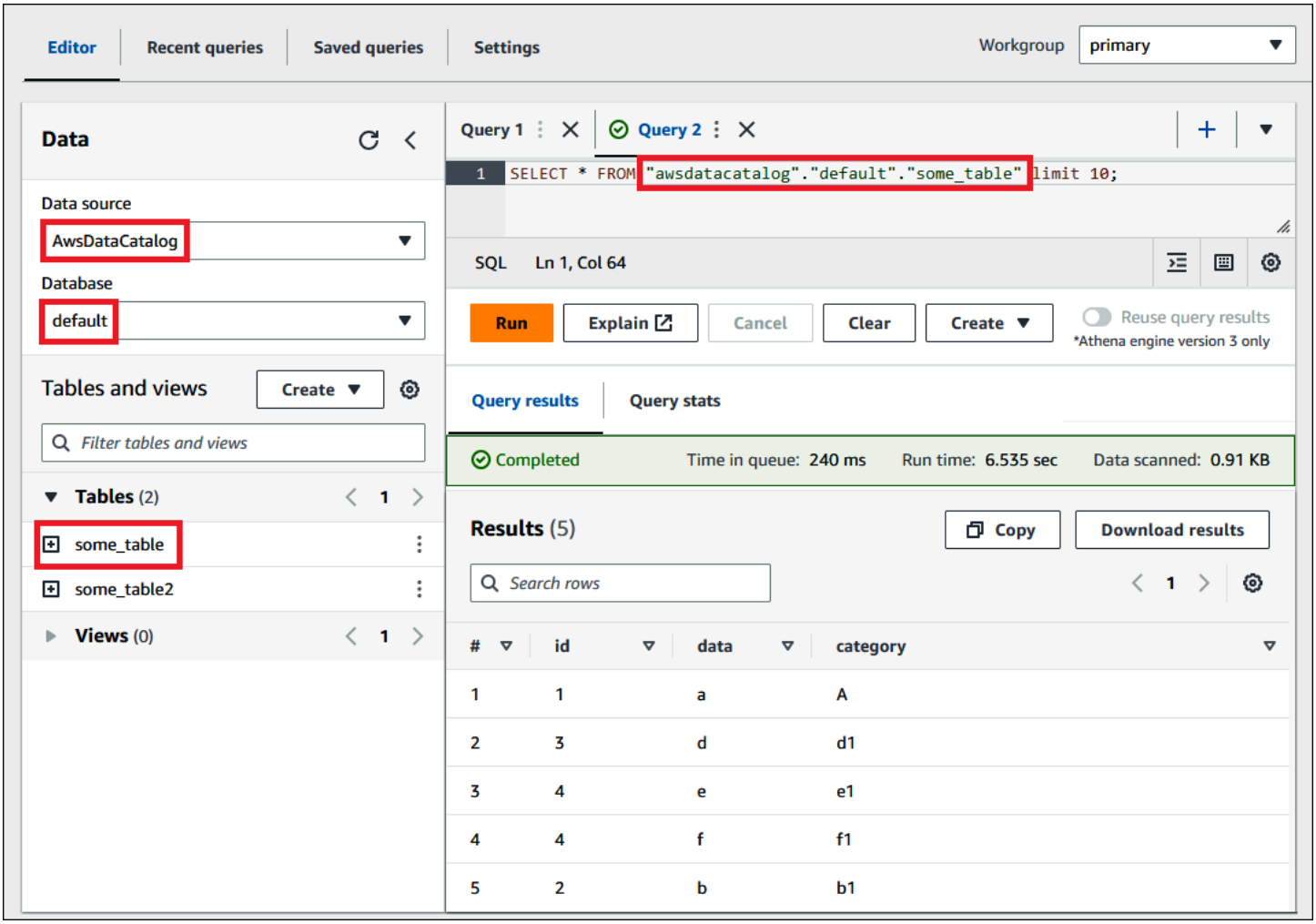
Athena 使用以下术语来指代数据对象的层次结构：

- 数据来源 - 一组数据库
- 数据库 - 一组表
- 表 - 按一组行或列组织的数据

有时，这些对象也会用替代但等效的名称来指代，如下所示：

- 数据来源有时也称为目录。
- 数据库有时也称为架构。

Athena 控制台中的以下示例查询使用 awsdatacatalog 数据来源、数据库 default 和表 some_table。



联合数据来源中的术语

在查询联合数据来源时，请注意，底层数据来源使用的术语可能与 Athena 不同。在编写联合查询时，请记住这一区别。以下各节介绍了 Athena 中的数据对象术语与联合数据来源中的数据对象术语的对应关系。

Amazon Redshift

Amazon Redshift 数据库是一组 Redshift 架构，其中包含一组 Redshift 表。

Athena	Redshift
Redshift 数据来源	配置为指向 Redshift `database` 的 Redshift 连接器 Lambda 函数。
`data_source.database.table`	`database.schema.table`

示例查询

```
SELECT * FROM
Athena_Redshift_connector_data_source.Redshift_schema_name.Redshift_table_name
```

有关此连接器的更多信息，请参阅 Amazon Athena Redshift 连接器。

Cloudera Hive

Cloudera Hive 服务器或集群是一组 Cloudera Hive 数据库，其中包含一组 Cloudera Hive 表。

Athena	Hive
Cloudera Hive 数据来源	Cloudera Hive 连接器 Lambda 函数配置为指向 Cloudera Hive `server`。
`data_source.database.table`	`server.database.table`

示例查询

```
SELECT * FROM
Athena_Cloudera_Hive_connector_data_source.Cloudera_Hive_database_name.Cloudera_Hive_table_name
```

有关此连接器的更多信息，请参阅 Amazon Athena Cloudera Hive 连接器。

Cloudera Impala

Impala 服务器或集群是一组 Impala 数据库，其中包含一组 Impala 表。

Athena	Impala
Impala 数据来源	Impala 连接器 Lambda 函数配置为指向 Impala `server`。
`data_source.database.table`	`server.database.table`

示例查询

```
SELECT * FROM
Athena_Impala_connector_data_source.Impala_database_name.Impala_table_name
```

有关此连接器的更多信息，请参阅 Amazon Athena Cloudera Impala 连接器。

MySQL

MySQL 服务器是一组 MySQL 数据库，其中包含一组 MySQL 表。

Athena	MySQL
MySQL 数据来源	MySQL 连接器 Lambda 函数配置为指向 MySQL `server`。
`data_source.database.table`	`server.database.table`

示例查询

```
SELECT * FROM
Athena_MySQL_connector_data source.MySQL_database_name.MySQL_table_name
```

有关此连接器的更多信息，请参阅 Amazon Athena MySQL 连接器。

Oracle

Oracle 服务器（或数据库）是一组 Oracle 架构，其中包含一组 Oracle 表。

Athena	Oracle
Oracle 数据来源	Oracle 连接器 Lambda 函数配置为指向 Oracle `server`。
`data_source.database.table`	`server.schema.table`

示例查询

```
SELECT * FROM
```

```
Athena_Oracle_connector_data_source.Oracle_schema_name.Oracle_table_name
```

有关此连接器的更多信息，请参阅 Amazon Athena Oracle 连接器。

Postgres

Postgres 服务器（或集群）是一组 Postgres 数据库。Postgres 数据库是一组 Postgres 架构，其中包含一组 Postgres 表。

Athena	Postgres
Postgres 数据来源	Postgres 连接器 Lambda 函数配置为指向 Postgres `server` 和 `database`。
`data_source.database.table`	`server.database.schema.table`

示例查询

```
SELECT * FROM
Athena_Postgres_connector_data_source.Postgres_schema_name.Postgres_table_name
```

有关此连接器的更多信息，请参阅 Amazon Athena PostgreSQL 连接器。

使用 Athena Query Federation SDK 编写数据来源连接器

要编写自己的数据源连接器，可以使用 Athena Query Federation 软件开发工具包。Athena Query Federation 软件开发工具包定义了一组接口和网络协议，使用它们让 Athena 可以将其查询执行计划的一部分委派给您编写和部署的代码。软件开发工具包包括一个连接器套件和一个示例连接器。

您还可以自定义 Amazon Athena 的预构建连接器供您自己使用。您可以修改来自 GitHub 的源代码的副本，然后使用连接器发布工具创建您自己的 AWS Serverless Application Repository 软件包。以这种方式部署连接器后，您可以在 Athena 查询中使用它。

有关如何下载软件开发工具包的信息和编写自己的连接器的详细说明，请参阅 GitHub 上的 Example Athena connector。

适用于 Apache Spark 的 Athena 数据来源连接器

某些 Athena 数据来源连接器可用作 Spark DSV2 连接器。Spark DSV2 连接器名称带有 `-dsv2` 后缀（例如 `athena-dynamodb-dsv2`）。

下面是当前可用的 DSV2 连接器、其 Spark `.format()` 类名称以及指向其相应的 Amazon Athena 联合查询文档的链接：

DSV2 连接器	Spark .format() 类名称	文档
athena-cloudwatch-dsv2	`com.amazonaws.athena.connectors.dsv2.cloudwatch.CloudwatchTableProvider`	CloudWatch
athena-cloudwatch-metrics-dsv2	`com.amazonaws.athena.connectors.dsv2.cloudwatch.metrics.CloudwatchMetricsTableProvider`	CloudWatch 指标
athena-aws-cmdb-dsv2	`com.amazonaws.athena.connectors.dsv2.aws.cmdb.AwsCmdbTableProvider`	CMDB
athena-dynamodb-dsv2	`com.amazonaws.athena.connectors.dsv2.dynamodb.DDBTableProvider`	DynamoDB

要下载 DSV2 连接器的 `.jar` 文件，请访问 Amazon Athena Query Federation DSV2 GitHub 页面并查看发布版本、发布版本 ***<version>***、资产部分。

为 Spark 指定 jar

要将 Athena DSV2 连接器与 Spark 配合使用，将连接器的 `.jar` 文件提交到您正在使用的 Spark 环境。以下几节介绍了具体的情况。

Athena for Spark

有关向 Amazon Athena for Apache Spark 添加自定义 `.jar` 文件和自定义配置的信息，请参阅 添加 JAR 文件和自定义 Spark 配置。

常规 Spark

要将连接器 .jar 文件传递给 Spark，使用 spark-submit 命令并在 --jars 选项中指定 .jar 文件，如以下示例所示：

```
spark-submit \
  --deploy-mode cluster \
  --jars https://github.com/awslabs/aws-athena-query-federation-dsv2/releases/
download/some_version/athena-dynamodb-dsv2-some_version.jar
```

Amazon EMR Spark

要在 Amazon EMR 上运行包含 --jars 参数的 spark-submit 命令，必须向 Amazon EMR Spark 集群添加一个步骤。有关如何在 Amazon EMR 上使用 spark-submit 的详细信息，请参阅《Amazon EMR 版本指南》中的添加 Spark 步骤。

AWS Glue ETL Spark

对于 AWS Glue ETL，可以将 .jar 文件的 GitHub.com URL 传递到 aws glue start-job-run 命令的 --extra-jars 参数。AWS Glue 文档将 --extra-jars 参数描述为采用 Amazon S3 路径，但该参数也可以采用 HTTPS URL。有关更多信息，请参阅《AWS Glue 开发人员指南》中的作业参数参考。

在 Spark 上查询连接器

要在 Apache Spark 上提交现有 Athena 联合查询的等效项，使用 spark.sql() 函数。例如，假设您拥有以下要在 Apache Spark 上使用的 Athena 查询。

```
SELECT somecola, somecolb, somecolc
FROM ddb_datasource.some_schema_or_glue_database.some_ddb_or_glue_table
WHERE somecola > 1
```

要使用 Amazon Athena DynamoDB DSV2 连接器在 Spark 上执行相同的查询，使用以下代码：

```
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
''')
```

指定参数

Athena 数据来源连接器的 DSV2 版本使用与相应的 Athena 数据来源连接器相同的参数。有关参数信息，请参阅相应的 Athena 数据来源连接器的文档。

在 Pyspark 代码中，使用以下语法配置参数。

```
spark.read.option("athena.connectors.conf.parameter", "value")
```

例如，以下代码将 Amazon Athena DynamoDB 连接器 `disable_projection_and_casing` 参数设置为 `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())
```

用于访问数据目录的 IAM policy

要控制对数据目录的访问，请使用资源级 IAM 权限或基于身份的 IAM policy。

以下是 Athena 的特定过程。

有关 IAM 的特定信息，请参阅本节末尾列出的链接。有关示例 JSON 数据目录策略的信息，请参阅 数据目录示例策略。

在 IAM 控制台中使用可视化编辑器创建数据目录策略

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧的导航窗格中，选择 Policies (策略)，然后选择 Create policy (创建策略)。

3. 在可视化编辑器选项卡上，选择选择服务。然后，选择要添加到策略的 Athena。
4. 选择 Select actions (选择操作)，然后选择要添加到策略的操作。可视化编辑器会显示 Athena 中可用的操作。有关更多信息，请参阅《服务授权参考》中的 Amazon Athena 的操作、资源和条件键。
5. 选择 add actions (添加操作) 以键入特定操作，或使用通配符 (*) 指定多个操作。

   预设情况下，您创建的策略允许执行选择的操作。如果您在 Athena 中选择对 `datacatalog` 资源执行一个或多个支持资源级权限的操作，则编辑器会列出 `datacatalog` 资源。
6. 选择资源，为您的策略指定特定的数据目录。有关 JSON 数据目录策略的示例，请参阅数据目录示例策略。
7. 指定 `datacatalog` 资源，如下所示：

   ```
   arn:aws:athena:<region>:<user-account>:datacatalog/<datacatalog-name>
   ```
8. 选择 Review policy (查看策略)，然后为您创建的策略键入 Name (名称) 和 Description (描述)（可选）。查看策略摘要以确保您已授予所需的权限。
9. 选择创建策略可保存您的新策略。
10. 将此基于身份的策略附加到用户、组或角色，并指定他们可访问的 `datacatalog` 资源。

有关详细信息，请参阅服务授权参考和 IAM 用户指南中的以下主题：

- Amazon Athena 的操作、资源和条件键
- 使用可视化编辑器创建策略
- 添加和移除 IAM policy
- 控制对资源的访问

有关 JSON 数据目录策略的示例，请参阅数据目录示例策略。

有关 AWS Glue 权限和 AWS Glue 爬网程序权限的信息，请参阅《AWS Glue 开发人员指南》中的 为 AWS Glue 设置 IAM 权限 和 爬网程序先决条件。

有关 Amazon Athena 操作的完整列表，请参阅 Amazon Athena API 参考中的 API 操作名称。

数据目录示例策略

此部分包含可用于允许对数据目录执行各种操作的示例策略。

数据目录是由 Athena 管理的 IAM 资源。因此，如果您的数据目录策略使用将 datacatalog 用作输入的操作，则您必须指定数据目录的 ARN，如下所示：

```
"Resource": [arn:aws:athena:<region>:<user-account>:datacatalog/<datacatalog-name>]
```

<datacatalog-name> 是数据目录的名称。例如，对于名为 test_datacatalog 的数据目录，请将其指定为资源，如下所示：

```
"Resource": ["arn:aws:athena:us-east-1:123456789012:datacatalog/test_datacatalog"]
```

有关 Amazon Athena 操作的完整列表，请参阅 Amazon Athena API 参考中的 API 操作名称。有关 IAM policy 的更多信息，请参阅《IAM 用户指南》中的 使用可视化编辑器创建策略。有关为工作组创建 IAM policy 的更多信息，请参阅 用于访问数据目录的 IAM policy。

- 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 对所有数据目录具有完全访问权限的策略示例

以下策略允许对账户中可能存在的所有数据目录资源进行完全访问。对于您账户中必须为其他所有用户监督和管理数据目录的那些用户，建议您使用此策略。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "athena:*"
         ],
         "Resource":[
            "*"
         ]
```

```
      }
   ]
}
```

Example 对指定数据目录具有完全访问权限的策略示例

以下策略允许对单个名为 datacatalogA 的特定数据目录资源进行完全访问。对于对特定数据目录具有完全控制权限的用户，您可以使用此策略。

```
{
   "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 用于查询指定数据目录的策略示例

在以下策略中，允许用户对指定的 datacatalogA 运行查询。不允许用户对数据目录本身执行管理任务，例如，更新或删除它。

```
{
   "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 对指定数据目录执行管理操作的策略示例

在以下策略中，允许用户创建、删除、获取详细信息并更新数据目录 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 列出数据目录的策略示例

以下策略允许所有用户列出所有数据目录：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListDataCatalogs"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

Example 对数据目录执行元数据操作的策略示例

以下策略允许对数据目录执行元数据操作：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:GetDatabase",
                "athena:GetTableMetadata",
                "athena:ListDatabases",
                "athena:ListTableMetadata"
            ],
            "Resource": "*"
        }
    ]
}
```

管理数据来源

您可以使用 Athena 控制台的 Data Sources（数据来源）页面，管理您创建的数据来源。

查看数据来源

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。

3. 在导航窗格中，选择 Data sources（数据来源）。
4. 从数据来源列表中，选择要查看的数据来源的名称。

> **Note**
>
> Data source name（数据来源名称）列中的项目对应于 ListDataCatalogs API 操作和 list-data-catalogs CLI 命令的输出。

编辑数据来源

1. 在 Data Sources（数据来源）页面上，执行以下操作：
   - 选择目录名称旁边的按钮，然后依次选择 Actions（操作）、Edit（编辑）。
   - 选择数据来源的名称。然后在详细信息页面上，依次选择 Actions（操作）、Edit（编辑）。
2. 在 Edit（编辑）页面上，您可以为数据来源选择另一个 Lambda 函数，或添加自定义标签。有关标签的更多信息，请参阅 为 Athena 资源添加标签。
3. 选择保存。
4. 要编辑 AwsDataCatalog 数据来源，请选择 AwsDataCatalog 链接以打开其详细信息页面。然后，在详细信息页面上，选择指向您可以在其中编辑目录的 AWS Glue 控制台的链接。

共享数据来源

有关共享数据来源的信息，请访问以下链接。

- 有关非 Hive Lambda 的数据来源，请参阅 启用跨账户联合查询。
- 有关 AWS Glue Data Catalog，请参阅 授予 AWS Glue 数据目录跨账户访问权限。

删除数据来源

1. 在 Data Sources（数据来源）页面上，执行以下操作：
   - 选择目录名称旁边的按钮，然后依次选择 Actions（操作）、Delete（删除）。
   - 选择数据来源的名称，然后在详细信息页面上，依次选择 Actions（操作）、Delete（删除）。

   > **Note**
   > AwsDataCatalog 是您账户中的默认数据来源，不能删除。

   系统会发出警告，当您删除数据来源时，其相应的数据目录、表和视图将从查询编辑器中删除。使用数据来源的已保存查询不再在 Athena 中运行。
2. 要确认删除，请键入数据来源的名称，然后选择 Delete（删除）。

在 Athena 中使用 Amazon DataZone

您可以使用 Amazon DataZone 跨组织边界大规模共享、搜索和发现数据。DataZone 简化了您在 Athena、AWS Glue 和 AWS Lake Formation 等 AWS 分析服务中的体验。例如，如果您在不同的数据来源中有 PB 级数据，则可使用 Amazon DataZone 根据人员、数据和工具分组构建业务用例。有关更多信息，请参阅 What is Amazon DataZone？。

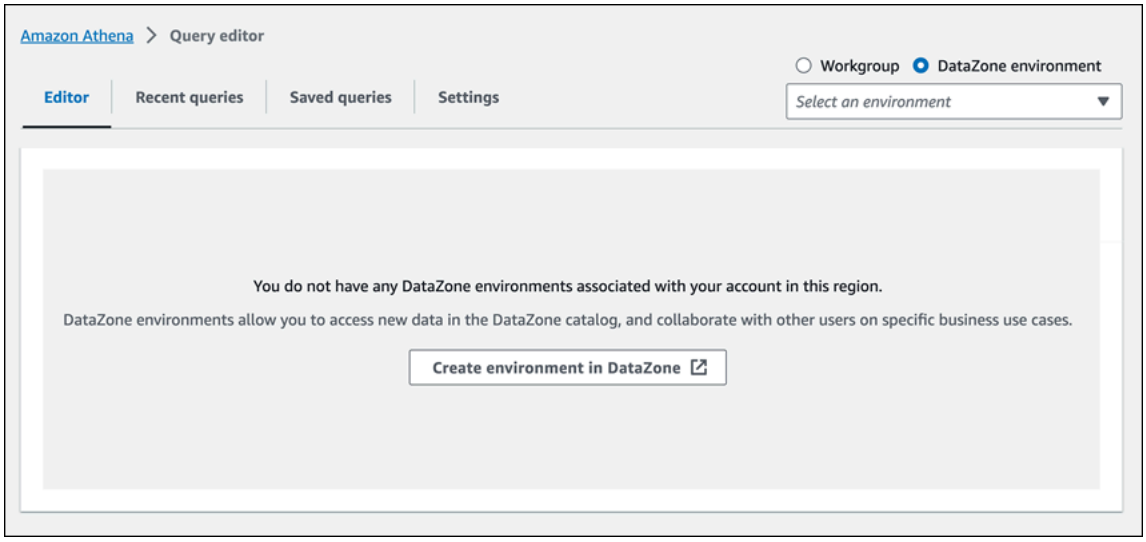
在 Athena 中，您可以使用查询编辑器来访问和查询 DataZone 环境。DataZone 环境指定了 DataZone 项目和域的组合。从 Athena 控制台使用 DataZone 环境时，您将代入 DataZone 环境的 IAM 角色，并且只能看到属于该环境的数据库和表。权限由您在 DataZone 中指定的角色决定。

在 Athena 中，您可以使用查询编辑器页面上的 DataZone 环境选择器来选择 DataZone 环境。

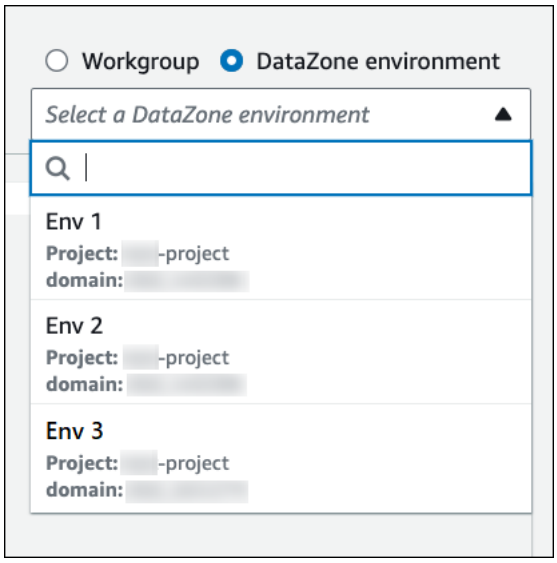
在 Athena 中打开 DataZone 环境

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 在 Athena 控制台右上角的工作组旁边，选择 DataZone 环境。

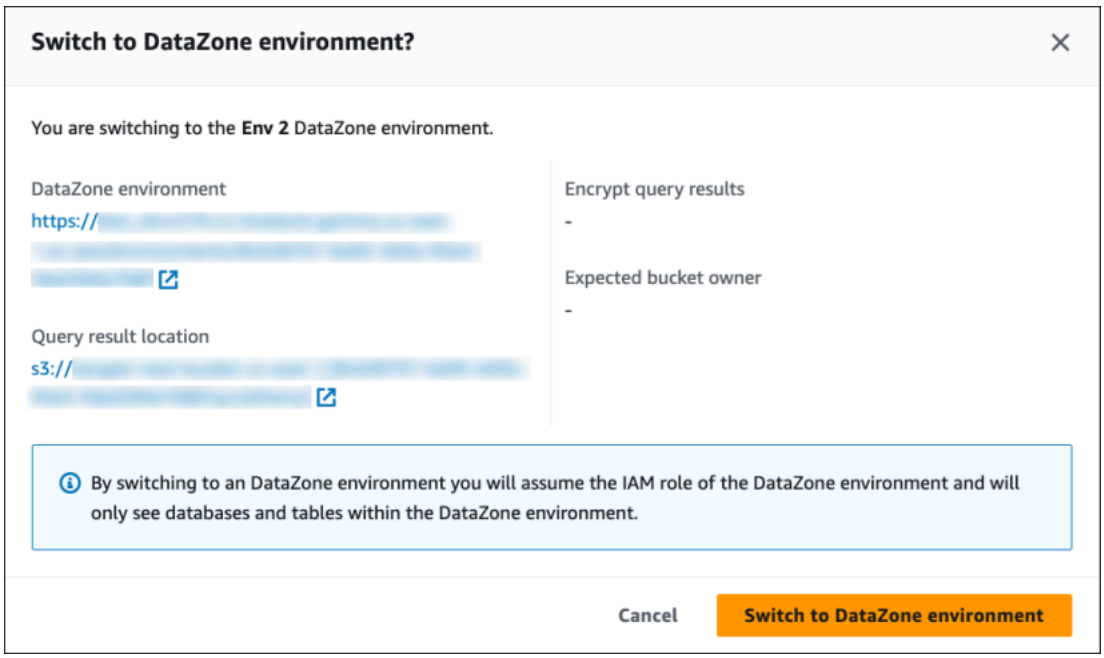
> **Note**
>
> 只有 DataZone 中有一个或多个可用域时，才会出现 DataZone 环境选项。



3. 使用 DataZone 环境选择器选择一个 DataZone 环境。



4. 在切换到 DataZone 环境对话框中，确认该环境是您想要的环境，然后选择切换到 DataZone 环境。



有关开始使用 DataZone 和 Athena 的更多信息，请参阅《Amazon DataZone User Guide》中的 Getting started 教程。

通过 ODBC 和 JDBC 驱动程序连接到 Amazon Athena

要使用业务智能工具浏览和可视化数据，请下载、安装并配置 ODBC (开放式数据库连接) 或 JDBC (Java 数据库连接) 驱动程序。

主题

- 通过 JDBC 连接到 Amazon Athena
- 通过 ODBC 连接到 Amazon Athena

另请参阅以下 AWS 知识中心和 AWS 大数据博客主题：

- 在使用 JDBC 驱动程序连接到 Athena 时，如何使用我的 IAM 角色证书或切换到另一个 IAM 角色？
- 设置在 ADFS 和 AWS 之间的信任并使用 ODBC 驱动程序借助 Active Directory 凭证连接到 Amazon Athena

通过 JDBC 连接到 Amazon Athena

Amazon Athena 提供两个 JDBC 驱动程序版本，即 2.x 和 3.x。Athena JDBC 3.x 驱动程序是新一代的驱动程序，具有更好的性能和兼容性。JDBC 3.x 驱动程序支持直接从 Amazon S3 读取查询结果，此举可提高使用大型查询结果的应用程序的性能。该新驱动程序还减少了第三方依赖项，让与商业智能工具和自定义应用程序的集成变得更加容易。在大多数情况下，您都可以直接使用该新驱动程序，无需对现有配置进行任何更改，或者只需进行极少更改。

- 要下载 JDBC 3.x 驱动程序，请参阅 Athena JDBC 3.x 驱动程序。
- 要下载 JDBC 2.x 驱动程序，请参阅 Athena JDBC 2.x 驱动程序。

主题

Athena JDBC 3.x 驱动程序

您可以使用 Athena JDBC 驱动程序从许多第三方 SQL 客户端工具和自定义应用程序连接到 Amazon Athena。

系统要求

- Java 8（或更高版本）的运行时系统环境
- 至少 20MB 可用磁盘空间

注意事项和限制

以下是 Athena JDBC 3.x 驱动程序的一些注意事项和限制。

- 日志记录 – 3.x 驱动程序使用 SLF4J，这是一个抽象层，支持在运行时系统中使用多种日志记录系统中的任一系统。
- 加密 – 当使用具有 CSE_KMS 加密选项的 Amazon S3 提取器时，Amazon S3 客户端无法解密在 Amazon S3 存储桶中存储的结果。如果您需要 CSE_KMS 加密，则可以继续使用流式传输提取器。计划支持将 CSE_KMS 加密用于 Amazon S3 提取器。

JDBC 3.x 驱动程序下载

本节包含 JDBC 3.x 驱动程序的下载和许可证信息。

> **Important**
>
> 使用 JDBC 3.x 驱动程序时，请务必注意以下要求：
>
> - 保留端口 444 — 保留 Athena 用于流式传输查询结果的端口 444，对出站流量开放。当您使用 PrivateLink 终端节点连接到 Athena 时，请确保附加到 PrivateLink 终端节点的安全组在端口 444 上对入站流量开放。
> - athena:GetQueryResultsStream 策略 — 将 `athena:GetQueryResultsStream` 策略操作添加到使用 JDBC 驱动程序的 IAM 主体。此策略操作并不通过 API 直接公开。它仅作为流式传输结果的一部分与 ODBC 和 JDBC 驱动程序配合使用。有关策略示例，请参阅AWS 托管策略：AWSQuicksightAthenaAccess。

要下载 Amazon Athena 3.x JDBC 驱动程序，请访问以下链接。

JDBC 驱动程序 uber jar

以下下载内容将驱动程序及其所有依赖项打包到同一个 `.jar` 文件中。此下载通常用于第三方 SQL 客户端。

3.2.0 uber jar

JDBC 驱动程序 lean jar

以下下载内容是一个 `.zip` 文件，其中包含驱动程序的 lean `.jar` 文件和驱动程序依赖项的单独 `.jar` 文件。此下载通常用于依赖项可能与驱动程序所使用的依赖项冲突的自定义应用程序。如果您想选择要在 lean jar 中包含哪些驱动程序依赖项，以及在自定义应用程序已经包含一个或多个驱动程序依赖项时应排除哪些驱动程序依赖项，则此下载非常有用。

3.2.0 lean jar

许可证

以下链接包含 JDBC 3.x 驱动程序的许可协议。

许可证

主题

JDBC 3.x 驱动程序入门

参照本节中的信息开始使用 Amazon Athena JDBC 3.x 驱动程序。

主题

安装说明

您可以在自定义应用程序中或通过第三方 SQL 客户端使用 JDBC 3.x 驱动程序。

在自定义应用程序中

下载包含驱动程序 jar 及其依赖项的 `.zip` 文件。每个依赖项都有自己的 `.jar` 文件。将驱动程序 jar 作为依赖项添加到自定义应用程序中。根据您是否已经将这些依赖项从其他来源添加到应用程序中，选择性地添加驱动程序 jar 的依赖项。

在第三方 SQL 客户端中

下载驱动程序 uber jar 文件，然后按照客户端的说明将其添加到该第三方 SQL 客户端。

运行驱动程序

要运行驱动程序，您可以使用自定义应用程序或第三方 SQL 客户端。

在自定义应用程序中

使用 JDBC 接口与程序中的 JDBC 驱动程序进行交互。以下代码显示了一个自定义 Java 应用程序示例。

```
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
}
```

在第三方 SQL 客户端中

请按照自己正在使用的 SQL 客户端的文档进行操作。您通常可以使用 SQL 客户端的图形用户界面来输入和提交查询，查询结果显示在同一个界面中。

配置驱动程序

您可以使用连接参数来配置 Amazon Athena JDBC 驱动程序。有关受支持的连接参数，请参阅 Amazon Athena JDBC 3.x 连接参数。

在自定义应用程序中

要在自定义应用程序中设置 JDBC 驱动程序的连接参数，请执行以下任一操作：

- 将参数名称及其值添加到 `Properties` 对象中。在调用 `Connection#connect` 时，请将该对象与 URL 一起传递。有关示例，请参阅运行驱动程序中的示例 Java 应用程序。
- 在连接字符串（URL）中，使用以下格式将参数名称及其值直接添加到协议前缀之后。

  ```
  <parameterName>=<parameterValue>;
  ```

  在每个参数名/参数值对的末尾使用分号，分号后面不留空格，如下例所示。

```
String url = "jdbc:athena://WorkGroup=primary;Region=us-east-1;...;";AthenaDriver
 driver = new AthenaDriver();Connection connection = driver.connect(url, null);
```

> **Note**
>
> 如果在连接字符串和 Properties 对象中都指定了参数，则优先使用连接字符串中的值。不建议在这两个地方指定相同的参数。

- 将参数值作为参数添加到 AthenaDataSource 的方法中，如下例所示。

```
AthenaDataSource dataSource = new AthenaDataSource();
    dataSource.setWorkGroup("primary");
    dataSource.setRegion("us-east-2");
    ...
    Connection connection = dataSource.getConnection();
    ...
```

在第三方 SQL 客户端中

按照自己正在使用的 SQL 客户端的说明进行操作。客户端通常会提供一个图形用户界面，用于输入参数名称及其值。

从 Athena JDBC 版本 2 驱动程序升级

大多数 JDBC 版本 3 的连接参数都向后兼容版本 2（Simba）JDBC 驱动程序。这表示版本 2 的连接字符串可以在版本 3 的驱动程序中重用。不过，某些连接参数已发生更改。此处对这些更改进行了介绍。在升级到 JDBC 驱动程序版本 3 时，如有必要，请更新现有配置。

驱动程序类

部分 BI 工具会要求您提供 JDBC 驱动程序 .jar 文件中的驱动程序类。大多数工具都会自动找到这个类。这个类在版本 3 驱动程序中的完全限定名称是 com.amazon.athena.jdbc.AthenaDriver。在版本 2 驱动程序中，这个类是 com.simba.athena.jdbc.Driver。

连接字符串

版本 3 驱动程序在 JDBC 连接字符串 URL 的开头使用 jdbc:athena:// 作为协议。版本 3 驱动程序也支持版本 2 协议 jdbc:awsathena://，但不推荐使用版本 2 协议。为避免未定义的行为，

如果版本 2（或接受以 jdbc:awsathena:// 开头的连接字符串的任何其他驱动程序）已注册到 DriverManager 类，则版本 3 不接受以 jdbc:awsathena:// 开头的连接字符串。

凭证提供程序

版本 2 驱动程序使用完全限定名称来标识不同的凭证提供程序（例如：com.simba.athena.amazonaws.auth.DefaultAWSCredentialsProviderChain）。版本 3 驱动程序使用较短的名称（例如：DefaultChain）。每个凭证提供程序的相应部分中会对新名称进行描述。

为版本 2 驱动程序编写的自定义凭证提供程序需要进行修改，以便版本 3 驱动程序实现来自新 AWS SDK for Java 的 AwsCredentialsProvider 接口，而不是来自以前 AWS SDK for Java 的 AWSCredentialsProvider 接口。

JDBC 3.x 驱动程序不支持 PropertiesFileCredentialsProvider。该提供程序曾在 JDBC 2.x 驱动程序中使用，但属于适用于 Java 的 AWS SDK 的先前版本，该版本的支持即将结束。要在 JDBC 3.x 驱动程序中实现相同的功能，请改用 AWS 配置文件凭证 提供程序。

日志级别

下表显示了 JDBC 版本 2 和版本 3 驱动程序中 LogLevel 参数的差异。

JDBC 驱动程序版本	参数名称	参数类型	默认值	可能的值	连接字符串示例
v2	LogLevel	可选	0	0-6	LogLevel=6;
v3	LogLevel	可选	TRACE	OFF、ERROR、WARN、INFO、DEBUG、TRACE	LogLevel=INFO;

查询 ID 检索

在版本 2 驱动程序中，您将 Statement 实例解包到 com.interfaces.core.IStatementQueryInfoProvider。该接口有两种方法：#getPReparedQueryId 和 #getQueryId。您可以使用这些方法来获取已运行查询的查询执行 ID。

在版本 3 驱动程序中，您将 Statement、PreparedStatement 和 ResultSet 实例解包到 com.amazon.athena.jdbc.AthenaResultSet 接口。该接口有一种方法：#getQueryExecutionId。

Amazon Athena JDBC 3.x 连接参数

这里将受支持的连接参数分为三个部分：基本连接参数、高级连接参数 和 身份验证连接参数。“高级连接参数”和“身份验证连接参数”部分包含将相关参数分组在一起的子部分。

主题

基本连接参数

以下各节旨在介绍 JDBC 3.x 驱动程序的基本连接参数。

区域

将运行查询的 AWS 区域。有关区域列表，请参阅 Amazon Athena endpoints and quotas。

参数名称	别名	参数类型	默认值
区域	AwsRegion（已弃用）	必填项（但如果未提供，将使用 DefaultAwsRegionProviderChain 进行搜索）	none

目录

包含将使用驱动程序访问的数据库和表的目录。有关目录的信息，请参阅 DataCatalog。

参数名称	别名	参数类型	默认值
目录	none	可选	AwsDataCatalog

数据库

将运行查询的数据库。未使用数据库名称显示限定的表将解析到此数据库。有关数据库的信息，请参阅 Database。

参数名称	别名	参数类型	默认值
数据库	架构	可选	默认值

工作组

将运行查询的工作组。有关工作组的信息，请参阅 WorkGroup。

参数名称	别名	参数类型	默认值
工作组	none	可选	主

输出位置

在 Amazon S3 中存储查询结果的位置。有关输出位置的信息，请参阅 ResultConfiguration。

参数名称	别名	参数类型	默认值
OutputLocation	S3OutputLocation（已弃用）	必填项（除非工作组指定输出位置）	none

高级连接参数

以下各节旨在介绍 JDBC 3.x 驱动程序的高级连接参数。

主题

- 结果加密参数
- 结果提取参数
- 查询结果重用参数
- 查询执行轮询参数

- 端点覆盖参数
- 代理配置参数
- 日志记录参数
- 应用程序名称
- 连接测试
- 重试次数

结果加密参数

请注意以下几点：

- 当 EncryptionOption 为 SSE_KMS 或 CSE_KMS 时，必须指定 AWS KMS 密钥。
- 当未指定 EncryptionOption 或当 EncryptionOption 为 SSE_S3 时，无法指定 AWS KMS 密钥。

加密选项

查询结果存储在 Amazon S3 中时使用的加密类型。有关查询结果加密的信息，请参阅《Amazon Athena API Reference》中的 EncryptionConfiguration。

参数名称	别名	参数类型	默认值	可能的值
EncryptionOption	S3OutputEncOption（已弃用）	可选	none	SSE_S3、SSE_KMS、CSE_KMS

KMS 密钥

KMS 密钥 ARN 或 ID（如果选择 SSE_KMS 或 CSE_KMS 作为加密选项）。有关更多信息，请参阅《Amazon Athena API Reference》中的 EncryptionConfiguration。

参数名称	别名	参数类型	默认值
KmsKey	S3OutputEncKMSKey（已弃用）	可选	none

结果提取参数

结果提取器

将用于下载查询结果的提取器。

默认的结果提取器 S3 会直接从 Amazon S3 下载查询结果，无需使用 Athena API。在大多数情况下，这是最快的选择。如果查询结果是使用 CSE_KMS 加密的，或者允许用户访问查询结果的策略仅允许使用 s3:CalledVia 调用来自 Athena 的内容，则此选项不可用。

参数名称	别名	参数类型	默认值	可能的值
ResultFetcher	none	可选	S3	S3、GetQueryResults、GetQueryResultsStream

> **Note**
>
> 在 JDBC 2.x 驱动程序中，UseResultsetStreaming = 1 设置将该驱动程序配置为使用结果集流式处理 API。在 JDBC 3.x 驱动程序中，等效设置为 ResultFetcher=GetQueryResultsStream。

提取大小

此参数的值用作内部缓冲区的最小值，也在提取结果时用作目标页面大小。值 0（零）表示驱动程序应使用其默认值，如下所述。最大值为 1,000,000。

参数名称	别名	参数类型	默认值
FetchSize	RowsToFetchPerBlock（已弃用）	可选	0

- GetQueryResults 提取器将始终使用 1,000 的页面大小，因为这是 API 调用支持的最大值。当提取大小超出 1,000 时，为了填充超过最小值的缓冲区，会连续进行多次 API 调用。

- `GetQueryResultsStream` 提取器将使用配置的提取大小或默认值 10,000 作为页面大小。
- `S3` 提取器将使用配置的提取大小或默认值 10,000 作为页面大小。

查询结果重用参数

启用结果重用

指定在运行查询时是否可以重用同一查询的先前结果。有关查询结果重用的信息，请参阅 ResultReuseByAgeConfiguration。

参数名称	别名	参数类型	默认值
EnableResultReuseByAge	none	可选	FALSE

结果重用最长使用期限

Athena 应考虑的先前查询结果的重用最长使用期限（以分钟为单位）。有关结果重用最长使用期限的信息，请参阅 ResultReuseByAgeConfiguration。

参数名称	别名	参数类型	默认值
MaxResultReuseAgeInMinutes	none	可选	60

查询执行轮询参数

查询执行的最小轮询间隔

在 Athena 中轮询查询执行状态之前要等待的最短时间（以毫秒为单位）。

参数名称	别名	参数类型	默认值
MinQueryExecutionPollingIntervalMillis	MinQueryExecutionPollingInterval（已弃用）	可选	100

查询执行的最大轮询间隔

在 Athena 中轮询查询执行状态之前要等待的最长时间（以毫秒为单位）。

参数名称	别名	参数类型	默认值
MaxQueryExecutionPollingIntervalMillis	MaxQueryExecutionPollingInterval（已弃用）	可选	5000

查询执行轮询间隔乘数

用于增加轮询期的系数。默认情况下，轮询将以 `MinQueryExecutionPollingIntervalMillis` 的值开始，每次轮询时都翻倍，直到达到 `MaxQueryExecutionPollingIntervalMillis` 的值。

参数名称	别名	参数类型	默认值
QueryExecutionPollingIntervalMultiplier	none	可选	2

端点覆盖参数

Athena 端点覆盖

驱动程序用于对 Athena 进行 API 调用的端点。

请注意以下几点：

- 如果未在提供的 URL 中指定 `https://` 或 `http://` 协议，则驱动程序会插入 `https://` 前缀。
- 如果未指定此参数，驱动程序会使用默认的端点。

参数名称	别名	参数类型	默认值
AthenaEndpoint	EndpointOverride（已弃用）	可选	none

Athena 流式传输服务端点覆盖

驱动程序在使用 Athena 流式传输服务时用于下载查询结果的端点。Athena 流式传输服务在端口 444 上可用。

请注意以下几点：

- 如果未在提供的 URL 中指定 `https://` 或 `http://` 协议，则驱动程序会插入 `https://` 前缀。
- 如果未在提供的 URL 中指定端口，则驱动程序会插入流式传输服务端口 444。
- 如果未指定 `AthenaStreamingEndpoint` 参数，则驱动程序将使用 `AthenaEndpoint` 覆盖。如果既未指定 `AthenaStreamingEndpoint` 也未指定 `AthenaEndpoint` 覆盖，则驱动程序将使用默认的流式传输端点。

参数名称	别名	参数类型	默认值
AthenaStreamingEndpoint	StreamingEndpointOverride（已弃用）	可选	none

Lake Formation 端点覆盖

当使用 AWS Lake Formation AssumeDecoratedRoleWithSAML API 检索临时凭证时，驱动程序将为 Lake Formation 服务使用的端点。如果未指定此参数，驱动程序会使用默认的 Lake Formation 端点。

请注意以下几点：

- 如果未在提供的 URL 中指定 `https://` 或 `http://` 协议，则驱动程序会插入 `https://` 前缀。

参数名称	别名	参数类型	默认值
LakeFormationEndpoint	LfEndpointOverride（已弃用）	可选	none

S3 端点覆盖

驱动程序在使用 Amazon S3 提取器时用于下载查询结果的端点。如果未指定此参数，驱动程序会使用默认的 Amazon S3 端点。

请注意以下几点：

- 如果未在提供的 URL 中指定 `https://` 或 `http://` 协议，则驱动程序会插入 `https://` 前缀。

参数名称	别名	参数类型	默认值
S3Endpoint	无	可选	none

STS 端点覆盖

当使用 AWS STS AssumeRoleWithSAML API 检索临时凭证时，驱动程序将为 AWS STS 服务使用的端点。如果未指定此参数，驱动程序会使用默认的 AWS STS 端点。

请注意以下几点：

- 如果未在提供的 URL 中指定 `https://` 或 `http://` 协议，则驱动程序会插入 `https://` 前缀。

参数名称	别名	参数类型	默认值
StsEndpoint	StsEndpointOverrid e（已弃用）	可选	none

代理配置参数

代理主机

代理主机的 URL。如果需要 Athena 请求来通过代理，请使用此参数。

> **Note**
>
> 请务必在 `ProxyHost` 的 URL 的开头处包含 `https://` 或 `http://` 协议。

参数名称	别名	参数类型	默认值
ProxyHost	none	可选	none

代理端口

要在代理主机上使用的端口。如果需要 Athena 请求来通过代理，请使用此参数。

参数名称	别名	参数类型	默认值
ProxyPort	none	可选	none

代理用户名

用于对代理服务器进行身份验证的用户名。如果需要 Athena 请求来通过代理，请使用此参数。

参数名称	别名	参数类型	默认值
ProxyUsername	ProxyUID（已弃用）	可选	none

代理密码

用于对代理服务器进行身份验证的密码。如果需要 Athena 请求来通过代理，请使用此参数。

参数名称	别名	参数类型	默认值
ProxyPassword	ProxyPWD（已弃用）	可选	none

代理豁免主机

启用代理后（即设置了 `ProxyHost` 和 `ProxyPort` 连接参数），驱动程序在不使用代理的情况下连接到的一组主机名。主机应用管道 (`|`) 字符分隔（例如 `host1.com|host2.com`）。

参数名称	别名	参数类型	默认值
ProxyExemptHosts	NonProxyHosts	可选	none

为身份提供者启用代理

指定当驱动程序连接到身份提供者时是否应使用代理。

参数名称	别名	参数类型	默认值
ProxyEnabledForIdP	UseProxyForIdP	可选	FALSE

日志记录参数

本节旨在介绍与日志记录相关的参数。

日志级别

指定驱动程序日志记录的级别。除非还设置了 LogPath 参数，否则不会记录任何内容。

> **Note**
>
> 除非有特殊要求，否则建议仅设置 LogPath 参数。仅设置 LogPath 参数即可启用日志记录并使用默认的 TRACE 日志级别。TRACE 日志级别提供了最详细的日志记录。

参数名称	别名	参数类型	默认值	可能的值
LogLevel	none	可选	TRACE	OFF、ERROR、WARN、INFO、DEBUG、TRACE

日志路径

运行驱动程序的计算机上存储驱动程序日志的目录路径。将在指定目录中创建的具有唯一名称的日志文件。如果已设置，则启用驱动程序日志记录。

参数名称	别名	参数类型	默认值
LogPath	none	可选	none

应用程序名称

使用该驱动程序的应用程序的名称。如果为此参数指定了值，则该值将包含在驱动程序对 Athena 进行的 API 调用的用户代理字符串中。

> **Note**
>
> 您也可以通过调用 `DataSource` 对象上的 `setApplicationName` 来设置应用程序名称。

参数名称	别名	参数类型	默认值
ApplicationName	none	可选	none

连接测试

如果设置为 TRUE，则每次创建 JDBC 连接时，驱动程序都会执行连接测试，即使未对该连接执行查询也是如此。

参数名称	别名	参数类型	默认值
ConnectionTest	none	可选	TRUE

> **Note**
>
> 连接测试会向 Athena 提交 `SELECT 1` 查询来验证连接配置是否正确。这意味着两个文件将存储在 Amazon S3 中（结果集和元数据），并且可能会根据Amazon Athena 定价策略收取额外费用。

重试次数

驱动程序应向 Athena 重新发送可检索请求的最大次数。

参数名称	别名	参数类型	默认值
NumRetries	MaxErrorRetry（已弃用）	可选	none

身份验证连接参数

Athena JDBC 3.x 驱动程序支持多种身份验证方法。所需的连接参数取决于您所使用的身份验证方法。

主题

- IAM 凭证
- 默认凭证
- AWS 配置文件凭证
- 实例配置文件凭证
- 自定义凭证
- JWT 凭证
- Azure AD 凭证
- Okta 凭证
- Ping 凭证
- AD FS 凭证
- 浏览器 Azure AD 凭证
- 浏览器 SAML 凭证

IAM 凭证

您可以通过设置以下连接参数，使用自己的 IAM 凭证与 JDBC 驱动程序连接到 Amazon Athena。

用户

您的 AWS 访问密钥 ID。有关访问密钥的信息，请参阅《IAM 用户指南》中的 AWS 安全凭证。

参数名称	别名	参数类型	默认值
用户	AccessKeyId	必需	none

密码

您的 AWS 密钥 ID。有关访问密钥的信息，请参阅《IAM 用户指南》中的 AWS 安全凭证。

参数名称	别名	参数类型	默认值
密码	SecretAccessKey	可选	none

会话令牌

如果您使用的是临时 AWS 凭证，则必须指定会话令牌。有关临时凭证的更多信息，请参阅《IAM 用户指南》中的 IAM 临时安全凭证。

参数名称	别名	参数类型	默认值
SessionToken	none	可选	none

默认凭证

您可以通过设置以下连接参数，使用在客户端系统上配置的默认凭证来连接 Amazon Athena。有关使用默认凭证的信息，请参阅《AWS SDK for Java 开发人员指南》中的使用默认凭证提供程序链。

凭证提供程序

将用于对 AWS 的请求进行身份验证的凭证提供程序。将此参数的值设置为 `DefaultChain`。

参数名称	别名	参数类型	默认值	要使用的值
CredentialsProvider	AWSCredentialsProviderClass（已弃用）	必需	none	`DefaultChain`

AWS 配置文件凭证

您可以通过设置以下连接参数来使用存储在 AWS 配置文件中的凭证。AWS 配置文件通常存储在 `~/.aws` 目录的文件中。有关 AWS 配置文件的信息，请参阅《AWS SDK for Java Developer Guide》中的 Use profiles。

凭证提供程序

将用于对 AWS 的请求进行身份验证的凭证提供程序。将此参数的值设置为 `ProfileCredentials`。

参数名称	别名	参数类型	默认值	要使用的值
CredentialsProvider	AWSCredentialsProviderClass（已弃用）	必需	none	`ProfileCredentials`

配置文件名称

AWS 配置文件的名称，其凭证应该用于对 Athena 的请求进行身份验证。

参数名称	别名	参数类型	默认值
ProfileName	none	必需	none

> **Note**
>
> 您也可以将配置文件名称指定为 `CredentialsProviderArguments` 参数的值，虽然这种用法已被弃用。

实例配置文件凭证

此身份验证类型用于 Amazon EC2 实例。实例配置文件是附加到 Amazon EC2 实例的配置文件。使用实例配置文件凭证提供程序，将 AWS 凭证的管理委托给 Amazon EC2 实例元数据服务。如此，开发人员就无需在 Amazon EC2 实例上永久存储凭证，也不必担心轮换或管理临时凭证了。

凭证提供程序

将用于对 AWS 的请求进行身份验证的凭证提供程序。将此参数的值设置为 `InstanceProfile`。

参数名称	别名	参数类型	默认值	要使用的值
CredentialsProvider	AWSCredentialsProv	必需	none	`InstanceProfile`

参数名称	别名	参数类型	默认值	要使用的值
	iderClass（已弃用）			

自定义凭证

您可以使用这种身份验证类型，通过使用实现 AwsCredentialsProvider 接口的 Java 类来提供自己的凭证。

凭证提供程序

将用于对 AWS 的请求进行身份验证的凭证提供程序。将此参数的值设置为实现 AwsCredentialsProvider 接口的自定义类的完全限定类名。处于运行时系统中时，这个类必须位于使用 JDBC 驱动程序的应用程序的 Java 类路径上。

参数名称	别名	参数类型	默认值	要使用的值
CredentialsProvider	AWSCredentialsProviderClass（已弃用）	必需	none	`AwsCredentialsProvider` 的自定义实现的完全限定类名

凭证提供程序参数

自定义凭证提供程序构造函数的字符串参数的逗号分隔列表。

参数名称	别名	参数类型	默认值
CredentialsProviderArguments	AwsCredentialsProviderArguments（已弃用）	可选	none

JWT 凭证

凭借此身份验证类型，您可以使用从外部身份提供者处获得的 JSON Web 令牌（JWT）作为连接参数，对 Athena 进行身份验证。外部凭证提供程序必须已经与 AWS 联合。

凭证提供程序

将用于对 AWS 的请求进行身份验证的凭证提供程序。将此参数的值设置为 `JWT`。

参数名称	别名	参数类型	默认值	要使用的值
CredentialsProvider	AWSCredentialsProviderClass（已弃用）	必需	none	`JWT`

JWT Web 身份令牌

从外部联合身份提供者处获得的 JWT 令牌。此令牌将用于向 Athena 进行身份验证。

参数名称	别名	参数类型	默认值
JwtWebIdentityToken	web_identity_token（已弃用）	必需	none

JWT 角色 ARN

担任角色的 Amazon Resource Name（ARN）。有关代入角色的信息，请参阅《AWS Security Token Service API Reference》中的 AssumeRole。

参数名称	别名	参数类型	默认值
JwtRoleArn	role_arn（已弃用）	必需	none

JWT 角色会话名称

使用 JWT 凭证进行身份验证时的会话名称。此名称可以是您选择的任何名称。

参数名称	别名	参数类型	默认值
JwtRoleSessionName	role_session_name（已弃用）	必需	none

Azure AD 凭证

这是一种基于 SAML 的身份验证机制，允许使用 Azure AD 身份提供者对 Athena 进行身份验证。此方法假设 Athena 和 Azure AD 之间已经建立了联合身份验证。

> **Note**
>
> 本节中的某些参数名称具有别名。这些别名在功能上等同于参数名称，提供这些别名的目的是与 JDBC 2.x 驱动程序向后兼容。由于参数名称经过改进以遵循更清晰、更一致的命名约定，因此建议使用这些名称而不是已经弃用的别名。

凭证提供程序

将用于对 AWS 的请求进行身份验证的凭证提供程序。将此参数的值设置为 `AzureAD`。

参数名称	别名	参数类型	默认值	要使用的值
CredentialsProvider	AWSCredentialsProviderClass（已弃用）	必需	none	`AzureAD`

用户

将用于对 Azure AD 进行身份验证的 Azure AD 用户的电子邮件地址。

参数名称	别名	参数类型	默认值
用户	UID（已弃用）	必需	none

密码

Azure AD 用户的密码。

参数名称	别名	参数类型	默认值
密码	PWD（已弃用）	必需	none

Azure AD 租户 ID

Azure AD 应用程序的租户 ID。

参数名称	别名	参数类型	默认值
AzureAdTenantId	tenant_id（已弃用）	必需	none

Azure AD 客户端 ID

Azure AD 应用程序的客户端 ID。

参数名称	别名	参数类型	默认值
AzureAdClientId	client_id（已弃用）	必需	none

Azure AD 客户端密钥

Azure AD 应用程序的客户端密钥。

参数名称	别名	参数类型	默认值
AzureAdClientSecret	client_secret（已弃用）	必需	none

Preferred role

担任角色的 Amazon Resource Name（ARN）。有关 ARN 角色的信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

参数名称	别名	参数类型	默认值
PreferredRole	preferred_role（已弃用）	可选	none

角色会话持续时间

角色会话的持续时间（以秒为单位）。有关更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

参数名称	别名	参数类型	默认值
RoleSessionDuration	Duration（已弃用）	可选	3600

Lake Formation 已启用

指定是否使用 AssumeDecoratedRoleWithSAML Lake Formation API 操作（而非 AssumeRoleWithSAML AWS STS API 操作）来检索临时 IAM 凭证。

参数名称	别名	参数类型	默认值
LakeFormationEnabled	none	可选	FALSE

Okta 凭证

这是一种基于 SAML 的身份验证机制，允许使用 Okta 身份提供者对 Athena 进行身份验证。此方法假设 Athena 和 Okta 之间已经建立了联合身份验证。

凭证提供程序

将用于对 AWS 的请求进行身份验证的凭证提供程序。将此参数的值设置为 `Okta`。

参数名称	别名	参数类型	默认值	要使用的值
CredentialsProvider	AWSCredentialsProviderClass（已弃用）	必需	none	`Okta`

用户

将用于对 Okta 进行身份验证的 Okta 用户的电子邮件地址。

参数名称	别名	参数类型	默认值
用户	UID（已弃用）	必需	none

密码

Okta 用户的密码。

参数名称	别名	参数类型	默认值
密码	PWD（已弃用）	必需	none

Okta 主机名称

您的 Okta 组织的 URL。您可以从 Okta 应用程序中的嵌入链接 URL 中提取 `idp_host` 参数。要查看步骤，请参阅 从 Okta 检索 ODBC 配置信息。从 `https://` 之后的第一个分段一直到 `okta.com`（包括在内）都是您的 IdP 主机（例如：以 `https://trial-1234567.okta.com` 开头的 URL 的 `trial-1234567.okta.com`）。

参数名称	别名	参数类型	默认值
OktaHostName	IdP_Host（已弃用）	必需	none

Okta 应用程序 ID

适用于您的应用程序的两部分标识符。您可以从 Okta 应用程序的嵌入链接 URL 中提取应用程序 ID。要查看步骤，请参阅 从 Okta 检索 ODBC 配置信息。应用程序 ID 是 URL 的最后两个分段，包括中间的正斜杠。这两个分段是两个长度为 20 个字符的字符串，其中包含数字和大小写字母（例如 `Abc1de2fghi3J45kL678/abc1defghij2klmNo3p4`）。

参数名称	别名	参数类型	默认值
OktaAppId	App_ID（已弃用）	必需	none

Okta 应用程序名称

Okta 应用程序的名称。

参数名称	别名	参数类型	默认值
OktaAppName	App_Name（已弃用）	必需	none

Okta MFA 类型

如果您已将 Okta 设置为要求多重身份验证（MFA），则需要根据要使用的第二个因素指定 Okta MFA 类型和其他参数。

Okta MFA 类型是用于向 Okta 进行身份验证的第二种身份验证因素类型（仅次于密码）。受支持的第二个因素包括通过 Okta Verify 应用程序传送的推送通知，以及由 Okta Verify、Google Authenticator 生成或通过 SMS 发送的临时一次性密码（TOTP）。各个组织的安全策略决定用户登录是否需要 MFA。

参数名称	别名	参数类型	默认值	可能的值
OktaMfaType	okta_mfa_type（已弃用）	如果将 Okta 设置为需要 MFA，则为必填项	none	`oktaverifywithpush`，`oktaverifywithtotp`，`googleauthenticator`，`smsauthentication`

Okta 电话号码

选择 `smsauthentication` MFA 类型时，Okta 将使用 SMS 向其发送临时一次性密码的电话号码。该电话号码必须是美国或加拿大电话号码。

参数名称	别名	参数类型	默认值
OktaPhoneNumber	okta_phone_number（已弃用）	如果 OktaMfaType 是 `smsauthentication` ，则为必填项	none

Okta MFA 等待时间

在驱动程序引发超时异常之前，等待用户确认来自 Okta 的推送通知的时长（以秒为单位）。

参数名称	别名	参数类型	默认值
OktaMfaWaitTime	okta_mfa_wait_time（已弃用）	可选	60

Preferred role

担任角色的 Amazon Resource Name（ARN）。有关 ARN 角色的信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

参数名称	别名	参数类型	默认值
PreferredRole	preferred_role（已弃用）	可选	none

角色会话持续时间

角色会话的持续时间（以秒为单位）。有关更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

参数名称	别名	参数类型	默认值
RoleSessionDuration	Duration（已弃用）	可选	3600

Lake Formation 已启用

指定是否使用 AssumeDecoratedRoleWithSAML Lake Formation API 操作（而非 AssumeRoleWithSAML AWS STS API 操作）来检索临时 IAM 凭证。

参数名称	别名	参数类型	默认值
LakeFormationEnabled	none	可选	FALSE

Ping 凭证

这是一种基于 SAML 的身份验证机制，允许使用 Ping Federate 身份提供者对 Athena 进行身份验证。此方法假设 Athena 和 Ping Federate 之间已经建立了联合身份验证。

凭证提供程序

将用于对 AWS 的请求进行身份验证的凭证提供程序。将此参数的值设置为 `Ping`。

参数名称	别名	参数类型	默认值	要使用的值
CredentialsProvider	AWSCredentialsProviderClass（已弃用）	必需	none	`Ping`

用户

Ping Federate 用户用于通过 Ping Federate 进行身份验证的电子邮件地址。

参数名称	别名	参数类型	默认值
用户	UID（已弃用）	必需	none

密码

Ping Federate 用户的密码。

参数名称	别名	参数类型	默认值
密码	PWD（已弃用）	必需	none

PingHostName

您的 Ping 服务器的地址。要查找您的地址，请访问以下 URL 并查看 SSO 应用程序端点字段。

```
https://your-pf-host-#:9999/pingfederate/your-pf-app#/spConnections
```

参数名称	别名	参数类型	默认值
PingHostName	IdP_Host（已弃用）	必需	none

PingPortNumber

用于连接 IdP 主机的端口号。

参数名称	别名	参数类型	默认值
PingPortNumber	IdP_Port（已弃用）	必需	none

PingPartnerSpId

服务提供商地址。要查找服务提供商地址，请访问以下 URL 并查看 SSO 应用程序端点字段。

```
https://your-pf-host-#:9999/pingfederate/your-pf-app#/spConnections
```

参数名称	别名	参数类型	默认值
PingPartnerSpId	Partner_SPID（已弃用）	必需	none

Preferred role

担任角色的 Amazon Resource Name（ARN）。有关 ARN 角色的信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

参数名称	别名	参数类型	默认值
PreferredRole	preferred_role（已弃用）	可选	none

角色会话持续时间

角色会话的持续时间（以秒为单位）。有关更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

参数名称	别名	参数类型	默认值
RoleSessionDuration	Duration（已弃用）	可选	3600

Lake Formation 已启用

指定是否使用 AssumeDecoratedRoleWithSAML Lake Formation API 操作（而非 AssumeRoleWithSAML AWS STS API 操作）来检索临时 IAM 凭证。

参数名称	别名	参数类型	默认值
LakeFormationEnabled	none	可选	FALSE

AD FS 凭证

这是一种基于 SAML 的身份验证机制，允许使用 Microsoft Active Directory 联合身份验证服务（AD FS）对 Athena 进行身份验证。此方法假设用户已在 Athena 和 AD FS 之间建立了联合身份验证。

凭证提供程序

将用于对 AWS 的请求进行身份验证的凭证提供程序。将此参数的值设置为 ADFS。

参数名称	别名	参数类型	默认值	要使用的值
CredentialsProvider	AWSCredentialsProviderClass（已弃用）	必需	none	ADFS

用户

AD FS 用户的电子邮件地址，用于对 AD FS 进行身份验证。

参数名称	别名	参数类型	默认值
用户	UID（已弃用）	基于表单的身份验证的必要项。Windows 集成式身份验证的可选项。	none

密码

AD FS 用户的密码。

参数名称	别名	参数类型	默认值
密码	PWD（已弃用）	基于表单的身份验证的必要项。Windows 集成式身份验证的可选项。	none

ADFS 主机名

AD FS 服务器的地址。

参数名称	别名	参数类型	默认值
AdfsHostName	IdP_Host（已弃用）	必需	none

ADFS 端口号

用于连接 AD FS 服务器的端口号。

参数名称	别名	参数类型	默认值
AdfsPortNumber	IdP_Port（已弃用）	必需	none

ADFS 依赖方

受信任的依赖方。使用此参数覆盖 AD FS 依赖方端点 URL。

参数名称	别名	参数类型	默认值
AdfsRelyingParty	LoginToRP（已弃用）	可选	`urn:amazon:webservices`

已启用 ADFS WIA

布尔值。使用此参数可通过 AD FS 启用 Windows 集成式身份验证（WIA）。

参数名称	别名	参数类型	默认值
AdfsWiaEnabled	none	可选	`FALSE`

Preferred role

担任角色的 Amazon Resource Name（ARN）。有关 ARN 角色的信息，请参阅《AWS Security Token Service API Reference》中的 `AssumeRole`。

参数名称	别名	参数类型	默认值
PreferredRole	preferred_role（已弃用）	可选	none

角色会话持续时间

角色会话的持续时间（以秒为单位）。有关更多信息，请参阅《AWS Security Token Service API 参考》中的 AssumeRole。

参数名称	别名	参数类型	默认值
RoleSessionDuration	Duration（已弃用）	可选	3600

Lake Formation 已启用

指定是否使用 AssumeDecoratedRoleWithSAML Lake Formation API 操作（而非 AssumeRoleWithSAML AWS STS 操作）来检索临时 IAM 凭证。

参数名称	别名	参数类型	默认值
LakeFormationEnabled	none	可选	FALSE

浏览器 Azure AD 凭证

浏览器 Azure AD 是一个基于 SAML 的身份验证机制，可与 Azure AD 身份提供者配合使用并支持多重身份验证。与标准 Azure AD 身份验证机制不同，该机制不需要在连接参数中输入用户名、密码或客户端密钥。与标准 Azure AD 身份验证机制一样，浏览器 Azure AD 还假设用户已经在 Athena 和 Azure AD 之间建立了联合身份验证。

凭证提供程序

将用于对 AWS 的请求进行身份验证的凭证提供程序。将此参数的值设置为 BrowserAzureAD。

参数名称	别名	参数类型	默认值	要使用的值
CredentialsProvider	AWSCredentialsProviderClass（已弃用）	必需	none	BrowserAzureAD

Azure AD 租户 ID

Azure AD 应用程序的租户 ID

参数名称	别名	参数类型	默认值
AzureAdTenantId	tenant_id（已弃用）	必需	none

Azure AD 客户端 ID

Azure AD 应用程序的客户端 ID

参数名称	别名	参数类型	默认值
AzureAdClientId	client_id（已弃用）	必需	none

身份提供者响应超时

驱动程序停止等待 Azure AD 发出 SAML 响应之前的时长（以秒为单位）。

参数名称	别名	参数类型	默认值
IdpResponseTimeout	idp_response_timeout（已弃用）	可选	120

Preferred role

担任角色的 Amazon Resource Name（ARN）。有关 ARN 角色的信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

参数名称	别名	参数类型	默认值
PreferredRole	preferred_role（已弃用）	可选	none

角色会话持续时间

角色会话的持续时间（以秒为单位）。有关更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

参数名称	别名	参数类型	默认值
RoleSessionDuration	Duration（已弃用）	可选	3600

Lake Formation 已启用

指定是否使用 AssumeDecoratedRoleWithSAML Lake Formation API 操作（而非 AssumeRoleWithSAML AWS STS API 操作）来检索临时 IAM 凭证。

参数名称	别名	参数类型	默认值
LakeFormationEnabled	none	可选	FALSE

浏览器 SAML 凭证

浏览器 SAML 是一个通用身份验证插件，可与基于 SAML 的身份提供者配合使用并支持多重身份验证。

凭证提供程序

将用于对 AWS 的请求进行身份验证的凭证提供程序。将此参数的值设置为 `BrowserSaml`。

参数名称	别名	参数类型	默认值	要使用的值
CredentialsProvider	AWSCredentialsProviderClass（已弃用）	必需	none	`BrowserSaml`

单点登录的登录 URL

应用程序在基于 SAML 的身份提供者上的单点登录 URL。

参数名称	别名	参数类型	默认值
SsoLoginUrl	login_url（已弃用）	必需	none

侦听端口

用于侦听 SAML 响应的端口号。此值应与配置基于 SAML 的身份提供者时所用的 URL 相匹配（例如：`http://localhost:7890/athena`）。

参数名称	别名	参数类型	默认值
ListenPort	listen_port（已弃用）	可选	7890

身份提供者响应超时

驱动程序停止等待 Azure AD 发出 SAML 响应之前的时长（以秒为单位）。

参数名称	别名	参数类型	默认值
IdpResponseTimeout	idp_response_timeout（已弃用）	可选	120

Preferred role

担任角色的 Amazon Resource Name（ARN）。有关 ARN 角色的信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

参数名称	别名	参数类型	默认值
PreferredRole	preferred_role（已弃用）	可选	none

角色会话持续时间

角色会话的持续时间（以秒为单位）。有关更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

参数名称	别名	参数类型	默认值
RoleSessionDuration	Duration（已弃用）	可选	3600

Lake Formation 已启用

指定是否使用 AssumeDecoratedRoleWithSAML Lake Formation API 操作（而非 AssumeRoleWithSAML AWS STS API 操作）来检索临时 IAM 凭证。

参数名称	别名	参数类型	默认值
LakeFormationEnabled	none	可选	FALSE

其他 JDBC 3.x 配置

以下各节旨在介绍 JDBC 3.x 驱动程序的一些其他配置设置。

网络超时

驱动程序对 Athena 进行 API 调用时等待响应的时长（以毫秒为单位）。在此时间之后，驱动程序会引发超时异常。

不能将网络超时设置为连接参数。要进行设置，请在 JDBC `Connection` 对象上调用 `setNetworkTimeout` 方法。此值可在 JDBC 连接的生命周期中更改。此参数的默认值为 `infinity`。

以下示例将网络超时设置为 5000 毫秒。

```
...
AthenaDriver driver = new AthenaDriver();
Connection connection = driver.connect(url, connectionParameters);
connection.setNetworkTimeout(null, 5000);
...
```

查询超时

在提交查询后，驱动程序在 Athena 上等待查询完成的时长（以秒为单位）。在此时间之后，驱动程序会尝试取消已提交的查询并引发超时异常。

不能将查询超时设置为连接参数。要进行设置，请在 JDBC `Statement` 对象上调用 `setQueryTimeout` 方法。此值可在 JDBC 语句的生命周期中更改。此参数的默认值为 `0`（零）。值 `0` 表示查询可以一直运行直到完成（视 服务限额 而定）。

以下示例将查询超时设置为 5 秒。

```
...
AthenaDriver driver = new AthenaDriver();
Connection connection = driver.connect(url, connectionParameters);
Statement statement = connection.createStatement();
statement.setQueryTimeout(5);
...
```

Amazon Athena JDBC 3.x 发布说明

这些发行说明详细介绍了 Amazon Athena JDBC 3.x 驱动程序中的改进和修复。

3.2.0

发布时间：2024 年 4 月 26 日

改进

- 目录操作性能 – 不使用通配符的目录操作的性能已得到提高。
- 最小轮询间隔更改 – 已修改最小轮询间隔默认值，以减少驱动程序对 Athena 进行的 API 调用次数。仍然会尽快检测到查询完成。
- BI 工具可发现性 – 商业智能工具更容易发现驱动程序。
- 数据类型映射 – Athena `binary`、`array` 和 `struct` DDL 数据类型的数据类型映射已得到改进。
- AWS SDK 版本 – 驱动程序中使用的 AWS SDK 版本已更新至 2.25.34。

修复

- 联合目录表列表 – 修复了导致联合目录返回空表列表的问题。
- getSchemas – 修复了导致 JDBC DatabaseMetaData#getSchemas 方法仅从默认目录而不是从所有目录中获取数据库的问题。
- getColumns – 修复了使用空目录名称调用 JDBC DatabaseMetaData#getColumns 方法时导致返回空目录的问题。

3.1.0

发布时间：2024 年 2 月 15 日

改进

- 添加了对 Microsoft Active Directory 联合身份验证服务（AD FS）Windows 集成身份验证和基于表单的身份验证的支持。
- 为了与 2.x 版本向后兼容，现在接受 awsathena JDBC 子协议，但会产生弃用警告。请改用 athena JDBC 子协议。
- AwsDataCatalog 现在是目录参数的默认值，default 是数据库参数的默认值。这些更改确保返回当前目录和数据库的正确值，而不是 null。
- 为符合 JDBC 规范，IS_AUTOINCREMENT 和 IS_GENERATEDCOLUMN 现在返回的是空字符串，而不是 NO。
- 现在，Athena int 数据类型映射到与 Athena integer 相同的 JDBC 类型，而不是映射到 other。
- 现在，当 Athena 中的列元数据不包含可选 precision 和 scale 字段时，驱动程序会为 ResultSet 列中的相应值返回零。
- AWS SDK 版本已更新为 2.21.39。

修复

- 修复了当来自 Athena 的纯文本结果的列数与 Athena 结果元数据中的列数不一致时导致出现异常的 GetQueryResultsStream 问题。

3.0.0

发布时间：2023 年 11 月 16 日

Athena JDBC 3.x 驱动程序是新一代的驱动程序，具有更好的性能和兼容性。JDBC 3.x 驱动程序支持直接从 Amazon S3 读取查询结果，此举可提高使用大型查询结果的应用程序的性能。该新驱动程序还减少了第三方依赖项，让与商业智能工具和自定义应用程序的集成变得更加容易。

Athena JDBC 3.x 驱动程序的早期版本

强烈建议您使用最新版本的 JDBC 3.x 驱动程序。最新版本的驱动程序包含最新改进和修复。只有当应用程序与最新版本不兼容时，才使用旧版本。

JDBC 驱动程序 uber jar

以下下载内容将驱动程序及其所有依赖项打包到同一个 `.jar` 文件中。此下载通常用于第三方 SQL 客户端。

- 3.1.0 uber jar
- 3.0.0 uber jar

JDBC 驱动程序 lean jar

以下下载内容是一个 `.zip` 文件，其中包含驱动程序的 lean `.jar` 文件和驱动程序依赖项的单独 `.jar` 文件。此下载通常用于依赖项可能与驱动程序所使用的依赖项冲突的自定义应用程序。如果您想选择要在 lean jar 中包含哪些驱动程序依赖项，以及在自定义应用程序已经包含一个或多个驱动程序依赖项时应排除哪些驱动程序依赖项，则此下载非常有用。

- 3.1.0 lean jar
- 3.0.0 lean jar

Athena JDBC 2.x 驱动程序

您可以使用 JDBC 连接将 Athena 连接到商业智能工具和其他应用程序，例如 SQL Workbench。要执行此操作，请使用此页面上的 Amazon S3 链接下载、安装和配置 Athena JDBC 2.x 驱动程序。有关构建 JDBC 连接 URL 的信息，请参阅可下载的JDBC 驱动程序安装和配置指南。有关许可的更多信息，请参阅 通过 JDBC 和 ODBC 连接访问。要提交有关 JDBC 驱动程序的反馈，请发送电子邮件至 athena-feedback@amazon.com。从版本 2.0.24 开始，有两个版本的驱动程序可用：一个包括 AWS SDK，一个不包括。

Important

使用 JDBC 驱动程序时，请务必注意以下要求：

- 保留端口 444 — 保留 Athena 用于流式传输查询结果的端口 444，对出站流量开放。当您使用 PrivateLink 终端节点连接到 Athena 时，请确保附加到 PrivateLink 终端节点的安全组在端口 444 上对入站流量开放。如果端口 444 被阻止，您可能会收到错误消息 [Simba][AthenaJDBC](100123) An error has occurred. Exception during column initialization（[Simba][AthenaJDBC](100123) 出现错误。列初始化期间出现异常）。
- athena:GetQueryResultsStream 策略 — 将 `athena:GetQueryResultsStream` 策略操作添加到使用 JDBC 驱动程序的 IAM 主体。此策略操作并不通过 API 直接公开。它仅作为流

式传输结果的一部分与 ODBC 和 JDBC 驱动程序配合使用。有关策略示例，请参阅AWS 托管策略：AWSQuicksightAthenaAccess。

- 将 JDBC 驱动程序用于多个数据目录 – 若要将 JDBC 驱动程序用于 Athena 的多个数据目录（例如，当使用外部 Hive 元存储或者联合查询时），请将 `MetadataRetrievalMethod=ProxyAPI` 包含在 JDBC 连接字符串中。
- 4.1 驱动程序 – 从 2023 年开始，将停止对 JDBC 版本 4.1 的驱动程序支持。不会发布进一步的更新。如果您使用的是 JDBC 4.1 驱动程序，我们强烈建议您迁移到 4.2 驱动程序。

带有 AWS SDK 的 JDBC 2.x 驱动程序

JDBC 驱动程序版本 2.1.5 符合 JDBC API 4.2 数据标准，且要求 JDK 8.0 或更高版本。有关检查您使用的 Java 运行时环境 (JRE) 版本的信息，请参阅 Java 文档。

使用以下链接下载 JDBC 4.2 驱动程序 `.jar` 文件。

- AthenaJDBC42-2.1.5.1000.jar

以下 `.zip` 文件下载包含适用于 JDBC 4.2 的 `.jar` 文件以及 AWS 开发工具包和随附文档、发布说明、许可和协议。

- SimbaAthenaJDBC-2.1.5.1000.zip

不带 AWS SDK 的 JDBC 2.x 驱动程序

JDBC 驱动程序版本 2.1.5 符合 JDBC API 4.2 数据标准，且要求 JDK 8.0 或更高版本。有关检查您使用的 Java 运行时环境 (JRE) 版本的信息，请参阅 Java 文档。

使用以下链接下载不包含 AWS 开发工具包的 JDBC 4.2 驱动程序 `.jar` 文件。

- AthenaJDBC42-2.1.5.1001.jar

以下 `.zip` 文件下载包含适用于 JDBC 4.2 的 `.jar` 文件以及随附文档、发布说明、许可和协议。不包括 AWS SDK。

- SimbaAthenaJDBC-2.1.5.1001.zip

JDBC 2.x 驱动程序发布说明、许可协议和版权声明

下载所需要的版本后，请阅读发布说明，查看许可协议和版权声明。

- 发行说明
- 许可协议
- 通知
- 第三方许可证

JDBC 2.x 驱动程序文档

下载驱动程序的以下文档：

- JDBC 驱动程序安装和配置指南。使用该指南安装和配置此驱动程序。
- JDBC 驱动程序迁移指南。使用该指南从早期版本迁移到最新版本。

通过 ODBC 连接到 Amazon Athena

Amazon Athena 提供两个 ODBC 驱动程序版本，即 1.x 和 2.x。Athena ODBC 2.x 驱动程序是全新替代方案，支持 Linux、macOS ARM、macOS Intel 和 Windows 64 位系统。Athena 2.x 驱动程序支持 1.x ODBC 驱动程序支持的所有身份验证插件，而且几乎所有连接参数都向后兼容。

- 要下载 ODBC 2.x 驱动程序，请参阅 Amazon Athena ODBC 2.x。
- 要下载 ODBC 1.x 驱动程序，请参阅 Athena ODBC 1.x 驱动程序。

主题

- Amazon Athena ODBC 2.x
- Athena ODBC 1.x 驱动程序
- 使用 Amazon Athena Power BI 连接器

Amazon Athena ODBC 2.x

您可以使用 ODBC 连接从许多第三方 SQL 客户端工具和应用程序连接到 Amazon Athena。可在客户端计算机上设置 ODBC 连接。

注意事项和限制

- 有关从 Athena ODBC 1.x 驱动程序迁移到 Athena 2.x ODBC 驱动程序的信息，请参阅 迁移到 ODBC 2.x 驱动程序。
- 当使用具有 CSE_KMS 加密选项的 S3 提取器时，Amazon S3 客户端无法解密在 Amazon S3 存储桶中存储的结果。要解决该问题，使用 Athena 流式处理 API 选项获取结果集。

ODBC 2.x 驱动程序下载

要下载 Amazon Athena 2.x ODBC 驱动程序，请访问此页面上的链接。

> **Important**
>
> 使用 ODBC 2.x 驱动程序时，请务必注意以下要求：
>
> - 保留端口 444 — 保留 Athena 用于流式传输查询结果的端口 444，对出站流量开放。当您使用 PrivateLink 终端节点连接到 Athena 时，请确保附加到 PrivateLink 终端节点的安全组在端口 444 上对入站流量开放。
> - athena：GetQueryResultsStream 策略 — 将 `athena:GetQueryResultsStream` 策略操作添加到使用 ODBC 驱动程序的 IAM 主体。此策略操作并不通过 API 直接公开。它仅作为流式传输结果的一部分与 ODBC 和 JDBC 驱动程序配合使用。有关策略示例，请参阅AWS 托管策略：AWSQuicksightAthenaAccess。

Linux

驱动程序版本	下载链接
适用于 Linux 64 位的 ODBC 2.0.3.0	Linux 64 位 ODBC 驱动程序 2.0.3.0

macOS（ARM）

驱动程序版本	下载链接
适用于 macOS 64 位的 ODBC 2.0.3.0（ARM）	macOS 64 位 ODBC 驱动程序 2.0.3.0（ARM）

macOS（英特尔）

驱动程序版本	下载链接
适用于 macOS 64 位的 ODBC 2.0.3.0（英特尔）	macOS 64 位 ODBC 驱动程序 2.0.3.0（英特尔）

Windows

驱动程序版本	下载链接
适用于 Windows 64 位的 ODBC 2.0.3.0	Windows 64 位 ODBC 驱动程序 2.0.3.0

主题

- ODBC 2.x 驱动程序入门
- Athena ODBC 2.x 连接参数
- 迁移到 ODBC 2.x 驱动程序
- 对 ODBC 2.x 驱动程序进行故障排除
- Amazon Athena ODBC 2.x 发布说明

ODBC 2.x 驱动程序入门

使用本节中的信息开始使用 Amazon Athena ODBC 2.x 驱动程序。Windows、Linux 和 macOS 操作系统均支持该驱动程序。

主题

- Windows
- Linux
- macOS

Windows

若想使用 Windows 客户端计算机访问 Amazon Athena，则需要 Amazon Athena ODBC 驱动程序。

Windows 系统要求

在可以直接访问 Amazon Athena 数据库的客户端计算机上安装 Amazon Athena ODBC 驱动程序，而无需使用 Web 浏览器。

所用的 Windows 系统必须满足以下要求：

- 你具有管理员权限
- 以下操作系统之一：
  - Windows 11、10 或 8.1
  - Windows Server 2019、2016 或 2012
- 至少 100 MB 可用磁盘空间
- 已安装适用于 64 位 Windows 的 Microsoft Visual C++ Redistributable for Visual Studio。

安装 Amazon Athena ODBC 驱动程序

下载并安装适用于 Windows 的 Amazon Athena ODBC 驱动程序

1. 下载 `AmazonAthenaODBC-2.x.x.x.msi` 安装文件。
2. 启动安装文件，然后选择下一步。
3. 要接受许可协议条款，选中复选框，然后选择下一步。
4. 要更改安装位置，选择浏览，浏览到所需的文件夹，然后选择确定。
5. 要接受安装位置，选择下一步。
6. 选择安装。
7. 在安装完成时，选择完成。

设置驱动程序配置选项的方法

要在 Windows 中控制 Amazon Athena ODBC 驱动程序的行为，可通过以下方式设置驱动程序配置选项：

- 配置数据来源名称（DSN）时，在 ODBC 数据来源管理器程序中。
- 通过在以下位置添加或更改 Windows 注册表项：

```
HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI\YOUR_DSN_NAME
```

- 在以编程方式连接时，通过在连接字符串中设置驱动程序选项。

在 Windows 上配置数据来源名称

在下载并安装 ODBC 驱动程序后，必须将数据来源名称（DSN）条目添加到客户端计算机或 Amazon EC2 实例。SQL 客户端工具将使用此数据来源连接和查询 Amazon Athena。

创建系统 DSN 条目

1. 在 Windows 的开始菜单中，右键单击 ODBC 数据来源（64 位），然后选择更多、以管理员身份运行。
2. 在 ODBC 数据来源管理器中，选择驱动程序选项卡。
3. 在名称列中，确认是否存在 Amazon Athena ODBC（x64）。
4. 请执行以下操作之一：

   - 要为计算机上的所有用户配置驱动程序，选择系统 DSN 选项卡。由于使用不同账户加载数据的应用程序可能检测不到来自其他账户的用户 DSN，因而建议使用系统 DSN 配置选项。

     > **Note**
     >
     > 要使用系统 DSN 选项，需要管理权限。

   - 要仅为您的用户账户配置驱动程序，选择用户 DSN 选项卡。
5. 选择 添加。随即打开新建数据来源对话框。
6. 选择 Amazon Athena ODBC（x64），然后选择完成。
7. 在 Amazon Athena ODBC 配置对话框中，输入以下信息。有关这些选项的详细信息，请参阅 主要 ODBC 2.x 连接参数。

   - 在数据来源名称中，输入要用于识别数据来源的名称。
   - 对于描述，输入描述以帮助您识别数据来源。
   - 在区域中，输入要在其中使用 Athena 的 AWS 区域 名称（例如， **`us-west-1`**）。
   - 在目录中，输入 Amazon Athena 目录的名称。默认为 AwsDataCatalog，供 AWS Glue 使用。
   - 在数据库中，输入 Amazon Athena 数据库的名称。默认为默认。
   - 对于工作组，输入 Amazon Athena 工作组的名称。默认为主要。
   - 对于 S3 输出位置，输入 Amazon S3 中存储查询结果的位置（例如 **`s3://DOC-EXAMPLE-BUCKET/`**）。
   - （可选）对于加密选项，选择一个加密选项。默认为 `NOT_SET`。
   - （可选）对于 KMS 密钥，根据需要选择加密 KMS 密钥。

8. 要指定用于 IAM 身份验证的配置选项，选择身份验证选项。
9. 输入以下信息：
   - 对于身份验证类型，选择 IAM 凭证。这是默认模式。有关可用身份验证类型的更多信息，请参阅 身份验证选项。
   - 对于用户名，输入用户名。
   - 对于密码，输入密码。
   - 对于会话令牌，如果要使用临时 AWS 凭证，则输入会话令牌。有关临时凭证的更多信息，请参阅《IAM 用户指南》中的将临时凭证用于 AWS 资源。
10. 选择确定。
11. 在 Amazon Athena ODBC 配置对话框的底部，选择测试。如果客户端计算机成功连接到 Amazon Athena，则连接测试框会报告连接成功。否则，该框将报告连接失败并显示相应的错误信息。
12. 要关闭测试连接，选择确定。现在，您创建的数据来源在数据来源名称列表中显示。

在 Windows 上使用无 DSN 连接

您可以使用无 DSN 连接来连接到没有数据来源名称（DSN）的数据库。以下示例显示了连接到 Amazon Athena 的 Amazon Athena ODBC（x64）ODBC 驱动程序的连接字符串。

```
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

若想使用 Linux 客户端计算机访问 Amazon Athena，则需要安装 Amazon Athena ODBC 驱动程序。

Linux 系统要求

安装该驱动程序的每台 Linux 客户端计算机都必须满足以下要求。

- 拥有根访问权限。
- 使用以下 Linux 发行版之一：
  - Red Hat Enterprise Linux（RHEL）7 或 8
  - CentOS 7 或 8。

- 有 100MB 可用磁盘空间。
- 使用版本 2.3.1 或更高版本的 unixODBC。
- 使用版本 2.26 或更高版本的 GNU C 库（glibc）。

在 Linux 上安装 ODBC 数据连接器

参照以下过程在 Linux 操作系统上安装 Amazon Athena ODBC 驱动程序。

在 Linux 上安装 Amazon Athena ODBC 驱动程序

1. 输入下列命令之一：

   ```
   sudo rpm -Uvh AmazonAthenaODBC-2.X.Y.Z.rpm
   ```

   或者

   ```
   sudo yum --nogpgcheck localinstall AmazonAthenaODBC-2.X.Y.Z.rpm
   ```

2. 安装完成后，输入以下命令之一，验证驱动程序是否安装成功：

   - ```
     yum list | grep amazon-athena-odbc-driver
     ```

     输出：

     ```
     amazon-athena-odbc-driver.x86_64 2.0.2.1-1.amzn2int installed
     ```

   - ```
     rpm -qa | grep amazon
     ```

     输出：

     ```
     amazon-athena-odbc-driver-2.0.2.1-1.amzn2int.x86_64
     ```

在 Linux 上配置数据来源名称

安装好驱动程序后，可以在以下位置找到示例 `.odbc.ini` 和 `.odbcinst.ini` 文件：

- `/opt/athena/odbc/ini/`.

使用此位置中的 .ini 文件作为配置 Amazon Athena ODBC 驱动程序和数据来源名称（DSN）的示例。

> **Note**
>
> 默认情况下，ODBC 驱动程序管理器使用位于主目录中的隐藏配置文件 .odbc.ini 和 .odbcinst.ini。

要使用 unixODBC 指定 .odbc.ini 和 .odbcinst.ini 文件的路径，请执行以下步骤。

使用 unixODBC 指定 ODBC **.ini** 文件的位置

1. 将 ODBCINI 设置为 odbc.ini 文件的完整路径和文件名，如下例所示。

```
export ODBCINI=/opt/athena/odbc/ini/odbc.ini
```

2. 将 ODBCSYSINI 设置为 odbcinst.ini 文件所在目录的完整路径，如下例所示。

```
export ODBCSYSINI=/opt/athena/odbc/ini
```

3. 输入以下命令，验证是否使用 unixODBC 驱动程序管理器，并验证 odbc*.ini 文件是否正确：

```
username % odbcinst -j
```

示例输出

```
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. 如果想要使用数据来源名称（DSN）连接到数据存储，请配置 odbc.ini 文件来定义数据源名称（DSN）。设置 odbc.ini 文件中的属性，以便创建指定数据存储连接信息的 DSN，如下例所示。

```
[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. 配置 `odbcinst.ini` 文件，如下例所示。

```
[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. 安装并配置 Amazon Athena ODBC 驱动程序后，使用 unixODBC `isql` 命令行工具来验证连接，如下例所示。

```
username % isql -v "athena_odbc_test"
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
```

```
SQL>
```

macOS

若想使用 macOS 客户端计算机访问 Amazon Athena，则需要安装 Amazon Athena ODBC 驱动程序。

macOS 系统要求

安装该驱动程序的每台 macOS 计算机都必须满足以下要求。

- 使用 macOS 版本 14 或更高版本。
- 有 100MB 可用磁盘空间。
- 使用版本 3.52.16 或更高版本的 iODBC。

在 macOS 上安装 ODBC 数据连接器

参照以下过程下载并安装适用于 macOS 操作系统的 Amazon Athena ODBC 驱动程序。

下载并安装适用于 macOS 的 Amazon Athena ODBC 驱动程序

1. 下载 .pkg 包文件。
2. 双击 .pkg 文件。
3. 按照向导中的步骤安装驱动程序。
4. 在许可协议页面上，按继续，然后选择同意。
5. 选择安装。
6. 在安装完成时，选择完成。
7. 输入以下命令，验证是否已安装驱动程序：

```
> pkgutil --pkgs | grep athenaodbc
```

输出因系统而异，具体内容如下。

```
com.amazon.athenaodbc-x86_64.Config
com.amazon.athenaodbc-x86_64.Driver
```

或者

```
com.amazon.athenaodbc-arm64.Config
com.amazon.athenaodbc-arm64.Driver
```

在 macOS 上配置数据来源名称

安装好驱动程序后，可以在以下位置找到示例 .odbc.ini 和 .odbcinst.ini 文件：

- 英特尔处理器计算机：/opt/athena/odbc/x86_64/ini/
- ARM 处理器计算机：/opt/athena/odbc/arm64/ini/

使用此位置中的 .ini 文件作为配置 Amazon Athena ODBC 驱动程序和数据来源名称（DSN）的示例。

> **Note**
>
> 默认情况下，ODBC 驱动程序管理器使用位于主目录中的隐藏配置文件 .odbc.ini 和 .odbcinst.ini。

要使用 iODBC 驱动程序管理器指定 .odbc.ini 和 .odbcinst.ini 文件的路径，请执行以下步骤。

使用 iODBC 驱动程序管理器指定 ODBC **.ini** 文件位置

1. 将 ODBCINI 设置到 odbc.ini 文件的完整路径和文件名。
   - 对于配备了英特尔处理器的 macOS 计算机，请使用以下语法。

     ```
     export ODBCINI=/opt/athena/odbc/x86_64/ini/odbc.ini
     ```
   - 对于配备了 ARM 处理器的 macOS 计算机，请使用以下语法。

     ```
     export ODBCINI=/opt/athena/odbc/arm64/ini/odbc.ini
     ```
2. 将 ODBCSYSINI 设置到 odbcinst.ini 文件的完整路径和文件名。
   - 对于配备了英特尔处理器的 macOS 计算机，请使用以下语法。

     ```
     export ODBCSYSINI=/opt/athena/odbc/x86_64/ini/odbcinst.ini
     ```

- 对于配备了 ARM 处理器的 macOS 计算机，请使用以下语法。

```
export ODBCSYSINI=/opt/athena/odbc/arm64/ini/odbcinst.ini
```

3. 如果想要使用数据来源名称（DSN）连接到数据存储，请配置 odbc.ini 文件来定义数据源名称（DSN）。设置 odbc.ini 文件中的属性，以便创建指定数据存储连接信息的 DSN，如下例所示。

```
[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. 配置 odbcinst.ini 文件，如下例所示。

```
[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. 安装并配置 Amazon Athena ODBC 驱动程序后，使用 `iodbctest` 命令行工具来验证连接，如下例所示。

```
username@ % iodbctest
iODBC Demonstration program
This program shows an interactive SQL processor
Driver Manager: 03.52.1623.0502

Enter ODBC connect string (? shows list): ?

DSN                             | Driver
-------------------------------------------------------------------------------
athena_odbc_test                | Amazon Athena ODBC (x64)

Enter ODBC connect string (? shows list): DSN=athena_odbc_test;
Driver: 2.0.2.1 (Amazon Athena ODBC Driver)

SQL>
```

Athena ODBC 2.x 连接参数

Amazon Athena ODBC 配置对话框选项包括身份验证选项、高级选项、日志记录选项、端点覆盖和代理选项。有关每个选项的详细信息，请访问相应的链接。

- 主要 ODBC 2.x 连接参数
- 身份验证选项
- 高级选项
- 日志记录选项
- 端点覆盖
- 代理选项

主要 ODBC 2.x 连接参数

以下各节介绍了每个主要连接参数。

Data source name

指定数据来源的名称。

连接字符串名称	参数类型	默认值	连接字符串示例
DSN	无 DSN 连接类型的可选项	none	DSN=AmazonAthenaOdbcUsWest1;

描述

包含数据来源的描述。

连接字符串名称	参数类型	默认值	连接字符串示例
描述	可选	none	Description=Connection to Amazon Athena us-west-1;

目录

指定数据目录名称。有关目录的更多信息，请参阅《Amazon Athena API 参考》中的 DataCatalog。

连接字符串名称	参数类型	默认值	连接字符串示例
目录	可选	AwsDataCatalog	Catalog=AwsDataCatalog;

区域

指定 AWS 区域。有关 AWS 区域 的更多信息，请参阅区域和可用区。

连接字符串名称	参数类型	默认值	连接字符串示例
AwsRegion	强制性	none	AwsRegion =us-west-1;

数据库

指定数据库名称。有关数据库的更多信息，请参阅 Amazon Athena API 参考中的数据库。

连接字符串名称	参数类型	默认值	连接字符串示例
架构	可选	default	Schema=default;

工作组

指定工作组名称。有关工作组的更多信息，请参阅 Amazon Athena API 参考中的工作组。

连接字符串名称	参数类型	默认值	连接字符串示例
工作组	可选	primary	Workgroup =primary;

输出位置

指定存储查询结果的 Amazon S3 位置。有关输出位置的更多信息，请参阅 Amazon Athena API 参考中的 ResultConfiguration。

连接字符串名称	参数类型	默认值	连接字符串示例
S3OutputLocation	强制性	none	S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/;

加密选项

对话框参数名称：加密选项

指定加密选项。有关加密选项的更多信息，请参阅 Amazon Athena API 参考中的 EncryptionConfiguration。

连接字符串名称	参数类型	默认值	可能的值	连接字符串示例
S3OutputEncOption	可选	none	NOT_SET, SSE_S3, SSE_KMS, CSE_KMS	S3OutputEncOption=SSE_S3;

KMS 密钥

指定用于加密的 KMS 密钥。有关 KMS 密钥的加密配置的更多信息，请参阅 Amazon Athena API 参考中的 EncryptionConfiguration。

连接字符串名称	参数类型	默认值	连接字符串示例
S3OutputEncKMSKey	可选	none	S3OutputEncKMSKey=your_key;

连接测试

ODBC 数据来源管理器提供了一个测试选项，可用于测试 ODBC 2.x 与 Amazon Athena 的连接。要查看步骤，请参阅 在 Windows 上配置数据来源名称。在测试连接时，ODBC 驱动程序会调用 GetWorkGroup Athena API 操作。该调用使用您指定的身份验证类型和相应的凭证提供程序来检索凭证。使用 ODBC 2.x 驱动程序时，连接测试不收费。该测试不会在 Amazon S3 存储桶中生成查询结果。

身份验证选项

您可以使用以下身份验证类型连接到 Amazon Athena。对于所有类型，连接字符串名称为 `AuthenticationType`，参数类型为 `Required`，默认值为 `IAM Credentials`。有关每种身份验证类型的参数的信息，请访问相应的链接。有关常见的身份验证参数，请参阅 通用身份验证参数。

身份验证类型	连接字符串示例
IAM 凭证	`AuthenticationType=IAM Credentials;`
IAM 配置文件	`AuthenticationType=IAM Profile;`
AD FS	`AuthenticationType=ADFS;`
Azure AD	`AuthenticationType=AzureAD;`
浏览器 Azure AD	`AuthenticationType=BrowserAzureAD;`
浏览器 SAML	`AuthenticationType=BrowserSAML;`
浏览器 SSO OIDC	`AuthenticationType=BrowserSSOOIDC;`
默认凭证	`AuthenticationType=Default Credentials;`
外部凭证	`AuthenticationType=External Credentials;`
实例配置文件	`AuthenticationType=Instance Profile;`
JWT	`AuthenticationType=JWT;`
Okta	`AuthenticationType=Okta;`
Ping	`AuthenticationType=Ping;`

IAM 凭证

您可以通过本节中描述的连接字符串参数，使用您的 IAM 凭证通过 ODBC 驱动程序连接到 Amazon Athena。

身份验证类型

连接字符串名称	参数类型	默认值	连接字符串示例
AuthenticationType	必需	IAM Credentials	AuthenticationType=IAM Credentials;

用户 ID

您的 AWS 访问密钥 ID。有关访问密钥的更多信息，请参阅《IAM 用户指南》中的 AWS 安全凭证。

连接字符串名称	参数类型	默认值	连接字符串示例
UID	必需	none	UID=AKIAIOSFODNN7EXAMPLE;

密码

您的 AWS 密钥 ID。有关访问密钥的更多信息，请参阅《IAM 用户指南》中的 AWS 安全凭证。

连接字符串名称	参数类型	默认值	连接字符串示例
PWD	必需	none	PWD=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKE;

会话令牌

如果您使用的是临时 AWS 凭证，则必须指定会话令牌。有关临时凭证的更多信息，请参阅《IAM 用户指南》中的 IAM 临时安全凭证。

连接字符串名称	参数类型	默认值	连接字符串示例
SessionToken	可选	none	`SessionToken=AQoDYXdzEJr...<remainder of session token>;`

IAM 配置文件

您可以使用 ODBC 驱动程序配置命名配置文件以连接到 Amazon Athena。要使用托管 Amazon EC2 实例配置文件中的可用凭证，将 `credential_source` 参数设置为 `Ec2InstanceMetadata`。如果要在命名配置文件中使用自定义凭证提供程序，在配置文件配置中指定 `plugin_name` 参数值。

身份验证类型

连接字符串名称	参数类型	默认值	连接字符串示例
AuthenticationType	必需	`IAM Credentials`	`AuthenticationType=IAM Profile;`

AWS 配置文件

用于 ODBC 连接的配置文件名称。有关命名配置文件的更多信息，请参阅《AWS Command Line Interface 用户指南》中的使用命名配置文件。

连接字符串名称	参数类型	默认值	连接字符串示例
AWS 配置文件	必需	none	`AWSProfile=default;`

Preferred role

担任角色的 Amazon Resource Name（ARN）。当在配置文件配置的 `plugin_name` 参数中指定自定义凭证提供程序时，将使用首选角色参数。有关 ARN 角色的更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

连接字符串名称	参数类型	默认值	连接字符串示例
preferred_role	可选	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

会话持续时间

角色会话的持续时间（以秒为单位）。有关会话持续时间的更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。当在配置文件配置的 `plugin_name` 参数中指定自定义凭证提供程序时，将使用会话持续时间参数。

连接字符串名称	参数类型	默认值	连接字符串示例
duration	可选	`900`	`duration=900;`

插件名称

指定命名配置文件中使用的自定义凭证提供程序的名称。此参数可以采用与 ODBC 数据来源管理器的身份验证类型字段相同的值，但只能在 `AWSProfile` 配置中使用。

连接字符串名称	参数类型	默认值	连接字符串示例
plugin_name	可选	none	`plugin_name=AzureAD;`

AD FS

AD FS 是一个基于 SAML 的身份验证插件，可与 Active Directory 联合身份验证服务（AD FS）身份提供商配合使用。该插件支持集成式 Windows 身份验证和基于表单的身份验证。如果您使用集成式

Windows 身份验证，则可以省略用户名和密码。有关配置 AD FS 和 Athena 的更多信息，请参阅 使用 ODBC 客户端为 Microsoft AD FS 用户配置对 Amazon Athena 的联合访问权限。

身份验证类型

连接字符串名称	参数类型	默认值	连接字符串示例
AuthenticationType	必需	`IAM Credentials`	`AuthenticationType=ADFS;`

用户 ID

用于连接 AD FS 服务器的用户名。对于集成式 Windows 身份验证，可省略用户名。如果您的 AD FS 设置需要用户名，则必须在连接参数中提供该用户名。

连接字符串名称	参数类型	默认值	连接字符串示例
UID	Windows 集成式身份验证的可选项	none	`UID=domain\username;`

密码

用于连接 AD FS 服务器的密码。与用户名字段一样，如果您使用集成式 Windows 身份验证，则可以省略用户名。如果您的 AD FS 设置需要密码，则必须在连接参数中提供该密码。

连接字符串名称	参数类型	默认值	连接字符串示例
PWD	Windows 集成式身份验证的可选项	none	`PWD=password_3EXAMPLE;`

Preferred role

担任角色的 Amazon Resource Name（ARN）。如果您的 SAML 断言具有多个角色，则可以指定此参数来选择要担任的角色。此角色应存在于 SAML 断言中。有关 ARN 角色的更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

连接字符串名称	参数类型	默认值	连接字符串示例
preferred_role	可选	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

会话持续时间

角色会话的持续时间（以秒为单位）。有关会话持续时间的更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

连接字符串名称	参数类型	默认值	连接字符串示例
duration	可选	900	`duration=900;`

IdP host

AD FS 服务主机的名称。

连接字符串名称	参数类型	默认值	连接字符串示例
idp_host	Require	none	`idp_host=<server-name>.<company.com>;`

IdP port

用于连接 AD FS 主机的端口。

连接字符串名称	参数类型	默认值	连接字符串示例
idp_port	必需	none	`idp_port=443;`

LoginToRP

受信任的依赖方。使用此参数覆盖 AD FS 依赖方端点 URL。

连接字符串名称	参数类型	默认值	连接字符串示例
LoginToRP	可选	`urn:amazon:webservices`	`LoginToRP=trustedparty;`

Azure AD

Azure AD 是一个基于 SAML 的身份验证插件，可与 Azure AD 身份提供商配合使用。该插件不支持多重身份验证（MFA）。如果您需要 MFA 支持，请考虑改用 `BrowserAzureAD` 插件。

身份验证类型

连接字符串名称	参数类型	默认值	连接字符串示例
AuthenticationType	必需	`IAM Credentials`	`AuthenticationType=AzureAD;`

Preferred role

担任角色的 Amazon Resource Name（ARN）。有关 ARN 角色的信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

连接字符串名称	参数类型	默认值	连接字符串示例
preferred_role	可选	none	`preferred_role=arn:aws:iam::123456789012:id/user1;`

会话持续时间

角色会话的持续时间（以秒为单位）。有关更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

连接字符串名称	参数类型	默认值	连接字符串示例
duration	可选	900	duration=900;

租户 ID

指定应用程序租户 ID。

连接字符串名称	参数类型	默认值	连接字符串示例
idp_tenant	必需	none	idp_tenant=123zz112z-z12d-1z1f-11zz-f111aa111234;

客户端 ID

指定应用程序客户端 ID。

连接字符串名称	参数类型	默认值	连接字符串示例
client_id	必需	none	client_id=9178ac27-a1bc-1a2b-1a2b-a123abcd1234;

客户端密钥

指定客户端密钥。

连接字符串名称	参数类型	默认值	连接字符串示例
client_secret	必需	none	client_secret=zG12q~.xzG1xxxZ1wX1.~ZzXXX1XxkHZizeT1zzZ;

浏览器 Azure AD

浏览器 Azure AD 是一个基于 SAML 的身份验证插件，可与 Azure AD 身份提供商配合使用并支持多重身份验证。与标准 Azure AD 插件不同，该插件不需要在连接参数中输入用户名、密码或客户端密钥。

身份验证类型

连接字符串名称	参数类型	默认值	连接字符串示例
AuthenticationType	必需	`IAM Credentials`	`AuthenticationType=BrowserAzureAD;`

Preferred role

担任角色的 Amazon Resource Name（ARN）。如果您的 SAML 断言具有多个角色，则可以指定此参数来选择要担任的角色。指定的角色应存在于 SAML 断言中。有关 ARN 角色的更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

连接字符串名称	参数类型	默认值	连接字符串示例
preferred_role	可选	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

会话持续时间

角色会话的持续时间（以秒为单位）。有关会话持续时间的更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

连接字符串名称	参数类型	默认值	连接字符串示例
duration	可选	900	`duration=900;`

租户 ID

指定应用程序租户 ID。

连接字符串名称	参数类型	默认值	连接字符串示例
idp_tenant	必需	none	idp_tenant=123zz112z-z12d-1z1f-11zz-f111aa111234;

客户端 ID

指定应用程序客户端 ID。

连接字符串名称	参数类型	默认值	连接字符串示例
client_id	必需	none	client_id=9178ac27-a1bc-1a2b-1a2b-a123abcd1234;

超时

插件停止等待 Azure AD 发出 SAML 响应之前的持续时间，以秒为单位。

连接字符串名称	参数类型	默认值	连接字符串示例
timeout	可选	120	timeout=90;

启用 Azure 文件缓存

启用临时凭证缓存。此连接参数允许在多个进程之间缓存和重复使用临时凭证。当您使用 Microsoft Power BI 等 BI 工具时，使用此选项可以减少打开的浏览器窗口数量。

连接字符串名称	参数类型	默认值	连接字符串示例
browser_azure_cache	可选	1	browser_azure_cache=0;

浏览器 SAML

浏览器 SAML 是一个通用身份验证插件，可与基于 SAML 的身份提供商配合使用并支持多重身份验证。有关详细的配置信息，请参阅 使用 ODBC、SAML 2.0 和 Okta 身份提供商配置单点登录。

身份验证类型

连接字符串名称	参数类型	默认值	连接字符串示例
AuthenticationType	必需	`IAM Credentials`	`AuthenticationType=BrowserSAML;`

Preferred role

担任角色的 Amazon Resource Name（ARN）。如果您的 SAML 断言具有多个角色，则可以指定此参数来选择要担任的角色。此角色应存在于 SAML 断言中。有关 ARN 角色的更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

连接字符串名称	参数类型	默认值	连接字符串示例
preferred_role	可选	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

会话持续时间

角色会话的持续时间（以秒为单位）。有关更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

连接字符串名称	参数类型	默认值	连接字符串示例
duration	可选	`900`	`duration=900;`

登录 URL

为应用程序显示的单点登录 URL。

连接字符串名称	参数类型	默认值	连接字符串示例
login_url	必需	none	`login_url =https:// trial-123 4567.okta.com/ app/trial-123 4567_okta browsersaml_1/ zzz4izzzAzDFB zZz1234/sso/ saml;`

侦听端口

用于侦听 SAML 响应的端口号。此值应与配置 IdP 时使用的 IAM Identity Center URL 相匹配（例如，`http://localhost:7890/athena`）。

连接字符串名称	参数类型	默认值	连接字符串示例
listen_port	可选	`7890`	`listen_po rt=7890;`

超时

插件停止等待身份提供商发出 SAML 响应之前的持续时间，以秒为单位。

连接字符串名称	参数类型	默认值	连接字符串示例
timeout	可选	`120`	`timeout=90;`

浏览器 SSO OIDC

浏览器 SSO OIDC 是一个可与 AWS IAM Identity Center 配合使用的身份验证插件。有关启用和使用 IAM Identity Center 的信息，请参阅《AWS IAM Identity Center 用户指南》中的步骤 1：启用 IAM Identity Center。

身份验证类型

连接字符串名称	参数类型	默认值	连接字符串示例
AuthenticationType	必需	`IAM Credentials`	`AuthenticationType=BrowserSSOOIDC;`

IAM Identity Center 启动 URL

AWS 访问门户的 URL。IAM Identity Center StartDeviceAuthorization API 操作将此值用于 `startUrl` 参数。

复制 AWS 访问门户 URL

1. 登录 AWS Management Console，打开 AWS IAM Identity Center 控制台：https://console.aws.amazon.com/singlesignon/。
2. 在导航窗格中，选择 Settings（设置）。
3. 在设置页面的身份源下，选择 AWS 访问门户 URL 的剪贴板图标。

连接字符串名称	参数类型	默认值	连接字符串示例
sso_oidc_start_url	必需	none	`sso_oidc_start_url=https://app_id.awsapps.com/start;`

IAM Identity Center 区域

配置 SSO 的 AWS 区域。`SSOOIDCClient` 和 `SSOClient` AWS SDK 客户端将此值用于 `region` 参数。

连接字符串名称	参数类型	默认值	连接字符串示例
sso_oidc_region	必需	none	`sso_oidc_region=us-east-1;`

范围

客户端定义的范围列表。进行授权时，此列表会限制授予访问令牌时的权限。IAM Identity Center RegisterClient API 操作将此值用于 scopes 参数。

连接字符串名称	参数类型	默认值	连接字符串示例
sso_oidc_scopes	可选	none	sso_oidc_scopes=scope1,scope2,scope3;

账户 ID

分配给用户的 AWS 账户 的标识符。IAM Identity Center GetRoleCredentials API 将此值用于 accountId 参数。

连接字符串名称	参数类型	默认值	连接字符串示例
sso_oidc_account_id	必需	none	sso_oidc_account_id=123456789123;

角色名称

分配给用户的角色的友好名称。您为此权限集指定的名称将作为可用角色出现在 AWS 访问门户中。IAM Identity Center GetRoleCredentials API 操作将此值用于 roleName 参数。

连接字符串名称	参数类型	默认值	连接字符串示例
sso_oidc_role_name	必需	none	sso_oidc_role_name=AthenaReadAccess;

超时

轮询 SSO API 需要检查是否存在访问令牌的秒数。

连接字符串名称	参数类型	默认值	连接字符串示例
sso_oidc_timeout	可选	120	sso_oidc_timeout=60;

启用文件缓存

启用临时凭证缓存。此连接参数允许在多个进程之间缓存和重复使用临时凭证。当您使用 Microsoft Power BI 等 BI 工具时，使用此选项可以减少打开的浏览器窗口数量。

连接字符串名称	参数类型	默认值	连接字符串示例
sso_oidc_cache	可选	1	sso_oidc_cache=0;

默认凭证

您可以使用在客户端系统上配置的默认凭证来连接 Amazon Athena。有关使用默认凭证的信息，请参阅《AWS SDK for Java 开发人员指南》中的使用默认凭证提供程序链。

身份验证类型

连接字符串名称	参数类型	默认值	连接字符串示例
AuthenticationType	必需	IAM Credentials	AuthenticationType=DefaultCredentials;

外部凭证

外部凭证是一个通用身份验证插件，可用于连接到任何外部基于 SAML 的身份提供商。要使用该插件，需要传递一个返回 SAML 响应的可执行文件。

身份验证类型

连接字符串名称	参数类型	默认值	连接字符串示例
AuthenticationType	必需	`IAM Credentials`	`AuthenticationType=External Credentials;`

可执行文件路径

具有基于 SAML 的自定义凭证提供程序逻辑的可执行文件的路径。可执行文件的输出必须是来自身份提供商的已解析 SAML 响应。

连接字符串名称	参数类型	默认值	连接字符串示例
ExecutablePath	必需	none	ExecutablePath=C:\Users*user_name* *external_credential.exe*

参数列表

要传递给可执行文件的参数列表。

连接字符串名称	参数类型	默认值	连接字符串示例
ArgumentList	可选	none	`ArgumentList=` *arg1 arg2 arg3*

实例配置文件

此身份验证类型用于 EC2 实例，并通过 Amazon EC2 元数据服务提供。

身份验证类型

连接字符串名称	参数类型	默认值	连接字符串示例
AuthenticationType	必需	`IAM Credentials`	`AuthenticationType=Instance Profile;`

JWT

JWT（JSON Web 令牌）插件提供了一个使用 JSON Web 令牌担任 Amazon IAM 角色的接口。配置取决于身份提供商。有关为 Google Cloud 和 AWS 配置联合身份的信息，请参阅 Google Cloud 文档中的使用 AWS 或 Azure 配置工作负载身份联合验证。

身份验证类型

连接字符串名称	参数类型	默认值	连接字符串示例
AuthenticationType	必需	`IAM Credentials`	`AuthenticationType=JWT;`

Preferred role

担任角色的 Amazon Resource Name（ARN）。有关 ARN 角色的更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

连接字符串名称	参数类型	默认值	连接字符串示例
preferred_role	可选	none	`preferred_role=arn:aws:IAM::123456789012:id/user1;`

会话持续时间

角色会话的持续时间（以秒为单位）。有关会话持续时间的更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

连接字符串名称	参数类型	默认值	连接字符串示例
duration	可选	900	duration=900;

JSON Web 令牌

用于使用 AssumeRoleWithWebIdentity AWS STS API 操作检索 IAM 临时凭证的 JSON Web 令牌。有关为 Google Cloud Platform（GCP）用户生成 JSON Web 令牌的信息，请参阅 Google Cloud 文档中的使用 JWT OAuth 令牌。

连接字符串名称	参数类型	默认值	连接字符串示例
web_identity_token	必需	none	web_identity_token=eyJhbGc...<remainder of token>;

角色会话名称

会话的名称。常用方法是使用应用程序用户的名称或标识符作为角色会话名称。这样方便将应用程序使用的临时安全凭证与相应的用户相关联。

连接字符串名称	参数类型	默认值	连接字符串示例
role_session_name	必需	none	role_session_name=familiarname;

Okta

Okta 是一个基于 SAML 的身份验证插件，可与 Okta 身份提供商配合使用。有关为 Okta 和 Amazon Athena 配置联合身份验证的信息，请参阅 使用 Okta 插件和 Okta 身份提供者为 ODBC 配置 SSO。

身份验证类型

连接字符串名称	参数类型	默认值	连接字符串示例
AuthenticationType	必需	IAM Credentials	AuthenticationType=Okta;

用户 ID

您的 Okta 用户名。

连接字符串名称	参数类型	默认值	连接字符串示例
UID	必需	none	UID=jane.doe@org.com;

密码

您的 Okta 用户密码。

连接字符串名称	参数类型	默认值	连接字符串示例
PWD	必需	none	PWD=oktauserpasswordexample;

Preferred role

担任角色的 Amazon Resource Name（ARN）。有关 ARN 角色的更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

连接字符串名称	参数类型	默认值	连接字符串示例
preferred_role	可选	none	preferred_role=arn

连接字符串名称	参数类型	默认值	连接字符串示例
			:aws:IAM::123456789012:id/user1;

会话持续时间

角色会话的持续时间（以秒为单位）。有关更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

连接字符串名称	参数类型	默认值	连接字符串示例
duration	可选	900	duration=900;

IdP host

您的 Okta 组织的 URL。您可以从 Okta 应用程序中的嵌入链接 URL 中提取 idp_host 参数。要查看步骤，请参阅 从 Okta 检索 ODBC 配置信息。https:// 之后的第一个分段一直到 okta.com（包括在内）都是您的 IdP 主机（例如 http://trial-1234567.okta.com）。

连接字符串名称	参数类型	默认值	连接字符串示例
idp_host	必需	None	idp_host=dev-99999999.okta.com;

IdP port

用于连接 IdP 主机的端口号。

连接字符串名称	参数类型	默认值	连接字符串示例
idp_port	必需	None	idp_port=443;

Okta app ID

适用于您的应用程序的两部分标识符。您可以从 Okta 应用程序中的嵌入链接 URL 中提取 app_id 参数。要查看步骤，请参阅 从 Okta 检索 ODBC 配置信息。应用程序 ID 是 URL 的最后两个分段，包括中间的正斜杠。这两个分段是两个长度为 20 个字符的字符串，其中包含数字和大小写字母（例如 Abc1de2fghi3J45kL678/abc1defghij2klmNo3p4）。

连接字符串名称	参数类型	默认值	连接字符串示例
app_id	必需	None	app_id=0oa25kx8ze9A3example/alnexamplea0piaWa0g7;

Okta 应用程序名称

Okta 应用程序的名称。

连接字符串名称	参数类型	默认值	连接字符串示例
app_name	必需	None	app_name=amazon_aws_redshift;

Okta 等待时间

指定等待多重身份验证（MFA）代码的持续时间（以秒为单位）。

连接字符串名称	参数类型	默认值	连接字符串示例
okta_mfa_wait_time	可选	10	okta_mfa_wait_time=20;

Okta MFA 类型

MFA 因素类型。支持的类型包括 Google Authenticator、SMS（Okta）、Okta Verify with Push 和 Okta Verify with TOTP。各个组织的安全策略决定用户登录是否需要 MFA。

连接字符串名称	参数类型	默认值	可能的值	连接字符串示例
okta_mfa_type	Optional	None	googleauthenticator, smsauthentication, oktaverifywithpush, oktaverifywithtotp	okta_mfa_type=oktaverifywithpush;

Okta 电话号码

用于 AWS SMS 身份验证的电话号码。此参数仅是多重注册的必需参数。如果您的手机号码已注册，或者如果安全策略未使用 AWS SMS 身份验证，则可以忽略此字段。

连接字符串名称	参数类型	默认值	连接字符串示例
okta_mfa_phone_number	对于 MFA 注册为必需项，否则为可选项	None	okta_mfa_phone_number=19991234567;

启用 Okta 文件缓存

启用临时凭证缓存。此连接参数允许在 BI 应用程序打开的多个进程之间缓存和重复使用临时凭证。使用此选项可以避免 Okta API 的节流限制。

连接字符串名称	参数类型	默认值	连接字符串示例
okta_cache	可选	0	okta_cache=1;

Ping

Ping 是一个基于 SAML 的插件，可与 PingFederate 身份提供商配合使用。

身份验证类型

连接字符串名称	参数类型	默认值	连接字符串示例
AuthenticationType	必需	`IAM Credentials`	`AuthenticationType=Ping;`

用户 ID

PingFederate 服务器的用户名。

连接字符串名称	参数类型	默认值	连接字符串示例
UID	必需	`none`	`UID=pingusername@domain.com;`

密码

PingFederate 服务器的密码。

连接字符串名称	参数类型	默认值	连接字符串示例
PWD	必需	`none`	`PWD=pingpassword;`

Preferred role

担任角色的 Amazon Resource Name（ARN）。如果您的 SAML 断言具有多个角色，则可以指定此参数来选择要担任的角色。此角色应存在于 SAML 断言中。有关 ARN 角色的更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

连接字符串名称	参数类型	默认值	连接字符串示例
preferred_role	可选	none	`preferred_role=arn:aws:iam::123456789012:id/user1;`

会话持续时间

角色会话的持续时间（以秒为单位）。有关会话持续时间的更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRole。

连接字符串名称	参数类型	默认值	连接字符串示例
duration	可选	900	`duration=900;`

IdP host

您的 Ping 服务器的地址。要查找您的地址，请访问以下 URL 并查看 SSO 应用程序端点字段。

```
https://your-pf-host-#:9999/pingfederate/your-pf-app#/spConnections
```

连接字符串名称	参数类型	默认值	连接字符串示例
idp_host	必需	none	`idp_host=ec2-1-83-65-12.compute-1.amazonaws.com;`

IdP port

用于连接 IdP 主机的端口号。

连接字符串名称	参数类型	默认值	连接字符串示例
idp_port	必需	None	`idp_port=443;`

合作伙伴 SPID

服务提供商地址。要查找服务提供商地址，请访问以下 URL 并查看 SSO 应用程序端点字段。

```
https://your-pf-host-#:9999/pingfederate/your-pf-app#/spConnections
```

连接字符串名称	参数类型	默认值	连接字符串示例
partner_spid	必需	None	partner_spid=https://us-east-1.signin.aws.amazon.com/platform/saml/<...>;

Ping URI 参数

将身份验证请求的 URI 参数传递给 Ping。使用此参数来绕过 Lake Formation 的单个角色限制。配置 Ping 识别传递的参数，并验证传递的角色是否存在于分配给用户的角色列表中。然后，在 SAML 断言中发送单个角色。

连接字符串名称	参数类型	默认值	连接字符串示例
ping_uri_param	可选	None	ping_uri_param=role=my_iam_role;

通用身份验证参数

本节中的参数在所述身份验证类型中通用。

对 IdP 使用代理

通过代理在驱动程序和 IdP 之间启用通信。此选项适用于以下身份验证插件：

- AD FS
- Azure AD
- 浏览器 Azure AD
- 浏览器 SSO OIDC
- JWT
- Okta
- Ping

连接字符串名称	参数类型	默认值	连接字符串示例
UseProxyForIdP	可选	0	UseProxyForIdP=1;

使用 Lake Formation

使用 AssumeDecoratedRoleWithSAML Lake Formation API 操作检索临时 IAM 凭证，而不是 AssumeRoleWithSAML AWS STS API 操作。此选项适用于 Azure AD、浏览器 Azure AD、浏览器 SAML、Okta、Ping 和 AD FS 身份验证插件。

连接字符串名称	参数类型	默认值	连接字符串示例
LakeformationEnabled	可选	0	LakeformationEnabled=1;

SSL 不安全（IdP）

与 IdP 通信时禁用 SSL。此选项适用于 Azure AD、浏览器 Azure AD、Okta、Ping 和 AD FS 身份验证插件。

连接字符串名称	参数类型	默认值	连接字符串示例
SSL_Insecure	可选	0	SSL_Insecure=1;

端点覆盖

Athena 端点覆盖

`endpointOverride` `ClientConfiguration` 类使用此值覆盖 Amazon Athena 客户端的默认 HTTP 端点。有关更多信息，请参阅《AWS SDK for C++ 开发人员指南》中的 AWS 客户端配置。

连接字符串名称	参数类型	默认值	连接字符串示例
EndpointOverride	可选	none	EndpointOverride=athena.us-west-2.amazonaws.com;

Athena 流式处理端点覆盖

`ClientConfiguration.endpointOverride` 方法使用此值覆盖 Amazon Athena 流式处理客户端的默认 HTTP 端点。有关更多信息，请参阅《AWS SDK for C++ 开发人员指南》中的 AWS 客户端配置。Athena 流式处理服务通过端口 444 提供。

连接字符串名称	参数类型	默认值	连接字符串示例
StreamingEndpointOverride	可选	none	`StreamingEndpointOverride=athena.us-west-1.amazonaws.com:444;`

AWS STS 端点覆盖

`ClientConfiguration.endpointOverride` 方法使用此值覆盖 AWS STS 客户端的默认 HTTP 端点。有关更多信息，请参阅《AWS SDK for C++ 开发人员指南》中的 AWS 客户端配置。

连接字符串名称	参数类型	默认值	连接字符串示例
StsEndpointOverride	可选	none	`StsEndpointOverride=sts.us-west-1.amazonaws.com;`

Lake Formation 端点覆盖

`ClientConfiguration.endpointOverride` 方法使用此值覆盖 Lake Formation 客户端的默认 HTTP 端点。有关更多信息，请参阅《AWS SDK for C++ 开发人员指南》中的 AWS 客户端配置。

连接字符串名称	参数类型	默认值	连接字符串示例
LakeFormationEndpointOverride	可选	none	`LakeFormationEndpointOverride=lakeformation.us-west-1.amazonaws.com;`

SSO 端点覆盖

`ClientConfiguration.endpointOverride` 方法使用此值覆盖 SSO 客户端的默认 HTTP 端点。有关更多信息，请参阅《AWS SDK for C++ 开发人员指南》中的 AWS 客户端配置。

连接字符串名称	参数类型	默认值	连接字符串示例
SSOEndpoi ntOverride	可选	none	`SSOEndpointOverride=portal.sso.us-east-2.amazonaws.com;`

SSO OIDC 端点覆盖

`ClientConfiguration.endpointOverride` 方法使用此值覆盖 SSO OIDC 客户端的默认 HTTP 端点。有关更多信息，请参阅《AWS SDK for C++ 开发人员指南》中的 AWS 客户端配置。

连接字符串名称	参数类型	默认值	连接字符串示例
SSOOIDCEndpointOve rride	可选	none	`SSOOIDCEndpointOverride=oidc.us-east-2.amazonaws.com`

高级选项

提取大小

要在此请求中返回的最大结果（行）数。有关参数信息，请参阅 GetQuery MaxResults。对于流式处理 API，最大值为 10000000。

连接字符串名称	参数类型	默认值	连接字符串示例
RowsToFetchPerBloc k	可选	`1000`（对于非流式处理） `20000`（对于流式处理）	`RowsToFetchPerBlock=20000;`

启用结果重用

指定运行查询时是否可以重用先前查询结果。有关参数信息，请参阅 ResultReuseByAgeConfiguration。

连接字符串名称	参数类型	默认值	连接字符串示例
EnableResultReuse	可选	0	EnableResultReuse=1;

结果重用最长使用期限

以分钟为单位指定 Athena 应考虑的先前查询结果的重用最长使用期限。有关参数信息，请参阅 ResultReuseByAgeConfiguration。

连接字符串名称	参数类型	默认值	连接字符串示例
ReusedResultMaxAgeInMinutes	可选	60	ReusedResultMaxAgeInMinutes=90;

启用流式处理 API

选择是否使用 Athena 流式处理 API 来获取结果集。

连接字符串名称	参数类型	默认值	连接字符串示例
UseResultsetStreaming	可选	0	UseResultsetStreaming=1;

启用 S3 提取器

通过直接与 Amazon S3 交互，从 Amazon S3 存储桶中获取 Athena 生成的结果集。

连接字符串名称	参数类型	默认值	连接字符串示例
EnableS3Fetcher	可选	1	EnableS3Fetcher=1;

使用多个 S3 线程

使用多个线程从 Amazon S3 获取数据。启用此选项后，存储在 Amazon S3 存储桶中的结果文件将使用多个线程并行获取。

仅当网络带宽适合时，才启用此选项。例如，在我们对 EC2 c5.2xlarge 实例进行的测量中，单线程 S3 客户端的网络吞吐量达到了 1Gbps，而多线程 S3 客户端达到了 4Gbps。

连接字符串名称	参数类型	默认值	连接字符串示例
UseMultipleS3Threads	可选	0	UseMultipleS3Threads=1;

使用单一目录和架构

默认情况下，ODBC 驱动程序会查询 Athena 以获取可用目录和架构列表。此选项将强制驱动程序使用 ODBC 数据来源管理器配置对话框或连接参数指定的目录和架构。

连接字符串名称	参数类型	默认值	连接字符串示例
UseSingleCatalogAndSchema	可选	0	UseSingleCatalogAndSchema=1;

使用查询来列出表

对于 LAMBDA 目录类型，允许 ODBC 驱动程序提交 SHOW TABLES 查询来获取可用表的列表。此设置是默认设置。如果此参数设置为 0，则 ODBC 驱动程序将使用 Athena ListTableMetadata API 来获取可用表的列表。请注意，对于 LAMBDA 目录类型，使用 ListTableMetadata 会导致性能下降。

连接字符串名称	参数类型	默认值	连接字符串示例
UseQueryToListTables	可选	1	UseQueryToListTables=1;

使用 WCHAR 作为字符串类型

默认情况下，ODBC 驱动程序在 Athena 中使用 SQL_CHAR 和 SQL_VARCHAR 字符串数据类型 char、varchar、string、array、map<>、struct<> 和 row。将此参数设置为 1，会强制驱动程序将 SQL_WCHAR 和 SQL_WVARCHAR 作为字符串数据类型。使用宽字符和宽可变字符类型，可确保正确存储和检索不同语言的字符。

连接字符串名称	参数类型	默认值	连接字符串示例
UseWCharForStringTypes	可选	0	UseWCharForStringTypes=1;

查询外部目录

指定驱动程序是否需要从 Athena 查询外部目录。有关更多信息，请参阅 迁移到 ODBC 2.x 驱动程序。

连接字符串名称	参数类型	默认值	连接字符串示例
QueryExternalCatalogs	可选	0	QueryExternalCatalogs=1;

验证 SSL

控制在使用 AWS SDK 时是否验证 SSL 证书。此值将传递给 ClientConfiguration.verifySSL 参数。有关更多信息，请参阅《AWS SDK for C++ 开发人员指南》中的 AWS 客户端配置。

连接字符串名称	参数类型	默认值	连接字符串示例
VerifySSL	可选	1	VerifySSL=0;

S3 结果块大小

以字节为单位指定要为单一 Amazon S3 GetObject API 请求下载的数据块大小。默认值为 67108864（64MB）。允许的最小值和最大值分别为 10485760（10MB）和 2146435072（约 2GB）。

连接字符串名称	参数类型	默认值	连接字符串示例
S3ResultBlockSize	可选	67108864	S3ResultBlockSize=268435456;

字符串列长度

为具有 string 数据类型的列指定列长度。由于 Athena 使用 Apache Hive 字符串数据类型，但该数据类型并未定义精度，所以 Athena 报告的默认长度为 2147483647 (INT_MAX)。商业智能工具通常会为列预先分配内存，这可能会导致内存消耗较高。为避免这种情况，Athena ODBC 驱动程序会限制 string 数据类型列的报告精度，也会公开 StringColumnLength 连接参数，以便可以更改默认值。

连接字符串名称	参数类型	默认值	连接字符串示例
StringColumnLength	可选	255	StringColumnLength=65535;

复杂类型列长度

为具有复杂数据类型（如 map、struct 和 array）的列指定列长度。与 StringColumnLength 一样，对于具有复杂数据类型的列，Athena 报告的精度为 0。Athena ODBC 驱动程序会为具有复杂数据类型的列设置默认精度，也会公开 ComplexTypeColumnLength 连接参数，以便可以更改默认值。

连接字符串名称	参数类型	默认值	连接字符串示例
ComplexTypeColumnLength	可选	65535	ComplexTypeColumnLength=123456;

可信 CA 证书

指示 HTTP 客户端查找 SSL 证书信任存储的位置。此值将传递给 ClientConfiguration.caFile 参数。有关更多信息，请参阅《AWS SDK for C++ 开发人员指南》中的 AWS 客户端配置。

连接字符串名称	参数类型	默认值	连接字符串示例
TrustedCerts	可选	%INSTALL_PATH%/bin	TrustedCerts=C:\\Program Files\\Amazon Athena ODBC Driver\\bin\\cacert.pem;

最小轮询期

指定在 Athena 中轮询查询执行状态之前要等待的最小时间值（以毫秒为单位）。

连接字符串名称	参数类型	默认值	连接字符串示例
MinQueryExecutionPollingInterval	可选	100	MinQueryExecutionPollingInterval=200;

最大轮询期

指定在 Athena 中轮询查询执行状态之前要等待的最大时间值（以毫秒为单位）。

连接字符串名称	参数类型	默认值	连接字符串示例
MaxQueryExecutionPollingInterval	可选	60000	MaxQueryExecutionPollingInterval=1000;

轮询乘数

指定增加轮询期的系数。默认情况下，轮询以最小轮询期值开始，每次轮询时都翻倍，直到达到最大轮询期值。

连接字符串名称	参数类型	默认值	连接字符串示例
QueryExecutionPollingIntervalMultiplier	可选	2	QueryExecutionPollingIntervalMultiplier=2;

最长轮询持续时间

指定驱动程序可以在 Athena 中轮询查询执行状态的最大时间值（以毫秒为单位）。

连接字符串名称	参数类型	默认值	连接字符串示例
MaxPollDuration	可选	1800000	MaxPollDuration=1800000;

连接超时

HTTP 连接等待建立连接的时间长度（以毫秒为单位）。此值是为 ClientConfiguration.connectTimeoutMs Athena 客户端设置的。如果未指定，则使用 curl 的默认值。有关连接参数的信息，请参阅《AWS SDK for Java 开发人员指南》中的客户端配置。

连接字符串名称	参数类型	默认值	连接字符串示例
ConnectionTimeout	可选	0	ConnectionTimeout=2000;

请求超时

指定 HTTP 客户端的套接字读取超时。此值是为 Athena 客户端的 ClientConfiguration.requestTimeoutMs 参数设置的。有关更多信息，请参阅《AWS SDK for Java 开发人员指南》中的客户端配置。

连接字符串名称	参数类型	默认值	连接字符串示例
RequestTimeout	可选	10000	RequestTimeout=30000;

代理选项

代理主机

如果您要求用户经过代理，使用此参数来设置代理主机。此参数对应于 AWS SDK 中的 `ClientConfiguration.proxyHost` 参数。有关更多信息，请参阅《AWS SDK for C++ 开发人员指南》中的 AWS 客户端配置。

连接字符串名称	参数类型	默认值	连接字符串示例
ProxyHost	可选	none	`ProxyHost =127.0.0.1;`

代理端口

使用此参数设置代理端口。此参数对应于 AWS SDK 中的 `ClientConfiguration.proxyPort` 参数。有关更多信息，请参阅《AWS SDK for C++ 开发人员指南》中的 AWS 客户端配置。

连接字符串名称	参数类型	默认值	连接字符串示例
ProxyPort	可选	none	`ProxyPort=8888;`

代理用户名

使用此参数设置代理用户名。此参数对应于 AWS SDK 中的 `ClientConfiguration.proxyUserName` 参数。有关更多信息，请参阅《AWS SDK for C++ 开发人员指南》中的 AWS 客户端配置。

连接字符串名称	参数类型	默认值	连接字符串示例
ProxyUID	可选	none	`ProxyUID= username;`

代理密码

使用此参数设置代理密码。此参数对应于 AWS SDK 中的 `ClientConfiguration.proxyPassword` 参数。有关更多信息，请参阅《AWS SDK for C++ 开发人员指南》中的 AWS 客户端配置。

连接字符串名称	参数类型	默认值	连接字符串示例
ProxyPWD	可选	none	`ProxyPWD=password;`

非代理主机

使用此可选参数指定驱动程序在不使用代理的情况下连接到的主机。此参数对应于 AWS SDK 中的 `ClientConfiguration.nonProxyHosts` 参数。有关更多信息，请参阅《AWS SDK for C++ 开发人员指南》中的 AWS 客户端配置。

`NonProxyHost` 连接参数传递给 `CURLOPT_NOPROXY` curl 选项。有关 `CURLOPT_NOPROXY` 格式的信息，请参阅 curl 文档中的 CURLOPT_NOPROXY。

连接字符串名称	参数类型	默认值	连接字符串示例
NonProxyHost	可选	none	`NonProxyHost=.amazonaws.com,localhost,.example.net,.example.com;`

使用代理

启用通过指定代理的用户流量。

连接字符串名称	参数类型	默认值	连接字符串示例
UseProxy	可选	none	`UseProxy=1;`

日志记录选项

要修改在此描述的设置，需要管理员权限。要进行更改，可使用 ODBC 数据来源管理器日志记录选项对话框或直接修改 Windows 注册表。

日志级别

此选项用于启用 ODBC 驱动程序日志。在 Windows 中，您可以使用注册表或对话框来启用或禁用日志记录。该选项位于以下注册表路径中：

```
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Amazon Athena\ODBC\Driver
```

连接字符串名称	参数类型	默认值	连接字符串示例
LogLevel	可选	0	LogLevel=1;

日志路径

指定存储 ODBC 驱动程序日志的文件的路径。您可以使用注册表或对话框来设置此值。该选项位于以下注册表路径中：

```
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Amazon Athena\ODBC\Driver
```

连接字符串名称	参数类型	默认值	连接字符串示例
LogPath	可选	none	LogPath=C:\Users\ *username*\projects\internal\trunk\;

使用 AWS 记录器

指定是否启用 AWS SDK 日志记录。要启用，指定 1；要禁用，指定 0。

连接字符串名称	参数类型	默认值	连接字符串示例
UseAwsLogger	可选	0	UseAwsLogger=1;

迁移到 ODBC 2.x 驱动程序

由于大多数 Athena ODBC 2.x 连接参数都与 ODBC 1.x 驱动程序向后兼容，因此您可以在 Athena ODBC 2.x 驱动程序中重用大部分现有连接字符串。但是，需要修改以下连接参数。

日志级别

虽然当前 ODBC 驱动程序提供了一系列可用日志记录选项，从 LOG_OFF （0）到 LOG_TRACE （6），但 Amazon Athena ODBC 驱动程序只有两个值：0（禁用）和 1（启用）。

有关记录 ODBC 2.x 驱动程序的更多信息，请参阅 日志记录选项。

	ODBC 1.x 驱动程序	ODBC 2.x 驱动程序
连接字符串名称	LogLevel	LogLevel
参数类型	可选	可选
默认值	0	0
可能的值	0-6	0,1
连接字符串示例	LogLevel=6;	LogLevel=1;

MetadataRetrievalMethod

当前 ODBC 驱动程序提供了多个用于从 Athena 检索元数据的选项。Amazon Athena ODBC 驱动程序已弃用 MetadataRetrievalMethod，并始终使用 Amazon Athena API 提取元数据。

Athena 引入了用于查询外部目录的 QueryExternalCatalogs 标志。要使用当前 ODBC 驱动程序查询外部目录，将 MetadataRetrievalMethod 设置为 ProxyAPI。要使用 Athena ODBC 驱动程序查询外部目录，将 QueryExternalCatalogs 设置为 1。

	ODBC 1.x 驱动程序	ODBC 2.x 驱动程序
连接字符串名称	MetadataRetrievalMethod	QueryExternalCatalogs
参数类型	可选	可选

	ODBC 1.x 驱动程序	ODBC 2.x 驱动程序
默认值	`Auto`	`0`
可能的值	`Auto`, `AWS Glue`, `ProxyAPI`, `Query`	`0`,`1`
连接字符串示例	`MetadataRetrievalMethod=ProxyAPI;`	`QueryExternalCatalogs=1;`

连接测试

当您测试 ODBC 1.x 驱动程序连接时，该驱动程序会运行一个 `SELECT 1` 查询，以在 Amazon S3 存储桶中生成两个文件：一个用于结果集，另一个用于元数据。测试连接根据 Amazon Athena 定价策略进行收费。

在测试 ODBC 2.x 驱动程序连接时，该驱动程序会调用 GetWorkGroup Athena API 操作。该调用使用您指定的身份验证类型和相应的凭证提供程序来检索凭证。在使用 ODBC 2.x 驱动程序时，不会收取连接测试费用，并且测试不会在 Amazon S3 存储桶中生成查询结果。

对 ODBC 2.x 驱动程序进行故障排除

如果您在使用 Amazon Athena ODBC 驱动程序时遇到问题，可以联系 AWS Support（在 AWS Management Console 中，选择支持、支持中心）。

一定要包含以下信息，并提供有助于支持团队了解您的使用案例的任何其他详细信息。

- 描述 -（必填）此描述包含有关您的使用案例的详细信息以及预期行为和观测到的行为之间的差异。包括任何可以帮助支持工程师轻松解决问题的信息。如果问题是间歇性的，指定问题的发生日期、时间戳或间隔点。
- 版本信息 -（必填）有关您使用的驱动程序版本、操作系统和应用程序的信息。例如，“ODBC 驱动程序版本 1.2.3、Windows 10（x64）、Power BI”。
- 日志文件 -（必填）了解问题所需的最少 ODBC 驱动程序日志文件数。有关 ODBC 2.x 驱动程序的日志记录选项的更多信息，请参阅 日志记录选项。
- 连接字符串 -（必填）您的 ODBC 连接字符串或显示所用连接参数的对话框的屏幕截图。有关连接参数的信息，请参阅 Athena ODBC 2.x 连接参数。
- 问题步骤 -（可选）如果可能，包含可以帮助重现问题的步骤或独立程序。
- 查询错误信息 -（可选）如果您遇到涉及 DML 或 DDL 查询的错误，包含以下信息：

  - 失败的 DML 或 DDL 查询的完整版或简化版。
  - 所用的账户 ID 和 AWS 区域，以及查询执行 ID。
- SAML 错误 -（可选）如果您遇到与使用 SAML 断言进行身份验证有关的问题，包含以下信息：
  - 所用的身份提供商和身份验证插件。
  - SAML 令牌示例。

Amazon Athena ODBC 2.x 发布说明

这些发布说明详细介绍了 Amazon Athena ODBC 2.x 驱动程序中的增强功能、特征、已知问题和工作流变更。

2.0.3.0

发布时间：2024 年 4 月 8 日

Amazon Athena ODBC 版本 2.0.3.0 驱动程序包含以下改进和修复。

改进

- Linux 和 Mac 平台上添加了对 Okta 身份验证插件的 MFA 支持。
- 适用于 Windows 的 `athena-odbc.dll` 库和 `AmazonAthenaODBC-2.x.x.x.msi` 安装程序现在均已签名。
- 更新了随驱动程序一起安装的 CA 证书 `cacert.pem` 文件。
- 缩短了列出 Lambda 目录下的表所需的时间。对于 `LAMBDA` 目录类型，ODBC 驱动程序现在可以提交 SHOW TABLES 查询来获取可用表的列表。有关更多信息，请参阅 使用查询来列出表。
- 引入了 `UseWCharForStringTypes` 连接参数，以便使用 `SQL_WCHAR` 和 `SQL_WVARCHAR` 来报告字符串数据类型。有关更多信息，请参阅 使用 WCHAR 作为字符串类型。

修复

- 修复了使用 Get-OdbcDsn PowerShell 工具时出现的注册表损坏警告。
- 更新了解析逻辑，以便处理查询字符串开头的注释。
- 日期和时间戳数据类型现在允许年份字段为零。

要下载新的 ODBC 版本 2 驱动程序，请参阅 ODBC 2.x 驱动程序下载。有关连接信息，请参阅 Amazon Athena ODBC 2.x。

2.0.2.2

发布时间：2024 年 2 月 13 日

Amazon Athena ODBC 版本 2.0.2.2 驱动程序包含以下改进和修复。

改进

- 添加了两个连接参数 `StringColumnLength` 和 `ComplexTypeColumnLength`，可用于更改字符串和复杂数据类型的默认列长度。有关更多信息，请参阅 字符串列长度和 复杂类型列长度。
- 添加了对 Linux 和 macOS（英特尔和 ARM）操作系统的支持。有关更多信息，请参阅 Linux和 macOS。
- AWS-SDK-CPP 已更新到 1.11.245 标签版本。
- curl 库已更新到 8.6.0 版本。

修复

- 解决了导致在精度列中类似字符串数据类型的结果集元数据中报告错误值的问题。

要下载 ODBC 版本 2 驱动程序，请参阅 ODBC 2.x 驱动程序下载。有关连接信息，请参阅 Amazon Athena ODBC 2.x。

2.0.2.1

发布时间：2023 年 12 月 7 日

Amazon Athena ODBC 版本 2.0.2.1 驱动程序包含以下改进和修复。

改进

- 改进了所有接口的 ODBC 驱动程序线程安全性。
- 启用日志记录后，现在会以毫秒精度记录日期时间值。
- 在使用 浏览器 SSO OIDC 插件进行身份验证期间，终端现在会打开，向用户显示设备代码。

修复

- 解决了在解析流式传输 API 的结果时出现的内存释放问题。
- 针对接口 `SQLTablePrivileges()`、`SQLSpecialColumns()` `SQLProcedureColumns()` 和 `SQLProcedures()` 的请求现在会返回空结果集。

要下载 ODBC 版本 2 驱动程序，请参阅 ODBC 2.x 驱动程序下载。有关连接信息，请参阅 Amazon Athena ODBC 2.x。

2.0.2.0

发布时间：2023 年 10 月 17 日

Amazon Athena ODBC 版本 2.0.2.0 驱动程序包含以下改进和修复。

改进

- 为浏览器 Azure AD、浏览器 SSO OIDC 和 Okta 基于浏览器的身份验证插件添加了文件缓存功能。

  像 Power BI 这样的 BI 工具和基于浏览器的插件会使用多个浏览器窗口。新的文件缓存连接参数允许在 BI 应用程序打开的多个进程之间缓存和重复使用临时凭证。
- 现在，应用程序可以在准备好语句后查询有关结果集的信息。
- 增加了默认连接和请求超时，以便在速度较慢的客户端网络中使用。有关更多信息，请参阅 连接超时和 请求超时。
- 为 SSO 和 SSO OIDC 添加了端点覆盖。有关更多信息，请参阅 端点覆盖。
- 添加了一个连接参数，可将身份验证请求的 URI 参数传递给 Ping。您可以使用此参数来绕过 Lake Formation 的单个角色限制。有关更多信息，请参阅 Ping URI 参数。

修复

- 修复了使用基于行的绑定机制时出现的整数溢出问题。
- 从浏览器 SSO OIDC 身份验证插件所需的连接参数列表中删除了超时。
- 为 `SQLStatistics()`、`SQLPrimaryKeys()`、`SQLForeignKeys()` 和 `SQLColumnPrivileges()` 添加了缺少的接口，并添加了根据请求返回空结果集的功能。

要下载新的 ODBC 版本 2 驱动程序，请参阅 ODBC 2.x 驱动程序下载。有关连接信息，请参阅 Amazon Athena ODBC 2.x。

2.0.1.1

发布时间：2023 年 8 月 10 日

Amazon Athena ODBC 版本 2.0.1.1 驱动程序包含以下改进和修复。

改进

- 在 Okta 身份验证插件中添加了 URI 日志记录。
- 向外部凭证提供程序插件添加了首选角色参数。
- 在 AWS 配置文件的配置文件名称中添加了对配置文件前缀的处理。

修复

- 更正了 Lake Formation 和 AWS STS 客户端一起使用时出现的 AWS 区域 使用问题。
- 已将缺少的分区键还原到表列列表中。
- 在 AWS 配置文件中添加了缺少的 `BrowserSSOOIDC` 身份验证类型。

要下载新的 ODBC 版本 2 驱动程序，请参阅 ODBC 2.x 驱动程序下载。

2.0.1.0

发布时间：2023 年 6 月 29 日

Amazon Athena 发布了 ODBC 版本 2.0.1.0 驱动程序。

Athena 发布了一个新的 ODBC 驱动程序，该驱动程序改善了连接、查询和可视化来自兼容的 SQL 开发和商业智能应用程序的数据的体验。最新版本的 Athena ODBC 驱动程序支持现有驱动程序的功能，并且易于升级。新版本支持通过 AWS IAM Identity Center 对用户进行身份验证。还提供了从 Amazon S3 读取查询结果的选项，这样就可以更快地获得查询结果。

有关更多信息，请参阅 Amazon Athena ODBC 2.x。

Athena ODBC 1.x 驱动程序

使用此页面上的链接下载 Amazon Athena 1.x ODBC 驱动程序许可协议、ODBC 驱动程序和 ODBC 文档。有关 ODBC 连接字符串的信息，请参阅《ODBC 驱动程序安装和配置指南》PDF 文件（可从此页面下载）。有关许可的更多信息，请参阅 通过 JDBC 和 ODBC 连接访问。

> **Important**
>
> 使用 ODBC 1.x 驱动程序时，请务必注意以下要求：
>
> - 保留端口 444 — 保留 Athena 用于流式传输查询结果的端口 444，对出站流量开放。当您使用 PrivateLink 终端节点连接到 Athena 时，请确保附加到 PrivateLink 终端节点的安全组在端口 444 上对入站流量开放。

- athena：GetQueryResultsStream 策略 — 将 `athena:GetQueryResultsStream` 策略操作添加到使用 ODBC 驱动程序的 IAM 主体。此策略操作并不通过 API 直接公开。它仅作为流式传输结果的一部分与 ODBC 和 JDBC 驱动程序配合使用。有关策略示例，请参阅AWS 托管策略：AWSQuicksightAthenaAccess。

Windows

驱动程序版本	下载链接
适用于 Windows 32 位的 ODBC 1.2.3.1000	Windows 32 位 ODBC 驱动程序 1.2.3.1000
适用于 Windows 64 位的 ODBC 1.2.3.1000	Windows 64 位 ODBC 驱动程序 1.2.3.1000

Linux

驱动程序版本	下载链接
适用于 Linux 32 位的 ODBC 1.2.3.1000	Linux 32 位 ODBC 驱动程序 1.2.3.1000
适用于 Linux 64 位的 ODBC 1.2.3.1000	Linux 64 位 ODBC 驱动程序 1.2.3.1000

OSX

驱动程序版本	下载链接
适用于 OSX 的 ODBC 1.2.3.1000	OSX ODBC 驱动程序 1.2.3.1000

文档

内容	文档链接
Amazon Athena ODBC 驱动程序许可协议	许可协议

内容	文档链接
适用于 ODBC 1.2.3.1000 的文档	ODBC 驱动程序安装和配置指南版本 1.2.3.1000
ODBC 1.2.3.1000 发布说明	ODBC 驱动程序发布说明版本 1.2.3.1000

ODBC 驱动程序注释

不使用代理进行连接

如果要指定驱动程序不使用代理连接到的某些主机，则可以在 ODBC 连接字符串中使用可选 `NonProxyHost` 属性。

`NonProxyHost` 属性指定了启用代理连接时连接器可以无需通过代理服务器访问的主机列表（以逗号分隔），如下例所示：

```
.amazonaws.com,localhost,.example.net,.example.com
```

`NonProxyHost` 连接参数传递给 `CURLOPT_NOPROXY` curl 选项。有关 `CURLOPT_NOPROXY` 格式的信息，请参阅 curl 文档中的 CURLOPT_NOPROXY。

使用 ODBC 客户端为 Microsoft AD FS 用户配置对 Amazon Athena 的联合访问权限

要使用 ODBC 客户端为 Microsoft Active Directory 联合身份验证服务（AD FS）用户设置对 Amazon Athena 的联合访问权限，您首先要在 AD FS 和 AWS 账户之间建立信任。获得信任后，您的 AD 用户便可使用其 AD 凭证对 AWS 进行联合身份验证，并担任 AWS Identity and Access Management（IAM）角色权限以访问 Athena API 等 AWS 资源。

要创建这种信任，您需要将 AD FS 作为 SAML 提供商添加到 AWS 账户 中，并创建可供联合用户担任的 IAM 角色。在 AD FS 端，您可以将 AWS 添加为依赖方并编写 SAML 声明规则，以便将正确的用户属性发送到 AWS 进行授权（尤其是 Athena 和 Amazon S3）。

配置 AD FS 对 Athena 的访问权限涉及以下主要步骤：

1. 设置 IAM SAML 提供商和角色

2. 配置 AD FS

3. 创建 Active Directory 用户和组

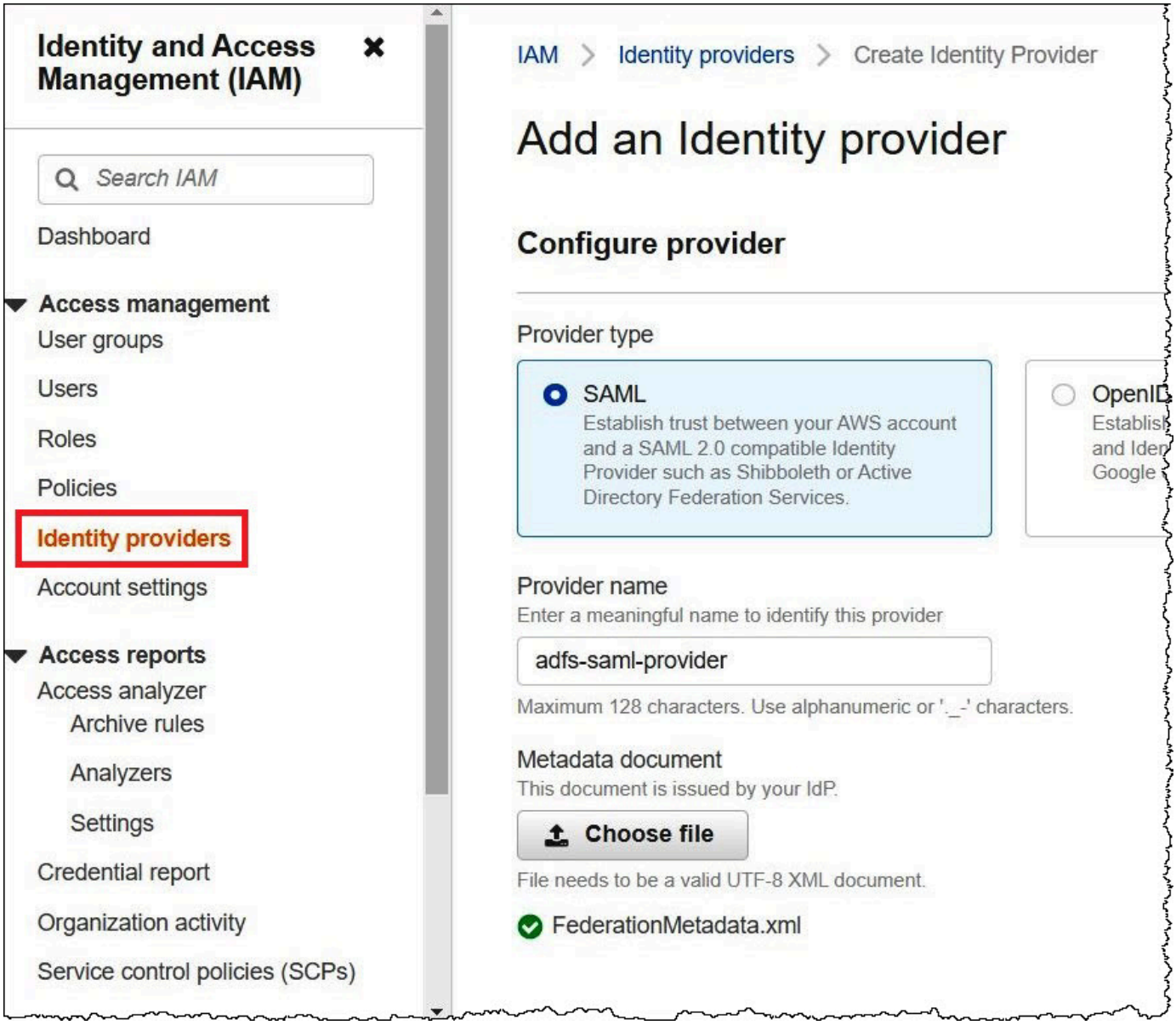
4. 配置 AD FS ODBC 与 Athena 的连接

1. 设置 IAM SAML 提供商和角色

在本节中，您需要将 AD FS 作为 SAML 提供商添加到 AWS 账户中，并创建可供联合用户担任的 IAM 角色。

设置 SAML 提供商

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Identity providers（身份提供程序）。
3. 选择 Add provider（添加提供程序）。
4. 对于 Provider type（提供程序类型），选择 SAML。



5. 对于 Provider name（提供程序名称），输入 **adfs-saml-provider**。
6. 在浏览器中，输入以下地址以下载适用于 AD FS 服务器的联合身份验证 XML 文件。要执行此步骤，您的浏览器必须具有访问 AD FS 服务器的权限。

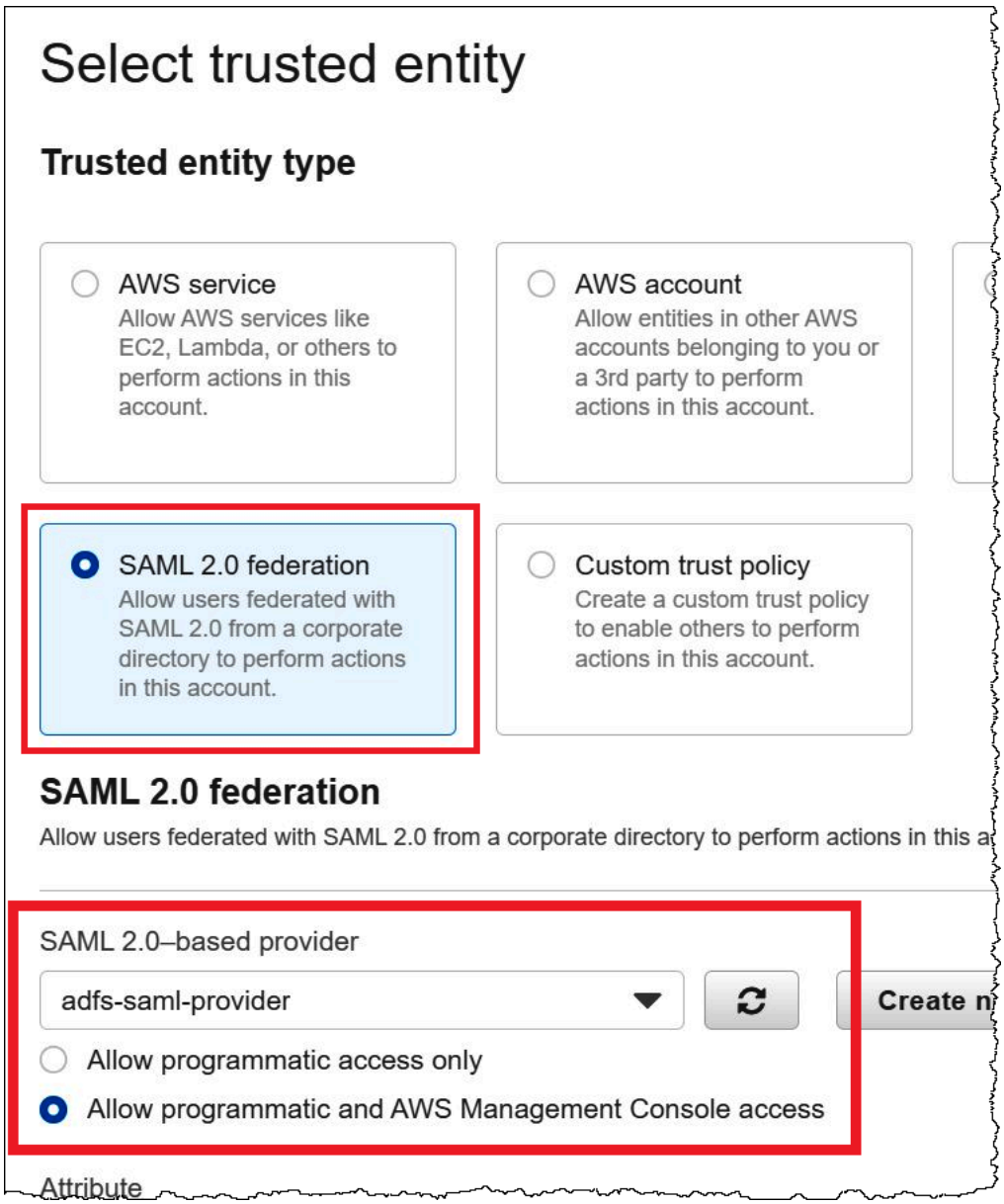
```
https://adfs-server-name/federationmetadata/2007-06/federationmetadata.xml
```

7. 在 IAM 控制台中，对于 Metadata document（元数据文档），选择 Choose file（选择文件），然后将联合身份验证元数据文件上传到 AWS。
8. 要完成操作，请选择 Add provider（添加提供商）。

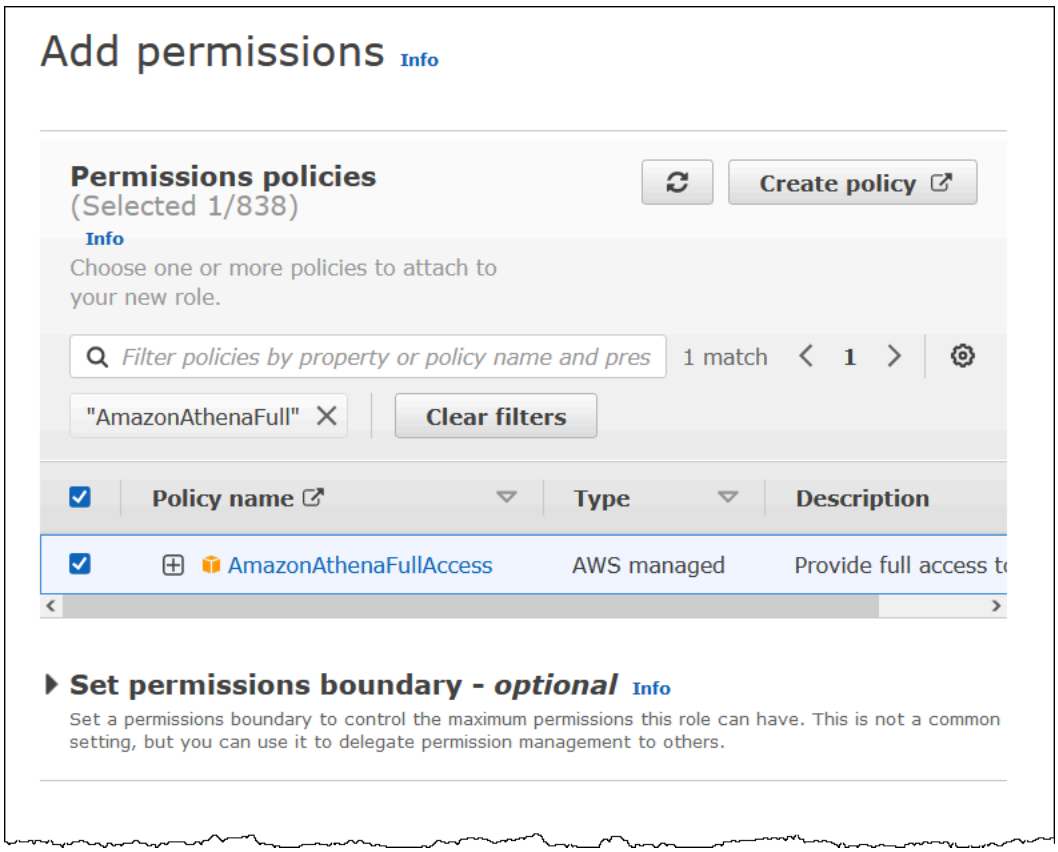
接下来，创建联合用户可担任的 IAM 角色。

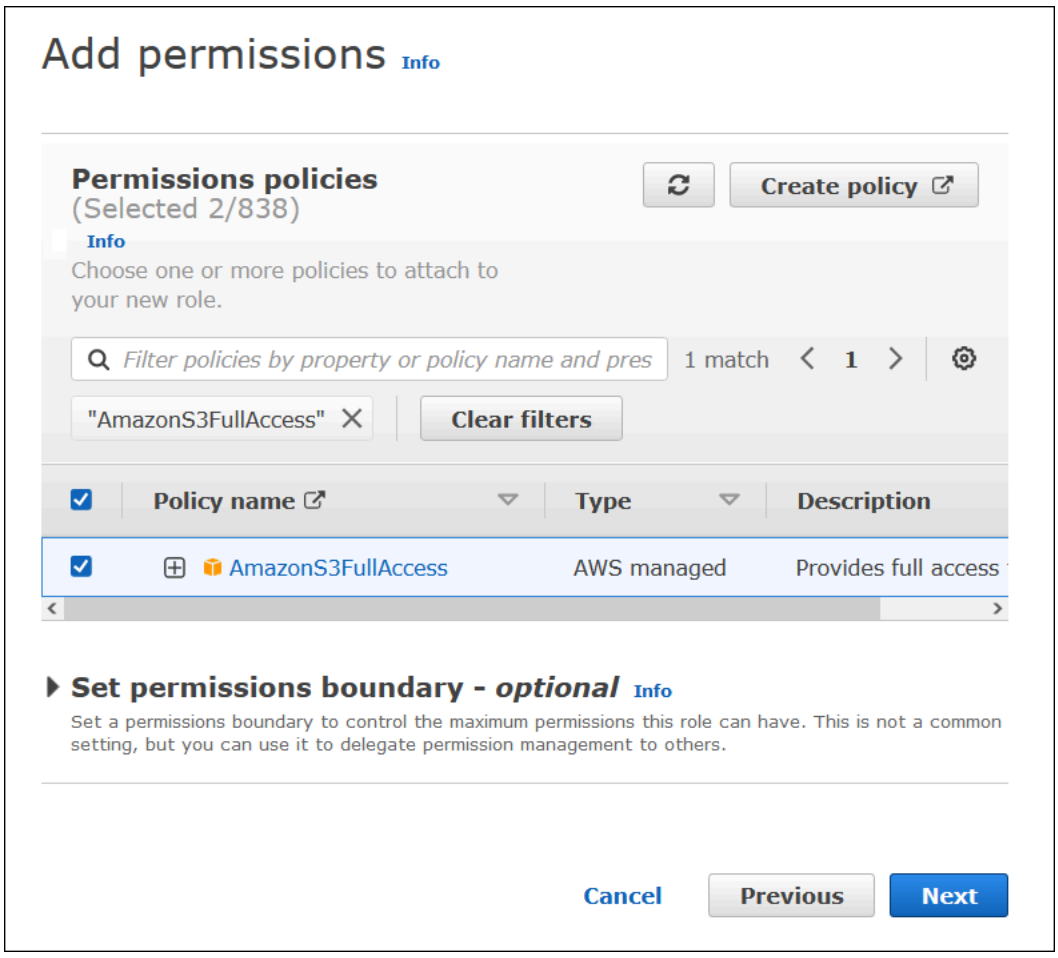
为联合用户创建 IAM 角色

1. 请在 IAM 控制台的导航窗格中，选择 Roles（角色）。
2. 选择创建角色。
3. 对于 Trusted entity type（受信任的实体类型），选择 SAML 2.0 Federation（SAML 2.0 联合身份验证）。
4. 对于 SAML 2.0-based provider（基于 SAML 2.0 的提供商），选择您创建的 adfs-saml-provider 提供商。
5. 选择允许编程访问和 AWS 管理控制台访问，然后选择下一步。



6. 在 Add permissions（添加权限）页面上，筛选此角色所需的 IAM 权限策略，然后选中相应的复选框。本教程附加了 `AmazonAthenaFullAccess` 和 `AmazonS3FullAccess` 策略。


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.

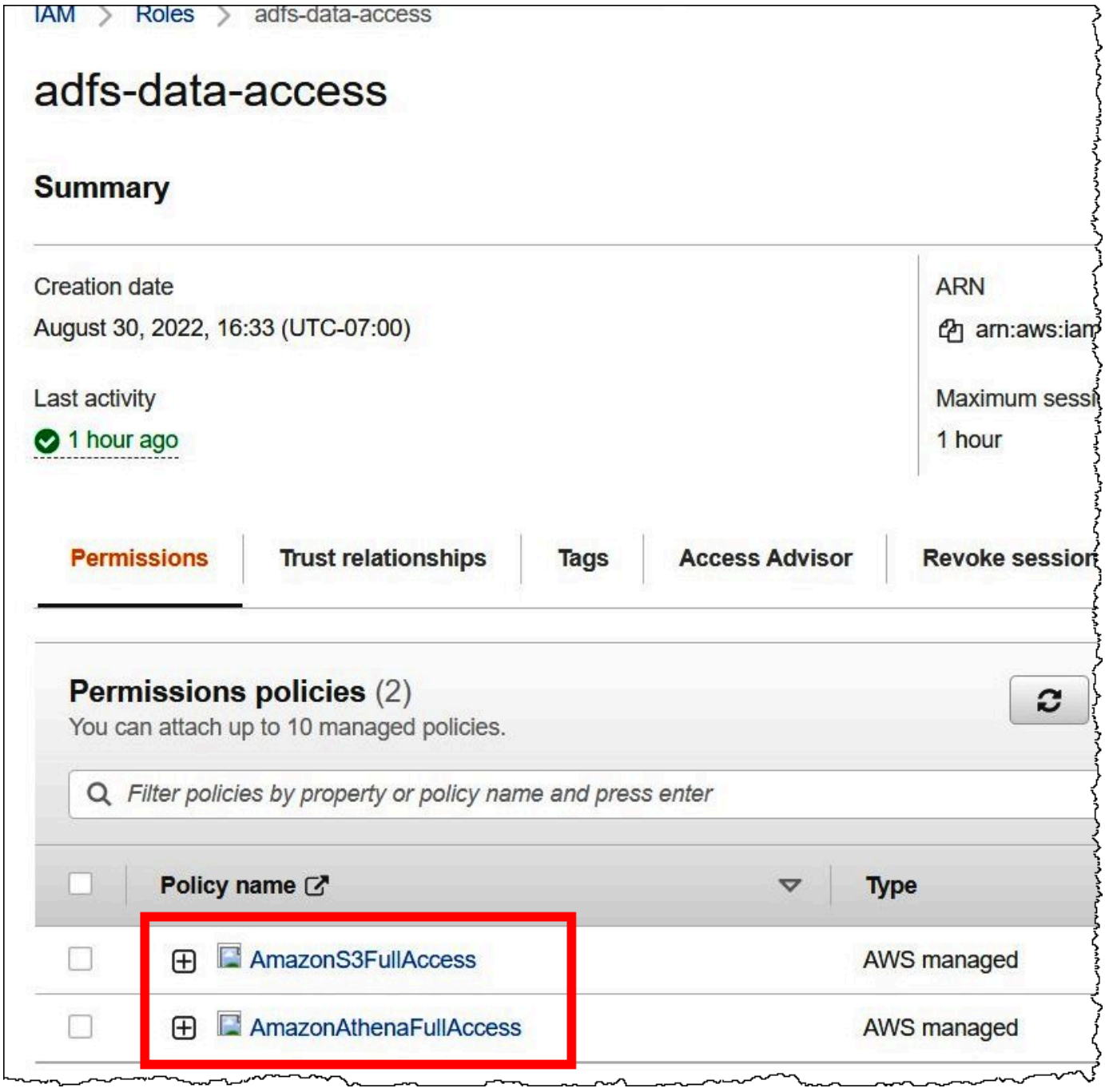


7. 选择下一步。
8. 在 Name, review, and create（名称、审核和创建）页面中，在 Role name（角色名称）中输入角色的名称。本教程使用的名称为 adfs-data-access。

   在 Step 1: Select trusted entities（步骤 1：选择受信任实体）中，Principal（主体）字段应自动填充为 `"Federated:" "arn:aws:iam::account_id:saml-provider/adfs-saml-provider"`。Condition 字段应包含 `"SAML:aud"` 和 `"https://signin.aws.amazon.com/saml"`。

Step 2: Add permissions（步骤 2：添加权限）将显示已附加到该角色的策略。

9. 选择创建角色。横幅消息会确认角色是否创建。
10. 在 Roles（角色）页面上，选择您刚刚创建的角色名称。该角色的摘要页面会显示已附加的策略。



2. 配置 AD FS

现在，您可以将 AWS 添加为依赖方并编写 SAML 声明规则，以便将正确的用户属性发送到 AWS 进行授权。

基于 SAML 的联合身份验证有两个参与方：IdP（Active Directory）和依赖方（AWS），后者是使用来自 IdP 的身份验证的服务或应用程序。

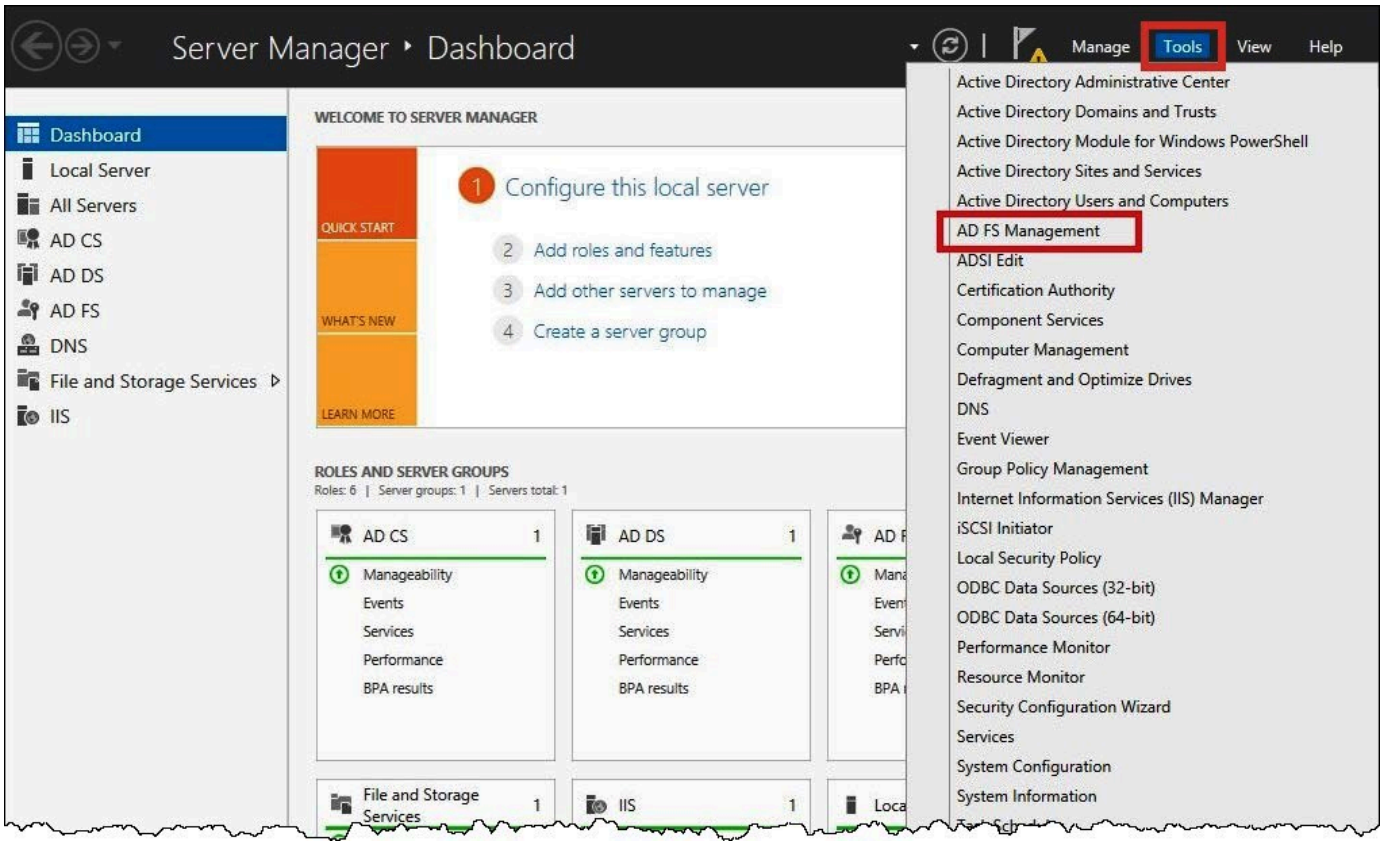
要配置 AD FS，请先添加依赖方信任，然后为依赖方配置 SAML 声明规则。AD FS 通过声明规则形成发送给依赖方的 SAML 断言。SAML 断言表明有关 AD 用户的信息是真实信息，并且已对用户进行身份验证。

添加依赖方信任

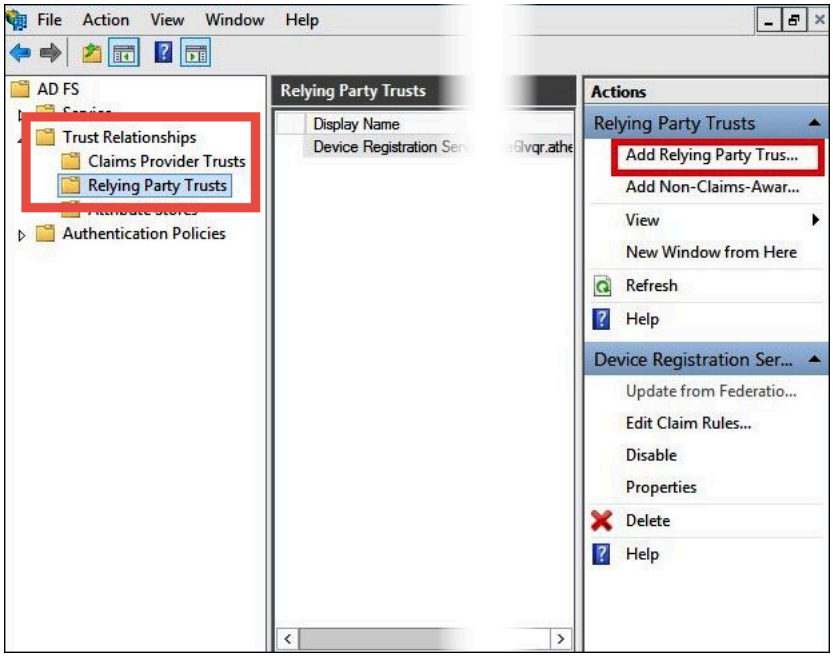
要在 AD FS 中添加依赖方信任，您可以使用 AD FS 服务器管理器。

在 AD FS 中添加依赖方信任

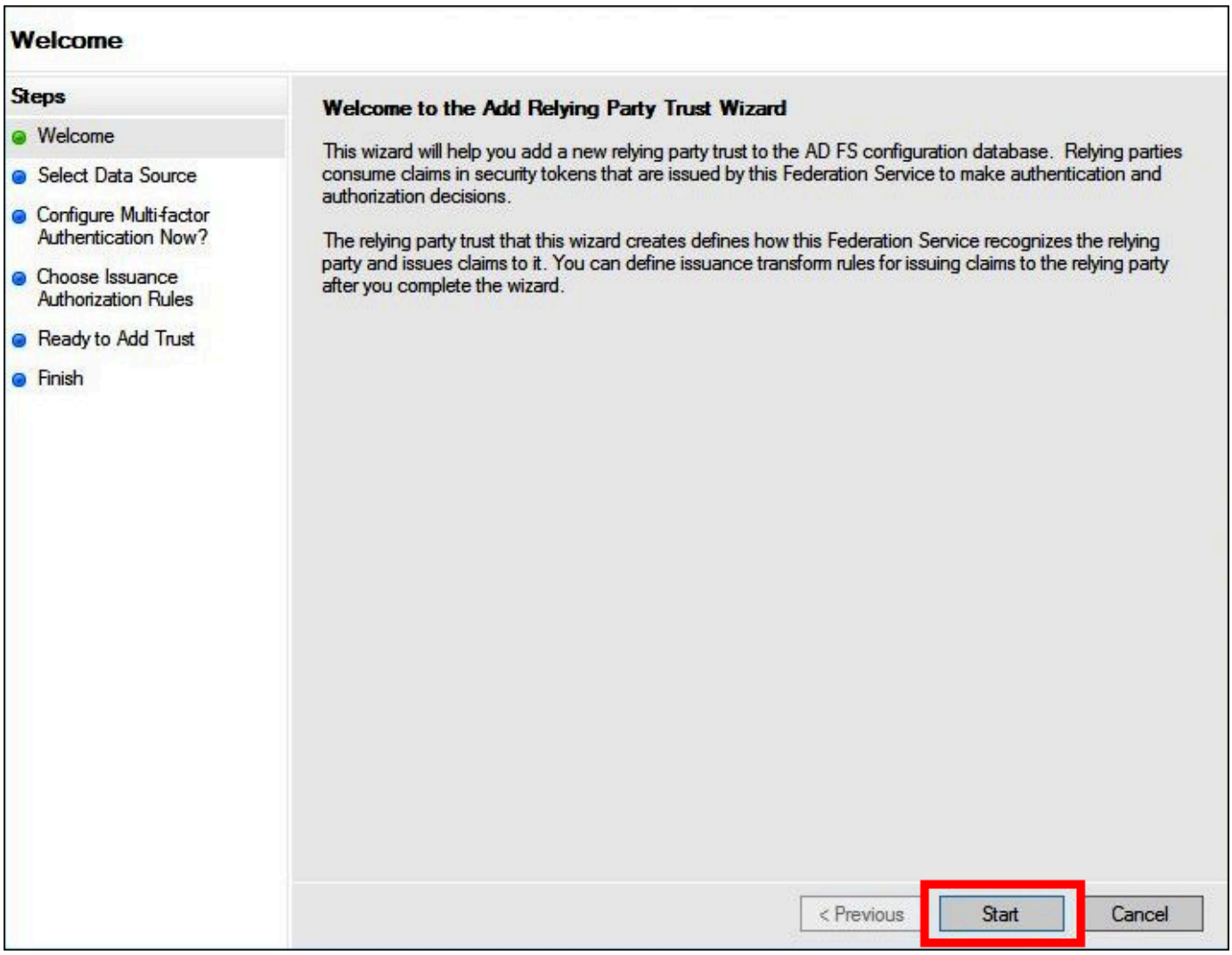
1. 登录 AD FS 服务器。
2. 在 Start（开始）菜单上，打开 Server Manager（服务器管理器）。
3. 选择 Tools（工具），然后选择 AD FS Management（AD FS 管理）。



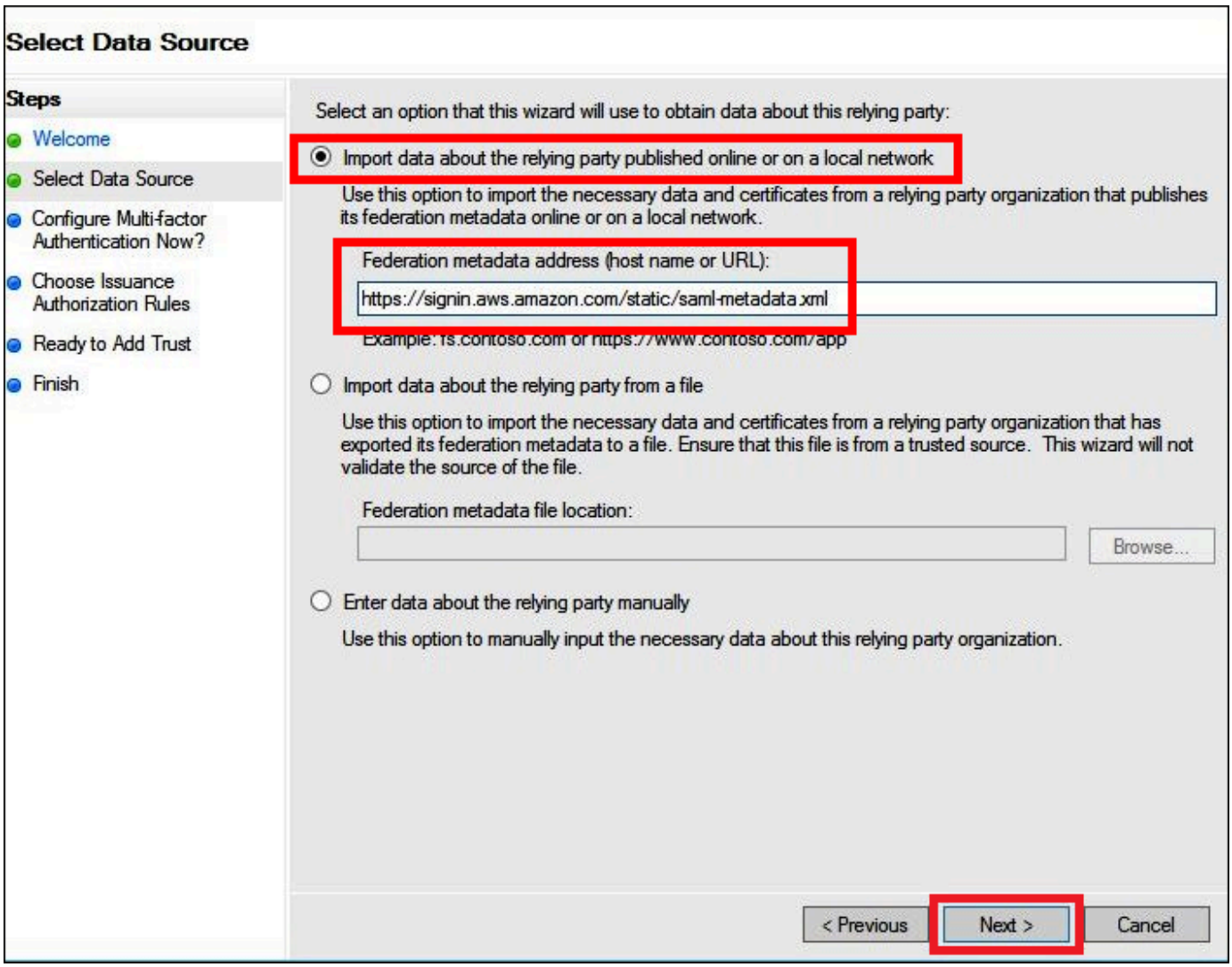
4. 在导航窗格中的 Trust Relationships（信任关系）下，选择 Relying Party Trusts（依赖方信任）。
5. 在 Actions（操作）下，选择 Add Relying Party Trust（添加依赖方信任）。



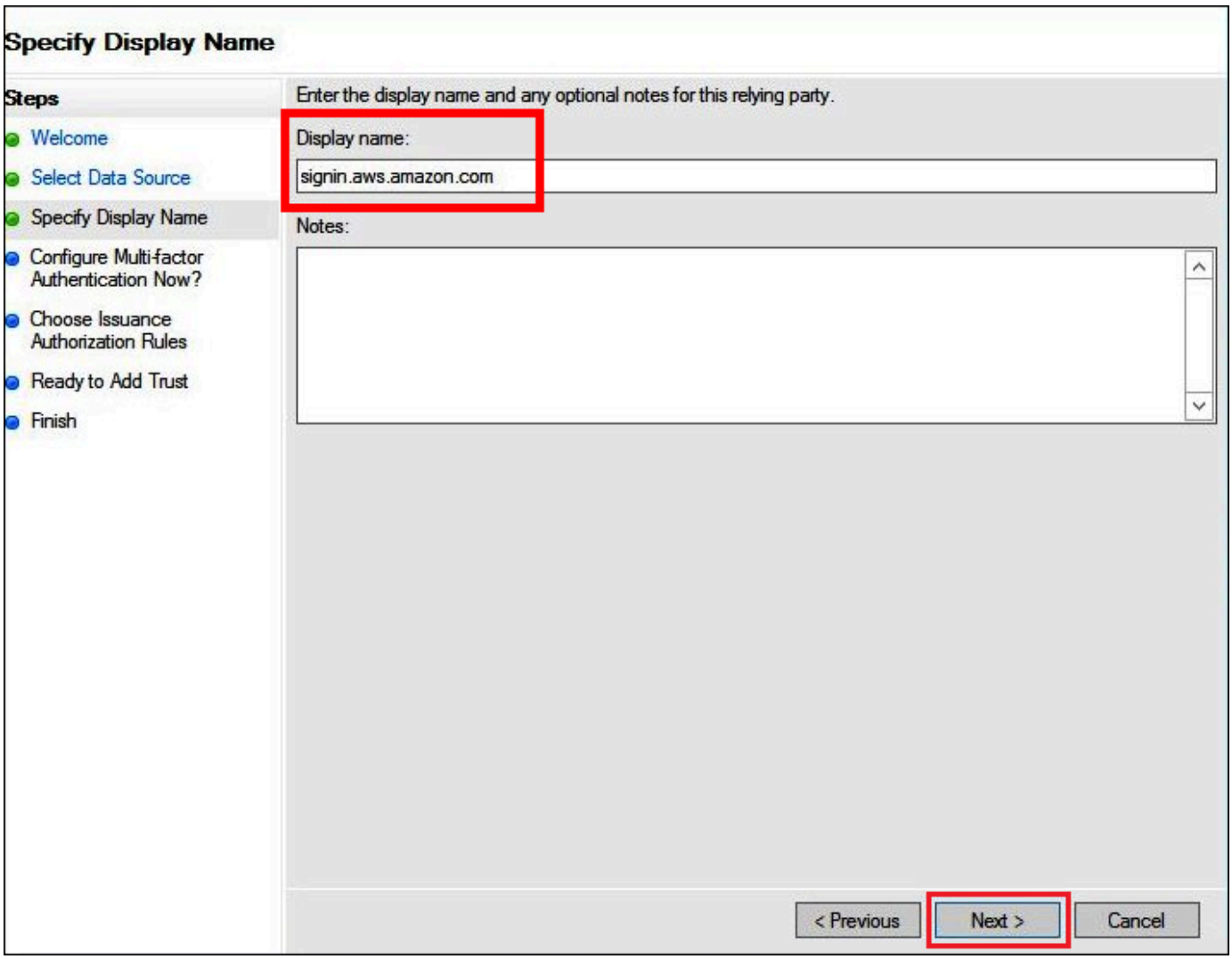
6. 在添加信赖方信任向导页面上，选择开始。



7. 在 Select Data Source（选择数据来源）屏幕上，选择 Import data about the relying party published online or on a local network（导入有关在线或在本地网络上发布的信赖方的数据）。
8. 对于 Federation metadata address (host name or URL) [联合身份验证元数据地址（主机名或 URL）]，请输入 URL **`https://signin.aws.amazon.com/static/saml-metadata.xml`**。
9. 选择下一步。

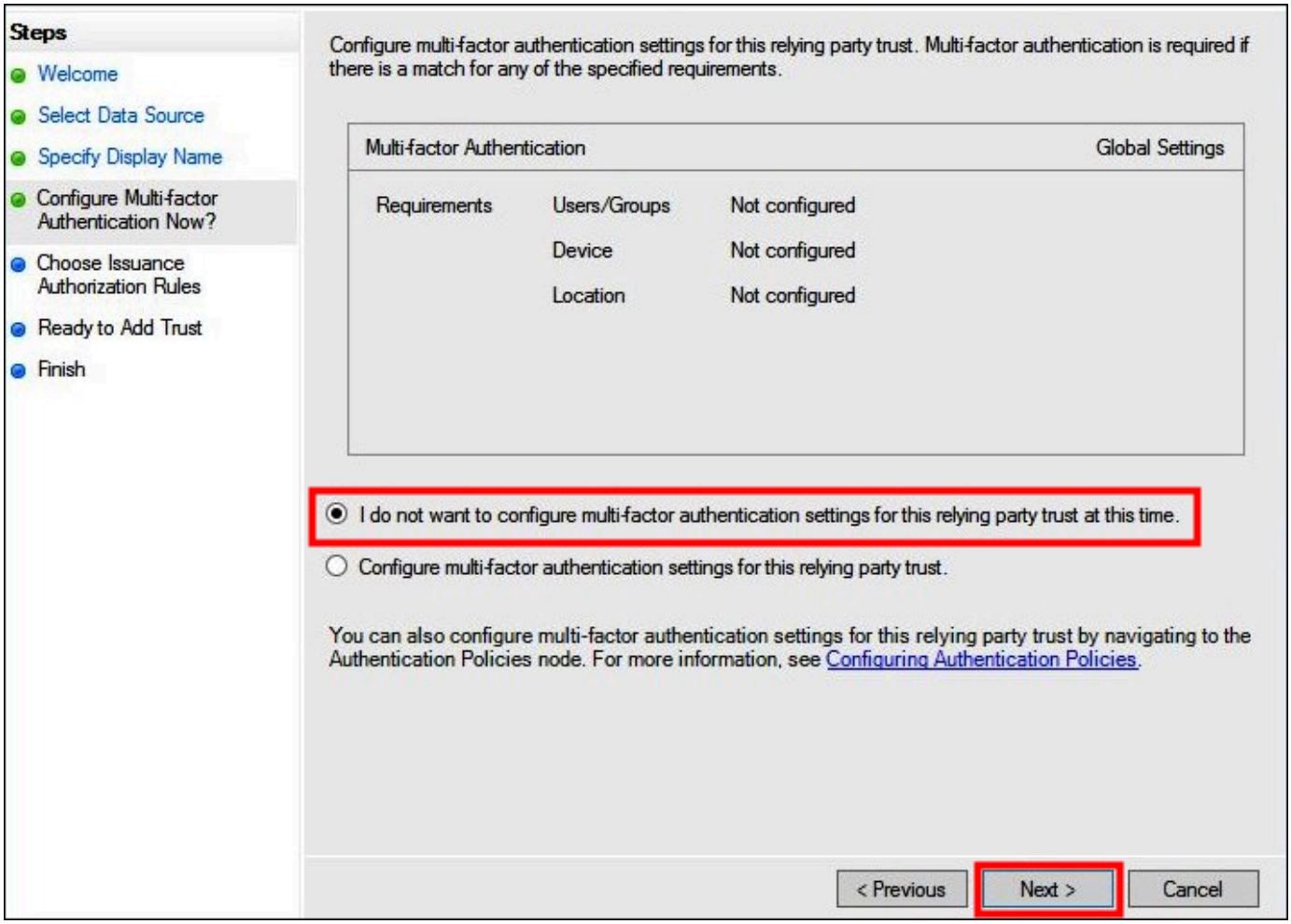


10. 在 Specify Display Name（指定显示名称）页面上，为 Display name（显示名称）输入依赖方的显示名称，然后选择 Next（下一步）。



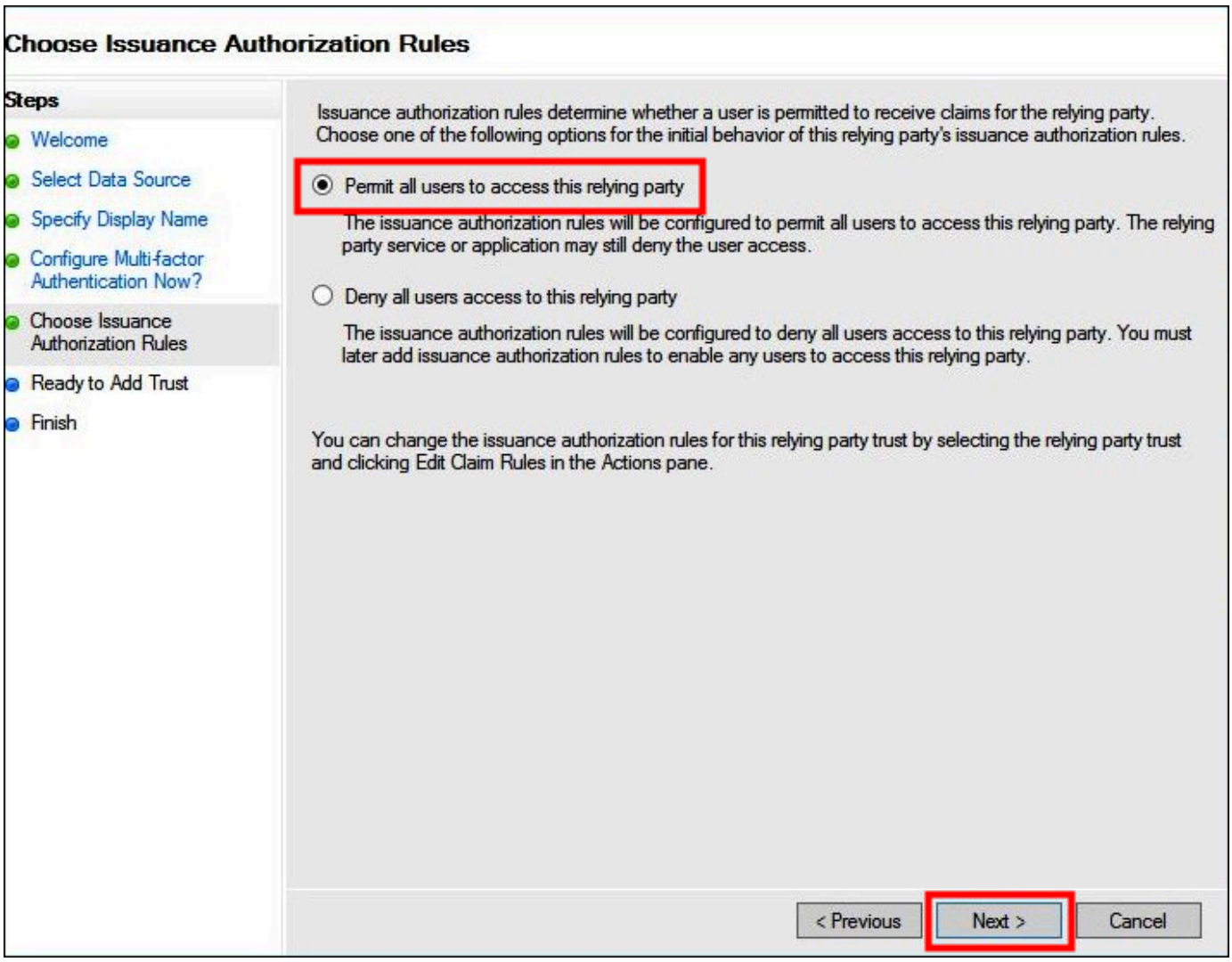
11. 在 Configure Multi-factor Authentication Now（立即配置多重身份验证）页面上，本教程选择 I do not want to configure multi-factor authentication for this relying party trust at this time（现在不想为此依赖方信任配置多重身份验证）。

    为增强安全性，我们建议您配置多重身份验证以帮助保护 AWS 资源。由于本教程使用示例数据集，因此不会启用多重身份验证。

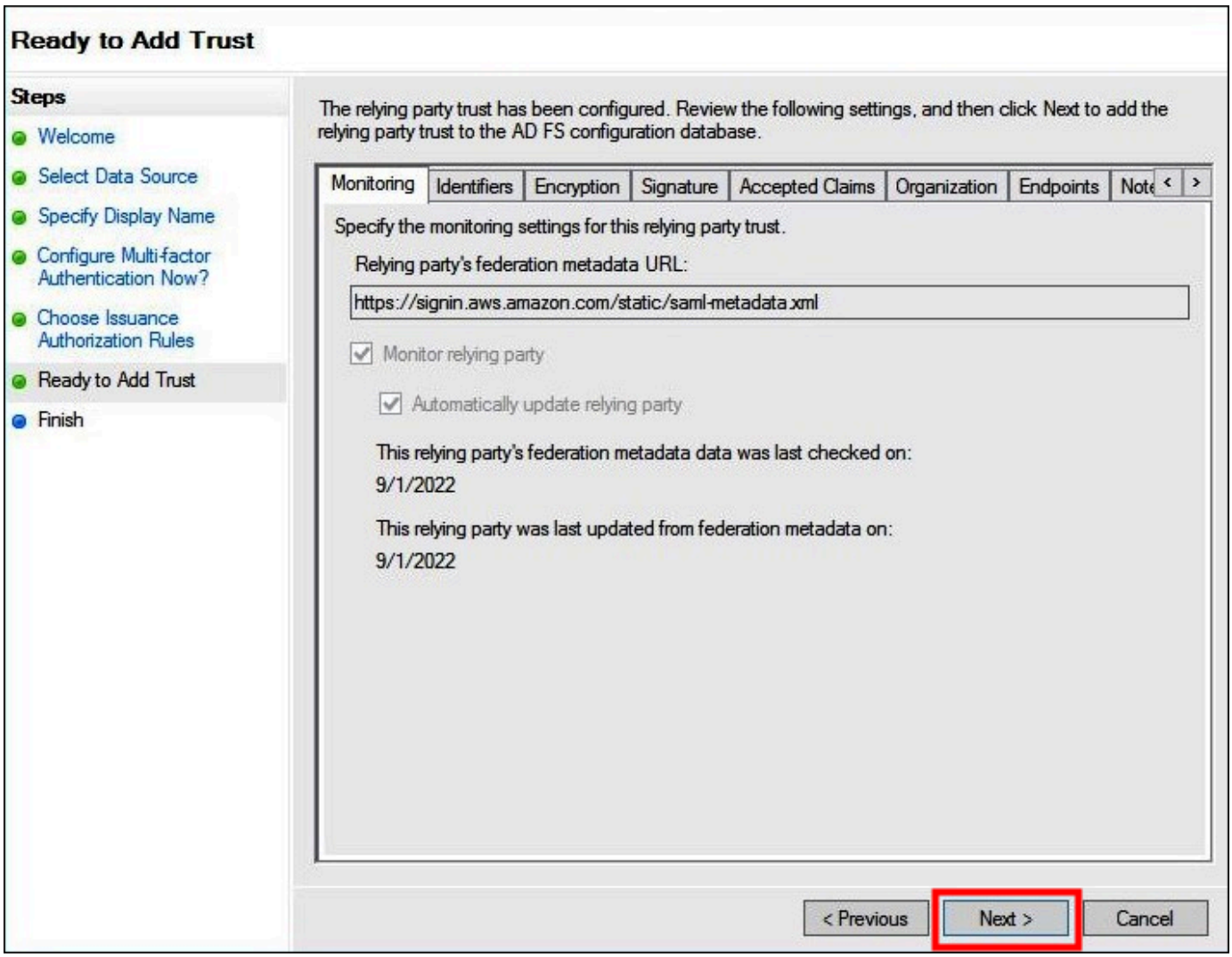


12. 选择下一步。
13. 在 Choose Issuance Authorization Rules（选择颁发授权规则）页面上，选择 Permit all users to access this relying party（允许所有用户访问此信赖方）。

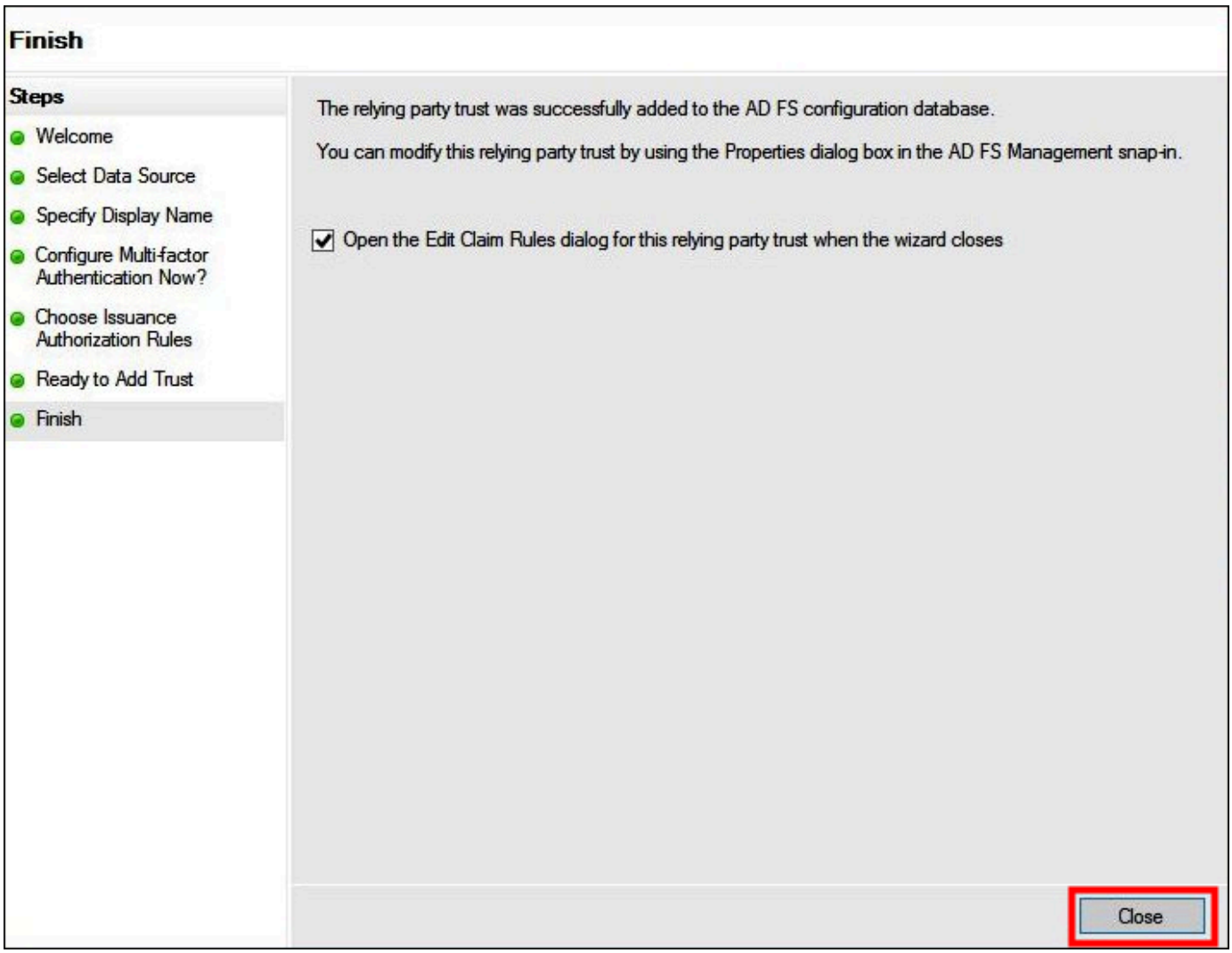
    此选项允许 Active Directory 中的所有用户将 AD FS 作为依赖方与 AWS 结合使用。您应考虑安全要求并对此配置进行相应调整。



14. 选择下一步。
15. 在 Ready to Add Trust（准备添加信任）页面上，选择 Next（下一步）将依赖方信任添加到 AD FS 配置数据库。



16. 在 Finish（完成）页面上，选择 Close（关闭）。



为依赖方配置 SAML 声明规则

在此任务中，您将创建两组声明规则。

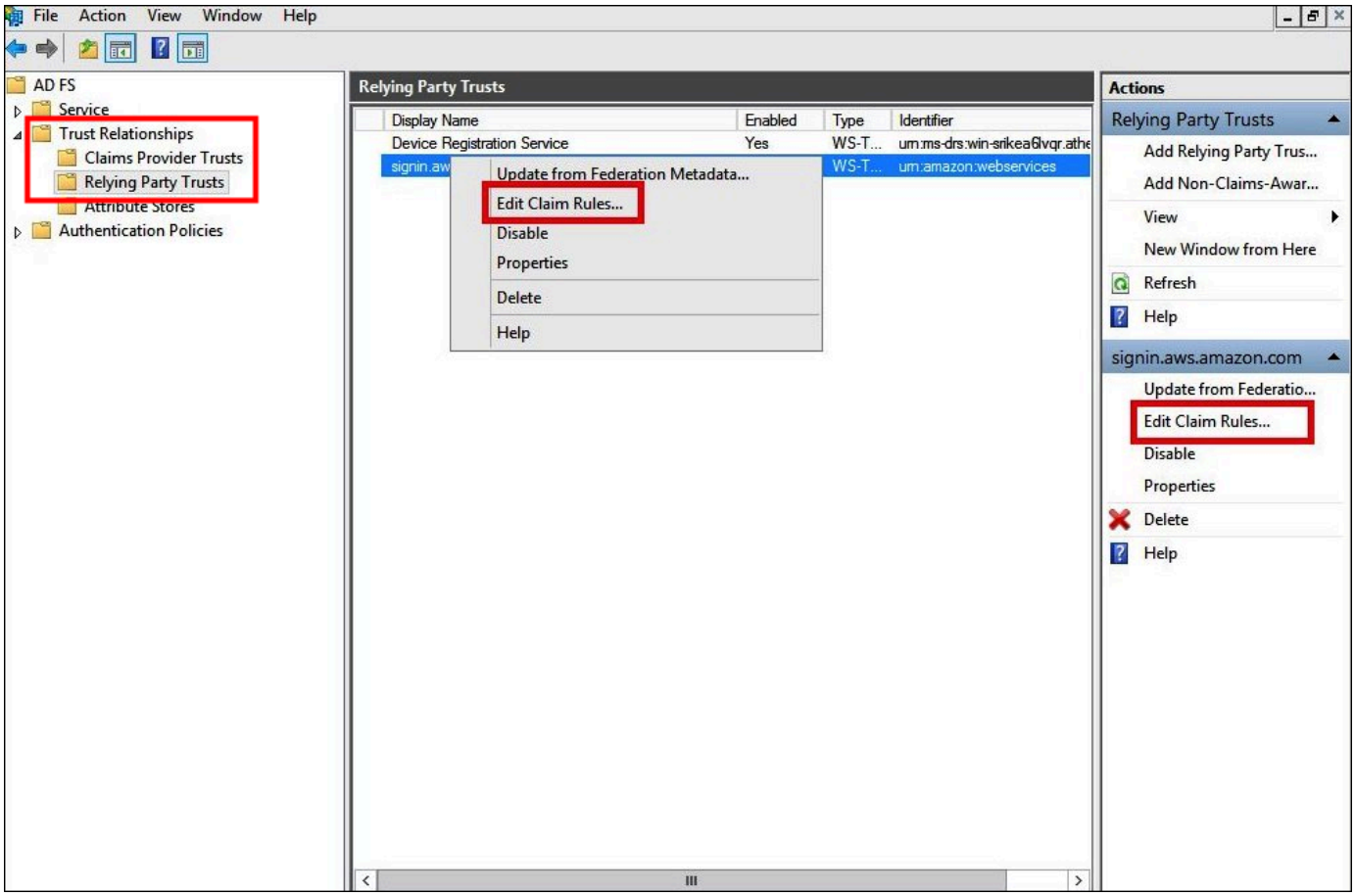
第一组规则 1-4 包含 AD FS 声明规则，这些规则需要基于 AD 组成员关系担任 IAM 角色。如果您想建立对 AWS Management Console 的联合访问权限，那么这些规则与您创建的规则相同。

第二组规则 5-6 是 Athena 访问控制所需的声明规则。

创建 AD FS 声明规则

1. 在“AD FS 管理”控制台导航窗格中，依次选择 Trust Relationships（信任关系）、Relying Party Trusts（依赖方信任）。
2. 查找您在上一节中创建的依赖方。

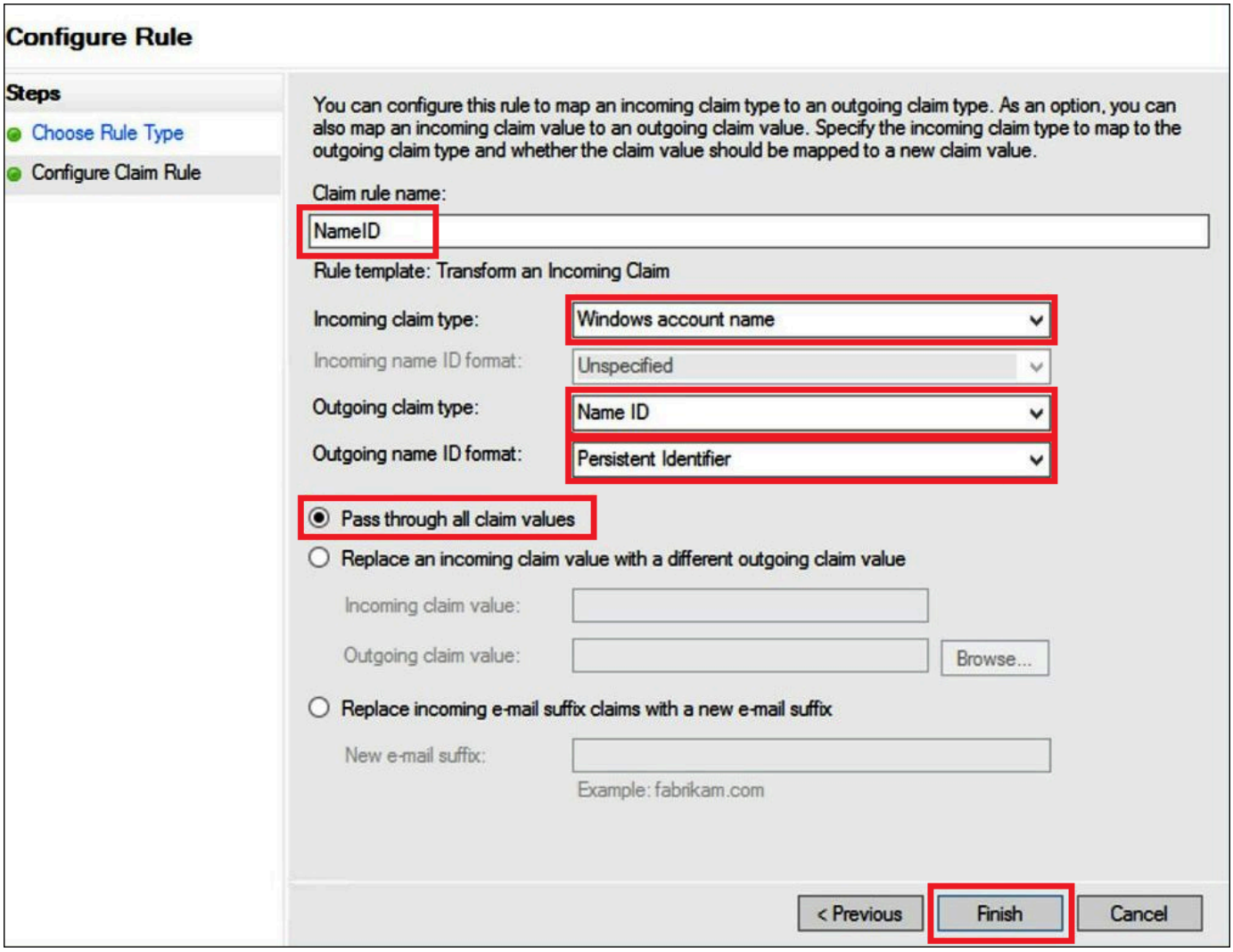
3. 右键单击依赖方并选择 Edit Claim Rules（编辑声明规则），或从 Actions（操作）菜单中选择 Edit Claim Rules（编辑声明规则）。



4. 选择添加规则。

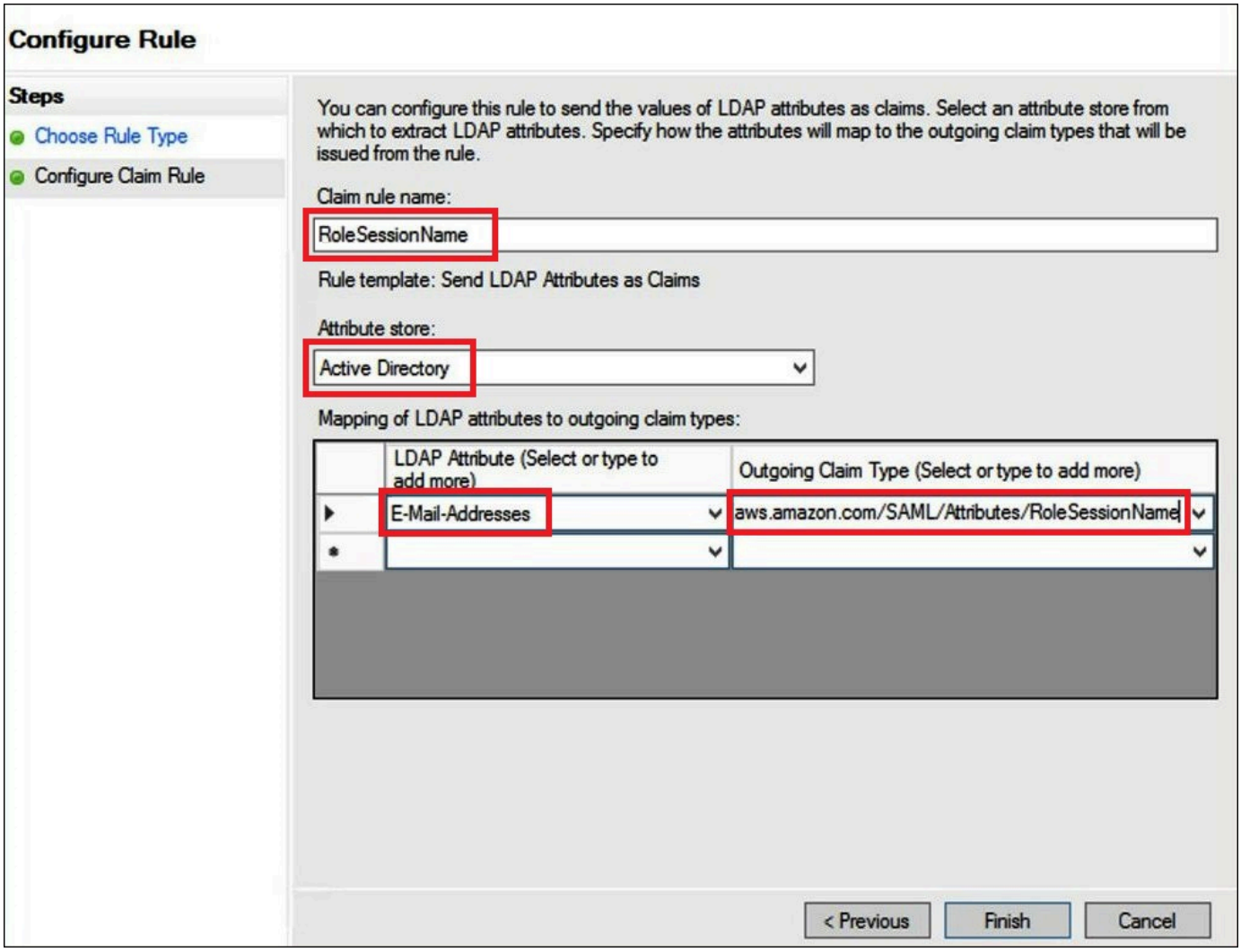
5. 在“添加转换声明规则向导”的 Configure Rule（配置规则）页面上，输入以下信息以创建声明规则 1，然后选择 Finish（完成）。

   - 对于 Claim Rule name（声明规则名称），输入 **`NameID`**。
   - 对于 Rule template（规则模板），使用 Transform an Incoming Claim（转换传入声明）。
   - 对于 Incoming claim type（传入声明类型），选择 Windows account name（Windows 账户名称）。
   - 对于 Outgoing claim type（传出声明类型），选择 Name ID（名称 ID）。
   - 对于传出名称 ID 格式，请选择持久性标识符。
   - 选择 Pass through all claim values（传递所有声明值）。



6. 选择 Add Rule（添加规则），然后输入以下信息以创建声明规则 2，然后选择 Finish（完成）。

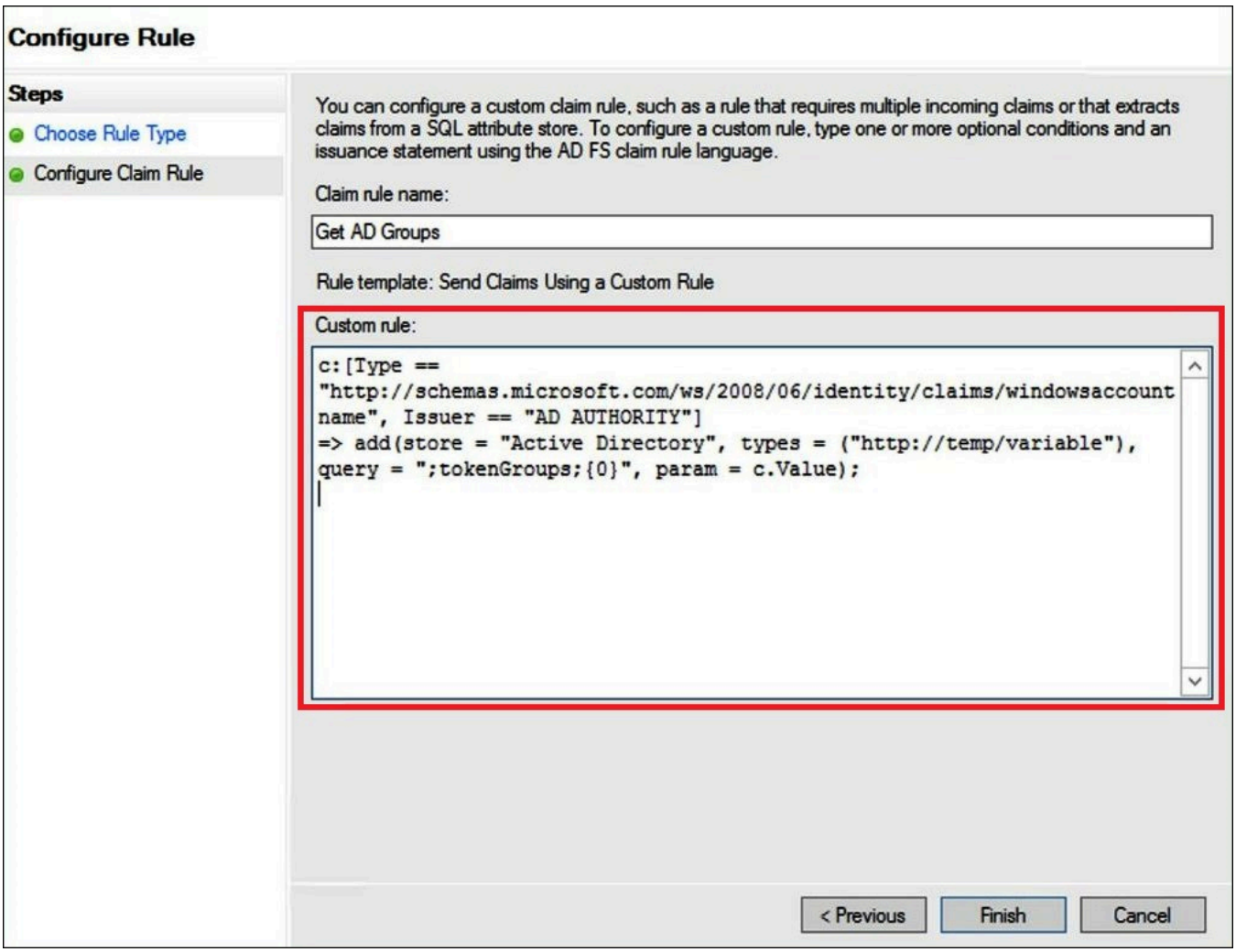
   - 对于 Claim Rule name（声明规则名称），输入 **`RoleSessionName`**。
   - 对于 Rule template（规则模板），使用 Send LDAP Attribute as Claims（将 LDAP 属性作为声明发送）。
   - 对于特性存储，请选择 Active Directory。
   - 对于 Mapping of LDAP attributes to outgoing claim types（将 LDAP 属性映射到传出声明类型），添加属性 **`E-Mail-Addresses`**。对于 Outgoing Claim Type（传出声明类型），输入 **`https://aws.amazon.com/SAML/Attributes/RoleSessionName`**。



7. 选择 Add Rule（添加规则），然后输入以下信息以创建声明规则 3，然后选择 Finish（完成）。

   - 对于 Claim Rule name（声明规则名称），输入 **`Get AD Groups`**。
   - 对于 Rule template（规则模板），使用 Send Claims Using a Custom Rule（通过自定义规则发送声明）。
   - 对于 Custom rule（自定义规则），输入以下代码：

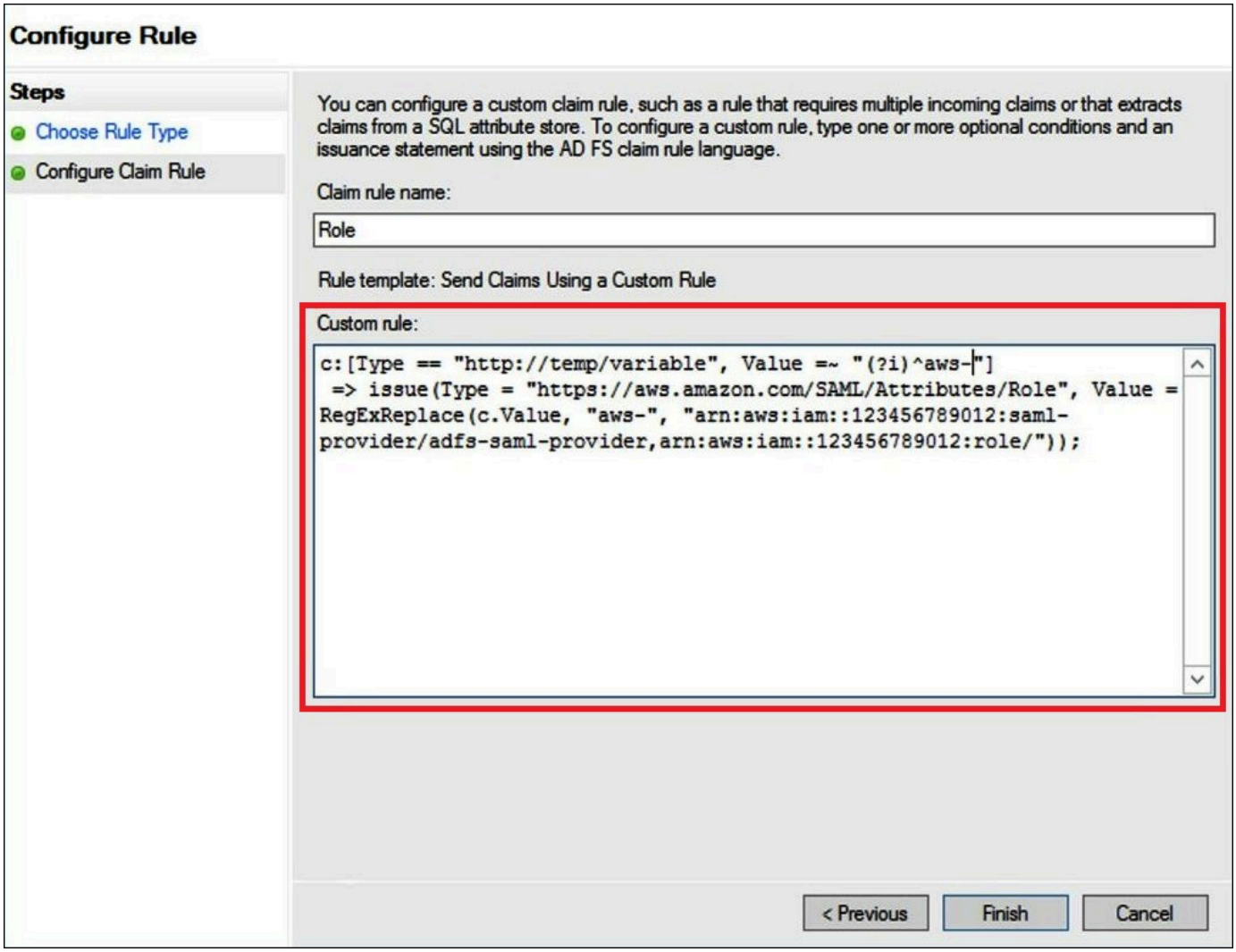
```
c:[Type == "http://schemas.microsoft.com/ws/2008/06/identity/claims/
windowsaccountname",
 Issuer == "AD AUTHORITY"]=> add(store = "Active Directory", types = ("http://
temp/variable"),
 query = ";tokenGroups;{0}", param = c.Value);
```



8. 选择添加规则。输入以下信息以创建声明规则 4，然后选择 Finish（完成）。

   - 对于 Claim Rule name（声明规则名称），输入 **`Role`**。
   - 对于 Rule template（规则模板），使用 Send Claims Using a Custom Rule（通过自定义规则发送声明）。
   - 对于 Custom rule（自定义规则），输入以下代码以及您之前创建的 SAML 提供商的账号和账户名称：

```
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/"));
```

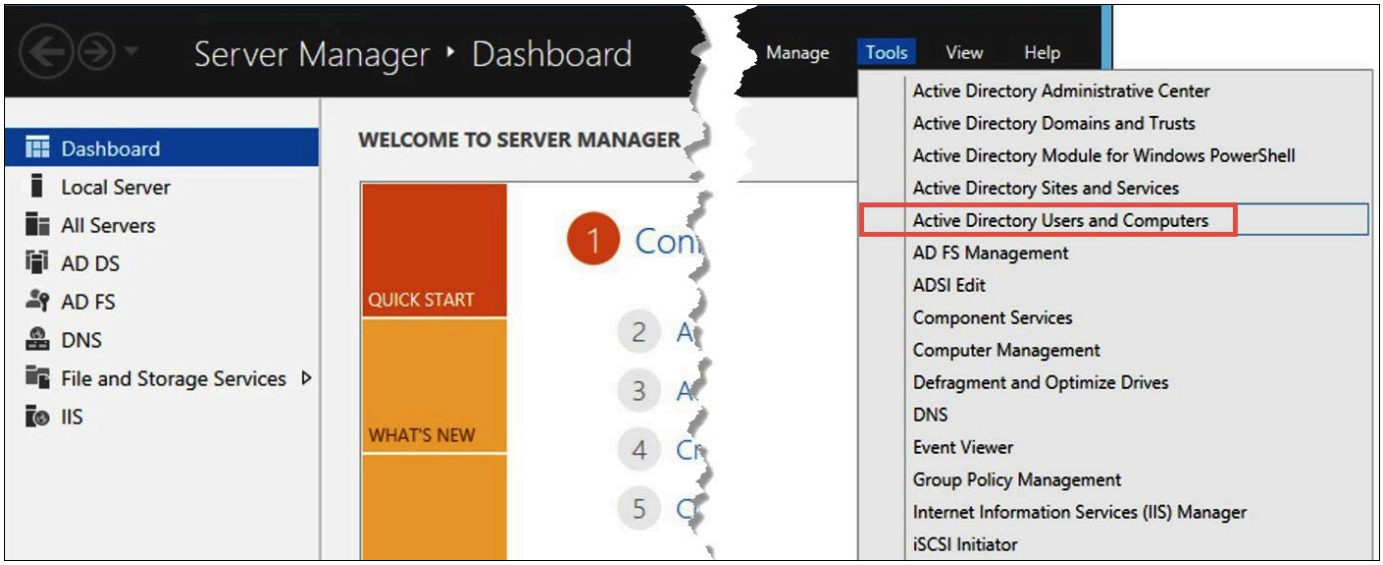


3. 创建 Active Directory 用户和组

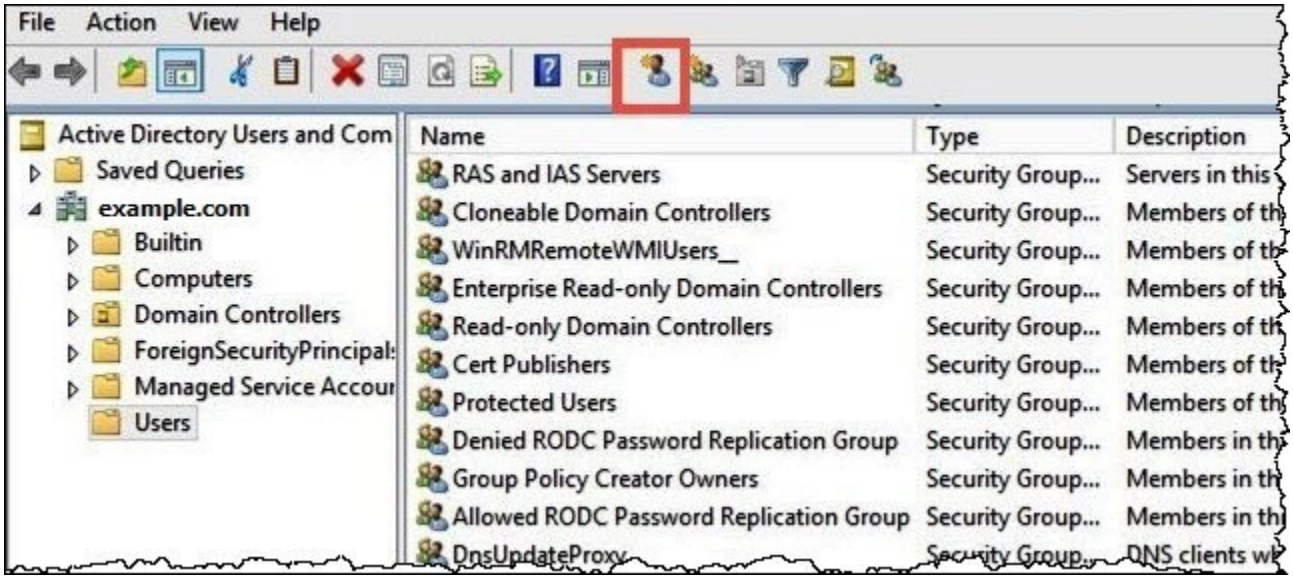
现在，您可以创建要访问 Athena 的 AD 用户以及置放这些用户的 AD 组，以便按组控制访问级别。创建对数据访问模式进行分类的 AD 组后，您可以将用户添加到这些组中。

创建要访问 Athena 的 AD 用户

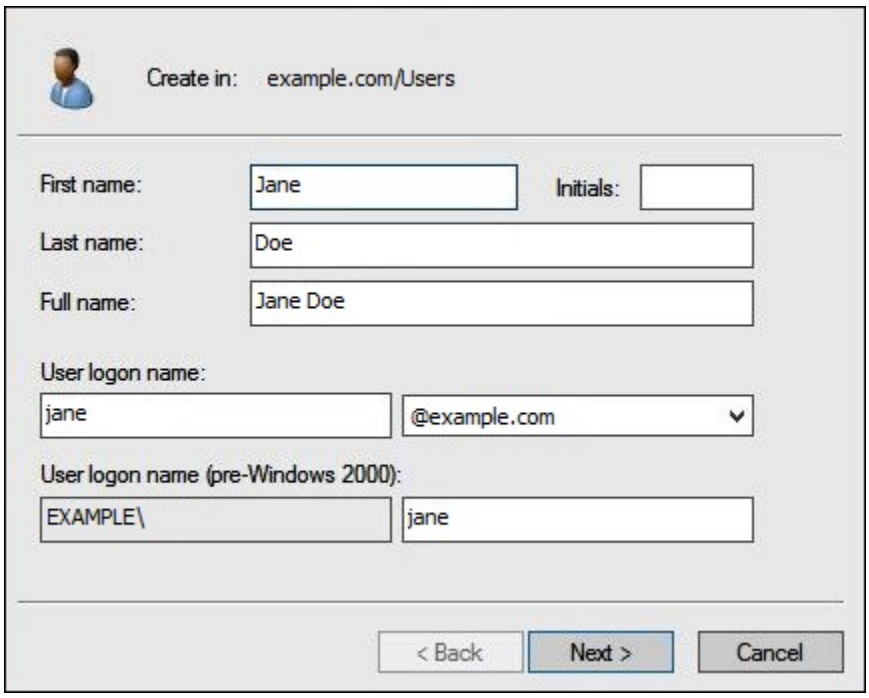
1. 在“服务器管理器”控制面板上，选择 Tools（工具），然后选择 Active Directory Users and Computers（Active Directory 用户和计算机）。



2. 在导航窗格中，选择用户。
3. 在 Active Directory Users and Computers（Active Directory 用户和计算机）工具栏上，选择 Create user（创建用户）选项。

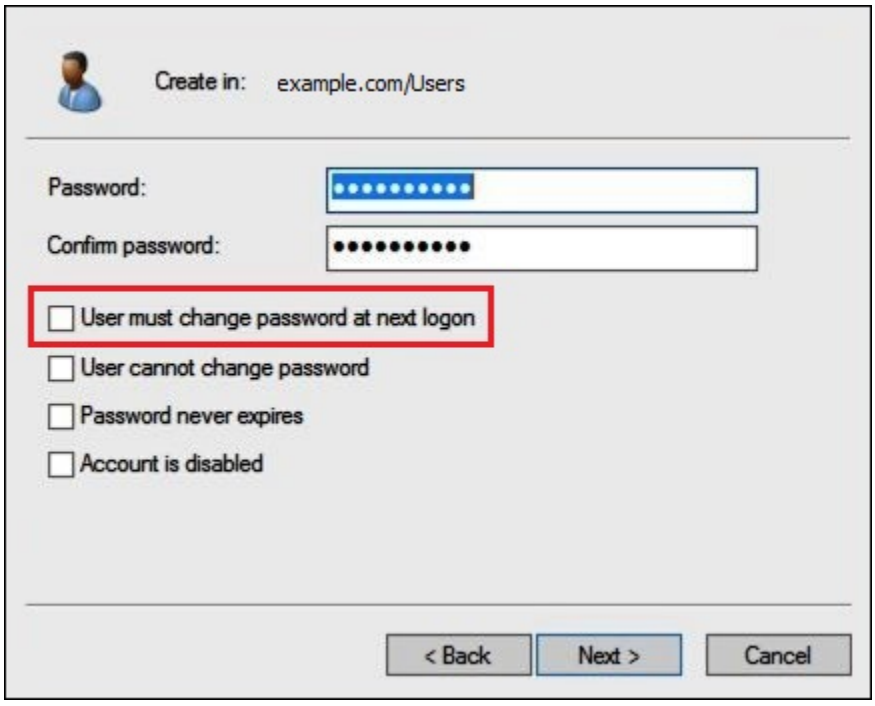


4. 在 New Object – User（新对象 – 用户）对话框中，对于 First name（名字）、Last name（姓氏）和 Full name（全名），输入名称。本教程使用的是 **Jane Doe**。



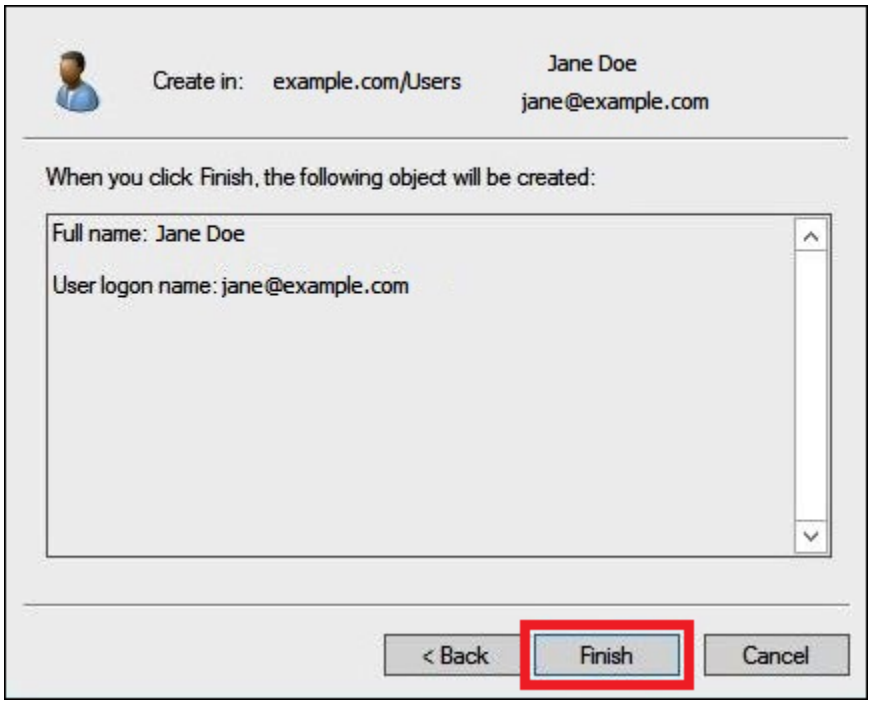
5. 选择下一步。
6. 对于 Password（密码），输入密码，然后重新键入以确认。

   为简单起见，本教程取消选择 User must change password at next sign on（用户下次登录时必须更改密码）。在真实场景中，您应要求新建用户更改其密码。

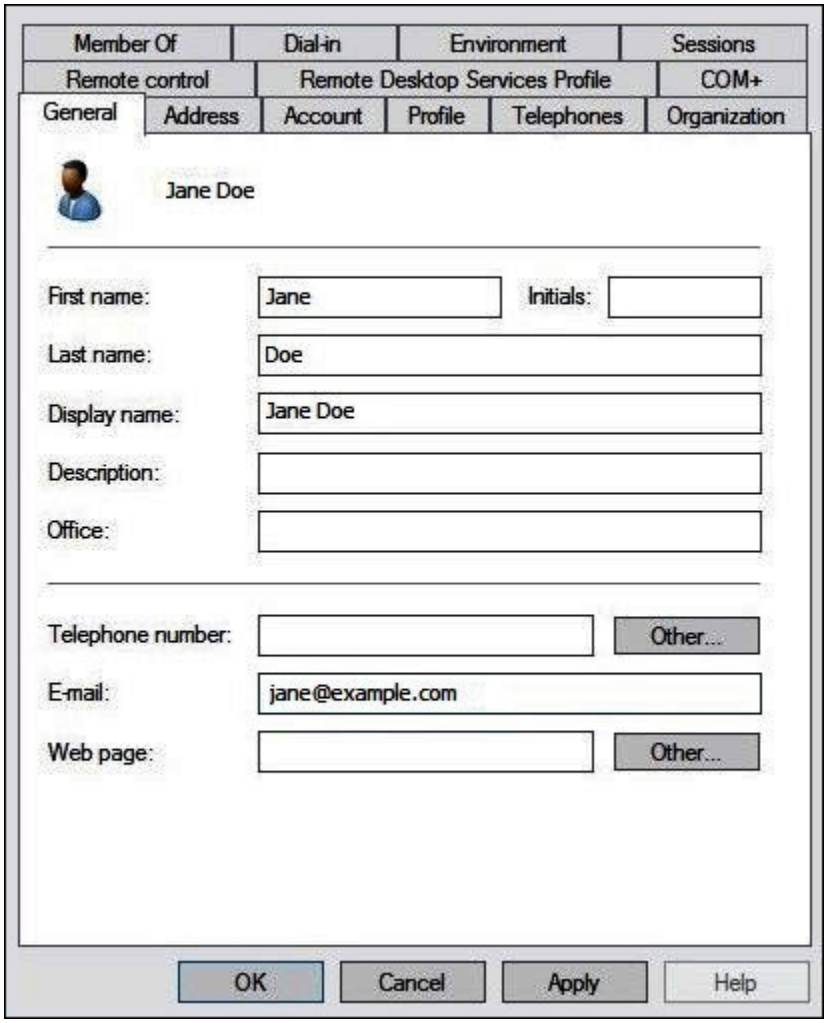


7. 选择下一步。

8. 选择完成。



9. 在 Active Directory Users and Computers（Active Directory 用户和计算机）中，选择用户名称。
10. 在用户的 Properties（属性）对话框中，对于 E-mail（电子邮件），输入电子邮件地址。本教程使用的是 **jane@example.com**。



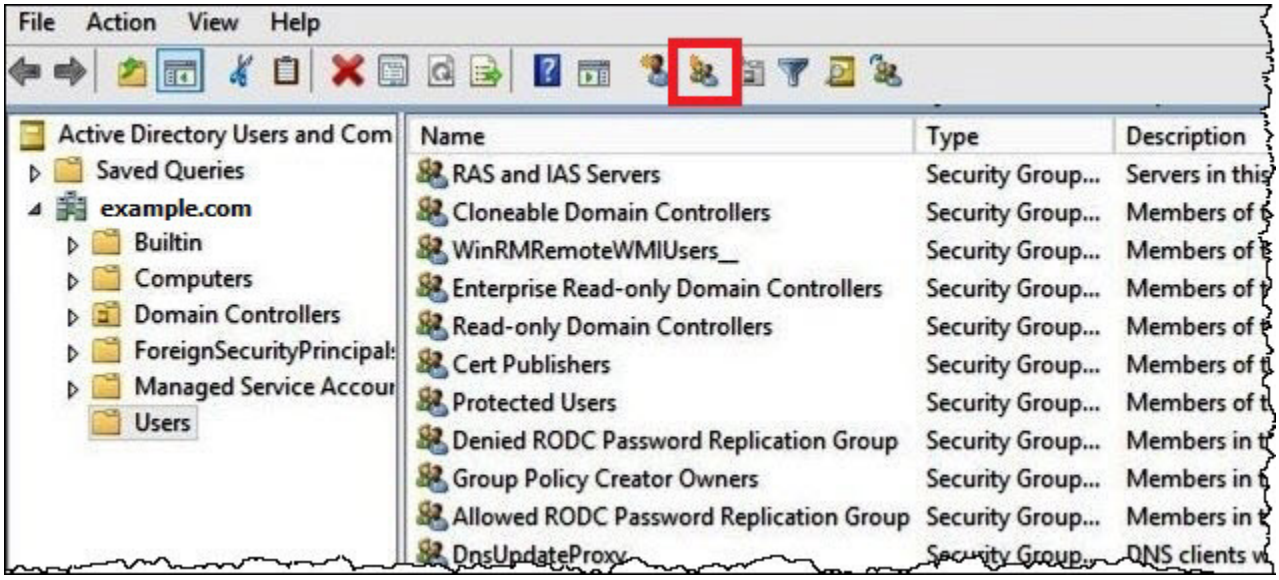
11. 选择确定。

创建 AD 组来表示数据访问模式

您可以创建 AD 组，其成员在登录 AWS 时担任 `adfs-data-access` IAM 角色。以下示例创建了一个名为 aws-adfs-data-access 的 AD 组。

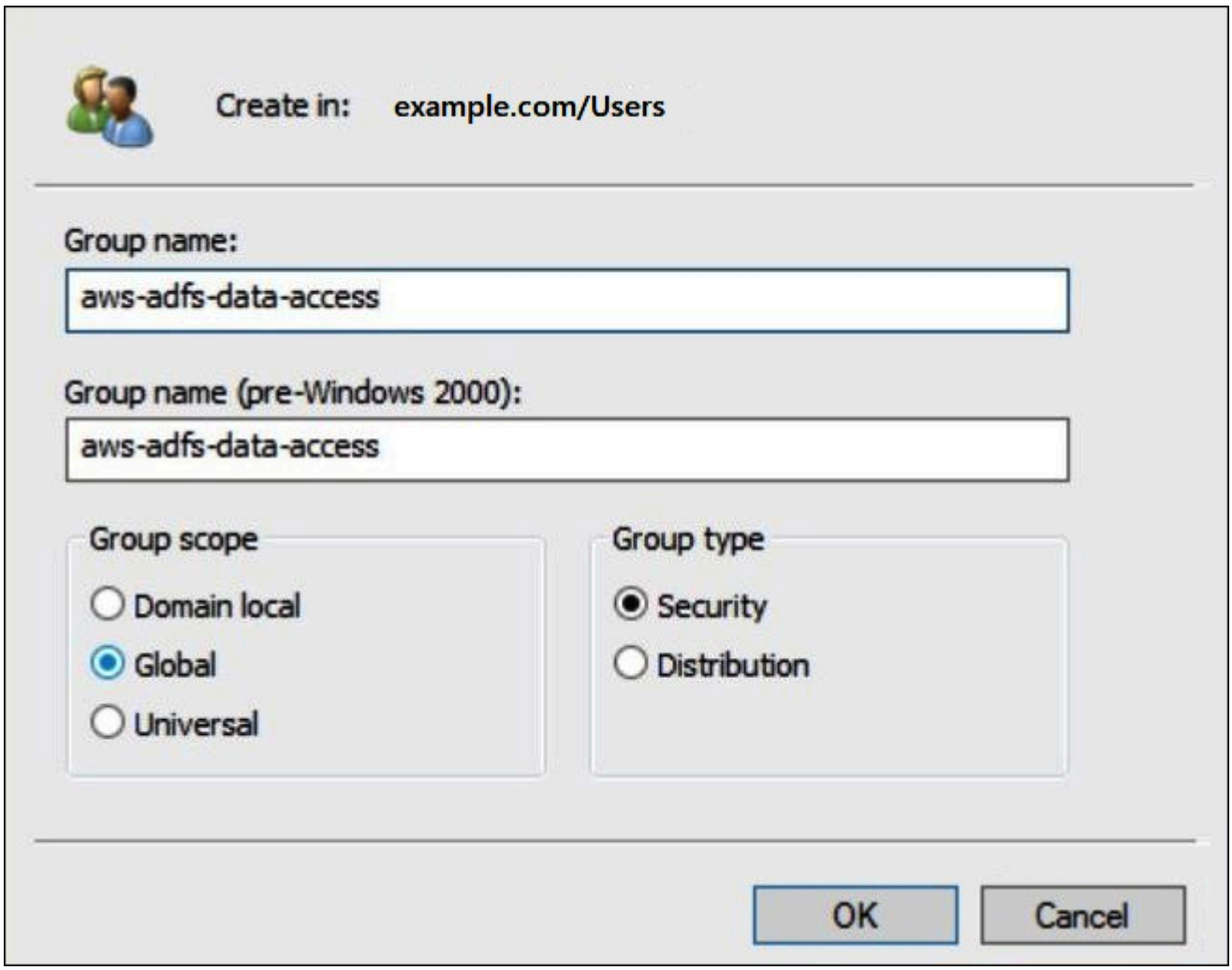
创建 AD 组

1. 在“服务器管理器”控制面板上，从 Tools（工具）菜单中，选择 Active Directory Users and Computers（Active Directory 用户和计算机）。
2. 在工具栏上，选择 Create new group（创建新组）选项。



3. 在 New Object – Group（新对象 – 组）对话框中，输入以下信息：

   - 对于 Group Name（组名称），输入 **`aws-adfs-data-access`**。
   - 对于 Group scope（组范围），选择 Global（全局）。
   - 对于 Group type（组类型），选择 Security（安全）。



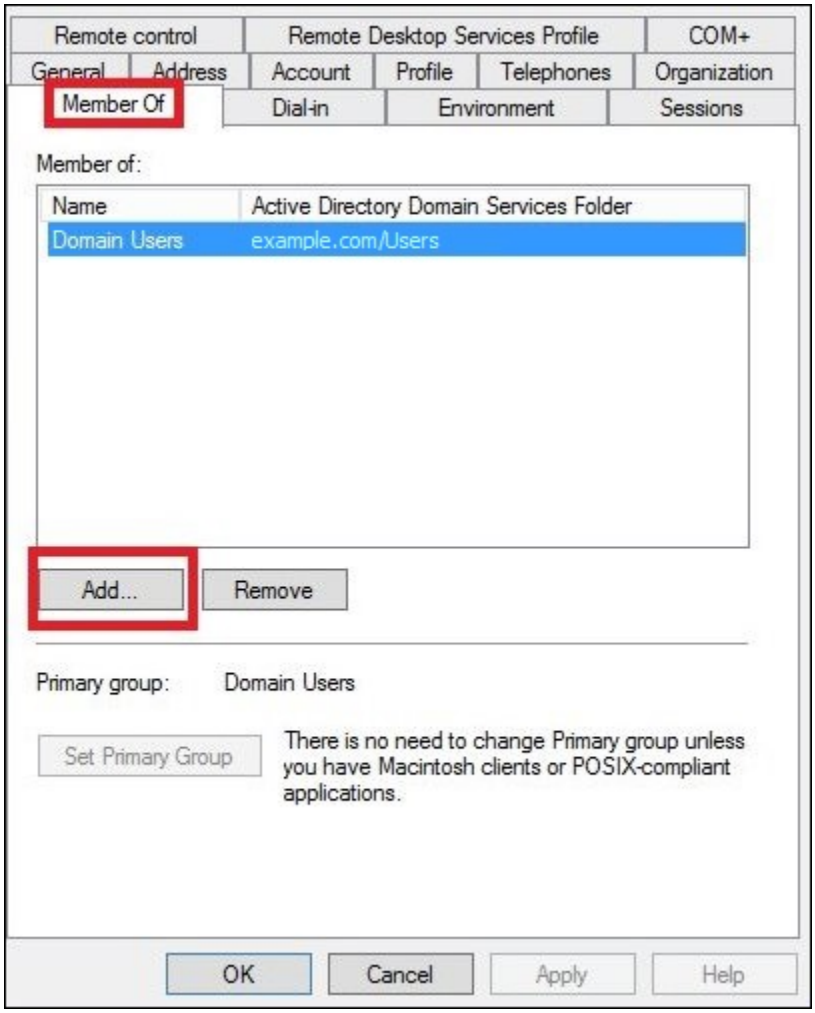
4. 选择确定。

将 AD 用户添加到相应的组

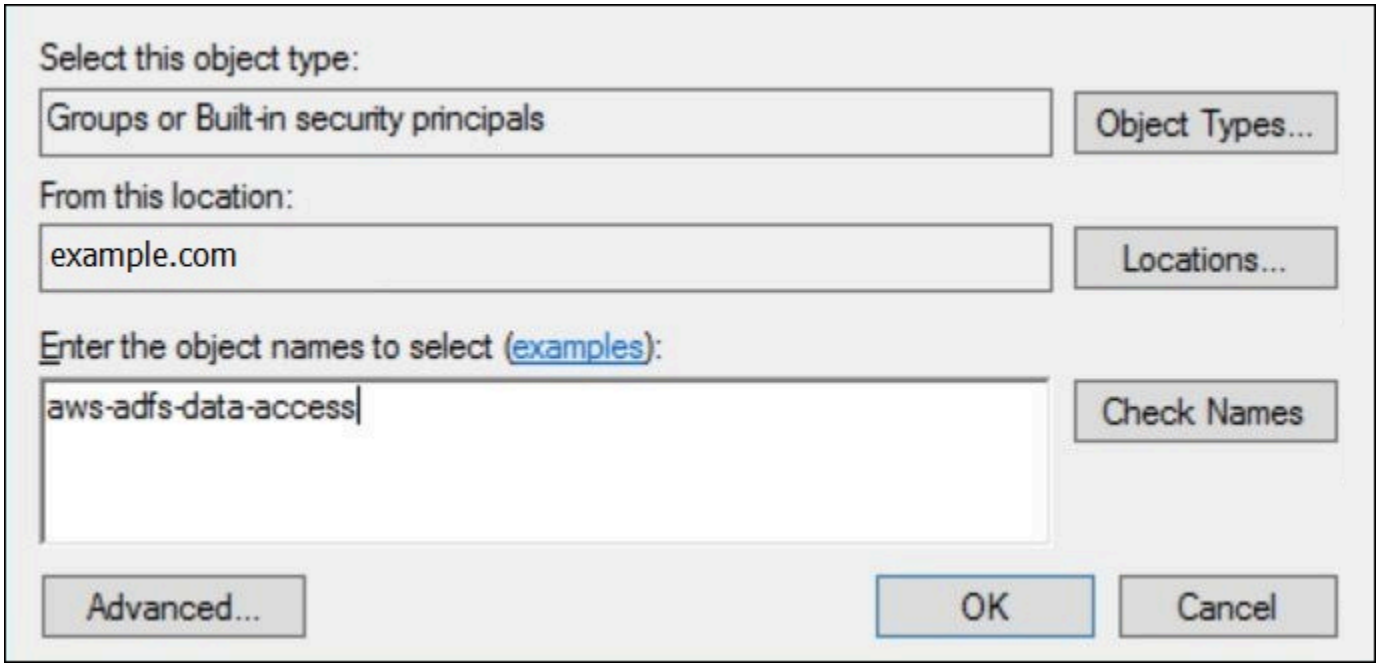
现在，您已创建 AD 用户和 AD 组，可将该用户添加到该组。

将 AD 用户添加到 AD 组

1. 在“服务器管理器”控制面板上，在 Tools（工具）菜单中，选择 Active Directory Users and Computers（Active Directory 用户和计算机）。
2. 对于 First name（名字）和 Last name（姓氏），选择一个用户（例如 Jane Doe）。
3. 在用户的 Properties（属性）对话框中，在 Member Of（隶属于）选项卡上，选择 Add（添加）。

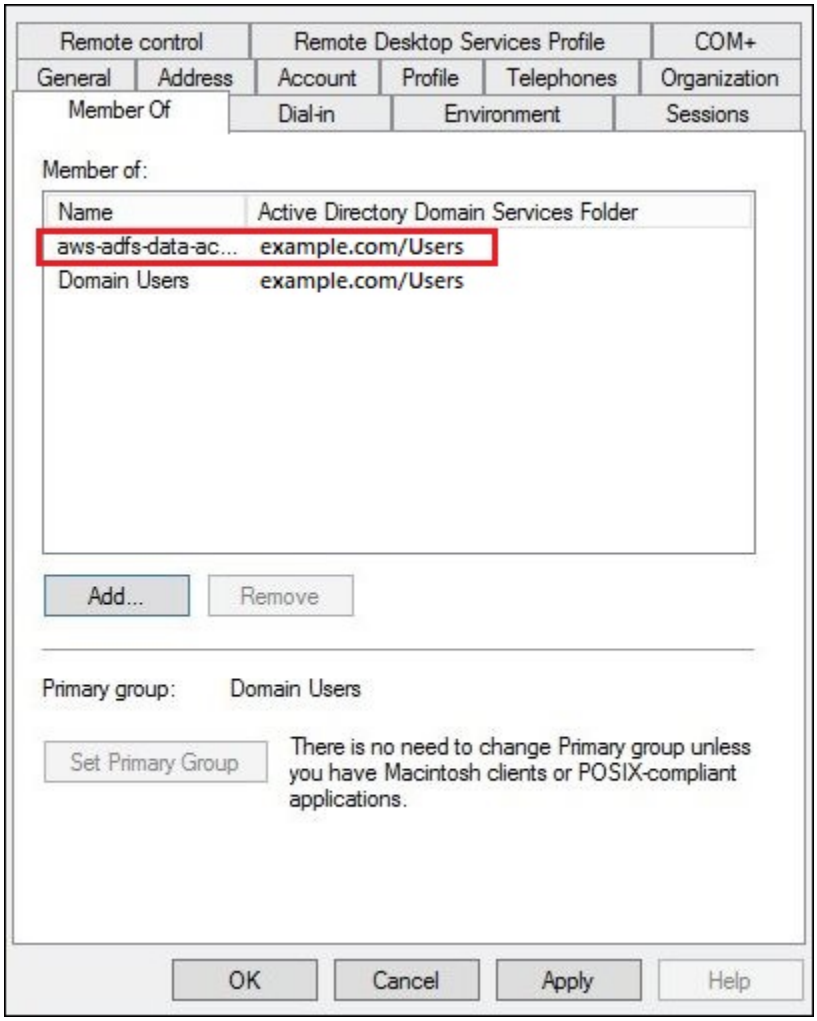


4. 根据要求添加一个或多个 AD FS 组。本教程添加了 aws-adfs-data-access 组。
5. 在 Select Groups（选择组）对话框中，对于 Enter the object names to select（输入要选择的对象名称），输入您创建的 AD FS 组名称（例如 **aws-adfs-data-access**），然后选择 Check Names（检查名称）。



6. 选择确定。

   在用户的 Properties（属性）对话框中，AD 组名称将在 Member of（隶属于）列表中显示。



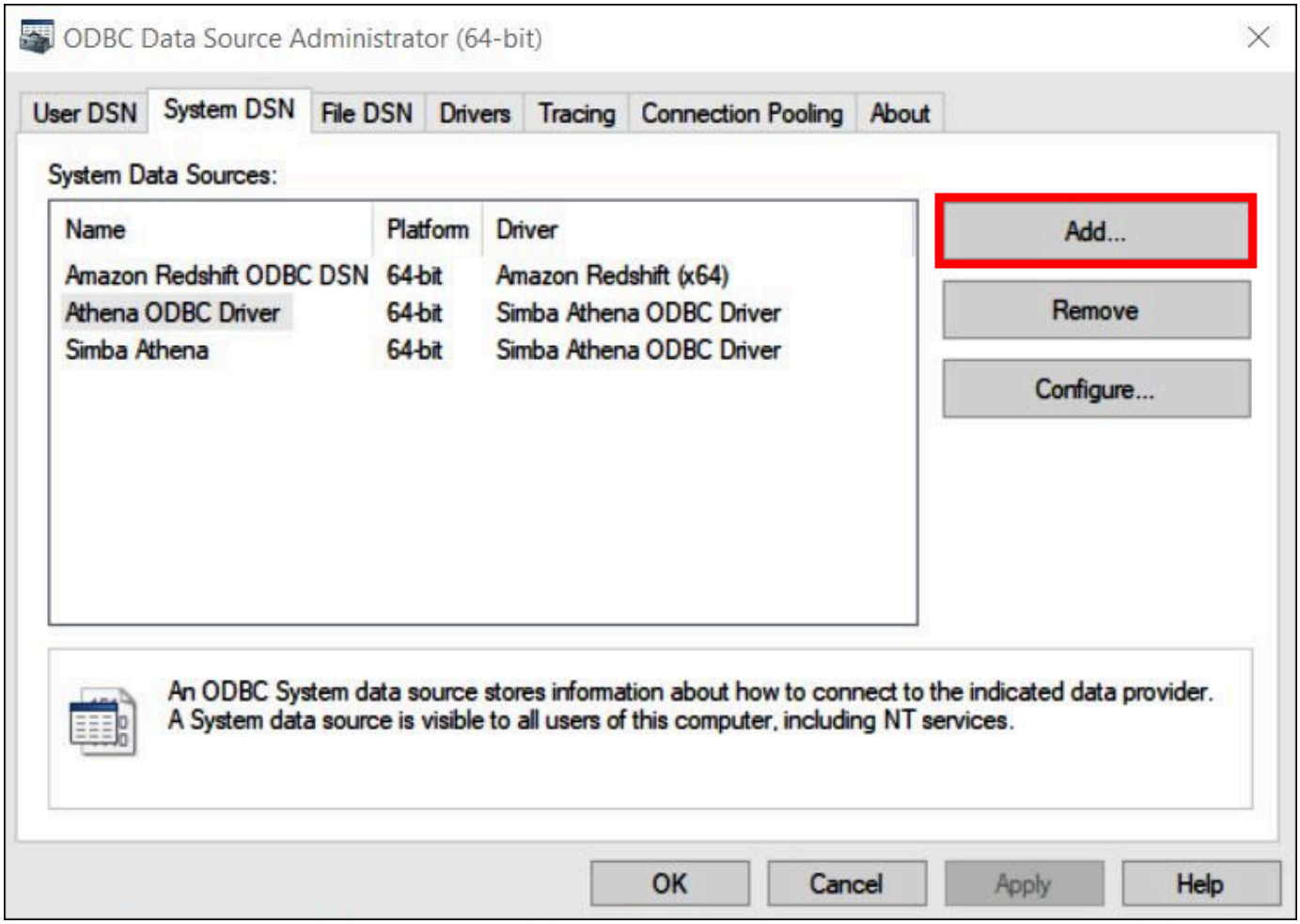
7. 选择 Apply（应用），然后选择 OK（确定）。

4. 配置 AD FS ODBC 与 Athena 的连接

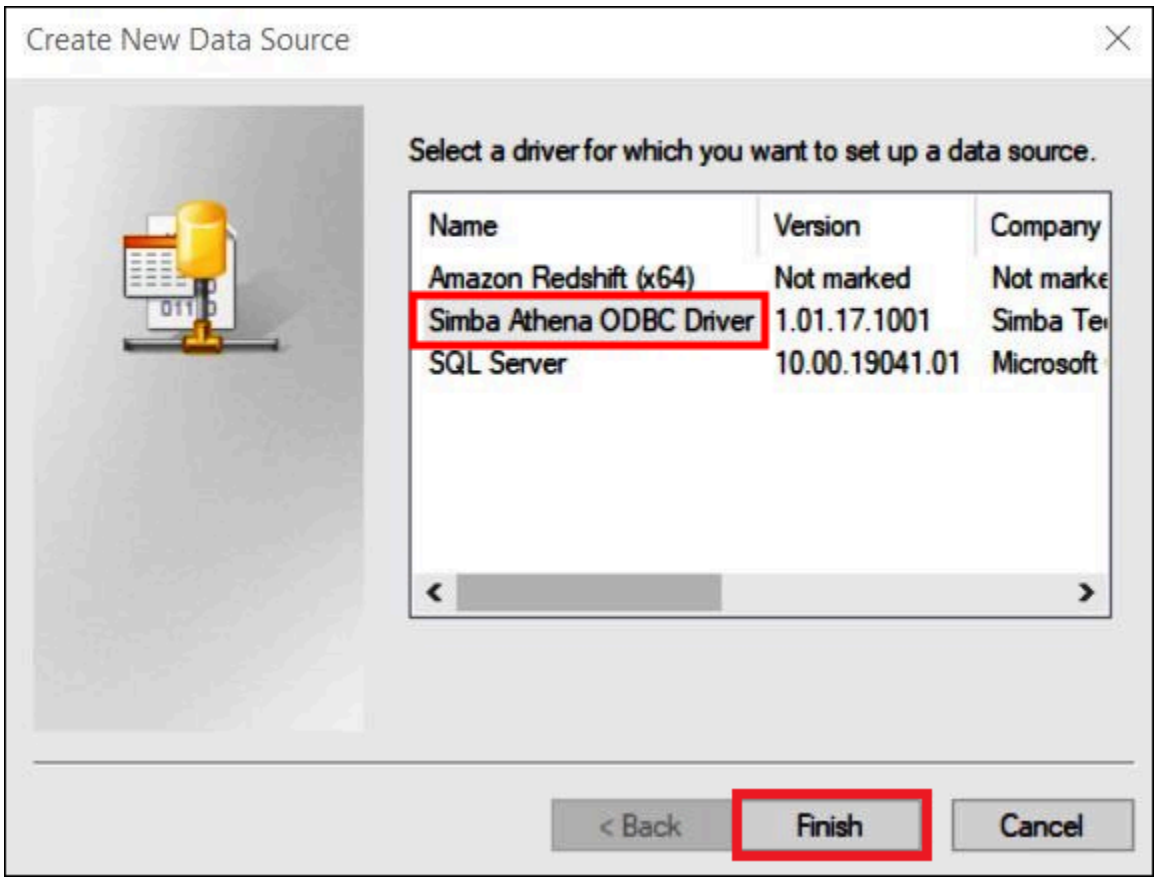
创建 AD 用户和组后，您就可以使用 Windows 中的 ODBC 数据来源程序为 AD FS 配置 Athena ODBC 连接。

配置 AD FS ODBC 与 Athena 的连接

1. 为 Athena 安装 ODBC 驱动程序。有关下载链接，请参阅 通过 ODBC 连接到 Amazon Athena。
2. 在 Windows 中，依次选择 Start（开始）、ODBC Data Sources（ODBC 数据来源）。
3. 在 ODBC 数据来源管理器程序中，选择 Add（添加）。

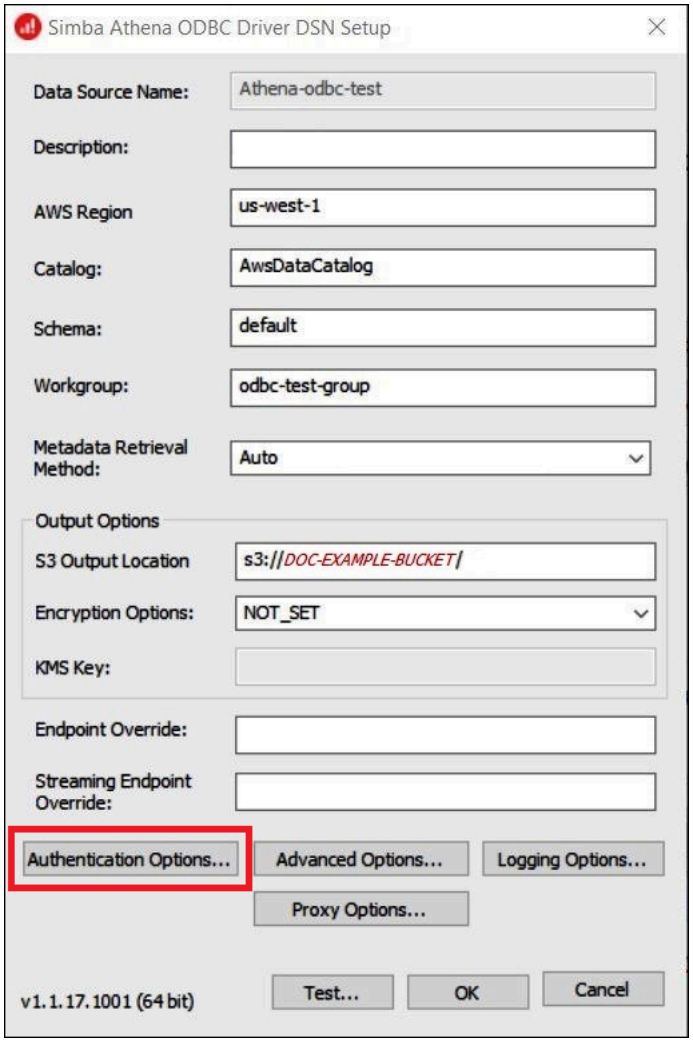


4. 在 Create New Data Source（创建新数据来源）对话框中，选择 Simba Athena ODBC 驱动程序，然后选择 Finish（完成）。

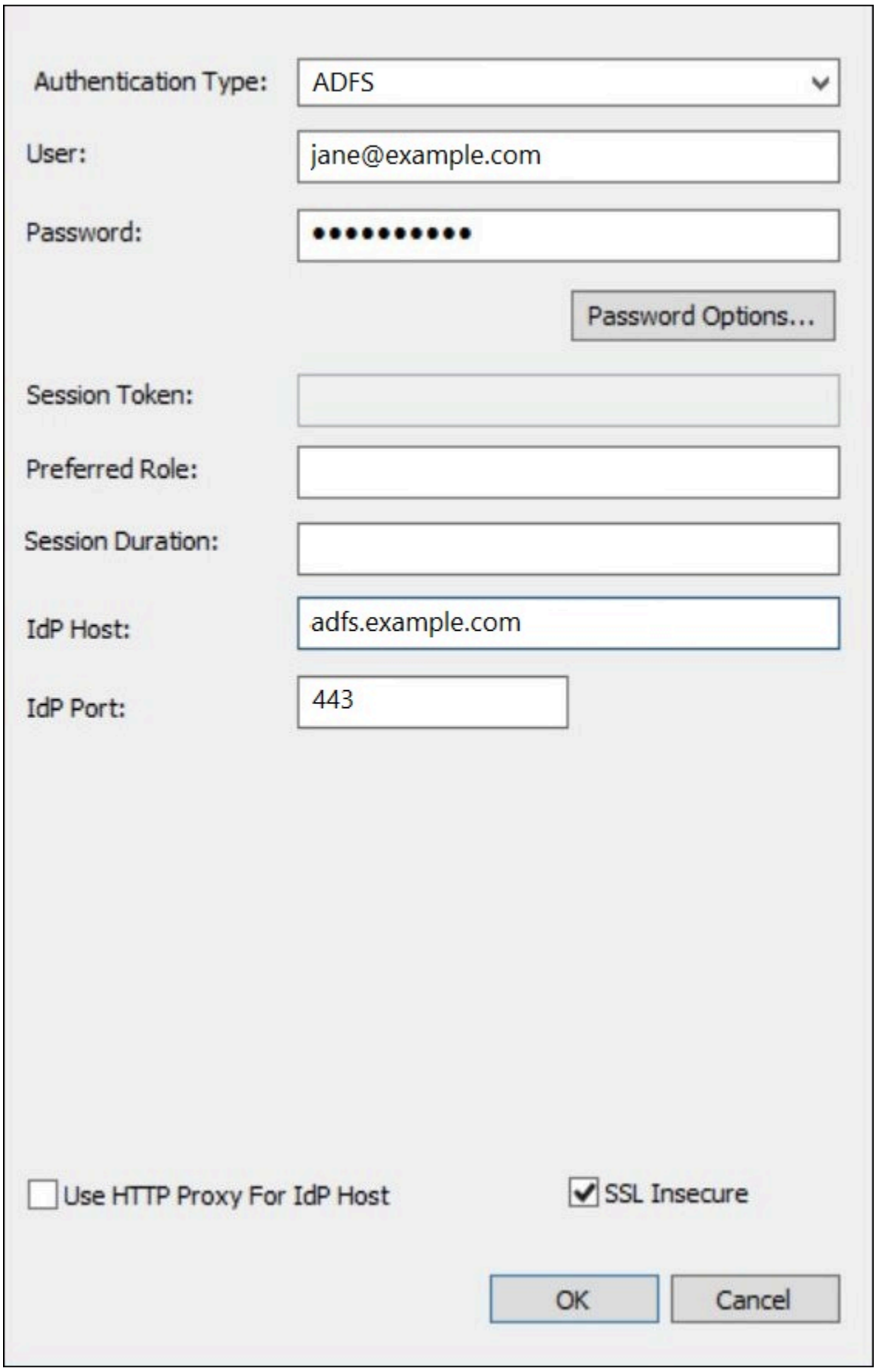


5. 在 Simba Athena ODBC Driver DSN Setup（Simba Athena ODBC 驱动程序 DSN 设置）对话框中，输入下列值：

   - 对于 Data Source Name（数据来源名称），输入数据来源的名称（例如， **`Athena-odbc-test`**）。
   - 在 Description（说明）中，为数据来源输入说明。
   - 在 AWS 区域 中，输入您正在使用的 AWS 区域（例如 **`us-west-1`**）。
   - 在 S3 Output Location（S3 输出位置）中，输入要存储输出的 Amazon S3 路径。



6. 选择 Authentication Options（身份验证选项）。
7. 在 Authentication Options（身份验证选项）对话框中，指定下列值：
   - 对于 Authentication Type（身份验证类型），选择 ADFS。
   - 对于 User（用户），输入用户的电子邮件地址（例如 **jane@example.com**）。
   - 对于 Password（密码），输入用户 ADFS 密码的值。
   - 对于 IdP Host（IdP 主机），输入 AD FS 服务器名称（例如 **adfs.example.com**）。
   - 对于 IdP Port（IdP 端口），使用默认值 443。
   - 选择 SSL Insecure（SSL 不安全）选项。


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. 选择 OK（确定）以关闭 Authentication Options（身份验证选项）。
9. 选择 Test（测试）以测试连接，或者 OK（确定）以完成。

使用 Okta 插件和 Okta 身份提供者为 ODBC 配置 SSO

本页说明了如何配置 Amazon Athena ODBC 驱动程序和 Okta 插件，以使用 Okta 身份提供者添加单点登录（SSO）功能。

先决条件

要完成本教程中的步骤，需要以下内容：

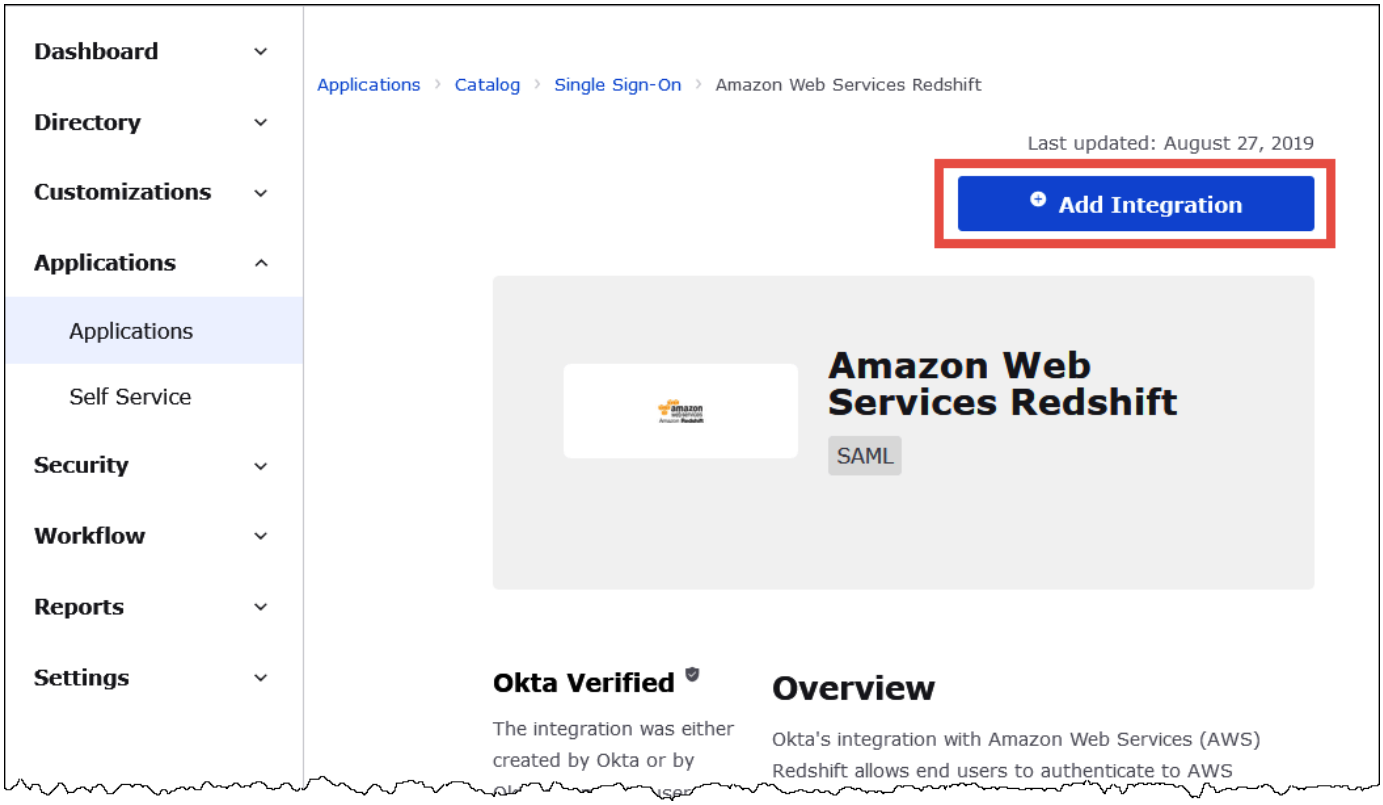
- Amazon Athena ODBC 驱动程序。有关下载链接，请参阅 通过 ODBC 连接到 Amazon Athena。
- 要与 SAML 一起使用的 IAM 角色。有关更多信息，请参阅《IAM 用户指南》中的创建用于 SAML 2.0 联合身份验证的角色。
- Okta 账户。要了解相关信息，请访问 Okta.com。

在 Okta 中创建应用程序集成

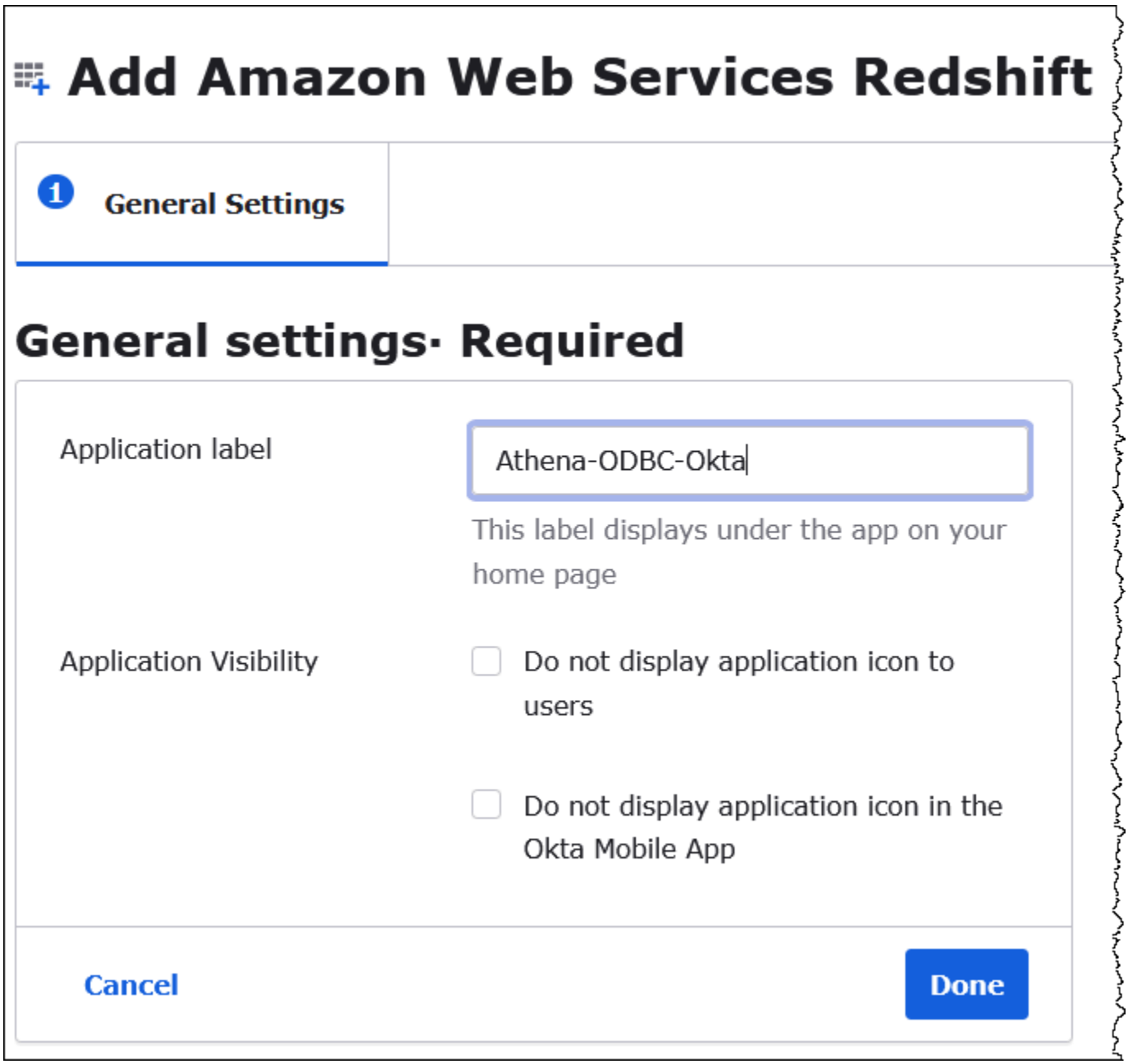
首先，使用 Okta 控制面板创建和配置 SAML 2.0 应用程序以单点登录 Athena。您可以在 Okta 中使用现有的 Redshift 应用程序来配置对 Athena 的访问权限。

在 Okta 中创建一个应用程序集成

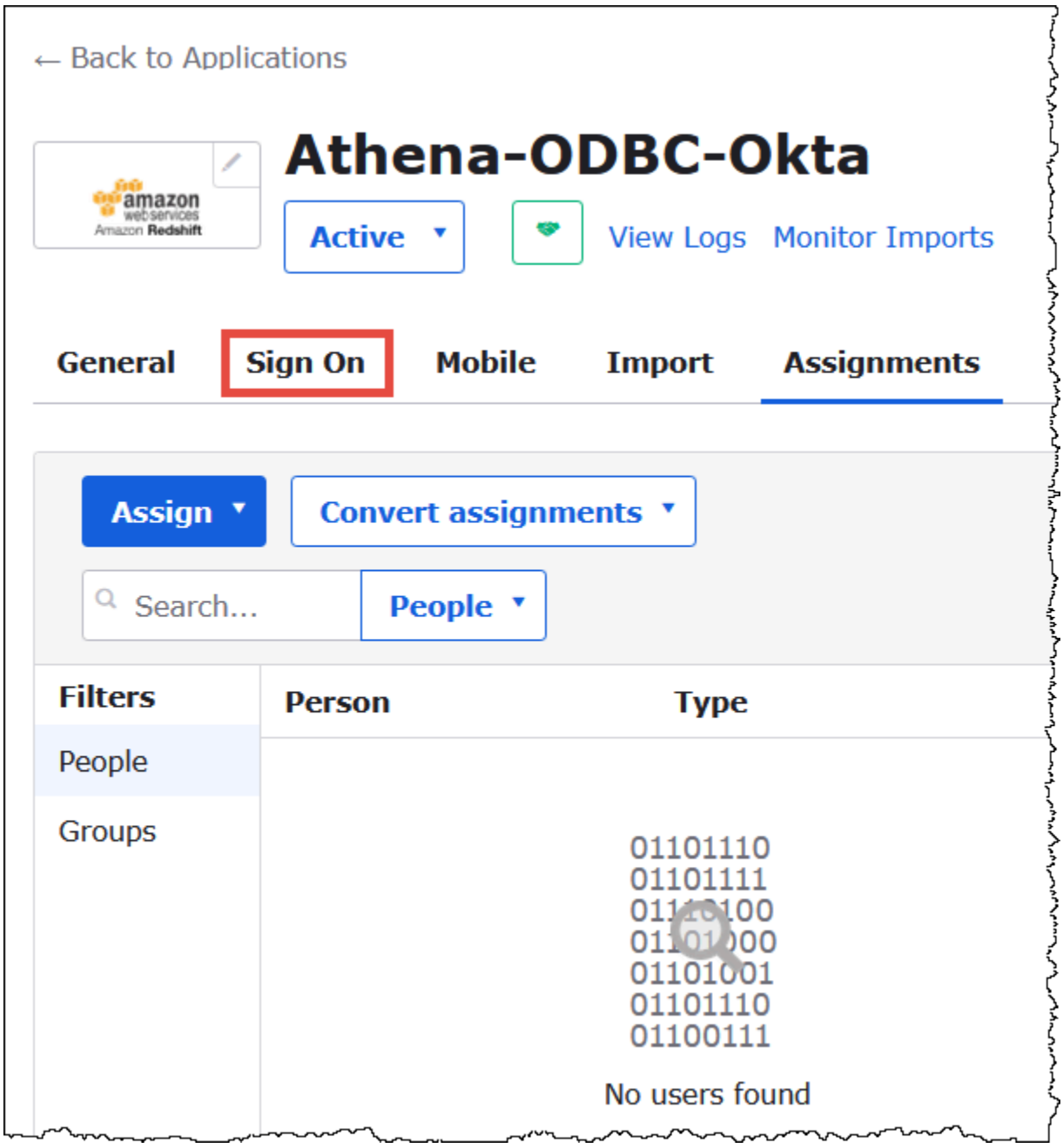
1. 在 Okta.com 上登录账户的管理员页面。
2. 在导航窗格中，选择 Applications（应用程序）、Applications（应用程序）。
3. 在 Applications（应用程序）页面上，选择 Browse App Catalog（浏览应用程序目录）。
4. 在 Browse App Integration Catalog（浏览应用程序集成目录）页面的 Use Case（使用场景）部分中，选择 All Integrations（所有集成）。
5. 在搜索框中输入 Amazon Web Services Redshift，然后选择 Amazon Web Services Redshift SAML。
6. 选择 Add Integration（添加集成）。



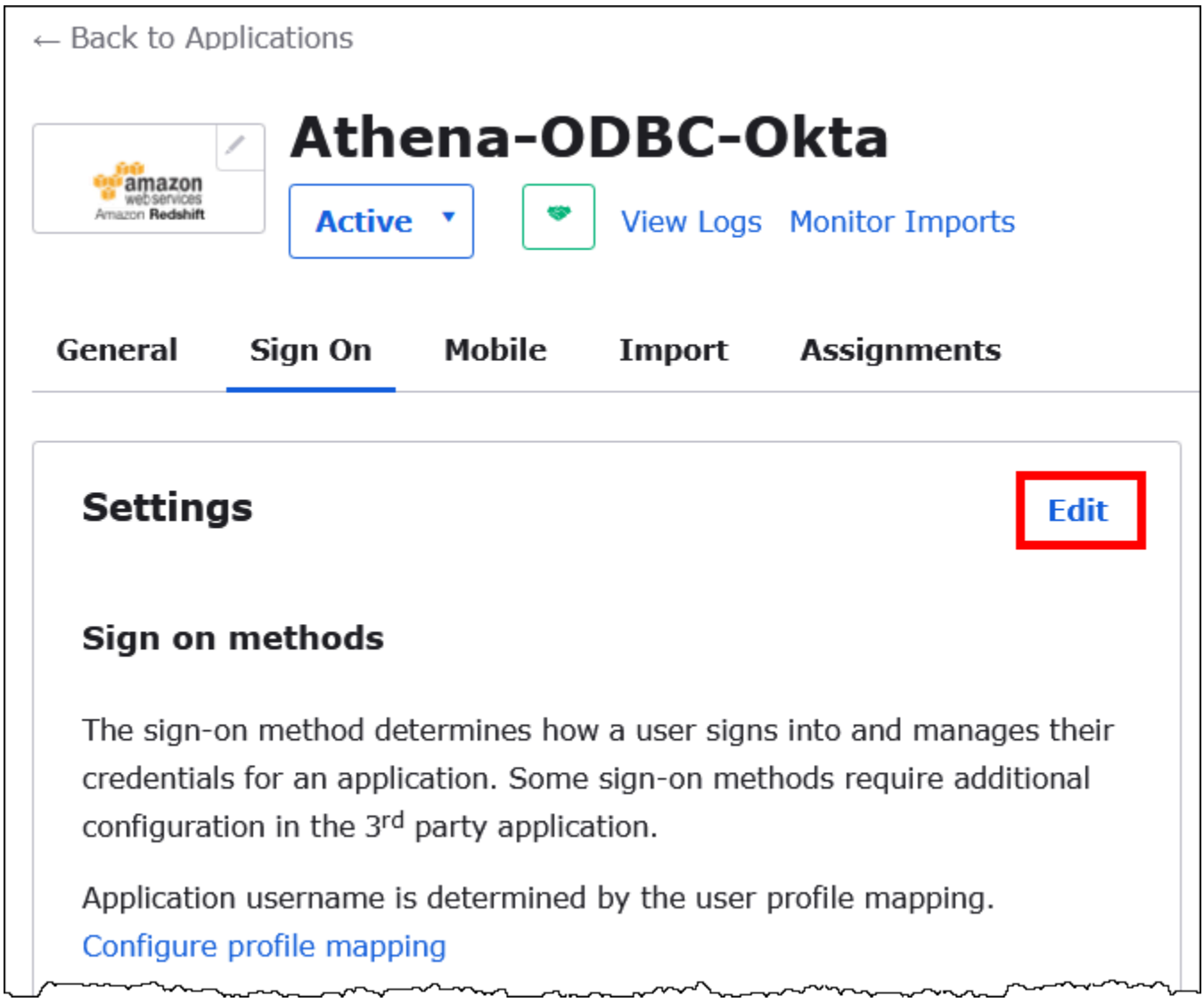
7. 在 General Settings Required（必填常规设置）部分中，对于 Application label（应用程序标注），输入应用程序的名称。本教程使用的名称是 Athena-ODBC-Okta。



8. 选择完成。
9. 在 Okta 应用程序页面（例如，Athena-ODBC-Okta）上，选择 Sign On（登录）。



10. 在 Settings（设置）部分中，选择 Edit（编辑）。



11. 在 Advanced Sign-on Settings（高级登录设置）部分中，配置以下值。

    - 对于 IdP ARN and Role ARN（IdP ARN 和角色 ARN），输入您的 AWS IDP ARN 和角色 ARN，用逗号分隔。有关 IAM 角色格式的信息，请参阅《IAM 用户指南》中的为身份验证响应配置 SAML 断言。
    - 对于 Session Duration（会话持续时间），输入一个介于 900 到 43200 秒之间的值。本教程使用的是默认值 3600（1 小时）。

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

Athena 不使用 DbUser Format（DbUser 格式）、AutoCreate（自动创建）和 Allowed DBGroups（允许的数据库组）设置。您不需要配置这些设置。

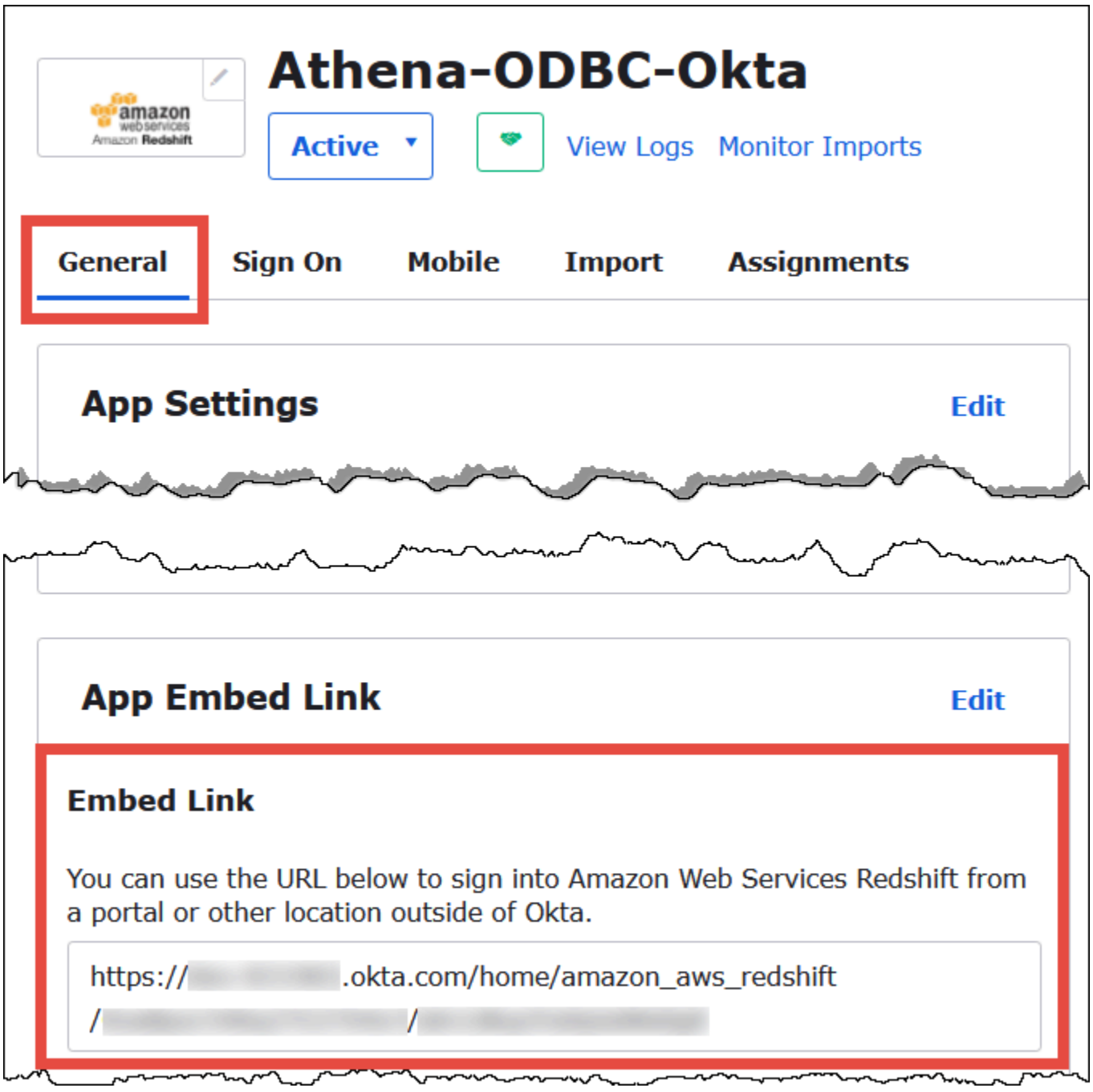
12. 选择保存。

从 Okta 检索 ODBC 配置信息

您现在已经创建了 Okta 应用程序，可以检索应用程序的 ID 和 IdP 主机 URL。稍后在配置 ODBC 以连接到 Athena 时将需要这些信息。

从 Okta 检索 ODBC 配置信息

1. 选择 Okta 应用程序的 General（常规）选项卡，然后向下滚动到 App Embed Link（应用程序嵌入式链接）部分。



Embed Link（嵌入式链接）URL 采用以下格式：

```
https://trial-1234567.okta.com/home/amazon_aws_redshift/Abc1de2fghi3J45kL678/
abc1defghij2klmNo3p4
```

2. 从您的 Embed Link（嵌入式链接）URL 中提取并保存以下片段：

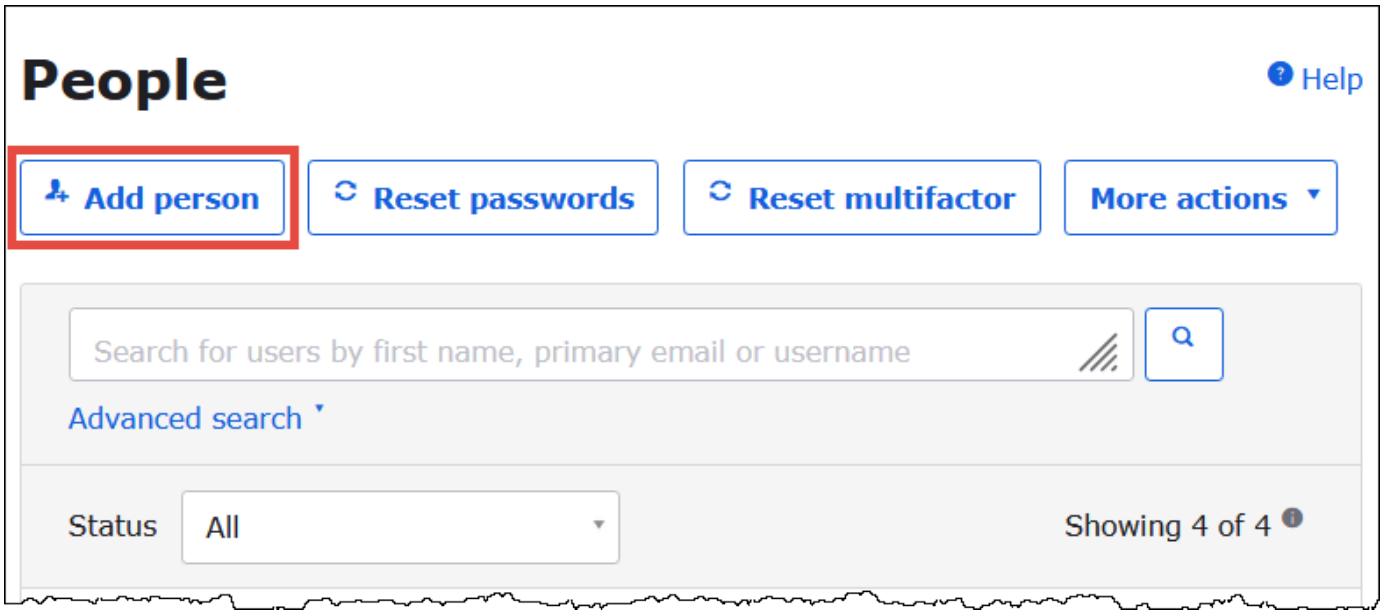
- `https://` 之后的第一个片段，直至（包含）`okta.com`（例如，trial-1234567.okta.com）。这是您的 IdP 主机。
- URL 的最后两个片段，包括中间的正斜杠。这些片段是两个有 20 个字符的字符串，混合了数字和大小写字母（例如 Abc1de2fghi3J45kL678/abc1defghij2klmNo3p4）。这是您的应用程序 ID。

为 Okta 应用程序添加一个用户

您现在可以为您的 Okta 应用程序添加一个用户。

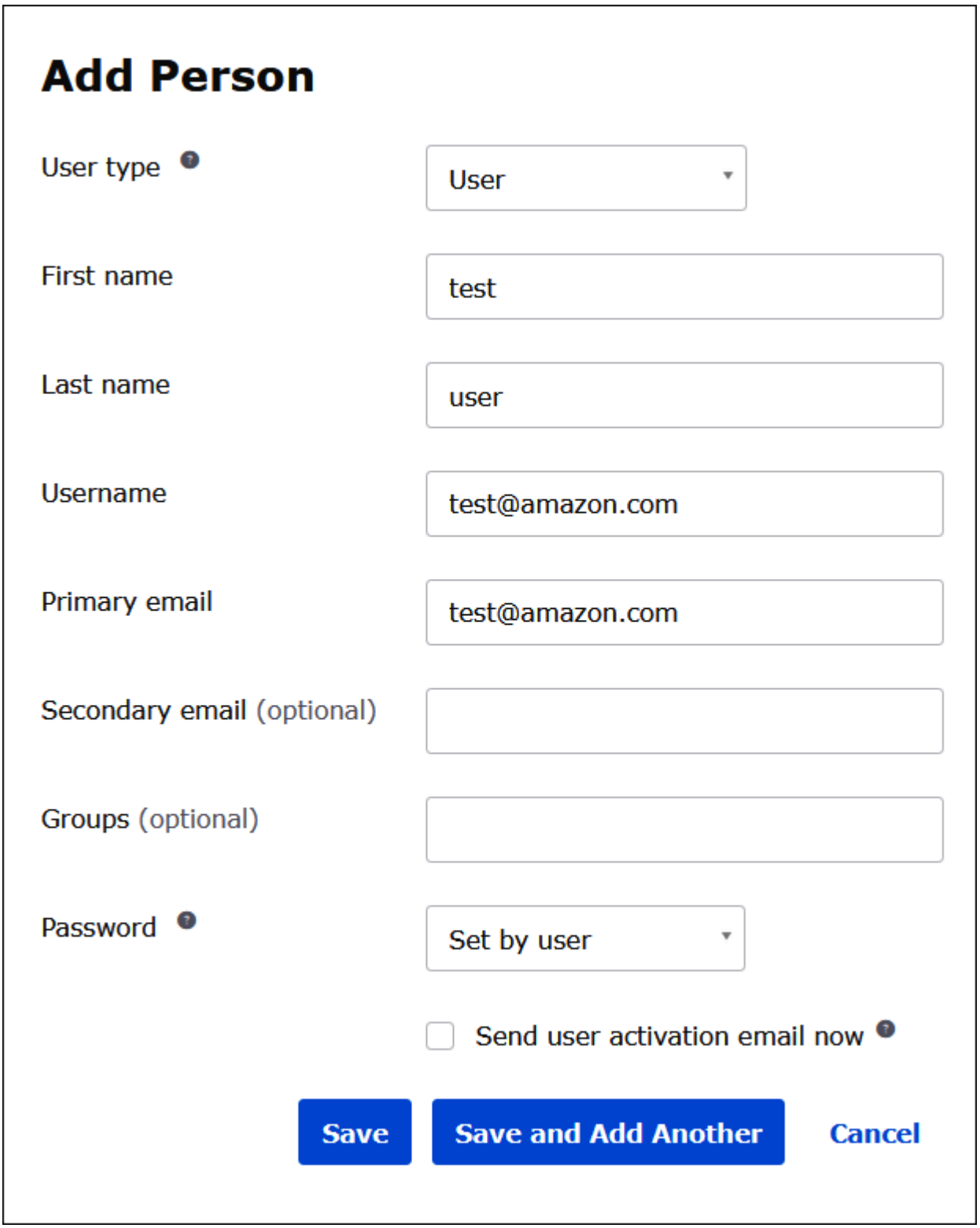
为 Okta 应用程序添加一个用户

1. 在左侧导航窗格中，选择 Directory（目录），然后选择 People（人员）。
2. 选择 Add person（添加人员）。



3. 在 Add Person（添加人员）对话框中，输入下列信息。

- 对于 First name（名字）和 Last name（姓氏），输入值。本教程使用的是 **`test user`**。
- 输入 Username（用户名）和 Primary email（主电子邮件）的值。对于这两个参数，本教程使用的都是 **`test@amazon.com`**。您对密码的安全要求可能会有所不同。

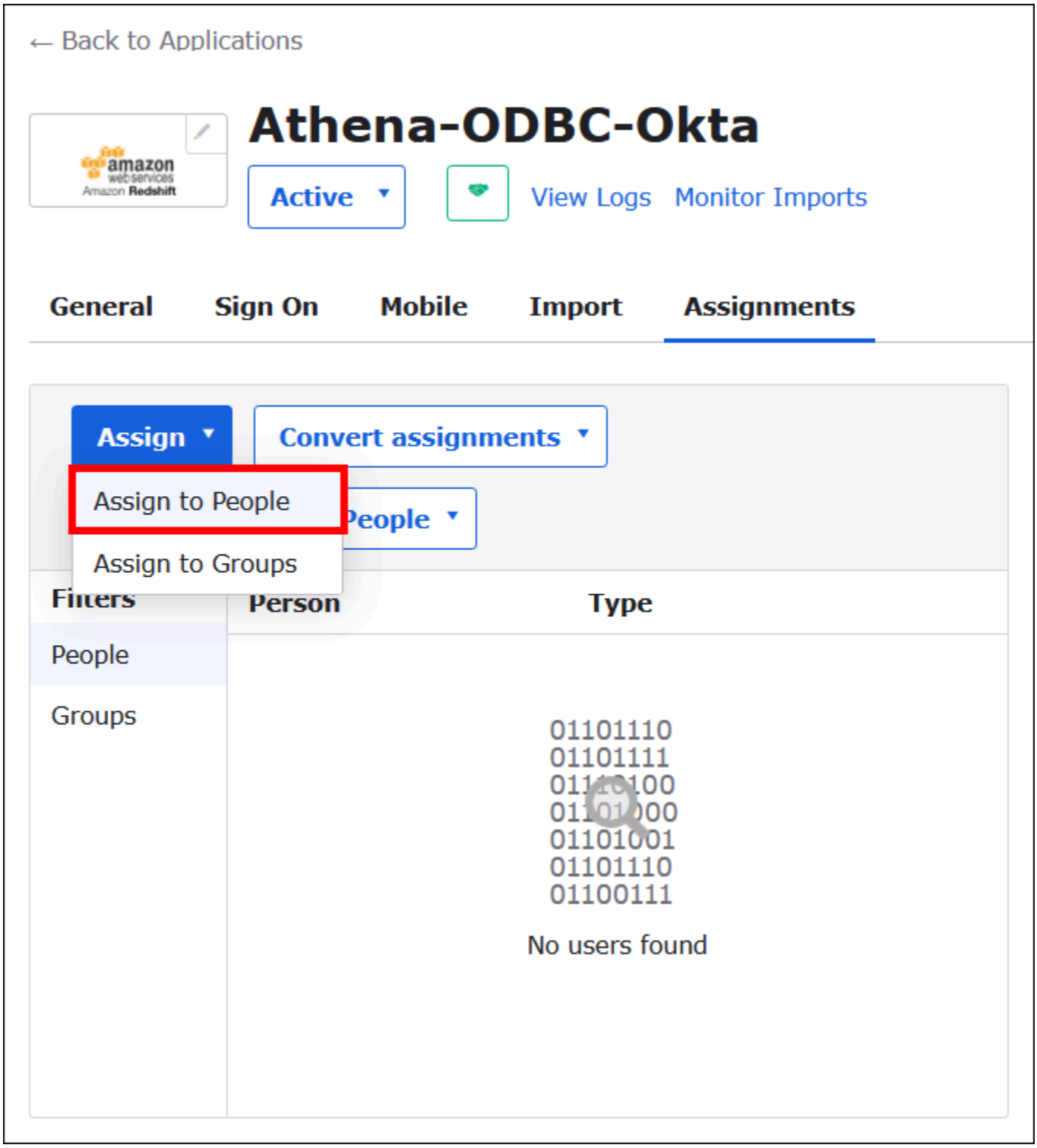


4. 选择保存。

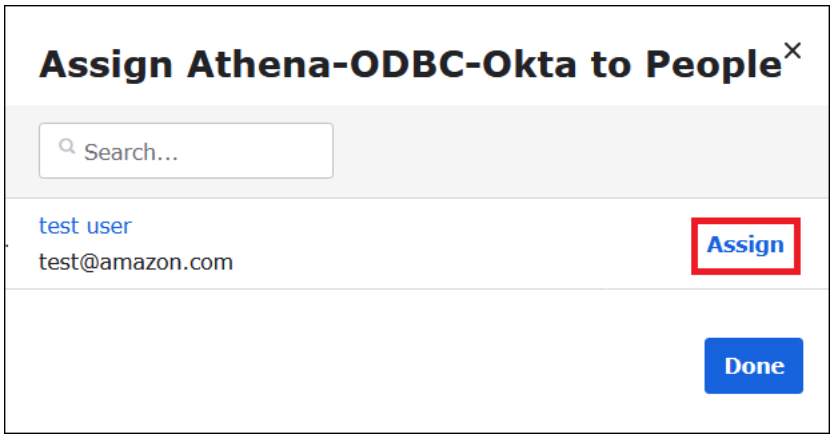
现在您可以将创建的用户分配到您的应用程序。

要将此用户分配给您的应用程序：

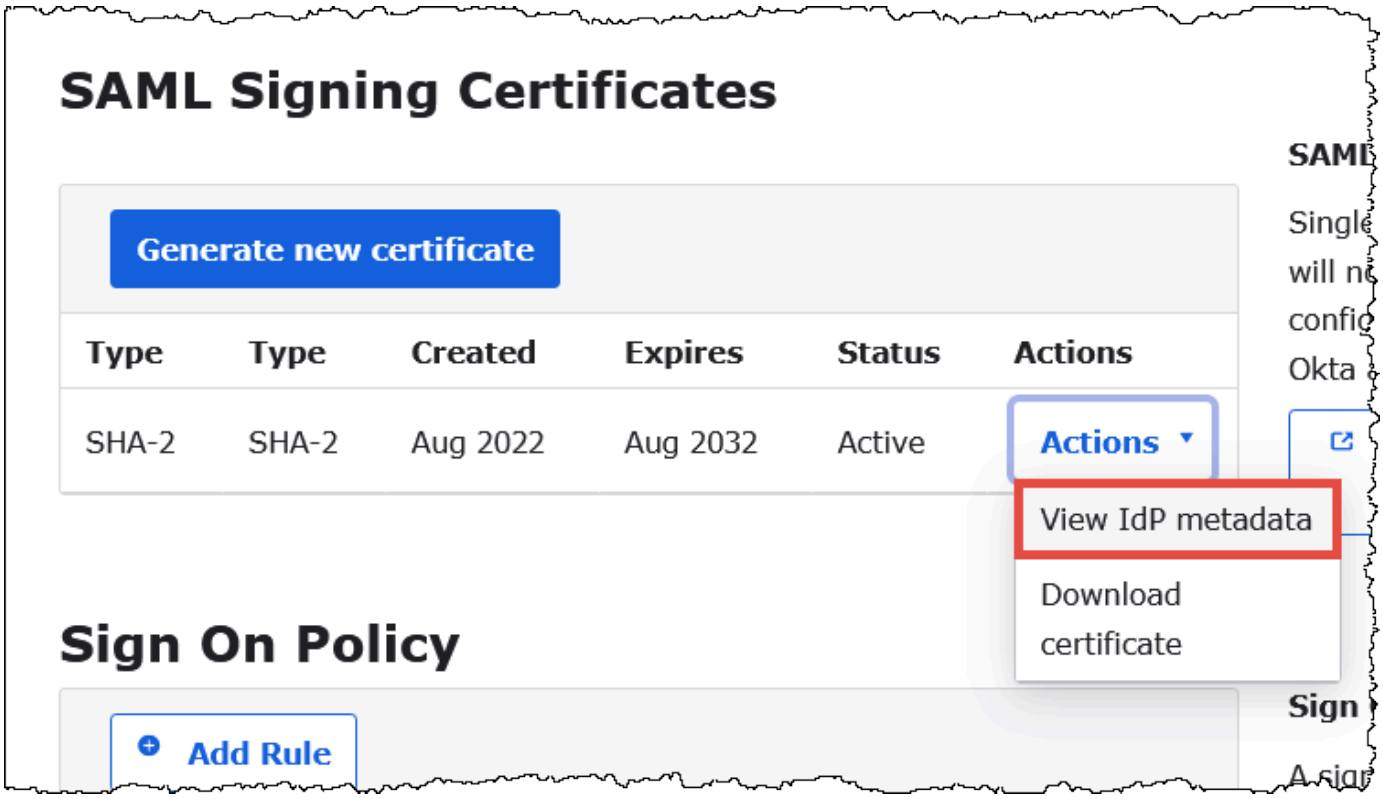
1. 在导航窗格中，选择 Applications（应用程序）、Applications（应用程序），然后选择应用程序的名称（例如，Athena-ODBC-Okta)。
2. 选择 Assign（分配），然后选择 Assign to People（分配给人员）。



3. 为您的用户选择 Assign（分配）选项，然后选择 Done（完成）。



4. 在提示时选择 Save and Go Back（保存并返回）。此对话框会显示用户的状态为 Assigned（已分配）。
5. 选择完成。
6. 选择 Sign On（登录）选项卡。
7. 向下滚动到 SAML Signing Certificates（SAML 签名证书）部分。
8. 选择操作。
9. 打开 View IdP metadata（查看 IdP 元数据）的上下文（右键单击），然后选择浏览器选项以保存文件。
10. 使用 `.xml` 扩展名保存文件。

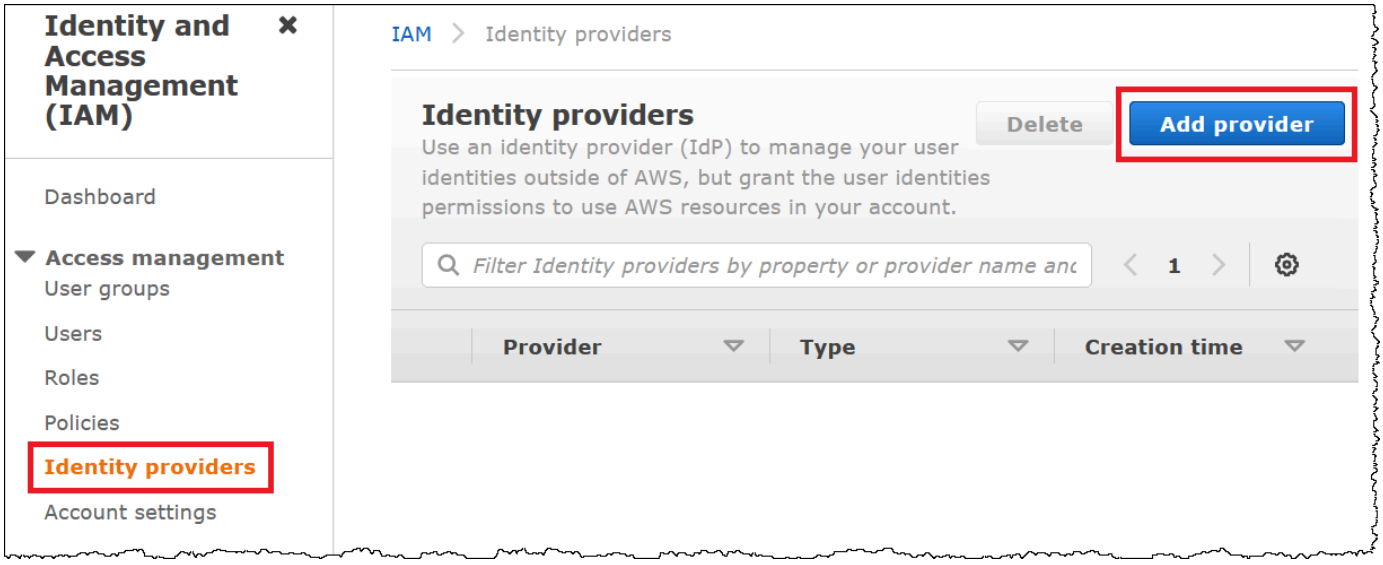


创建一个 AWS SAML 身份提供者和角色

现在您可以在 AWS 中将元数据 XML 文件上传到 IAM 控制台。您将使用此文件来创建一个 AWSSAML 身份提供者和角色。使用 AWS 服务管理员帐户来执行这些步骤。

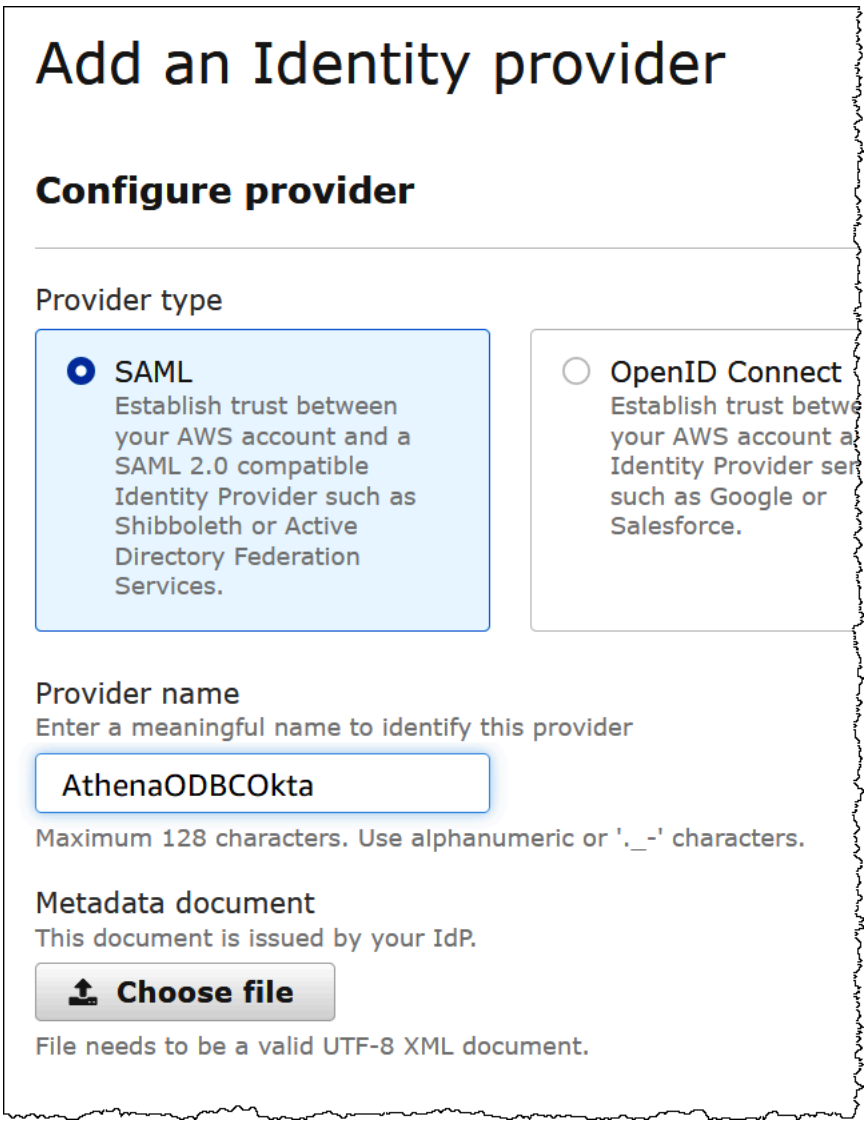
在 AWS 中创建一个 SAML 身份提供者和角色

1. 登录AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/IAM/。
2. 在导航窗格中，选择 Identity providers（身份提供程序），然后选择 Add provider（添加提供商）。



3. 在 Add an Identity provider（添加身份提供者）页面上，对于 Configure provider（配置提供者），输入以下信息。

   - 对于 Provider type（提供程序类型），选择 SAML。
   - 对于 Provider name（提供者名称），为您的提供者输入一个名称（例如 **`AthenaODBCOkta`**）。
   - 对于 Metadata document（元数据文档），请使用 Select file（选择文件）选项上载您下载的身份提供程序 (IdP) 元数据 XML 文件。



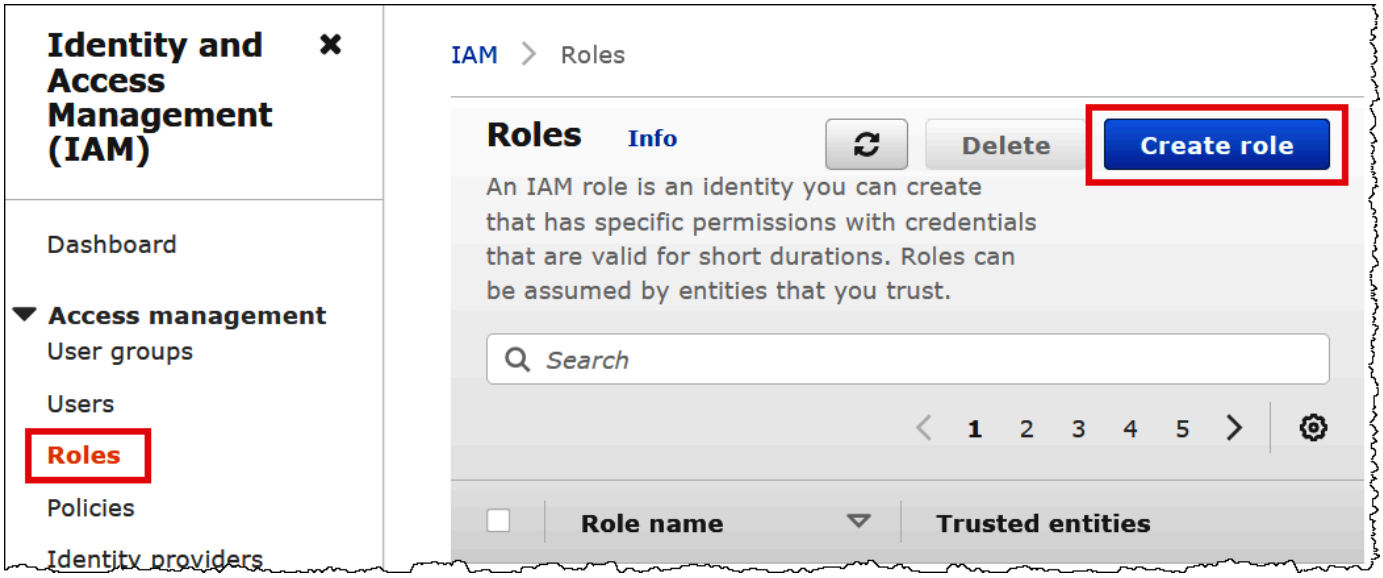
4. 选择 Add provider（添加提供程序）。

创建一个用于访问 Athena 和 Amazon S3 的 IAM 角色

您现在可以创建一个用于访问 Athena 和 Amazon S3 的 IAM 角色。您需要将此角色分配给您的用户，从而可以为该用户提供对 Athena 的单点登录访问权限。

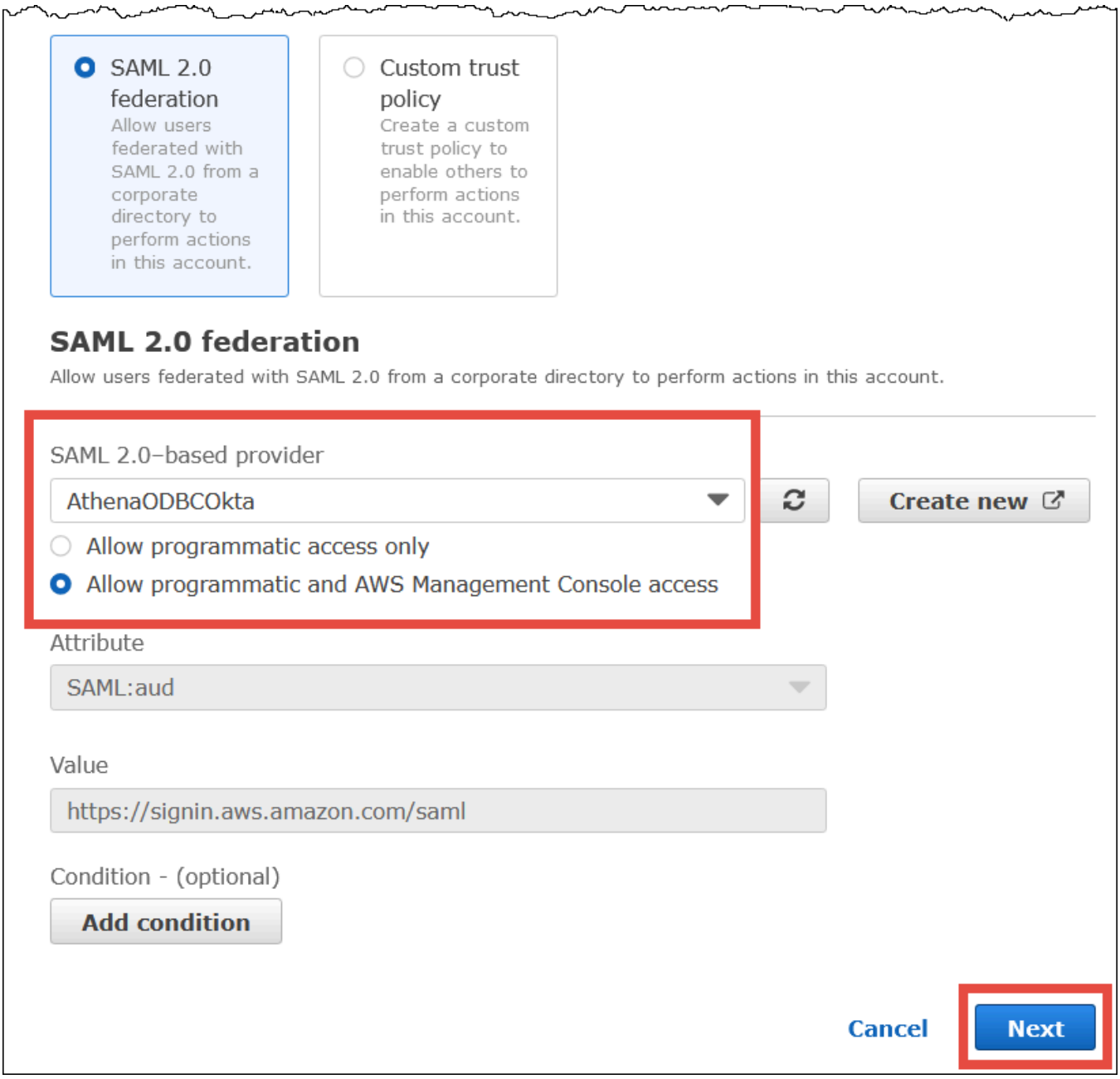
为您的用户创建一个 IAM 角色

1. 在 IAM 控制台的导航窗格中，选择 Roles（角色），然后选择 Create role（创建角色）。

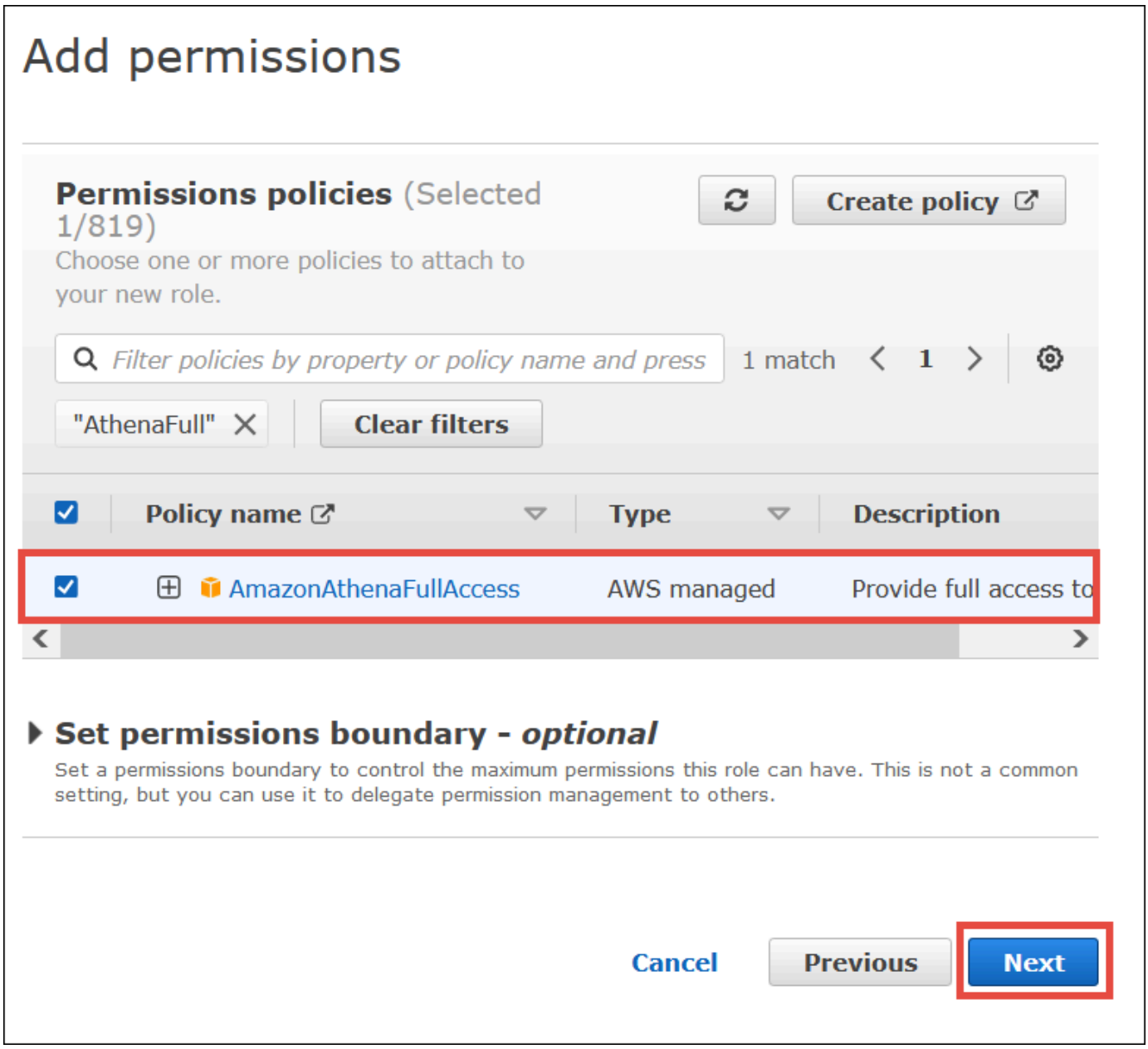


2. 在 Create role（创建角色）页面上，选择以下选项：

   - 对于 Select type of trusted entity（选择受信任实体的类型），选择 SAML 2.0 Federation。
   - 对于 SAML 2.0–based provider（基于 SAML 2.0 的提供者），请选择您创建的 SAML 身份提供者（例如，AthenaODBCOkta）。
   - 选择 Allow programmatic and AWS Management Console access（允许编程和访问）。



3. 选择下一步。
4. 在 Add Permissions（添加权限）页面上，对于 Filter policies（筛选策略），输入 **`AthenaFull`**，然后按 ENTER 键。
5. 选择名为 `AmazonAthenaFullAccess` 的托管式策略，然后选择 Next（下一步）。



6. 在 Name, review, and create（命名、检查并创建）页面上，对于 Role name（角色名称），输入角色的名称（例如，**`Athena-ODBC-OktaRole`**），然后选择 Create role（创建角色）。

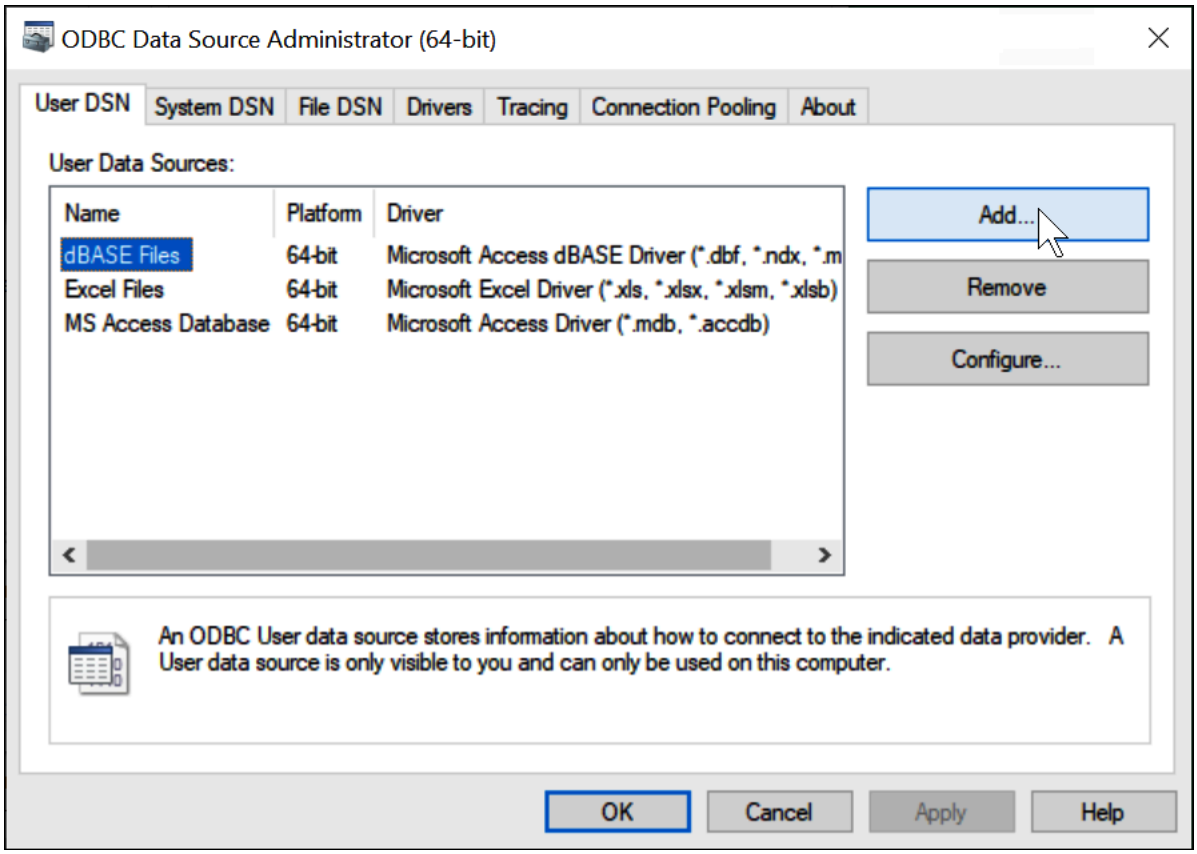
配置 Okta ODBC 到 Athena 的连接

您现在可以在 Windows 中使用 ODBC 数据源程序配置 Okta ODBC 到 Athena 的连接。

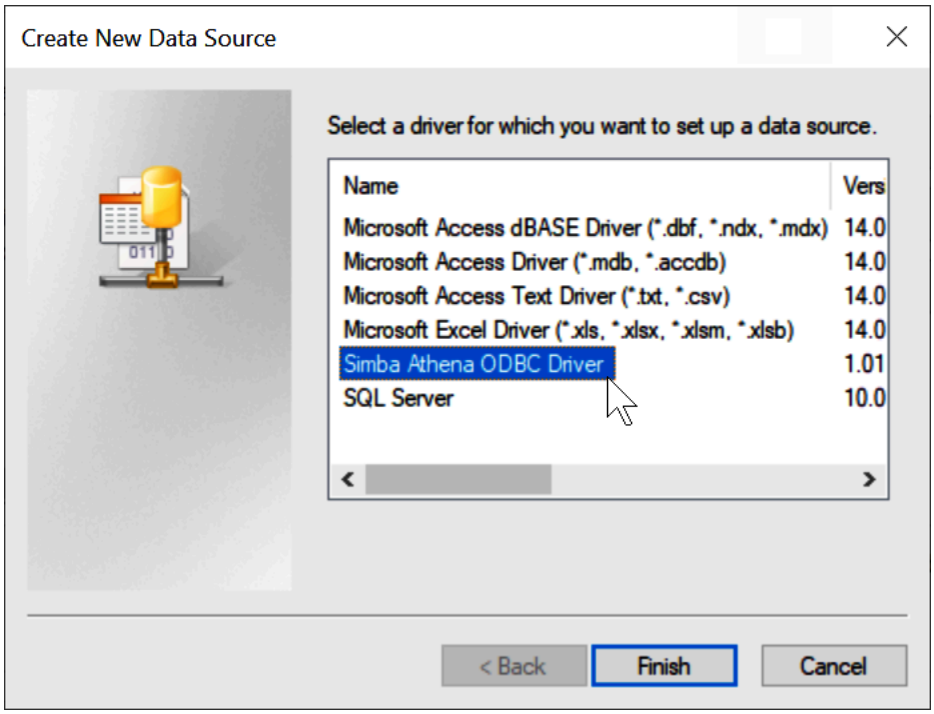
配置 Okta ODBC 到 Athena 的连接

1. 在 Windows 中，启动 ODBC 数据源程序。

2. 在 ODBC 数据源管理器程序中，选择 Add（添加）。

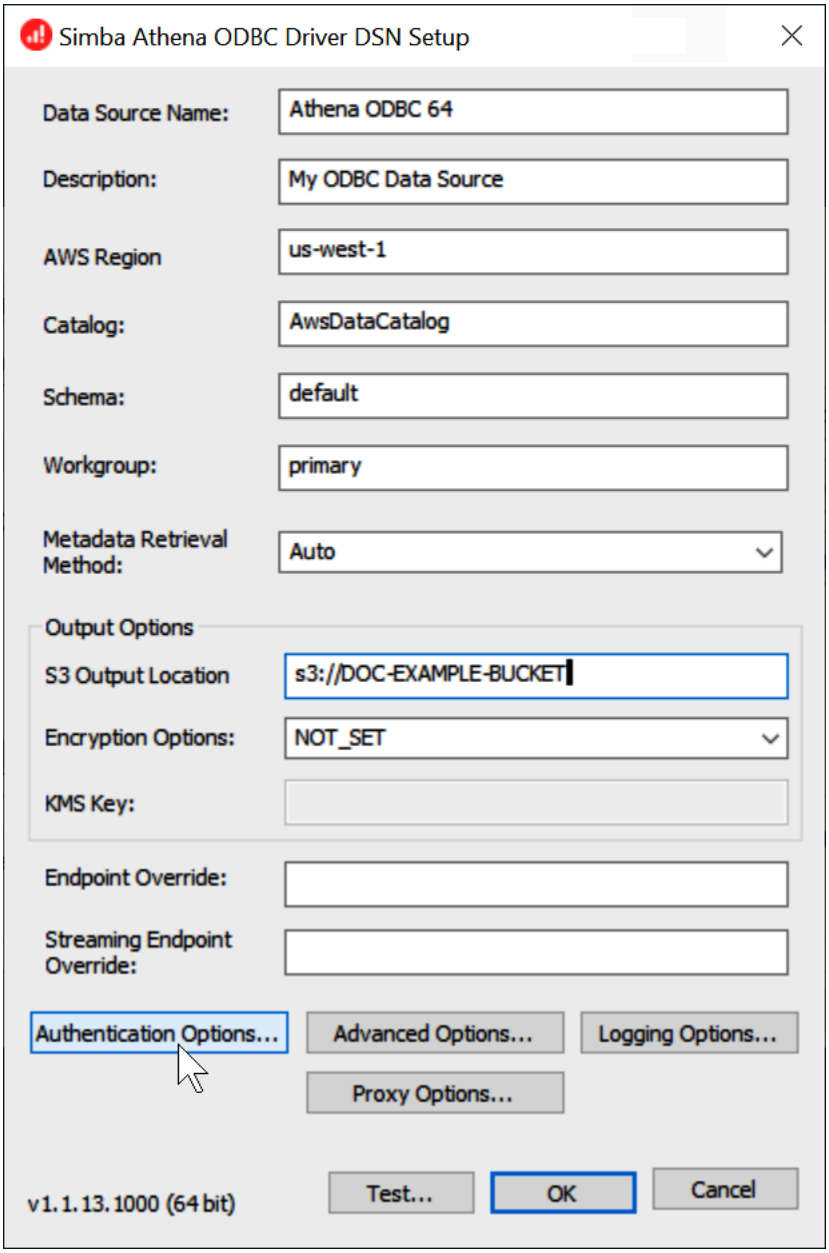


3. 选择 Simba Athena ODBC 驱动程序，然后选择 Finish（完成）。



4. 在 SIMBA Athena ODBC 驱动程序 DSN 设置对话框中，输入描述的值。

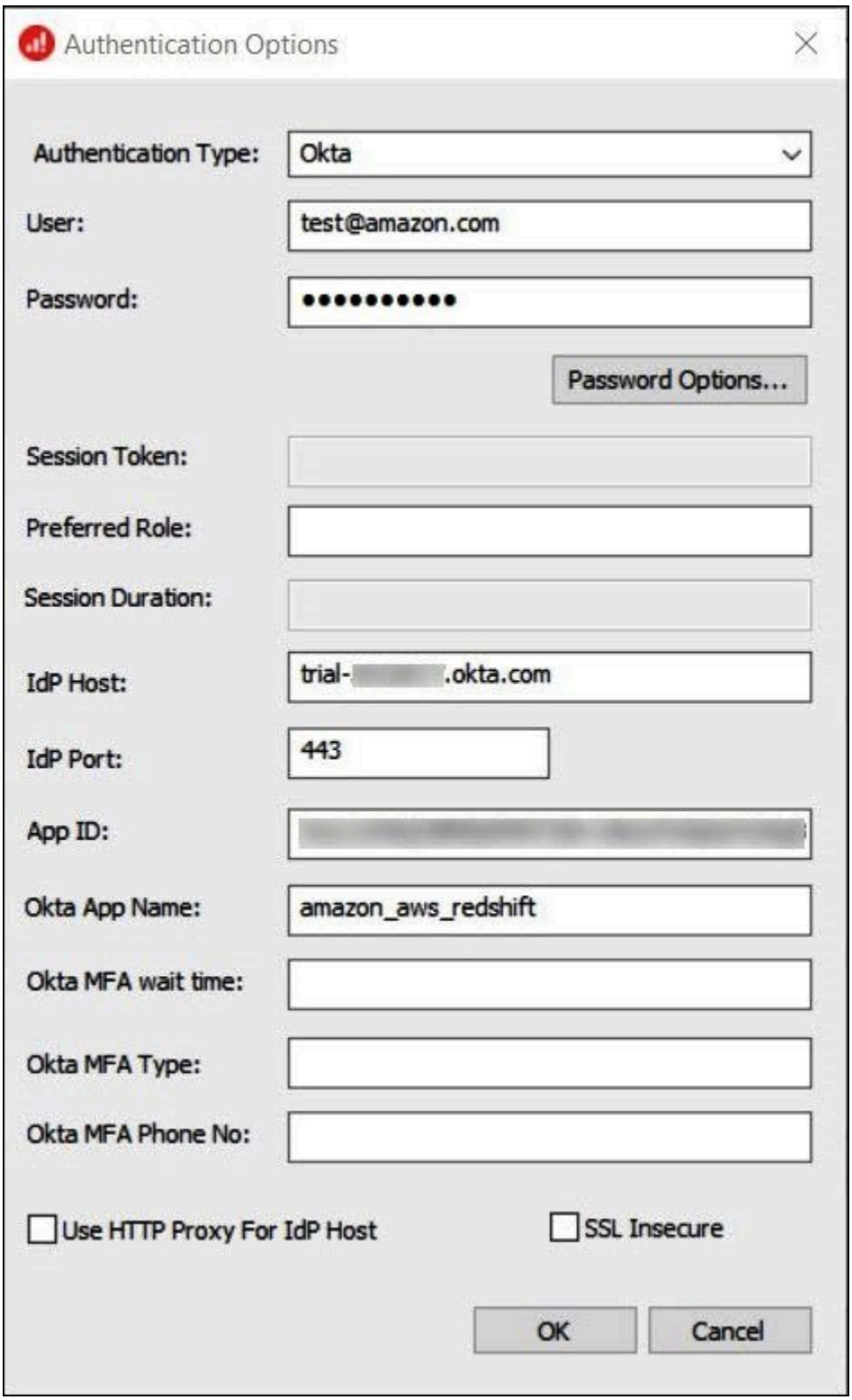
- 对于 Data Source Name（数据源名称），输入数据源的名称（例如，**`Athena ODBC 64`**）。
- 在 Description（说明）中，为数据来源输入说明。
- 对于 AWS 区域，输入您正在使用的 AWS 区域（例如 **`us-west-1`**）。
- 在 S3 Output Location（S3 输出位置）中，输入要存储输出的 Amazon S3 路径。



5. 选择 Authentication Options（身份验证选项）。
6. 在 Authentication Options（身份验证选项）对话框中，选择或输入以下值。

- 对于 Authentication Type（身份验证类型），选择 Okta。

- 对于用户，请输入您的 Okta 用户名。
- 对于密码，请输入您的 Okta 密码。
- 对于 IdP Host（IdP 主机），输入您之前记录的值（例如，**trial-1234567.okta.com**）。
- 对于 IdP Port（IdP 端口），输入 **443**。
- 对于 App ID（应用程序 ID），输入您之前记录的值（Okta 嵌入式链接的最后两个片段）。
- 对于 Okta App Name（Okta 应用程序名称），输入 **amazon_aws_redshift**。



7. 选择确定。
8. 选择 Test（测试）以测试连接，或者选择 OK（确定）以完成。

使用 ODBC、SAML 2.0 和 Okta 身份提供商配置单点登录

要连接到数据源，您可以将 Amazon Athena 与 PingOne、Okta、OneLogin 等身份提供商 (IdP) 一起使用。从 Athena ODBC 驱动程序版本 1.1.13 和 Athena JDBC 驱动程序版本 2.0.25 开始，驱动程序中都包括一个 SAML 浏览器插件，您可以配置此插件以与任何 SAML 2.0 提供商一起使用。本主题向您展示如何配置 Amazon Athena ODBC 驱动程序和基于浏览器的 SAML 插件，从而使用 Okta 身份提供商添加单点登录（SSO）功能。

先决条件

要完成本教程中的步骤，需要以下内容：

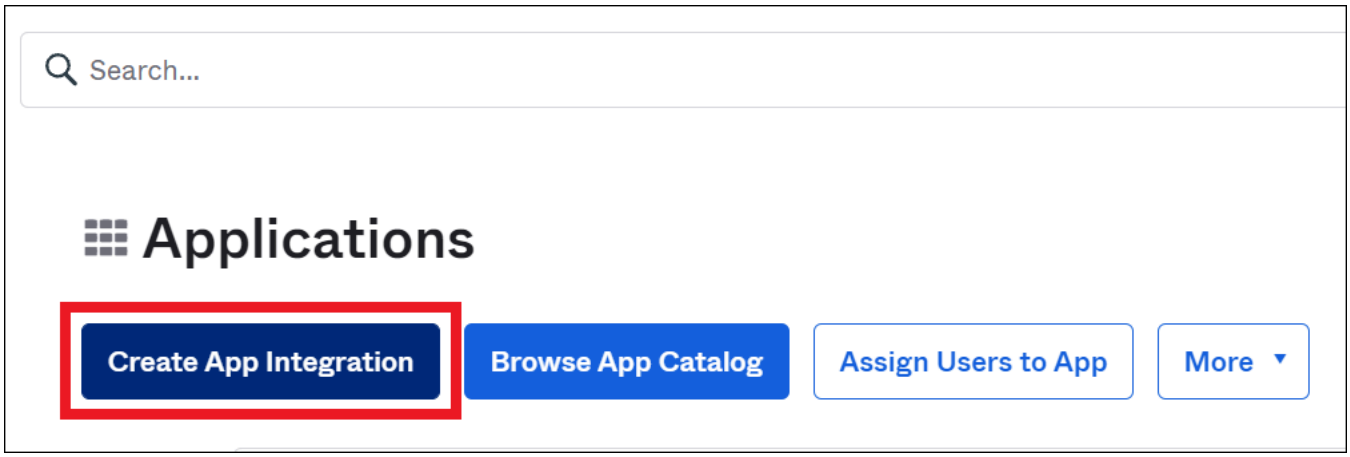
- Athena ODBC 驱动程序版本 1.1.13 或更高版本。支持浏览器 SAML 的1.1.13 及更高版本。有关下载链接，请参阅使用 ODBC 连接到 Amazon Athena。
- 要与 SAML 一起使用的 IAM 角色。有关更多信息，请参阅《IAM 用户指南》中的创建用于 SAML 2.0 联合身份验证的角色。
- Okta 账户。有关信息，请访问 okta.com。

在 Okta 中创建应用程序集成

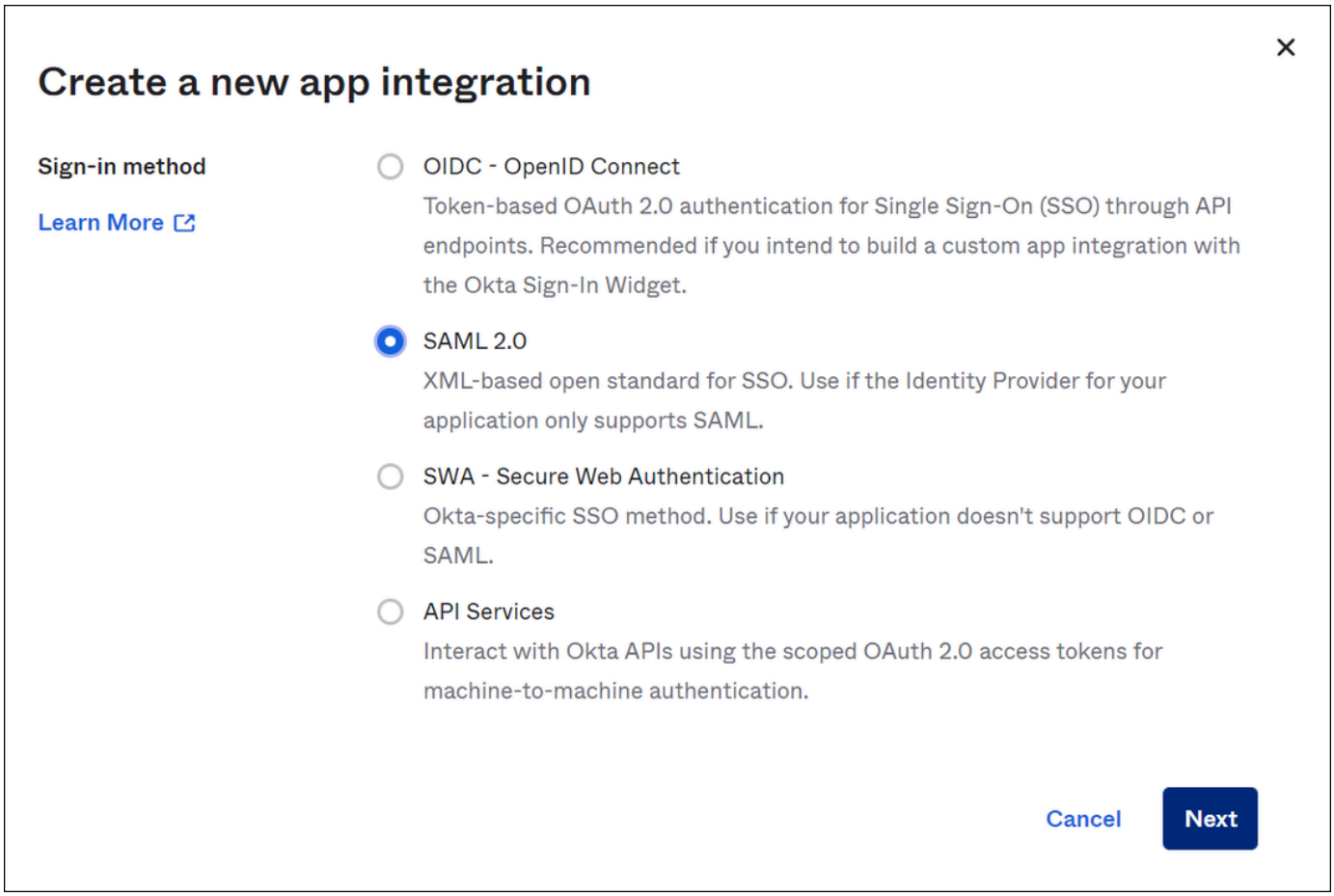
首先，使用 Okta 控制面板创建和配置 SAML 2.0 应用程序以单点登录 Athena。

使用 Okta 控制面板设置 Athena 的单点登录

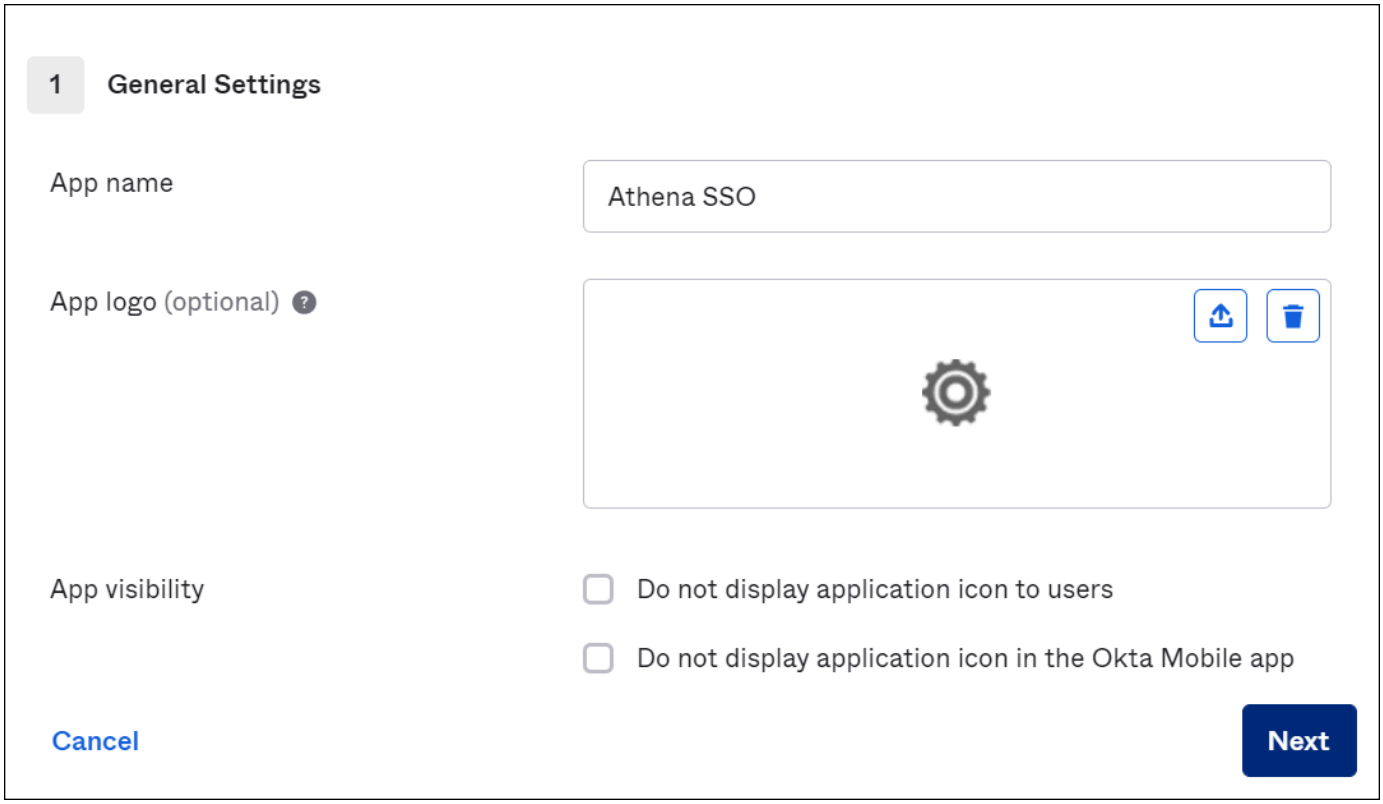
1. 在 `okta.com` 上登录到 Okta 管理员页面。
2. 在导航窗格中，选择 Application（应用程序），Application（应用程序）。
3. 在 Applications（应用程序）页面上，选择 Create App Integration（创建应用程序集成）。



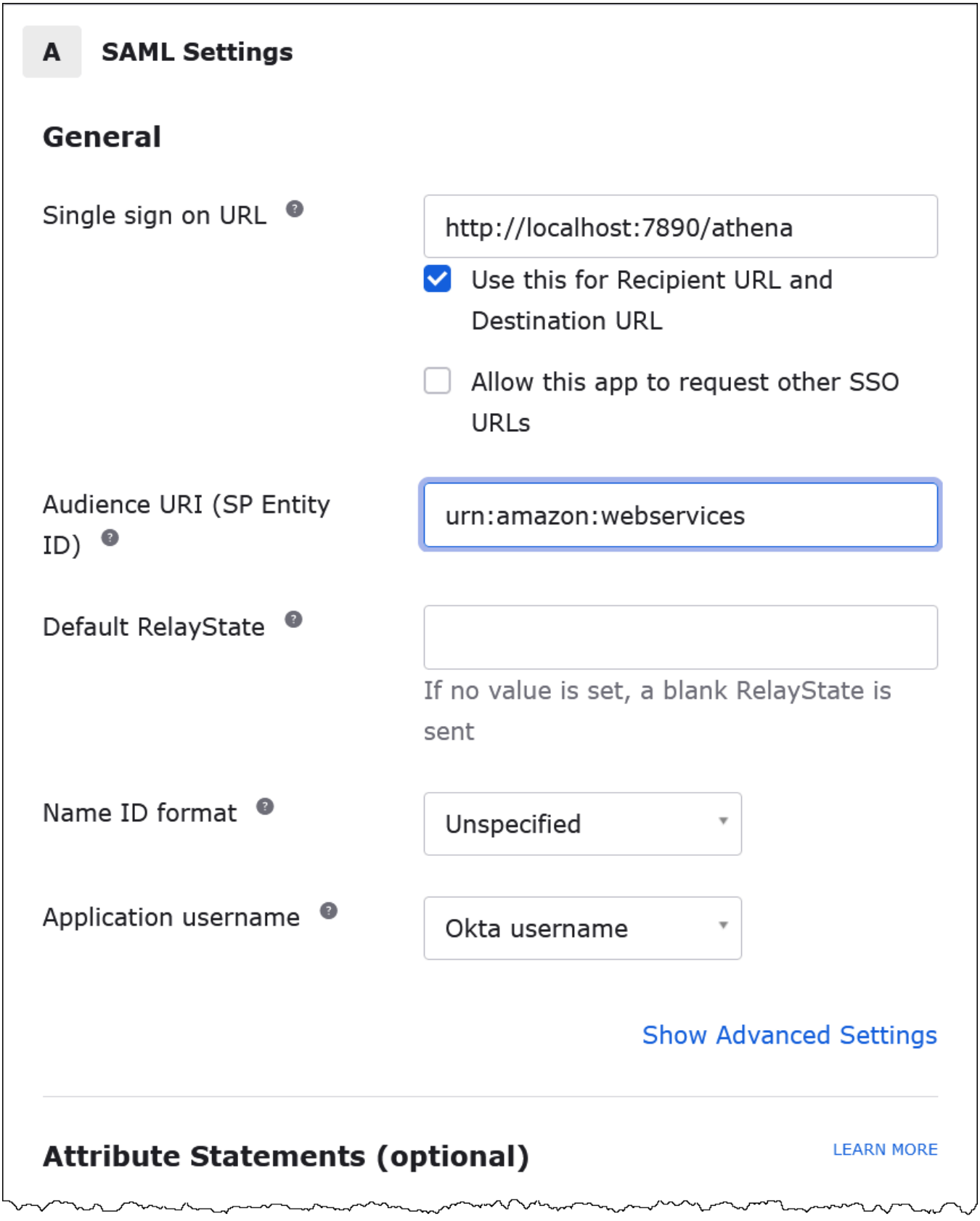
4. 在 Create a new app integration（创建新应用程序集成）对话框中，找到 Sign-in method（登录方法），选择 SAML 2.0，然后选择 Next（下一步）。



5. 在 Create SAML Integration（创建 SAML 集成）页面的 General Settings（常规设置）部分中，输入应用程序的名称。本教程使用名称 SSO Athena。



6. 选择下一步。
7. 在 Configure SAML（配置 SAML）页面的 SAML Settings（SAML 设置）部分，输入以下值：
   - 在 Single sign on URL（单点登录 URL）中，输入 **`http://localhost:7890/athena`**
   - 在 Audience URI（受众 URI）中，输入 **`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. 对于 Attribute Statements (optional)（属性语句（可选）），输入以下两个名称/值对。这些是必填的映射属性。

   - 在 Name（名称）中，输入以下 URL：

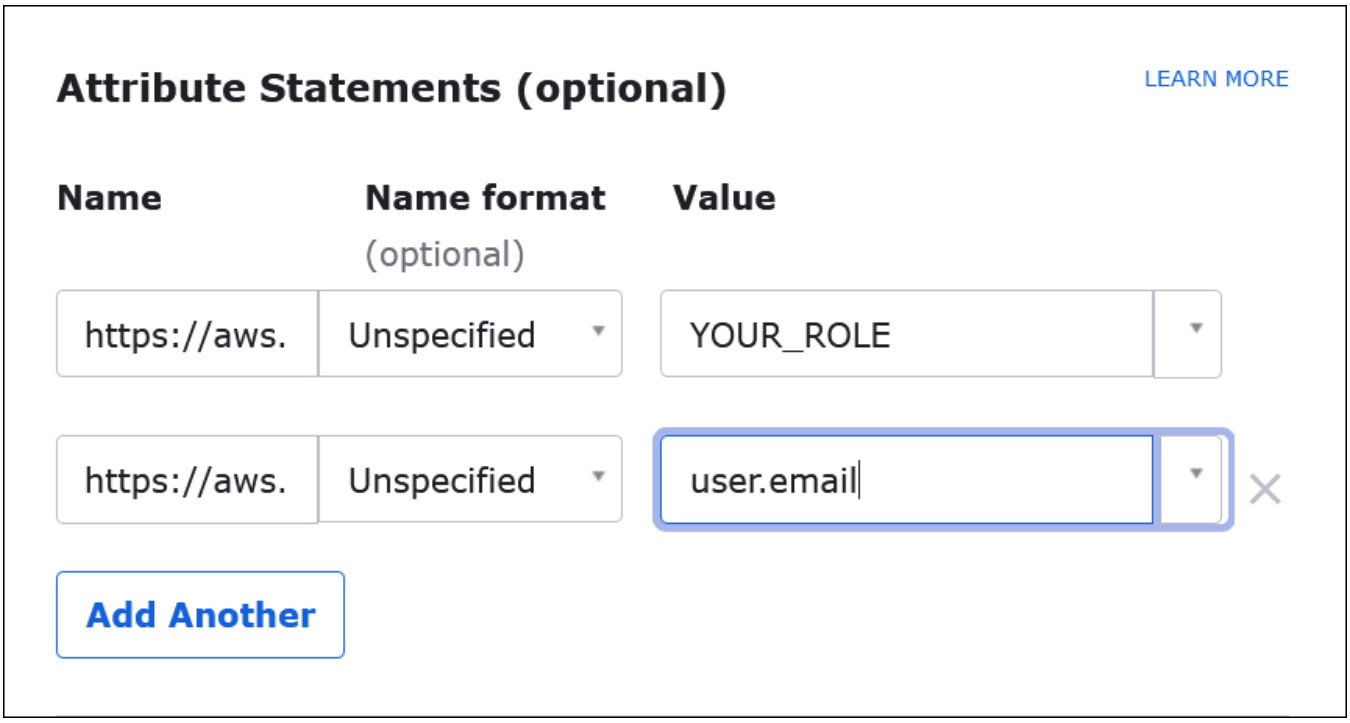
     `https://aws.amazon.com/SAML/Attributes/Role`

     在 Value（值）中，输入 IAM 角色的名称。有关 IAM 角色格式的信息，请参阅《IAM 用户指南》中的为身份验证响应配置 SAML 断言。

   - 在 Name（名称）中，输入以下 URL：

     `https://aws.amazon.com/SAML/Attributes/RoleSessionName`

     对于值，请输入 **`user.email`**。



9. 选择下一步，然后选择完成。

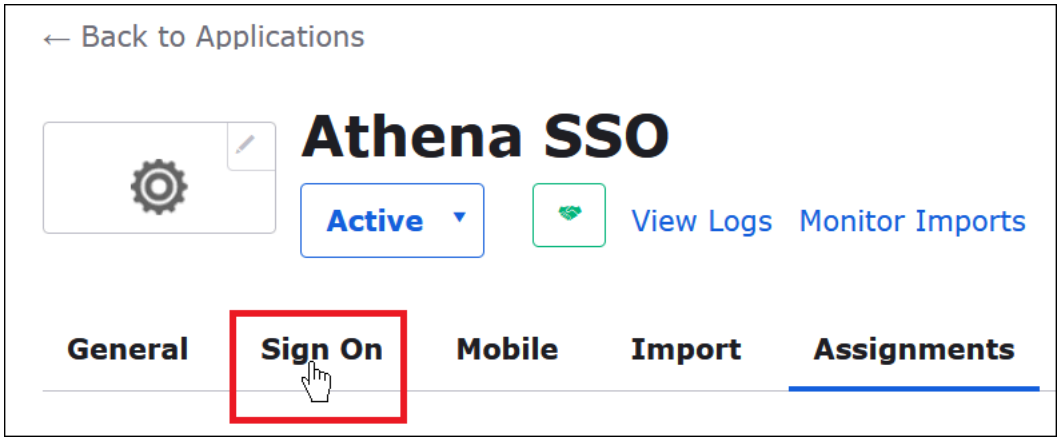
   当 Okta 创建应用程序时，它还会创建您的登录 URL，接下来您将检索该 URL。

从 Okta 控制面板获取登录 URL

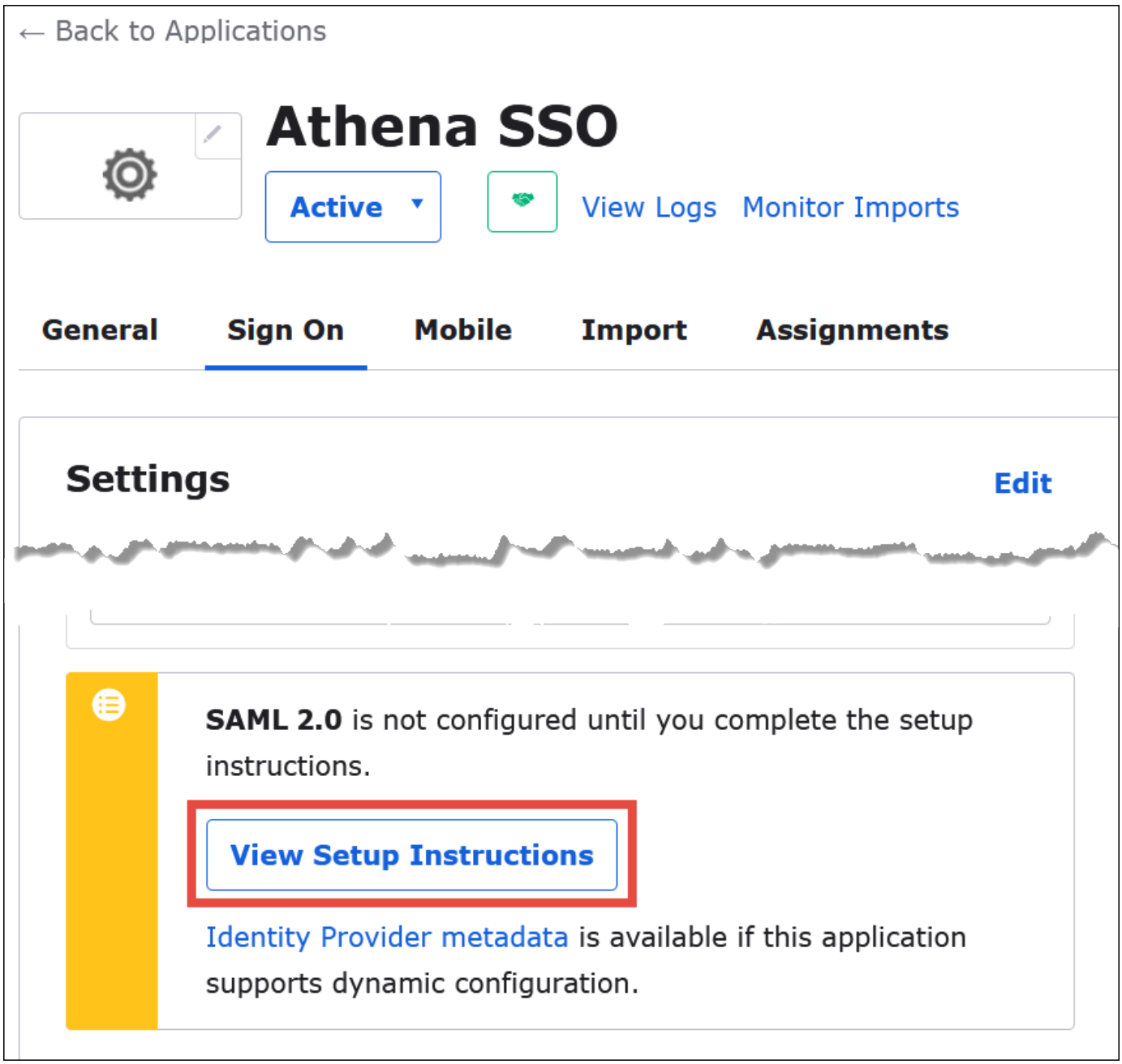
现在您的应用程序已经创建完毕，您可以从 Okta 控制面板获取其登录 URL 和其他元数据。

从 Okta 控制面板获取登录 URL

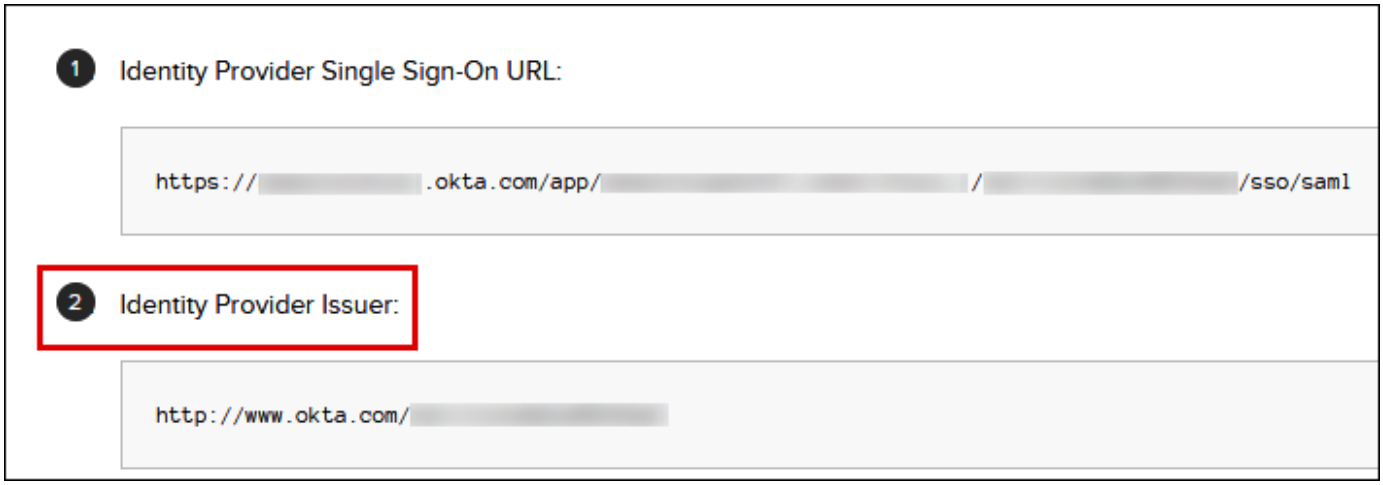
1. 在 Okta 导航窗格中，选择 Application（应用程序），Application（应用程序）。
2. 选择要查找登录 URL 的应用程序（例如，AthenNasso)。
3. 在应用程序页面上，选择 Sign On（登录）。



4. 选择 View Setup Instructions（查看设置说明）。



5. 在 How to Configure SAML 2.0 for Athena SSO（如何为 Athena SSO 配置 SAML 2.0）页面中，找到 Identity Provider Issuer（身份提供商发布者）的 URL。Okta 控制面板中的某些地方会将此 URL 称为 SAML 发布者 ID。



6. 复制或存储 Identity Provider Single Sign-On URL（身份提供商单点登录 URL）的值。

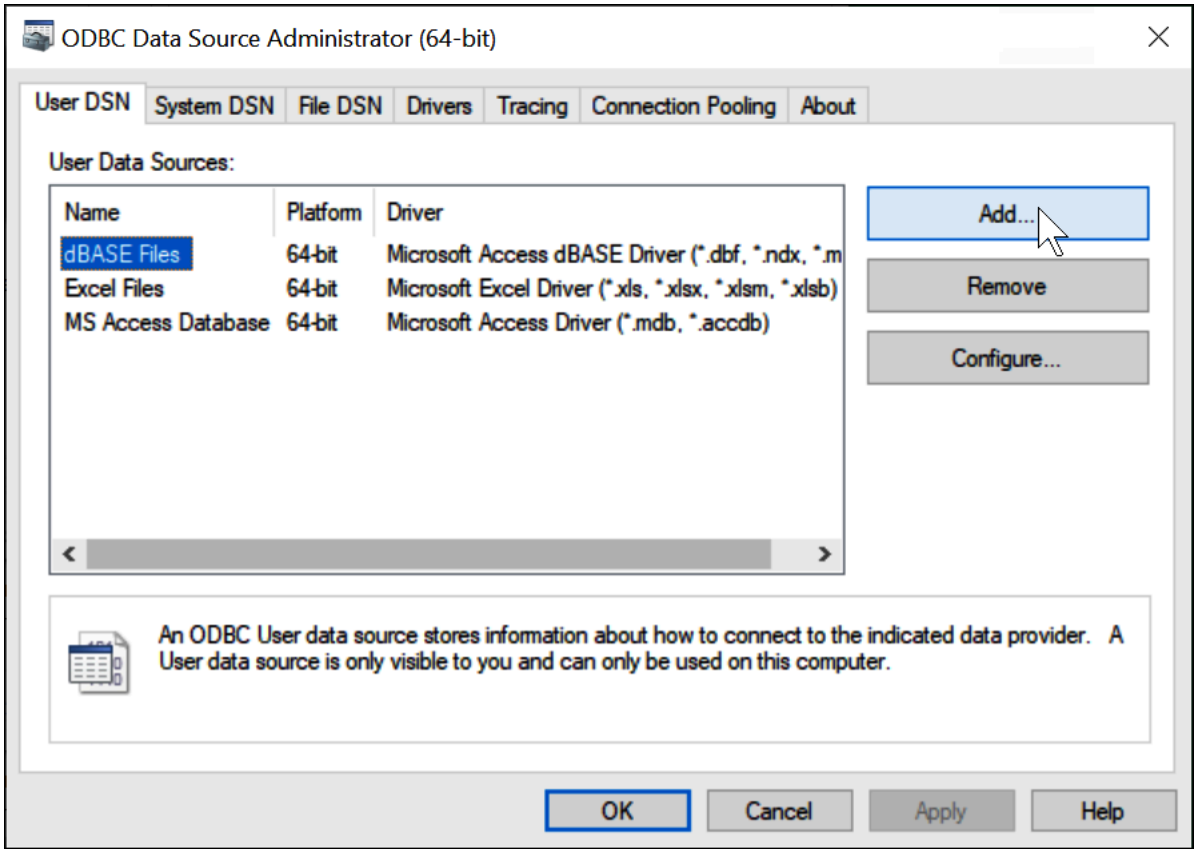
   在下一节中，配置 ODBC 连接时，您将提供此值作为 SAML 浏览器插件的登录 URL 连接参数。

配置浏览器 SAML ODBC 与 Athena 的连接

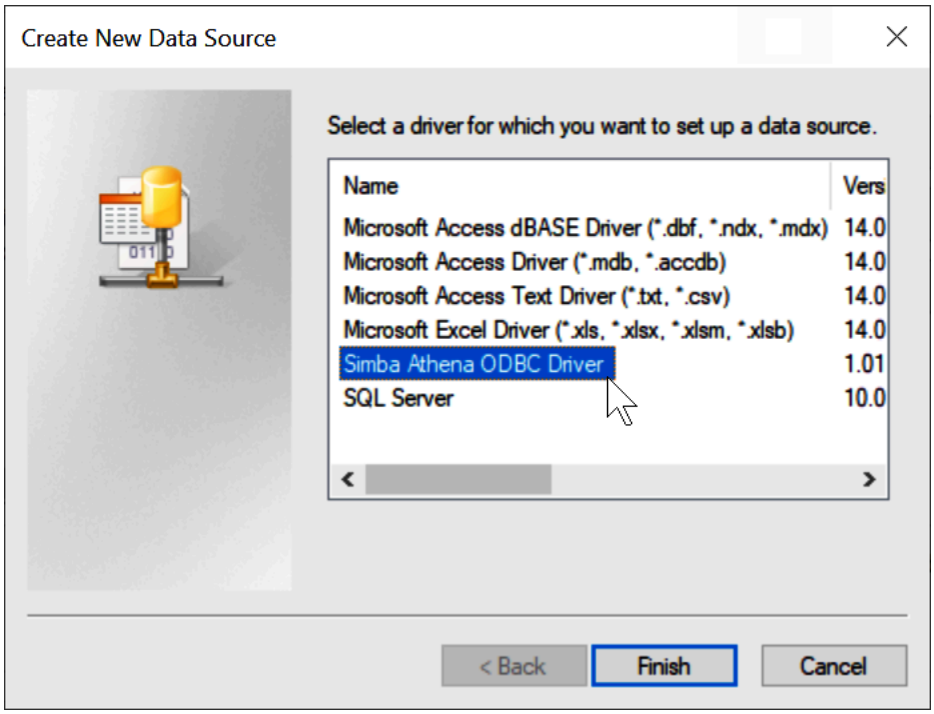
现在，您可以在 Windows 中使用 ODBC 数据源程序配置浏览器 SAML 到 Athena 的连接。

配置浏览器 SAML ODBC 到 Athena 的连接

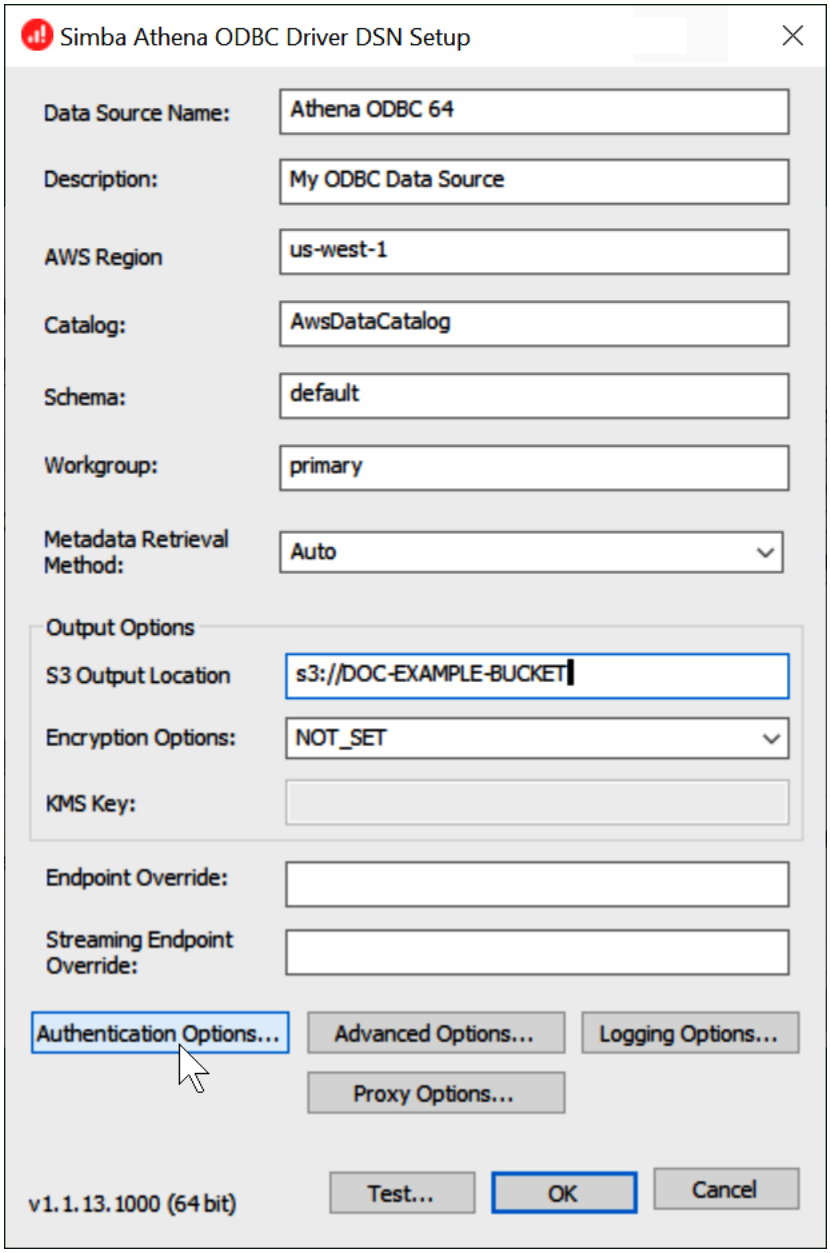
1. 在 Windows 中，启动 ODBC 数据源程序。
2. 在 ODBC 数据源管理器程序中，选择 Add（添加）。



3. 选择 Simba Athena ODBC 驱动程序，然后选择 Finish（完成）。

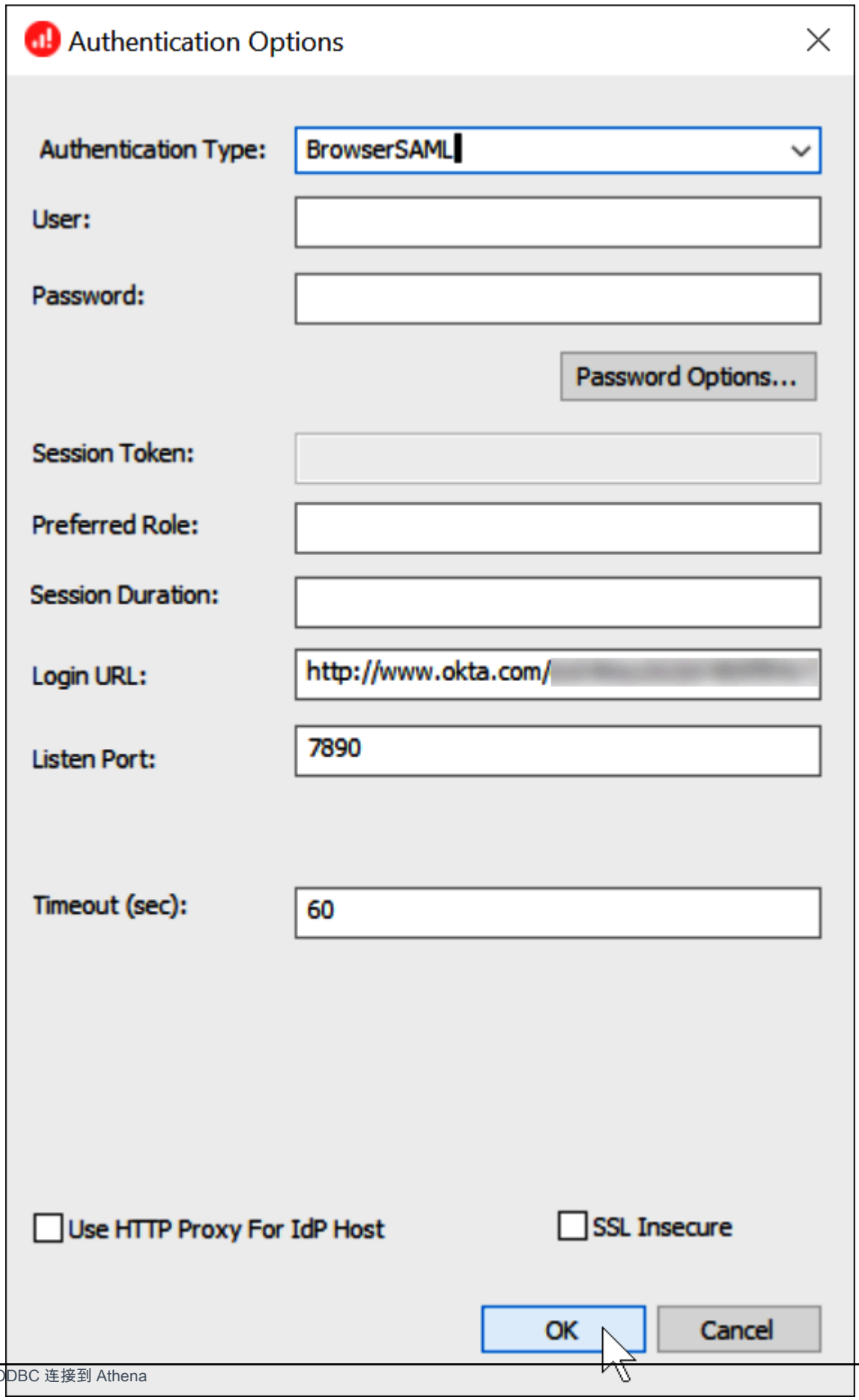


4. 在 SIMBA Athena ODBC 驱动程序 DSN 设置对话框中，输入描述的值。



- 在 Data Source Name（数据源名称）中，输入数据源的名称（例如，64 Athena ODBC）。
- 在 Description（说明）中，为数据源输入说明。
- 在 AWS 区域 中，输入您正在使用的 AWS 区域（例如 **us-west-1**）。
- 在 S3 Output Location（S3 输出位置）中，输入要存储输出的 Amazon S3 路径。

5. 选择 Authentication Options（身份验证选项）。
6. 在 Authentication Options（身份验证选项）对话框中，选择或输入以下值。


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

- 在 Authentication Type（身份验证类型）中，选择 BrowserSAML。
- 在 Login URL（登录 URL）中，输入您从 Okta 控制面板中获得的 Identity Provider Single Sign-On URL（身份提供商单点登录 URL）。
- 对于 Listen Port（侦听端口），请输入 7890。
- 对于 Timeout (sec)（超时（秒）），请输入连接超时的秒数。

7. 选择 OK（确定）以关闭 Authentication Options（身份验证选项）。
8. 选择 Test（测试）以测试连接，或者 OK（确定）以完成。

使用 Amazon Athena Power BI 连接器

在 Windows 操作系统上，您可以使用适用于 Amazon Athena 的 Microsoft Power BI 连接器来分析来自 Microsoft Power BI 桌面中的 Amazon Athena 的数据。有关 Power BI 的信息，请参阅 Microsoft Power BI。将内容发布到 Power BI 服务后，可以使用 2021 年 7 月或更新版本的 Power BI 网关，通过按需刷新或计划刷新使内容保持最新状态。

先决条件

在开始使用之前，请确保您的环境满足以下要求。必须具备 Amazon Athena ODBC 驱动程序。

- AWS 账户
- 使用 Athena 的权限
- Amazon Athena ODBC 驱动程序
- Power BI 桌面

支持的功能

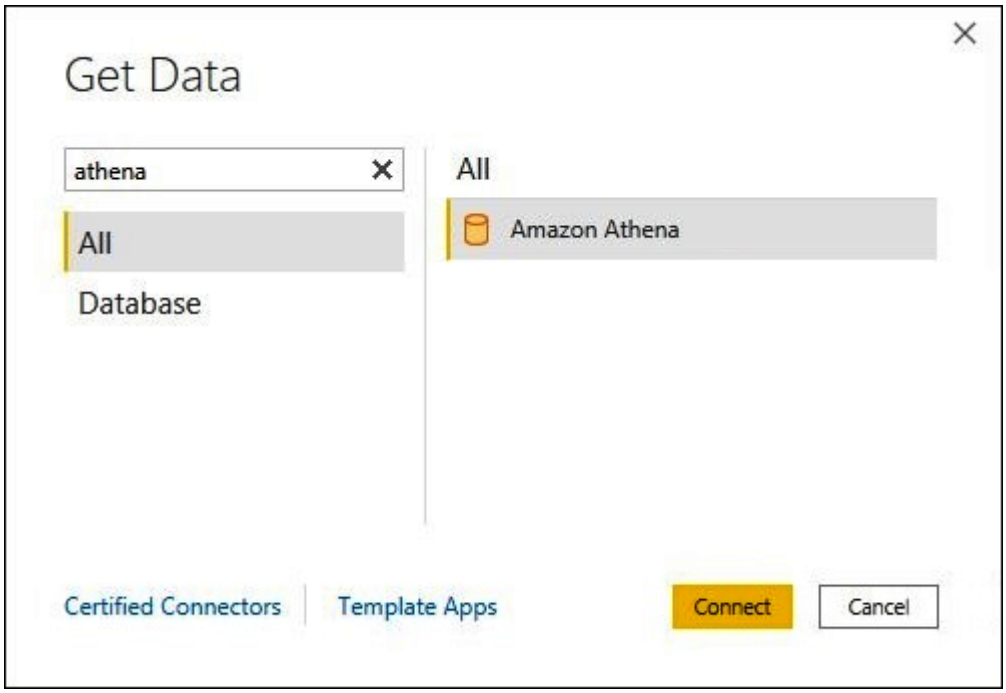
- 导入 – 选定的表和列将导入到 Power BI 桌面进行查询。
- DirectQuery – 不会将任何数据导入或复制到 Power BI 桌面。Power BI 桌面将直接查询底层数据源。
- Power BI 网关 – 您的 AWS 账户 中的本地部署数据网关，其职责类似于在 Microsoft Power BI 服务和 Athena 之间搭建桥梁。网关需要查看您在 Microsoft Power BI 服务上的数据。

连接到 Amazon Athena

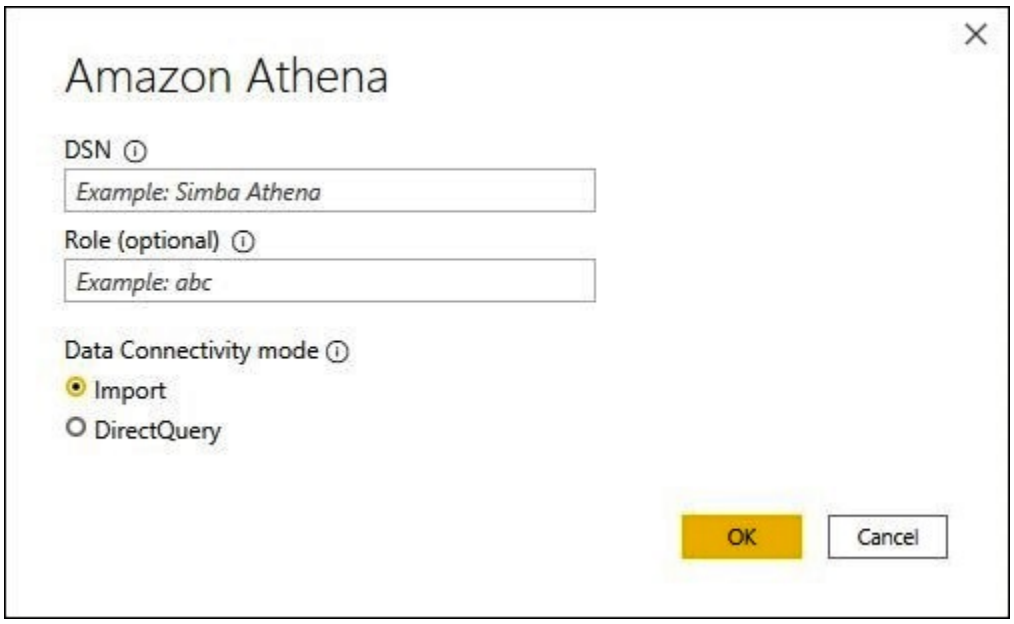
要将 Power BI 桌面连接到 Amazon Athena 数据，请执行以下步骤。

从 Power BI 桌面连接到 Athena 数据

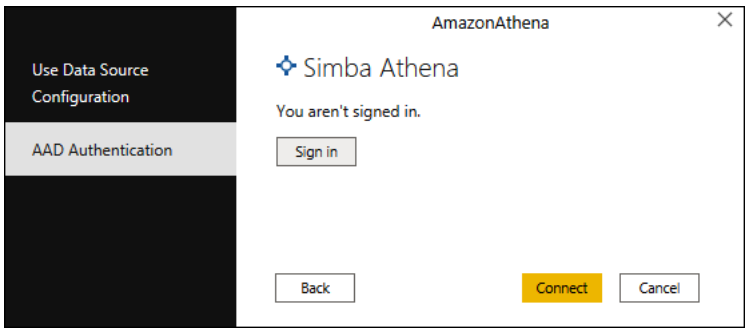
1. 启动 Power BI 桌面。
2. 请执行以下操作之一：
   - 选择 File（文件）、Get Data（获取数据）
   - 从 Home（主页）功能区中，选择 Get Data（获取数据）。
3. 在搜索框中，输入 Athena。
4. 选择 Amazon Athena，然后选择 Connect（连接）。



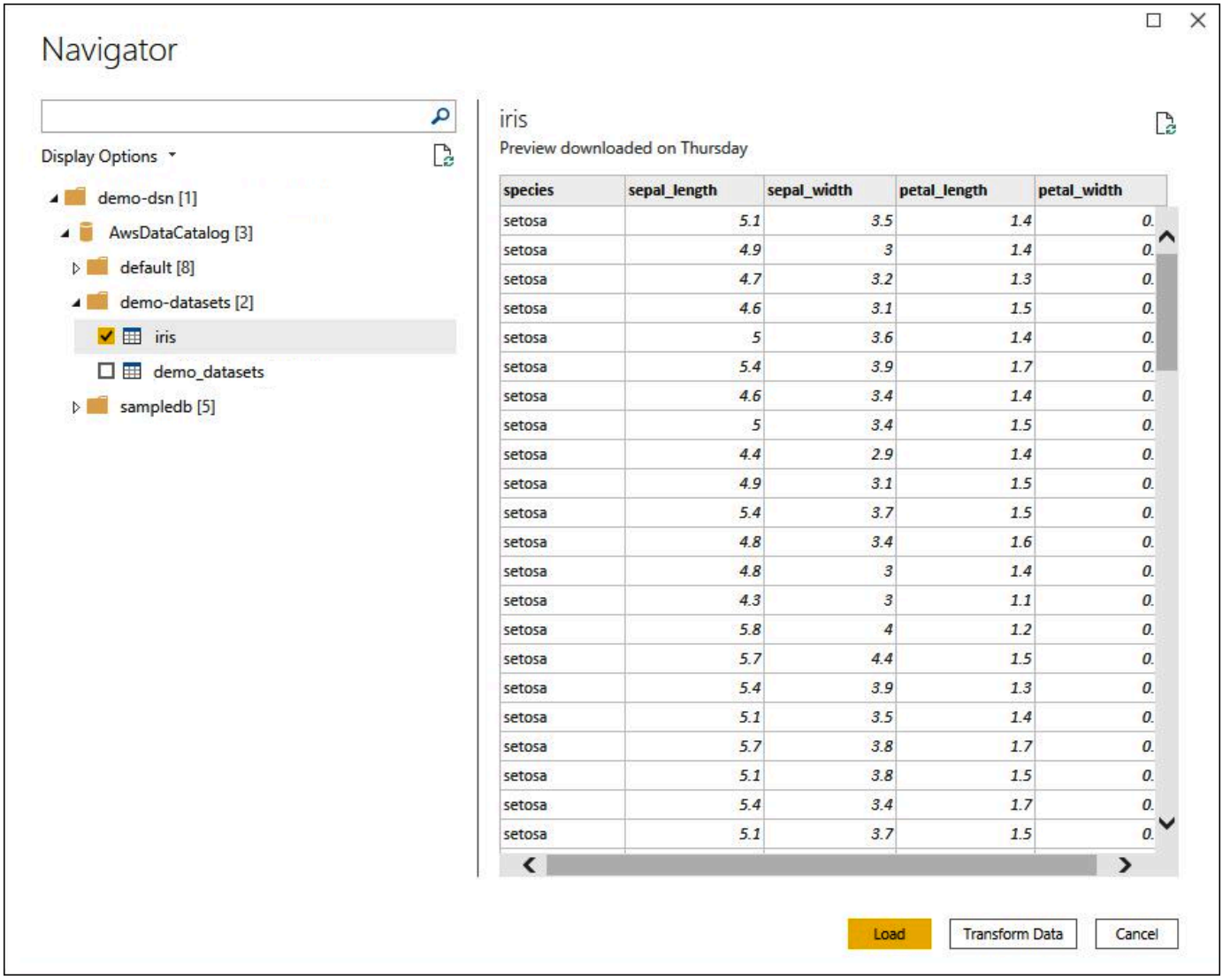
5. 在 Amazon Athena连接器页面上，输入以下信息。
   - 对于 DSN，输入要使用的 ODBC DSN 的名称。有关配置 DSN 的说明，请参阅 ODBC 驱动程序文档。
   - 对于 Data Connectivity mode（数据连接模式），选择适合您的使用案例的模式，并遵循以下常规指南：
     - 对于较小的数据集，请选择 Import（导入）。使用“导入”模式时，Power BI 与 Athena 协作导入整个数据集的内容，以便用于可视化操作。
     - 对于较大的数据集，请选择 DirectQuery。在 DirectQuery 模式下，不会将任何数据下载到您的工作站。在创建可视化或与可视化进行交互时，Microsoft Power BI 与 Athena 一起动态查询基础数据源，以便您始终都能查看当前数据。有关 DirectQuery 的更多信息，请参阅 Microsoft 文档中的在 Power BI 桌面中使用 DirectQuery。



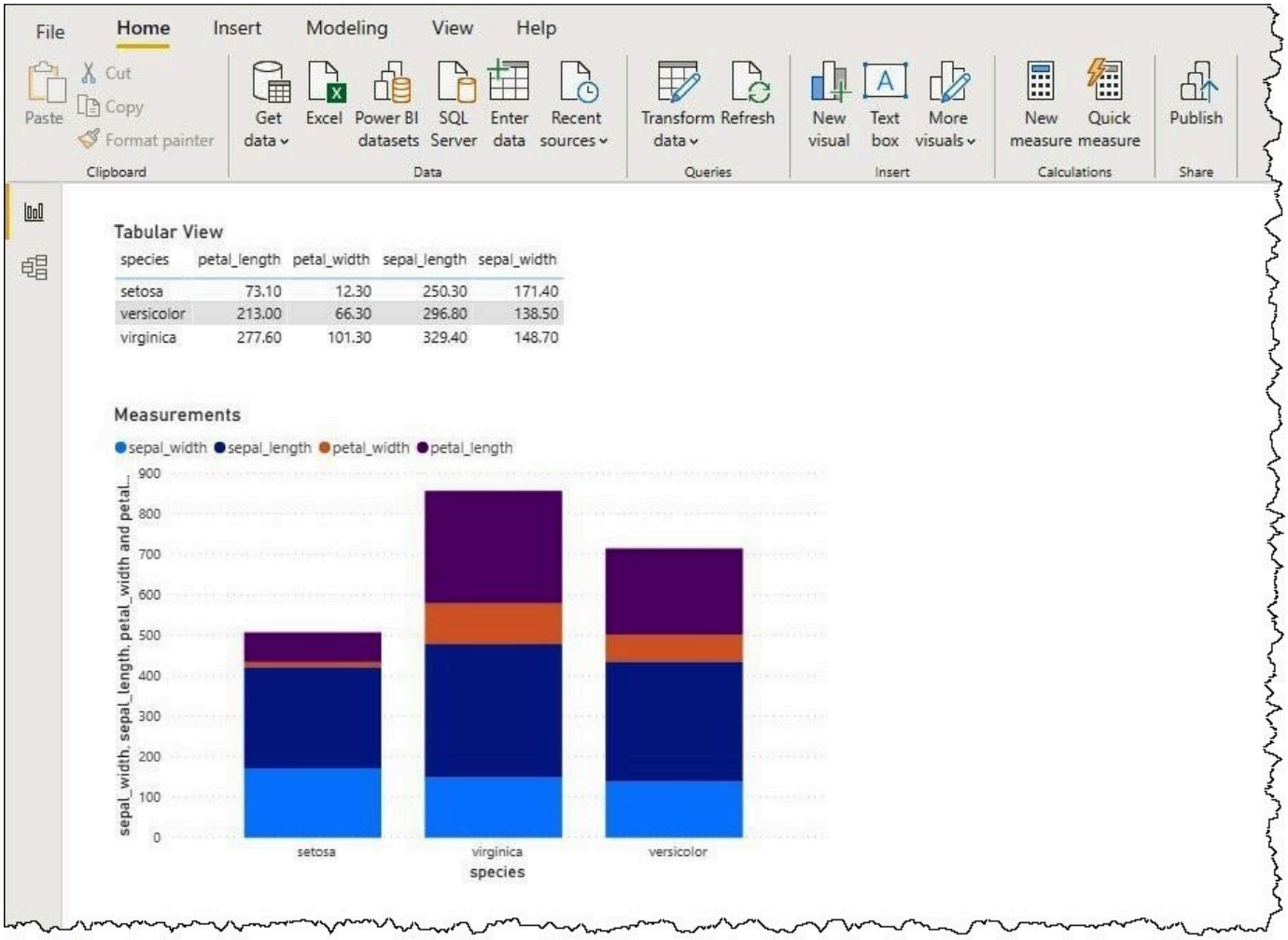
6. 选择确定。
7. 在提示配置数据源身份验证时，选择 Use Data Source Configuration（使用数据源配置）或者 AAD Authentication（AAD 身份验证），然后选择 Connect（连接）。



您的数据目录、数据库和表将显示在 Navigator（导航）对话框。



8. 在 Display Options（显示选项）窗格中，为数据集选中要使用的复选框。
9. 如果要在导入数据集之前对其进行转换，请转到对话框底部，然后选择 Transform Data（转换数据）。这将打开 Power 查询编辑器，以便您筛选和优化要使用的数据集。
10. 选择 Load（加载）。加载完成后，您可以创建类似于以下图像中的可视化。如果您选择 DirectQuery 作为导入模式，Power BI 会向 Athena 发出查询，以用于您请求的可视化效果。



设置本地部署网关

您可以将控制面板和数据集发布到 Power BI 服务，以便其他用户可以通过 Web、移动端和嵌入式应用与它们进行交互。若要在 Microsoft Power BI 服务中查看您的数据，请将 Microsoft Power BI 本地部署数据网关安装在 AWS 账户 中。该网关的职责类似于在 Microsoft Power BI 服务和 Athena 之间搭建桥梁。

要下载、安装和测试本地部署数据网关

1. 访问 Microsoft Power BI 网关下载页面，然后选择个人模式或标准模式。个人模式对于在本地测试 Athena 连接器非常有用。标准模式适用于多用户生产设置。
2. 要安装本地部署网关（个人模式或标准模式），请参阅 Microsoft 文档中的安装本地部署数据网关。
3. 要测试网关，请按照 Microsoft 文档中的将自定义数据连接器与本地部署数据网关结合使用。

有关本地部署数据网关的更多信息，请参阅以下 Microsoft 资源。

- 什么是本地部署数据网关？
- 为 Power BI 部署数据网关的指南

有关配置 Power BI 网关以与 Athena 一起使用的示例，请参阅 AWS 大数据博客文章：使用 Amazon Athena 在 Microsoft Power BI 上快速创建控制面板。

创建数据库和表

Amazon Athena 支持一部分数据定义语言 (DDL) 语句以及 ANSI SQL 函数和运算符，用以定义和查询数据在 Amazon Simple Storage Service 中的外部表。

当您在 Athena 中创建数据库和表时，您需要描述数据的架构和位置，使表中的数据为实时查询做好准备。

为了提高查询性能并降低成本，我们建议您对数据进行分区，并使用开源列式格式存储在 Simple Storage Service (Amazon S3) 中，如 Apache Parquet 或 ORC。

主题

- 在 Athena 中创建数据库
- 在 Athena 中创建表
- 表、数据库和列的名称
- 保留关键字
- Amazon S3 中的表位置
- 列式存储格式
- 转换为列式格式
- 在 Athena 中对数据进行分区
- 使用 Amazon Athena 分区投影

在 Athena 中创建数据库

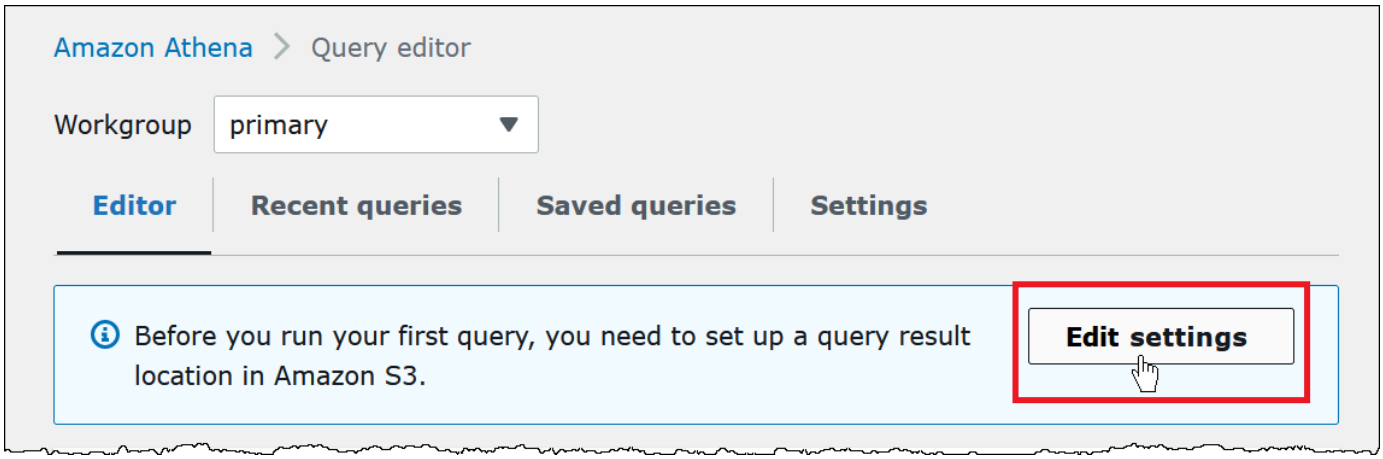
Athena 中的数据库是您在其中创建的表的逻辑分组。

先决条件

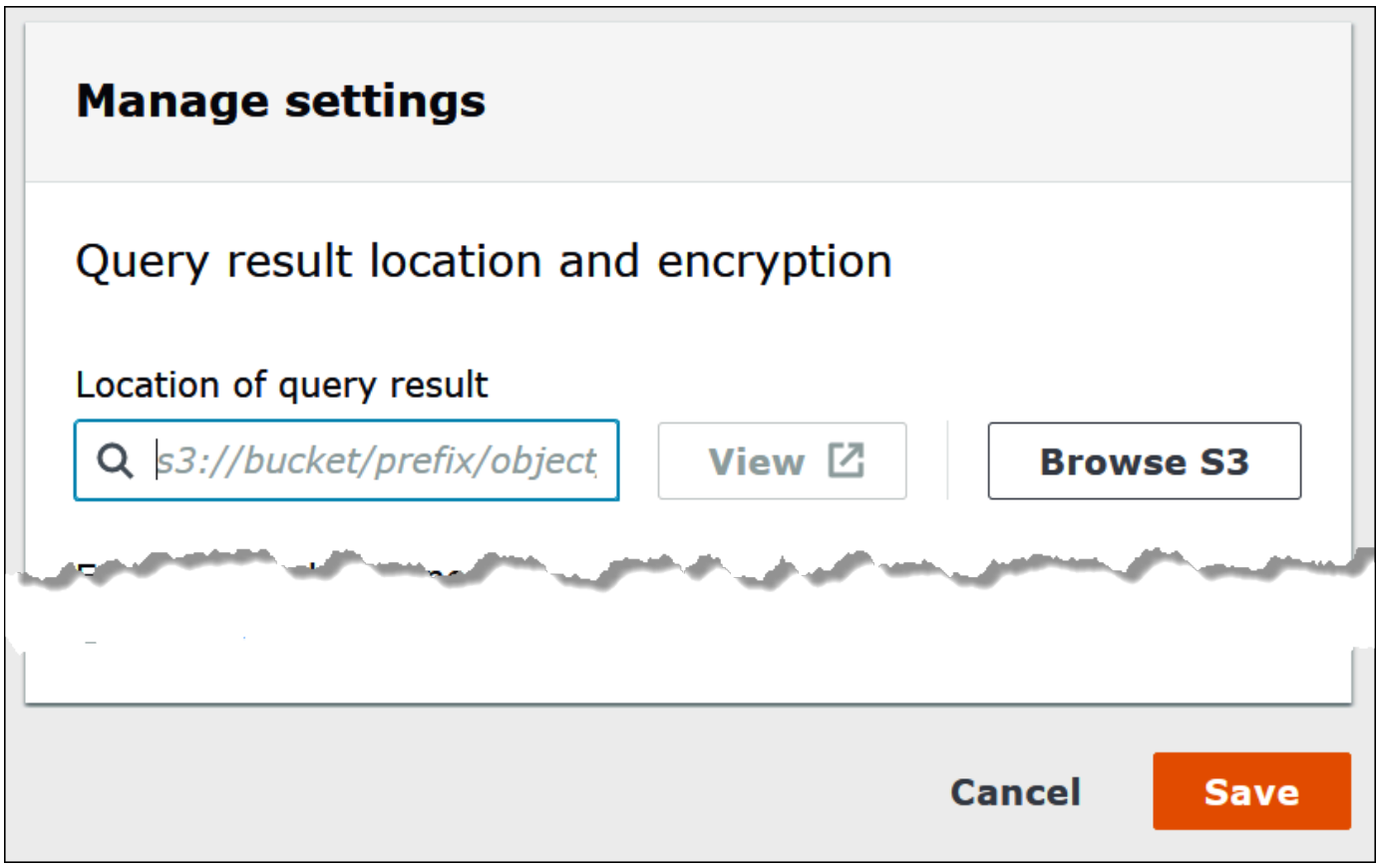
如果您尚未在 Amazon S3 中设置查询输出位置，执行以下先决条件步骤进行设置。

创建查询输出位置

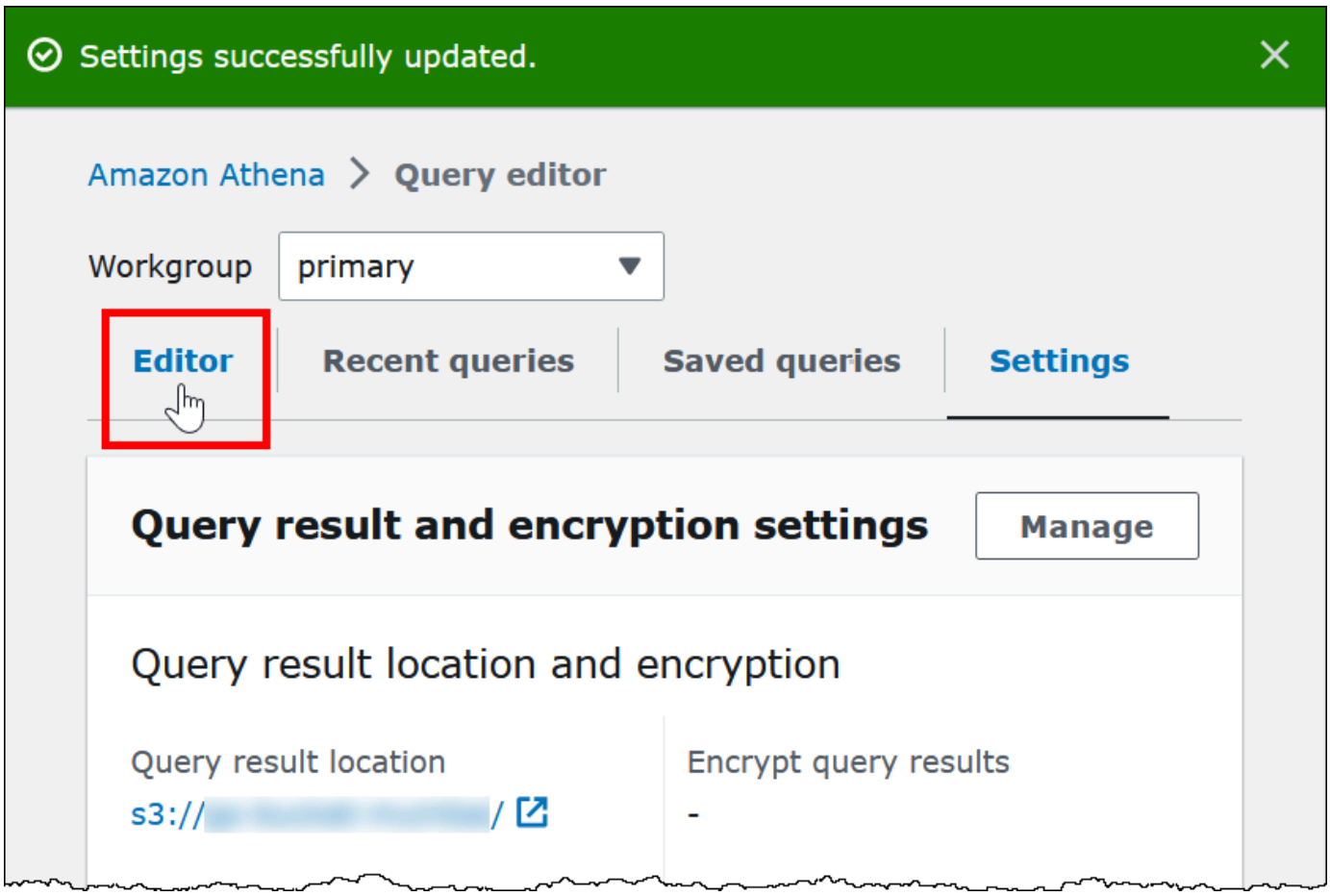
1. 使用在 Athena 中使用的 AWS 区域 和账户，按照相应的步骤（例如，通过使用 Amazon S3 控制台）在 Amazon S3 中创建存储桶以保存您的 Athena 查询结果。您需要将此存储桶配置为查询输出位置。
2. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
3. 如果这是您首次在此 AWS 区域 中访问 Athena 控制台，选择浏览查询编辑器打开查询编辑器。否则，Athena 会在查询编辑器中打开。
4. 选择 Edit Settings（编辑设置）以在 Amazon S3 中设置查询结果位置。



5. 对于 Manage settings（管理设置），执行以下操作之一：
   - 在 Location of query result（查询结果位置）文本框中，输入您在 Amazon S3 中为查询结果创建的存储桶路径。在路径前添加前缀 `s3://`。
   - 选择 Browse S3（浏览 S3），选择您为当前区域创建的 Amazon S3 存储桶，然后选择 Choose（选择）。



6. 选择保存。
7. 选择 Editor（编辑器）以切换到查询编辑器。



创建数据库

在设置查询结果位置后，即可在 Athena 控制台查询编辑器中轻松创建数据库。

使用 Athena 查询编辑器创建数据库

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 在编辑器选项卡的查询编辑器中，输入 Hive 数据定义语言（DDL）命令 `CREATE DATABASE myDataBase`。将 *myDatabase* 替换为要使用的名称。有关数据库名称的限制，请参阅 表、数据库和列的名称。
3. 选择 Run（运行）或者按 **Ctrl+ENTER**。
4. 要将数据库设为当前数据库，请从查询编辑器左侧的 Database（数据库）菜单中选择该数据库。

有关控制 Athena 数据库权限的信息，请参阅 针对 AWS Glue Data Catalog 中数据库和表的精细访问权限。

在 Athena 中创建表

您可以在 Athena 控制台中、使用 JDBC 或 ODBC 驱动程序，或使用 Athena Create table（创建表）表单运行 DDL 语句。

当您在 Athena 中创建新表架构时，Athena 会将架构存储在数据目录中并在您运行查询时使用它。

Athena 采用称作基于读取的架构的方法，这意味着，架构将在您运行查询时投影到您的数据上。这样就无需加载或转换数据。

Athena 不修改您在 Amazon S3 中存储的数据。

Athena 使用 Apache Hive 定义表和创建在本质上为表的逻辑命名空间的数据库。

当您在 Athena 中创建数据库和表时，只需描述架构和表数据在 Amazon S3 中的位置，以便进行读取时查询。因此，数据库和表与传统关系数据库系统的含义稍有不同，因为数据不与数据库和表的架构定义一起存储。

当您查询时，您会使用标准 SQL 来查询表，此时将读取数据。您可以使用 Apache Hive 文档查找有关如何创建数据库和表的指导，但下面提供专门针对 Athena 的指导。

最大查询字符串长度为 256 KB。

Hive 通过使用串行器/解串器 (SerDe) 库来支持多种数据格式。您还可以使用正则表达式来定义复杂架构。有关受支持的 SerDe 库的列表，请参阅支持的 SerDes 和数据格式。

注意事项和限制

以下是 Athena 中表的一些重要限制和注意事项。

对 Athena 中的数据和 Simple Storage Service (Amazon S3) 中的表的要求

创建表时，请使用 `LOCATION` 子句指定底层数据在 Amazon S3 存储桶中的位置。请考虑以下事项：

- Athena 只能在版本化的 Amazon S3 存储桶中查询最新版本的数据，并且无法查询以前版本的数据。
- 您必须拥有合适的权限才能使用 Amazon S3 位置的数据。有关更多信息，请参阅 对 Amazon S3 的访问权限。
- Athena 支持查询使用多个存储类存储在由 `LOCATION` 子句指定的相同存储桶中的对象。例如，您可以查询存储在 Amazon S3 中不同存储类（标准、标准 IA 和智能分层）的对象中的数据。

- Athena 支持申请方付款存储桶。要了解如何为包含您计划在 Athena 中查询的源数据的存储桶启用申请方付款，请参阅创建工作组。
- Athena 不支持查询 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类中的数据。已忽略 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 存储类中的对象。作为替代方法，您可以使用 Amazon S3 Glacier Instant Retrieval 存储类，Athena 可以查询该存储类。有关更多信息，请参阅 Amazon S3 Glacier Instant Retrieval 存储类。

  有关存储类的信息，请参阅《Amazon Simple Storage Service 用户指南》中的 存储类、更改 Simple Storage Service (Amazon S3) 中的对象的存储类、转换为 GLACIER 存储类（对象归档）和申请方付款存储桶。
- 如果您针对包含大量对象且数据未分区的 Amazon S3 存储桶发出查询，则此类查询可能影响 Amazon S3 中的 Get 请求速率限制并导致 Amazon S3 异常。为防止错误发生，请将数据分区。另外，考虑调整 Amazon S3 的请求速率。有关更多信息，请参阅请求速率和性能注意事项。
- 如果您将 AWS Glue CreateTable API 操作或 AWS CloudFormation `AWS::Glue::Table` 模板创建用于 Athena 的表，而不指定 `TableType` 属性，然后运行 DDL 查询，如 `SHOW CREATE TABLE` 或者 `MSCK REPAIR TABLE`，则您将收到错误消息失败：NullPointerException 名称为空。

  要纠正该错误，请为 TableInput `TableType` 属性指定值，使其作为 AWS Glue `CreateTable` API 调用或 AWS CloudFormation 模板的一部分。`TableType` 可能的值包括 `EXTERNAL_TABLE` 或 `VIRTUAL_VIEW`。

  此要求仅适用于使用 AWS Glue `CreateTable` API 操作或 `AWS::Glue::Table` 模板创建表的情形。如果您适用 DDL 语句或 AWS Glue 爬网程序为 Athena 创建表，则 `TableType` 属性将自动定义。

支持的函数

Athena 查询中支持的函数对应于 Trino 和 Presto 中的函数。有关单个函数的信息，请参阅 Trino 或 Presto 文档中的函数和运算符部分。

不支持事务数据转换

Athena 不支持对表数据执行基于事务的操作（例如，在 Hive 或 Presto 中找到的操作）。有关不支持的关键字的完整列表，请参阅不支持的 DDL。

更改表状态的操作是 ACID

当您创建、更新或删除表时，这些操作保证与 ACID 兼容。例如，如果多个用户或客户端同时尝试创建或更改现有的表，则只有一个会成功。

表为 EXTERNAL

除非是创建 Iceberg 表，否则请始终使用 EXTERNAL 关键字。如果您将没有 EXTERNAL 关键字的 CREATE TABLE 用于非 Iceberg 表，Athena 会发出错误。当您在 Athena 中删除表时，仅删除表元数据；数据将保留在 Amazon S3 中。

使用 AWS Glue 或 Athena 控制台创建表

您可以使用 AWS Glue、“添加表”表单或在 Athena 查询编辑器中运行 DDL 语句，从而在 Athena 中创建表。

使用 AWS Glue 爬网程序创建表

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 在查询编辑器中，选择 Tables and views（表和视图）旁的 Create（创建），然后选择 AWS Glue crawler（爬网程序）。
3. 按照 AWS Glue 控制台的 Add crawler（添加爬网程序）页面上的步骤，添加爬网程序。

   有关更多信息，请参阅 使用 AWS Glue 爬网程序。

使用 Athena“创建表”表单创建表

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 在查询编辑器中，选择 Tables and views（表和视图）旁边的 Create（创建），然后选择 S3 bucket data（S3 存储桶数据）。
3. 在 Create Table From S3 bucket data（从 S3 存储桶数据创建表）表单中，输入创建表所需的信息，然后选择 Create table（创建表）。有关表单中字段的更多信息，请参阅 使用表单添加表。

使用 Hive DDL 创建表

1. 从 Database (数据库) 菜单，选择要为其创建表的数据库。如果您未在 CREATE TABLE 语句中指定数据库，则将在查询编辑器中当前选定的数据库内创建表。
2. 在查询编辑器中输入类似以下内容的语句，然后选择 Run Query（运行查询），或者按 **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/';
```

显示表信息

在 Athena 中创建表后，其名称将显示在左侧的 Tables（表）列表中。要显示有关该表的信息并对其进行管理，在 Athena 控制台中选择表名称旁边的垂直三个点。

- Preview Table（预览表）– 通过在 Athena 查询编辑器中运行 `SELECT * FROM "database_name"."table_name" LIMIT 10` 语句来显示所有列的前 10 行。
- Generate table DDL（生成表 DDL）– 生成 DDL 语句，您可以使用该语句通过在 Athena 查询编辑器中运行 `SHOW CREATE TABLE table_name` 语句来重新创建表。
- Load partitions（加载分区）– 在 Athena 查询编辑器中运行 `MSCK REPAIR TABLE table_name` 语句。仅当表具有分区时，此选项才可用。
- Insert into editor（插入编辑器）– 将表名称插入当前编辑位置的查询编辑器中。
- 删除表 – 显示一个确认对话框，询问是否要删除表。如果您同意，请在 Athena 查询编辑器中运行 `DROP TABLE table_name` 语句。
- 表属性 – 显示表名称、数据库名称、创建的时间以及表是否包含已加密数据。

表、数据库和列的名称

使用这些技巧在 Athena 中命名数据库对象。

数据库名称、表名和列名的要求

- AWS Glue 中可接受的数据库名称、表名以及列名的字符必须是 UTF-8 字符串。字符串长度不得少于 1 个字节，也不得超过 255 个字节。超过此限制会生成错误，例如“名称”处的值无法满足约束条件：成员的长度必须小于或等于 255 个字符。其中可以使用的字符包括空格，并由以下单行字符串模式定义字符：

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

- 目前，AWS Glue 正则表达式模式允许在名称的开头添加前导空格。但这些前导空格可能难以检测且创建后可能导致可用性问题，所以请避免创建带有前导空格的对象名称。
- 如果您使用 AWS::Glue::Database AWS CloudFormation 模板创建 AWS Glue 数据库而未指定数据库名称，AWS Glue 会自动以与 Athena 不兼容的 *resource_name-random_string* 格式生成数据库名称。
- 您可以使用 AWS Glue Catalog Manager 重命名列，但不重命名表名称或数据库名称。要解决此限制，必须使用旧数据库的定义来创建具有新名称的数据库。然后，使用旧数据库中的表定义在新数据库中重新创建表。为此，您可以使用 AWS CLI 或 AWS Glue 软件开发工具包。要查看步骤，请参阅使用 AWS CLI 重新创建 AWS Glue 数据库及其表。

Athena 中的表名称和表列名称使用小写

Athena 在 DDL 和 DML 查询中接受混合大小写，但在执行查询时使用小写名称。因此，请避免使用混合大小写表或列名称，并且不要单独依赖 Athena 中的大小写来区分这些名称。例如，如果您使用 DDL 语句创建名为 `Castle` 的列，则创建的列将被转换成小写，`castle`。如果您随后将 DML 查询中的列名指定为 `Castle` 或者 `CASTLE`，Athena 会将名称转换为小写以为您运行查询，但使用您在查询中选择的大小写显示列标题。

数据库、表和列名的长度必须小于或等于 255 个字符。

以下划线开头的名称

创建表时，使用反引号将以下划线开头的表、视图或列名称括起来。例如：

```
CREATE EXTERNAL TABLE IF NOT EXISTS `_myunderscoretable`(
  `_id` string, `_index` string)
LOCATION 's3://DOC-EXAMPLE-BUCKET/'
```

以数字开头的表、视图或列名称

在运行 SELECT、CTAS 或 VIEW 查询中，用引号将以数字开头的表、视图或列名称等标识符括起来。例如：

```
CREATE OR REPLACE VIEW "123view" AS
SELECT "123columnone", "123columntwo"
FROM "234table"
```

列名和复杂类型

对于复杂类型，只有字母数字字符、下划线 (_) 和句号 (.) 允许在列名称中使用。要为包含受限字符的键创建表和映射，可以使用自定义 DDL 语句。有关更多信息，请参阅 AWS 大数据博客文章：使用 JSONSerDe 在 Amazon Athena 中通过嵌套 JSON 和映射来创建表。

保留字

必须对 Athena 中的某些预留字进行转义。要对 DDL 语句中的保留关键字进行转义，请使用反引号 (`) 将其括起来。要对 SQL SELECT 语句以及视图的查询中的保留关键字进行转义，请使用双引号 (") 将其括起来。

有关更多信息，请参阅 保留关键字。

其他 资源

有关完整的数据库和表创建语法，请参阅以下页面。

- CREATE DATABASE
- CREATE TABLE

有关 AWS Glue 中数据库和表的更多信息，请参阅《AWS Glue Developer Guide》中的 Databases 和 Tables。

保留关键字

当您在 Athena 中运行包含保留关键字的查询时，必须用特殊字符将保留关键字括起来以便对它们进行转义。可使用本主题中的列表检查 Athena 中保留的关键字。

要对 DDL 语句中的保留关键字进行转义，请使用反引号 (`) 将其括起来。要对 SQL SELECT 语句以及视图的查询中的保留关键字进行转义，请使用双引号 (") 将其括起来。

DDL 语句中保留关键字的列表

Athena 在其 DDL 语句中使用以下保留关键字列表。如果您未经转义即使用它们，Athena 会发出错误。要进行转义，请使用反引号 (`) 将其括起来。

如果未使用反引号 (`) 将 DDL 保留关键字括起来，则不能在 DDL 语句中将这些保留关键字用作标识符名称。

```
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
```

SQL SELECT 语句中保留关键字的列表

Athena 在 SQL SELECT 语句以及视图查询中使用以下保留关键字列表。

如果您将这些关键字用作标识符，则必须在查询语句中使用双引号 (") 将它们括起来。

```
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
```

保留关键字的查询示例

以下示例中的查询使用反引号 (`) 来转义 DDL 相关的保留关键字 partition 和 date（用于表名称和其中一个列名称）：

```
CREATE EXTERNAL TABLE `partition` (
`date` INT,
col2 STRING
)
PARTITIONED BY (year STRING)
STORED AS TEXTFILE
LOCATION 's3://DOC-EXAMPLE-BUCKET/test_examples/';
```

以下示例查询包含一个列名称，此列名称在 ALTER TABLE ADD PARTITION 和 ALTER TABLE DROP PARTITION 语句中包含 DDL 相关的保留关键字。DDL 保留关键字括在反引号 (`) 中：

```
ALTER TABLE test_table
ADD PARTITION (`date` = '2018-05-14')
```

```
ALTER TABLE test_table
DROP PARTITION (`partition` = 'test_partition_value')
```

以下查询示例包含一个保留关键字 (end) 作为 SELECT 语句中的标识符。此关键字用双引号进行转义：

```
SELECT *
FROM TestTable
WHERE "end" != nil;
```

以下查询示例在 SELECT 语句中包含一个保留关键字 (first)。此关键字用双引号进行转义：

```
SELECT "itemId"."first"
FROM testTable
LIMIT 10;
```

Amazon S3 中的表位置

在 Athena 中运行 CREATE TABLE 查询时，Athena 将您的表注册到 AWS Glue Data Catalog 中，这是 Athena 存储元数据的地方。

要在 Amazon S3 中指定数据的路径，请使用 LOCATION 属性，如以下示例中所示：

```
CREATE EXTERNAL TABLE `test_table`(
...
)
ROW FORMAT ...
STORED AS INPUTFORMAT ...
OUTPUTFORMAT ...
LOCATION s3://DOC-EXAMPLE-BUCKET/folder/
```

- 有关命名存储桶的信息，请参阅《Amazon Simple Storage Service 用户指南》中的 存储桶限制和局限。
- 有关在 Amazon S3 中使用文件夹的信息，请参阅《Amazon Simple Storage Service 用户指南》中的 使用文件夹。

Amazon S3 中的 LOCATION 指定表示您的表的所有文件。

> **Important**
>
> Athena 会读取存储在您指定的 Amazon S3 文件夹中的全部数据。如果您有不希望 Athena 读取的数据，请不要在与您希望 Athena 读取的数据所在的相同 Amazon S3 文件夹中存储这些数据。如果您在使用分区确保 Athena 扫描某个分区中的数据，则 WHERE 筛选条件必须包括该分区。有关更多信息，请参阅 表位置和分区。

在 CREATE TABLE 语句中指定 LOCATION 时，请使用以下准则：

- 使用尾部斜杠。

- 您可以使用 Amazon S3 文件夹或 Amazon S3 访问点别名的路径。有关 Amazon S3 访问点别名的信息，请参阅《Amazon S3 用户指南》中的为您的访问点使用存储桶式别名。

使用:

```
s3://DOC-EXAMPLE-BUCKET/folder/
```

```
s3://DOC-EXAMPLE-BUCKET-metadata-s3alias/folder/
```

请勿使用以下任何项目来指定数据的 LOCATION。

- 请勿使用文件名、下划线、通配符或 glob 模式来指定文件位置。
- 请勿将完整 HTTP 表示法（例如 s3.amazon.com）添加到 Amazon S3 存储桶路径。
- 请勿在路径中使用空文件夹（如 //），如下所示：S3://DOC-EXAMPLE-BUCKET/folder//folder/。虽然这是有效的 Amazon S3 路径，但 Athena 不允许并会将其更改为 s3://DOC-EXAMPLE-BUCKET/folder/folder/，删除额外的 /。

  请勿使用：

```
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
```

表位置和分区

您的源数据可以基于一组列按 Amazon S3 文件夹（称为分区）分组。例如，这些列可以表示创建特定记录的年、月和日。

创建表时，您可以选择对其进行分区。当 Athena 针对未分区的表执行 SQL 查询时，它使用来自表分区定义的 LOCATION 属性作为基本路径，以列出并随后扫描所有可用文件。但是，您必须使用分区信息更新 AWS Glue Data Catalog，然后才能查询已分区的表。此信息表示特定分区中文件的架构，以及该分区的文件在 Amazon S3 中的 LOCATION。

- 要了解如何操作 AWS Glue 爬虫程序添加分区，请参阅《AWS Glue 开发人员指南》中的 爬网程序如何确定何时创建分区？
- 要了解如何配置爬网程序以使其为现有分区中的数据创建表，请参阅将多个数据源和爬网程序结合使用。
- 在 Athena 中，您还可以直接在表中创建分区。有关更多信息，请参阅 在 Athena 中对数据进行分区。

Athena 对已分区的表运行查询时，它检查以确认在查询的 `WHERE` 子句中是否使用了任何分区列。如果使用了分区列，则 Athena 请求 AWS Glue Data Catalog 返回与指定分区列匹配的分区规范。分区规范包含 `LOCATION` 属性，从而告知 Athena 在读取数据时应使用哪个 Amazon S3 前缀。在这种情况下，仅 扫描使用此前缀存储的数据。如果不在 `WHERE` 子句中使用已分区的列，则 Athena 将扫描属于表的分区的所有文件。

有关在 Athena 中使用分区来提升查询性能并降低查询成本的示例，请参阅 Top 10 performance tuning tips for Amazon Athena。

列式存储格式

Apache Parquet 和 ORC 是针对快速检索数据进行了优化的列式存储格式，用于 AWS 分析应用程序中。

列式存储格式具有以下特征，使其适合用于 Athena：

- 按列压缩，针对列数据类型选择压缩算法，可以节省 Amazon S3 中的存储空间，并减少查询处理期间的磁盘空间和输入/输出。
- Parquet and ORC 中的谓词下推使得 Athena 查询可以只提取所需的数据块，从而提高查询性能。当 Athena 查询从您的数据获取特定列值时，它使用来自数据块谓词的统计信息（例如最大/最小值）来确定读取还是跳过改数据块。
- Parquet 和 ORC 中的数据拆分使得 Athena 可以将数据读取拆分为多个读进程，在查询处理期间增加并行度。

要将现有原始数据从其他存储格式转换为 Parquet 或 ORC，您可以在 Athena 中运行 CREATE TABLE AS SELECT (CATS)查询，并将数据存储格式指定为 Parquet 或 ORC，或使用 AWS Glue 爬网程序。

在 Parquet 和 ORC 之间进行选择

将根据具体的使用需求在 ORC（优化的行列式）和 Parquet 之间进行选择。

Apache Parquet 可实现高效的数据压缩和编码方案，非常适合运行复杂的查询和处理大量数据。Parquet 已针对与 Apache Arrow 配合使用进行了优化，如果使用的工具与 Arrow 相关，这可能会很有优势。

可通过 ORC 提高效存储 Hive 数据。ORC 文件通常比 Parquet 文件小，ORC 索引可以加快查询速度。此外，ORC 还支持结构、映射和列表等复杂类型。

在 Parquet 和 ORC 之间进行选择时，请注意以下事项：

查询性能 - 由于 Parquet 支持更广泛的查询类型，如果您打算执行复杂查询，建议选择 Parquet。

复杂数据类型 - 如果您使用的是复杂数据类型，建议选择 ORC，因为它支持更广泛的复杂数据类型。

文件大小 - 如果需要考虑磁盘空间，ORC 生成的文件通常较小，可以降低存储成本。

压缩 - Parquet 和 ORC 都可实现优质压缩效果，但最适合您的格式可能取决于您的具体使用案例。

发展 - Parquet 和 ORC 都支持架构发展，这意味着您可以随着时间的推移添加、移除或修改列。

Parquet 和 ORC 都适用于大数据应用程序，但是还是要根据场景需求进行选择。您可能需要对数据和查询进行基准测试，才能了解哪种格式更适合您的使用案例。

转换为列式格式

如果您将数据转换为开源列式格式（如 Apache Parquet 或 ORC），您的 Amazon Athena 查询性能会提高。

将 JSON 或 CSV 等源数据轻松转换为列式格式的选项包括使用 CREATE TABLE AS 查询或在 AWS Glue 中运行任务。

- 您可以使用 `CREATE TABLE AS` (CTAS) 查询一步将数据转换为 Parquet 或 ORC。有关示例，请参阅 CTAS 查询的示例 页面上的示例：将查询结果写入不同的格式。
- 有关运行 AWS Glue 任务以将 CSV 数据转换为 Parquet 的信息，请参阅 AWS 大数据博客文章 使用 AWS Glue 和 Simple Storage Service (Amazon S3) 构建数据湖基础中的“将数据从 CSV 转换为 Parquet 格式”。AWS Glue 支持使用相同的技术将 CSV 数据转换为 ORC，或将 JSON 数据转换为 Parquet 或 ORC。

在 Athena 中对数据进行分区

通过分区您的数据，您可以限制每个查询扫描的数据量，从而提高性能并降低成本。您可按任何键对数据进行分区。一种常见的做法是根据时间对数据进行分区，通常会导致多级别分区方案。例如，每个小

时都有数据传入的客户可能决定按年、月、日期和小时进行分区。另一位客户的数据来自许多不同的来源，但每天只加载一次，则可以按数据源标识符和日期进行分区。

Athena 可以使用 Apache Hive 风格的分区，其数据路径包含通过等号连接的键值对（例如，`country=us/...` 或 `year=2021/month=01/day=26/...`）。因此，路径包含分区键的名称和每个路径所表示的值。要将新的 Hive 分区加载到分区表中，您可以使用仅适用于 Hive 风格分区的 MSCK REPAIR TABLE 命令。

Athena 还可以使用非 Hive 风格的分区方案。例如，CloudTrail 日志和 Firehose 传输流在日期部分使用单独的路径组件，例如 `data/2021/01/26/us/6fc7845e.json`。对于此类非 Hive 样式的分区，您可以使用 ALTER TABLE ADD PARTITION 来手动添加分区。

注意事项和限制

在使用分区时，请记住以下几点：

- 如果您查询分区表并在 `WHERE` 子句中指定分区，Athena 仅从该分区扫描数据。有关更多信息，请参阅 表位置和分区。
- 如果您针对包含大量对象且数据未分区的 Amazon S3 存储桶发出查询，则此类查询可能影响 Amazon S3 中的 `GET` 请求速率限制并导致 Amazon S3 异常。为防止错误发生，请将数据分区。另外，考虑调整 Amazon S3 的请求速率。有关更多信息，请参阅最佳实践设计模式：优化 Simple Storage Service (Amazon S3) 性能。
- 要与 Athena 结合使用的分区位置必须使用 `s3` 协议（例如，`s3://DOC-EXAMPLE-BUCKET/folder/`）。在 Athena 中，当对包含的表运行 `MSCK REPAIR TABLE` 查询时，使用其他协议的位置（例如，`s3a://DOC-EXAMPLE-BUCKET/folder/`）将导致查询失败。
- 确保 Amazon S3 路径为小写而不是驼峰式大小写（例如，`userid` 而不是 `userId`）。如果 S3 路径为驼峰式大小写，则 `MSCK REPAIR TABLE` 不会将分区添加到 AWS Glue Data Catalog。有关更多信息，请参阅 MSCK REPAIR TABLE。
- 由于 `MSCK REPAIR TABLE` 同时扫描文件夹及其子文件夹以查找匹配的分区方案，请确保在单独的文件夹层次结构中保留单独表的数据。例如，假设您在 `s3://DOC-EXAMPLE-BUCKET1` 中拥有表 1 的数据，在 `s3://DOC-EXAMPLE-BUCKET1/table-2-data` 中拥有表 2 的数据。如果两个表都是按字符串分区的，则 `MSCK REPAIR TABLE` 会将表 2 的分区添加到表 1 中。为了避免这种情况，请使用单独的文件夹结构，如 `s3://DOC-EXAMPLE-BUCKET1` 和 `s3://DOC-EXAMPLE-BUCKET2`。请注意，此行为与 Amazon EMR 和 Apache Hive 一致。
- 如果将 AWS Glue Data Catalog 与 Athena 一起使用，请参阅 AWS Glue 端点和限额，以了解每账户和每表的分区服务限额。

- 尽管 Athena 支持查询具有 1000 万个分区的 AWS Glue 表，但 Athena 的单次扫描读取量最多为 100 万个分区。在这种情况下，分区索引可能有益。有关更多信息，请参阅 AWS 大数据博客文章：使用 AWS Glue Data Catalog 分区索引提高 Amazon Athena 查询性能。
- 如果您使用的是 AWS Glue Data Catalog，请求增加分区限额，请访问 AWS Glue 的服务限额控制台。

使用分区数据创建和加载表

要创建使用分区的表，请在 CREATE TABLE 语句中使用 PARTITIONED BY 子句。PARTITIONED BY 子句定义了对数据进行分区所用的键，如以下示例所示。LOCATION 子句指定了分区数据的根位置。

```
CREATE EXTERNAL TABLE users (
first string,
last string,
username string
)
PARTITIONED BY (id string)
STORED AS parquet
LOCATION 's3://DOC-EXAMPLE-BUCKET'
```

创建表之后，您在分区中加载数据以进行查询。对于 Hive 样式的分区，您可以运行 MSCK REPAIR TABLE。对于非 Hive 样式的分区，您可以使用 ALTER TABLE ADD PARTITION 来手动添加分区。

准备 Hive 风格和非 Hive 风格的数据用于查询

以下部分介绍了如何准备 Hive 风格和非 Hive 风格的数据以便在 Athena 中查询。

场景 1：以 Hive 格式存储在 Amazon S3 上的数据

在此场景中，分区存储在 Amazon S3 中的单独文件夹内。例如，以下是由 `aws s3 ls` 命令输出的示例广告展示的部分列表，其中列出了指定前缀下的 S3 对象：

```
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/
```

日志存储在这里，列名称 (DT) 设置为等于日期、小时和分钟增量。当您向 DDL 提供父文件夹的位置、架构和分区列的名称时，Athena 可以查询这些子文件夹中的数据。

创建表

要从此类数据中生成一个表，请连同“dt”一起创建一个分区，如以下 Athena DDL 语句所示：

```
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/' ;
```

此表使用 Hive 的本机 JSON 串行器/解串器来读取在 Amazon S3 中存储的 JSON 数据。有关支持的格式的更多信息，请参阅支持的 SerDes 和数据格式。

运行 MSCK REPAIR TABLE

运行 `CREATE TABLE` 查询后，在 Athena 查询编辑器中运行 `MSCK REPAIR TABLE` 命令来加载分区，如以下示例所示。

```
MSCK REPAIR TABLE impressions
```

运行此命令后，即可进行数据查询。

查询数据

使用分区列从展示表中查询数据。示例如下：

```
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
```

此查询应显示与以下内容类似的结果：

```
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
```

方案 2：数据未以 Hive 格式进行分区

在以下示例中，aws s3 ls 命令显示存储在 Amazon S3 中的 ELB 日志。请注意，数据布局不使用 key=value 对，因此不是 Hive 格式。（aws s3 ls 命令的 --recursive 选项指定了列出的指定目录或前缀下的所有文件或对象。）

```
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_$folder$
```

运行 ALTER TABLE ADD PARTITION

由于数据不是 Hive 格式，因此创建表后，您不能使用 `MSCK REPAIR TABLE` 命令将分区添加到表中。相反，您可以使用 ALTER TABLE ADD PARTITION 命令手动添加每个分区。例如，要加载 s3://athena-examples-*myregion*/elb/plaintext/2015/01/01/ 中的数据，您可以运行以下查询。请注意，对于每个 Amazon S3 文件夹不需要单独的分区列，并且分区键值可能与 Amazon S3 键不同。

```
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/'
```

如果分区已经存在，您会收到错误 Partition already exists（分区已存在）。要避免此错误，您可以使用 `IF NOT EXISTS` 子句。有关更多信息，请参阅 ALTER TABLE ADD PARTITION。要删除分区，您可以使用 ALTER TABLE DROP PARTITION。

分区投影

要避免管理分区，您可以使用分区投影。对于预先知道其结构的高度分区表，分区投影是一个选项。在分区投影中，分区值和位置是根据配置的表属性计算得出的，而不是从元数据存储库中读取出的。由于内存式计算比远程查找更快，因此使用分区投影可以显著减少查询运行时。

有关更多信息，请参阅 使用 Amazon Athena 分区投影。

其他 资源

- 有关 Firehose 数据分区选项的信息，请参阅 Amazon Data Firehose 示例。
- 您可以通过使用 JDBC 驱动程序自动添加分区。
- 您可以使用 CTAS 和 INSERT INTO 对数据集进行分区。有关更多信息，请参阅 将 CTAS 和 INSERT INTO 用于 ETL 和数据分析。

使用 Amazon Athena 分区投影

可以在 Athena 中使用分区投影来加快对高度分区表的查询处理，并自动执行分区管理。

在分区投影中，Athena 使用直接在 AWS Glue 表上配置的表属性计算分区值和位置。表属性允许 Athena“投影”或确定必要的分区信息，无需在 AWS Glue Data Catalog 中进行更耗时的元数据查找。由于内存中的操作通常快于远程操作，因此，分区投影可以减少针对高度分区表的查询的运行时间。根据查询和基础数据的具体特征，分区投影可以显著减少在分区元数据检索方面受到限制的查询的运行时间。

高度分区表的修剪和投影

分区修剪将收集元数据并对其进行“修剪”，以便只存在适用于查询的分区。这通常会加快查询速度。Athena 对所有带分区列的表（包括为分区投影配置的表）使用分区修剪。

通常，在处理查询时，Athena 会在执行分区修剪之前向 AWS Glue Data Catalog 发出 `GetPartitions` 调用。如果表具有大量分区，则使用 `GetPartitions` 会导致性能降低。要避免发生此情况，您可以使用分区投影。分区投影可让 Athena 避免调用 `GetPartitions`，因为分区投影配置为 Athena 提供了自行构建分区所需的所有信息。

使用分区投影

要使用分区投影，请在 AWS Glue Data Catalog 或外部 Hive 元存储中的表属性中为每个分区列指定分区值和投影类型的范围。表的这些自定义属性可让 Athena 了解在对表运行查询时应使用哪种分区模式。在查询执行期间，Athena 使用此信息对分区值进行投影，而不是从 AWS Glue Data Catalog 或外部 Hive 元存储中检索这些值。这不仅将减少查询执行时间，还将自动执行分区管理，因为不再需要在 Athena、AWS Glue 或外部 Hive 元存储中手动创建分区。

> **Important**
>
> 在表上启用分区投影会导致 Athena 忽略在 AWS Glue Data Catalog 或 Hive 元存储中注册到表的任何分区元数据。

使用案例

分区投影在许多场景中都很有用，其中包括：

- 对高度分区表的查询不会像您希望的那样快速完成。
- 在数据中创建新的日期或时间分区时，可以定期向表添加分区。利用分区投影，可以配置可在新数据到达时使用的相对日期范围。
- 您在 Amazon S3 中有高度分区的数据。在 AWS Glue Data Catalog 或 Hive 元存储中为数据建模是不切实际的，并且您的查询仅读取其中的一小部分。

可投影的分区结构

当您的分区遵循可预测的模式时（包括但不限于以下情况），分区投影最易配置：

- 整数 – 任何连续的整数序列，例如 `[1, 2, 3, 4, ..., 1000]` 或 `[0500, 0550, 0600, ..., 2500]`。
- 日期 – 任何连续的日期或日期时间序列，例如 `[20200101, 20200102, ..., 20201231]` 或 `[1-1-2020 00:00:00, 1-1-2020 01:00:00, ..., 12-31-2020 23:00:00]`。
- 枚举值 – 一组有限的枚举值，例如机场代码或 AWS 区域。
- AWS 服务 日志 – AWS 服务 日志通常具有一个已知结构，您可以在 AWS Glue 中指定该结构的分区方案，并且 Athena 可以将其用于分区投影。

自定义分区路径模板

预设情况下，Athena 使用 `s3://DOC-EXAMPLE-BUCKET/<table-root>/partition-col-1=<partition-col-1-val>/partition-col-2=<partition-col-2-val>/` 格式生成分区位置，但如果您数据的组织方式不同，则 Athena 会提供用于自定义此路径模板的机制。要查看步骤，请参阅 指定自定义 S3 存储位置。

注意事项和限制

请注意以下事项：

- 有了分区投影，便无需在 AWS Glue 或外部 Hive 元存储中手动指定分区。
- 在表上启用分区投影时，Athena 将为该表忽略 AWS Glue Data Catalog 或外部 Hive 元存储中的任何分区元数据。
- 如果 Amazon S3 中没有投影分区，Athena 仍会对分区进行投影。Athena 不会引发错误，但也未返回任何数据。不过，如果您的空分区过多，则性能会低于传统的 AWS Glue 分区。如果有一半以上的投影分区为空，建议您使用传统分区。
- 超出为分区投影定义的范围边界的值的查询不会返回错误。相反，查询将会运行，但返回零行。例如，如果您的时间相关数据从 2020 年开始，并且定义为 `'projection.timestamp.range'='2020/01/01,NOW'`，一个类似 `SELECT * FROM table-name WHERE timestamp = '2019/02/02'` 的查询将成功执行，但将返回零行。
- 分区投影仅在通过 Athena 查询表时可用。如果通过其他服务（例如 Amazon Redshift Spectrum、Athena for Spark 或 Amazon EMR）读取同一个表，则使用标准分区元数据。
- 由于分区投影是一项仅限 DML 的功能，因此 `SHOW PARTITIONS` 不会列出由 Athena 投影但未注册到 AWS Glue 目录或外部 Hive 元存储中的分区。
- Athena 不使用视图的表属性作为分区投影的配置。要变通解决此限制，请在视图引用的表的表属性中配置和启用分区投影。

- Lake Formation 数据筛选条件不能与 Athena 中的分区投影结合使用。

视频

下面的视频演示了如何使用分区投影来提高 Athena 中查询的性能。

使用 Amazon Athena 分区投影

主题

- 设置分区投影
- 受支持的分区投影类型
- 动态 ID 分区
- Amazon Data Firehose 示例

设置分区投影

在表的属性中设置分区投影是一个包含两个步骤的过程：

1. 指定每个分区列的数据范围和相关模式，或使用自定义模板。
2. 为表启用分区投影。

> **Note**
>
> 在向现有表添加分区投影属性之前，要为其设置分区投影属性的分区列必须已存在于表架构中。如果分区列尚不存在，则必须将分区列手动添加到现有表中。AWS Glue 不会自动为您执行此步骤。

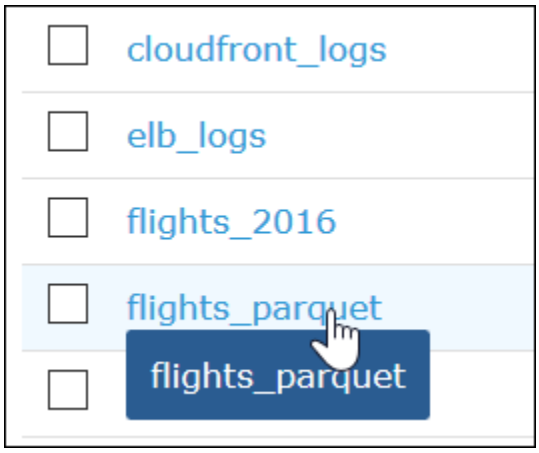
本节介绍如何为 AWS Glue 设置表属性。要设置这些表属性，您可以使用 AWS Glue 控制台、Athena CREATE TABLE 查询或 AWS Glue API 操作。以下过程说明如何在 AWS Glue 控制台中设置属性。

要使用 AWS Glue 控制台配置并启用分区投影

1. 登录 AWS Management Console，然后打开 AWS Glue 控制台，网址为：https://console.aws.amazon.com/glue/。
2. 选择 Tables (表) 选项卡。

在 Tables (表) 选项卡上，您可以编辑现有表，也可以选择 Add tables (添加表) 来创建新表。有关手动添加表或使用爬网程序添加表的信息，请参阅《AWS Glue 开发人员指南》中的在 AWS Glue 控制台上使用表。

3. 在表的列表中，选择要编辑的表的链接。



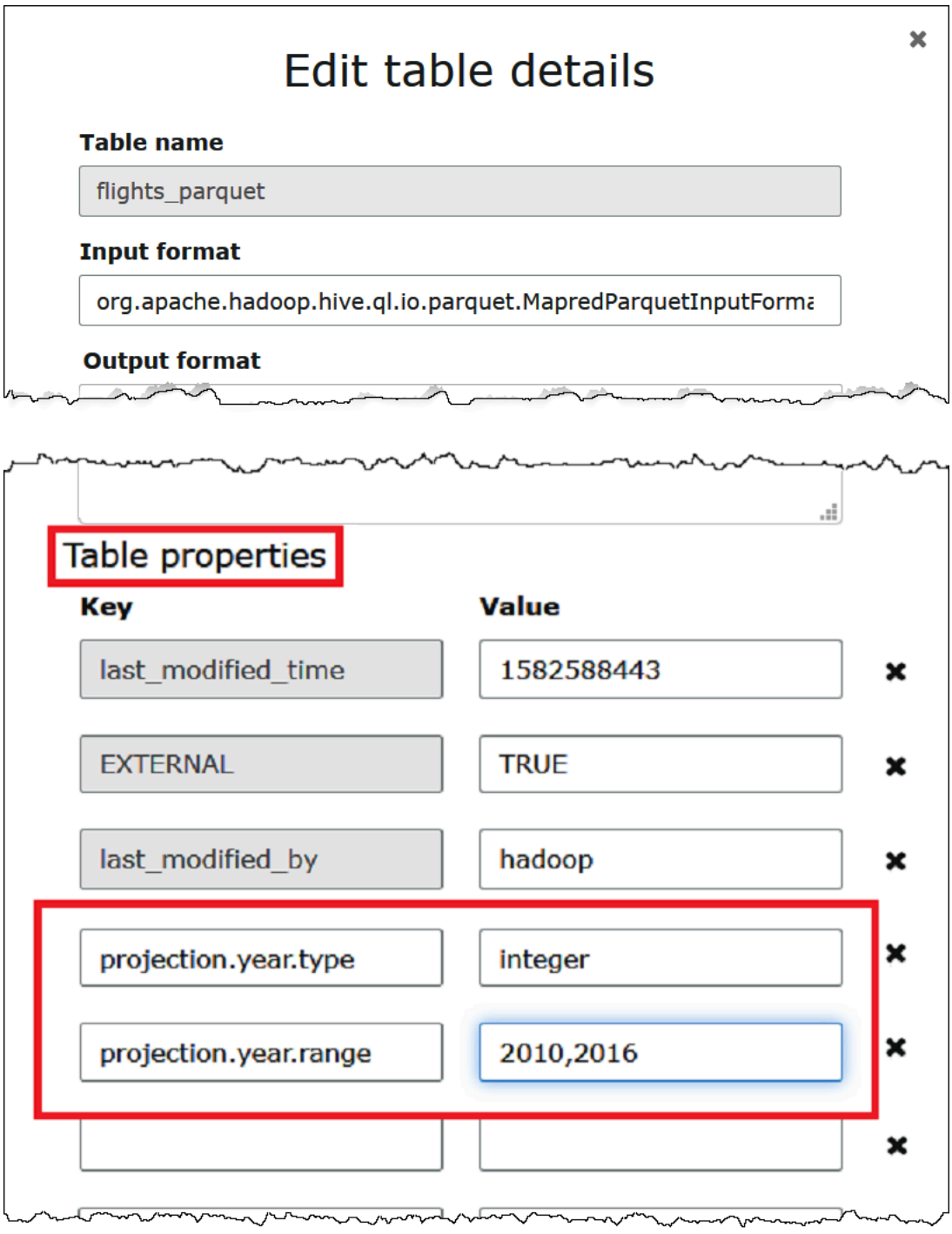
4. 依次选择 Actions（操作）、Edit table（编辑表）。
5. 在 Edit table（编辑表）页面的 Table properties（表属性）部分中，对于每个分区列，添加以下键值对：

   a. 对于 Key (键)，添加 `projection.`*`columnName`*`.type`。

   b. 对于 Value (值)，添加受支持的类型之一：`enum`、`integer`、`date` 或 `injected`。有关更多信息，请参阅 受支持的分区投影类型。

6. 按照受支持的分区投影类型中的指导信息进行操作，根据配置要求添加其他键/值对。

   以下示例表配置将为分区投影配置 `year` 列，从而将可返回的值限定为 2010 年至 2016 年的数据。

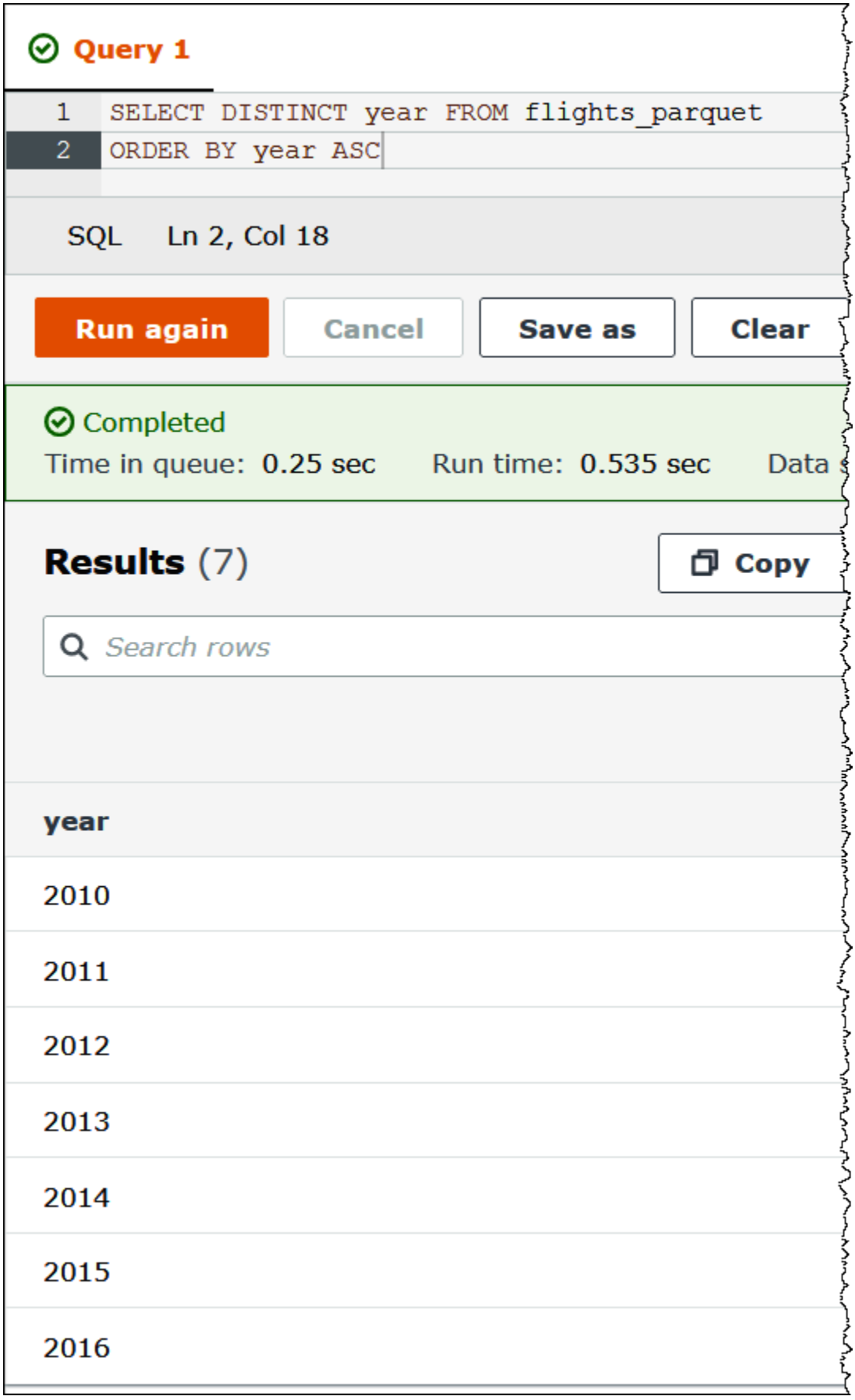


7. 添加键/值对以启用分区投影。对于 Key (键)，输入 `projection.enabled`，对于其 Value (值)，输入 `true`。

> **Note**
>
> 您可以随时通过将 `projection.enabled` 设置为 `false` 来对此表禁用分区投影。

8. 完成后，选择 Save。
9. 在 Athena 查询编辑器中，测试查询为表配置的列。

   以下示例查询使用 `SELECT DISTINCT` 从 `year` 列返回唯一值。数据库包含 1987 年至 2016 年的数据，但 `projection.year.range` 属性将返回的值限定为 2010 年至 2016 年的数据。


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**
>
> 如果将 `projection.enabled` 设置为 `true`，但未能配置一个或多个分区列，则会收到与以下内容类似的错误消息：`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).`

指定自定义 S3 存储位置

在 AWS Glue 中编辑表属性时，还可以为投影分区指定自定义 Amazon S3 路径模板。利用自定义模板，Athena 可以将分区值正确映射到未遵循典型 `.../column=value/...` 模式的自定义 Amazon S3 文件位置。

可以选择使用自定义模板。但是，如果您使用自定义模板，则该模板中必须为每个分区列包含一个占位符。模板化位置必须以正斜杠结尾，以便分区的数据文件位于每个分区的“文件夹”中。

指定自定义分区位置模板

1. 按照步骤使用 AWS Glue 控制台配置并启用分区投影，添加一个指定自定义模板的附加键/值对，如下所示：

   a. 对于键，输入 `storage.location.template`。

   b. 对于 Value (值)，指定一个位置，该位置为每个分区列包含一个占位符。确保每个占位符(和 S3 路径本身)都以单个正斜杠结束。

      以下示例模板值假定一个具有分区列 a、b 和 c 的表。

      ```
      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}/
      ```

      对于同一个表，以下示例模板值无效，因为它不包含列 c 的占位符。

```
s3://DOC-EXAMPLE-BUCKET/table_root/a=${a}/${b}/some_static_subdirectory/
```

2. 选择 应用。

受支持的分区投影类型

表可以具有 enum、integer、date, 或 injected 分区列类型的任意组合。

枚举类型

将 enum 类型用于其值为枚举集成员（例如，机场代码或 AWS 区域）的分区列。

在表中定义分区属性，如下所示：

属性名称	示例值	描述
projection. *columnNam e* .type	enum	必需。要用于 *columnName* 列的投影类型。该值必须是 enum（不区分大小写）才能发出使用枚举类型的信号。允许前导和尾随空格。
projection. *columnNam e* .values	A,B,C,D,E,F,G,Unkn own	必需。*columnName* 列的枚举分区值的逗号分隔列表。任何空格均被视为枚举值的一部分。

> **Note**
>
> 作为最佳实践，我们建议将基于 enum 的分区投影使用数量限制在数十个或更少的范围。虽然 enum 投影没有特定限制，但表元数据的总大小在 gzip 压缩时不能超过约 1 MB 的 AWS Glue 限制。请注意，此限制将在表格的关键部分（如列名称、位置、存储格式和其他部分）之间共享。如果您发现自己在 enum 投影使用的唯一 ID 多于数十个，请考虑一种替代方法，例如在代理字段中分桶为较少数量的唯一值。通过折衷基数，您可以控制 enum 字段中返回的唯一值数量。

整数类型

将整数类型用于其可能的值可被解释为定义的范围内的整数的分区列。投影整数列目前仅限于 Java 有符号长整型值的范围（-2^{63} 到 $2^{63}-1$，含这两个值）。

属性名称	示例值	描述
projection. *columnName* .type	integer	必需。要用于 *columnName* 列的投影类型。该值必须是 integer（不区分大小写）才能发出使用整数类型的信号。允许前导和尾随空格。
projection. *columnName* .range	0,10 -1,8675309 0001,9999	必需。一个包含两个元素的逗号分隔列表，该列表提供将由对 *columnName* 列进行的查询返回的最小范围值和最大范围值。请注意，必须使用逗号来分隔值，而不是使用连字符。这些值包含在内，它们可以是负数，并且可以有前导零。允许前导和尾随空格。
projection. *columnName* .interval	1 5	可选。一个正整数，它指定 *columnName* 列的连续分区值之间的间隔。例如，具有 interval 值“1”的 range 值“1,3”将产生值 1、2 和 3。具有 interval 值“2”的相同 range 值将产生值 1 和 3，并跳过 2。允许前导和尾随空格。默认 为 1。
projection. *columnName* .digits	1 5	可选。一个正整数，它指定要包含在 *columnName* 列的分区值的最终表示形式中的位数。例如，具有 digits 值“1”的 range 值“1,3”将产生值 1、2

属性名称	示例值	描述
		和 3。具有 digits 值“2”的相同 range 值将产生值 01、02 和 03。允许前导和尾随空格。默认值不包含静态位数和前导零。

日期类型

将日期类型用于其值可被解释为定义的范围内的日期（带可选时间）的分区列。

> **Important**
>
> 投影日期列在查询执行时以协调世界时 (UTC) 格式生成。

属性名称	示例值	描述
projection. *columnName* .type	date	必需。要用于 *columnName* 列的投影类型。该值必须是 date（不区分大小写）才能发出使用日期类型的信号。允许前导和尾随空格。
projection. *columnName* .range	201701,201812 01-01-2010,12-31-2018 NOW-3YEARS,NOW 201801,NOW+1MONTH	必需。一个包含两个元素的逗号分隔列表，该列表提供 *columnName* 列的最小范围和最大范围 range 值。这些值包含在内，它们可以使用任何与 Java java.time.* 日期类型兼容的格式。最小值和最大值必须使用相同的格式。.format 属性中指定的格式必须是用于这些值的格式。 此列还可以包含格式为以下正则表达式模式的相对日期字符串： \s*NOW\s*(([\+\-])\s*([0-9]+)\s*(YEARS?|MONTHS?|WEEKS?|DAYS?|HOURS?|MINUTES?|SECONDS?)\s*)?

属性名称	示例值	描述
		允许使用空格，但在日期中，文本将被视为日期字符串的一部分。
projection. *columnName* .format	yyyyMM dd-MM-yyyy dd-MM-yyyy-HH-mm-ss	必需。基于 Java 日期格式 DateTimeFormatter 的日期格式字符串。可以是任何受支持的 Java.time.* 类型。
projection. *columnName* .interval	1 5	一个正整数，它指定 *columName* 列的连续分区值之间的间隔。例如，具有 interval 值 1 和 interval.unit 值 MONTHS 的 range 值 2017-01,2018-12 将产生值 2017-01、2017-02、2017-03，依此类推。具有 interval 值 2 和 interval.unit 值 MONTHS 的相同 range 值将产生值 2017-01、2017-03、2017-05，依此类推。允许前导和尾随空格。 如果提供的日期的精度为单日或单月，则 interval 是可选的，默认值分别为 1 天或 1 个月。否则，interval 是必需的。

属性名称	示例值	描述
projection. *columnName* .interval.unit	YEARS MONTHS WEEKS DAYS HOURS MINUTES SECONDS MILLIS	一个时间单位词，它表示 ChronoUnit 的序列化形式。可能的值为 YEARS、MONTHS、WEEKS、DAYS、HOURS、MINUTES、或 MILLIS。这些值不区分大小写。 如果提供的日期的精度为单日或单月，则 interval.unit 是可选的，默认值分别为 1 天或 1 个月。否则，interval.unit 是必需的。

注入的类型

将注入的类型用于分区列，这些分区列的可能值无法在某个逻辑范围内通过程序生成，而是在查询的 WHERE 子句中作为单个值提供。

请务必记住以下几点：

- 如果未为每个注入列提供筛选条件表达式，则对注入列进行的查询将失败。
- 对于在注入列上具有多个筛选条件表达式值的查询，只有在这些值分离时，这些查询才会成功。
- 仅支持 string 类型的列。

属性名称	值	描述
projection. *columnName* .type	inje	必需。要用于 *columnName* 列的投影类型。仅支持 string 类型。指定的值必须是 injected（不区分大小写）。允许前导和尾随空格。

有关更多信息，请参阅 使用 injected 投影类型。

动态 ID 分区

当数据按高基数属性分区时，或者无法事先知道这些值时，您可以使用 `injected` 投影类型。此类属性的示例包括用户名以及设备或产品的 ID。当您使用 `injected` 投影类型配置分区键时，Athena 使用查询本身的值来计算要读取的分区集。

如果某个表具有通过 `injected` 投影类型配置的分区键，则要使 Athena 能够在该表上运行查询，必须满足以下条件：

- 查询必须包括至少一个分区键值。
- 值必须是无需读取任何数据即可评估的文字或表达式。

只要其中一个条件不满足，查询就会失败，并显示以下错误：

CONSTRAINT_VIOLATION：对于注入的投影分区列 *`column_name`*，WHERE 子句必须仅包含静态相等条件，并且必须至少存在一个此类条件。

使用 **`injected`** 投影类型

设想您的数据集包含来自 IoT 设备的事件，并按设备的 ID 进行分区。该数据集具有以下特征：

- 设备 ID 随机生成。
- 新设备经常预置。
- 目前有数十万台设备，将来会有数百万台。

使用传统的元数据存储很难管理此数据集。难以确保数据存储与元存储之间保持分区同步，而且在查询规划期间筛选分区可能很慢。但是，如果您将表配置为使用分区投影并使用 `injected` 投影类型，您会获得两个优势：您不必在元存储中管理分区，并且您的查询不必查找分区元数据。

以下 CREATE TABLE 示例会为刚刚描述的设备事件数据集创建一个表。该表使用注入投影类型。

```
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}"
)
```

以下示例查询会查找在 12 小时内从三台特定设备收到的事件数。

```
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
```

当您运行此查询时，Athena 会看到 `device_id` 分区键的三个值，并使用这些值来计算分区位置。Athena 使用 `storage.location.template` 属性的值来生成以下位置：

- `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`

如果您在分区投影配置中省略了 `storage.location.template` 属性，Athena 会使用 Hive 风格分区，根据 `LOCATION` 中的值（例如，`s3://DOC-EXAMPLE-BUCKET/prefix/device_id=4a770164-0392-4a41-8565-40ed8cec737e`）投影分区位置。

Amazon Data Firehose 示例

当您使用 Firehose 将数据传输到 Amazon S3 时，默认配置会使用类似于以下示例的键写入对象：

```
s3://DOC-EXAMPLE-BUCKET/prefix/yyyy/MM/dd/HH/file.extension
```

要创建在查询时自动查找分区的 Athena 表，而不必在新数据到达时将它们添加到 AWS Glue Data Catalog，您可以使用分区投影。

以下 `CREATE TABLE` 示例使用了默认 Firehose 配置。

```
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}/"
)
```

CREATE TABLE 语句中的 TBLPROPERTIES 子句告诉 Athena 以下内容：

- 查询表时使用分区投影
- 分区键 datehour 的类型为 date（包括可选时间）
- 如何格式化日期
- 日期时间范围 请注意，必须使用逗号来分隔值，而不是使用连字符。
- Amazon S3 上数据所在的位置。

当您查询表时，Athena 会计算 datehour 值并使用存储位置模板生成分区位置列表。

使用 **date** 类型

当您使用投影分区键的 date 类型时，您必须指定一个范围。由于在创建 Firehose 传输流之前没有日期数据，您可以使用创建日期作为开始日期。此外，由于您没有将来日期的数据，您可以使用特殊令牌 NOW 作为结束日期。

在 CREATE TABLE 示例中，开始日期指定为 UTC 时间 2021 年 1 月 1 日午夜。

> **Note**
>
> 配置一个与您的数据尽可能匹配的范围，以便 Athena 仅查找现有分区。

在示例表上运行查询时，Athena 将 `datehour` 分区键上的条件与范围结合使用来生成值。请考虑以下查询：

```
SELECT *
FROM my_ingested_data
WHERE datehour >= '2020/12/15/00'
AND datehour < '2021/02/03/15'
```

SELECT 查询中的第一个条件使用在 CREATE TABLE 语句指定的日期范围开始之前的日期。由于分区投影配置没有为 2021 年 1 月 1 日之前的日期指定分区，因此 Athena 仅在以下位置查找数据，并忽略查询中较早的日期。

```
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/
```

同样，如果查询在 2021 年 2 月 3 日 15:00 之前的日期和时间运行，则最后一个分区将反映当前日期和时间，而不是查询条件中的日期和时间。

如果您想查询最新数据，您可以利用 Athena 不生成未来日期的事实，并仅指定起始 `datehour`，如以下示例所示。

```
SELECT *
FROM my_ingested_data
WHERE datehour >= '2021/11/09/00'
```

选择分区键

您可以指定分区投影如何将分区位置映射到分区键。在上一节的 CREATE TABLE 示例中，日期和小时组合成一个名为 datehour 的分区键，但也可以使用其他方案。例如，您还可以为年、月、日和小时配置具有单独分区键的表。

不过，将日期拆分为年、月、日意味着无法使用 date 分区投影类型。另一种方法是将日期与小时分开，以便仍然利用 date 分区投影类型，但要确保用来指定小时范围的查询更加易于阅读。

考虑到这一点，以下 CREATE TABLE 示例将日期与小时分开。由于 date 是 SQL 中的保留字，因此这一示例使用 day 作为表示日期的分区键的名称。

```
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}/"
)
```

在示例 CREATE TABLE 语句中，小时是一个单独的分区键，配置为整数。小时分区键的配置指定了 0 到 23 的范围，并且当 Athena 生成分区位置时，小时的格式应为两位数。

对 my_ingested_data2 表的查询可能如下所示：

```
SELECT *
FROM my_ingested_data2
WHERE day = '2021/11/09'
AND hour > 3
```

分区键类型和分区投影类型

请注意，第一个 CREATE TABLE 示例中的 datehour 键在分区投影配置中配置为 date，但分区键的类型为 string。第二个示例中的 day 也是如此。分区投影配置中的类型仅告诉 Athena 在生成分区位置时如何格式化值。您指定的类型不会更改分区键的类型 – 在查询中，datehour 和 day 属于 string 类型。

当查询包含与 day = '2021/11/09' 类似的条件时，Athena 会使用分区投影配置中指定的日期格式解析表达式右侧的字符串。在 Athena 验证日期是否在配置范围内后，它会再次使用日期格式将日期作为字符串插入到存储位置模板中。

同样，对于像 day > '2021/11/09' 这样的查询条件，Athena 会解析右侧并生成配置范围内所有匹配日期的列表。然后使用日期格式将每个日期插入到存储位置模板中以创建分区位置列表。

编写与 day > '2021-11-09' 或 day > DATE '2021-11-09' 相同的条件将无效。在第一种情况下，日期格式不匹配（注意连字符而不是正斜杠），在第二种情况下，数据类型不匹配。

使用自定义前缀和动态分区

可以使用自定义前缀和动态分区配置 Firehose。使用这些功能，您可以配置 Amazon S3 键并设置更好地支持您的使用案例的分区方案。您还可以将分区投影与这些分区方案一起使用，然后进行相应的配置。

例如，您可以使用自定义前缀功能获取具有 ISO 格式日期而非默认 yyyy/MM/dd/HH 方案的 Amazon S3 键。

您还可以将自定义前缀与动态分区结合使用，以便从 Firehose 消息中提取 customer_id 之类的属性，如下例所示。

```
prefix/!{timestamp:yyyy}-!{timestamp:MM}-!{timestamp:dd}/!
{partitionKeyFromQuery:customer_id}/
```

使用该 Amazon S3 前缀，Firehose 传输流会将对象写入 s3://DOC-EXAMPLE-BUCKET/prefix/2021-11-01/customer-1234/file.extension 等键。对于像 customer_id 这样的属性，其值可能事先未知，您可以使用分区投影类型 injected 并使用如下所示的 CREATE TABLE 语句：

```
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}/"
)
```

当您查询具有类型 injected 的分区键的表时，您的查询必须包含该分区键的值。对 my_ingested_data3 表的查询可能如下所示：

```
SELECT *
FROM my_ingested_data3
WHERE day BETWEEN '2021-11-01' AND '2021-11-30'
AND customer_id = 'customer-1234'
```

ISO 格式日期

由于 day 分区键的值是 ISO 格式的，您也可以使用 DATE 类型而不是 STRING 类型作为日期分区键，如以下示例所示：

```
PARTITIONED BY (day DATE, customer_id STRING)
```

查询时，此策略允许您在分区键上使用日期函数而无需解析或转换，如以下示例所示：

```
SELECT *
FROM my_ingested_data3
WHERE day > CURRENT_DATE - INTERVAL '7' DAY
AND customer_id = 'customer-1234'
```

> **Note**
>
> 指定 DATE 类型的分区键时，假设您已使用自定义前缀功能创建了日期为 ISO 格式的 Amazon S3 键。如果您使用默认的 Firehose 格式 yyyy/MM/dd/HH，则即使相应表属性的类型为 date，也必须将分区键指定为类型 string，如下例所示：
>
> ```
> PARTITIONED BY (
>   `mydate` string)
> TBLPROPERTIES (
>   'projection.enabled'='true',
>    ...
>   'projection.mydate.type'='date',
>   'storage.location.template'='s3://DOC-EXAMPLE-BUCKET/prefix/${mydate}')
> ```

从查询结果创建表（CTAS）

CREATE TABLE AS SELECT (CTAS) 查询根据另一个查询的 SELECT 语句的结果在 Athena 中创建一个新表。Athena 将由 CTAS 语句创建的数据文件存储在 Amazon S3 中的指定位置。有关语法，请参阅 CREATE TABLE AS。

CREATE TABLE AS 将 CREATE TABLE DDL 语句与 SELECT DML 语句结合在一起，因此严格地说同时包含 DDL 和 DML。但是，请注意，出于服务限额目的，Athena 中的 CTAS 查询被视为 DML。有关 Athena 服务限额的信息，请参阅 服务限额。

使用 CTAS 查询可以：

- 在一个步骤中从查询结果创建表，无需反复查询原始数据集。这样可以更轻松地处理原始数据集。
- 转换查询结果并将表迁移到其他表格式，例如 Apache Iceberg。这可以在 Athena 中提高性能并降低查询成本。有关信息，请参阅创建 Iceberg 表。
- 将查询结果转换为其他存储格式，例如 Parquet 和 ORC。这可以在 Athena 中提高性能并降低查询成本。有关信息，请参阅列式存储格式。
- 创建仅包含所需数据的现有表的副本。

主题

- CTAS 查询的注意事项和限制
- 在控制台中运行 CTAS 查询

- 在 Athena 中进行分区和分桶
- CTAS 查询的示例
- 将 CTAS 和 INSERT INTO 用于 ETL 和数据分析
- 使用 CTAS 和 INSERT INTO 绕过 100 分区限制

CTAS 查询的注意事项和限制

以下章节说明了在 Athena 中使用 CREATE TABLE AS SELECT（CTAS）查询时需注意的考虑因素和限制。

CTAS 查询语法

CTAS 查询语法不同于用于创建表的 CREATE [EXTERNAL] TABLE 语法。请参阅 CREATE TABLE AS。

CTAS 查询与视图

CTAS 查询将新数据写入 Amazon S3 中的指定位置，而视图不会写入任何数据。

CTAS 查询结果的位置

如果您的工作组为查询结果位置覆盖客户端设置，则 Athena 会在位置 s3://DOC-EXAMPLE-BUCKET/tables/*<query-id>*/ 创建表。要查看为工作组指定的查询结果位置，请查看工作组的详细信息。

如果您的工作组不覆盖查询结果位置，则可在 CTAS 查询中使用语法 WITH (external_location ='s3://DOC-EXAMPLE-BUCKET/') 来指定 CTAS 查询结果的存储位置。

> **Note**
>
> external_location 属性必须指定一个空位置。CTAS 查询检查存储桶中的路径位置（前缀）是否为空，如果该位置中已有数据，永远不会覆盖这些数据。要再次使用相同的位置，请删除存储桶中键前缀位置的数据。

如果省略 external_location 语法并且未使用工作组设置，则 Athena 会使用查询结果位置的客户端设置，并在位置 s3://DOC-EXAMPLE-BUCKET/*<Unsaved-or-query-name>*/*<year>*/*<month/<date>*/tables/*<query-id>*/ 创建表。

查找孤立文件

如果 CTAS 或 INSERT INTO 语句失败，则可能会在数据位置留下孤立数据。由于 Athena 在某些情况下不会从存储桶中删除数据或部分数据，您或许会在后续查询中读取这些部分数据。若要查找孤立文件以执行检查或删除操作，您可以使用 Athena 提供的数据清单文件跟踪要写入的文件列表。有关更多信息，请参阅识别查询输出文件和 DataManifestLocation。

已忽略 ORDER BY 子句

在 CTAS 查询中，Athena 会忽略查询的 SELECT 部分中的 ORDER BY 子句。

根据 SQL 规范（ISO 9075 第 2 部分），只有直接包含 ORDER BY 子句的查询表达式才能保证由查询表达式指定的表的行顺序。在任何情况下，SQL 中的表本质上都是无序的，在子查询子句中实施 ORDER BY 会导致查询性能不佳，同时不会生成输出有序。因此，在 Athena CTAS 查询中，无法保证在写入数据时保留 ORDER BY 子句指定的顺序。

用于存储查询结果的格式

如果您未指定数据存储格式，预设情况下，CTAS 查询的结果以 Parquet 格式存储。您可以使用 PARQUET、ORC、AVRO、JSON 和 TEXTFILE 格式存储 CTAS 结果。CTAS TEXTFILE 格式不支持多字符分隔符。CTAS 查询不需要指定 SerDe 来解释格式转换。请参阅 Example: Writing query results to a different format。

压缩格式

GZIP 压缩用于 JSON 和 TEXTFILE 格式的 CTAS 查询结果。对于 Parquet，您可以使用 GZIP 或者 SNAPPY，默认为 GZIP。对于 ORC，您可以使用 LZ4、SNAPPY、ZLIB，或者 ZSTD，默认为 ZLIB。有关指定压缩的 CTAS 示例，请参阅 Example: Specifying data storage and compression formats。有关 Athena 中压缩的更多信息，请参阅 Athena 压缩支持。

分区和存储桶限制

您可以对 CTAS 查询的结果数据进行分区和分桶。有关更多信息，请参阅 在 Athena 中进行分区和分桶。使用 CTAS 创建分区表时，Athena 的写入限制为 100 个分区。

在指定目标表的属性的 WITH 子句末尾包括分区和分桶谓词。有关更多信息，请参阅Example: Creating bucketed and partitioned tables 和在 Athena 中进行分区和分桶。

有关绕过 100 个分区的限制的信息，请参阅使用 CTAS 和 INSERT INTO 绕过 100 分区限制。

加密

您可以加密 Amazon S3 中的 CTAS 查询结果，其方法类似于在 Athena 中加密其他查询结果。有关更多信息，请参阅 加密在 Amazon S3 中存储的 Athena 查询结果。

预期存储桶拥有者

对于 CTAS 语句，预期存储桶拥有者设置不适用于 Amazon S3 中的目标表位置。预期存储桶拥有者设置仅适用于您为 Athena 查询结果指定的 Amazon S3 输出位置。有关更多信息，请参阅 使用 Athena 控制台指定查询结果位置。

数据类型

CTAS 查询的列数据类型与为原始查询指定的类型相同。

在控制台中运行 CTAS 查询

在 Athena 控制台中，您可以通过另一个查询创建 CTAS 查询。

通过另一个查询创建 CTAS 查询

1. 在 Athena 控制台查询编辑器中运行查询。
2. 在查询编辑器底部，选择 Create（创建）选项，然后选择 Table from query（根据查询创建表）。
3. 在 Create table as select（根据选择创建表）表单中，填写字段，如下所示：

    a. 对于 Table name（表名称），指定新表的名称。仅使用小写和下划线，例如 `my_select_query_parquet`。

    b. 对于 Database configuration（数据库配置），使用选项选择一个现有的数据库或创建一个数据库。

    c. （可选）在 Result configuration（结果配置）中，对于 Location of CTAS query results（CTAS 查询结果的位置），如果您的工作组查询结果位置设置未覆盖此选项，请执行以下操作之一：

    - 在搜索框中输入一个现有 S3 位置的路径，或选择 Browse S3（浏览 S3）以从列表中选择一个位置。
    - 选择 View（查看）以打开 Amazon S3 控制台的 Buckets （存储桶）页面，您可以在其中查看有关现有存储桶的更多信息并进行选择，或者使用自己的设置创建一个存储桶。

您需要在 Amazon S3 中指定一个将用于输出数据的空位置。如果指定位置中已存在数据，则查询会失败并出现错误。

如果工作组的查询结果位置设置覆盖了此位置设置，则 Athena 会在位置 `s3://DOC-EXAMPLE-BUCKET/tables/query_id/` 中创建您的表。

d. 对于 Data format（数据格式），指定您的数据将采用的格式。

- 表类型 – Athena 中的默认表类型是 Apache Hive。
- 文件格式 – 可选择 CSV、TSV、JSON、Parquet 或 ORC 等选项。有关 Parquet 和 ORC 格式的信息，请参阅 列式存储格式。
- 写入压缩 –（可选）选择一个压缩格式。Athena 支持多种用于读取和写入数据的压缩格式，例如从使用多种压缩格式的表中进行读取。例如，当某些 Parquet 文件使用 Snappy 压缩而其他 Parquet 文件使用 GZIP 压缩时，Athena 可以成功读取使用 Parquet 文件格式的表中的数据。同样的原则适用于 ORC、文本文件和 JSON 存储格式。有关更多信息，请参阅 Athena 压缩支持。
- 分区 –（可选）选择要分区的列。通过对数据进行分区，您可以限制每个查询扫描的数据量，从而提高性能并降低成本。您可按任何键对数据进行分区。有关更多信息，请参阅 在 Athena 中对数据进行分区。
- 桶 –（可选）选择要分桶的列。分桶是一种在单个分区内基于特定列将数据进行分组的技术。这些列被称为 桶键。通过将相关数据分组到单个桶（分区内的一个文件）中，您可以显著减少 Athena 扫描的数据量，从而提高查询性能并降低成本。有关更多信息，请参阅 在 Athena 中进行分区和分桶。

e. 对于 Preview table query（预览表查询），检查您的查询。有关查询语法，请参阅 CREATE TABLE AS。

f. 选择创建表。

使用 SQL 模板创建 CTAS 查询

使用 `CREATE TABLE AS SELECT` 模板在查询编辑器中创建 CTAS 查询。

1. 在 Athena 控制台中，选择 Tables and views（表和视图）旁边的 Create table（创建表），然后选择 CREATE TABLE AS SELECT。这会使用具有占位符值的 CTAS 查询填充查询编辑器。
2. 在查询编辑器中，根据需要编辑查询。有关查询语法，请参阅 CREATE TABLE AS。
3. 选择运行。

有关示例，请参阅CTAS 查询的示例。

在 Athena 中进行分区和分桶

分区和分桶是减少运行查询时 Athena 必须扫描的数据量的两种方法。分区与分区互补，可以一起使用。可通过减少扫描的数据量提高性能并降低成本。有关 Athena 查询性能的一般准则，请参阅 Amazon Athena 的十大性能优化技巧。

什么是分区？

分区指根据数据的特定属性将数据分到 Amazon S3 上的目录（或“前缀”）中。此类属性称为分区键。常见的分区键是日期或其他时间单位，例如年或月。但是，可按多个键一个数据集进行分区。例如，有关产品销售的数据可按日期、产品类别和市场进行分区。

确定分区方式

分区键的理想候选项为在查询中始终或经常使用且基数较低的属性。需要权衡分区数量，不能过多也不能过少。如果分区过多，文件数会增加，从而产生开销。筛选分区本身也会产生一些开销。如果分区过少，查询通常必须扫描更多数据。

创建分区表

对数据集进行分区后，可以在 Athena 中创建分区表。分区表是具有分区键的表。使用 CREATE TABLE 时，可以向表添加分区。使用 CREATE TABLE AS 时，在 Amazon S3 上创建的分区会自动添加至表。

在 CREATE TABLE 语句中，可在 PARTITIONED BY (*column_name data_type*) 子句中指定分区键。在 CREATE TABLE AS 语句中，可在 WITH (partitioned_by = ARRAY['*partition_key*']) 子句中或 Iceberg 表的 WITH (partitioning = ARRAY['*partition_key*']) 中指定分区键。出于性能考虑，分区键的类型应始终为 STRING。有关更多信息，请参阅 使用字符串作为分区键的数据类型。

有关其他 CREATE TABLE 和 CREATE TABLE AS 语法详细信息，请参阅 CREATE TABLE 和 CTAS 表属性。

查询分区表

当您查询分区表时，Athena 使用查询中的谓词来筛选分区列表。然后，它使用匹配分区的位置来处理找到的文件。通过不读取与查询谓词不匹配的分区中的数据，Athena 可以有效地减少扫描的数据量。

示例

假设您拥有一个按 sales_date 和 product_category 分区的表，您希望了解一周内特定类别的总收入。您可以在 sales_date 和 product_category 列中包含谓词，以确保 Athena 仅扫描最低数量的数据，如以下示例所示。

```
SELECT SUM(amount) AS total_revenue
FROM sales
WHERE sales_date BETWEEN '2023-02-27' AND '2023-03-05'
AND product_category = 'Toys'
```

假设您的数据集按日期进行分区，同时具有精细的时间戳。

对于 Iceberg 表，您可以将分区键声明为与列有关系，但是对于 Hive 表，查询引擎对列和分区键之间的关系一无所知。因此，您必须在列和查询的分区键中包含一个谓词，以确保查询仅扫描所需数量的数据。

例如，假设上一个示例中的 sales 表还包含一个数据类型为 TIMESTAMP 的 sold_at 列。如果您只需要特定时间范围内的收入，查询编写如下：

```
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'
```

有关查询 Hive 和 Iceberg 表之间的差异的更多信息，请参阅 如何为同时按时间分区的时间戳字段编写查询。

什么是分桶？

分桶是一种将数据集记录分为称为存储桶的类别的方法。

这里的存储桶和分桶的含义与 Amazon S3 存储桶不同，不应与之混淆。在数据分桶中，具有相同属性值的记录会进入同一存储桶中。记录在存储桶之间尽可能均匀地分布，因此每个存储桶的数据量大致相同。

实际上，存储桶就是文件，哈希函数决定了记录进入哪个存储桶。分桶数据集的每个分区的每个存储桶将包含一个或多个文件。文件所属的存储桶以文件名编码。

分桶优势

如果数据集按特定属性进行分桶，并且您想要检索该属性具有特定值的记录，建议使用分桶。由于数据已分桶，Athena 可以使用该值来确定要查看的文件。例如，假设一个数据集按 `customer_id` 进行分桶，并且您想要查找特定客户的所有记录。Athena 确定包含这些记录的存储桶，并且仅读取该存储桶中的文件。

如果您的列具有高基数（即具有许多不同值）、分布均匀且您经常查询特定值，这些都适合作为分桶的依据。

> **Note**
>
> Athena 不支持使用 `INSERT INTO` 向分桶表添加新记录。

对分桶列进行筛选所支持的数据类型

您可以在具有特定数据类型的分桶列中添加筛选器。Athena 在具有以下数据类型的分桶列上支持筛选：

- BOOLEAN
- BYTE
- DATE
- DOUBLE
- FLOAT
- INT
- LONG
- SHORT
- string
- VARCHAR

Hive 和 Spark 支持

Athena 引擎版本 2 支持使用 Hive 存储桶算法对数据集进行分区，而 Athena 引擎版本 3 也支持 Apache Spark 分桶算法。Hive 分桶为默认设置。如果您的数据集使用 Spark 算法进行分桶，使用 `TBLPROPERTIES` 子句将 `bucketing_format` 属性值设置为 `spark`。

> **Note**
>
> 在一个 CREATE TABLE AS SELECT（CTAS）查询中，Athena 的分区数量不能超过 100 个。同样，您可以使用 INSERT INTO 语句向目标表最多添加 100 个分区。100 个分区的限制仅在对表进行分桶和分区时适用。
> 如果超过此限制，则可能会收到错误消息 HIVE_TOO_MANY_OPEN_PARTITIONS: Exceeded limit of 100 open writers for partitions/buckets（HIVE_TOO_MANY_OPEN_PARTITIONS：分区/存储桶超过 100 个打开的写入程序限制）。要绕过此限制，您可以使用一个 CTAS 语句和一系列 INSERT INTO 语句，每个后一种语句将创建或插入不超过 100 个分区。有关更多信息，请参阅 使用 CTAS 和 INSERT INTO 绕过 100 分区限制。

分桶 CREATE TABLE 示例

要为现有分桶数据集创建表，使用 CLUSTERED BY (*column*) 子句后跟 INTO *N* BUCKETS 子句。INTO *N* BUCKETS 子句指定用于对数据进行分桶的存储桶数。

在以下 CREATE TABLE 示例中，使用 Spark 算法按 customer_id 将 sales 数据集分为 8 个存储桶。CREATE TABLE 语句使用 CLUSTERED BY 和 TBLPROPERTIES 子句相应地设置属性。

```
CREATE EXTERNAL TABLE sales (...)
...
CLUSTERED BY (`customer_id`) INTO 8 BUCKETS
...
TBLPROPERTIES (
  'bucketing_format' = 'spark'
)
```

分桶 CREATE TABLE AS（CTAS）示例

要使用 CREATE TABLE AS 指定分桶，使用 bucketed_by 和 bucket_count 参数，如以下示例所示。

```
CREATE TABLE sales
WITH (
  ...
  bucketed_by = ARRAY['customer_id'],
  bucket_count = 8
)
AS SELECT ...
```

分桶查询示例

以下示例查询将查找特定客户在一周内购买的产品的名称。

```
SELECT DISTINCT product_name
FROM sales
WHERE sales_date BETWEEN '2023-02-27' AND '2023-03-05'
AND customer_id = 'c123'
```

如果此表按 `sales_date` 进行分区并按 `customer_id` 进行分桶，则 Athena 可以计算出客户记录所在的存储桶。Athena 最多读取每个分区中的一个文件。

其他 资源

有关创建分桶表和分区表的 `CREATE TABLE AS` 示例，请参阅示例：创建分桶表和分区表。

CTAS 查询的示例

使用以下示例创建 CTAS 查询。有关 CTAS 语法的信息，请参阅CREATE TABLE AS。

本节内容：

Example 示例：通过选择所有列复制表

以下示例通过复制表的所有列来创建表：

```
CREATE TABLE new_table AS
SELECT *
FROM old_table;
```

在同一个示例的下列变化中，您的 SELECT 语句还包括 WHERE 子句。在这种情况下，查询将只从表中选择满足 WHERE 子句的行：

```
CREATE TABLE new_table AS
SELECT *
FROM old_table
WHERE condition;
```

Example 示例：从一个或多个表选择特定列

以下示例创建运行在其他表的一组列上的新查询：

```
CREATE TABLE new_table AS
SELECT column_1, column_2, ... column_n
FROM old_table;
```

同一个示例的此变化从多个表的特定列创建新表：

```
CREATE TABLE new_table AS
SELECT column_1, column_2, ... column_n
FROM old_table_1, old_table_2, ... old_table_n;
```

Example 示例：创建现有表的空副本

以下示例使用 WITH NO DATA 创建空的新表，该表与原始表具有相同架构：

```
CREATE TABLE new_table
AS SELECT *
FROM old_table
WITH NO DATA;
```

Example 示例：指定数据存储和压缩格式

借助 CTAS，您可以使用一种存储格式的源表创建不同存储格式的另一个表。

使用 format 属性以将 ORC、PARQUET、AVRO、JSON 或 TEXTFILE 指定为新表的存储格式。

对于 PARQUET、ORC、TEXTFILE 和 JSON 存储格式，使用 write_compression 属性指定新表数据的压缩格式。有关每种文件格式支持的压缩格式的信息，请参阅 Athena 压缩支持。

以下示例指定表 new_table 中的数据以 Parquet 格式存储并使用 Snappy 压缩。Parquet 的默认压缩格式为 GZIP。

```
CREATE TABLE new_table
WITH (
      format = 'Parquet',
      write_compression = 'SNAPPY')
AS SELECT *
FROM old_table;
```

以下示例指定表 new_table 中的数据以 ORC 格式存储并使用 Snappy 压缩。ORC 的默认压缩格式为 ZLIB。

```
CREATE TABLE new_table
WITH (format = 'ORC',
      write_compression = 'SNAPPY')
AS SELECT *
FROM old_table ;
```

以下示例指定表 new_table 中的数据以 Textfile 格式存储并使用 Snappy 压缩。Textfile 和 JSON 格式的默认压缩格式都是 GZIP。

```
CREATE TABLE new_table
WITH (format = 'TEXTFILE',
      write_compression = 'SNAPPY')
AS SELECT *
FROM old_table ;
```

Example 示例：将查询结果写入不同格式

以下 CTAS 查询从 old_table 中选择能够以 CSV 或其他格式存储的所有记录，并创建一个新表，其中包含以 ORC 格式保存到 Amazon S3 的基础数据：

```
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 示例：创建未分区表

以下示例创建未分区的表。表数据以不同格式存储。其中一些示例指定了外部位置。

以下示例创建使用文本文件存储结果的 CTAS 查询：

```
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;
```

在以下示例中，结果以 Parquet 格式存储，并使用默认结果位置：

```
CREATE TABLE ctas_parquet_unpartitioned
WITH (format = 'PARQUET')
AS SELECT key1, name1, comment1
FROM table1;
```

在以下查询中，表以 JSON 格式存储，并从原始表的结果中选择特定列：

```
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;
```

在下面的示例中，格式为 ORC：

```
CREATE TABLE ctas_orc_unpartitioned
WITH (
     format = 'ORC')
AS SELECT key1, name1, comment1
FROM table1;
```

在下面的示例中，格式为 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 示例：创建分区表

以下示例显示了对不同存储格式的分区表的 CREATE TABLE AS SELECT 查询，在 WITH 子句中使用 partitioned_by 和其他属性。有关语法，请参阅 CTAS 表属性。有关为分区选择列的更多信息，请参阅在 Athena 中进行分区和分桶。

> **Note**
>
> 在 SELECT 语句中列列表的结尾列出分区列。您可以按多个列进行分区，并拥有多达 100 个唯一的分区和存储桶组合。例如，如果未指定存储桶，则可以有 100 个分区。

```
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 示例：创建分桶和分区表

以下示例显示同时使用分区和分桶在 Amazon S3 中存储查询结果的 CREATE TABLE AS SELECT 查询。表结果按照不同列分区和分桶。Athena 支持最多 100 个唯一的存储桶和分区组合。例如，如果您

创建包含五个存储桶的表，则支持 20 个分区，每个分区包含五个存储桶。有关语法，请参阅 CTAS 表属性。

有关为分桶选择列的信息，请参阅在 Athena 中进行分区和分桶。

```
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 示例：使用 Parquet 数据创建 Iceberg 表

以下示例将创建一个包含有 Parquet 数据文件的 Iceberg 表。文件按月根据 table1 中的 dt 列进行分区。该示例更新了表的保留属性，因此默认情况下，表中每个分支上都会保留 10 个快照。过去 7 天内的快照也会保留。有关 Athena 中 Iceberg 表属性的更多信息，请参阅 表属性。

```
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 示例：使用 Avro 数据创建 Iceberg 表

以下示例将创建一个按 key1 分区的包含有 Avro 数据文件的 Iceberg 表。

```
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;
```

将 CTAS 和 INSERT INTO 用于 ETL 和数据分析

在 Athena 中使用 Create Table as Select (CTAS) 和 INSERT INTO 语句可将数据提取、转换和加载 (ETL) 到 Amazon S3 中以进行数据处理。本主题说明了如何使用这些语句对数据集进行分区并将其转换为列式数据格式，以对其进行优化来进行数据分析。

CTAS 语句使用标准 SELECT 查询来创建新表。您可以使用 CTAS 语句创建数据子集以进行分析。在一个 CTAS 语句中，您可以对数据进行分区、指定压缩并将数据转换为列状格式，如 Apache Parquet 或 Apache ORC。在运行 CTAS 查询时，该查询创建的表和分区将自动添加到 AWS Glue Data Catalog。这使得它创建的新表和分区立即对后续查询可用。

INSERT INTO 语句基于在源表上运行的 SELECT 查询语句将新行插入到目标表中。您可以使用 INSERT INTO 语句通过 CTAS 支持的所有转换来转换 CSV 格式的源表数据并将其加载到目标表数据中。

概述

在 Athena 中，使用 CTAS 语句执行数据的初始批量转换。然后，使用多个 INSERT INTO 语句对 CTAS 语句所创建的表进行增量更新。

步骤

- 步骤 1：基于原始数据集创建表
- 步骤 2：使用 CTAS 对数据进行分区、转换和压缩
- 步骤 3：使用 INSERT INTO 添加数据
- 步骤 4：衡量性能和成本差异

步骤 1：基于原始数据集创建表

本主题中的示例使用 Simple Storage Service (Amazon S3) 可读取的公开发布的 NOAA global historical climatology network daily (GHCN-d) 数据集的子集。Amazon S3 上的数据具有以下特性。

```
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
```

原始数据存储在不带分区的 Amazon S3 中。文件中的数据采用 CSV 格式，如下所示。

```
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
```

此示例中的文件大小相对较小。通过将这些文件合并为更大的文件，可以减少文件总数，从而优化查询执行的性能。可以使用 CTAS 和 INSERT INTO 语句来增强查询性能。

创建基于示例数据集的数据库和表

1. 在 Athena 控制台中，选择 US East (N. Virginia) [美国东部（弗吉尼亚北部）]AWS 区域。请务必在 `us-east-1` 中运行本教程中的所有查询。
2. 在 Athena 查询编辑器中，运行 CREATE DATABASE 命令以创建数据库。

   ```
   CREATE DATABASE blogdb
   ```

3. 运行以下语句以创建表。

   ```
   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/'
   ```

步骤 2：使用 CTAS 对数据进行分区、转换和压缩

创建表后，您可以使用单个 CTAS 语句通过 Snappy 压缩将数据转换为 Parquet 格式，并按年份对数据进行分区。

您在步骤 1 中创建的表具有一个 `date` 字段，其数据格式化为 YYYYMMDD（例如，`20100104`）。由于将按 `year` 对新表进行分区，因此，以下过程中的示例语句使用 Presto 函数 `substr("date",1,4)` 从 `date` 字段中提取 `year` 值。

使用 Snappy 压缩将数据转换为 Parquet 格式（按年份进行分区）

- 运行以下 CTAS 语句，并将 *your-bucket* 替换为您的 Amazon S3 存储桶位置。

```
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**
>
> 在此示例中，您创建的表仅包含 2015 年至 2019 年的数据。在步骤 3 中，使用 INSERT INTO 命令将新数据添加到此表。

在查询完成后，请使用以下过程验证您在 CTAS 语句中指定的 Amazon S3 位置中的输出。

查看 CTAS 语句所创建的分区和 parquet 文件

1. 要显示创建的分区，请运行以下 AWS CLI 命令。请务必包含最后的正斜杠 (/)。

```
aws s3 ls s3://DOC-EXAMPLE-BUCKET/optimized-data/
```

输出将显示分区。

```
      PRE year=2015/
      PRE year=2016/
      PRE year=2017/
      PRE year=2018/
      PRE year=2019/
```

2. 要查看 Parquet 文件，请运行以下命令。请注意，| head -5 选项（该选项将输出限制为前五个结果）在 Windows 上不可用。

```
aws s3 ls s3://DOC-EXAMPLE-BUCKET/optimized-data/ --recursive --human-readable |
 head -5
```

输出与以下内容类似。

```
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
```

步骤 3：使用 INSERT INTO 添加数据

在步骤 2 中，您已使用 CTA 创建一个表，其中包含 2015 年至 2019 年的分区。但原始数据集还包含 2010 年至 2014 年的数据。现在，您使用 INSERT INTO 语句添加该数据。

使用一个或多个 INSERT INTO 语句向表添加数据

1. 运行以下 INSERT INTO 命令，并在 WHERE 子句中指定 2015 年之前的年份。

```
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. 再次运行 `aws s3 ls` 命令，并使用以下语法。

```
aws s3 ls s3://DOC-EXAMPLE-BUCKET/optimized-data/
```

输出将显示新分区。

```
      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. 要查看通过使用 Parquet 格式的压缩和列式存储获得的数据集的大小减少量，请运行以下命令。

```
aws s3 ls s3://DOC-EXAMPLE-BUCKET/optimized-data/ --recursive --human-readable --
summarize
```

以下结果表明，使用 Snappy 压缩进行 parquet 操作后的数据集的大小为 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. 如果将更多的 CSV 数据添加到原始表中，则可使用 INSERT INTO 语句将该数据添加到 parquet 表中。例如，如果您有 2020 年的新数据，则可运行以下 INSERT INTO 语句。该语句可将数据和相关分区添加到 new_parquet 表中。

```
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**
>
> INSERT INTO 语句支持向目标表写入最多 100 个分区。不过，要添加 100 个以上的分区，您可以运行多个 INSERT INTO 语句。有关更多信息，请参阅 使用 CTAS 和 INSERT INTO 绕过 100 分区限制。

步骤 4：衡量性能和成本差异

在转换数据后，您可以通过对新表和旧表运行相同的查询并比较结果来衡量性能提升和成本节省。

> **Note**
>
> 有关 Athena 每查询成本信息，请参阅 Amazon Athena 定价。

衡量性能提升和成本差异

1. 对原始表运行以下查询。此查询将查找每个年份值的不同 ID 的数量。

```
SELECT substr("date",1,4) as year,
       COUNT(DISTINCT id)
FROM original_csv
GROUP BY 1 ORDER BY 1 DESC
```

2. 请注意查询运行的时间和扫描的数据量。
3. 对新表运行相同的查询，并记下查询运行时和扫描的数据量。

```
SELECT year,
  COUNT(DISTINCT id)
FROM new_parquet
GROUP BY 1 ORDER BY 1 DESC
```

4. 比较结果并计算性能和成本差异。以下示例结果表明，与对旧表执行的测试查询相比，对新表执行的测试查询的速度更快、成本更低。

表	运行时	扫描的数据
原始	16.88 秒	11.35 GB
New	3.79 秒	428.05 MB

5. 对原始表运行以下示例查询。该查询将计算 2018 年地球上的平均最高温度（摄氏度）、平均最低温度（摄氏度）和平均降雨量（毫米）。

```
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. 请注意查询运行的时间和扫描的数据量。
7. 对新表运行相同的查询，并记下查询运行时和扫描的数据量。

```
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. 比较结果并计算性能和成本差异。以下示例结果表明，与对旧表执行的测试查询相比，对新表执行的测试查询的速度更快、成本更低。

表	运行时	扫描的数据
原始	18.65 秒	11.35 GB
New	1.92 秒	68 MB

Summary

本主题介绍了如何在 Athena 中使用 CTAS 和 INSERT INTO 语句来执行 ETL 操作。您使用一个 CTAS 语句执行了第一组转换，通过 Snappy 压缩将数据转换为了 Parquet 格式。该 CTAS 语句还将数据集从未分区的数据集转换为已分区的数据集。这减小了其大小，并降低了查询的运行成本。在提供新的数据时，您可以使用 INSERT INTO 语句转换数据并将其加载到使用 CTAS 语句创建的表中。

使用 CTAS 和 INSERT INTO 绕过 100 分区限制

Athena 的每个 `CREATE TABLE AS SELECT`（CTAS）查询的分区数不能超过 100 个。同样，您可以使用 INSERT INTO 语句向目标表添加最多 100 个分区。

如果超过此限制，则可能会收到错误消息 HIVE_TOO_MANY_OPEN_PARTITIONS: Exceeded limit of 100 open writers for partitions/buckets（HIVE_TOO_MANY_OPEN_PARTITIONS：分区/存储桶超过 100 个打开的写入程序限制）。要绕过此限制，您可以使用一个 CTAS 语句和一系列 `INSERT INTO` 语句，每个后一种语句将创建或插入不超过 100 个分区。

本主题中的示例使用名为 `tpch100` 的数据库，其数据驻留在 Amazon S3 存储桶位置 s3://DOC-EXAMPLE-BUCKET/ 中。

使用 CTAS 和 INSERT INTO 创建带 100 多个分区的表

1. 使用 `CREATE EXTERNAL TABLE` 语句创建一个基于所需字段进行分区的表。

   以下示例语句按列 `l_shipdate` 对数据进行分区。该表具有 2525 个分区。

```
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. 运行类似于下文的 SHOW PARTITIONS *<table_name>* 命令以列出分区。

```
SHOW PARTITIONS lineitem_parq_partitioned
```

以下是部分示例结果。

```
/*
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. 运行 CTAS 查询以创建已分区的表。

   以下示例创建一个名为 my_lineitem_parq_partitioned 的表，并使用 WHERE 子句将 DATE 限制为 1992-02-01 之前的日期。由于示例数据集的开始日期为 1992 年 1 月，因此仅创建 1992 年 1 月的分区。

```
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. 运行 SHOW PARTITIONS 命令可验证表是否包含所需的分区。

```
SHOW PARTITIONS my_lineitem_parq_partitioned;
```

   示例中的分区对应于 1992 年 1 月内的日期。

```
/*
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. 使用 INSERT INTO 语句可向表添加分区。

   以下示例为 1992 年 2 月内的日期添加分区。

```
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. 再次运行 SHOW PARTITIONS。

```
SHOW PARTITIONS my_lineitem_parq_partitioned;
```

示例表现在具有对应于 1992 年 1 月和 2 月内日期的分区。

```
/*
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. 继续使用 INSERT INTO 语句，其中每个语句读取和添加的分区数不超过 100 个。继续操作，直到您达到所需的分区数。

> **Important**
>
> 在设置 WHERE 条件时，请确保查询不会重叠。否则，某些分区可能具有重复数据。

SerDe 引用

Athena 支持多个 SerDe 库，用于分析来自不同数据格式的数据，例如 CSV、JSON、Parquet 和 ORC。Athena 不支持自定义 SerDes。

主题

使用 SerDe

SerDe（串行器/解串器）是一种可供 Athena 与各种格式的数据交互的方式。

它是您指定的 SerDe，而不是用于定义表架构的 DDL。换言之，SerDe 可以覆盖您在创建表时在 Athena 中指定的 DDL 配置。

在查询中使用 SerDe

要在 Athena 中创建表时使用 SerDe，请使用以下方法之一：

- 指定 `ROW FORMAT DELIMITED`，然后使用 DDL 语句指定字段分隔符，如下例所示。当您指定 `ROW FORMAT DELIMITED`，Athena 在预设情况下使用 LazySimpleSerDe。

```
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
ESCAPED BY '\\'
COLLECTION ITEMS TERMINATED BY '|'
MAP KEYS TERMINATED BY ':'
```

  对于示例 `ROW FORMAT DELIMITED`，请参阅以下主题：

  用于 CSV、TSV 和自定义分隔文件的 LazySimpleSerDe

  查询 Amazon CloudFront 日志

  查询 Amazon EMR 日志

  查询 Amazon VPC 流日志

  将 CTAS 和 INSERT INTO 用于 ETL 和数据分析

- 使用 ROW FORMAT SERDE 显式在向表中读取和写入数据时 Athena 应使用的 SerDe 类型。以下示例指定 LazySimpleSerDe。要指定分隔符，请使用 WITH SERDEPROPERTIES。WITH SERDEPROPERTIES 指定的属性对应于 ROW FORMAT DELIMITED 示例中的单独的语句（如 FIELDS TERMINATED BY）。

```
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
WITH SERDEPROPERTIES (
'serialization.format' = ',',
'field.delim' = ',',
'collection.delim' = '|',
'mapkey.delim' = ':',
'escape.delim' = '\\'
)
```

对于示例 ROW FORMAT SERDE，请参阅以下主题：

Avro SerDe

Grok SerDe

JSON SerDe 库

用于处理 CSV 的 OpenCSVSerDe

Regex SerDe

支持的 SerDes 和数据格式

Athena 支持采用以下格式创建表和查询数据：CSV、TSV、自定义分隔符和 JSON 格式，Hadoop 相关格式（ORC、Apache Avro 和 Parquet），Logstash 日志、AWS CloudTrail 日志和 Apache WebServer 日志。

> **Note**
>
> 本节中列出的格式由 Athena 用于读取数据。有关 Athena 在运行 CTAS 查询时用于写入数据的格式，请参阅 从查询结果创建表（CTAS）。

要在 Athena 中采用这些格式创建表和查询数据，请指定一个串行器-解串器类 (SerDe)，以便 Athena 知道使用哪种格式以及如何解析数据。

此表列出了 Athena 中支持的数据格式及其对应的 SerDe 库。

SerDe 是一个自定义库，可以告诉 Athena 使用的数据目录如何处理数据。要指定 SerDe 类型，可以在 Athena 中的 `CREATE TABLE` 语句的 `ROW FORMAT` 部分中显式列出该类型。在某些情况下，您可以省略 SerDe 名称，因为 Athena 在默认情况下对特定类型的数据格式使用某些 SerDe 类型。

支持的数据格式和 SerDes

Data format（数据格式）	描述	Athena 中支持的 SerDe 类型
Amazon Ion	Amazon Ion 是一种类型丰富、自描述的数据格式，是 JSON 的超集，由 Amazon 开发和开源。	使用 Amazon Ion Hive SerDe。
Apache Avro	用于在 Hadoop 中存储数据的一种格式，它对记录值使用基于 JSON 的架构。	使用 Avro SerDe。
Apache Parquet	一种用于在 Hadoop 中列式存储数据的格式。	使用 Parquet SerDe 和 SNAPPY 压缩。
Apache WebServer 日志	一种用于在 Apache WebServer 中存储日志的格式。	使用 Grok SerDe 或 Regex SerDe。
CloudTrail 日志	一种用于在 CloudTrail 中存储日志的格式。	• 使用 Hive JSON SerDe。有关更多信息，请参阅 查询 AWS CloudTrail 日志。
CSV（逗号分隔的值）	对于 CSV 格式的数据，每个行均表示一个数据记录，每个记录包含一个或多个字段，以逗号分隔。	• 如果数据不包括括在引号中的值或其使用 `java.sql.Timestamp` 格式，请使用 用于 CSV、TSV 和自定义分隔文件的 LazySimpleSerDe。 • 当数据包含引号中的值或为 用于处理 CSV 的 OpenCSVSerDe 使用

Data format（数据格式）	描述	Athena 中支持的 SerDe 类型
		UNIX 数字格式时（例如，`TIMESTAMP` ），请使用 `1564610311` 。
自定义分隔的	对于采用此格式的数据，每一行表示一个数据记录，记录由自定义的单字符分隔符分隔。	使用 用于 CSV、TSV 和自定义分隔文件的 LazySimpleSerDe 并指定自定义单字符分隔符。
JSON（JavaScript 对象表示法）	对于 JSON 数据，每个行均表示一个数据记录，每个记录包含属性-值对和数组，以逗号分隔。	• 使用 Hive JSON SerDe。 • 使用 OpenX JSON SerDe。
Logstash 日志	一种用于在 Logstash 中存储日志的格式。	使用 Grok SerDe。
ORC（优化的行列式）	一种用于优化 Hive 数据列式存储的格式。	使用 ORC SerDe 和 ZLIB 压缩。
TSV（制表符分隔的值）	对于 TSV 格式的数据，每个行均表示一个数据记录，每个记录包含一个或多个字段，以制表符分隔。	使用 用于 CSV、TSV 和自定义分隔文件的 LazySimpleSerDe 并将分隔符指定为 `FIELDS TERMINATED BY '\t'`。

主题

- Amazon Ion Hive SerDe
- Avro SerDe
- Grok SerDe
- JSON SerDe 库
- 用于 CSV、TSV 和自定义分隔文件的 LazySimpleSerDe
- 用于处理 CSV 的 OpenCSVSerDe
- ORC SerDe

- Parquet SerDe
- Regex SerDe

Amazon Ion Hive SerDe

您可以使用 Amazon Ion Hive SerDe 查询以 Amazon Ion 格式存储的数据。Amazon Ion 是一种类型丰富、自描述的开源数据格式。Amazon Quantum Ledger Database（Amazon QLDB）等服务和开源 SQL 查询语言 PartiQL 使用Amazon Ion 格式。

Amazon Ion 具有可互换的二进制格式和文本格式。此功能结合了文本的易用性和二进制编码的效率。

若要从 Athena 查询 Amazon Ion 数据，您可以使用 Amazon Ion Hive SerDe，其对 Amazon Ion 数据进行序列化和反序列化操作。反序列化允许您对 Amazon Ion 数据运行查询，或读取数据以便以不同的格式（如 Parquet 或 ORC）写出。序列化允许您通过使用 `CREATE TABLE AS SELECT`（CTAS）或 `INSERT INTO` 查询从现有表中复制数据来生成 Amazon Ion 格式的数据。

> **Note**
>
> 由于 Amazon Ion 是 JSON 的超集，因此您可以使用 Amazon Ion Hive SerDe 查询非 Amazon Ion JSON 数据集。与其他 JSON SerDe 库不同，Amazon Ion SerDe 不希望每行数据都在一行上。如果您想查询“漂亮打印”格式的 JSON 数据集，或以其他方式采用换行符将字段拆分为一行，则此功能非常有用。

有关使用 Athena 查询 Amazon Ion 的其他信息和示例，请参阅使用 Amazon Athena 分析 Amazon Ion 数据集。

SerDe 名称

- com.amazon.ionhiveserde.IonHiveSerDe

注意事项和限制

- 重复的字段 – Amazon Ion 结构是有序的并支持重复字段，而 Hive `STRUCT<>` 和 `MAP<>` 不支持。因此，当您将 Amazon Ion 结构中的重复字段反序列化时，将不确定地选择单个值，而忽略其他值。
- 不支持外部符号表 – 目前，Athena 不支持外部符号表或下列 Amazon Ion Hive SerDe 属性：
  - `ion.catalog.class`

  - `ion.catalog.file`
  - `ion.catalog.url`
  - `ion.symbol_table_imports`
- 文件扩展名 – Amazon Ion 使用文件扩展名来确定用于反序列化 Amazon Ion 文件的压缩编解码器。因此，压缩文件必须具有与使用的压缩算法相对应的文件扩展名。例如，如果使用 ZSTD，相应的文件扩展名为 `.zst`。
- 同类数据 – Amazon Ion 对可用于特定字段值的数据类型没有任何限制。例如，两个不同的 Amazon Ion 文档可能有一个名称相同但数据类型不同的字段。但是，由于 Hive 使用架构，所以您提取到单个 Hive 列的所有值都必须具有相同的数据类型。
- 映射密钥类型限制 – 您将其他格式的数据序列化到 Amazon Ion 时，请确保映射密钥类型为 `STRING`、`VARCHAR` 或者 `CHAR`。尽管 Hive 允许您使用任何原始数据类型作为映射密钥，Amazon Ion 符号必须是字符串类型。
- 联合类型 – Athena 目前不支持 Hive 联合类型。
- 双精度数据类型 – Amazon Ion 目前不支持 `double` 数据类型。

主题

使用 CREATE TABLE 创建 Amazon Ion 表

要使用以 Amazon Ion 格式存储的数据创建 Athena 表，您可以在 CREATE TABLE 语句中使用以下方法之一：

- 指定 `STORED AS ION`。在此用法中，您无需明确指定 Amazon Ion Hive SerDe。这是一种更直接的选择。
- 在 `ROW FORMAT SERDE`、`INPUTFORMAT` 和 `OUTPUTFORMAT` 字段中指定 Amazon Ion 类路径。

您还可以使用 `CREATE TABLE AS SELECT`（CTAS）语句在 Athena 中创建 Amazon Ion 表。有关信息，请参阅使用 CTAS 和 INSERT INTO 创建 Amazon Ion 表。

指定 STORED AS ION

以下示例 CREATE TABLE 语句在 LOCATION 子句之前使用 STORED AS ION 创建基于 Amazon Ion 格式的航班数据的表。LOCATION 子句指定了 Ion 格式的输入文件所在的存储桶或文件夹。扫描指定位置的所有文件。

```
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/'
```

指定 Amazon Ion 类路径

您可以不使用 STORED AS ION 语法，而是明确指定 ROW FORMAT SERDE、INPUTFORMAT 和 OUTPUTFORMAT 子句的 Ion 类路径值，如下所示。

参数	Ion 类路径
ROW FORMAT SERDE	'com.amazon.ionhiveserde.IonHiveSerDe'
STORED AS INPUTFORMAT	'com.amazon.ionhiveserde.formats.IonInputFormat'
OUTPUTFORMAT	'com.amazon.ionhiveserde.formats.IonOutputFormat'

以下的 DDL 查询使用此方法创建与前面示例中相同的外部表。

```
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/'
```

有关的 Athena 中 CREATE TABLE 语句的 SerDe 属性信息，请参阅 使用 Amazon Ion SerDe 属性。

使用 CTAS 和 INSERT INTO 创建 Amazon Ion 表

您可以使用 CREATE TABLE AS SELECT（CTAS）和 INSERT INTO 语句将数据从表复制或插入到 Athena 中 Amazon Ion 格式的新表中。

在 CTAS 查询中，在 format='ION' 子句中指定 WITH，如以下示例所示。

```
CREATE TABLE new_table
WITH (format='ION')
AS SELECT * from existing_table
```

默认情况下，Athena 以 Ion 二进制格式序列化 Amazon Ion 结果，但您也可以使用文本格式。若要使用文本格式，请在 CTAS WITH 子句中指定 ion_encoding = 'TEXT'，如以下示例所示。

```
CREATE TABLE new_table
WITH (format='ION', ion_encoding = 'TEXT')
AS SELECT * from existing_table
```

有关 CTAS WITH 子句中 Amazon Ion 特定属性的更多信息，请参阅以下部分。

CTAS WITH 子句 Amazon Ion 属性

在 CTAS 查询中，您可以使用 WITH 子句指定 Amazon Ion 格式，并可选择指定要使用的 Amazon Ion 编码和/或写入压缩算法。

format

您可以将 ION 关键字指定为 CTAS 查询的 WITH 子句中的格式选项。执行此操作时，您创建的表将使用您为 IonInputFormat 指定的读取格式，并以您为 IonOutputFormat 指定的格式序列化数据。

以下示例指定 CTAS 查询使用 Amazon Ion 格式。

```
WITH (format='ION')
```

ion_encoding

可选

默认：BINARY

值：BINARY、TEXT

指定以 Amazon Ion 二进制格式还是 Amazon Ion 文本格式序列化数据。以下示例指定 Amazon Ion 文本格式。

```
WITH (format='ION', ion_encoding='TEXT')
```

write_compression

可选

默认：GZIP

值：GZIP、ZSTD、BZIP2、SNAPPY、NONE

指定用于压缩输出文件的压缩算法。

以下示例指定 CTAS 查询使用 Zstandard 压缩算法以 Amazon Ion 格式写入其输出。

```
WITH (format='ION', write_compression = 'ZSTD')
```

有关在 Athena 中使用压缩的信息，请参阅 Athena 压缩支持。

有关 Athena 中其他 CTAS 属性的信息，请参阅 CTAS 表属性。

使用 Amazon Ion SerDe 属性

本主题包含有关 Athena 中 CREATE TABLE 语句的 SerDe 属性信息。有关 Amazon Ion SerDe 属性用法的更多信息和示例，请参阅 GitHub 上 Amazon Ion Hive SerDe 文档中的 SerDe 属性。

指定 Amazon Ion SerDe 属性

若要在 CREATE TABLE 语句中指定 Amazon Ion Hive SerDe 的属性，请使用 WITH SERDEPROPERTIES 子句。由于 WITH SERDEPROPERTIES 是 ROW FORMAT SERDE 子句的子字段，因此您必须首先指定 ROW FORMAT SERDE 和 Amazon Ion Hive SerDe 类路径，如以下语法所示。

```
...
ROW FORMAT SERDE
 'com.amazon.ionhiveserde.IonHiveSerDe'
WITH SERDEPROPERTIES (
 'property' = 'value',
 'property' = 'value',
...
)
```

请注意，如果要使用 ROW FORMAT SERDE，尽管需要 WITH SERDEPROPERTIES 子句，但您可以使用 STORED AS ION 或者更长的 INPUTFORMAT 和 OUTPUTFORMAT 语法来指定 Amazon Ion 格式。

Amazon Ion SerDe 属性

以下是可以在 Athena 中的 CREATE TABLE 语句中使用的 Amazon Ion SerDe 属性。

ion.encoding

可选

默认：BINARY

值：BINARY、TEXT

此属性声明添加的新值是否序列化为 Amazon Ion 二进制或者 Amazon Ion 文本格式。

以下 SerDe 属性示例指定 Amazon Ion 文本格式。

```
'ion.encoding' = 'TEXT'
```

ion.fail_on_overflow

可选

默认：`true`

值：`true`、`false`

Amazon Ion 允许任意大的数字类型，而 Hive 则不允许。默认情况下，如果 Amazon Ion 值不适合 Hive 列，SerDe 将失败，但是您可以使用 `fail_on_overflow` 配置选项让值溢出而不是失败。

可以在表或列级别设置此属性。若要在表级别指定属性，请按以下示例指定 `ion.fail_on_overflow`。这将为所有列设置默认行为。

```
'ion.fail_on_overflow' = 'true'
```

若要控制特定列，请在 `ion` 和 `fail_on_overflow` 之间指定列名称，以句点分隔，如以下示例所示。

```
'ion.<column>.fail_on_overflow' = 'false'
```

ion.path_extractor.case_sensitive

可选

默认：`false`

值：`true`、`false`

确定是否将 Amazon Ion 字段名称视为区分大小写。`false` 时，SerDe 忽略解析 Amazon Ion 字段名称的大小写。

例如，假设您有一个 Hive 表架构，其定义小写 `alias` 字段和包含 `alias` 字段和 `ALIAS` 字段的 Amazon Ion 文档，如以下示例所示。

```
-- Hive Table Schema
alias: STRING

-- Amazon Ion Document
{ 'ALIAS': 'value1'}
```

```
{ 'alias': 'value2'}
```

以下示例显示区分大小写设置为 `false` 时的 SerDe 属性和生成的提取表：

```
-- Serde properties
'ion.alias.path_extractor' = '(alias)'
'ion.path_extractor.case_sensitive' = 'false'

--Extracted Table
| alias    |
|----------|
| "value1" |
| "value2" |
```

以下示例显示区分大小写设置为 `true` 时的 SerDe 属性和生成的提取表：

```
-- Serde properties
'ion.alias.path_extractor' = '(alias)'
'ion.path_extractor.case_sensitive' = 'true'

--Extracted Table
| alias    |
|----------|
| "value2" |
```

在第二种情况下，区分大小写设置为 `true` 并且路径提取器指定为 `alias` 时，将忽略 `ALIAS` 字段的 `value1`。

ion.***<column>***.path_extractor

可选

默认：NA

值：带搜索路径的字符串

使用给定列的指定搜索路径创建路径提取器。路径提取器将 Amazon Ion 字段映射到 Hive 列。如果未指定路径提取器，则 Athena 在运行时根据列名动态创建路径提取器。

以下示例路径提取器将 `example_ion_field` 映射到 `example_hive_column`。

```
'ion.example_hive_column.path_extractor' = '(example_ion_field)'
```

有关路径提取器和搜索路径的更多信息，请参阅 使用路径提取器。

ion.timestamp.serialization_offset

可选

默认：'Z'

值：OFFSET，其中 OFFSET 表示为 *<signal>*hh:mm。示例值：01:00、+01:00、-09:30、Z（UTC，与 00:00 相同）

Apache Hive 时间戳没有内置时区，并且存储为 UNIX 纪元的偏移量，而 Amazon Ion 时间戳则有偏移量。序列化到 Amazon Ion 时，使用此属性指定偏移量。

以下示例添加一小时的偏移量。

```
'ion.timestamp.serialization_offset' = '+01:00'
```

ion.serialize_null

可选

默认：OMIT

值：OMIT、UNTYPED、TYPED

Amazon Ion SerDe 可以配置为序列化或省略具有空值的列。您可以选择写出强类型空值（TYPED）或无类型空值（UNTYPED）。基于默认的 Amazon Ion 到 Hive 类型映射确定强类型空值。

以下示例指定强类型空值。

```
'ion.serialize_null'='TYPED'
```

ion.ignore_malformed

可选

默认：false

值：true、false

`true` 时，如果 SerDe 无法读取格式错误的条目或整个文件，则忽略条目或整个文件。有关更多信息，请参阅 GibHub 上文档中的忽略格式错误。

ion.*<column>*.serialize_as

可选

默认：列的默认类型。

值：包含 Amazon Ion 类型的字符串

确定序列化值的 Amazon Ion 数据类型。由于 Amazon Ion 和 Hive 类型并不总是具有直接映射关系，因此一些 Hive 类型具有多个用于序列化的有效数据类型。若要将数据序列化为非默认数据类型，请使用此属性。有关类型映射的更多信息，请参阅 GitHub 上的 Amazon Ion 类型映射页面。

默认情况下，二进制 Hive 列序列化为 Amazon Ion Blob，但也可以序列化为 Amazon Ion Clob（字符大对象）。以下示例将列 `example_hive_binary_column` 序列化为 Clob。

```
'ion.example_hive_binary_column.serialize_as' = 'clob'
```

使用路径提取器

Amazon Ion 是一种文档样式的文件格式，但 Apache Hive 是一种平面列式格式。您可以使用名为 `path extractors` 的特殊 Amazon Ion SerDe 属性在两种格式之间进行映射。路径提取器将分层的 Amazon Ion 格式展平，将 Amazon Ion 值映射到 Hive 列，并可用于重命名字段。

Athena 可以为您生成提取器，但如有必要，您也可以定义自己的提取器。

生成的路径提取器

默认情况下，Athena 会搜索与 Hive 列名称匹配的顶级 Amazon Ion 值，并根据这些匹配值在运行时创建路径提取器。如果您的 Amazon Ion 数据格式与 Hive 表架构匹配，则 Athena 会动态为您生成提取器，而且您无需添加任何其他路径提取器。这些默认路径提取器不存储于表元数据中。

以下示例演示了 Athena 如何根据列名称生成提取器。

```
-- 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/'
```

以下示例提取器由 Athena 生成。第一个将 identification 字段提取到 identification 列，第二个将 alias 字段提取到 alias 列。

```
'ion.identification.path_extractor' = '(identification)'
'ion.alias.path_extractor' = '(alias)'
```

以下示例显示提取的表。

```
|                  identification                    |  alias   |
|----------------------------------------------------|----------|
|{["name", "driver_license"],["John Smith", "XXXX"]} | "Johnny" |
```

指定自己的路径提取器

如果您的 Amazon Ion 字段没有整齐地映射到 Hive 列，则您可以指定自己的路径提取器。在 CREATE TABLE 语句的 WITH SERDEPROPERTIES 子句中，请使用以下语法。

```
WITH SERDEPROPERTIES (
   "ion.path_extractor.case_sensitive" = "<Boolean>",
   "ion.<column_name>.path_extractor" = "<path_extractor_expression>"
)
```

> **Note**
>
> 默认情况下，路径提取器不区分大小写。若要覆盖此设置，请将 ion.path_extractor.case_sensitive SerDe 属性设置为 true。

在路径提取器中使用搜索路径

路径提取器的 SerDe 属性语法包含 *<path_extractor_expression>*：

```
"ion.<column_name>.path_extractor" = "<path_extractor_expression>"
```

您可以使用 *<path_extractor_expression>* 以指定解析 Amazon Ion 文档并查找匹配数据的搜索路径。搜索路径用括号括起来，可以包含以下一个或多个以空格分隔的组件。

- 通配符 – 匹配所有值。
- 索引 – 匹配指定数字索引处的值。指数从零开始。
- 文本 – 匹配字段名称与指定文本匹配的所有值。
- 注释 – 匹配由具有指定注释的包装路径组件指定的值。

以下示例显示了 Amazon Ion 文档和一些示例搜索路径。

```
-- 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"
```

提取器示例

拼合和重命名字段

以下示例显示了用于拼合和重命名字段的搜索路径。此示例使用搜索路径来执行以下操作：

- 将 `nickname` 列映射到 `alias` 字段
- 将 `name` 列映射到位于 `identification` 结构中的 `name` 子字段。

以下是 Amazon Ion 文档示例。

```
-- Example Amazon Ion Document
{
    identification: {
        name: "John Smith",
        driver_license: "XXXX"
    },

    alias: "Johnny"
}
```

以下是定义路径提取器的示例 CREATE TABLE 语句。

```
-- 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/'
```

以下示例显示提取的数据。

```
-- Extracted Table
| name         |   nickname   |
|--------------|--------------|
| "John Smith" |  "Johnny"    |
```

有关搜索路径和其他搜索路径示例的更多信息，请参阅 GitHub 上的 Ion Java 路径提取页面。

将航班数据提取为文本格式

以下示例 CREATE TABLE 查询使用 WITH SERDEPROPERTIES 添加路径提取器以提取航班数据，并将输出编码指定为 Amazon Ion 文本。此示例使用 STORED AS ION 语法。

```
CREATE EXTERNAL TABLE flights_ion (
    yr INT,
    quarter INT,
    month INT,
    dayofmonth INT,
    dayofweek INT,
    flightdate STRING,
    uniquecarrier STRING,
    airlineid INT,
)
ROW FORMAT SERDE
 'com.amazon.ionhiveserde.IonHiveSerDe'
WITH SERDEPROPERTIES (
 'ion.encoding' = 'TEXT',
 'ion.yr.path_extractor'='(year)',
 'ion.quarter.path_extractor'='(results quarter)',
 'ion.month.path_extractor'='(date month)')
STORED AS ION
LOCATION 's3://DOC-EXAMPLE-BUCKET/'
```

Avro SerDe

SerDe 名称

Avro SerDe

库名称

org.apache.hadoop.hive.serde2.avro.AvroSerDe

示例

出于安全原因，Athena 不支持使用 `avro.schema.url` 指定表架构。使用 `avro.schema.literal`。要从 Avro 格式的数据中提取架构，请将 Apache `avro-tools-<version>.jar` 与 `getschema` 参数配合使用。这会返回一个架构，您可以在 `WITH SERDEPROPERTIES` 语句中使用它。例如：

```
java -jar avro-tools-1.8.2.jar getschema my_data.avro
```

`avro-tools-<version>.jar` 文件位于您安装的 Avro 版本的 `java` 子目录中。要下载 Avro，请参阅 Apache Avro 版本。要直接下载 Apache Avro 工具，请参阅 Apache Avro 工具 Maven 存储库。

获取架构后，请使用 CREATE TABLE 语句根据 Amazon S3 中存储的底层 Avro 数据创建一个 Athena 表。要指定 Avro SerDe，请使用 ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'。如以下示例所示，除了为表指定列名和相应的数据类型之外，还必须使用 WITH SERDEPROPERTIES 子句指定架构。

> **Note**
>
> 在 s3://athena-examples-*myregion*/path/to/data/ 中，将 *myregion* 替换为您运行 Athena 所在的区域标识符，例如 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/';
```

在该表上运行 MSCK REPAIR TABLE 语句以刷新分区元数据。

```
MSCK REPAIR TABLE flights_avro_example;
```

按照出发总数查询前 10 个出发城市。

```
SELECT origin, count(*) AS total_departures
FROM flights_avro_example
WHERE year >= '2000'
GROUP BY origin
ORDER BY total_departures DESC
LIMIT 10;
```

> **Note**
>
> 飞行表数据来自由美国运输部交通统计局提供的航班。从原来的数据中进行稀释。

Grok SerDe

Logstash Grok SerDe 是一个库，它具有一组专门用于对非结构化文本数据（通常是日志）进行反序列化的模式。每个 Grok 模式都是一个命名的正则表达式。您可以根据需要识别并重新使用这些反序列化模式。这使得使用 Grok 比使用正则表达式更容易。Grok 提供了一组预定义模式。您也可以创建自定义模式。

要当在 Athena 中创建表时指定 Grok SerDe，请使用 `ROW FORMAT SERDE 'com.amazonaws.glue.serde.GrokSerDe'` 子句，后跟 `WITH SERDEPROPERTIES` 子句，用于指定要在您的数据中匹配的模式，其中：

- `input.format` 表达式定义要在数据中匹配的模式。版本是必需的。
- `input.grokCustomPatterns` 表达式定义了一个命名的自定义模式，您可以随后在 `input.format` 表达式中使用它。这是可选的。要将多个模式条目加入 `input.grokCustomPatterns` 表达式中，请使用换行转义字符 (\n) 分隔它们，如下所示：`'input.grokCustomPatterns'='INSIDE_QS ([^\"]*)\nINSIDE_BRACKETS ([^\]]*)')`。
- `STORED AS INPUTFORMAT` 和 `OUTPUTFORMAT` 子句是必需的。
- `LOCATION` 子句指定一个 Amazon S3 存储桶，它可以包含多个数据对象。将会对存储桶中的所有数据对象进行反序列化以创建表。

示例

这些示例依赖于预定义 Grok 模式的列表。请参阅预定义模式。

示例 1

此示例使用来自保存在 s3://DOC-EXAMPLE-BUCKET/groksample/ 中的 Postfix maillog 条目中的源数据。

```
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>
```

以下语句使用自定义模式和您指定的预定义模式，根据源数据文件在 Athena 中创建一个名为 mygroktable 的表：

```
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/';
```

从 %{NOTSPACE:column} 等简单的模式开始，以便先映射列，然后根据需要对列进行专业化处理。

示例 2

在以下示例中，您将为 Log4j 日志创建一个查询。这些示例日志具有采用以下格式的条目：

```
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
```

查询此日志数据：

- 将 Grok 模式添加到每个列的 input.format 中。例如，对于 timestamp，添加 %{TIMESTAMP_ISO8601:timestamp}。对于 loglevel，添加 %{LOGLEVEL:loglevel}。
- 通过映射短划线 (-) 和分隔日志格式中的条目的逗号，确保 input.format 中的模式与日志的格式完全匹配。

```
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/';
```

示例 3

查询 Amazon S3 日志的以下示例显示了 'input.grokCustomPatterns' 表达式，该表达式包含两个模式条目，由换行转义字符 (\n) 进行分隔，如示例查询的此代码段中所示：'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'
```

JSON SerDe 库

在 Athena 中，您可以使用 SerDe 库来将 JSON 数据反序列化。反序列化会转换 JSON 数据，以便它可以被序列化（写出）为不同的格式，如 Parquet 或 ORC。

- 本机 Hive JSON SerDe
- 这些区域有：OpenX JSON SerDe
- 这些区域有：Amazon Ion Hive SerDe

> **Note**
>
> Hive 和 OpenX 库期望 JSON 数据位于单行上（未格式化），用新的行字符分隔记录。Amazon Ion Hive SerDe 没有该要求，可以用作替代方案，因为 Ion 数据格式是 JSON 的超集。

库名称

使用以下值之一：

org.apache.hive.hcatalog.data.JsonSerDe

org.openx.data.jsonserde.JsonSerDe

com.amazon.ionhiveserde.IonHiveSerDe

Hive JSON SerDe

Hive JSON SerDe 常用于处理诸如事件之类的 JSON 数据。这些事件表示为用新行分隔的 JSON 编码文本的单行字符串。Hive JSON SerDe 不允许 `map` 或 `struct` 键名称中出现重复的键。

> **Note**
>
> SerDe 期望每个 JSON 文档都位于单行文本中，并且不使用行终止字符分隔记录中的字段。如果 JSON 文本采用美观的打印格式，当您在创建表后尝试对其进行查询时，可能会收到类似以下内容的错误消息：HIVE_CURSOR_ERROR: Row is not a valid JSON Object（HIVE_CURSOR_ERROR：行不是有效的 JSON 对象）或 HIVE_CURSOR_ERROR: JsonParseException: Unexpected end-of-input: expected close

marker for OBJECT（HIVE_CURSOR_ERROR：JsonParseException：意外的输入结束：对象的预期关闭标记）。有关更多信息，请参阅 GitHub 上 OpenX SerDe 文档中的 JSON 数据文件。

以下示例 DDL 语句使用 Hive JSON SerDe 基于示例在线广告数据创建表。在 LOCATION 子句中，将 s3://DOC-EXAMPLE-BUCKET.elasticmapreduce/samples/hive-ads/tables/impressions 中的 *myregion* 替换为您运行 Athena 所在的区域标识符（例如 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';
```

使用 Hive JSON SerDe 指定时间戳格式

要解析字符串中的时间戳值，可以将 WITH SERDEPROPERTIES 子字段添加到 ROW FORMAT SERDE 子句，然后用它来指定 timestamp.formats 参数。在此参数中，指定一个逗号分隔的时间戳模式列表，其中包含一个或多个时间戳模式，如以下示例所示：

```
...
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")
...
```

有关更多信息，请参阅 Apache Hive 文档中的 时间戳。

加载用于查询的表

创建表后，运行 MSCK REPAIR TABLE 以加载表并使其可从 Athena 进行查询：

```
MSCK REPAIR TABLE impressions
```

查询 CloudTrail 日志

要查询 CloudTrail 日志，您可以使用 Hive JSON SerDe。有关更多信息以及示例 `CREATE TABLE` 语句，请参阅 查询 AWS CloudTrail 日志。

OpenX JSON SerDe

与 Hive JSON SerDe 一样，您可以使用 OpenX JSON 来处理 JSON 数据。这些数据还表示为用新行分隔的 JSON 编码文本的单行字符串。与 Hive JSON SerDe 一样，OpenX JSON SerDe 不允许 `map` 或 `struct` 键名称中出现重复的键。

> **Note**
>
> SerDe 期望每个 JSON 文档都位于单行文本中，并且不使用行终止字符分隔记录中的字段。如果 JSON 文本采用美观的打印格式，当您在创建表后尝试对其进行查询时，可能会收到类似以下内容的错误消息：HIVE_CURSOR_ERROR: Row is not a valid JSON Object（HIVE_CURSOR_ERROR：行不是有效的 JSON 对象）或 HIVE_CURSOR_ERROR: JsonParseException: Unexpected end-of-input: expected close marker for OBJECT（HIVE_CURSOR_ERROR：JsonParseException：意外的输入结束：对象的预期关闭标记）。有关更多信息，请参阅 GitHub 上 OpenX SerDe 文档中的 JSON 数据文件。

可选属性

与 Hive JSON SerDe 不同，OpenX JSON SerDe 还具有以下可选的 SerDe 属性，非常适合用于解决数据中的不一致问题。

ignore.malformed.json

可选。设置为 TRUE 时，可让您跳过格式错误的 JSON 语法。默认为 FALSE。

dots.in.keys

可选。默认为 FALSE。设置为 TRUE 时，允许 SerDe 使用下划线替换键名称中的点。例如，如果 JSON 数据集包含名为 "a.b" 的键，您可以在 Athena 中使用此属性来定义列名 "a_b"。预设情况下（没有此 SerDe），Athena 不允许在列名中使用点。

case.insensitive

可选。默认为 TRUE。设置为 TRUE 时，SerDe 将所有大写列转换为小写。

要在数据中使用区分大小写的键名，请使用 WITH SERDEPROPERTIES ("case.insensitive"= FALSE;)。然后，对于每个非全部小写的键，请使用以下语法提供从列名到属性名的映射：

```
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES ("case.insensitive" = "FALSE", "mapping.userid" = "userId")
```

如果您有两个键（例如 URL 和 Url），并且二者在小写时是相同的，则可能会发生与以下内容类似的错误：

HIVE_CURSOR_ERROR: Row is not a valid JSON Object - JSONException: Duplicate key "url"（HIVE_CURSOR_ERROR：行不是有效的 JSON 对象 - JSONException：重复的键“url”）

要纠正此错误，请将 case.insensitive 属性设置为 FALSE，并将键映射到不同的名称，如以下示例所示：

```
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES ("case.insensitive" = "FALSE", "mapping.url1" = "URL",
 "mapping.url2" = "Url")
```

映射

可选。将列名映射到与列名不同的 JSON 键。当 JSON 数据包含作为关键字的键时，mapping 参数很有用。例如，如果您有名为 timestamp 的 JSON 键，请使用以下语法将该键映射到名为 ts 的列：

```
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES ("mapping.ts" = "timestamp")
```

将带有冒号的嵌套字段名称映射到与 Hive 兼容的名称

如果字段名称的 struct 中包含冒号，则可以使用 mapping 属性将该字段映射到与 Hive 兼容的名称。例如，假设您的列类型定义包含 my:struct:field:string，则可以通过在 WITH SERDEPROPERTIES 中加入以下条目来将定义映射到 my_struct_field:string：

```
("mapping.my_struct_field" = "my:struct:field")
```

以下示例显示了对应的 CREATE TABLE 语句。

```
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")
```

示例：广告数据

以下示例 DDL 语句使用 OpenX JSON SerDe 基于 Hive JSON SerDe 示例中使用的相同示例在线广告数据创建表。在 LOCATION 子句中，将 *myregion* 替换为您运行 Athena 的区域的标识符。

```
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';
```

示例：反序列化嵌套 JSON

您可以使用 JSON SerDes 来解析更复杂的 JSON 编码数据。这要求使用 CREATE TABLE 语句，而这些语句使用 struct 和 array 元素来表示嵌套结构。

以下示例根据具有嵌套结构的 JSON 数据创建 Athena 表。要在 Athena 中解析 JSON 编码的数据，请确保每个 JSON 文档都各占一行，并用换行符分隔。

此示例假定 JSON 编码的数据具有以下结构：

```
{
"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"
   }
  ]
```

```
 }
}
```

以下 CREATE TABLE 语句将 OpenX-JSONSerde 与 struct 和 array 集合数据类型结合使用来建立对象组。每个 JSON 文档都在其自己的行上列出，并用换行符分隔。为了避免错误，所查询的数据不会在 struct 或映射键名称中包含重复的键。

```
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/';
```

其他 资源

有关在 Athena 中使用 JSON 和嵌套 JSON 的更多信息，请参阅以下资源：

- 使用 JSONSerDe 在 Amazon Athena 中通过嵌套 JSON 和映射来创建表（AWS 大数据博客）
- 我在 Amazon Athena 中尝试读取 JSON 数据时收到错误（AWS 知识中心文章）
- hive-json-schema (GitHub) – 用 Java 编写的工具，可从示例 JSON 文档生成 CREATE TABLE 语句。生成的 CREATE TABLE 语句使用 OpenX JSON Serde。

用于 CSV、TSV 和自定义分隔文件的 LazySimpleSerDe

此 SerDe 的指定是可选的。这是 Athena 在预设情况下使用的 CSV、TSV 和自定义分隔格式数据的 SerDe。如果不指定任何 SerDe，并且只指定 `ROW FORMAT DELIMITED`，则会使用此 SerDe。如果您的数据没有用引号引起来的值，请使用此 SerDe。

有关 LazySimpleSerDe 的参考文档，请参阅《Apache Hive 开发人员指南》中的 Hive SerDe 部分。

库名称

LazySimpleSerDe 的类库名称为 `org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe`。有关 LazySimpleSerDe 类的信息，请参阅 GitHub.com 上的 LazySimpleSerDe.java。

忽略标题

要在您定义表时忽略标题，可以使用 `skip.header.line.count` 表属性，如下例所示。

```
TBLPROPERTIES ("skip.header.line.count"="1")
```

有关示例，请参阅 查询 Amazon VPC 流日志 和 查询 Amazon CloudFront 日志 中的 `CREATE TABLE` 语句。

CSV 示例

以下示例显示如何使用 `LazySimpleSerDe` 根据 CSV 数据在 Athena 中创建表。要使用此 SerDe 反序列化自定义分隔的文件，请遵循示例中的模式，但使用 `FIELDS TERMINATED BY` 子句指定单字符分隔符。LazySimpleSerDe 不支持多字符分隔符。

> **Note**
>
> 在 `s3://athena-examples-myregion/path/to/data/` 中，将 *myregion* 替换为您运行 Athena 所在的区域标识符，例如 `s3://athena-examples-us-west-1/path/to/data/`。

使用 `CREATE TABLE` 语句根据在 Amazon S3 中存储的底层 CSV 数据创建一个 Athena 表。

```
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/';
```

每次将新分区添加到此表时，请运行 MSCK REPAIR TABLE 以刷新分区元数据：

```
MSCK REPAIR TABLE flight_delays_csv;
```

查询前 10 个延迟超过 1 小时的路线：

```
SELECT origin, dest, count(*) as delays
FROM flight_delays_csv
WHERE depdelayminutes > 60
GROUP BY origin, dest
ORDER BY 3 DESC
LIMIT 10;
```

> **Note**
>
> 飞行表数据来自由美国运输部交通统计局提供的航班。从原来的数据中进行稀释。

TSV 示例

要根据存储在 Amazon S3 中的 TSV 数据创建 Athena 表，请使用 `ROW FORMAT DELIMITED` 并指定 `\t` 为制表符字段分隔符，指定 `\n` 为行分隔符，指定 `\` 为转义字符。以下示例显示该语法。`athena-examples` 地点没有示例 TSV 传输数据，但与 CSV 表一样，每次添加新分区时，您都需要运行 `MSCK REPAIR TABLE` 刷新分区元数据。

```
...
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
ESCAPED BY '\\'
LINES TERMINATED BY '\n'
...
```

用于处理 CSV 的 OpenCSVSerDe

当您为 CSV 数据创建 Athena 表时，请根据您的数据包含的值类型确定要使用的 SerDe：

- 如果数据包含使用双引号 (") 括起的值，则可以使用 OpenCSV SerDe 在 Athena 中将这些值反序列化。如果您的数据不包含使用双引号 (") 括起的值，则无需指定任何 SerDe。在此情况下，Athena 使用默认 `LazySimpleSerDe`。有关信息，请参阅用于 CSV、TSV 和自定义分隔文件的 LazySimpleSerDe。
- 如果您的数据具有 UNIX 数字 `TIMESTAMP` 值（例如，`1579059880000`），请使用 OpenCSVSerDe。如果您的数据使用 `java.sql.Timestamp` 格式，请使用 LazySimpleSerDe。

CSV SerDe (OpenCSVSerDe)

OpenCSV SerDe 具有以下字符串数据特性：

- 使用双引号 (") 作为默认引号字符，还允许您指定分隔符、引号和转义符，例如：

```
WITH SERDEPROPERTIES ("separatorChar" = ",", "quoteChar" = "`", "escapeChar" = "\\" )
```

- \t 或 \n 无法直接转义。要对它们进行转义，请使用 "escapeChar" = "\\"。请参阅本主题中的示例。
- 不支持 CSV 文件中的嵌入换行符。

对于不是 STRING 的数据类型，OpenCSVSerDe 的行为如下所示：

- 识别 BOOLEAN、BIGINT、INT 和 DOUBLE 数据类型。
- 在定义为数值数据类型的列中，无法识别空或空值，请将其保留为 string。一种解决方法是创建带有空值为 string 的列，然后使用 CAST 将查询中的字段转换为数字数据类型，为空值提供 0 的默认值。有关更多信息，请参阅 AWS 知识中心中的当我在 Athena 中查询 CSV 数据时，出现错误 HIVE_BAD_DATA：错误解析字段值。
- 对于使用 CREATE TABLE 语句中的 timestamp 数据类型指定的列，如果它使用的以毫秒为单位指定的 UNIX 数字格式，例如 1579059880000，则识别 TIMESTAMP 数据。
  - OpenCSVSerDe 不支持采用 JDBC 兼容 java.sql.Timestamp 格式的 TIMESTAMP，例如 "YYYY-MM-DD HH:MM:SS.fffffffff"（9 位小数精度）。
- 对于使用 CREATE TABLE 语句中的 DATE 数据类型指定的列，如果值表示自 1970 年 1 月 1 日以来已过去的天数，则会将值识别为日期。例如，列中带有 date 数据类型的值 18276 在查询时渲染为 2020-01-15。在这种 UNIX 格式下，每天都被认为有 86,400 秒。
  - OpenCSVSerDe 不直接支持任何其他格式的 DATE。要处理其他格式的时间戳数据，可以将列定义为 string，然后使用时间转换函数在 SELECT 查询中返回所需的值。有关更多信息，请参阅 AWS 知识中心中的文章：当在 Amazon Athena 中查询表时，时间戳结果为空。
- 要进一步将表中的列转换为所需的类型，您可以针对表创建视图，并使用 CAST 转换为所需的类型。

Example 示例：使用以 UNIX 数字格式指定的 TIMESTAMP 类型和 DATE 类型。

请考虑以下三列逗号分隔的数据。每列中的值都包含在双引号内。

```
"unixvalue creationdate 18276 creationdatetime 1579059880000","18276","1579059880000"
```

以下语句在 Athena 中根据指定的 Amazon S3 存储桶位置创建表。

```
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'
```

接下来运行以下查询：

```
SELECT * FROM testtimestamp1
```

查询返回以下结果，同时显示日期和时间数据：

```
profile_id                                                          creationdate
 creationdatetime
unixvalue creationdate 18276 creationdatetime 1579146280000        2020-01-15
 2020-01-15 03:44:40.000
```

Example 示例：针对 **\t** 或 **\n** 进行转义

请考虑使用以下测试数据：

```
" \\t\\t\\n 123 \\t\\t\\n ",abc
" 456 ",xyz
```

以下语句在 Athena 中创建一个表，指定 "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/'
```

接下来运行以下查询：

```
SELECT * FROM test1;
```

它会返回此结果，针对 \t 或 \n 正确进行转义：

```
f1            s2
\t\t\n 123 \t\t\n            abc
456                          xyz
```

SerDe 名称

CSV SerDe

库名称

要使用此 SerDe，请在 ROW FORMAT SERDE 后指定其完全限定类名。还需指定在 SERDEPROPERTIES 中指定分隔符，如下所示：

```
...
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
  "separatorChar" = ",",
  "quoteChar"     = "`",
  "escapeChar"    = "\\"
)
```

忽略标题

要在您定义表时忽略标题，可以使用 skip.header.line.count 表属性，如下例所示。

```
TBLPROPERTIES ("skip.header.line.count"="1")
```

有关示例，请参阅 查询 Amazon VPC 流日志 和 查询 Amazon CloudFront 日志 中的 CREATE TABLE 语句。

示例

此示例假定 CSV 中的数据保存在 s3://DOC-EXAMPLE-BUCKET/mycsv/ 中且具有以下内容：

```
"a1","a2","a3","a4"
"1","2","abc","def"
"a","a1","abc3","ab4"
```

使用 CREATE TABLE 语句根据数据创建 Athena 表。ROW FORMAT SERDE 之后应用 OpenCSVSerDe 类并指定 WITH SERDEPROPERTIES 中的字符分隔符、引号字符和转义字符，如以下示例所示。

```
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/';
```

查询表中的所有值：

```
SELECT * FROM myopencsvtable;
```

查询将返回以下值：

```
col1     col2    col3    col4
------------------------------
a1       a2      a3      a4
1        2       abc     def
a        a1      abc3    ab4
```

ORC SerDe

SerDe 名称

OrcSerDe

库名称

该库为 ORC 格式的数据使用 Apache Hive OrcSerde.java 类。它将对象从 ORC 传递到读取器，从 ORC 传递到写入器。

示例

> **Note**
>
> 在 `s3://athena-examples-myregion/path/to/data/` 中，将 *myregion* 替换为您运行 Athena 所在的区域标识符，例如 `s3://athena-examples-us-west-1/path/to/data/`。

以下示例为 ORC 中的航班延迟数据创建一个表。该表包括分区：

```
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");
```

在该表上运行 MSCK REPAIR TABLE 语句以刷新分区元数据：

```
MSCK REPAIR TABLE flight_delays_orc;
```

使用此查询以获取前 10 个延迟超过 1 小时的路线：

```
SELECT origin, dest, count(*) as delays
FROM flight_delays_orc
WHERE depdelayminutes > 60
GROUP BY origin, dest
ORDER BY 3 DESC
```

```
LIMIT 10;
```

Parquet SerDe

SerDe 名称

ParquetHiveSerDe 用于采用 Parquet 格式存储的数据。

> **Note**
>
> 要将数据转换为 Parquet 格式，您可以使用 CREATE TABLE AS SELECT (CTAS) 查询。有关更多信息，请参阅从查询结果创建表（CTAS）、CTAS 查询的示例和将 CTAS 和 INSERT INTO 用于 ETL 和数据分析。

库名称

Athena 在需要将存储在 Parquet 中的数据反序列化时，会使用以下类：`org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe`

示例：查询在 Parquet 中存储的文件

> **Note**
>
> 在 `s3://athena-examples-myregion/path/to/data/` 中，将 *myregion* 替换为您运行 Athena 所在的区域标识符，例如 `s3://athena-examples-us-west-1/path/to/data/`。

使用以下 `CREATE TABLE` 语句，根据 Amazon S3 中以 Parquet 格式存储的底层数据创建一个 Athena 表：

```
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");
```

在该表上运行 MSCK REPAIR TABLE 语句以刷新分区元数据：

```
MSCK REPAIR TABLE flight_delays_pq;
```

查询前 10 个延迟超过 1 小时的路线：

```
SELECT origin, dest, count(*) as delays
FROM flight_delays_pq
WHERE depdelayminutes > 60
GROUP BY origin, dest
ORDER BY 3 DESC
LIMIT 10;
```

> **Note**
>
> 飞行表数据来自由美国运输部交通统计局提供的航班。从原来的数据中进行稀释。

忽略 Parque 统计数据

当读取 Parquet 数据时，您可能会收到如下错误消息：

```
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.
```

要解决此问题，请使用 CREATE TABLE 或 ALTER TABLE SET TBLPROPERTIES 语句将 Parquet Serde `parquet.ignore.statistics` 属性设置为 `true`，如以下示例所示。

CREATE TABLE 示例

```
...
ROW FORMAT SERDE
'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
WITH SERDEPROPERTIES (
'parquet.ignore.statistics'='true')
STORED AS PARQUET
...
```

ALTER TABLE 示例

```
ALTER TABLE ... SET TBLPROPERTIES ('parquet.ignore.statistics'='true')
```

Regex SerDe

Regex SerDe 使用正则表达式通过将正则表达式组提取到表列中来反序列化数据。

如果数据中的某一行与正则表达式不匹配，则该行中的所有列都作为 NULL 返回。如果某行与正则表达式匹配，但其组少于预期，则缺少的组为 NULL。如果数据中的一行与正则表达式匹配，但其列多于正则表达式中的组，则会忽略其他列。

有关更多信息，请参阅 Apache Hive 文档中的 类 RegexSerDe。

SerDe 名称

RegexSerDe

库名称

RegexSerDe

示例

以下示例使用 RegExSerDe 从 CloudFront 日志创建一个表。在 s3://athena-examples-*myregion*/cloudfront/plaintext/ 中，将 *myregion* 替换为您运行 Athena 所在的区域标识符（例如 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/';
```

使用 Amazon Athena 运行 SQL 查询

您可以使用 Amazon Athena 对在 AWS Glue Data Catalog 中注册的以及诸如 Hive 元数据仓等数据源及您使用 Athena 联合查询功能连接的 Amazon DocumentDB 实例运行 SQL 查询。有关使用数据源的更多信息，请参阅连接到数据源。运行修改架构的数据定义语言(DDL)查询时，Athena 将元数据写入与数据源关联的元数据仓。此外，一些查询(如 CREATE TABLE AS 和 INSERT INTO)可以将记录写入数据集，例如，将 CSV 记录添加到 Amazon S3 位置。运行查询时，Athena 将查询的结果保存在您指定的查询结果位置。这使您能够查看查询历史记录以及下载和查看查询结果集。

本节提供了使用各种 SQL 语句对常见数据源和数据类型运行 Athena 查询的指导。为使用常见结构和运算符提供了一般指导，例如，如何使用数组、连接、过滤、拼合和排序。其他示例包括查询具有嵌套结构和映射的表、基于 JSON 编码数据集的表以及与 AWS CloudTrail 日志和 Amazon EMR 日志等

AWS 服务 关联的数据集中的数据。本文档讨论不包含标准 SQL 使用步骤的说明。有关 SQL 的更多信息，请参阅 Trino 和 Presto 语言参考。

主题

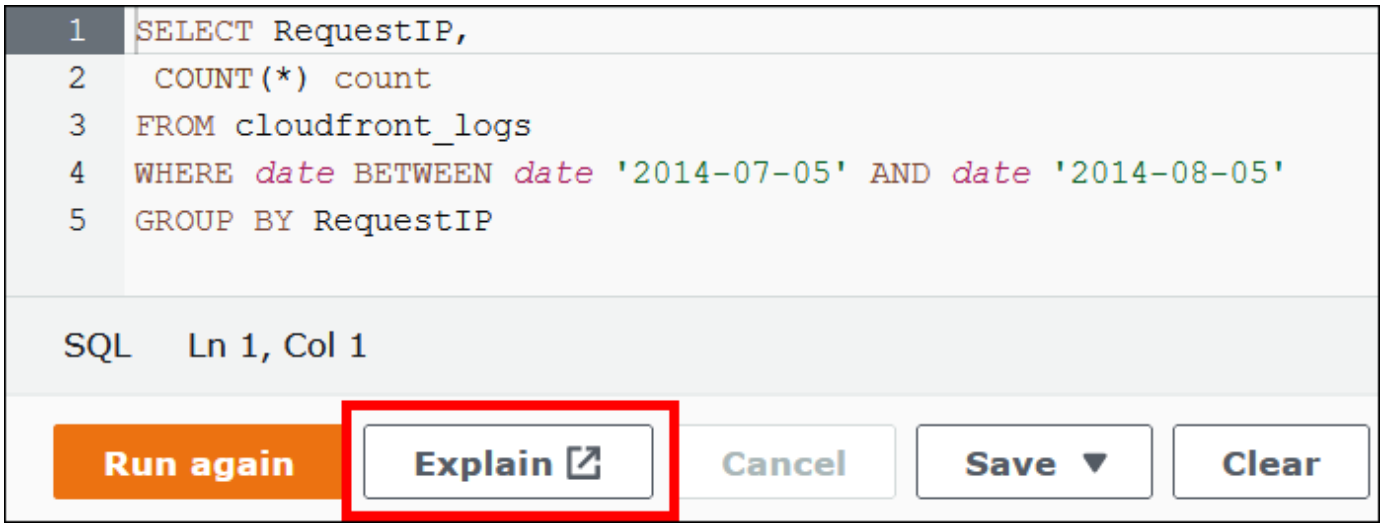
有关注意事项和限制，请参阅Amazon Athena 中 SQL 查询的注意事项和限制。

查看 SQL 查询的执行计划

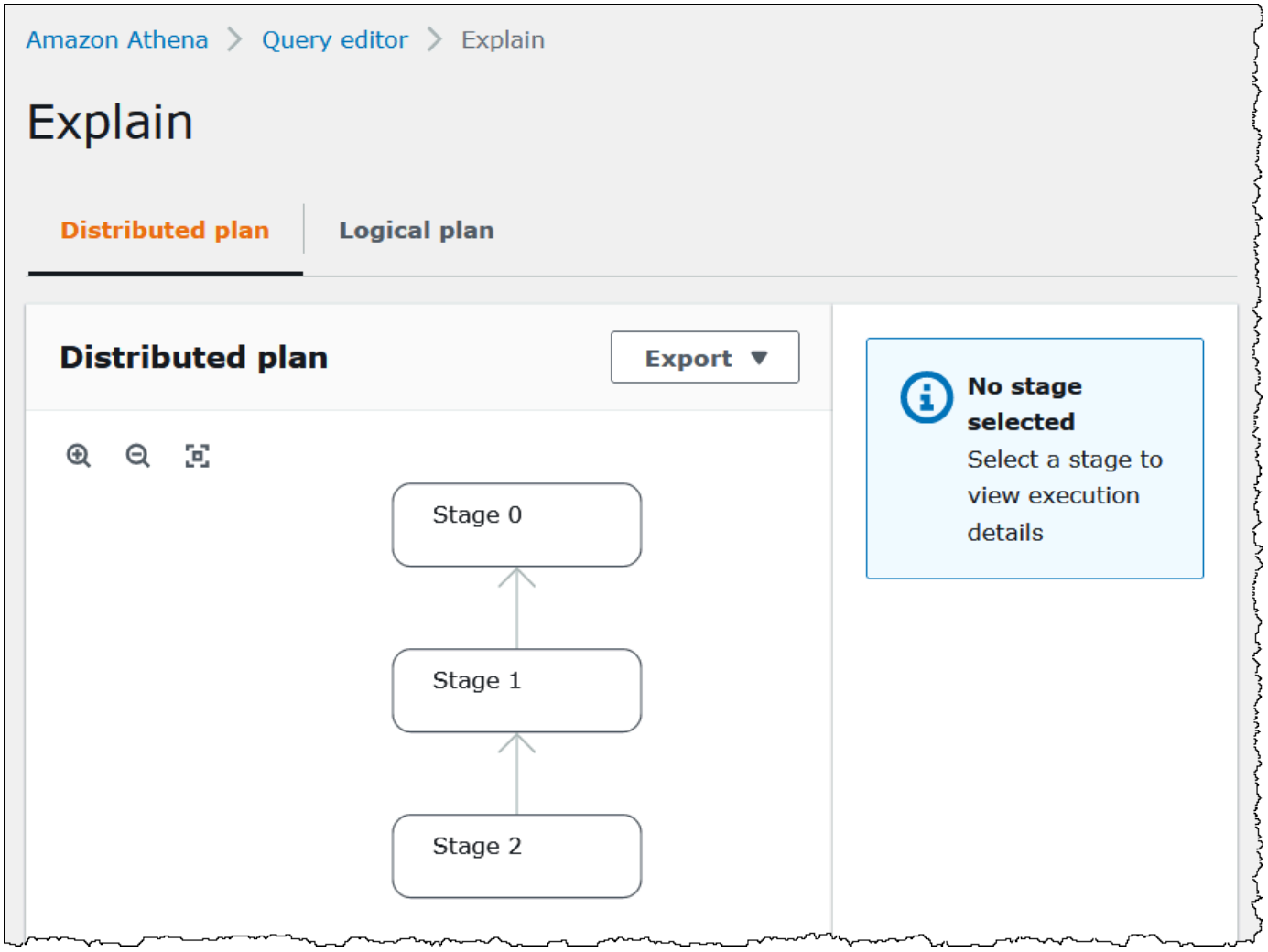
您可以使用 Athena 查询编辑器查看查询如何运行的图形展示。当您在编辑器中输入查询并选择 Explain（说明）选项时，Athena 使用查询上的 EXPLAIN SQL 语句创建两个相应图表：分布式执行计划和逻辑执行计划。您可以使用这些图表对查询进行分析、排除故障并提高查询的效率。

要查看查询的执行计划

1. 在查询编辑器中输入您的查询，然后选择 Explain（说明）。



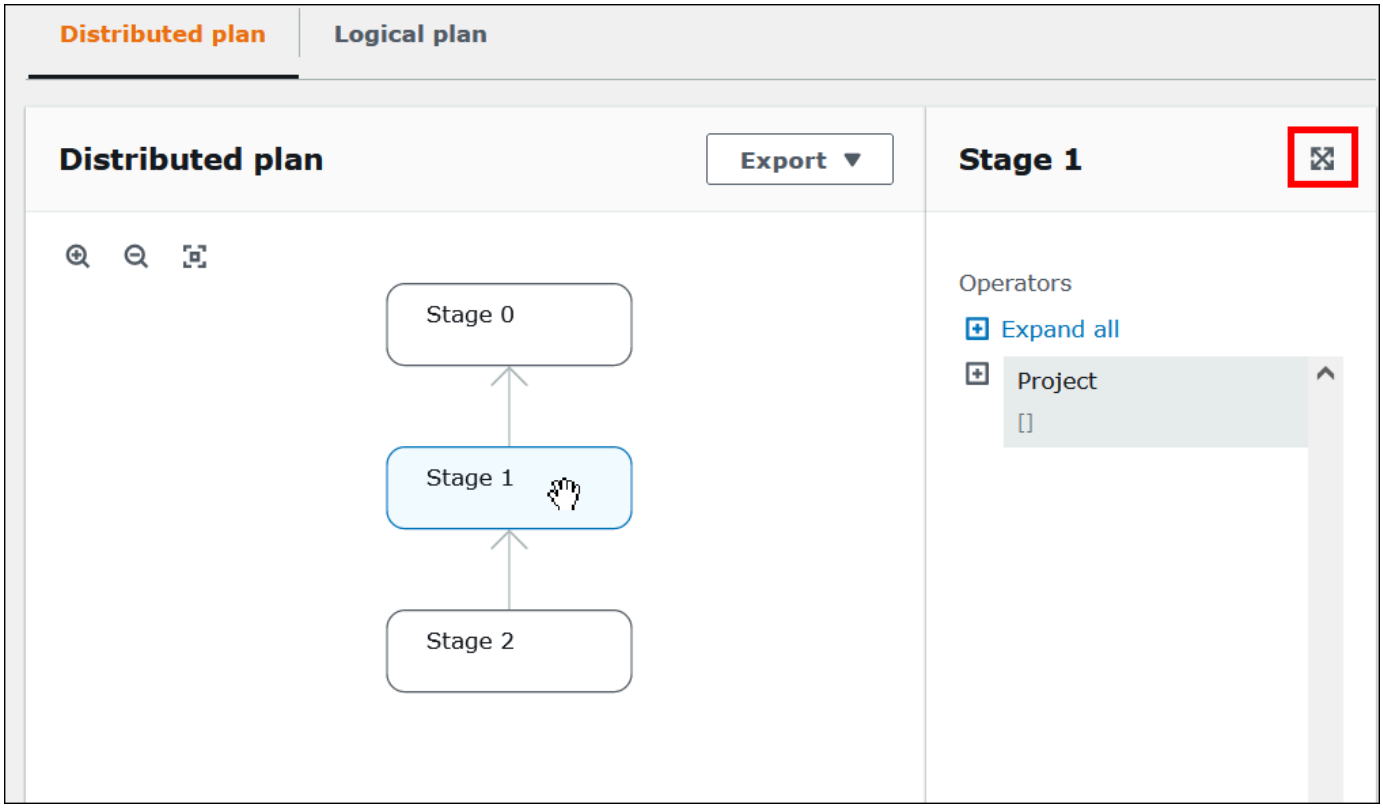
Distributed plan（分布式计划）选项卡将显示分布式环境中查询的执行计划。分布式计划具有处理片段或阶段。每个阶段都有从零开始的索引编号，并由一个或多个节点进行处理。数据可以在节点之间进行交换。



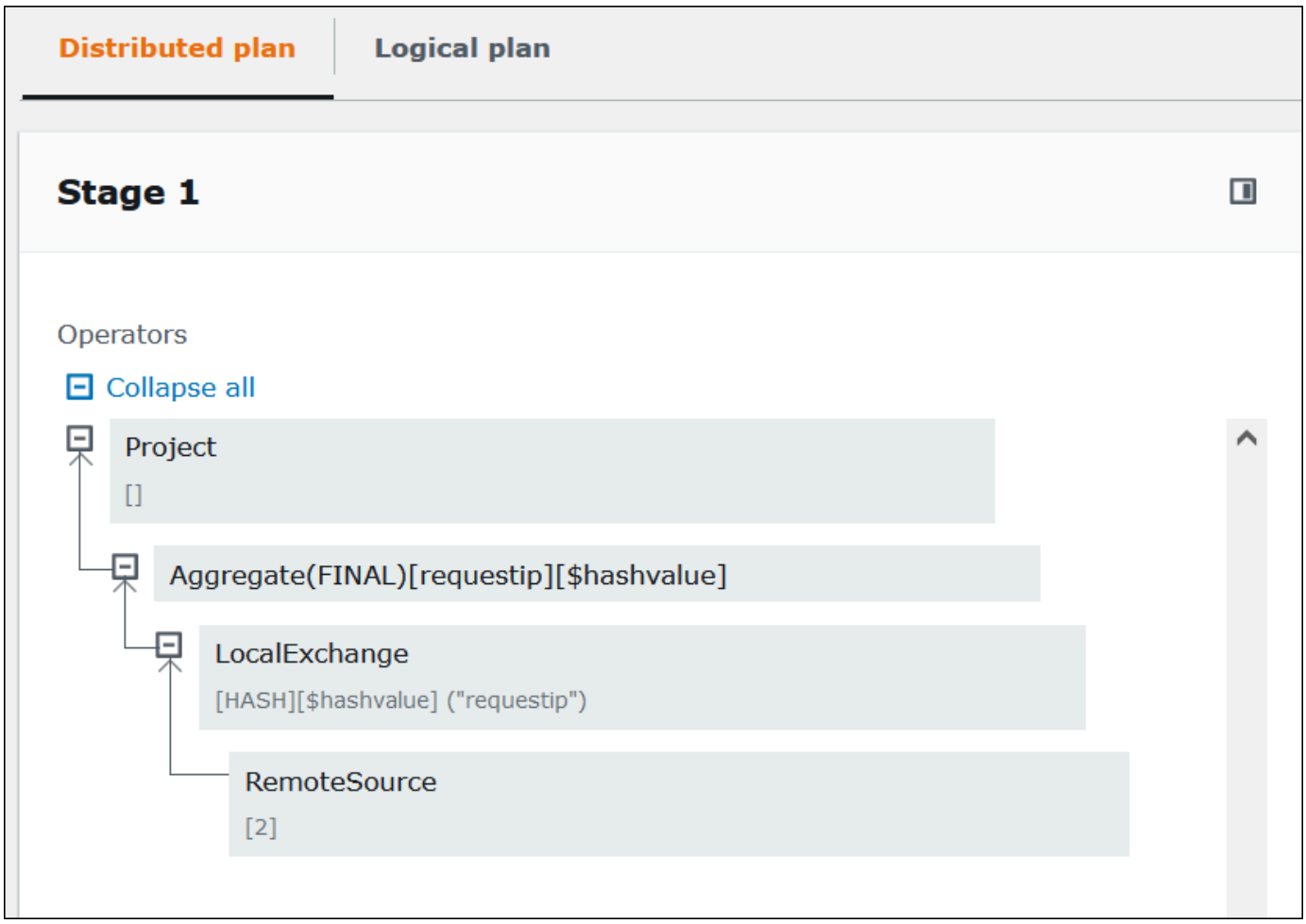
2. 要导航图表，请使用以下选项：

   - 要放大或缩小，请滚动鼠标或使用放大图标。
   - 要调整图表以适合屏幕，请选择缩放至适合图标。
   - 要移动图表，请拖动鼠标指针。

3. 要查看阶段的详细信息，请选择阶段。

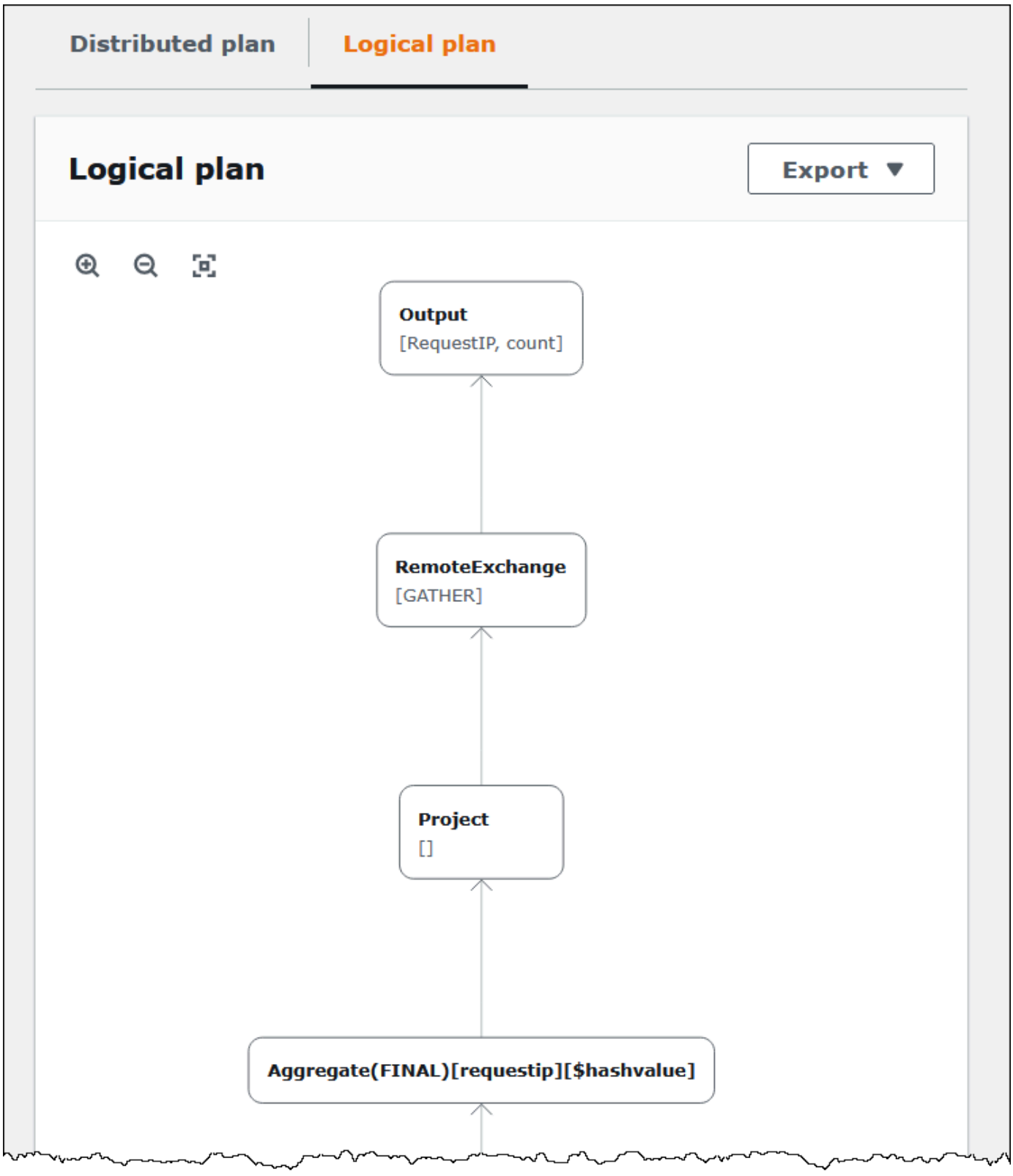


4. 要查看阶段的整体详细信息，请选择详细信息窗格右上角的展开图标。
5. 要查看更多详细信息，请展开运算符树中的一个或多个项目。有关分布式计划分段的信息，请参见 EXPLAIN 语句输出类型。



> **Important**
>
> 目前，某些分区筛选器可能在嵌套运算符树图表中不可见，即使 Athena 确实将其应用于您的查询。要验证此类筛选的效果，请在您的查询中运行 EXPLAIN 或 EXPLAIN ANALYZE 并查看结果。

6. 选择 Logical plan（逻辑计划）选项卡。该图表显示了运行查询的逻辑计划。有关操作术语的更多信息，请参阅 了解 Athena EXPLAIN 语句结果。



7. 要将计划导出为 SVG 或 PNG 图像或 JSON 文本，请选择 Export（导出）。

其他 资源

有关详细信息，请参阅以下资源：

在 Athena 中使用 EXPLAIN 和 EXPLAIN ANALYZE

了解 Athena EXPLAIN 语句结果

查看已完成查询的统计数据和执行详细信息

使用查询结果、最近查询和输出文件

Amazon Athena 会将运行的每个查询的查询结果和元数据信息自动存储在查询结果位置（可以在 Amazon S3 中指定）中。如有必要，您可以在此位置访问这些文件以对其进行处理。您还可以直接从 Athena 控制台下载查询结果文件。

要首次设置 Amazon S3 查询结果位置，请参阅 使用 Athena 控制台指定查询结果位置。

对于运行的每个查询，将自动保存输出文件。要使用 Athena 控制台访问和查看查询输出文件，IAM 主体（用户和角色）需要 Amazon S3 GetObject 操作的权限来获得查询结果位置，并需要 Athena GetQueryResults 操作的权限。可对查询结果位置进行加密。如果该位置已加密，用户必须具有加密和解密查询结果位置的相应密钥权限。

> **Important**
>
> 有权限对查询结果位置执行 Amazon S3 `GetObject` 操作的 IAM 委托人可以从 Amazon S3 中检索查询结果，即使 Athena `GetQueryResults` 操作的权限被拒绝。

指定查询结果位置

Athena 使用的查询结果位置由工作组设置和客户端设置共同决定。客户端设置取决于您运行查询的方式。

- 如果使用 Athena 控制台运行查询，在导航栏中 Settings（设置）项下输入的 Query result location（查询结果位置）将决定客户端设置。
- 如果您使用 Athena API 运行查询，则 StartQueryExecution 操作的 `OutputLocation` 参数将决定客户端设置。
- 如果您使用 ODBC 或 JDBC 驱动程序来运行查询，则连接 URL 中指定的 `S3OutputLocation` 属性决定客户端设置。

> **Important**
>
> 当您使用 API 或使用 ODBC 或 JDBC 驱动程序运行查询时，控制台设置不适用。

每个工作组配置都有一个可启用的Override client-side settings (覆盖客户端设置) 选项。如果启用此选项，当与工作组关联的 IAM 委托人运行该查询时，工作组设置优先于适用的客户端设置。

使用 Athena 控制台指定查询结果位置

在运行查询之前，必须指定 Amazon S3 中的查询结果存储桶位置，或者您必须使用已指定存储桶且其配置覆盖客户端设置的工作组。

使用 Athena 控制台指定客户端设置查询结果位置

1. 切换到要为其指定查询结果位置的工作组。工作组的默认名称为 primary。
2. 从导航栏中选择 Settings（设置）。
3. 从导航栏中选择 Manage（管理）。
4. 对于 Manage settings（管理设置），执行以下操作之一：
   - 在 Location of query result（查询结果位置）文本框中，输入您在 Amazon S3 中为查询结果创建的存储桶路径。在路径前添加前缀 `s3://`。
   - 选择 Browse S3（浏览 S3），选择您为当前区域创建的 Amazon S3 存储桶，然后选择 Choose（选择）。

   > **Note**
   >
   > 如果使用的工作组为工作组的所有用户指定查询结果位置，则更改查询结果位置的选项不可用。

5. （可选）选择 View lifecycle configuration（查看生命周期配置），以查看和配置查询结果存储桶上的 Amazon S3 生命周期规则。您创建的 Amazon S3 生命周期规则可以是到期规则或转移规则。到期规则将在经过一定的时间后自动删除查询结果。转换规则会将查询结果转移到其他 Amazon S3 存储层。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的在存储桶上设置生命周期配置。
6. （可选）在 Expected bucket owner（预期存储桶拥有者）中，输入您希望成为输出位置存储桶的拥有者 AWS 账户 的 ID。这是附加安全措施。如果存储桶拥有者的账户 ID 与您在此处指定的 ID

不匹配，则输出到存储桶的尝试将失败。有关更多信息，请参阅《Amazon S3 用户指南》中的使用存储桶拥有者条件验证存储桶所有权

> **Note**
>
> 预期存储桶拥有者设置仅适用于您为 Athena 查询结果指定的 Amazon S3 输出位置。其不适用于其他 Amazon S3 位置，例如外部 Amazon S3 存储桶中的数据源位置、CTAS 和 INSERT INTO 目标表位置、UNLOAD 语句输出位置、为联合查询溢出存储桶的操作，或针对另一个账户中的表运行的 SELECT 查询。

7. （可选）如果要加密存储在 Amazon S3 中的查询结果，则选择 Encrypt query results（加密查询结果）。要详细了解 Athena 的加密，请参阅静态加密。
8. （可选）如果查询结果存储桶启用了 ACL，则选择 Assign bucket owner full control over query results（为存储桶拥有者分配对查询结果的完全控制权），以向存储桶拥有者授予对查询结果的完全控制权。例如，假设您的查询结果位置属于其他账户所有，则可以将所有权以及对查询结果的完全控制权授予该其他账户。有关更多信息，请参阅《Amazon S3 用户指南》中的控制存储桶的对象所有权和禁用 ACL。
9. 选择 Save（保存）。

以前创建的默认位置

以前，在 Athena 中,如果您在运行查询时未为 Query result location（查询结果位置）指定值，并且查询结果位置设置未被工作组覆盖，则 Athena 会为您创建默认位置。默认位置为 `aws-athena-query-results-MyAcctID-MyRegion`，其中 *MyAcctID* 是运行查询的 IAM 委托人的 Amazon Web Services 账户 ID，*MyRegion* 是运行查询的区域（例如 `us-west-1`）。

现在，您必须指定查询结果位置或使用覆盖查询结果位置设置的工作组，然后才能在您的账户以前未使用 Athena 的区域中运行 Athena 查询。虽然 Athena 不再为您创建默认查询结果位置，但之前创建的默认 `aws-athena-query-results-MyAcctID-MyRegion` 位置仍然有效，您可以继续使用它们。

使用工作组指定查询结果位置

您使用 AWS Management Console、AWS CLI 或 Athena API 在工作组配置中指定查询结果位置。

使用 AWS CLI 时，请在运行 `aws athena create-work-group` 或者 `aws athena update-work-group` 命令时使用 `--configuration` 选项的 `OutputLocation` 参数指定查询结果位置。

使用 Athena 控制台指定工作组的查询结果位置

1. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。

2. 在导航窗格中，选择 Workgroups（工作组）。
3. 在工作组列表中，选择要编辑的工作组的链接。
4. 选择编辑。
5. 对于 Query result location and encryption（查询结果位置和加密），请执行以下操作之一：
   - 在 Location of query result（查询结果位置）框中，输入 Amazon S3 中用于存储查询结果的存储桶的路径。在路径前添加前缀 `s3://`。
   - 选择 Browse S3（浏览 S3），选择为您要使用的当前区域创建的 Amazon S3 存储桶，然后选择 Choose（选择）。
6. （可选）在 Expected bucket owner（预期存储桶拥有者）中，输入您希望成为输出位置存储桶的拥有者 AWS 账户 的 ID。这是附加安全措施。如果存储桶拥有者的账户 ID 与您在此处指定的 ID 不匹配，则输出到存储桶的尝试将失败。有关更多信息，请参阅《Amazon S3 用户指南》中的使用存储桶拥有者条件验证存储桶所有权

> **Note**
>
> 预期存储桶拥有者设置仅适用于您为 Athena 查询结果指定的 Amazon S3 输出位置。其不适用于其他 Amazon S3 位置，例如外部 Amazon S3 存储桶中的数据源位置、CTAS 和 `INSERT INTO` 目标表位置、`UNLOAD` 语句输出位置、为联合查询溢出存储桶的操作，或针对另一个账户中的表运行的 `SELECT` 查询。

7. （可选）如果要加密存储在 Amazon S3 中的查询结果，则选择 Encrypt query results（加密查询结果）。要详细了解 Athena 的加密，请参阅静态加密。
8. （可选）如果查询结果存储桶启用了 ACL，则选择 Assign bucket owner full control over query results（为存储桶拥有者分配对查询结果的完全控制权），以向存储桶拥有者授予对查询结果的完全控制权。例如，假设您的查询结果位置属于其他账户所有，则可以将所有权以及对查询结果的完全控制权授予该其他账户。

   如果存储桶的 S3 Object Ownership（S3 对象所有权）设置为 Bucket owner preferred（存储桶拥有者优先），则存储桶拥有者还拥有从此工作组写入的所有查询结果对象。例如，假设某个外部账户的工作组启用了此选项并将其查询结果位置设置为您账户的 Amazon S3 存储桶，并且该存储桶的 S3 Object Ownership（S3 对象所有权）设置为 Bucket owner preferred（存储桶拥有者优先），则您将拥有并完全控制该外部工作组的查询结果的访问权限。

   如果查询结果存储桶的 S3 Object Ownership（S3 对象所有权）设置为 Bucket owner enforced（存储桶拥有者强制），则选择此选项将不具有效力。有关更多信息，请参阅《Amazon S3 用户指南》中的控制存储桶的对象所有权和禁用 ACL。
9. 如果您希望工作组的所有用户使用您指定的查询结果位置，请向下滚动到 Settings（设置）部分，然后选择 Override client-side settings（覆盖客户端侧设置）。
10. 选择 Save changes（保存更改）。

使用 Athena 控制台下载查询结果文件

运行查询后，您可以立即从查询窗格中下载查询结果 CSV 文件。您还可以从 Recent queries（最近的查询）选项卡下载最近查询的查询结果。

> **Note**
>
> Athena 查询结果文件为数据文件，其中包含了可以被单个用户配置的信息。有些数据在用来读取和分析该数据的程序中有可能会将部分数据解释为命令（CSV 注入）。因此，将查询结果 CSV 数据导入到某个电子表格程序时，该程序可能警告您注意安全风险。为了确保系统安全，您应始终选择禁用下载查询结果中的链接或宏。

运行查询并下载查询结果

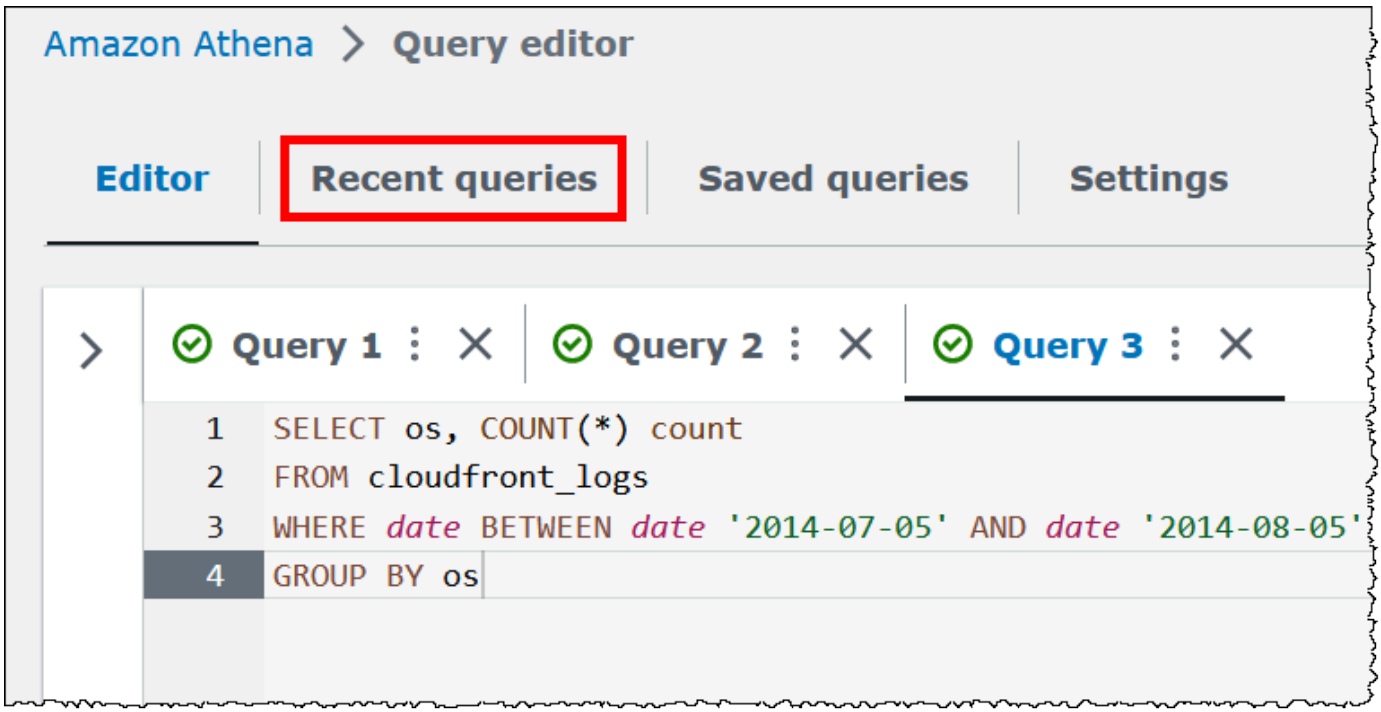
1. 在查询编辑器中输入您的查询，然后选择 Run（运行）。

   查询完成运行后，Results (结果) 窗格将显示查询结果。

2. 要下载查询结果的 CSV 文件，请选择查询结果窗格上方的 Download results（下载结果）。根据您的浏览器和浏览器配置，您可能需要确认下载。

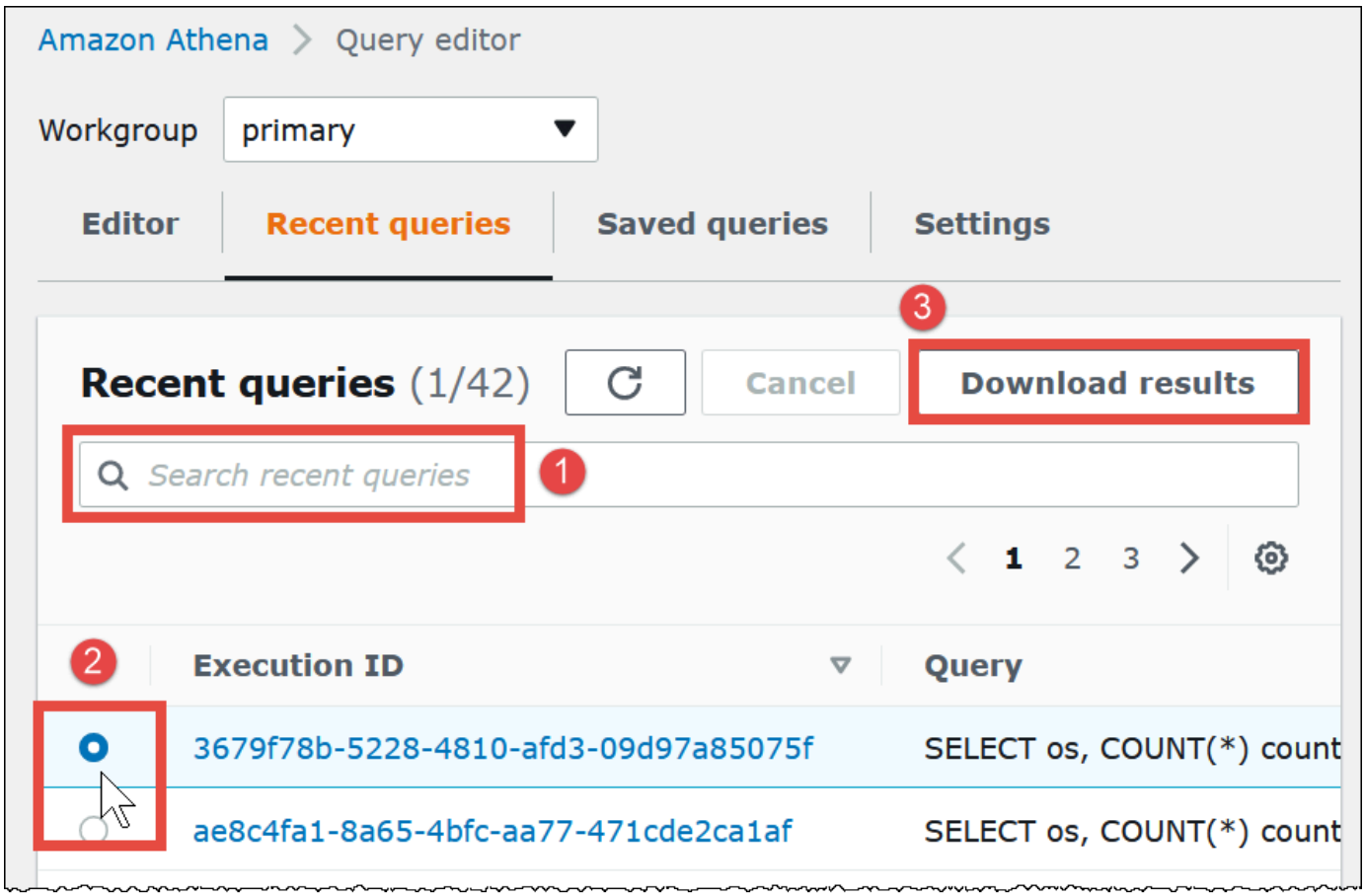
下载早期查询的查询结果文件

1. 选择 Recent queries（最近的查询）。



2. 使用搜索框查找查询，选择查询，然后选择 Download results（下载结果）。

> **Note**
>
> 您无法使用 Download results（下载结果）来选项检索已手动删除的查询结果，也无法检索已被 Amazon S3 生命周期规则 删除或转移至其他位置的查询结果。

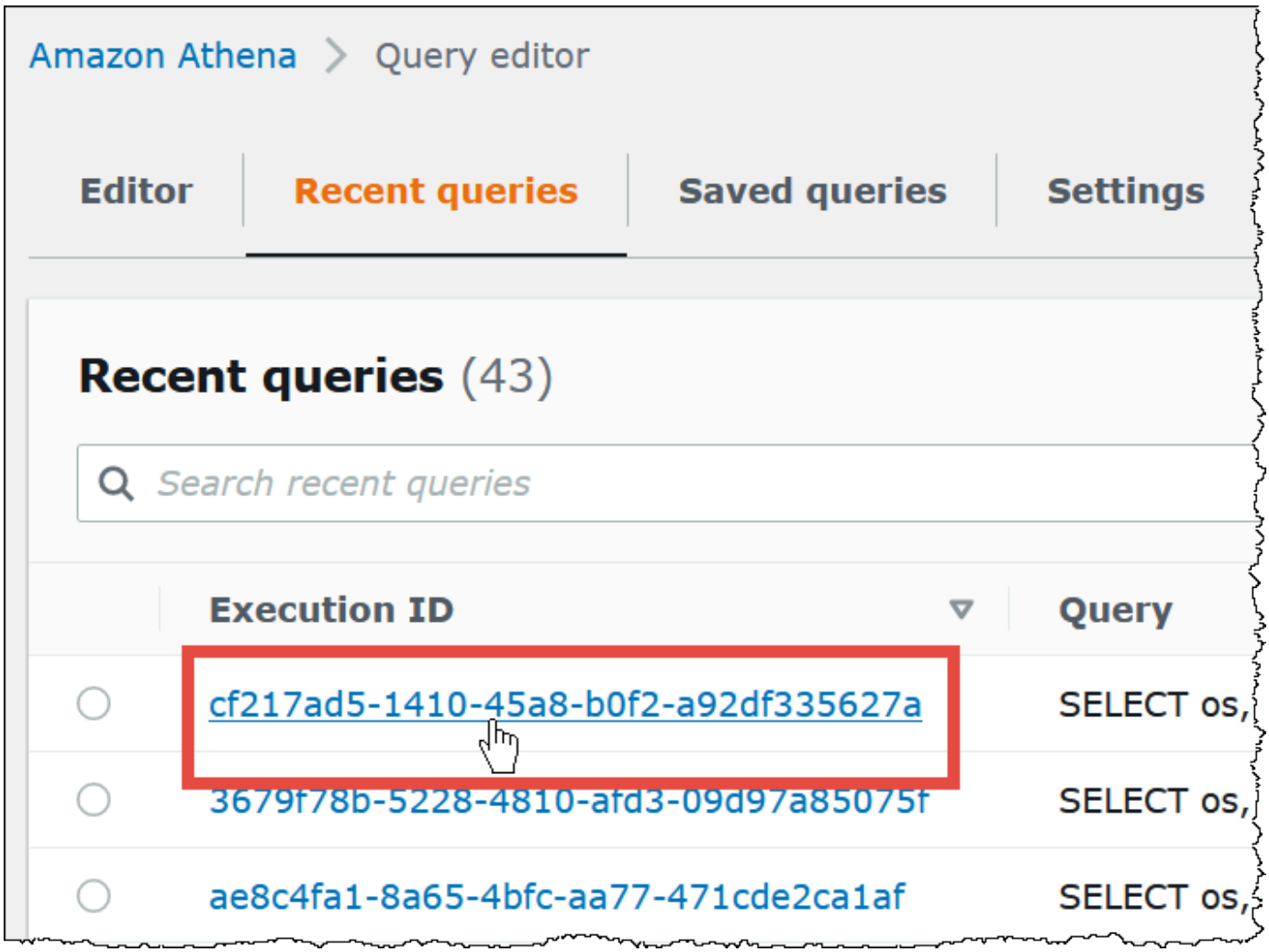


查看最近的查询

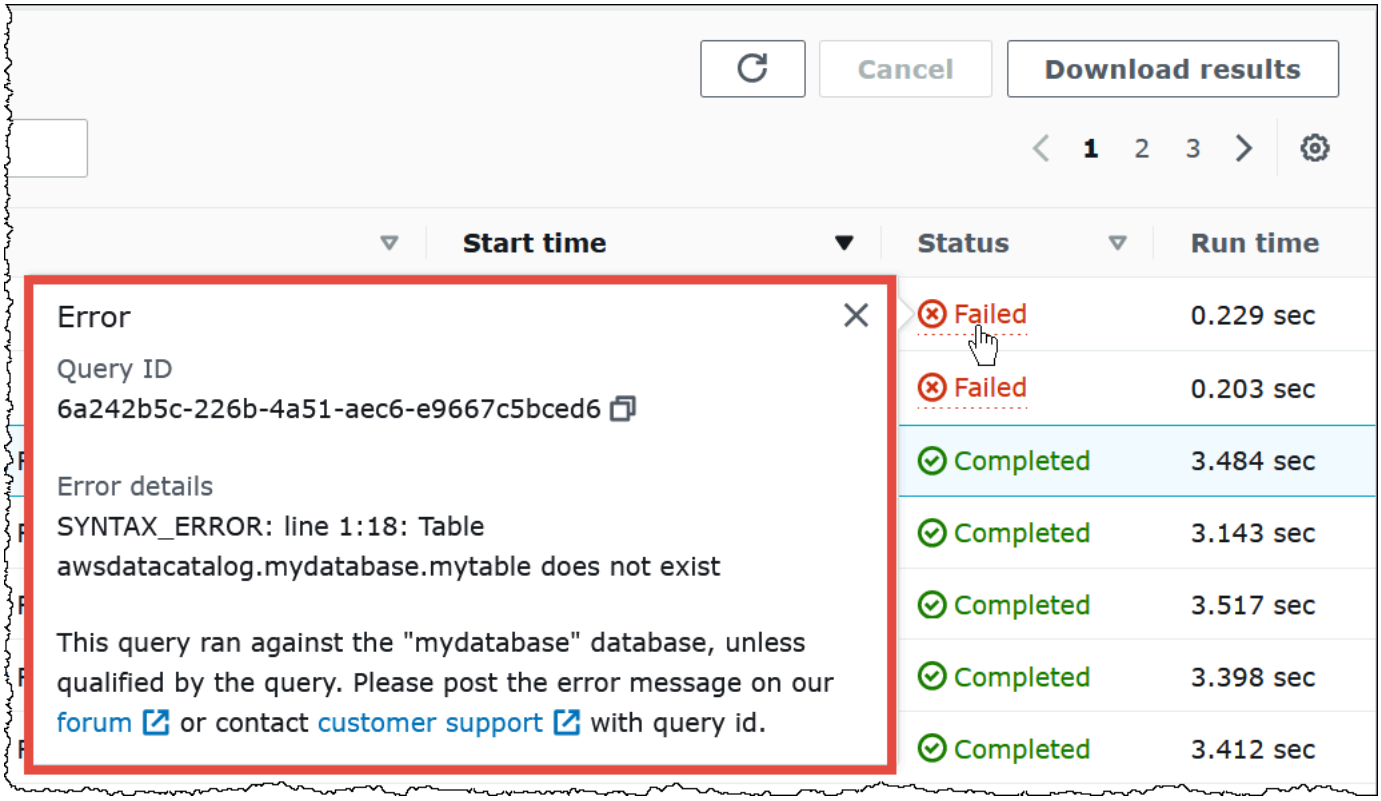
您可以使用 Athena 控制台查看哪些查询成功或失败，并查看失败查询的错误详细信息。Athena 会将查询历史记录保留 45 天。

在 Athena 控制台中查看最近的查询

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 选择 Recent queries（最近的查询）。Recent queries（最近的查询）选项卡显示有关运行的每个查询的信息。
3. 要在查询编辑器中打开查询语句，请选择查询的执行 ID。



4. 要查看失败查询的详细信息，请选择该查询的 Failed（失败）链接。

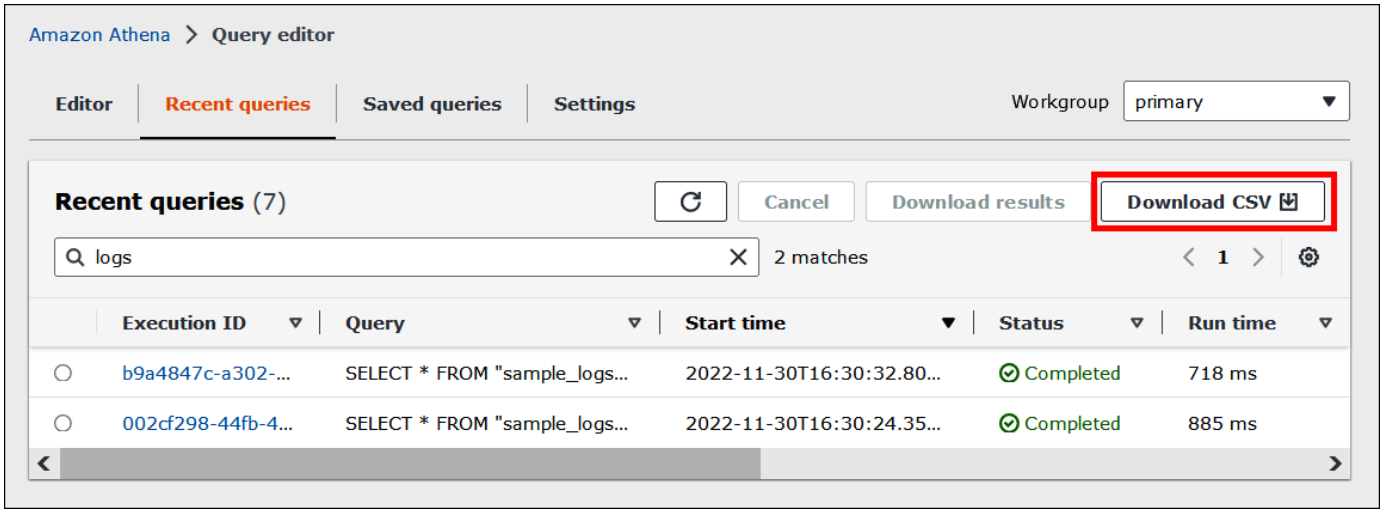


将多个最近的查询下载到 CSV 文件

您可以使用 Athena 控制台的 Recent queries（最近查询）选项卡将一个或多个最近的查询导出到 CSV 文件，以便以表格格式进行查看。下载的文件不包含查询结果，而是包含 SQL 查询字符串本身及有关查询的其他信息。导出的字段包括执行 ID、查询字符串内容、查询开始时间、状态、运行时、扫描的数据量、使用的查询引擎版本和加密方法。您最多可以导出 500 个最近的查询，或者使用搜索框中输入的条件导出最多 500 个筛选后的查询。

将一个或多个最近的查询导出到 CSV 文件

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 选择 Recent queries（最近的查询）。
3. （可选）使用搜索框筛选要下载的最近查询。
4. 选择下载 CSV。



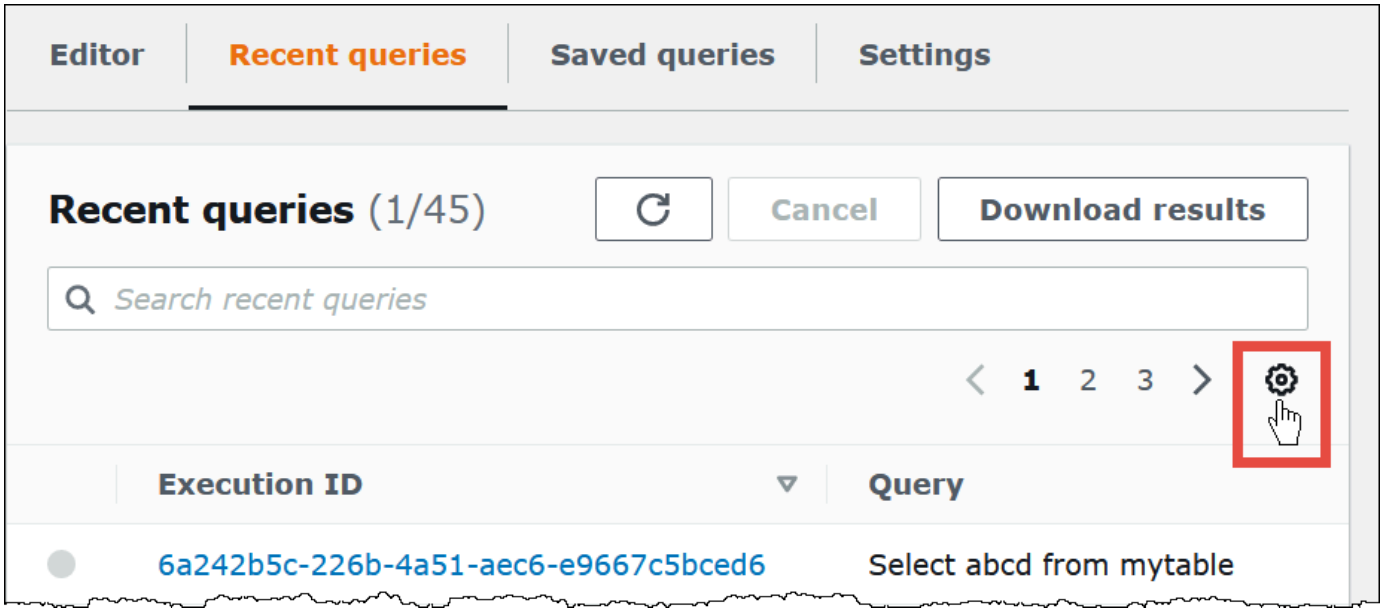
5. 出现文件保存提示时，选择 Save（保存）。默认文件名为 `Recent Queries`，后跟时间戳（例如，`Recent Queries 2022-12-05T16 04 27.352-08 00.csv`）

配置最近的查询显示选项

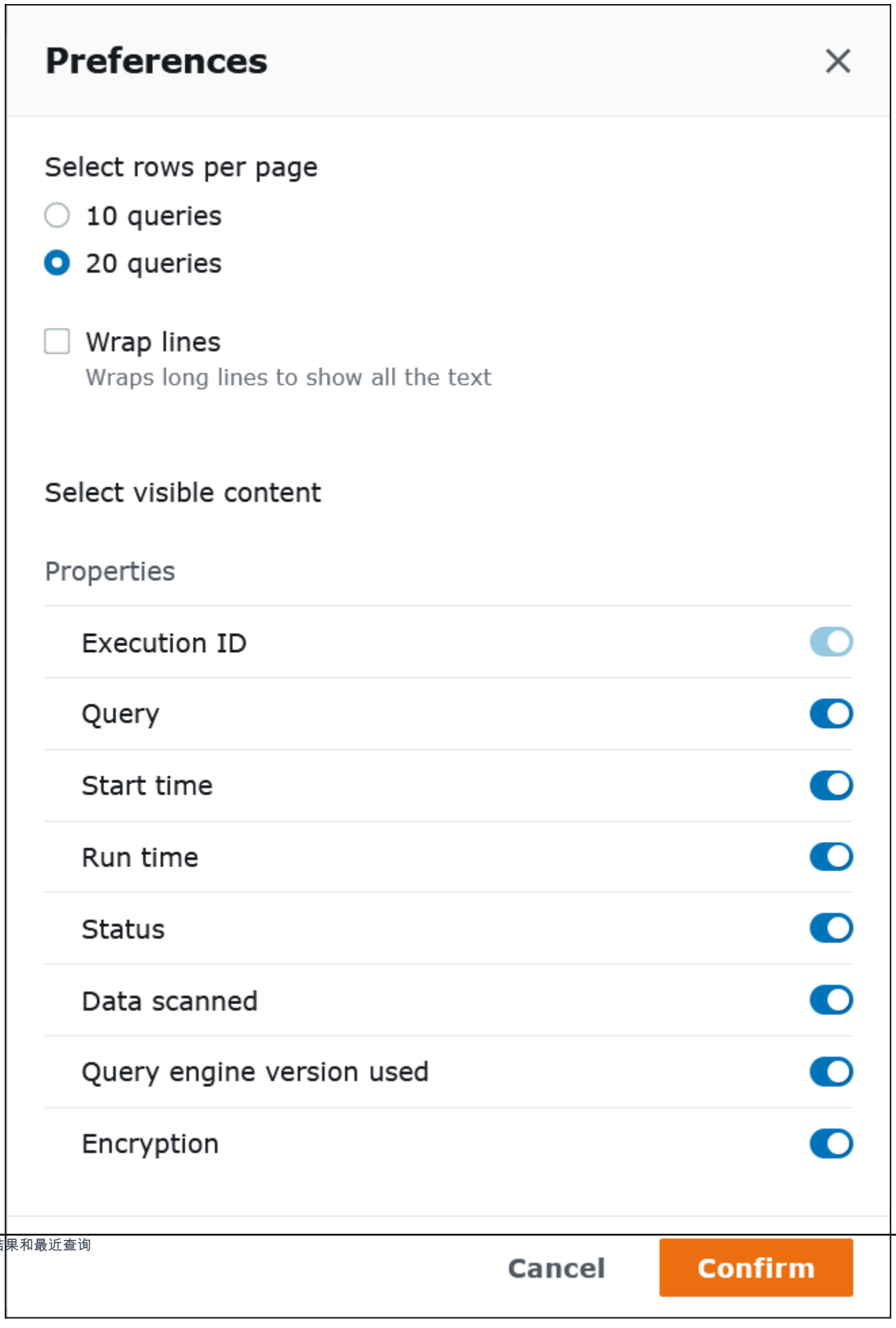
您可以配置 Recent queries（最近的查询）选项卡的选项，例如，要显示的列和文本换行。

配置 Recent queries（最近的查询）选项卡的选项

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 选择 Recent queries（最近的查询）。
3. 选择选项按钮（齿轮图标）。



4. 在 Preferences（首选项）对话框中，选择每页行数、换行行为和要显示的列。


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. 选择确认。

将查询历史记录保留 45 天以上

如果要将查询历史记录保留超过 45 天，您可以检索查询历史记录并将其保存到 Amazon S3 等数据存储中。要自动执行此过程，您可以使用 Athena 和 Amazon S3 API 操作和 CLI 命令。以下过程总结了这些步骤。

以编程方式检索和保存查询历史记录

1. 使用 Athena ListQueryExecutions API 操作或 list-query-executions CLI 命令检索查询 ID。
2. 使用 Athena GetQueryExecution API 操作或 get-query-execution CLI 命令，根据查询的 ID 检索有关各个查询的信息。
3. 使用 Amazon S3 PutObject API 操作或 put-object CLI 命令将信息保存到 Amazon S3 中。

在 Amazon S3 中查找查询输出文件

查询输出文件存储在采用以下路径模式的 Amazon S3 子文件夹中，除非在其配置覆盖了客户端设置的工作组中进行查询。当工作组配置覆盖客户端设置时，查询将使用工作组指定的结果路径。

```
QueryResultsLocationInS3/[QueryName|Unsaved/yyyy/mm/dd/]
```

- *QueryResultsLocationInS3* 是由工作组设置或客户端设置指定的查询结果位置。有关更多信息，请参阅本文后面的the section called “指定查询结果位置”。
- 以下子文件夹仅为从控制台运行且工作组配置未覆盖其结果路径的查询而创建。从 AWS CLI 或使用 Athena API 运行的查询将直接保存到 *QueryResultsLocationInS3*。
  - *QueryName* 是要保存其结果的查询的名称。如果查询已运行但未保存，则使用 Unsaved。
  - *yyyy/mm/dd* 是查询运行的日期。

与 CREATE TABLE AS SELECT 查询关联的文件存储在以上模式的 tables 子文件夹中。

识别查询输出文件

查询输出文件会按照查询名称、查询 ID 和查询运行日期保存到 Amazon S3 中的查询结果位置。每个查询的文件都使用 *QueryID* 命名，这是在每个查询运行时 Athena 为其分配的唯一标识符。

会保存以下文件类型：

文件类型	文件命名模式	描述
查询结果文件	*QueryID*.csv *QueryID*.txt	DML 查询结果文件以逗号分隔值 (CSV) 格式保存。 DDL 查询结果保存为纯文本文件。 您可以在使用控制台时从控制台的 Results（结果）窗格下载这些文件，也可以从查询 History（历史记录）中下载。有关更多信息，请参阅 使用 Athena 控制台下载查询结果文件。
查询元数据文件	*QueryID*.csv.metadata *QueryID*.txt.metadata	DML 和 DDL 查询元数据文件以二进制格式保存，无法人为读取。文件扩展名对应于相关的查询结果文件。Athena 在使用 GetQueryResults 操作读取查询结果时会使用元数据。虽然可以删除这些文件，但我们不建议这样做，因为这样会丢失关于查询的重要信息。
数据清单文件	*QueryID*-manifest.csv	生成数据清单文件以跟踪当 INSERT INTO 查询运行时 Athena 在 Amazon S3 数据源位置创建的文件。如果查询失败，清单也将跟踪查询要写入的文件。清单对于识别由于查询失败而导致的孤立文件很有用。

使用 AWS CLI 标识查询输出位置和文件

要使用 AWS CLI 标识查询输出位置和结果文件，请运行 `aws athena get-query-execution` 命令，如以下示例所示。用查询 ID 替换 *abc1234d-5efg-67hi-jklm-89n0op12qr34*。

```
aws athena get-query-execution --query-execution-id abc1234d-5efg-67hi-
jklm-89n0op12qr34
```

该命令返回的输出类似于下方内容。有关每个输出参数的说明，请参阅 AWS CLI 命令参考中的 get-query-execution。

```
{
    "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"
    }
}
```

重用查询结果

在 Athena 中重新运行查询时，可以选择重用上次存储的查询结果。此选项可以提高性能并降低扫描字节数方面的成本。重用查询结果非常有用，例如，如果您知道结果在指定时间范围内不会发生变化，则可以指定重用查询结果的最大期限。只要存储结果不超过指定期限，Athena 就会使用该结果。有关更多信息，请参阅 AWS 大数据博客中的使用 Amazon Athena 降低成本并提高查询性能。

> **Note**
>
> 查询结果重用功能需要 Athena 引擎版本 3。有关更改引擎版本的信息，请参阅 更改 Athena 引擎版本。

主要特征

- 重用查询结果是一项按查询选择使用的功能。您可以针对每个查询启用查询结果重用。
- 可以指定重用查询结果的最大期限（以分钟、小时或天为单位）。无论使用何种时间单位，可指定的最大期限均为 7 天。（ 默认为 60 分钟。 ）
- 当您为查询启用结果重用时，Athena 会在同一工作组中查找该查询的先前执行情况。如果 Athena 找到相应的存储查询结果，它不会重新运行查询，而是指向先前的结果位置或从中获取数据。

- 对于启用结果重用选项的任何查询，Athena 仅在满足以下所有条件时才会重用保存到工作组文件夹的最后一个查询结果：
  - 查询字符串完全匹配。
  - 数据库与目录名称匹配。
  - 先前的结果不超过指定的最大期限，或者如果未指定最大期限，则不超过 60 分钟。
  - Athena 仅重用与当前执行具有完全相同的结果配置的执行文件。
  - 您可以访问查询中引用的所有表。
  - 您可以访问存储先前结果的 S3 文件位置。

如果不满足这些条件中的任何一个，Athena 将在不使用缓存结果的情况下运行查询。

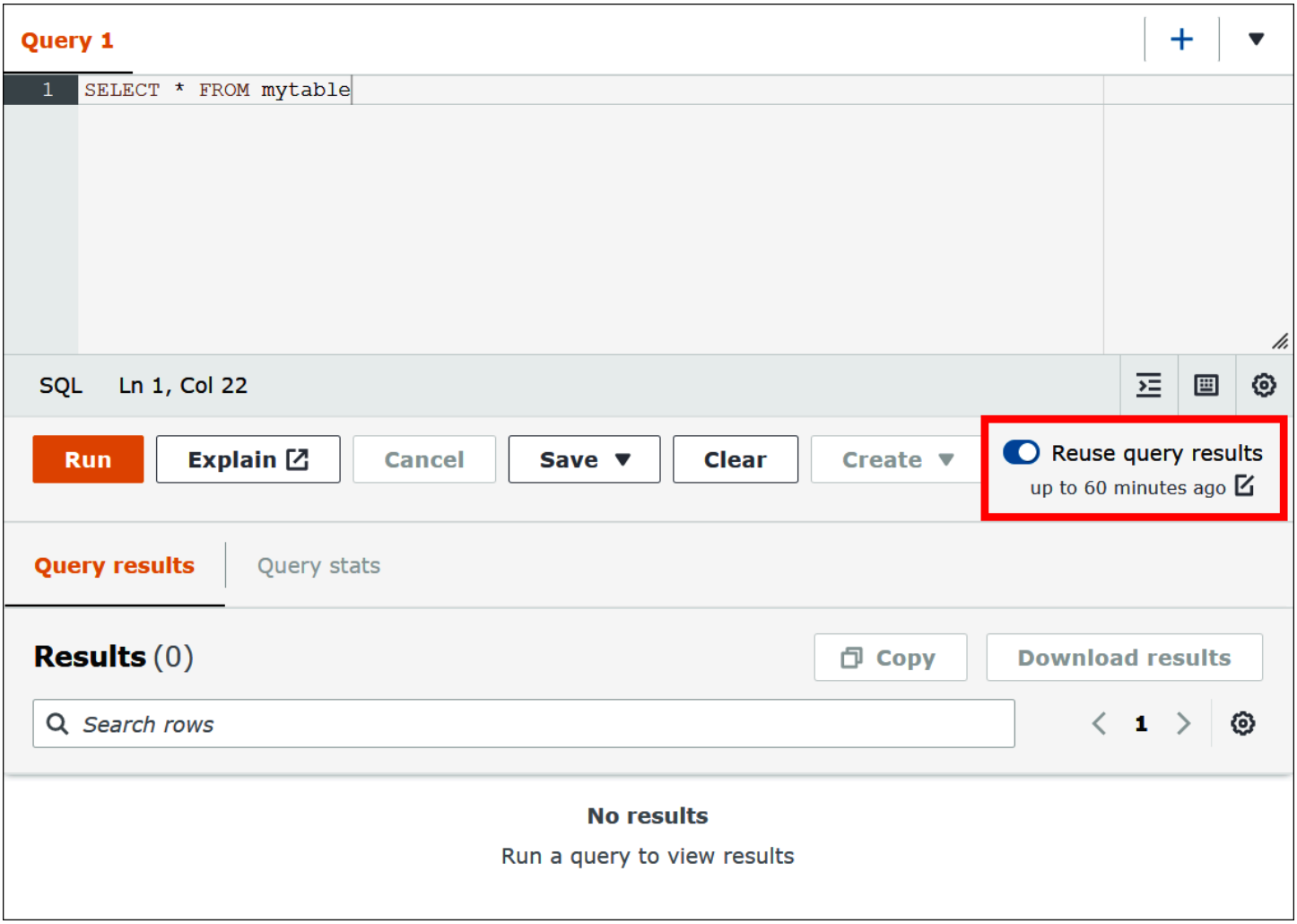
注意事项和限制

使用查询结果重用功能时，请记住以下几点：

- Athena 仅在同一工作组内重用查询结果。
- 重用查询结果功能遵循工作组配置。如果您覆盖查询的结果配置，该功能将被禁用。
- 支持向 AWS Glue 注册 Apache Hive、Apache Hudi、Apache Iceberg 和 Linux Foundation Delta Lake 表。不支持外部 Hive 元存储。
- 不支持引用联合目录或外部 Hive 元存储的查询。
- Lake Formation 受管表不支持查询结果重用。
- 在 Lake Formation 中将表源的 Amazon S3 位置注册为数据位置时，不支持重用查询结果。
- 不支持具有行和列权限的表。
- 不支持具有精细访问控制（例如，列或行筛选）的表。
- 任何引用不支持的表的查询都不能重用查询结果。
- Athena 要求您具有 Amazon S3 读取权限，以便重用先前生成的输出文件。
- 重用查询结果功能假定先前结果的内容未被修改。Athena 在使用先前的结果之前，不会检查其完整性。
- 如果先前执行的查询结果已被删除或移至 Amazon S3 中的其他位置，则随后执行的同一查询将不会重用查询结果。
- 因而可能会返回过时的结果。在达到指定的最大重用期限之前，Athena 不会检查源数据是否更改。
- 如果有多个结果可供重用，Athena 会使用最新的结果。
- 使用非确定性运算符或函数（例如 `rand()` 或 `shuffle()`）的查询不使用缓存结果。例如，不包含 `ORDER BY` 的 `LIMIT` 是不确定性的，因而不会缓存，但包含 `ORDER BY` 的 `LIMIT` 是确定性的，所以会缓存。
- Athena 控制台、Athena API 和 JDBC 驱动程序支持查询结果重用。目前，ODBC 驱动程序对查询结果重用的支持仅适用于 Windows。
- 要将查询结果重用功能与 JDBC 结合使用，所需的最低驱动程序版本为 2.0.34.1000。对于 ODBC，所需的最低驱动程序版本为 1.1.19.1002。有关驱动程序下载信息，请参阅 通过 ODBC 和 JDBC 驱动程序连接到 Amazon Athena。
- 对于使用多个数据目录的查询，不支持查询结果重用。
- 对于包含超过 20 个表的查询，不支持查询结果重用。

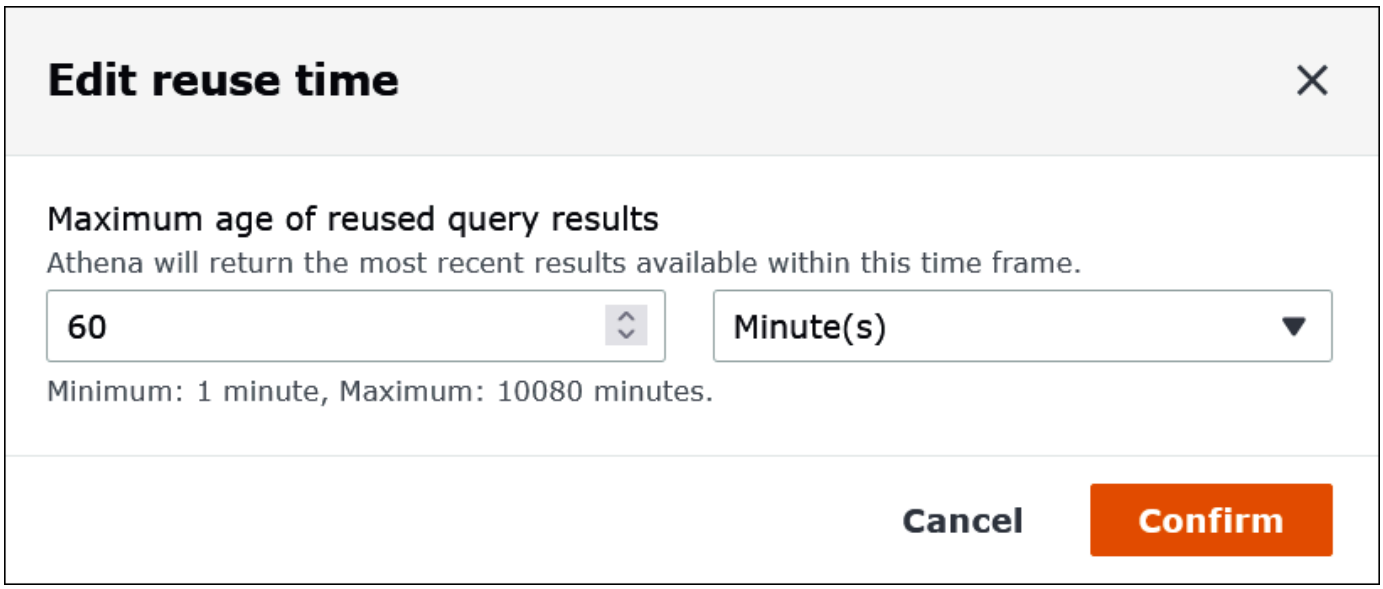
在 Athena 控制台中重用查询结果

要使用该功能，请在 Athena 查询编辑器中启用 Reuse query results（重用查询结果）选项。

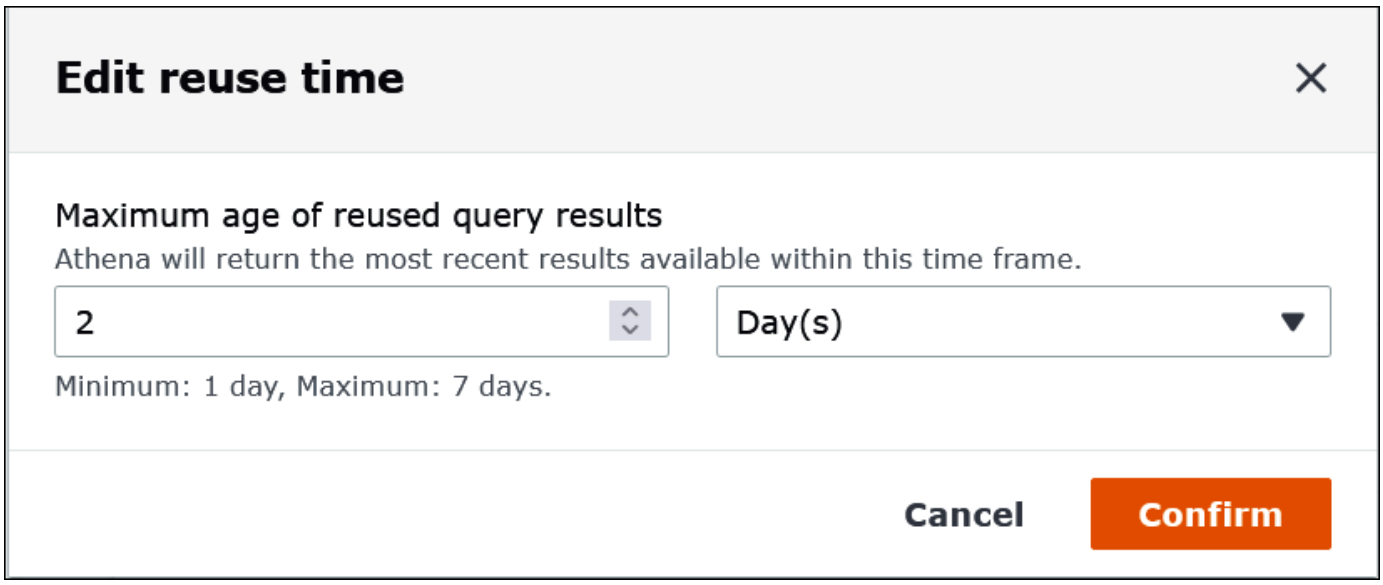


配置重用查询结果功能

1. 在 Athena 查询编辑器中的 Reuse query results（重用查询结果）选项下，选择 up to 60 minutes ago（截至 60 分钟前）旁边的编辑图标。
2. 在 Edit reuse time（编辑重用时间）对话框中，从右侧的框中选择一个时间单位（分钟、小时或天）。
3. 在左侧的框中，输入或选择要指定的时间单位数。无论选择何种时间单位，可输入的最大时间均为七天。

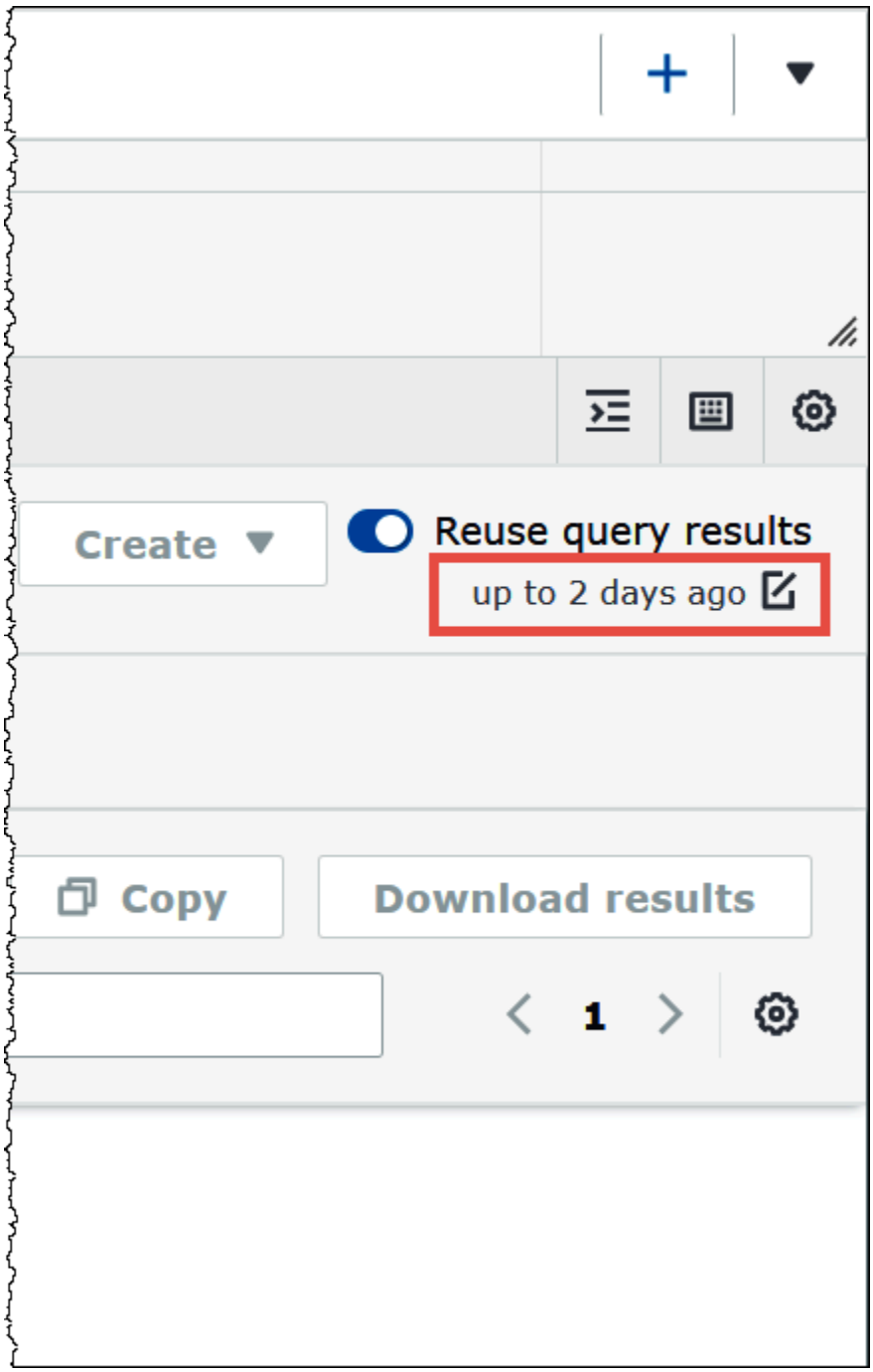


以下示例指定最大重用时间为两天。



4. 选择确认。

系统会显示一条横幅确认您的配置更改，同时 Reuse query results（重用查询结果）选项会显示新设置。

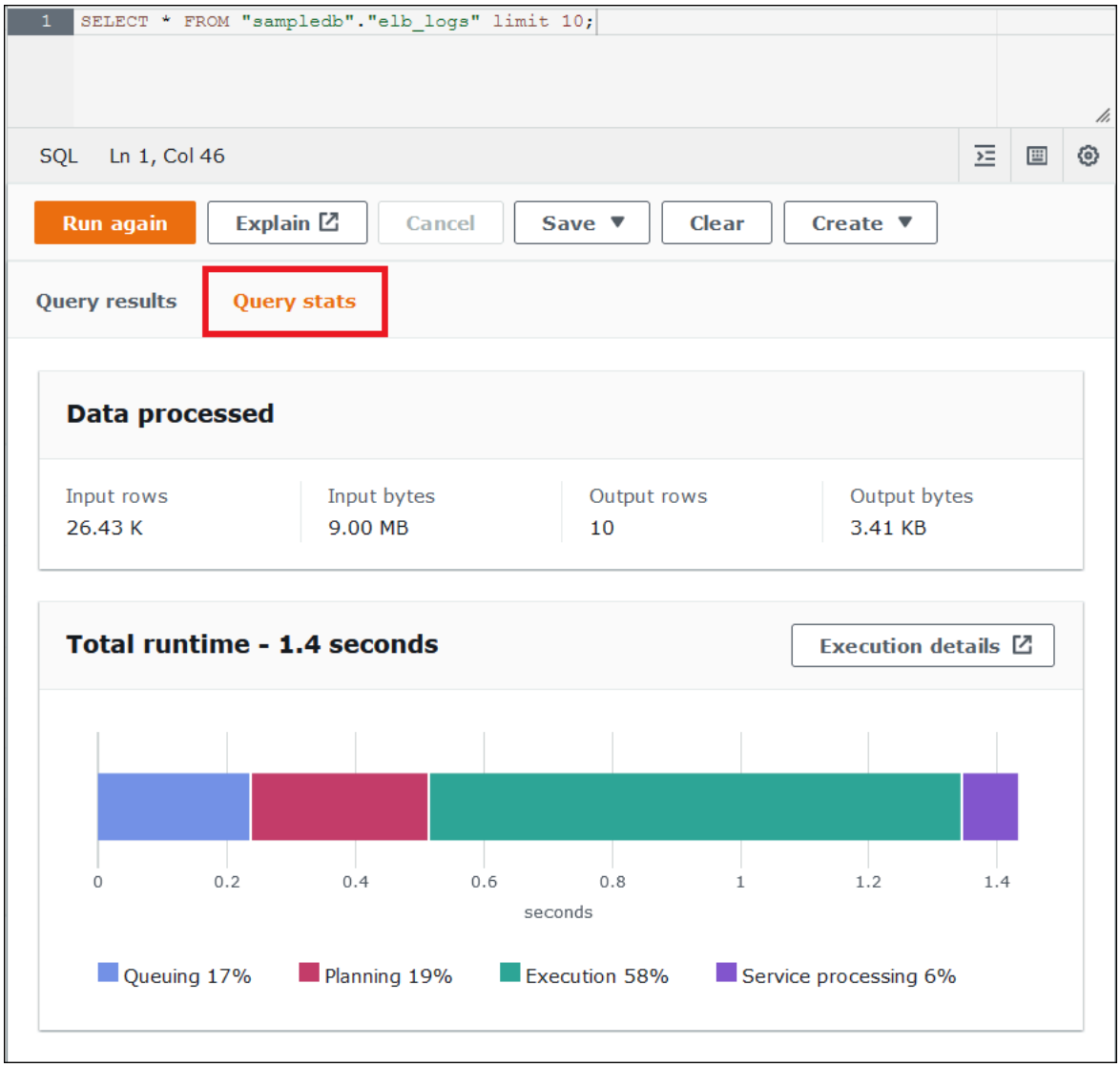


查看已完成查询的统计数据和执行详细信息

运行查询后，您可以获得处理的输入和输出数据的统计信息，查看查询每个阶段所用时间的图形展示，并以交互方式浏览执行详细信息。

查看已完成查询的查询统计信息

1. 在 Athena 查询编辑器中运行查询后，请选择 Query stats（查询统计信息）选项卡。



Query stats（查询统计信息）选项卡提供以下信息：

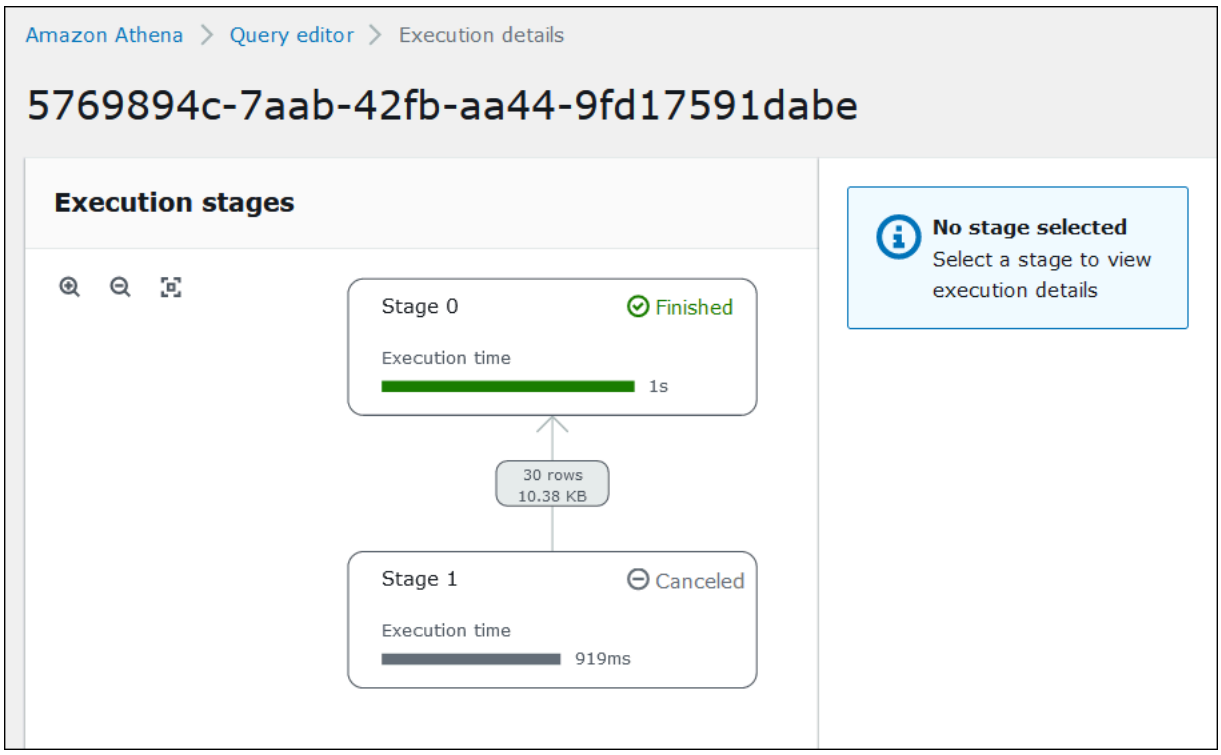
- 已处理数据 — 显示已处理的输入行数和字节数，以及输出的行数和字节数。
- 运行时合计 — 显示运行查询所用的总时间，以及这段时间内用于排队、计划、执行和服务处理的时间的图形展示。

> **Note**
>
> 如果查询具有在 Lake Formation 中定义的行级筛选条件，不会显示阶段级输入和输出行数以及数据大小信息。

2. 要交互式浏览查询的运行方式的相关信息，请选择 Execution details（执行详细信息）。

Execution details（执行详细信息）页面显示查询的执行 ID 和查询中从零开始的各阶段图表。各个阶段从下到上按顺序从开始到结束排列。每个阶段的标签都显示了阶段运行所花费的总时间。

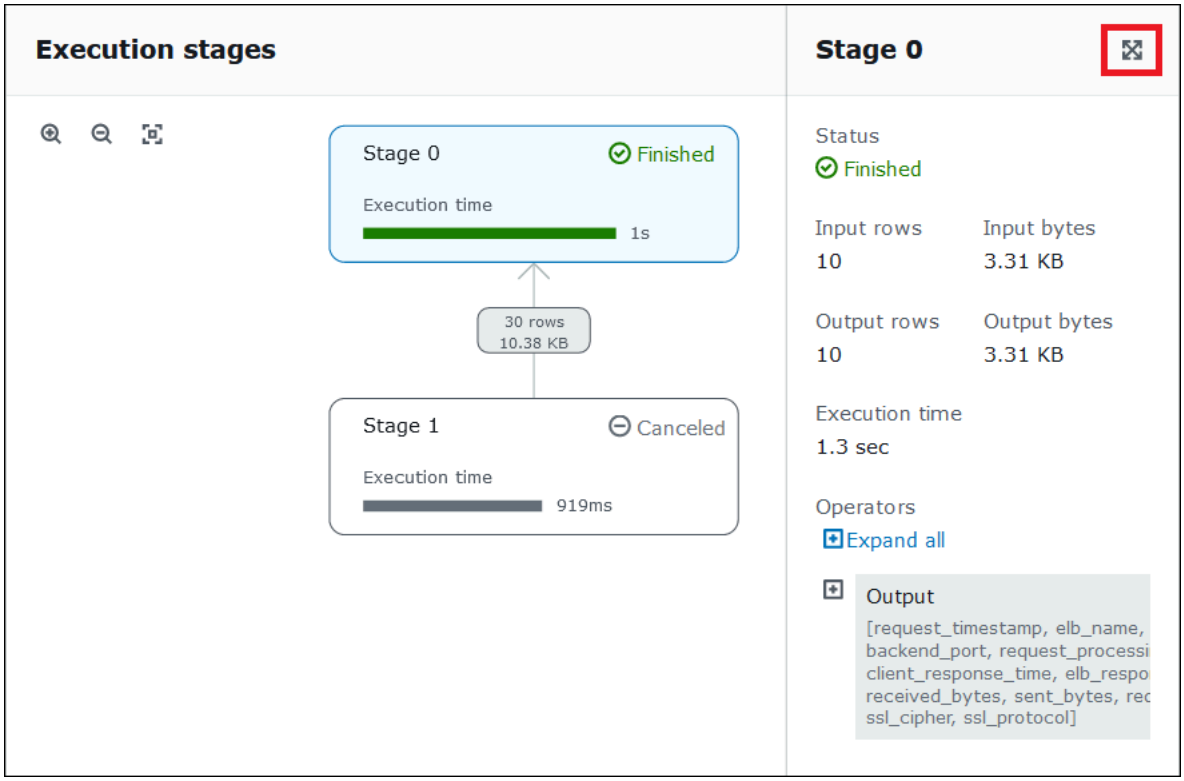
> **Note**
>
> 查询的总运行时间和执行阶段时间之间通常会有很大差异。例如，以分钟为单位的总运行时间的查询，可以显示以小时为单位的某个阶段的执行时间。由于阶段是跨多个任务并行执行的逻辑计算单元，所以阶段的执行时间就是其所有任务的总执行时间。虽然存在这种差异，但阶段的执行时间可以作为查询中计算密集度最高的阶段的相对指标。



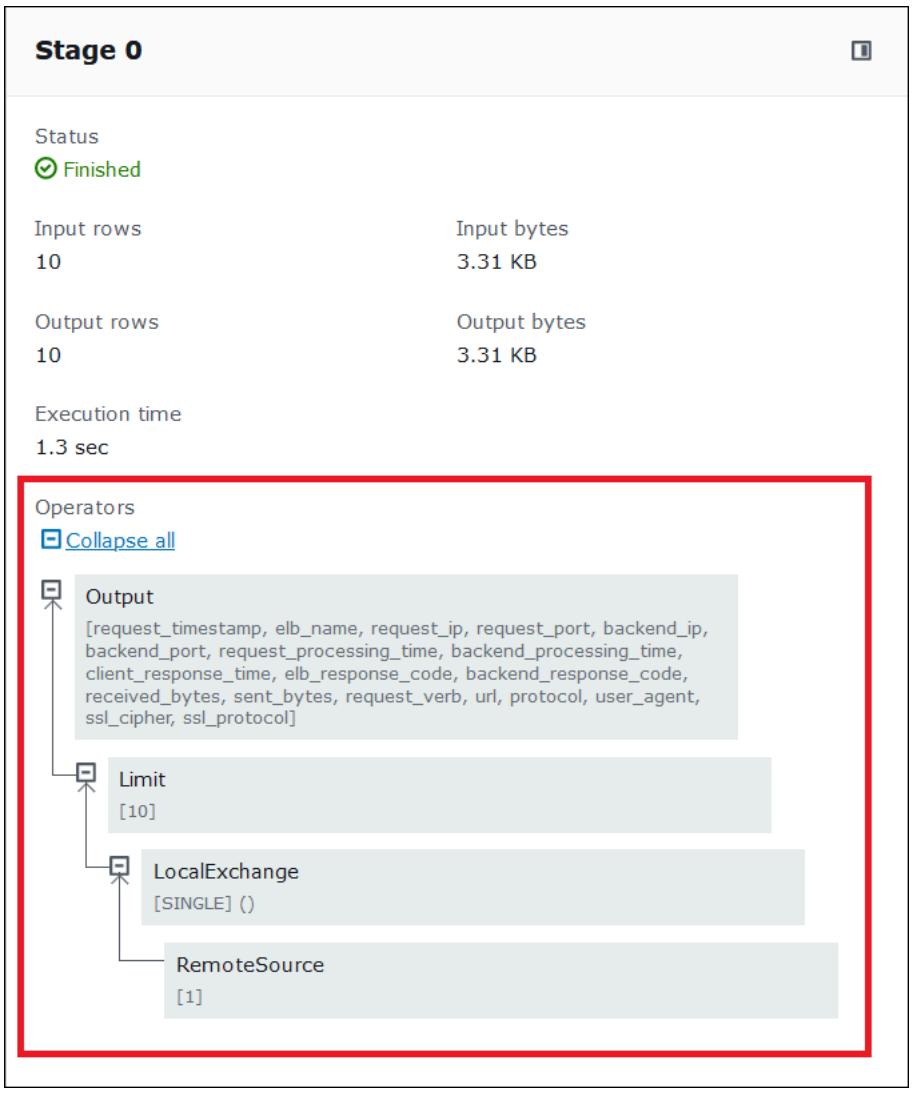
要导航图表，请使用以下选项：

- 要放大或缩小，请滚动鼠标或使用放大图标。
- 要调整图表以适合屏幕，请选择缩放至适合图标。
- 要移动图表，请拖动鼠标指针。

3. 要查看阶段的更多详细信息，请选择阶段。右侧的阶段详细信息窗格显示输入和输出的行数和字节数，以及运算符树。



4. 要查看阶段的整体详细信息，请选择详细信息窗格右上角的展开图标。
5. 要获取有关阶段各部分的信息，请展开运算符树中的一个或多个项目。



有关执行详情的更多信息，请参阅 了解 Athena EXPLAIN 语句结果。

其他 资源

有关详细信息，请参阅以下资源：

查看 SQL 查询的执行计划

在 Athena 中使用 EXPLAIN 和 EXPLAIN ANALYZE

使用视图

在 Amazon Athena 中，视图是逻辑表，而非物理表。每次在查询中引用视图时，定义该视图的查询都会运行。

您可以从 SELECT 查询创建视图，然后在未来的查询中引用该视图。有关更多信息，请参阅 CREATE VIEW。

主题

何时使用视图？

您可能希望创建视图以达到以下目的：

- 查询数据子集。例如，您可以根据原始表创建一个包含列子集的视图，以简化查询数据。
- 将多个表合并在一个查询中。如果您有多个表，希望使用 UNION ALL 将它们组合起来，可以创建一个视图，利用表达式简化针对组合表的查询。
- 隐藏现有基本查询的复杂性和简化用户运行的查询。基本查询通常包括表之间的联接、列列表中的表达式和其他 SQL 语法，这使理解和调试查询变得困难。您可以创建一个能够隐藏复杂性和简化查询的视图。
- 试验优化方法和创建优化查询。例如，如果您发现某种 WHERE 条件、JOIN 命令或其他表达式的组合能带来最佳性能，则可以使用这些子句和表达式创建一个视图。应用程序然后可以针对此视图执行相对简单的查询。如果后来您又发现更好的优化原始查询的办法，则当您重新创建视图时，所有应用程序会立即利用优化的基本查询。
- 隐藏底层表和列名称，并最大程度减少维护问题（如果这些列名称改变）。在这种情况下，您使用新名称重新创建视图。使用该视图而非底层表的所有查询保持运行，无需更改。

Athena 中受支持的视图操作

Athena 支持下列视图操作。您可以在查询编辑器中运行这些命令。

语句	描述
CREATE VIEW	从指定的 SELECT 查询创建新视图。有关更多信息，请参阅 创建视图。 可选的 OR REPLACE 子句允许您通过替换来更新现有视图。
DESCRIBE VIEW	显示命名视图的列列表。这能让您检查复杂视图的属性。
DROP VIEW	删除现有视图。如果该视图不存在，可选 IF EXISTS 子句将抑制错误出现。
SHOW CREATE VIEW	显示创建指定视图的 SQL 语句。
SHOW VIEWS	列出指定数据库或当前数据库（如果省略数据库的名称）中的视图。将可选 LIKE 子句与一个正则表达式结合使用来限制视图名称列表。您还可以在控制台的左窗格中看到视图列表。
SHOW COLUMNS	列出视图的架构中的列。

视图注意事项

以下注意事项适用于在 Athena 中创建和使用视图：

- 在 Athena 中，您可以预览和使用在 Athena 控制台、AWS Glue Data Catalog（如果您已迁移并使用它）中或通过运行于连接到同一目录的 Amazon EMR 集群上的 Presto 创建的视图。不能预览或添加以其他方式创建的 Athena 视图。
- 如果您在通过 AWS Glue Data Catalog 创建视图，则必须包括 PartitionKeys 参数并将其值设置为空列表，如下所示："PartitionKeys":[]。否则，您在 Athena 中的视图查询将失败。以下示例显示使用 "PartitionKeys":[] 从 Data Catalog 创建的视图：

```
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":[]}'
```

- 如果您已在 Data Catalog 中创建了 Athena 视图，则数据目录会将视图视为表。您可以使用在 Data Catalog 中表级精细访问控制来限制对这些视图的访问。
- Athena 可防止您运行递归视图，如有运行，则会显示错误消息。递归视图是引用自身的视图查询。
- Athena 在检测到过时视图时会显示错误消息。发生以下一种情况时，会报告过时的视图：
  - 视图引用了不存在的表或数据库。
  - 在引用的表中进行了架构或元数据更改。
  - 删除了引用的表并使用不同的架构或配置重新创建。
- 您可以创建并运行嵌套视图，只要嵌套视图背后的查询有效以及表和数据库存在。

视图限制

- Athena 视图名称不能包含下划线 (_) 之外的特殊字符。有关更多信息，请参阅 表、数据库和列的名称。
- 避免使用保留关键字来命名视图。如果使用保留关键字，请在视图查询中使用双引号将保留关键字括起来。请参阅 保留关键字。
- 您不能将 Athena 中创建的视图与外部 Hive 元存储或 UDF 结合使用。有关使用在 Hive 外部创建的视图的信息，请参阅 使用 Hive 视图。
- 您不能将视图与地理空间函数结合使用。
- 您无法使用视图管理 Amazon S3 中数据的访问控制。要查询视图，您需要相应权限才能访问存储在 Amazon S3 中的数据。有关更多信息，请参阅 对 Amazon S3 的访问权限。
- 虽然 Athena 引擎版本 2 和 Athena 引擎版本 3 都支持跨账户查询视图，但您无法创建包含跨账户 AWS Glue Data Catalog 的视图。有关访问跨账户数据目录的信息，请参阅 授予 AWS Glue 数据目录跨账户访问权限。
- Athena 视图不支持 Hive 或 Iceberg 隐藏的元数据列 `$bucket`、`$file_modified_time`、`$file_size` 和 `$partition`。有关在 Athena 中使用 `$path` 元数据列的信息，请参阅 获取 Amazon S3 中源数据的文件位置 。

在控制台中使用视图

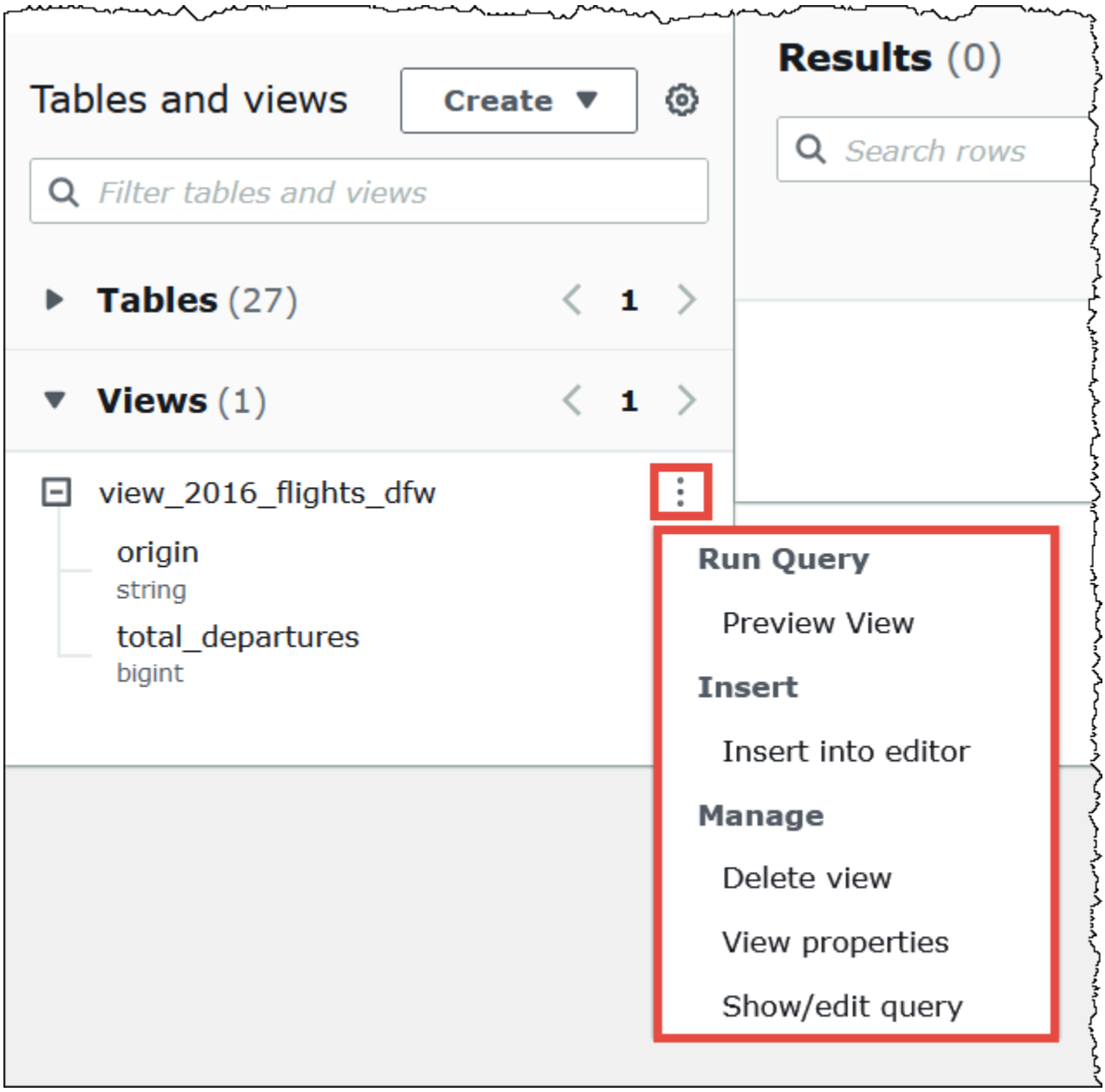
在 Athena 控制台中，您可以：

- 在列出表的左窗格中找到所有视图。
- 筛选视图。
- 预览视图、显示其属性、编辑或删除视图。

显示视图的操作

只有创建了视图后，该视图才会出现在控制台中。

1. 在 Athena 控制台中，选择 Views（视图），然后选择一个视图，展开它并显示视图中的列。
2. 选择视图旁边的三个竖直的点，以显示视图的操作列表。



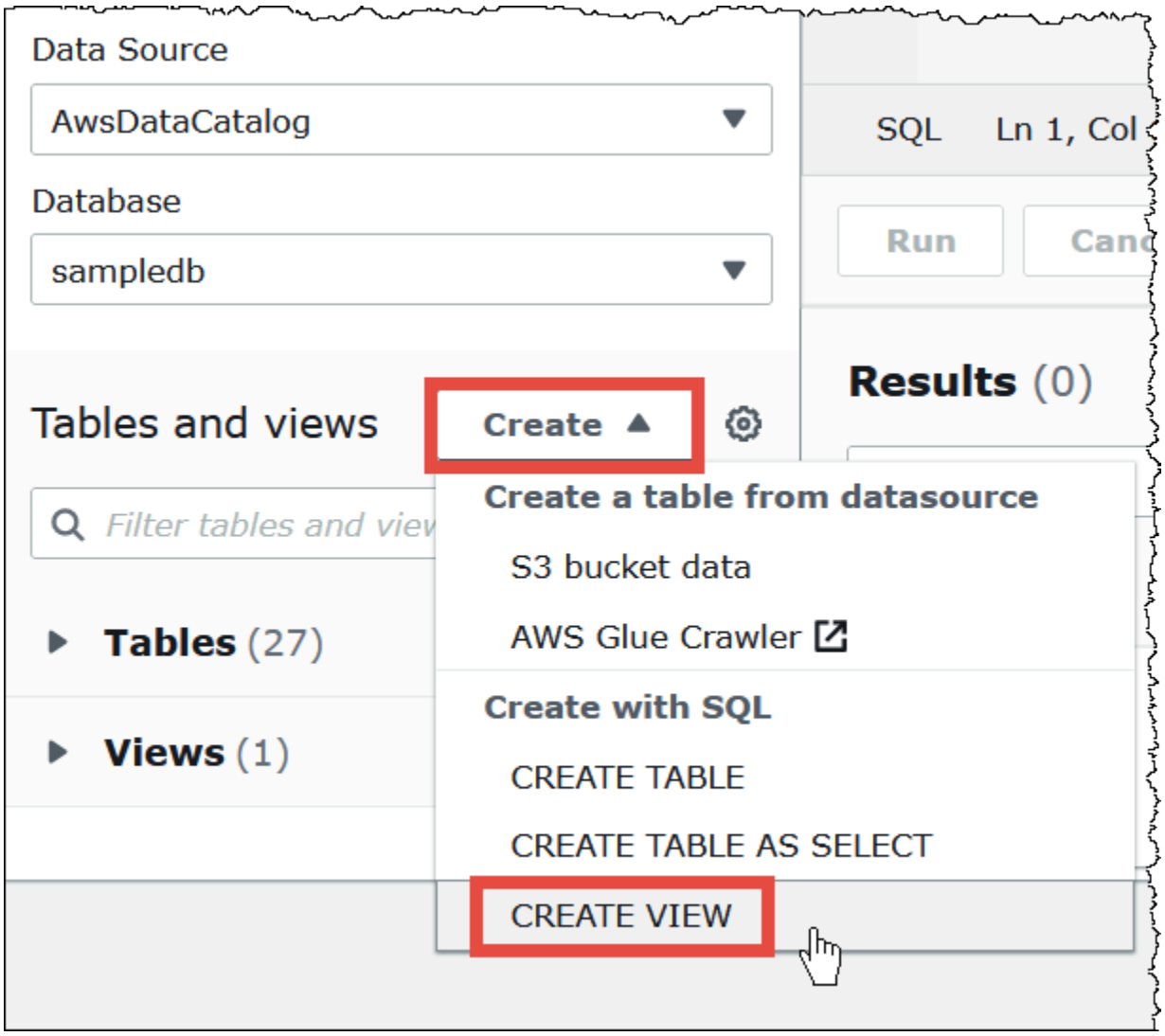
3. 选择操作以预览视图、将视图名称插入查询编辑器、删除视图、查看视图属性，或在查询编辑器中显示并编辑视图。

创建视图

您可以通过使用模板或运行现有查询以在 Athena 控制台中创建视图。

使用模板创建视图

1. 在 Athena 控制台中，选择 Tables and views（表和视图）旁边的 Create（创建），然后选择 Create view（创建视图）。



   此操作会将可编辑的视图模板放入查询编辑器中。

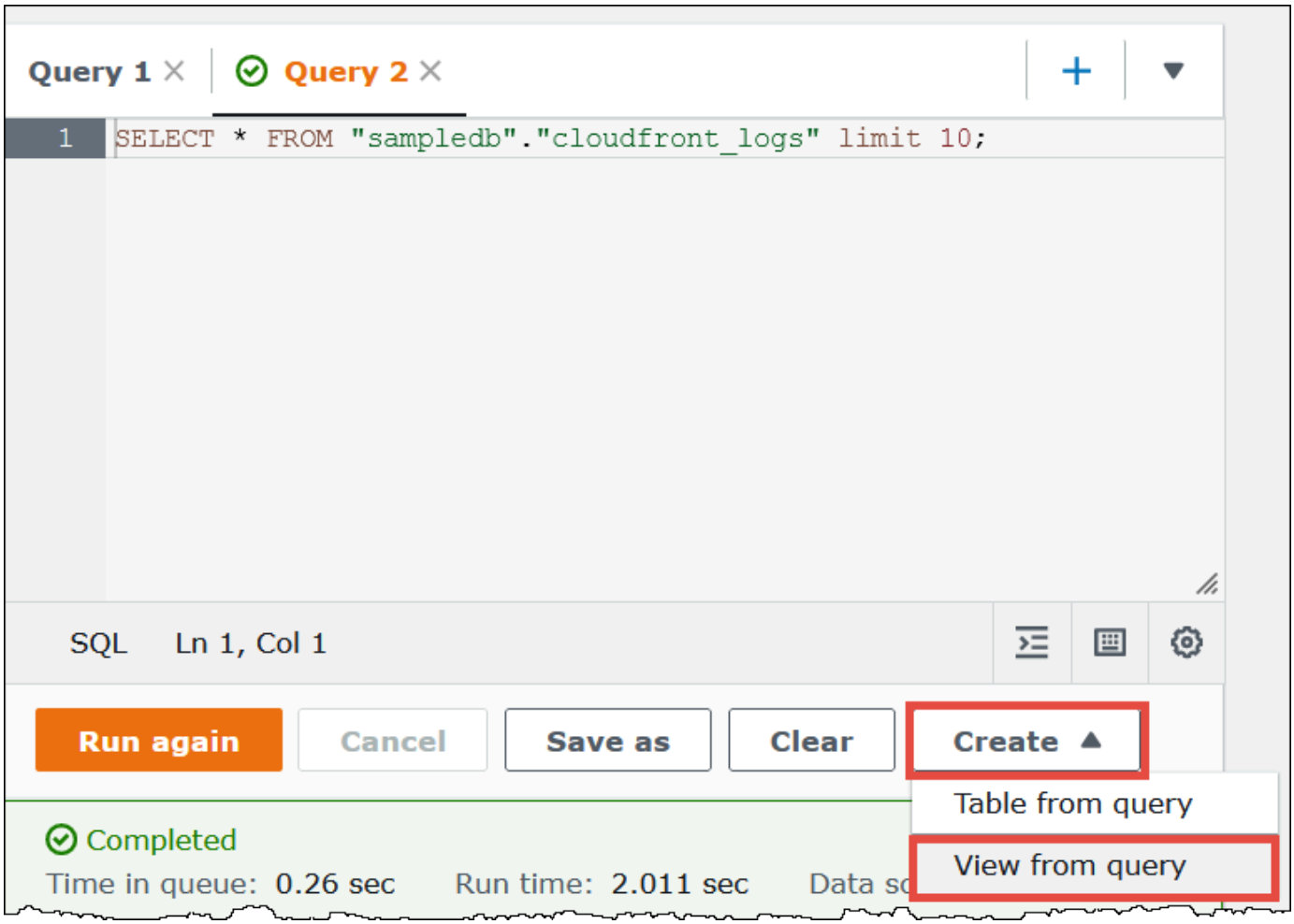
2. 根据您的要求编辑视图模板。在语句中输入视图的名称时，请记住，视图名称不能包含除下划线 ((_)) 以外的特殊字符。请参阅 表、数据库和列的名称。避免使用保留关键字来命名视图。

   有关创建视图的更多信息，请参阅 CREATE VIEW 和 视图示例。

3. 选择 Run（运行）以创建视图。该视图将显示在 Athena 控制台中的视图列表中。

通过现有查询创建视图

1. 使用 Athena 查询编辑器运行现有查询。
2. 在查询编辑器窗口下，选择 Create（创建），然后选择 View from query（通过查询查看）。



3. 在 Create View（创建视图）对话框中，输入视图名称并选择 Create（创建）。视图名称不能包含除下划线（_）以外的特殊字符。请参阅 表、数据库和列的名称。避免使用保留关键字来命名视图。

   Athena 将视图添加到控制台的视图列表中，并显示查询编辑器中视图的 `CREATE VIEW` 语句。

注意

- 如果删除了作为其他表的依据的表，并且尝试运行该视图，Athena 将显示一条错误消息。
- 可以创建一个嵌套视图，即在现有视图之上的视图。Athena 可防止您运行引用自身的递归视图。

视图示例

要显示视图查询的语法，请使用 SHOW CREATE VIEW。

Example 示例 1

请考虑以下两个表：表 employees，具有两列，分别是 id 和 name，以及表 salaries，也具有两列，分别是 id 和 salary。

在本示例中，我们创建名为 name_salary 的视图作为 SELECT 查询，该查询将获得一个 ID 列表，这些 ID 映射到来自表 employees 和 salaries 的薪水：

```
CREATE VIEW name_salary AS
SELECT
 employees.name,
 salaries.salary
FROM employees, salaries
WHERE employees.id = salaries.id
```

Example 示例 2

在以下示例中，我们创建一个名为 view1 的视图，以使您能够隐藏较复杂的查询语法。

该视图运行在两个表，即 table1 和 table2 上，其中每个表都是不同的 SELECT 查询。该视图选择来自 table1 的列并将结果与 table2 联接。联接基于在两个表中都存在的列 a。

```
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;
```

有关查询联合视图的信息，请参阅 查询联合视图。

使用 AWS Glue Data Catalog 视图

> 该功能为预览版，可能会发生变化。有关更多信息，请参阅 AWS 服务条款文档中的“测试版和预览”部分。

当您想跨 AWS 服务（例如 Amazon Athena 和 Amazon Redshift）使用单一通用视图时，请使用 AWS Glue Data Catalog 视图。在 Data Catalog 视图中，访问权限由创建视图的用户（而不是查询视图的用户）定义。这种授权方法称为定义程序语义。

以下使用案例展示如何使用 Data Catalog 视图。

- 优化访问控制 – 您可以创建视图，根据用户所需的权限级别来限制数据访问。例如，您可以使用 Data Catalog 视图来防止非人力资源（HR）部门的员工查看个人身份信息。
- 确保记录完整 – 通过对 Data Catalog 视图应用某些筛选器，您可以确保 Data Catalog 视图中的数据记录始终完整。
- 增强安全性 – 在 Data Catalog 视图中，用于创建视图的查询定义必须完好无损，才能创建视图。这样的话，Data Catalog 视图不易受到恶意行为者的 SQL 命令的影响。
- 防止访问基础表 – 定义程序语义允许用户访问视图，而无需向他们提供基础表。只有定义视图的用户才需要访问表。

Data Catalog 视图定义存储于 AWS Glue Data Catalog。这意味着，您可以使用 AWS Lake Formation 通过资源授权、列授权或基于标签的访问控制来授予访问权限。有关在 Lake Formation 中授予和撤消访问权限的更多信息，请参阅《AWS Lake Formation Developer Guide》中的 Granting and revoking permissions on Data Catalog resources。

权限

Data Catalog 视图需要三个角色：`Lake Formation Admin`、`Definer` 和 `Invoker`。

- **`Lake Formation Admin`** – 有权配置所有 Lake Formation 权限。
- **`Definer`** – 创建 Data Catalog 视图。`Definer` 角色必须对视图定义引用的所有基础表具有完全的可授予 `SELECT` 权限。
- **`Invoker`** – 可以查询 Data Catalog 视图或检查其元数据。

Definer 角色的信任关系必须允许 AWS Glue 和 Lake Formation 服务主体采取 sts:AssumeRole 操作，如以下示例所示。

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

还需要针对 Athena 访问的 IAM 权限。有关更多信息，请参阅 Amazon Athena 的 AWS 托管策略。

限制

- Data Catalog 视图不能引用其他视图。
- 在视图定义中，您最多可以引用 10 个表。
- 基础表必须使用 Lake Formation 注册。
- DEFINER 主体只能是 IAM 角色。
- DEFINER 角色必须对基础表具有完全的 SELECT（可授予）权限。
- 不支持 UNPROTECTED Data Catalog 视图。
- 视图定义中不支持用户定义的函数（UDF）。
- Athena 联合数据来源不能用于 Data Catalog 视图。
- 外部 Hive 元存储不支持 Data Catalog 视图。
- Athena 在检测到过时视图时会显示错误消息。发生以下一种情况时，会报告过时的视图：
  - 视图引用了不存在的表或数据库。
  - 在引用的表中进行了架构或元数据更改。
  - 删除了引用的表并使用不同的架构或配置重新创建。

创建 Data Catalog 视图

以下示例语法展示 Definer 角色的用户如何创建 orders_by_date Data Catalog 视图。该示例假设 Definer 角色对 default 数据库中的 orders 表具有完全 SELECT 权限。

```
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
```

查询 Data Catalog 视图

视图创建后，Lake Formation Admin 可以向 Invoker 主体授予对 Data Catalog 视图的 SELECT 权限。然后，Invoker 主体无需访问视图所引用的基础基表，即可查询视图。以下是 Invoker 查询示例。

```
SELECT * from orders_by_date where price > 5000
```

更新 Data Catalog 视图

Lake Formation Admin 或 Definer 可以使用 ALTER VIEW UPDATE DIALECT 语法来更新视图定义。以下示例修改视图定义，从 returns 表（而不是 orders 表）中选择列。

```
ALTER VIEW orders_by_date UPDATE DIALECT
AS
SELECT return_date, sum(totalprice) AS price
FROM returns
WHERE order_city = 'SEATTLE'
GROUP BY orderdate
```

有关用于创建和管理 Data Catalog 视图的语法的更多信息，请参阅 Glue Data Catalog 视图语法。

Glue Data Catalog 视图语法

该功能为预览版，可能会发生变化。有关更多信息，请参阅 AWS 服务条款文档中的“测试版和预览”部分。

本部分介绍用于创建和管理 AWS Glue Data Catalog 视图的数据定义语言（DDL）命令。

ALTER VIEW DIALECT

要更新 Data Catalog 视图，您可以添加引擎方言，或者更新或删除现有的引擎方言。只有 Lake FormationAdmin 和 Definer（创建视图的用户）才有权在 Data Catalog 视图上使用 ALTER VIEW DIALECT 语句。

语法

```
ALTER VIEW name [ FORCE ] [ ADD|UPDATE ] DIALECT AS query
```

```
ALTER VIEW name [ DROP ] DIALECT
```

FORCE

FORCE 关键字会导致视图中相互冲突的引擎方言信息被新定义覆盖。当 Data Catalog 视图更新导致现有引擎方言中的视图定义相互冲突时，FORCE 关键字很有用。假设 Data Catalog 视图同时使用 Athena 和 Amazon Redshift 方言，并且更新导致与视图定义中的 Amazon Redshift 发生冲突。在这种情况下，您可以使用 FORCE 关键字来允许更新完成，并将 Amazon Redshift 方言标记为过时。当标记为过时的引擎查询视图时，查询会失败。引擎会抛出异常以禁止过时结果。要更正此问题，请更新视图中的过时方言。

ADD

为 Data Catalog 视图添加新的引擎方言。指定的引擎不得已存在于 Data Catalog 视图中。

UPDATE

更新 Data Catalog 视图中已存在的引擎方言。

DROP

从 Data Catalog 视图中删除现有的引擎方言。从 Data Catalog 视图中删除引擎后，已删除的引擎无法查询 Data Catalog 视图。视图中的其他引擎方言仍然可以查询视图。

DIALECT AS

引入特定于引擎的 SQL 查询。

示例

```
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

在 AWS Glue Data Catalog 中创建 Data Catalog 视图。Data Catalog 视图是一种单一视图架构，可以在 Athena 和其他 SQL 引擎（例如 Amazon Redshift 和 Amazon EMR）之间无缝运行。

语法

```
CREATE [ OR REPLACE ] PROTECTED MULTI DIALECT VIEW view_name
 [ SECURITY DEFINER ]
AS query
```

PROTECTED

所需关键字。指定视图受到保护以防数据泄露。Data Catalog 视图只能作为 PROTECTED 视图创建。

MULTI DIALECT

指定视图支持不同查询引擎的 SQL 方言，因此可以由这些引擎读取。

SECURITY DEFINER

指定定义程序语义适用于此视图。定义程序语义意味着基础表的有效读取权限属于定义视图的主体或角色，而不是执行实际读取的主体。

OR REPLACE

如果 Data Catalog 视图中存在其他引擎的 SQL 方言，则无法替换该视图。如果调用引擎拥有视图中唯一的 SQL 方言，则可以替换视图。

示例

以下示例基于 `orders` 表查询创建 `orders_by_date` Data Catalog 视图。

```
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

显示指定 Data Catalog 视图的列的列表。`DESCRIBE` 语句与 Athena 视图的 `DESCRIBE` 语句类似。与 Athena 视图不同的是，该语句的输出通过 Lake Formation 访问控制进行控制。因此，此查询的输出不是视图的所有列，而只是调用者有权访问的列。

语法

```
DESCRIBE [db_name.]view_name
```

示例

```
DESCRIBE orders
```

DROP VIEW

仅当 Data Catalog 视图中存在调用引擎方言时，才删除 Data Catalog 视图。例如，如果用户从 Athena 调用 `DROP VIEW`，则仅当视图中存在 Athena 的方言时，视图才会被删除。否则，该操作将失败。只有 Lake Formation 管理员和视图定义者有权在 Data Catalog 视图上使用 `DROP VIEW` 语句。

语法

```
DROP VIEW [ IF EXISTS ] view_name
```

示例

```
DROP VIEW orders_by_date
```

```
DROP FORCE VIEW IF EXISTS orders_by_date
```

如果该视图不存在，可选 IF EXISTS 子句将抑制错误出现。

SHOW COLUMNS

仅显示单个指定 Data Catalog 视图的列名。SHOW COLUMNS 语句与 Athena 视图的 SHOW COLUMNS 语句类似。与 Athena 视图不同的是，该语句的输出通过 Lake Formation 访问控制进行控制。因此，此查询的输出不是视图的所有列，而只是调用者有权访问的列。

语法

```
SHOW COLUMNS {FROM|IN} database_name.view_name
```

```
SHOW COLUMNS {FROM|IN} view_name [{FROM|IN} database_name]
```

SHOW CREATE VIEW

显示用于创建 Data Catalog 视图的 SQL 语法。返回的 SQL 显示了 Athena 中使用的创建视图语法。只有 Lake Formation 管理员和视图定义者主体有权在 Data Catalog 视图上调用 SHOW CREATE VIEW。

语法

```
SHOW CREATE VIEW view_name
```

示例

```
SHOW CREATE VIEW orders_by_date
```

SHOW VIEWS

列出数据库中所有视图的名称。列出数据库中拥有 Athena 引擎 SQL 方言的所有 Data Catalog 视图。没有视图中存在的 Athena 引擎方言的其他 Data Catalog 视图会被过滤掉。

语法

```
SHOW VIEWS [IN database_name] [LIKE 'regular_expression']
```

示例

```
SHOW VIEWS
```

```
SHOW VIEWS IN marketing_analytics LIKE 'orders*'
```

使用已保存的查询

您可以使用 Athena 控制台保存、编辑、运行、重命名和删除您在查询编辑器中创建的查询。

注意事项和限制

- 您可以更新已保存的查询的名称、描述和查询文本。
- 您只能更新自己账户中的查询。
- 您无法更改查询所属的工作组或数据库。
- Athena 不会保存查询的修订历史记录。如果您需要保留特定版本的查询，请将其保存为其他名称。

在 Athena 控制台中使用已保存的查询

保存查询并为其命名

1. 在 Athena 控制台查询编辑器中输入或运行一个查询。
2. 在查询编辑器窗口上方的查询选项卡上，选择三个竖直点，然后选择 Save as（另存为）。
3. 在 Save query（保存查询）对话框中，输入查询的名称和可选的描述。您可以使用可展开的 Preview SQL query（预览 SQL 查询）窗口，以在保存查询之前验证查询的内容。
4. 选择 Save query（保存查询）。

   在查询编辑器中，该查询的选项卡将显示您指定的名称。

运行保存的查询

1. 在 Athena 控制台中，选择 Saved queries（已保存的查询）选项卡。
2. 在 Saved queries（已保存的查询）列表中，选择要运行的查询 ID。

   查询编辑器将会显示您选择的查询。
3. 选择运行。

编辑已保存的查询

1. 在 Athena 控制台中，选择 Saved queries（已保存的查询）选项卡。
2. 在 Saved queries（已保存的查询）列表中，选择要编辑的查询 ID。
3. 在查询编辑器中编辑查询。
4. 执行下列步骤之一：
   - 要运行查询，请选择 Run（运行）。
   - 要保存查询，请在查询选项卡上选择三个竖直点，然后选择 Save（保存）。
   - 要使用其他名称保存查询，请在查询选项卡上选择三个竖直点，然后选择 Save as（另存为）。

重命名或删除在查询编辑器中显示的已保存查询

1. 在查询选项卡上选择三个竖直点，然后选择 Rename（重命名）或 Delete（删除）。
2. 按照提示重命名或删除查询。

重命名未在查询编辑器中显示的已保存查询

1. 在 Athena 控制台中，选择 Saved queries（已保存的查询）选项卡。
2. 选择要重命名的查询对应的复选框。
3. 选择 Rename (重命名)。
4. 在 Rename query（重命名查询）对话框中，编辑查询名称和查询描述。您可以使用可展开的 Preview SQL query（预览 SQL 查询）窗口，以在重命名查询之前验证查询的内容。
5. 选择 Rename query（重命名查询）。

   重命名的查询将在 Saved queries（已保存的查询）列表中显示。

删除未在查询编辑器中显示的已保存查询

1. 在 Athena 控制台中，选择 Saved queries（已保存的查询）选项卡。
2. 选择要删除的一个或多个查询对应的复选框。
3. 选择 删除。
4. 出现确认提示时，选择 Delete（删除）。

   一个或多个查询将从 Saved queries（已保存的查询）列表中删除。

使用 Athena API 更新已保存的查询

有关使用 Athena API 更新已保存的查询的信息，请参阅《Athena API 参考》中的 UpdateNamedQuery 操作。

使用参数化查询

您可以使用 Athena 参数化查询在执行时使用不同的参数值重新运行同一个查询，从而帮助防止 SQL 注入攻击。在 Athena 中，参数化查询可以在任何 DML 查询或 SQL 预准备语句中采用执行参数的形式。

- 带有执行参数的查询可以在单个步骤中完成，而不是特定于工作组。您可以在任何 DML 查询中为要参数化的值放置问号。运行查询时，按顺序声明执行参数的值。参数的声明和参数的赋值可以在同一查询中完成，但单独进行。与预准备语句不同，您可以在提交带有执行参数的查询时选择工作组。
- 预准备语句需要两个单独的 SQL 语句：PREPARE 和 EXECUTE。首先，在 PREPARE 语句中定义参数。然后，运行为您定义的参数提供值的 EXECUTE 语句。预准备语句特定于工作组；您不能在预准备语句所属的工作组的上下文以外运行它们。

注意事项和限制

- 参数化查询仅在 Athena 引擎版本 2 及更高版本中受支持。有关 Athena 引擎版本的更多信息，请参阅 Athena 引擎版本控制。
- 目前，参数化查询仅在 SELECT、INSERT INTO、CTAS 和 UNLOAD 语句中受支持。
- 在参数化查询中，参数是位置参数，并用 ? 指示。参数按其在查询中的顺序赋值。命名参数不受支持。
- 目前，? 参数只能放在 WHERE 子句中。不支持 SELECT ? FROM table 之类的语法。
- 问号参数不能放在双引号或单引号以内（即 '?' 和 "?" 不是有效的语法）。
- 要将 SQL 执行参数视为字符串，它们必须用单引号而不是双引号括起来。
- 如有必要，您可以在为参数化术语输入值时使用 CAST 函数。例如，如果您有一列 date 类型在查询中参数化，并且想要查询日期 2014-07-05，则为参数值输入 CAST('2014-07-05' AS DATE) 将返回结果。
- 预准备语句特定于工作组，预准备语句名称在工作组中必须唯一。
- 预准备语句的 IAM 权限是必需的。有关更多信息，请参阅 允许访问预编译语句。
- 在 Athena 控制台中使用执行参数的查询最多只能有 25 个问号。

使用执行参数查询

您可以在任何 DML 查询中使用问号占位符创建参数化查询，而无需先创建预准备语句。要运行这些查询，您可以使用 Athena 控制台，或者使用 AWS CLI 或 AWS SDK，并在 `execution-parameters` 参数中声明变量。

在 Athena 控制台中运行具有执行参数的查询

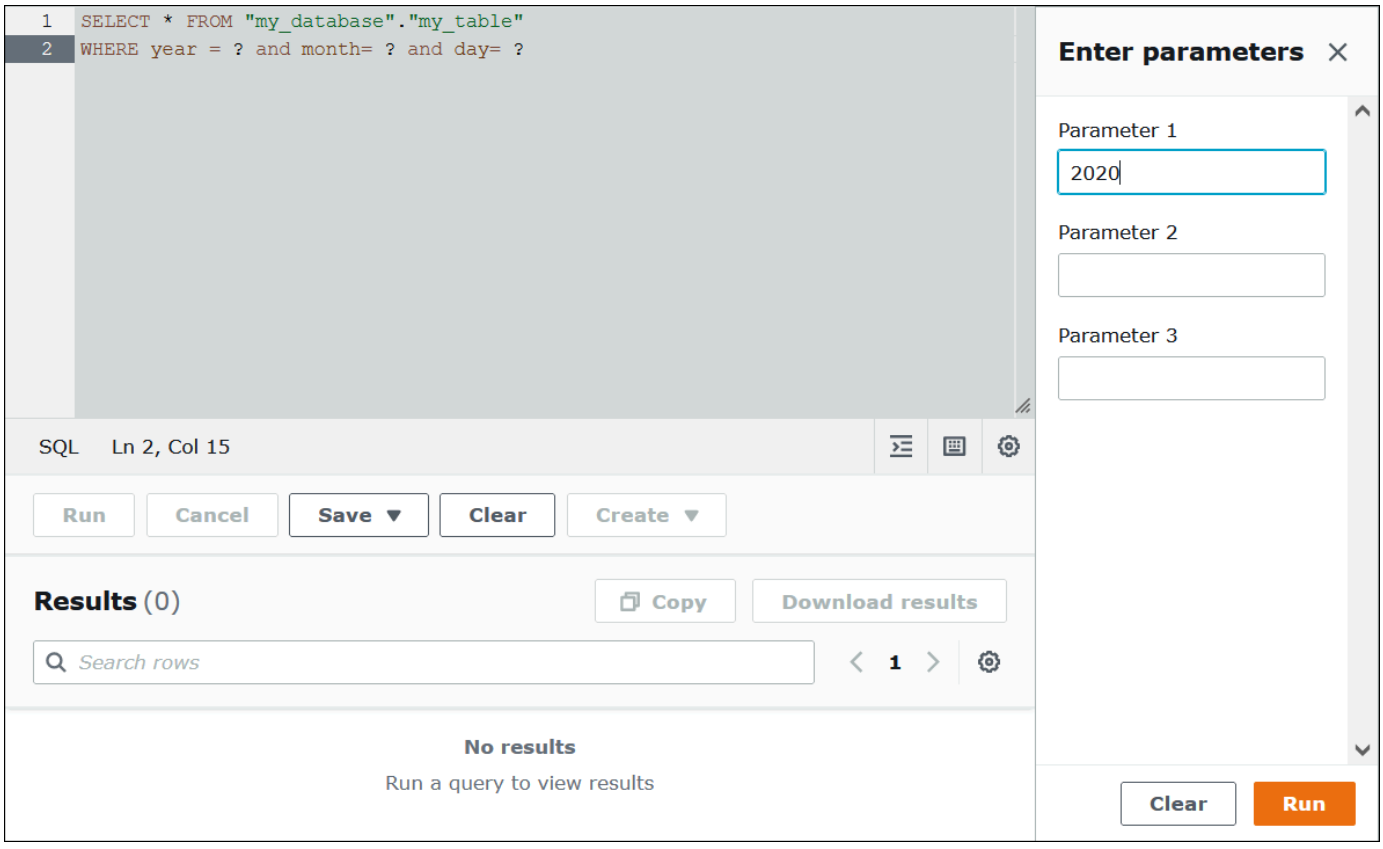
在 Athena 控制台中运行具有执行参数（问号）的参数化查询时，系统会按照问号在查询中出现的顺序提示您输入值。

要运行具有执行参数的查询

1. 请在 Athena 编辑器中输入带有问号占位符的查询，如以下示例所示。

```
SELECT * FROM "my_database"."my_table"
WHERE year = ? and month= ? and day= ?
```

2. 选择运行。
3. 在 Enter parameters（输入参数）对话框中，按查询中每个问号的顺序输入值。



4. 输入完参数后，选择 Run（运行）。编辑器会显示您输入的参数值的查询结果。

在这种情况下，您可以执行下列操作之一：

- 为同一个查询输入不同的参数值，然后再次选择 Run（运行）。
- 要立即清除输入的所有值，请选择 Clear（清除）。
- 要直接编辑查询（例如，添加或删除问号），请关闭 Enter parameters（输入参数）对话框。
- 要保存参数化查询以供后续使用，请选择 Save（保存）或者 Save as（另存为），然后为查询命名。有关使用已保存查询的更多信息，请参阅 使用已保存的查询。

为方便起见，Enter parameters（输入参数）对话框会记住先前为查询输入的值，只要您在查询编辑器中使用同一选项卡。

使用 AWS CLI 运行具有执行参数的查询

要使用 AWS CLI 运行具有执行参数的查询，请使用 `start-query-execution` 命令并在 `query-string` 参数中提供参数化查询。然后，在 `execution-parameters` 参数中，提供执行参数的值。以下示例对此方法进行了说明。

```
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"
```

使用预准备语句进行查询

您可以使用预编译语句重复执行具有不同查询参数的同一查询。预准备语句包含参数占位符，其值在执行时提供。

> **Note**
>
> 工作组中的最大预处理语句数为 1000。

SQL 语句

您可以使用 `PREPARE`、`EXECUTE` 和 `DEALLOCATE PREPARE` SQL 语句在 Athena 控制台查询编辑器中运行参数化查询。

- 要指定通常会使用文字值的参数，请在 PREPARE 语句中使用问号。
- 若要在运行查询时将参数替换为值，请在 EXECUTE 语句中使用 USING 子句。
- 要从工作组中的预准备语句中删除预准备语句，请使用 DEALLOCATE PREPARE 语句。

以下各部分提供了有关这些语句的其他详细信息。

PREPARE

准备要在之后运行的语句。预准备语句将以您指定的名称保存在当前工作组中。语句可以包含参数来代替要在查询运行时替换的文字。要替换为值的参数用问号表示。

语法

```
PREPARE statement_name FROM statement
```

下表介绍了这些参数。

参数	描述
statement_name	要执行的预准备语句的名称。此名称在工作组范围内必须唯一。
##	SELECT、CTAS 或 INSERT INTO 查询。

PREPARE 示例

以下示例将演示如何使用 PREPARE 语句。问号表示在运行查询时由 EXECUTE 语句提供的值。

```
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

运行预准备语句。参数的值在 USING 子句中指定。

语法

```
EXECUTE statement_name [USING value1 [ ,value2, ... ] ]
```

statement_name 是预准备语句的名称。*value1* 和 *value2* 是要为语句中的参数指定的值。

EXECUTE 示例

以下示例将运行 my_select1 预准备语句，该语句不包含任何参数。

```
EXECUTE my_select1
```

以下示例将运行 my_select2 预准备语句，其中包含一个参数。

```
EXECUTE my_select2 USING 2012
```

以下示例将运行 my_select3 预准备语句，它使用两个参数。

```
EXECUTE my_select3 USING 346078, 12
```

以下示例为预准备语句 my_insert 中的参数提供字符串值。

```
EXECUTE my_insert USING 'usa'
```

以下示例为预准备语句 my_unload 中的 productid 参数提供数字值。

```
EXECUTE my_unload USING 12
```

DEALLOCATE PREPARE

从当前工作组中的预准备语句列表中删除具有指定名称的预准备语句。

语法

```
DEALLOCATE PREPARE statement_name
```

statement_name 是要删除的预准备语句的名称。

示例

以下示例从当前工作组中删除 my_select1 预准备语句。

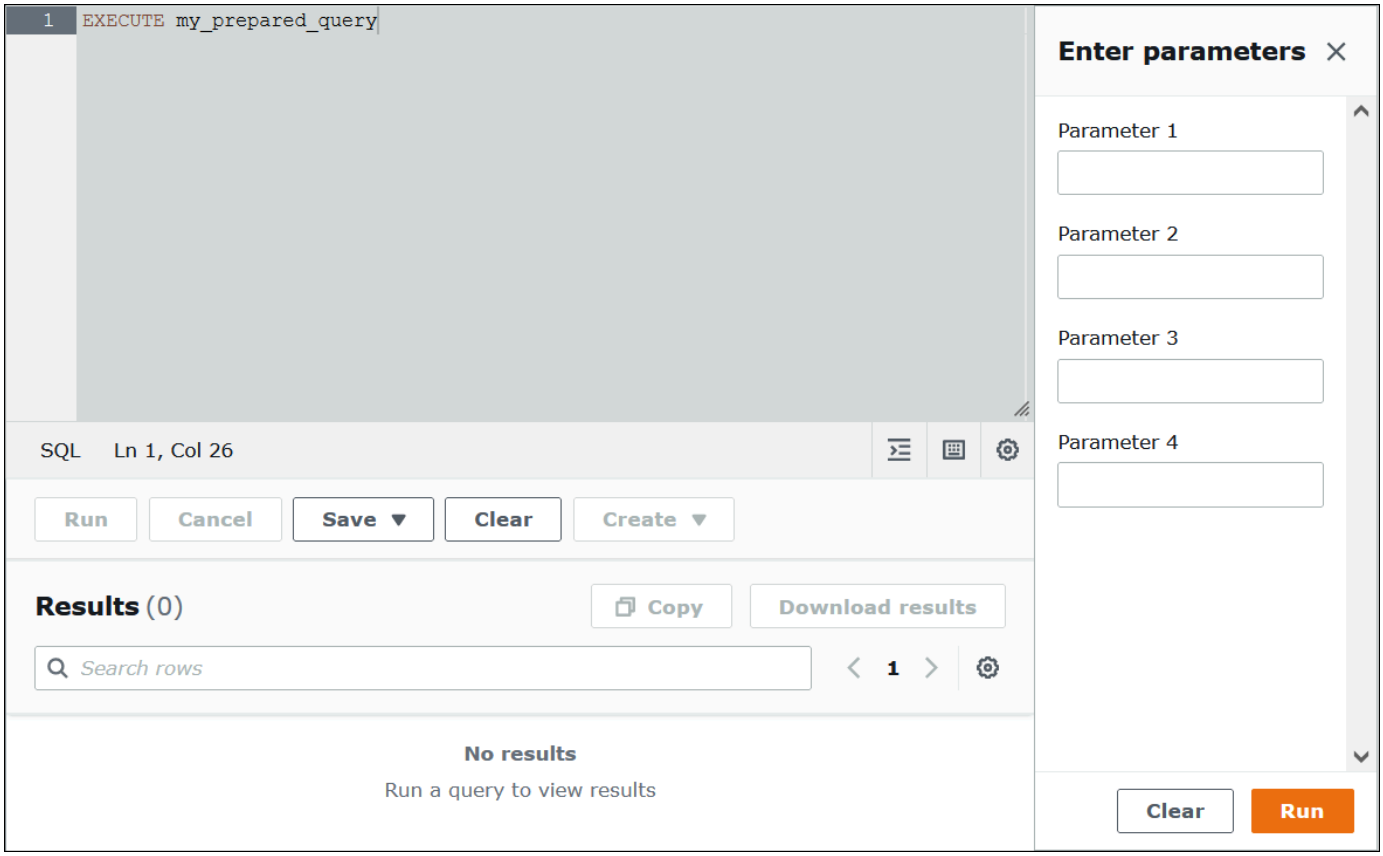
```
DEALLOCATE PREPARE my_select1
```

在 Athena 控制台中执行不带 USING 子句的预准备语句

如果在查询编辑器中使用语法 EXECUTE *prepared_stament* 运行现有预准备语句，Athena 会打开 Enter parameters（输入参数）对话框，这样您就可以输入通常出现在 EXECUTE ... USING 语句的 USING 子句中的值。

使用 Enter parameters（输入参数）对话框运行预准备语句

1. 在查询编辑器中，使用语法 EXECUTE *prepared_stament* 而不是 EXECUTE prepared_statement USING *Value1* , *Value2* ...。
2. 选择运行。将显示 Enter parameters（输入参数）对话框。



3. 按照 Execution parameters（执行参数）对话框中的顺序输入值。由于查询的原始文本不可见，因此必须记住每个位置参数的含义或使预准备语句可供参考。
4. 选择运行。

使用 AWS CLI 创建预准备语句

要使用 AWS CLI 创建预准备语句，您可以使用以下 `athena` 命令中的一个：

- 使用 `create-prepared-statement` 命令并提供包含执行参数的查询语句。
- 使用 `start-query-execution` 命令并提供使用 `PREPARE` 语法的查询字符串。

使用 create-prepared-statement

在 `create-prepared-statement` 命令中，定义 `query-statement` 参数中的查询文本，如以下示例所示。

```
aws athena create-prepared-statement
--statement-name PreparedStatement1
--query-statement "SELECT * FROM table WHERE x = ?"
```

```
--work-group athena-engine-v2
```

使用 start-query-execution 和 PREPARE 语法

使用 `start-query-execution` 命令。在 `query-string` 参数中放置 PREPARE 语句，如以下示例所示：

```
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/..."}'
```

使用 AWS CLI 执行预准备语句

要使用 AWS CLI 执行预准备语句，您可以使用以下方法之一为参数提供值：

- 使用 `execution-parameters` 参数。
- 使用 `query-string` 参数中的 `EXECUTE ... USING` SQL 语法。

使用 execution-parameters 参数

在此方法中，您可以使用 `start-query-execution` 命令并提供 `query-string` 参数现有预准备语句的名称。然后，在 `execution-parameters` 参数中，提供执行参数的值。以下示例说明了这一方法：

```
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"
```

使用 EXECUTE... 使用 SQL 语法

要使用 `EXECUTE ... USING` 语法运行现有的预准备语句，您可以使用 `start-query-execution` 命令并将预准备语句的名称和参数值都放在 `query-string` 参数中，如以下示例所示：

```
aws athena start-query-execution
--query-string "EXECUTE PreparedStatement1 USING 1"
--query-execution-context '{"Database": "default"}'
--result-configuration '{"OutputLocation": "s3://DOC-EXAMPLE-BUCKET/..."}'
```

列出预准备语句列表

要列出特定工作组的预准备语句，可以使用 Athena list-prepared-statements AWS CLI 命令或 ListPreparedStatements Athena API 操作。`--work-group` 参数是必需的。

```
aws athena list-prepared-statements --work-group primary
```

其他 资源

请参阅 AWS 大数据博客中的以下相关文章。

- 使用 Amazon Athena 参数化查询提高可重用性和安全性
- 使用 Amazon Athena 参数化查询将数据作为服务提供

使用成本型优化器

您可以使用 Athena SQL 中的成本型优化器（CBO）功能来优化查询。您可以选择请求 Athena 为 AWS Glue 中的一个表收集表级或列级统计数据。如果查询中的所有表都有统计数据，Athena 会使用这些统计数据来创建它认为性能最佳的执行计划。查询优化器会根据统计模型来计算备选计划，然后选择速度可能最快的计划来运行查询。

AWS Glue 表上的统计数据被收集并存储在 AWS Glue Data Catalog 中，也提供给 Athena 来改进查询计划和执行。这些统计数据是列级统计数据，例如 Parquet、ORC、JSON、ION、CSV 和 XML 等文件类型的不同值、空值、最大值和最小值的个数。Amazon Athena 通过在查询处理中尽早应用最严格的筛选条件来使用这些统计数据优化查询。这种筛选功能限制了内存使用量和为提供查询结果而必须读取的记录数。

Athena 还会将 CBO 和规则型优化器（RBO）功能结合起来使用。RBO 以机械方式应用可提高查询性能的规则。RBO 通常会带来益处，因为它的转换是为了简化查询计划。不过，由于 RBO 不进行成本计算或计划比较，查询较复杂便会让 RBO 难以创建最佳计划。

因此，Athena 会同时使用 RBO 和 CBO 来优化您的查询。在确定改进查询执行的机会后，Athena 就会创建最佳计划。有关执行计划详情的信息，请参阅查看 SQL 查询的执行计划。有关 CBO 工作原理的详细讨论，请参阅 AWS 大数据成本型优化器博客文章。

要为 AWS Glue Catalog 表生成统计数据，可以使用 Athena 控制台、AWS Glue 控制台或 AWS Glue API。由于 Athena 已与 AWS Glue Catalog 集成，在运行来自 Amazon Athena 的查询时，您会自动获得相应的查询性能改进。

注意事项和限制

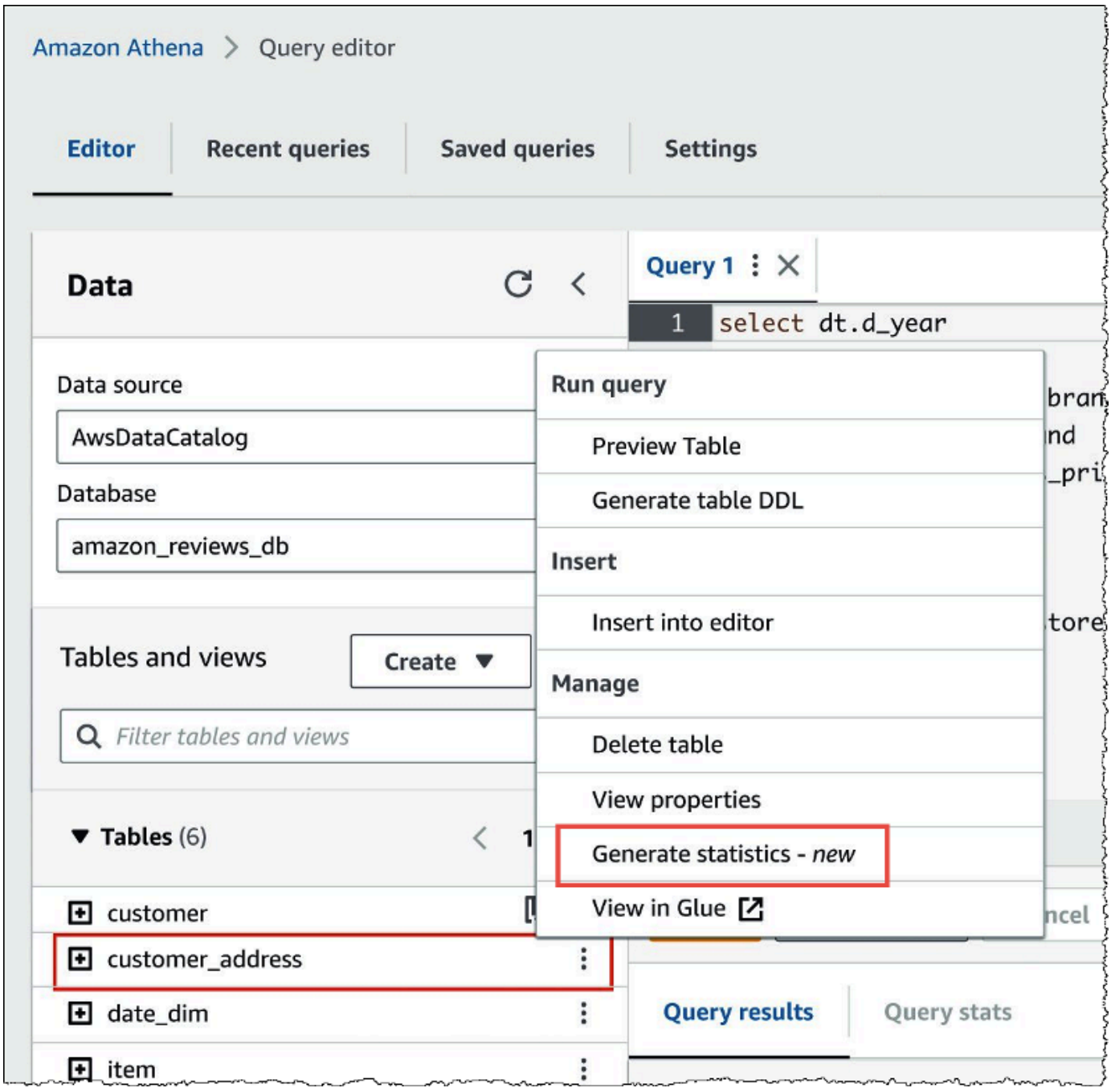
- 表类型 – 目前，Athena 中的 CBO 功能仅支持 AWS Glue Data Catalog 中的 Hive 表。
- Athena for Spark – CBO 功能在 Athena for Spark 中不可用。
- 定价 – 有关定价信息，请访问 AWS Glue 定价页面。

使用 Athena 控制台生成表统计数据

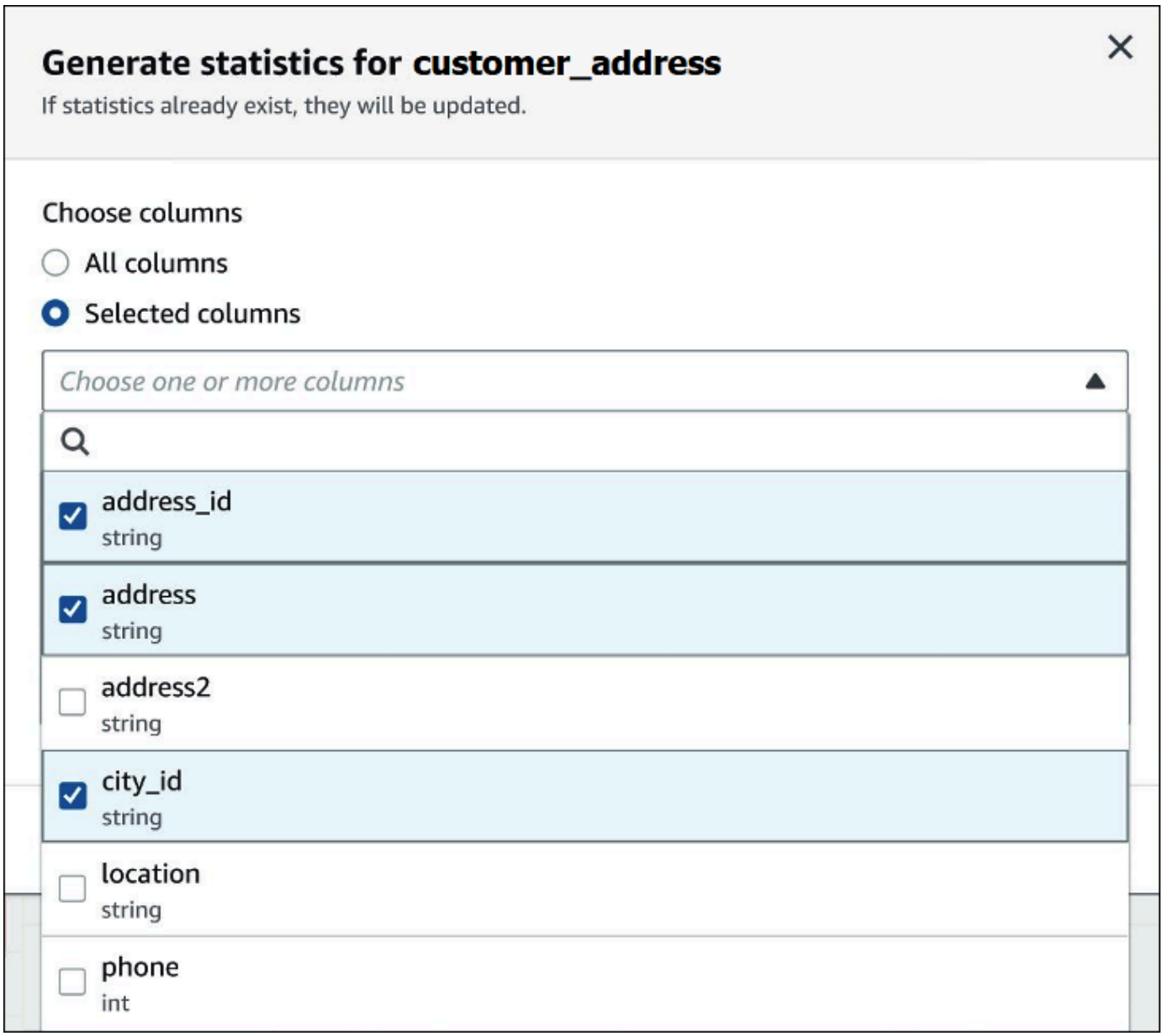
本节旨在介绍如何使用 Athena 控制台为 AWS Glue 中的表生成表级或列级统计数据。有关使用 AWS Glue 生成表统计数据的信息，请参阅《AWS Glue Developer Guide》中的 Working with column statistics。

使用 Athena 控制台为表生成统计数据

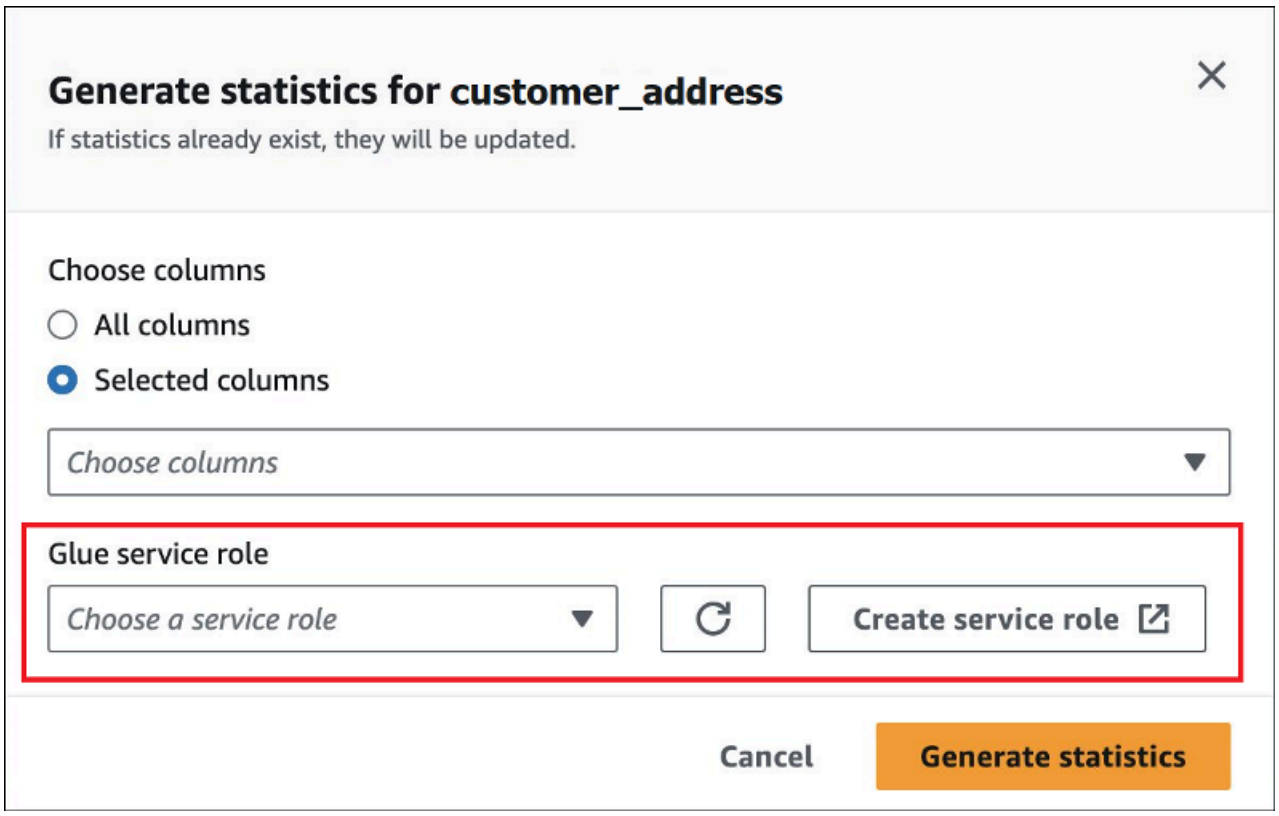
1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 在 Athena 查询编辑器的表列表中，选择所需表的三个垂直点，然后选择生成统计数据。



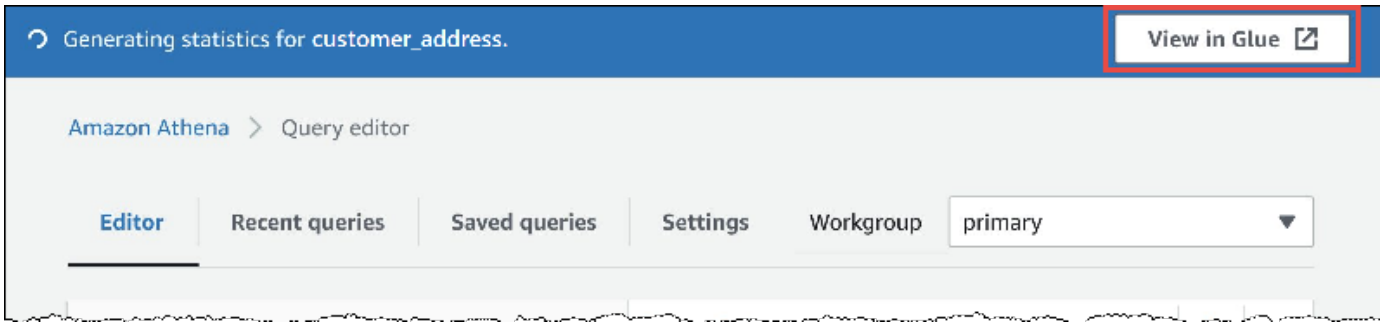
3. 在生成统计数据对话框中，选择所有列为表中的所有列生成统计数据，或者选择所选列来选择特定列。所有列为默认设置。



4. 对于 AWS Glue 服务角色，创建服务角色或选择现有服务角色来授予 AWS Glue 生成统计数据的权限。对于包含表数据的 Amazon S3 存储桶，AWS Glue 服务角色还需要 `S3:GetObject` 权限。



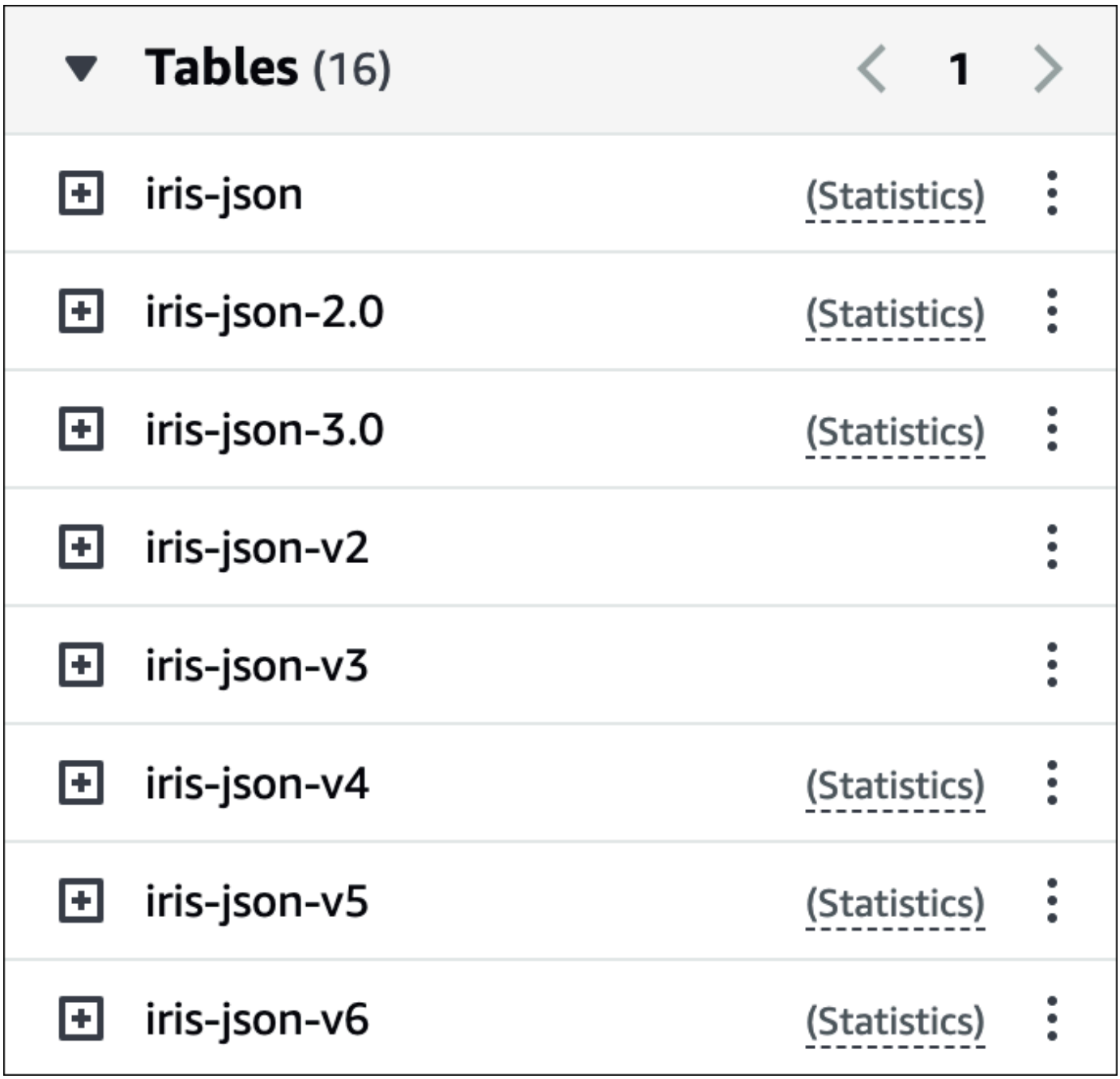
5. 选择生成统计数据。为 ***table_name*** 生成统计数据通知横幅显示任务状态。



6. 要在 AWS Glue 控制台中查看详细信息，请选择在 Glue 中查看。

   有关在 AWS Glue 控制台中查看统计数据的信息，请参阅《AWS Glue 开发人员指南》中的 Viewing column statistics。

7. 生成统计数据后，包含统计数据的表和列在括号中显示统计数据一词，如下图所示。



现在，如果运行查询，Athena 会对生成统计数据的表和列执行基于成本的优化。

其他 资源

有关更多信息，请参阅以下资源。

查询 S3 Express One Zone 数据

Amazon S3 Express One Zone 存储类是一种可提供个位数毫秒级响应时间的高性能 Amazon S3 存储类。因此，它非常适合以每秒数十万个请求频繁访问数据的应用程序。

S3 Express One Zone 在同一个可用区内复制和存储数据，优化了速度和成本。这与 Amazon S3 区域存储类不同，后者会自动在一个 AWS 区域 内的至少三个 AWS 可用区之间复制数据。

有关更多信息，请参阅《Amazon S3 用户指南》中的 What is S3 Express One Zone?。

先决条件

在开始使用之前，请确认满足以下条件：

- Athena 引擎版本 3 – 要将 S3 Express One Zone 与 Athena SQL 结合使用，必须将您的工作组配置为使用 Athena 引擎版本 3。
- S3 Express One Zone 权限 – 当 S3 Express One Zone 在 Amazon S3 对象上调用 `GET`、`LIST` 或 `PUT` 等操作时，存储类会代表您调用 `CreateSession`。因此，您的 IAM policy 必须允许 `s3express:CreateSession` 操作，这样才能允许 Athena 调用相应的 API 操作。

注意事项和限制

使用 Athena 查询 S3 Express One Zone 时，请注意以下几点。

- S3 Express One Zone 桶仅支持 `SSE_S3` 加密。无论您在工作组设置中选择哪个选项来加密查询结果，Athena 查询结果都将使用 `SSE_S3` 加密写入。此限制包括 Athena 向 S3 Express One Zone 桶写入数据的所有场景，包括 `CREATE TABLE AS`（CTAS）和 `INSERT INTO` 语句。
- 不支持使用 AWS Glue 爬网程序在 S3 Express One Zone 数据上创建表。
- 不支持 `MSCK REPAIR TABLE` 语句。一个解决方法是使用 ALTER TABLE ADD PARTITION。
- 不支持以下文件和表格格式或支持受限。如果格式未列出，但受 Athena 支持（例如 Parquet、ORC 和 JSON），则 S3 Express One Zone 存储也支持使用这些格式。

文件或表格格式	限制
Apache Avro	不支持
CloudTrail 日志	不支持

文件或表格格式	限制
Apache Hudi	不支持
Amazon Ion	不支持
Logstash 日志	不支持
Apache WebServer 日志	不支持
Delta Lake	不支持 DDL。有关使用虚拟架构创建 Delta Lake 表的信息，请参阅同步 Delta Lake 元数据。支持对表进行 SELECT 查询。

开始使用

使用 Athena 查询 S3 Express One Zone 数据非常简单。要开始使用，请按照以下过程操作。

使用 Athena SQL 查询 S3 Express One Zone 数据

1. 将数据转移到 S3 Express One Zone 存储。有关更多信息，请参阅《Amazon S3 用户指南》中的设置对象的存储类。
2. 在 Athena 中使用 CREATE TABLE 语句将数据编入 AWS Glue Data Catalog 目录。有关在 Athena 中创建表的更多信息，请参阅 在 Athena 中创建表 和 CREATE TABLE 语句。
3. （可选）将 Athena 工作组的查询结果位置配置为使用 Amazon S3 目录桶。与普通桶相比，Amazon S3 目录桶的性能更高，专为需要一致的个位数毫秒延迟的工作负载或性能至关重要的应用程序而设计。有关更多信息，请参阅《Amazon S3 用户指南》中的 Directory buckets overview。

查询还原的 Amazon S3 Glacier 对象

您可以使用 Athena 查询从 S3 Glacier Flexible Retrieval（以前称为 Glacier）和 S3 Glacier Deep Archive Amazon S3 存储类还原的对象。必须针对每个表启用此功能。如果您在运行查询之前未在表上启用该功能，Athena 将在查询执行期间跳过该表的所有 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 对象。

注意事项和限制

- 只有 Athena 引擎版本 3 支持查询还原的 Amazon S3 Glacier 对象。
- 只有 Apache Hive 表支持该功能。
- 在查询数据之前，您必须还原对象；Athena 不会为您还原对象。

将表配置为使用已还原的对象

要将 Athena 表配置为在查询中包含还原的对象，必须将其 `read_restored_glacier_objects` 表属性设置为 `true`。为此，您可以使用 Athena 查询编辑器或 AWS Glue 控制台。您还可以使用 AWS Glue CLI、AWS Glue API 或 AWS Glue SDK。

使用 Athena 查询编辑器

在 Athena 中，您可以使用 ALTER TABLE SET TBLPROPERTIES 命令来设置表属性，如下例所示。

```
ALTER TABLE table_name SET TBLPROPERTIES ('read_restored_glacier_objects' = 'true')
```

使用 AWS Glue控制台

在 AWS Glue 控制台中，执行以下步骤添加 `read_restored_glacier_objects` 表属性。

在 AWS Glue 控制台中配置表属性

1. 登录 AWS Management Console，然后打开 AWS Glue 控制台，网址为：https://console.aws.amazon.com/glue/。
2. 请执行以下操作之一：
   - 选择转到数据目录。
   - 在导航窗格中，选择数据目录表。
3. 在表页面的表列表中+，选择要编辑的表的链接。
4. 依次选择 Actions（操作）、Edit table（编辑表）。
5. 在编辑表页面的表属性部分中，添加以下键值对：
   - 对于 Key (键)，添加 `read_restored_glacier_objects`。
   - 对于 Value（值），输入 `true`。
6. 请选择保存。

使用 AWS CLI

在 AWS CLI 中，您可以使用 AWS Glue update-table 命令及其 --table-input 参数来重新定义表，并在此过程中添加 read_restored_glacier_objects 属性。在 --table-input 参数中，使用 Parameters 结构来指定 read_restored_glacier_objects 属性和 true 值。请注意：--table-input 的参数不能有空格，并且必须使用反斜杠来转义双引号。在以下示例中，将 *my_database* 和 *my_table* 替换为数据库和表的名称。

```
aws glue update-table \
   --database-name my_database \
   --table-input={\"Name\":\"my_table\",\"Parameters\":{\"read_restored_glacier_objects
\":\"true\"}}
```

> **Important**
>
> AWS Glue update-table 命令在覆盖模式下运行，这意味着其将用 table-input 参数指定的新定义替换现有的表定义。因此，在添加 read_restored_glacier_objects 属性时，请务必在 table-input 参数中指定希望表中包含的所有字段。

处理架构更新

本节提供了有关为各种数据格式处理架构更新的指导。Athena 是一种基于读取的查询引擎。这意味着，当您在 Athena 中创建一个表时，它会在读取数据时应用架构。它不会改变或者重写底层数据。

如果您期望表架构有所变化，则在创建表架构时应考虑采用一种适合您的需求的数据格式。您的目标是对于不断演进的架构仍能再利用现有的 Athena 查询，同时避免在查询包含分区的表时发生架构不匹配错误。

为实现这些目标，请基于下列主题中的表选择表的数据格式。

主题

- 摘要：Athena 中的更新和数据格式
- ORC 和 Parquet 中的索引访问
- 更新类型
- 包含分区的表中的更新

摘要：Athena 中的更新和数据格式

下表总结了数据存储格式及其支持的架构处理方式。使用此表有助于您选择格式，即使架构随时间发生了变化，也可继续使用 Athena 查询。

在此表中，您可以观察到 Parquet 和 ORC 的列式格式具有不同的默认列访问方法。预设情况下，Parquet 将按名称访问列，ORC 按索引（序数值）访问列。因此，Athena 可在创建表时定义一个 SerDe 属性，以切换默认列访问方法，从而在架构演进的过程中实现更大的灵活度。

对于 Parquet，可将 `parquet.column.index.access` 属性设为 `true`，这样可将列访问方法设置为使用列的序号。将此属性设为 `false` 会将列访问方法改为使用列名称。同样，ORC 使用 `orc.column.index.access` 属性控制列访问方法。有关更多信息，请参阅 ORC 和 Parquet 中的索引访问。

除了将列重新排序或在表的开头添加列之外，可通过 CSV 和 TSV 进行所有其他架构操作。例如，如果架构演进只需要将列重新命名而不需要删除它们，您可以选择创建 CSV 或 TSV 格式的表。如果您需要删除列，请不要使用 CSV 或 TSV，而要使用任何其他支持的格式，最好是列式格式（例如 Parquet 或 ORC）。

Athena 中的架构更新和数据格式

架构更新的预期类型	Summary	CSV（带或不带标头）和 TSV	JSO	AVF	PARQUE 按名称读取（默认）	PARQUE 按索引读取	ORC：按索引读取（默认）	ORC：按名称读取
重命名列	将您的数据存储为 CSV 和 TSV；或存储为 ORC 和 Parquet（如果按索引读取）。	Y	否	否	否	Y	Y	否
在表的开头或中间添加新列	将您的数据存储为 JSON、AVRO；或存储为 Parquet 和 ORC（如果按名称读取）。不	否	Y	Y	Y	否	否	Y

架构更新的预期类型	Summary	CSV（带或不带标头）和 TSV	JSO	AVF	PARQUE 按名称读取（默认）	PARQUE 按索引读取	ORC：按索引读取（默认）	ORC：按名称读取
	要使用 CSV 和 TSV。							
在表的末尾添加列	将您的数据以 CSV 或 TSV、JSON、AVRO、ORC 或 Parquet 格式存储。	Y	Y	Y	Y	Y	Y	Y
删除列	将您的数据存储为 JSON、AVRO；或存储为 Parquet 和 ORC（如果按名称读取）。不要使用 CSV 和 TSV。	否	Y	Y	Y	否	否	Y
对列重新排序	将您的数据存储为 AVRO、JSON；或存储为 ORC 和 Parquet（如果按名称读取）。	否	Y	Y	Y	否	否	Y

架构更新的预期类型	Summary	CSV（带或不带标头）和 TSV	JSO	AVF	PARQUE 按名称读取（默认）	PARQUE 按索引读取	ORC：按索引读取（默认）	ORC：按名称读取
更改列的数据类型	将您的数据存储为任何格式，但请在 Athena 中测试您的查询，以确保数据类型兼容。对于 Parquet 和 ORC，更改数据类型仅适用于分区表。	Y	Y	Y	Y	Y	Y	Y

ORC 和 Parquet 中的索引访问

PARQUET 和 ORC 是列式数据存储格式，可以按索引或名称读取。将数据存储为这两种格式之一，可在执行所有架构操作和运行 Athena 查询时确保不会产生架构不匹配的错误。

- Athena 预设情况下按索引读取 ORC，如 `SERDEPROPERTIES ( 'orc.column.index.access'='true')` 中所定义。有关更多信息，请参阅 ORC：按索引读取。
- Athena 预设情况下按名称读取 Parquet，在 `SERDEPROPERTIES ( 'parquet.column.index.access'='false')` 中定义。有关更多信息，请参阅 Parquet：按名称读取。

这些是默认设置，因此在您的 `CREATE TABLE` 查询中指定这些 SerDe 属性是可选操作，它们是隐式使用的。如果使用这些属性，则允许您运行一些架构更新操作，同时防止其他类似操作。要支持这些操作，请运行另一 `CREATE TABLE` 查询并更改 SerDe 设置。

> **Note**
>
> 这些 SerDe 属性不会自动传播到每个分区。使用 `ALTER TABLE ADD PARTITION` 语句为每个分区设置 SerDe 属性。要自动执行该流程，请编写一个运行 `ALTER TABLE ADD PARTITION` 语句的脚本。

以下各部分详细描述了这些情况。

ORC：按索引读取

预设情况下，ORC 格式的表是按索引读取的。这是由以下语法定义的：

```
WITH SERDEPROPERTIES (
  'orc.column.index.access'='true')
```

按索引读取允许您将列重新命名。但这样的设置无法删除列或在表的中间添加列。

如果按名称读取 ORC，您就可以在 ORC 表的中间添加列，或删除列，请在 `CREATE TABLE` 语句中将 SerDe 属性 `orc.column.index.access` 设为 `false`。如使用此配置，将无法将列重新命名。

> **Note**
>
> 在 Athena 引擎版本 2 中，当 ORC 表设置为按名称读取时，Athena 要求 ORC 文件中的所有列名称均为小写。由于 Apache Spark 在生成 ORC 文件时不会使用小写字段名称，因此 Athena 可能无法读取如此生成的数据。解决方法是使用小写对列重命名，或使用 Athena 引擎版本 3。

以下示例演示了如何将 ORC 更改为按名称读取：

```
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：按名称读取

预设情况下，Parquet 格式的表是按名称读取的。这是由以下语法定义的：

```
WITH SERDEPROPERTIES (
  'parquet.column.index.access'='false')
```

按名称读取允许您在表中间添加列或删除列。但该设置无法将列重新命名。

要将 Parquet 设为按索引读取，使您可以将列重新命名，则必须在创建表时将 `parquet.column.index.access` SerDe 属性设为 `true`。

更新类型

本主题介绍了无需实际更改数据即可在 `CREATE TABLE` 语句中对架构进行的一些更改。我们将介绍每种架构更新类型，并指定哪些数据格式允许在 Athena 中实现这些更新。要更新架构，在某些情况下可以使用 `ALTER TABLE` 命令；而在其他情况下，实际上并不需要修改现有表。而是使用新名称创建一个表，该表修改了在原始 `CREATE TABLE` 语句中使用的架构。

- 将列添加到表的开头或中间
- 在表的末尾添加列
- 删除列
- 重命名列
- 重新排序列
- 更改列的数据类型

根据您期望架构的演进方式，选择一种兼容的数据格式，以继续使用 Athena 查询。

考察一个应用程序，该应用程序从 orders 表中读取订单信息，而该表存在两种格式：CSV 和 Parquet。

以下示例用 Parquet 格式创建一个表：

```
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/';
```

以下示例用 CSV 格式创建同样的表：

```
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/';
```

在以下部分中，我们将介绍这些表的更新如何影响 Athena 查询。

将列添加到表的开头或中间

添加列是最常见的架构变化之一。例如，您可以添加新列，用新的数据来充实表。或者，如果一个现有列的源发生了变化，您可以添加一个新列，同时保留该列表的前一版本，以调整依赖它们的应用程序。

要将列添加到表的开头或中间，并继续针对现有表运行查询，请使用 AVRO、JSON 以及 Parquet 和 ORC（如果它们的 SerDe 属性设为按名称读取）。有关信息，请参阅ORC 和 Parquet 中的索引访问。

不要在 CSV 和 TSV 表的开头或中间添加列，因为这些格式取决于排序。如果在这些情况下添加列，分区架构改变将导致架构不匹配的错误。

以下示例创建了一个新表，该表基于 JSON 数据在表的中间添加了一个 o_comment 列。

```
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/';
```

在表的末尾添加列

如果您使用 Athena 支持的任意格式（例如 Parquet、ORC、Avro、JSON、CSV 和 TSV）创建表，则可使用 ALTER TABLE ADD COLUMNS 语句在现有列之后但在分区列之前添加列。

以下示例在 orders_parquet 表末尾的任何分区列之前添加一个 comment 列：

```
ALTER TABLE orders_parquet ADD COLUMNS (comment string)
```

> **Note**
>
> 要在运行 ALTER TABLE ADD COLUMNS 后在 Athena 查询编辑器中查看新的表列，请手动刷新编辑器中的表列表，然后重新展开表。

删除列

如果表中的列不再包含数据，您可能需要删除它们，或者，您可能需要限制对于列数据的访问。

- 您可以从 JSON、Avro 表，以及按名称读取的 Parquet 和 ORC 表中删除列。有关信息，请参阅ORC 和 Parquet 中的索引访问。
- 如果您希望保留已在 Athena 中创建的表，我们不建议从 CSV 和 TSV 表中删除列。删除列会破坏架构，需要您重新创建不包含已删除列的表。

在本示例中，将从 Parquet 表中删除一列`totalprice`并运行查询。在 Athena 中，Parquet 默认是按名称读取的，因此我们省略了指定按名称读取的 SERDEPROPERTIES 配置。请注意，即使更改了架构，以下查询也会成功：

```
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/';
```

重命名列

有时，您可能需要重命名表列，以便纠正拼写、让列名称含义更清晰或重用现有列以避免列重新排序。

如果将数据存储为 CSV 和 TSV，或 Parquet 和 ORC（配置为按索引读取），则可以将列重新命名。有关信息，请参阅ORC 和 Parquet 中的索引访问。

Athena 会按照架构中的列顺序读取 CSV 和 TSV 数据并以同样的顺序返回。它不会使用列名称来将数据映射到列，因此，如果不中断 Athena 查询，您就可以重命名 CSV 或 TSV 格式列。

重命名列的一种策略是基于相同的基础数据创建新表，但使用新的列名。以下示例会创建一个名为 `orders_parquet_column_renamed` 新 `orders_parquet` 表。该示例将列 `o_totalprice` 名称更改为 `o_total_price`，然后在 Athena 中运行查询：

```
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/';
```

对于 Parquet 表而言，以下查询可以运行，但经过重命名的列不会显示数据，因为列是按名称访问的（Parquet 的默认设置）而不是按索引访问的：

```
SELECT *
FROM orders_parquet_column_renamed;
```

对于 CSV 表的查询看起来类似：

```
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/';
```

对于 CSV 表而言，以下查询可以运行，并将显示所有列的数据，包括经过重新命名的列：

```
SELECT *
FROM orders_csv_column_renamed;
```

重新排序列

只有表的数据格式是按名称读取的，才可以将列重新排序，例如 JSON 或默认为按名称读取的 Parquet。如有需要，也可以将 ORC 设为按名称读取。有关信息，请参阅ORC 和 Parquet 中的索引访问。

以下示例创建了一个新表，其中的列顺序不同：

```
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/';
```

更改列的数据类型

当现有类型无法再容纳所需的信息量时，您可能需要使用其他列类型。例如，ID 列的值可能超过 INT 数据类型的大小，需要使用 BIGINT 数据类型。

在计划为列使用不同的数据类型时，请考虑以下几点：

- 在大多数情况下，不能直接更改列的数据类型。而是可以重新创建 Athena 表，并使用新的数据类型定义该列。
- 只有特定数据类型可以读取为其他数据类型。有关可以进行这种处理的数据类型，请参阅本部分中的表。
- 对于 Parquet 和 ORC 格式的数据，如果表未进行分区，则将无法为列使用其他数据类型。
- 对于 Parquet 和 ORC 格式的分区表，一个分区的列类型可以不同于另一个分区的列类型，而且 Athena 将 CAST 为所需的类型（如果可能）。有关信息，请参阅避免带有分区的表出现架构不匹配错误。
- 对于仅使用 LazySimpleSerDe 创建的表，可以使用 ALTER TABLE REPLACE COLUMNS 语句将现有列替换为不同的数据类型，但是还必须在语句中重新定义要保留的所有现有列，否则它们将被删除。有关更多信息，请参阅 ALTER TABLE REPLACE COLUMNS。
- 仅对于 Apache Iceberg 表而言，您可以使用 ALTER TABLE CHANGE COLUMN 语句更改列的数据类型。ALTER TABLE REPLACE COLUMNS 不支持 Iceberg 表。有关更多信息，请参阅 不断变化的 Iceberg 表架构。

> **Important**
>
> 我们强烈建议您在执行数据类型转换时，先测试和验证一下您的查询。如果 Athena 无法使用目标数据类型，则 CREATE TABLE 查询可能会失败。

下表列出了被视为其他数据类型的数据类型：

兼容的数据类型

原始数据类型	可用的目标数据类型
STRING	BYTE, TINYINT, SMALLINT, INT, BIGINT
BYTE	TINYINT, SMALLINT, INT, BIGINT
TINYINT	SMALLINT, INT, BIGINT
SMALLINT	INT, BIGINT
INT	BIGINT
FLOAT	DOUBLE

以下示例为原始 orders_json 表使用 CREATE TABLE 语句创建一个名为 orders_json_bigint 的新表。新表使用 BIGINT 而不是 INT 作为 `o_shippriority` 列的数据类型。

```
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';
```

以下查询可成功运行，与更改数据类型前的原始 SELECT 查询类似：

```
Select * from orders_json
LIMIT 10;
```

包含分区的表中的更新

在 Athena 中，一个表及其分区必须使用相同的数据格式，但它们的架构可以不同。当您创建一个新分区时，该分区通常继承表的架构。随着时间的推移，该架构可能开始变得不同。原因包括：

- 当您的表的架构发生变化时，分区架构没有更新，从而没有保持与表的架构同步。
- AWS Glue 爬网程序允许您发现具有不同架构的分区中的数据。这意味着，假如您使用 AWS Glue 在 Athena 中创建了一个表，则当爬网程序完成处理后，该表的架构及其分区可能会有所不同。
- 如果您直接使用 AWS API 添加分区。

如果分区满足以下约束，Athena 会成功地处理具有分区的表。如果不满足这些约束，Athena 会发出 HIVE_PARTITION_SCHEMA_MISMATCH 错误。

- 每个分区的架构与表的架构兼容。
- 表的数据格式允许您要执行的更新类型：添加、删除、重新排序列或更改列的数据类型。

  例如，对于 CSV 和 TSV 格式，您可以将列重命名，在表的最后添加新列，并将列的数据类型更改为其他兼容类型，但您无法删除列。对于其他格式，可以添加或删除列，或者将列的数据类型更改为其他兼容类型。有关信息，请参阅摘要：Athena 中的更新和数据格式。

避免带有分区的表出现架构不匹配错误

在查询执行开始时，Athena 通过检查表和分区之间每个列数据类型是否兼容来验证表的架构。

- 对于 Parquet 和 ORC 数据存储类型，Athena 依赖于列名，并将其用于基于列名的架构验证。这消除了具有 Parquet 和 ORC 格式分区的表的 `HIVE_PARTITION_SCHEMA_MISMATCH` 错误。(如果 SerDe 属性设置为按名称访问索引，则对于 ORC 来说此为真：`orc.column.index.access=FALSE`。Parquet 默认按名称读取索引)。
- 对于 CSV、JSON 和 Avro，Athena 使用基于索引的架构验证。这意味着，如果您遇到架构不匹配错误，则应删除导致架构不匹配的分区并创新创建它，以便 Athena 可以成功地查询它。

Athena 会将表的架构与分区架构做比较。假如您使用 AWS Glue 爬网程序在 Athena 中创建了一个 CSV、JSON 和 AVRO 格式的表，则在爬网程序完成处理后，该表的架构和其分区的架构可能会不同。如果表的架构与分区架构之间存在不匹配，则您的 Athena 查询将因类似如下的架构验证错误而失败：'crawler_test.click_avro' is declared as type 'string', but partition 'partition_0=2017-01-17' declared column 'col68' as type 'double'. ('crawler_test.click_avro' 声明为类型 'string'，但分区 'partition_0=2017-01-17' 声明列 'col68' 的类型为 'double'。)

对于此类错误，通常的解决方法是删除导致错误的分区并重新创建它。有关更多信息，请参阅 ALTER TABLE DROP PARTITION 和 ALTER TABLE ADD PARTITION。

查询数组

Amazon Athena 允许您创建数组，合并它们，将它们转换为其他数据类型，然后筛选、展平和排序它们。

主题

创建数组

要在 Athena 中生成数组文本，请使用 ARRAY 关键字，后跟方括号 [ ]，并包括以逗号分隔的数组元素。

示例

此查询会创建一个具有四个元素的数组。

```
SELECT ARRAY [1,2,3,4] AS items
```

它返回：

```
+-----------+
```

```
| items     |
+-----------+
| [1,2,3,4] |
+-----------+
```

此查询创建两个数组。

```
SELECT ARRAY[ ARRAY[1,2], ARRAY[3,4] ] AS items
```

它返回：

```
+--------------------+
| items              |
+--------------------+
| [[1, 2], [3, 4]]   |
+--------------------+
```

要从兼容类型的选定列创建数组，请如以下示例所示使用查询：

```
WITH
dataset AS (
  SELECT 1 AS x, 2 AS y, 3 AS z
)
SELECT ARRAY [x,y,z] AS items FROM dataset
```

此查询返回：

```
+-----------+
| items     |
+-----------+
| [1,2,3]   |
+-----------+
```

在以下示例中，选定了两个数组，并将其作为欢迎词返回。

```
WITH
dataset AS (
  SELECT
    ARRAY ['hello', 'amazon', 'athena'] AS words,
    ARRAY ['hi', 'alexa'] AS alexa
```

```
)
SELECT ARRAY[words, alexa] AS welcome_msg
FROM dataset
```

此查询返回：

```
+----------------------------------------+
| welcome_msg                            |
+----------------------------------------+
| [[hello, amazon, athena], [hi, alexa]] |
+----------------------------------------+
```

要创建一个键值对，请使用 MAP 运算符，以便采用一个键数组后跟一个值数组，如以下示例所示：

```
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
```

此查询返回：

```
+-------------------------------------------------------------------------------------------------------------
+
| people
               |
+-------------------------------------------------------------------------------------------------------------
+
| [{last=Smith, first=Bob, age=40}, {last=Doe, first=Jane, age=30}, {last=Smith,
 first=Billy, age=8}] |
+-------------------------------------------------------------------------------------------------------------
+
```

连接字符串和数组

连接字符串

要连接两个字符串，可以使用双管道 **||** 运算符，如以下示例中所示。

```
SELECT 'This' || ' is' || ' a' || ' test.' AS Concatenated_String
```

此查询返回：

#	Concatenated_String
1	This is a test.

您可以使用 concat() 函数来获得相同的结果。

```
SELECT concat('This', ' is', ' a', ' test.') AS Concatenated_String
```

此查询返回：

#	Concatenated_String
1	This is a test.

您可以使用 concat_ws() 函数将字符串与第一个参数中的指定分隔符连接起来。

```
SELECT concat_ws(' ', 'This', 'is', 'a', 'test.') as Concatenated_String
```

此查询返回：

#	Concatenated_String
1	This is a test.

要使用点来连接字符串数据类型的两列，请使用双引号引用这两列，并将点括在单引号中作为硬编码字符串。如果某列不是字符串数据类型，则可以先使用 CAST("*column_name*" as VARCHAR) 转换该列。

```
SELECT "col1" || '.' || "col2" as Concatenated_String
FROM my_table
```

此查询返回：

#	Concatenated_String
1	*col1_string_value* .*col2_string_value*

连接数组

您可以使用相同的技术来连接数组。

要连接多个数组，请使用双管道 || 运算符。

```
SELECT ARRAY [4,5] || ARRAY[ ARRAY[1,2], ARRAY[3,4] ] AS items
```

此查询返回：

#	项目
1	[[4, 5], [1, 2], [3, 4]]

要将多个数组组合成一个数组，请使用双管道运算符或 concat() 函数。

```
WITH
dataset AS (
  SELECT
    ARRAY ['Hello', 'Amazon', 'Athena'] AS words,
    ARRAY ['Hi', 'Alexa'] AS alexa
)
SELECT concat(words, alexa) AS welcome_msg
FROM dataset
```

此查询返回：

#	welcome_msg
1	[Hello, Amazon, Athena, Hi, Alexa]

有关 concat() 或其他字符串函数的更多信息，请参阅 Trino 文档中的 字符串函数和运算符。

转换数组数据类型

要将数组中的数据转换为支持的数据类型，请使用 CAST 运算符，例如 `CAST(value AS type)`。Athena 支持所有本机 Presto 数据类型。

```
SELECT
   ARRAY [CAST(4 AS VARCHAR), CAST(5 AS VARCHAR)]
AS items
```

此查询返回：

```
+-------+
| items |
+-------+
| [4,5] |
+-------+
```

创建两个具有键值对元素的数组，将其转换为 JSON，并串联起来，如以下示例所示：

```
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
```

此查询返回：

```
+-------------------------------------------------+
| items                                           |
+-------------------------------------------------+
| [{"a1":1,"a2":2,"a3":3}, {"b1":4,"b2":5,"b3":6}] |
+-------------------------------------------------+
```

查找长度

`cardinality` 函数返回数组的长度，如下例所示：

```
SELECT cardinality(ARRAY[1,2,3,4]) AS item_count
```

此查询返回：

```
+------------+
| item_count |
+------------+
| 4          |
+------------+
```

访问数组元素

要访问数组元素，请使用 [] 运算符，用 1 指定第一个元素，用 2 指定第二个元素，以此类推，如以下示例所示：

```
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
```

此查询返回：

```
+------------------------+
| item                   |
+------------------------+
| {"a1":1,"a2":2,"a3":3} |
+------------------------+
```

要访问给定位置 (称为索引位置) 的数组元素，请使用 element_at() 函数并指定数组名称和索引位置：

- 如果索引大于 0，element_at() 将会返回您指定的元素，从数组的开头计数到末尾。它的行为与 [] 运算符一样。
- 如果索引小于 0，element_at() 将会返回元素，从数组的末尾计数到开头。

以下查询将创建一个数组 words，并从中选择第一个元素 hello 作为 first_word，选择第二个元素 amazon (从数组末尾计数) 作为 middle_word，选择第三个元素 athena 作为 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
```

此查询返回：

```
+----------------------------------------+
| first_word  | middle_word | last_word  |
+----------------------------------------+
| hello       | amazon      | athena     |
+----------------------------------------+
```

展平嵌套数组

当使用嵌套数组时，您通常需要将嵌套数组元素展开到单个阵列中，或将元素展开到多个行中。

示例

要将嵌套数组的元素展平为单个值数组，请使用 `flatten` 函数。此查询为数组中的每个元素返回一行。

```
SELECT flatten(ARRAY[ ARRAY[1,2], ARRAY[3,4] ]) AS items
```

此查询返回：

```
+-----------+
| items     |
+-----------+
| [1,2,3,4] |
+-----------+
```

要将数组展平为多个行，请将 `CROSS JOIN` 与 `UNNEST` 运算符结合使用，如以下示例所示：

```
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)
```

此查询返回：

```
+----------------------+
| department  | names  |
+----------------------+
| engineering | Sharon |
+----------------------|
| engineering | John   |
+----------------------|
| engineering | Bob    |
+----------------------|
| engineering | Sally  |
+----------------------+
```

展平一个键值对数组，将选定的键变换到列中，如以下示例所示：

```
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)
```

此查询返回：

```
+---------------------------------------+
| first_name | last_name | department  |
+---------------------------------------+
| Bob        | Smith     | engineering |
| Jane       | Doe       | engineering |
```

```
| Billy      | Smith     | engineering |
+-------------------------------------+
```

从员工列表中，选择具有最高组合分数的员工。可以在 FROM 子句中使用 UNNEST，而无需前面的 CROSS JOIN，因为它是默认的联接运算符，因此是隐含的。

```
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
```

此查询返回：

```
+---------------------------------+
| name | department | total_score |
+---------------------------------+
| Amy  | devops     | 54          |
+---------------------------------+
```

从员工列表中，选择具有最高个人分数的员工。

```
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
```

此查询返回：

```
+--------------+
| name | score |
+--------------+
| Amy  | 15    |
+--------------+
```

注意事项和限制

如果在查询中的一个或多个数组上使用 UNNEST，并且其中一个数组是 NULL，则查询不返回任何行。如果在空字符串数组上使用 UNNEST，则返回空字符串。

例如，在以下查询中，由于第二个数组为空值，因此查询不返回任何行。

```
SELECT
    col1,
    col2
FROM UNNEST (ARRAY ['apples','oranges','lemons']) AS t(col1)
CROSS JOIN UNNEST (ARRAY []) AS t(col2)
```

在下一个示例中，第二个数组被修改为包含一个空字符串。对于每一行，查询都会返回 col1 中的值，并为 col2 中的值返回空字符串。要返回第一个数组中的值，需要第二个数组中的空字符串。

```
SELECT
    col1,
    col2
FROM UNNEST (ARRAY ['apples','oranges','lemons']) AS t(col1)
CROSS JOIN UNNEST (ARRAY ['']) AS t(col2)
```

从子查询创建数组

从一组行创建数组。

```
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)
```

此查询返回：

```
+-----------------+
| array_items     |
+-----------------+
| [1, 2, 3, 4, 5] |
+-----------------+
```

要从一组行创建唯一值的数组，请使用 distinct 关键字。

```
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)
```

此查询返回以下结果。请注意，不保证排序。

```
+-----------------+
| array_items     |
+-----------------+
```

```
| [1, 2, 3, 4, 5] |
+-----------------+
```

有关使用 array_agg 函数的更多信息，请参阅 Trino 文档的聚合函数。

筛选数组

如果行集合与筛选条件匹配，则从这些行创建一个数组。

```
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
```

此查询返回：

```
+-------------+
| array_items |
+-------------+
| [4, 5]      |
+-------------+
```

根据数组的一个元素是否包含特定值 (例如 2) 来筛选该数组，如以下示例所示：

```
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)
```

此查询返回：

```
+--------------+
| array_items  |
+--------------+
| [1, 2, 3, 4] |
+--------------+
```

filter 函数

```
 filter(ARRAY [list_of_values], boolean_function)
```

您可以使用 filter 表达式上的 ARRAY 函数来创建新数组，该数组是 *boolean_function* 为真的 *list_of_values* 中项目的子集。filter 函数在您无法使用 *UNNEST* 函数时可起到帮助。

以下示例将筛选数组 [1,0,5,-1] 中大于零的值。

```
SELECT filter(ARRAY [1,0,5,-1], x -> x>0)
```

结果

```
[1,5]
```

以下示例将筛选数组 [-1, NULL, 10, NULL] 中非空的值。

```
SELECT filter(ARRAY [-1, NULL, 10, NULL], q -> q IS NOT NULL)
```

结果

```
[-1,10]
```

排序数组

要从一组行创建唯一值的排序数组，您可以使用 array_sort 函数，如以下示例中所示。

```
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)
```

此查询返回：

```
+--------------------+
| array_items        |
+--------------------+
| [1, 2, 3, 4, 5, 6] |
+--------------------+
```

要了解如何将数组展开为多行，请参阅 展平嵌套数组。

将聚合函数与数组结合使用

- 要在数组中添加值，请使用 SUM，如以下示例所示。
- 要在数组中聚合多个行，请使用 array_agg。有关信息，请参阅从子查询创建数组。

> **Note**
>
> 从 Athena 引擎版本 2 开始的聚合函数支持 ORDER BY。

```
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;
```

在最后一个 SELECT 语句中，您可以使用 reduce() 而不是 sum() 和 UNNEST 来减少处理时间和数据传输，如以下示例所示。

```
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;
```

任一查询返回以下结果。不保证返回结果的顺序。

```
+----------------------+
| array_items  | total |
+----------------------+
| [1, 2, 3, 4] | 10    |
| [5, 6, 7, 8] | 26    |
| [9, 0]       | 9     |
+----------------------+
```

将数组转换为字符串

要将数组转换为字符串，请使用 array_join 函数。下面的独立示例创建了一个名为的表 dataset，它包含一个名为 words 的已指定别名数组。该查询使用 array_join 将数组元素连接到 words，用空格分隔它们，然后将结果字符串返回到名为 welcome_msg 的别名列中。

```
WITH
dataset AS (
  SELECT ARRAY ['hello', 'amazon', 'athena'] AS words
)
SELECT array_join(words, ' ') AS welcome_msg
FROM dataset
```

此查询返回：

```
+---------------------+
```

```
| welcome_msg         |
+---------------------+
| hello amazon athena |
+---------------------+
```

使用数组创建映射

映射是由在 Athena 中可用的数据类型组成的键值对。要创建映射，请使用 MAP 运算符并将其传递给两个数组：第一个是列 (键) 名称，第二个是值。数组中的所有值都必须具有相同类型。如果有任何映射值数组元素需要具有不同类型，您可以以后转换它们。

示例

此示例会从数据集中选择用户。它使用 MAP 运算符并将其传递给两个数组。第一个数组包含列名称的值，例如“第一个”、“最后一个”和“年龄”。第二个数组包含每个列的值，例如“Bob”、“Smith”、“35”。

```
WITH dataset AS (
  SELECT MAP(
    ARRAY['first', 'last', 'age'],
    ARRAY['Bob', 'Smith', '35']
  ) AS user
)
SELECT user FROM dataset
```

此查询返回：

```
+---------------------------------+
| user                            |
+---------------------------------+
| {last=Smith, first=Bob, age=35} |
+---------------------------------+
```

您可以通过选择字段名称后跟 [key_name] 来检索 Map 值，如以下示例所示：

```
WITH dataset AS (
 SELECT MAP(
   ARRAY['first', 'last', 'age'],
   ARRAY['Bob', 'Smith', '35']
 ) AS user
)
SELECT user['first'] AS first_name FROM dataset
```

此查询返回：

```
+------------+
| first_name |
+------------+
| Bob        |
+------------+
```

查询具有复杂类型和嵌套结构的数组

您的源数据通常包含具有复杂数据类型和嵌套结构的数组。本部分中的示例显示如何使用 Athena 查询更改元素的数据类型，在数组内找到元素，以及查找关键字。

- 创建ROW
- 使用 CAST 更改数组中的字段名称
- 使用 . 表示法筛选数组
- 筛选具有嵌套值的数组
- 使用 UNNEST 筛选数组
- 使用 regexp_like 在数组中查找关键字

创建**ROW**

> **Note**
>
> 本部分中的示例使用 ROW 作为一种方法来创建样本数据以供使用。当您在 Athena 中查询表时，无需创建 ROW 数据类型，因为它们已从数据源创建。当您使用 CREATE_TABLE 时，Athena 将针对数据集中的每一行，在其中定义 STRUCT，向其填充数据，并为您创建 ROW 数据类型。底层 ROW 数据类型包含支持的任何 SQL 数据类型的命名字段。

```
WITH dataset AS (
 SELECT
   ROW('Bob', 38) AS users
 )
SELECT * FROM dataset
```

此查询返回：

```
+-------------------------+
| users                   |
+-------------------------+
| {field0=Bob, field1=38} |
+-------------------------+
```

使用 **CAST** 更改数组中的字段名称

要更改包含 ROW 值的数组中的字段名称，您可以对 ROW 声明执行 CAST 操作：

```
WITH dataset AS (
  SELECT
    CAST(
      ROW('Bob', 38) AS ROW(name VARCHAR, age INTEGER)
    ) AS users
)
SELECT * FROM dataset
```

此查询返回：

```
+--------------------+
| users              |
+--------------------+
| {NAME=Bob, AGE=38} |
+--------------------+
```

> **Note**
>
> 在上述示例中，您可以将 name 声明为 VARCHAR，因为这是它在 Presto 中的类型。如果您在 CREATE TABLE 语句中声明此 STRUCT，请使用 String 类型，因为 Hive 将此数据类型定义为 String。

使用 **.** 表示法筛选数组

在以下示例中，使用点 . 表示法从 AWS CloudTrail 日志表的 userIdentity 列中选择 accountId 字段。有关更多信息，请参阅查询 AWS CloudTrail 日志。

```
SELECT
  CAST(useridentity.accountid AS bigint) as newid
```

```
FROM cloudtrail_logs
LIMIT 2;
```

此查询返回：

```
+--------------+
| newid        |
+--------------+
| 112233445566 |
+--------------+
| 998877665544 |
+--------------+
```

要查询一个值数组，请发出以下查询：

```
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
```

它将返回此结果：

```
+-----------------------------------------------------------------+
| users                                                           |
+-----------------------------------------------------------------+
| [{NAME=Bob, AGE=38}, {NAME=Alice, AGE=35}, {NAME=Jane, AGE=27}] |
+-----------------------------------------------------------------+
```

筛选具有嵌套值的数组

大型数组通常包含嵌套结构，您需要能够对其中的值进行筛选或搜索。

要为包含嵌套 BOOLEAN 值的值数组定义数据集，请发出以下查询：

```
WITH dataset AS (
  SELECT
    CAST(
```

```
      ROW('aws.amazon.com', ROW(true)) AS ROW(hostname VARCHAR, flaggedActivity
 ROW(isNew BOOLEAN))
    ) AS sites
)
SELECT * FROM dataset
```

它将返回此结果：

```
+-----------------------------------------------------------+
| sites                                                     |
+-----------------------------------------------------------+
| {HOSTNAME=aws.amazon.com, FLAGGEDACTIVITY={ISNEW=true}}   |
+-----------------------------------------------------------+
```

接下来，要筛选和访问该元素的 BOOLEAN 值，请继续使用点 . 表示法。

```
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
```

此查询选择嵌套字段并返回此结果：

```
+------------------------+
| hostname       | isnew |
+------------------------+
| aws.amazon.com | true  |
+------------------------+
```

使用 UNNEST 筛选数组

要按照一个子元素筛选包含嵌套结构的数组，请发出具有 UNNEST 运算符的查询。有关 UNNEST 的更多信息，请参阅展平嵌套数组。

例如，此查询在数据集中查找站点的主机名。

```
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
```

它返回：

```
+-------------------------+
| hostname       | isnew |
+-------------------------+
| aws.amazon.com | true  |
+-------------------------+
```

使用 **regexp_like** 在数组中查找关键字

以下示例说明如何使用 regexp_like 函数搜索数据集以查找数组内元素中的关键字。它以一个要计算的正则表达式模式（或一个由竖线 (|) 分隔的搜索词列表）作为输入，计算此模式，并确定指定的字符串是否包含此模式。

此正则表达式模式需要包含在此字符串内，但并不一定要与此字符串匹配。要匹配整个字符串，请在模式开头使用 ^ 并在末尾使用 $ 将模式括起来，例如 '^pattern$'。

请考虑包含其主机名的站点和 flaggedActivity 元素的数组。此元素包含一个 ARRAY，其中包含多个 MAP 元素，每个元素列出不同的流行关键字及其受欢迎程度计数。假设您要在此数组中的 MAP 内查找特定关键字。

要搜索此数据集以查找具有特定关键字的网站，我们使用 regexp_like 而不是类似的 SQL LIKE 运算符，因为使用 regexp_like 搜索大量关键字的效率更高。

Example 示例 1：使用 **regexp_like**

本示例中的查询使用 regexp_like 函数来搜索可在数组内的值中找到的词 'politics|bigdata'：

```
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)
```

此查询返回两个站点：

```
+----------------+
| hostname       |
+----------------+
| aws.amazon.com |
+----------------+
| news.cnn.com   |
+----------------+
```

Example 示例 2：使用 **regexp_like**

以下示例中的查询将搜索词与 regexp_like 函数匹配的站点的总受欢迎度分数相加，然后将它们从最高到最低排序。

```
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
```

此查询返回两个站点：

```
+-------------------------------------+
| hostname       | terms    | total  |
+----------------+------------------+
| news.cnn.com   | politics |  241   |
+----------------+------------------+
| aws.amazon.com | bigdata |  10    |
+----------------+------------------+
```

查询地理空间数据

地理空间数据包含为对象指定地理位置的标识符。此类数据的示例包括天气报告、地图方向、包含地理位置的推特、店铺位置以及航空公司路线。地理空间数据在业务分析、报告和预测中起着重要的作用。

地理空间标识符 (如经度和纬度) 允许您将任何邮寄地址转换为一组地理坐标。

主题

- 什么是地理空间查询？
- 输入数据格式和 Geometry 数据类型

- 支持的地理空间函数
- 示例：地理空间查询

什么是地理空间查询？

地理空间查询是在 Athena 中支持的专门类型 SQL 查询。它们在以下方面与非空间 SQL 查询不同：

- 使用以下专门 geometry 数据类型：`point`、`line`、`multiline`、`polygon` 和 `multipolygon`。
- 表示 geometry 数据类型之间的关系，例如 `distance`、`equals`、`crosses`、`touches`、`overlaps`、`disjoint` 等。

通过在 Athena 中使用地理空间查询，您可以运行这些操作和其他类似操作：

- 找出两点之间的距离。
- 检查一个区域 (多边形) 是否包含另一个区域。
- 检查一条线是否穿过或接触另一条线或多边形。

例如，要在 Athena 中从雷尼尔山地理坐标的类型 `double` 的值获得 `point` 几何数据类型，请使用 `ST_Point (longitude, latitude)` 几何空间函数，如以下示例所示。

```
ST_Point(-121.7602, 46.8527)
```

输入数据格式和 Geometry 数据类型

要在 Athena 中使用地理空间函数，请以 WKT 格式输入数据，或使用 Hive JSON SerDe。您还可以使用在 Athena 中受支持的几何数据类型。

输入数据格式

为处理地理空间查询，Athena 支持以下数据格式的输入数据：

- WKT (已知文本)。在 Athena 中，WKT 表示为 `varchar(x)` 或 `string` 数据类型。
- JSON 编码的地理空间数据。为解析具有地理空间数据的 JSON 文件并为它们创建表，Athena 使用 Hive JSON SerDe。有关在 Athena 中使用此 SerDe 的更多信息，请参阅 JSON SerDe 库。

Geometry 数据类型

为处理地理空间查询，Athena 支持以下专用几何数据类型：

- `point`
- `line`
- `polygon`
- `multiline`
- `multipolygon`

支持的地理空间函数

Athena 中可用的地理空间函数取决于您使用的引擎版本。

- 有关 Athena 引擎版本 3 中地理空间函数的信息，请参阅 Trino 文档中的 地理空间函数。
- 有关从 Athena 引擎版本 2 以来的函数名称更改和新函数列表，请参阅 Athena 引擎版本 2 中的地理空间函数名称更改和新函数。

有关 Athena 引擎版本控制的更多信息，请参阅 Athena 引擎版本控制。

主题

- Athena 引擎版本 3 中的地理空间函数
- Athena 引擎版本 2 中的地理空间函数

Athena 引擎版本 3 中的地理空间函数

有关 Athena 引擎版本 3 中地理空间函数的信息，请参阅 Trino 文档中的 地理空间函数。

Athena 引擎版本 2 中的地理空间函数

本主题列出了从 Athena 引擎版本 2 开始支持的 ESRI 地理空间函数。有关 Athena 引擎版本的更多信息，请参阅 Athena 引擎版本控制。

Athena 引擎版本 2 中的更改

- 某些函数的输入和输出类型已经改变。最值得注意的是，输入不再直接支持 `VARBINARY` 类型。有关更多信息，请参阅 地理空间函数的更改。

- 一些地理空间函数的名称已经改变。有关更多信息，请参阅 Athena 引擎版本 2 中的地理空间函数名称更改。
- 增加了新函数。有关更多信息，请参阅 Athena 引擎版本 2 中新增的地理空间函数。

Athena 支持以下类型的地理空间函数：

- 构造函数
- 地理空间关系函数
- 操作函数
- 访问器函数
- 聚合函数
- Bing 磁贴函数

构造函数

使用构造函数可获取 `point`、`line` 或 `polygon` geometry 数据类型的二进制表示。您也可以使用这些函数来将二进制数据转换为文本数据，以及获取以已知文本 (WKT) 格式表示的 geometry 数据的二进制值。

ST_AsBinary(geometry)

返回包含指定几何的 WKB 表示形式的变量二进制数据类型。例如：

```
SELECT ST_AsBinary(ST_Point(-158.54, 61.56))
```

ST_AsText(geometry)

将每个指定的 geometry 数据类型转换为文本。返回一个 varchar 数据类型的值，它是该 geometry 数据类型的 WKT 表示。例如：

```
SELECT ST_AsText(ST_Point(-158.54, 61.56))
```

ST_GeomAsLegacyBinary(geometry)

从指定几何类型返回遗留变量二进制对象。例如：

```
SELECT ST_GeomAsLegacyBinary(ST_Point(-158.54, 61.56)
```

ST_GeometryFromText(varchar)

将 WKT 格式的文本转换为 geometry 数据类型。返回一个几何数据类型的值。例如：

```
SELECT ST_GeometryFromText(ST_AsText(ST_Point(1, 2)))
```

ST_GeomFromBinary(varbinary)

从 WKB 表示形式中返回几何类型对象。例如：

```
SELECT ST_GeomFromBinary(ST_AsBinary(ST_Point(-158.54, 61.56)))
```

ST_GeomFromLegacyBinary(varbinary)

从遗留变量二进制类型返回几何类型对象。例如：

```
SELECT ST_GeomFromLegacyBinary(ST_GeomAsLegacyBinary(ST_Point(-158.54, 61.56)))
```

ST_LineFromText(varchar)

返回几何数据类型 line 中的值。例如：

```
SELECT ST_Line('linestring(1 1, 2 2, 3 3)')
```

ST_LineString(array(point))

返回从点几何类型数组中形成的 LineString 几何类型。如果指定数组中的非空点少于两个，则返回空 LineString。如果数组中的任何元素为空值、空或与前一个元素相同，则引发异常。返回的几何体可能不是简单的几何体。根据指定的输入，返回的几何可以自相交或包含重复的顶点。例如：

```
SELECT ST_LineString(ARRAY[ST_Point(-158.54, 61.56), ST_Point(-158.55, 61.56)])
```

ST_MultiPoint(array(point))

返回从指定点形成的 MultiPoint 几何对象。如果指定的数组为空，则返回空值。如果数组中的任何元素为空值或空，则引发异常。返回的几何体可能不是简单的几何体，如果指定的数组具有重复，则返回的几何体可能会包含重复点。例如：

```
SELECT ST_MultiPoint(ARRAY[ST_Point(-158.54, 61.56), ST_Point(-158.55, 61.56)])
```

ST_Point(double, double)

返回几何类型 point 对象。对于此函数的输入数据值，请使用几何值，例如墨卡托方位法 (UTM) 笛卡尔坐标系中的值，或者采用十进制度的地图单位（经度和纬度）。经度和纬度值使用世界大地测量系统，也称为 WGS 1984 或 EPSG:4326。WGS 1984 是全球定位系统 (GPS) 使用的坐标系。

例如，在以下表示法中，使用经度和纬度指定地图坐标，值 .072284 是缓冲距离，使用十进制度的角度单位指定。

```
SELECT ST_Buffer(ST_Point(-74.006801, 40.705220), .072284)
```

语法：

```
SELECT ST_Point(longitude, latitude) FROM earthquakes LIMIT 1
```

以下示例使用特定经度和纬度坐标：

```
SELECT ST_Point(-158.54, 61.56)
FROM earthquakes
LIMIT 1
```

下一个示例使用特定经度和纬度坐标：

```
SELECT ST_Point(-74.006801, 40.705220)
```

以下示例使用 ST_AsText 函数来从 WKT 获取几何：

```
SELECT ST_AsText(ST_Point(-74.006801, 40.705220)) AS WKT
```

ST_Polygon(varchar)

使用顺时针（从左到右）提供的纵坐标序列，返回几何数据类型 polygon。从 Athena 引擎版本 2 开始，只接受多边形作为输入。例如：

```
SELECT ST_Polygon('polygon ((1 1, 1 4, 4 4, 4 1))')
```

to_geometry(sphericalGeography)

从指定的球形地理对象返回几何对象。例如：

```
SELECT to_geometry(to_spherical_geography(ST_Point(-158.54, 61.56)))
```

to_spherical_geography(geometry)

返回指定几何中的球形地理对象。使用此函数可将几何对象转换为地球半径球体上的球形地理对象。此函数只能用于在 2D 空间中定义的 POINT、MULTIPOINT、LINESTRING、MULTILINESTRING、POLYGON 和 MULTIPOLYGON 几何或该等几何形状的 GEOMETRYCOLLECTION。对于指定几何形状的每个点，函数将验证 point.x 位于 [-180.0, 180.0] 内，且 point.y 位于 [-90.0, 90.0] 内。该函数使用这些点作为经度和纬度来构建 sphericalGeography 结果的形状。

例如：

```
SELECT to_spherical_geography(ST_Point(-158.54, 61.56))
```

地理空间关系函数

以下函数表示您指定为输入并返回类型 boolean 结果的两个不同几何体之间的关系。指定几何体对的顺序很重要：第一个几何体值称为左几何体，第二个几何体值称为右几何体。

这些函数返回：

- 当且仅当满足了函数所描述的关系时为 TRUE。
- 当且仅当不满足函数所描述的关系时为 FALSE。

ST_Contains(geometry, geometry)

当且仅当左几何体包含右几何体时返回 TRUE。示例：

```
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)

当且仅当左几何体穿过右几何体时返回 TRUE。例如：

```
SELECT ST_Crosses(ST_Line('linestring(1 1, 2 2 )'), ST_Line('linestring(0 1, 2 2)'))
```

ST_Disjoint(geometry, geometry)

当且仅当左几何体和右几何体的交集为空时返回 TRUE。例如：

```
SELECT ST_Disjoint(ST_Line('linestring(0 0, 0 1)'), ST_Line('linestring(1 1, 1 0)'))
```

ST_Equals(geometry, geometry)

当且仅当左几何体等于右几何体时返回 TRUE。例如：

```
SELECT ST_Equals(ST_Line('linestring( 0 0, 1 1)'), ST_Line('linestring(1 3, 2 2)'))
```

ST_Intersects(geometry, geometry)

当且仅当左几何体与右几何体相交时返回 TRUE。例如：

```
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)

当且仅当左几何体与右几何体重叠时返回 TRUE。例如：

```
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)

当且仅当左几何体与右几何体具有指定的尺寸扩展九交集模型 (DE-9IM) 关系时返回 TRUE。第三个 (varchar) 输入接受关系。例如：

```
SELECT ST_Relate(ST_Line('linestring(0 0, 3 3)'), ST_Line('linestring(1 1, 4 4)'),
 'T********')
```

ST_Touches(geometry, geometry)

当且仅当左几何体与右几何体接触时返回 TRUE。

例如：

```
SELECT ST_Touches(ST_Point(8, 8), ST_Polygon('polygon((1  1, 1  4, 4  4, 4 1))'))
```

ST_Within(geometry, geometry)

当且仅当左几何体位于右几何体内时返回 TRUE。

例如：

```
SELECT ST_Within(ST_Point(8, 8), ST_Polygon('polygon((1  1, 1  4, 4  4, 4 1))'))
```

操作函数

使用操作函数可对 geometry 数据类型值执行操作。例如，您可以获取单个 geometry 数据类型的边界；两个 geometry 数据类型之间的交集；左、右几何体之间的差异 (其中每个几何体都具有相同的 geometry 数据类型)；或围绕特定 geometry 数据类型的外部缓冲区或环。

geometry_union(array(geometry))

返回表示指定几何体的点集合并集的几何体。例如：

```
SELECT geometry_union(ARRAY[ST_Point(-158.54, 61.56), ST_Point(-158.55, 61.56)])
```

ST_Boundary(geometry)

采用一个几何数据类型作为输入，并返回该 boundary 几何数据类型。

示例：

```
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)

将 geometry 数据类型（例如点、线、多边形、复线和多个多边形）之一作为输入，并将距离作为类型 double。返回按指定距离（或半径）缓冲的几何数据类型。例如：

```
SELECT ST_Buffer(ST_Point(1, 2), 2.0)
```

在以下示例中，使用经度和纬度指定地图坐标，值 .072284 是缓冲距离，使用十进制度的角度单位指定。

```
SELECT ST_Buffer(ST_Point(-74.006801, 40.705220), .072284)
```

ST_Difference(geometry, geometry)

返回左几何体和右几何体之间的差异几何体。例如：

```
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)

获取作为输入 line、polygon、multiline 和 multipolygon geometry 数据类型。不支持 point geometry 数据类型。返回一个信封作为几何体，其中信封是一个围绕指定几何数据类型的矩形。示例：

```
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)

返回一个由两个点组成的数组，它们表示几何的边界矩形多边形的左下角和右上角。当指定的几何体为空时返回空值。例如：

```
SELECT ST_EnvelopeAsPts(ST_Point(-158.54, 61.56))
```

ST_ExteriorRing(geometry)

返回输入类型 polygon 的外部环的几何体。从 Athena 引擎版本 2 开始，多边形是唯一接受的几何体输入。示例：

```
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)

返回左几何体和右几何体交集的几何体。示例：

```
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)

返回左几何体和右几何体之间的几何对称差异的几何体。例如：

```
SELECT ST_AsText(ST_SymDifference(ST_Line('linestring(0 2, 2 2)'),
 ST_Line('linestring(1 2, 3 2)')))
```

ST_Union(geometry, geometry)

返回表示指定几何体的点集合并集的几何数据类型。例如：

```
SELECT ST_Union(ST_Point(-158.54, 61.56),ST_LineString(array[ST_Point(1,2),
 ST_Point(3,4)]))
```

访问器函数

访问器函数可用于从不同 geometry 数据类型中获取类型 varchar、bigint 或 double 的值，其中 geometry 是 Athena 中支持的任何几何数据类型：point、line、polygon、multiline 和 multipolygon。例如，您可以获取 polygon geometry 数据类型的面积、指定 geometry 数据类型的最大和最小 x 和 y 值，获取 line 的长度，或接收指定 geometry 数据类型中的点数。

geometry_invalid_reason(geometry)

以 varchar 数据类型返回指定几何体无效或不简单的原因。如果指定的几何体既无效也不是简单的几何体，则返回其无效的原因。如果指定的几何体有效且简单，则返回空值。例如：

```
SELECT geometry_invalid_reason(ST_Point(-158.54, 61.56))
```

great_circle_distance(latitude1, longitude1, latitude2, longitude2)

以双精度形式返回地球表面上两点之间的大圆距离（以千米为单位）。例如：

```
SELECT great_circle_distance(36.12, -86.67, 33.94, -118.40)
```

line_locate_point(lineString, point)

返回 0 到 1 之间的双精度值，表示指定线字符串上与指定点的最近点的位置，作为 2D 线总长度的一部分。

如果指定的线字符串或点为空或空值，则返回空值。例如：

```
SELECT line_locate_point(ST_GeometryFromText('LINESTRING (0 0, 0 1)'), ST_Point(0,
 0.2))
```

simplify_geometry(geometry, double)

使用 Ramer-Douglas-Peucker 算法返回指定几何体的简化版本的几何体数据类型。避免创建无效的派生几何体（特别是多边形）。例如：

```
SELECT simplify_geometry(ST_GeometryFromText('POLYGON ((1 0, 2 1, 3 1, 3 1, 4 1, 1
 0))'), 1.5)
```

ST_Area(geometry)

采用几何数据类型作为输入，返回类型为 double 的面积。例如：

```
SELECT ST_Area(ST_Polygon('polygon((1 1, 4 1, 4 4, 1 4))'))
```

ST_Centroid(geometry)

采用几何体数据类型 polygon 作为输入，并返回作为多边形信封中心的 point 几何体数据类型。示例：

```
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)

返回几何体数据类型，该类型是包含指定输入中所有几何体的最小凸几何体。例如：

```
SELECT ST_ConvexHull(ST_Point(-158.54, 61.56))
```

ST_CoordDim(geometry)

采用支持的 geometry 数据类型作为输入，并返回坐标分量的计数（类型为 tinyint）。例如：

```
SELECT ST_CoordDim(ST_Point(1.5,2.5))
```

ST_Dimension(geometry)

采用一个支持的 geometry 数据类型作为输入，并返回一个几何体的空间维度 (类型为 tinyint）。例如：

```
SELECT ST_Dimension(ST_Polygon('polygon((1 1, 4 1, 4 4, 1 4))'))
```

ST_Distance(geometry, geometry)

根据空间参考，双精度形式包含以投影单位返回两个几何体之间的二维最小笛卡尔距离。从 Athena 引擎版本 2 开始，如果其中一个输入为空几何体，则返回空值。例如：

```
SELECT ST_Distance(ST_Point(0.0,0.0), ST_Point(3.0,4.0))
```

ST_Distance(sphericalGeography, sphericalGeography)

以双精度形式返回两个球形地理点之间的大圆距离（以米为单位）。例如：

```
SELECT ST_Distance(to_spherical_geography(ST_Point(61.56,
 -86.67)),to_spherical_geography(ST_Point(61.56, -86.68)))
```

ST_EndPoint(geometry)

返回 line 几何体数据类型的最后一个点（类型为 point）。例如：

```
SELECT ST_EndPoint(ST_Line('linestring(0 2, 2 2)'))
```

ST_Geometries(geometry)

返回指定集合中的几何体数组。如果指定的几何不是多几何体，则返回一个元素的数组。如果指定的几何体为空，将返回空值。

例如，假设 MultiLineString 对象，ST_Geometries 将创建一个 LineString 对象数组。假设 GeometryCollection 对象，ST_Geometries 将返回一个由其成分组成的未拼合数组。例如：

```
SELECT ST_Geometries(GEOMETRYCOLLECTION(MULTIPOINT(0 0, 1 1),
 GEOMETRYCOLLECTION(MULTILINESTRING((2 2, 3 3)))))
```

结果：

```
array[MULTIPOINT(0 0, 1 1),GEOMETRYCOLLECTION(MULTILINESTRING((2 2, 3 3)))]
```

ST_GeometryN(geometry, index)

作为几何数据类型返回指定整数索引处的几何体元素。索引从 1 开始。如果指定的几何体是几何体的集合（例如，GEOMETRYCOLLECTION 或者 MULTI* 对象），则返回位于指定索引处的几何体。如果指定的索引小于 1 或大于集合中元素的总数，则返回空值。要查找元素总数，请使用 ST_NumGeometries。奇异几何体（例如 POINT、LINESTRING 或者 POLYGON）将被视为一个元素的集合。空几何体被视为空集合。例如：

```
SELECT ST_GeometryN(ST_Point(-158.54, 61.56),1)
```

ST_GeometryType(geometry)

以 varchar 的形式返回几何体的类型。例如：

```
SELECT ST_GeometryType(ST_Point(-158.54, 61.56))
```

ST_InteriorRingN(geometry, index)

返回指定索引处的内部环形元素（指数从 1 开始）。如果给定索引小于 1 或大于指定几何中的内环总数，则返回空值。如果指定的几何体不是多边形，则会引发错误。要查找元素总数，请使用 ST_NumInteriorRing。例如：

```
SELECT ST_InteriorRingN(st_polygon('polygon ((0 0, 1 0, 1 1, 0 1, 0 0))'),1)
```

ST_InteriorRings(geometry)

返回指定几何体中找到的所有内环的几何体数组，如果多边形没有内环，则返回一个空数组。如果指定的几何体为空，将返回空值。如果指定的几何体不是多边形，则会引发错误。例如：

```
SELECT ST_InteriorRings(st_polygon('polygon ((0 0, 1 0, 1 1, 0 1, 0 0))'))
```

ST_IsClosed(geometry)

获取作为仅用于输入的 line 和 multiline geometry 数据类型。当且仅当线条闭合时返回 TRUE (类型 boolean)。例如：

```
SELECT ST_IsClosed(ST_Line('linestring(0 2, 2 2)'))
```

ST_IsEmpty(geometry)

获取作为仅用于输入的 line 和 multiline geometry 数据类型。当且仅当指定的几何体为空时，返回 TRUE（类型 boolean），换言之，在 line 开始值和结束值均位于内部时。例如：

```
SELECT ST_IsEmpty(ST_Point(1.5, 2.5))
```

ST_IsRing(geometry)

当且仅当 line 类型闭合且简单时返回 TRUE (类型 boolean)。例如：

```
SELECT ST_IsRing(ST_Line('linestring(0 2, 2 2)'))
```

ST_IsSimple(geometry)

如果指定的几何体没有异常几何点（例如，自相交或自相切），则返回 true。要确定几何体不是简单几何体的原因，请使用 geometry_invalid_reason()。例如：

```
SELECT ST_IsSimple(ST_LineString(array[ST_Point(1,2), ST_Point(3,4)]))
```

ST_IsValid(geometry)

当且仅当指定的几何体形态良好时返回 true。要确定几何体形态不佳的原因，请使用 geometry_invalid_reason()。例如：

```
SELECT ST_IsValid(ST_Point(61.56, -86.68))
```

ST_Length(geometry)

返回 line 的长度 (类型为 double)。例如：

```
SELECT ST_Length(ST_Line('linestring(0 2, 2 2)'))
```

ST_NumGeometries(geometry)

以整数形式返回集合中的几何体数。如果几何体是几何体的集合（例如，GEOMETRYCOLLECTION 或者 MULTI* 对象），则返回几何体的数量。单个几何体返回 1；空几何体返回 0。GEOMETRYCOLLECTION 对象中的一个空的几何体计为一个几何体。例如，以下示例估算为 1：

```
ST_NumGeometries(ST_GeometryFromText('GEOMETRYCOLLECTION(MULTIPOINT EMPTY)'))
```

ST_NumInteriorRing(geometry)

返回 polygon 几何体中的内部环数 (类型为 bigint)。例如：

```
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)

返回几何体中的点数 (类型为 bigint)。例如：

```
SELECT ST_NumPoints(ST_Point(1.5, 2.5))
```

ST_PointN(lineString, index)

以点几何体数据类型返回指定整数索引处指定线字符串的折点。索引从 1 开始。如果给定索引小于 1 或大于集合中元素的总数，则返回空值。要查找元素总数，请使用 ST_NumPoints。例如：

```
SELECT ST_PointN(ST_LineString(array[ST_Point(1,2), ST_Point(3,4)]),1)
```

ST_Points(geometry)

返回指定线字符串几何体对象中的点数组。例如：

```
SELECT ST_Points(ST_LineString(array[ST_Point(1,2), ST_Point(3,4)]))
```

ST_StartPoint(geometry)

返回 line 几何体数据类型的第一个点（类型为 point）。例如：

```
SELECT ST_StartPoint(ST_Line('linestring(0 2, 2 2)'))
```

ST_X(point)

返回点的 X 坐标 (类型为 double)。例如：

```
SELECT ST_X(ST_Point(1.5, 2.5))
```

ST_XMax(geometry)

返回几何体的最大 X 坐标 (类型为 double)。例如：

```
SELECT ST_XMax(ST_Line('linestring(0 2, 2 2)'))
```

ST_XMin(geometry)

返回几何体的最小 X 坐标 (类型为 double)。例如：

```
SELECT ST_XMin(ST_Line('linestring(0 2, 2 2)'))
```

ST_Y(point)

返回点的 Y 坐标 (类型为 double)。例如：

```
SELECT ST_Y(ST_Point(1.5, 2.5))
```

ST_YMax(geometry)

返回几何体的最大 Y 坐标 (类型为 double)。例如：

```
SELECT ST_YMax(ST_Line('linestring(0 2, 2 2)'))
```

ST_YMin(geometry)

返回几何体的最小 Y 坐标 (类型为 double)。例如：

```
SELECT ST_YMin(ST_Line('linestring(0 2, 2 2)'))
```

聚合函数

convex_hull_agg(geometry)

返回包含作为输入传递的所有几何体的最小凸几何体。

geometry_union_agg(geometry)

返回一个几何体，该几何体表示作为输入传递的所有几何体的点集并集。

Bing 磁贴函数

以下函数在 Microsoft Bing 地图磁贴系统中的几何体和磁贴之间进行转换。

bing_tile(x, y, zoom_level)

从整数坐标 x 和 y 并按照指定缩放级别返回 Bing 磁贴目标。缩放级别必须是介于 1 到 23 之间的整数。例如：

```
SELECT bing_tile(10, 20, 12)
```

bing_tile(quadKey)

从四键返回 Bing 磁贴对象。例如：

```
SELECT bing_tile(bing_tile_quadkey(bing_tile(10, 20, 12)))
```

bing_tile_at(latitude, longitude, zoom_level)

返回指定纬度、经度和缩放级别的 Bing 磁贴对象。纬度必须介于 -85.05112878 和 85.05112878 之间。经度必须介于-180 和 180 之间。`latitude` 和 `longitude` 值必须为 `double` 和 `zoom_level` 一个整数。例如：

```
SELECT bing_tile_at(37.431944, -122.166111, 12)
```

bing_tiles_around(latitude, longitude, zoom_level)

返回指定缩放级别且环绕指定纬度和经度点的 Bing 磁贴数组。例如：

```
SELECT bing_tiles_around(47.265511, -122.465691, 14)
```

bing_tiles_around(latitude, longitude, zoom_level, radius_in_km)

返回指定缩放级别的 Bing 磁贴数组。该数组包含最小的 Bing 磁贴集，这些磁贴覆盖指定纬度和经度周围的指定半径（以千米为单位）。latitude、longitude 和 radius_in_km 值为 double；缩放级别为 integer。例如：

```
SELECT bing_tiles_around(37.8475, 112.596667, 10, .5)
```

bing_tile_coordinates(tile)

返回指定 Bing 磁贴的 x 和 y 坐标。例如：

```
SELECT bing_tile_coordinates(bing_tile_at(37.431944, -122.166111, 12))
```

bing_tile_polygon(tile)

返回指定 Bing 磁贴的多边形表示形式。例如：

```
SELECT bing_tile_polygon(bing_tile_at(47.265511, -122.465691, 4))
```

bing_tile_quadkey(tile)

返回指定 Bing 磁贴的四键。例如：

```
SELECT bing_tile_quadkey(bing_tile(52, 143, 10))
```

bing_tile_zoom_level(tile)

以整数形式返回指定 Bing 磁贴的缩放级别。例如：

```
SELECT bing_tile_zoom_level(bing_tile(52, 143, 10))
```

`geometry_to_bing_tiles(geometry, zoom_level)`

返回在指定缩放级别且完全覆盖指定几何体的 Bing 磁贴的最小集合。支持从 1 到 23 的缩放级别。例如：

```
SELECT geometry_to_bing_tiles(ST_Point(61.56, 58.54), 10)
```

Athena 引擎版本 2 中的地理空间函数名称更改和新函数

本部分列出了 Athena 引擎版本 2 中地理空间函数名称 的更改以及新增的地理空间函数。有关从 Athena 引擎版本 2 以来的其他更改信息，请参阅 Athena 引擎版本 2。

有关 Athena 引擎版本控制的更多信息，请参阅 Athena 引擎版本控制。

Athena 引擎版本 2 中的地理空间函数名称更改

以下函数的名称已更改。在某些情况下，输入和输出类型也发生了变化。有关更多信息，请访问相应的链接。

旧函数名称	从 Athena 引擎版本 2 开始的函数名称
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

旧函数名称	从 Athena 引擎版本 2 开始的函数名称
st_min_y	ST_YMin
st_point_number	ST_NumPoints
st_start_point	ST_StartPoint
st_symmetric_difference	ST_SymDifference

Athena 引擎版本 2 中新增的地理空间函数

以下地理空间函数是从 Athena 引擎版本 2 以来新增的函数。有关更多信息，请访问相应的链接。

构造函数

- ST_AsBinary
- ST_GeomAsLegacyBinary
- ST_GeomFromBinary
- ST_GeomFromLegacyBinary
- ST_LineString
- ST_MultiPoint
- to_geometry
- to_spherical_geography

操作函数

- geometry_union
- ST_EnvelopeAsPts
- ST_Union

访问器函数

- geometry_invalid_reason
- great_circle_distance

- line_locate_point
- simplify_geometry
- ST_ConvexHull
- ST_Distance (spherical geography)
- ST_Geometries
- ST_GeometryN
- ST_GeometryType
- ST_InteriorRingN
- ST_InteriorRings
- ST_IsSimple
- ST_IsValid
- ST_NumGeometries
- ST_PointN
- ST_Points

聚合函数

- convex_hull_agg
- geometry_union_agg

Bing 磁贴函数

- bing_tile
- bing_tile (quadkey)
- bing_tile_at
- bing_tiles_around
- bing_tiles_around (radius)
- bing_tile_coordinates
- bing_tile_polygon
- bing_tile_quadkey
- bing_tile_zoom_level

- geometry_to_bing_tiles

示例：地理空间查询

本主题中的示例从 GitHub 上提供的示例数据创建两个表，并根据数据查询这两个表。以下文件中的示例数据仅用于说明目的，并不能保证准确性：

- **earthquakes.csv** – 列出了加利福尼亚发生的地震。示例 earthquakes 表使用此数据中的字段。
- **california-counties.json** – 使用符合 ESRI 标准的 GeoJSON 格式列出加利福尼亚州的县级数据。此数据包含多个字段，例如 AREA、PERIMETER、STATE、COUNTY 和 NAME，但示例 counties 表仅使用两个字段：Name（字符串）和 BoundaryShape（二进制）。

> **Note**
>
> Athena 使用 com.esri.json.hadoop.EnclosedEsriJsonInputFormat 将 JSON 数据转换为地理空间二进制格式。

以下代码示例创建一个名为 earthquakes 的表：

```
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/';
```

以下代码示例创建一个名为 counties 的表：

```
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/';
```

以下示例查询在 counties 和 earthquake 表中使用 CROSS JOIN 函数。此示例使用 ST_CONTAINS 来查询边界包含地震位置的县，这些位置由 ST_POINT 指定。然后，查询会按名称对这些县进行分组，然后按计数对其进行排序，并以降序返回它们。

> **Note**
>
> 从 Athena 引擎版本 2 开始，ST_CONTAINS 等函数不再支持将 VARBINARY 类型作为输入。因此，该示例使用 ST_GeomFromLegacyBinary(varbinary) 函数来将 boundaryshape 二进制值转换为几何体。有关更多信息，请参阅 Athena 引擎版本 2 参考中的 地理空间函数的更改。

```
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
```

此查询返回：

```
+-------------------------+
| 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   |
+-----------------------+
```

其他 资源

有关地理空间查询的更多示例，请参阅以下博客文章：

- 在 Amazon Athena 中使用 UDF 和 AWS Lambda 扩展地理空间查询
- 使用 Amazon Athena 和 Amazon QuickSight 可视化超过 200 年的全球气候数据。
- 使用 Amazon Athena 查询 OpenStreetMap

查询 JSON

Amazon Athena 可让您解析 JSON 编码的值，从 JSON 中提取数据，搜索值，以及查找 JSON 数组的长度和大小。

主题

读取 JSON 数据的最佳实践

JavaScript 对象表示法 (JSON) 是将数据结构编码为文本的常用方法。许多应用程序和工具输出 JSON 编码的数据。

在 Amazon Athena 中，您可以从外部数据创建表，并在其中包含 JSON 编码的数据。对于此类类型的源数据，请将 Athena 与 JSON SerDe 库 结合使用。

使用以下提示来读取 JSON 编码的数据：

- 选择正确的 SerDe、本机 JSON SerDe、`org.apache.hive.hcatalog.data.JsonSerDe` 或 OpenX SerDe、`org.openx.data.jsonserde.JsonSerDe`。有关更多信息，请参阅 JSON SerDe 库。
- 确保每个 JSON 编码的记录表示在单独的行上，而不是采用美观的打印格式。

> **Note**
>
> SerDe 期望每个 JSON 文档都位于单行文本中，并且不使用行终止字符分隔记录中的字段。如果 JSON 文本采用美观的打印格式，当您在创建表后尝试对其进行查询时，可能会收到类似以下内容的错误消息：HIVE_CURSOR_ERROR: Row is not a valid JSON Object（HIVE_CURSOR_ERROR：行不是有效的 JSON 对象）或 HIVE_CURSOR_ERROR: JsonParseException: Unexpected end-of-input: expected close marker for OBJECT（HIVE_CURSOR_ERROR：JsonParseException：意外的输入结束：对象的预期关闭标记）。有关更多信息，请参阅 GitHub 上 OpenX SerDe 文档中的 JSON 数据文件。

- 在不区分大小写的列中生成 JSON 编码的数据。
- 提供用于忽略格式错误的记录的选项，如本示例中所示。

```
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/';
```

- 将源数据中具有待定架构的字段转换为 Athena 中的 JSON 编码字符串。

当 Athena 创建由 JSON 数据提供支持的表时，它会基于现有的和预定义的架构解析数据。但并非您的所有数据都可能具有预定义的架构。要在这种情况下简化架构管理，通常有用的做法是将源数据中具有不确定架构的字段转换为 Athena 中的 JSON 字符串，然后使用 JSON SerDe 库。

例如，请考虑一个从不同传感器发布具有公共字段的事件的 IoT 应用程序。其中一个字段必须存储一个对发送事件的传感器唯一的自定义有效负载。在这种情况下，因为您不知道架构，我们建议您将信息存储为 JSON 编码的字符串。为此，请将 Athena 表中的数据转换为 JSON，如下例所示。您还可以将 JSON 编码的数据转换为 Athena 数据类型。

- 将 Athena 数据类型转换为 JSON
- 将 JSON 转换为 Athena 数据类型

将 Athena 数据类型转换为 JSON

要将 Athena 数据转换为 JSON 数据类型，请使用 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
```

此查询返回：

```
+-----------------------------------------+
| hello_msg      | some_int | some_map      |
+-----------------------------------------+
| "HELLO ATHENA" | 12345    | {"a":1,"b":2} |
```

```
+----------------------------------------+
```

将 JSON 转换为 Athena 数据类型

要将 JSON 数据转换为 Athena 数据类型，请使用 CAST。

> **Note**
>
> 在本示例中，为将字符串表示为 JSON 编码，一开始就使用 JSON 关键字并使用单引号，例如 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
```

此查询返回：

```
+-------------------------------------+
| hello_msg    | some_int | some_map  |
+-------------------------------------+
| HELLO ATHENA | 12345    | {a:1,b:2} |
+-------------------------------------+
```

从 JSON 中提取数据

您可能拥有包含 JSON 编码字符串的源数据，您不一定要将其反序列化到 Athena 的表中。在这种情况下，您仍然可以使用 Presto 中提供的 JSON 函数对此数据运行 SQL 操作。

请考虑以下 JSON 字符串作为示例数据集。

```
{"name": "Susan Smith",
"org": "engineering",
"projects":
    [
```

```
     {"name":"project1", "completed":false},
     {"name":"project2", "completed":true}
    ]
}
```

示例：提取属性

要从 JSON 字符串中提取 name 和 projects 属性，请使用 json_extract 函数，如以下示例所示。json_extract 函数将采用包含 JSON 字符串的列，并使用类似 JSONPath 的表达式 (使用点 . 表示法) 搜索它。

> **Note**
>
> JSONPath 执行简单的树遍历。它使用 $ 符号来表示 JSON 文档的根，后跟一个句点和一个直接位于根下面的嵌套元素，例如 $.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
```

返回的值是一个 JSON 编码的字符串，而不是本机 Athena 数据类型。

```
+-------------------------------------------------------------------------------------------------
+
| name           | projects
         |
+-------------------------------------------------------------------------------------------------
+
| "Susan Smith"  | [{"name":"project1","completed":false},
{"name":"project2","completed":true}] |
+-------------------------------------------------------------------------------------------------
+
```

要从 JSON 字符串中提取标量值，请使用 `json_extract_scalar` 函数。它类似于 `json_extract`，但仅返回标量值 (布尔值、数字或字符串)。

> **Note**
>
> 请勿对数组、映射或结构使用 `json_extract_scalar` 函数。

```
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
```

此查询返回：

```
+---------------------------+
| name           | projects |
+---------------------------+
| Susan Smith    |          |
+---------------------------+
```

要获取示例数组中的 `projects` 属性的第一个元素，请使用 `json_array_get` 函数并指定索引位置。

```
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
```

它返回 JSON 编码数组中指定索引位置处的值。

```
+---------------------------------------+
| item                                  |
+---------------------------------------+
| {"name":"project1","completed":false} |
+---------------------------------------+
```

要返回一个 Athena 字符串类型，请在 JSONPath 表达式中使用 [] 运算符，然后使用 json_extract_scalar 函数。有关 [] 的更多信息，请参阅 访问数组元素。

```
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
```

它将返回此结果：

```
+--------------+
| project_name |
+--------------+
| project1     |
+--------------+
```

在 JSON 数组中搜索值

要确定特定值是否存在于 JSON 编码的数组内，请使用 json_array_contains 函数。

以下查询列出参加“project2”的用户的名称。

```
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')
```

此查询会返回一个用户列表。

```
+-------------+
| user        |
+-------------+
| Susan Smith |
+-------------+
| Jane Smith  |
+-------------+
```

以下查询示例列出了完成项目的用户的名称以及已完成的项目的总数。它执行以下操作：

- 为清晰起见，使用嵌套 SELECT 语句。
- 提取项目的数组。
- 使用 CAST 将数组转换为键-值对的本机数组。
- 使用 UNNEST 运算符提取每个数组元素。
- 按完成项目来筛选获取的值，并对其进行计数。

> **Note**
>
> 当使用 CAST 到 MAP 时，您可以将键元素指定为 VARCHAR（Presto 中的本机字符串），但将值保留为 JSON，因为 MAP 中的值具有不同类型：第一个键-值对为字符串，第二个键-值对为布尔型。

```
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
```

此查询返回以下结果：

```
+----------------------------------+
| name        | completed_projects |
+----------------------------------+
| Susan Smith | 2                  |
+----------------------------------+
| Jane Smith  | 1                  |
+----------------------------------+
```

获取 JSON 数组的长度和大小

例如：**json_array_length**

要获取 JSON 编码数组的长度，请使用 `json_array_length` 函数。

```
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
```

此查询返回以下结果：

```
+----------------------+
| name        | count |
+----------------------+
| Susan Smith | 2     |
+----------------------+
| Bob Smith   | 1     |
+----------------------+
| Jane Smith  | 1     |
+----------------------+
```

例如：**json_size**

要获取 JSON 编码的数组或对象的大小，请使用 `json_size` 函数，并指定将 JSON 字符串和 JSONPath 表达式包含到该数组或对象中的列。

```
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
```

此查询返回以下结果：

```
+---------------------+
| name        | count |
+---------------------+
| Susan Smith | 2     |
+---------------------+
| Bob Smith   | 1     |
+---------------------+
| Jane Smith  | 1     |
+---------------------+
```

JSON 查询问题排查

有关排查 JSON 相关查询的问题的帮助，请参阅 JSON 相关错误 或咨询以下资源：

- 我在 Amazon Athena 中尝试读取 JSON 数据时收到错误
- 从 Athena 中的 AWS Config 读取文件时，如何解决“HIVE_CURSOR_ERROR: Row is not a valid JSON Object – JSONException: Duplicate key（HIVE_CURSOR_ERROR：行是无效的 JSON 对象 – JSONException：重复键）”错误？
- 即使输入 JSON 文件有多条记录，Amazon Athena 中的 SELECT COUNT 查询也仅返回一条记录
- 如何在 Amazon S3 源文件中查找 Athena 表中的某行？

另请参阅 Amazon Athena 中 SQL 查询的注意事项和限制。

在 Amazon Athena 中使用机器学习（ML）

将机器学习（ML）与 Amazon Athena 结合使用可让您借助 Athena 编写使用 Amazon SageMaker 运行机器学习（ML）推理的 SQL 语句。此功能简化了对 ML 模型的访问以进行数据分析，无需使用复杂的编程方法来运行推理。

要将机器学习（ML）与 Athena 结合使用，您可以使用 `USING EXTERNAL FUNCTION` 子句定义将机器学习（ML）与 Athena 结合的函数。该函数指向要使用的 SageMaker 模型端点，并指定要传递

给模型的变量名称和数据类型。查询中的后续子句引用该函数以将值传递给模型。模型根据查询传递的值运行推理，然后返回推理结果。有关 SageMaker 以及 SageMaker 端点工作原理的信息，请参阅 Amazon SageMaker 开发人员指南。

有关将机器学习（ML）与 Athena 和 SageMaker 推理结果结合使用以检测结果集中异常值的示例，请参阅 AWS 大数据博客文章：通过调用 Amazon Athena 机器学习推理函数来检测异常值。

注意事项和限制

- 可用区域 – Athena ML 功能在支持 Athena 引擎版本 2 或更高版本的 AWS 区域 中可用。
- SageMaker 模型端点必须接受并返回 **`text/csv`** – 有关数据格式的更多信息，请参阅《Amazon SageMaker 开发人员指南》中的 用于推理的常见数据格式。
- Athena 不发送 CSV 标头 - 如果您的 SageMaker 端点是 `text/csv`，则您的输入处理程序不应假设输入的第一行是 CSV 标头。由于 Athena 不发送 CSV 标头，因此返回给 Athena 的输出将比 Athena 预期的少一行，从而导致错误。
- SageMaker 端点扩展 – 确保引用的 SageMaker 模型端点已充分扩展，以便 Athena 调用端点。有关更多信息，请参阅《Amazon SageMaker 开发人员指南》中的 自动扩展 SageMaker 模型 和《Amazon SageMaker API 参考》中的 CreateEndpointConfig。
- IAM 权限 – 要运行指定机器学习（ML）与 Athena 相结合的函数的查询，必须允许运行该查询的 IAM 委托人，以便为引用的 SageMaker 模型端点执行 `sagemaker:InvokeEndpoint` 操作。有关更多信息，请参阅 允许使用 Athena 访问机器学习。
- 机器学习与 Athena 相结合的函数不能直接在 **`GROUP BY`** 子句中使用

机器学习与 Athena 相结合的语法

`USING EXTERNAL FUNCTION` 子句指定可由查询中的后续 `SELECT` 语句引用的机器学习（ML）与 Athena 相结合的函数或多个函数。您定义函数名称、变量名称以及变量和返回值的数据类型。

摘要

下面的语法显示了 `USING EXTERNAL FUNCTION` 子句，该子句指定了机器学习（ML）与 Athena 相结合的函数。

```
USING EXTERNAL FUNCTION ml_function_name (variable1 data_type[, variable2 data_type]
[,...])
RETURNS data_type
SAGEMAKER 'sagemaker_endpoint'
```

```
SELECT ml_function_name()
```

参数

USING EXTERNAL FUNCTION ***ml_function_name*** (***variable1data_type***[, ***variable2data_type***][,...])

ml_function_name 定义函数名称，可以在后续查询子句中使用该函数名称。每个 *variable data_type* 指定一个命名变量，且其相应的数据类型应可为 SageMaker 模型接受作为输入。指定的数据类型必须是支持的 Athena 数据类型。

RETURNS ***data_type***

data_type 指定作为 SageMaker 模型的输出由 *ml_function_name* 返回到查询的 SQL 数据类型。

SAGEMAKER '***sagemaker_endpoint***'

sagemaker_endpoint 指定 SageMaker 模型的端点。

SELECT [...]***ml_function_name***(***expression***) [...]

将值传递给函数变量和 SageMaker 模型以返回结果的 SELECT 查询。*ml_function_name* 指定之前在查询中定义的函数，后跟进行计算以传递值的*###*。传递和返回的值必须与 USING EXTERNAL FUNCTION 子句中为函数指定的相应数据类型匹配。

示例

以下示例演示了使用机器学习（ML）与 Athena 结合的查询。

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;
```

客户使用示例

以下视频使用了机器学习（ML）与 Amazon Athena 相结合函数的预览版，展示了您可以在 Athena 中使用 SageMaker 的方式。

预测客户流失

以下视频展示了如何将 Athena 与 Amazon SageMaker 的机器学习功能相结合，以预测客户流失情况。

使用 Amazon Athena 和 Amazon SageMaker 预测客户流失

检测僵尸网络

以下视频展示了一家公司如何使用 Amazon Athena 和 Amazon SageMaker 来检测僵尸网络。

使用 Amazon Athena 和 Amazon SageMaker 检测僵尸网络

使用用户定义函数进行查询

Amazon Athena 中的用户定义函数 (UDF) 使您可以创建自定义函数来处理记录或记录组。UDF 接受参数，执行工作，然后返回结果。

要在 Athena 中使用 UDF，请在 SQL 查询中的 `SELECT` 语句之前写入 `USING EXTERNAL FUNCTION` 子句。`SELECT` 语句引用 UDF 并定义在查询运行时传递给 UDF 的变量。SQL 查询在调用 UDF 时使用 Java 运行时调用 Lambda 函数。UDF 在 Lambda 函数中定义为 Java 部署包中的方法。可以在同一个 Java 部署包中为某个 Lambda 函数定义多个 UDF。您还可以在 `USING EXTERNAL FUNCTION` 子句中指定 Lambda 函数的名称。

部署 Athena UDF 的 Lambda 函数有两个选项。您可以直接使用 Lambda 部署函数，也可以使用 AWS Serverless Application Repository 来部署。要查找 UDF 的现有 Lambda 函数，您可以搜索公共 AWS Serverless Application Repository 或私有存储库，然后部署到 Lambda。您还可以创建或修改 Java 源代码，将其打包到 JAR 文件中，然后使用 Lambda 或 AWS Serverless Application Repository 来部署它。有关能够帮助您开始使用的示例 Java 源代码和软件包，请参阅 使用 Lambda 创建和部署 UDF。有关 Lambda 的更多信息，请参阅《AWS Lambda 开发人员指南》。有关 AWS Serverless Application Repository 的更多信息，请参阅《AWS Serverless Application Repository 开发人员指南》。

有关使用 UDF 与 Athena 一起翻译和分析文本的示例，请参阅 AWS Machine Learning 博客文章：使用 Amazon Athena、Amazon Translate 和 Amazon Comprehend 的 SQL 函数翻译和分析文本，或观看 video。

有关在 Amazon Athena 中使用 UDF 扩展地理空间查询的示例，请参阅 AWS 大数据博客中的使用 UDF 和 AWS Lambda 在 Amazon Athena 中扩展地理空间查询。

注意事项和限制

- 内置 Athena 函数 – Athena 中的内置函数旨在实现高性能。我们建议尽可能使用内置函数而不是 UDF。有关内置函数的更多信息，请参阅 Amazon Athena 中的函数。
- 仅标量 UDF – Athena 仅支持标量 UDF，它将一次处理一行并返回单个列值。每次调用 Lambda 时，Athena 都会将行批量（可能并行）传递给 UDF。在设计 UDF 和查询时，请注意此处理可能对网络流量产生的潜在影响。
- UDF 处理函数使用缩写格式– 对于 UDF 函数使用缩写格式（而非完整格式）（例如，使用 `package.Class` 而非 `package.Class::method`）。
- UDF 方法必须为小写– UDF 方法必须为小写字母；不允许使用骆驼大小写。
- UDF 方法需要参数 - UDF 方法必须至少有一个输入参数。尝试调用未定义输入参数的 UDF 会导致运行时异常。UDF 旨在对数据记录执行函数，但是没有参数的 UDF 不接受任何数据，因此会出现异常。
- Java 运行时支持 – 目前，Athena UDF 支持用于 Lambda 的 Java 8 和 Java 11 运行时。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的使用 Jav 构建 Lambda 函数。
- IAM 权限 – 要在 Athena 中创建并运行 UDF 查询语句，则除 Athena 函数之外，还必须允许运行查询的 IAM 委托人执行操作。有关更多信息，请参阅 允许 Amazon Athena 用户定义函数（UDF）的 IAM 权限策略示例。
- Lambda 配额 – Lambda 配额适用于 UDF。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。
- 行级别筛选 - UDF 不支持 Lake Formation 行级别筛选。
- 视图 – 您不能将视图与 UDF 同时使用。
- 已知问题 – 有关已知问题的最新列表，请参阅 GitHub 的 awslabs/aws-athena-query-federation 部分中的 Limitations and issues（限制和问题）。

视频

观看以下视频，了解有关在 Athena 中使用 UDF 的更多信息。

视频：在 Amazon Athena 中引入用户定义函数 (UDF)

以下视频展示了如何在 Amazon Athena 中使用 UDF 来编辑敏感信息。

> **Note**
>
> 本视频中的语法是预发行的，但概念是相同的。在没有 `AmazonAthenaPreviewFunctionality` 工作组的情况下使用 Athena。

[在 Amazon Athena 中引入用户定义函数（UDF）]

视频：使用 Amazon Athena 中的 SQL 查询翻译、分析和编辑文本字段

以下视频展示了如何在 Amazon Athena 中使用 UDF 与其他 AWS 服务 相结合来翻译和分析文本。

> **Note**
>
> 本视频中的语法是预发行的，但概念是相同的。有关正确的语法，请参阅相关博客文章：AWS Machine Learning 博客中的使用 Amazon Athena、Amazon Translate 和 Amazon Comprehend 的 SQL 函数翻译、编辑和分析文本。

[使用 Amazon Athena 中的 SQL 查询翻译、分析和编辑文本字段]

UDF 查询语法

`USING EXTERNAL FUNCTION` 子句指定可由查询中的后续 `SELECT` 语句引用的 UDF 或多个 UDF。您需要 UDF 的方法名称和托管 UDF 的 Lambda 函数的名称。您可以使用 Lambda ARN 代替 Lambda 函数名称。在跨账户场景中，需要提供 Lambda ARN。

摘要

```
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)] [...]
```

参数

USING EXTERNAL FUNCTION *UDF_name*(*variable1data_type*[, *variable2data_type*][,...])

UDF_name 指定 UDF 的名称，该名称必须与引用的 Lambda 函数中的 Java 方法对应。每个 *variable data_type* 指定一个命名变量，且其相应的数据类型应可为 UDF 接受作为输入。*data_type* 必须是下表中列出的受支持 Athena 数据类型之一，并映射到相应的 Java 数据类型。

Athena 数据类型	Java 数据类型
TIMESTAMP	java.time.LocalDateTime (UTC)
DATE	java.time.LocalDate (UTC)
TINYINT	java.lang.Byte
SMALLINT	java.lang.Short
REAL	java.lang.Float
DOUBLE	java.lang.Double
DECIMAL（请参阅 RETURNS 注释）	java.math.BigDecimal
BIGINT	java.lang.Long
INTEGER	java.lang.Int
VARCHAR	java.lang.String
VARBINARY	byte[]
BOOLEAN	java.lang.Boolean
ARRAY	java.util.List
ROW	java.util.Map<String, Object>

RETURNS ***data_type***

data_type 指定 UDF 作为输出返回的 SQL 数据类型。上表中列出的 Athena 数据类型均受支持。对于 DECIMAL 数据类型，请使用语法 RETURNS DECIMAL(*precision*, *scale*)，其中 *precision*（精度）和 *scale*（小数位数）是整数。

LAMBDA '***lambda_function***'

lambda_function 指定运行 UDF 时要调用的 Lambda 函数的名称。

SELECT [...]***UDF_name***(***expression***) [...]

SELECT 查询会将值传递给 UDF 并返回结果。*UDF_name* 指定要使用的 UDF，后跟一个*###*，该表达式将进行估算以传递值。传递和返回的值必须与 USING EXTERNAL FUNCTION 子句中为 UDF 指定的相应数据类型匹配。

示例

如需获取 GitHub 上基于AthenaUDFHandler.java 代码的示例查询，请参阅 GitHub 的 Amazon Athena UDF connector（Amazon Athena UDF 连接器）页面。

使用 Lambda 创建和部署 UDF

要创建自定义 UDF，可以通过扩展 UserDefinedFunctionHandler 类来创建新的 Java 类。开发工具包中的 UserDefinedFunctionHandler.java 的源代码在 GitHub 上的 awslabs/aws-athena-query-federation/athena-federation-sdk 存储库中提供，其中还提供有示例 UDF 实施，您可以对其进行检查和修改，以创建自定义 UDF。

本节中的步骤演示从命令行使用 Apache Maven 编写和构建自定义 UDF Jar 文件和部署。

使用 Maven 为 Athena 创建自定义 UDF 的步骤

- 克隆开发工具包并准备开发环境
- 创建您的 Maven 项目
- 将依赖项和插件添加到您的 Maven 项目中
- 为 UDF 编写 Java 代码
- 构建 JAR 文件
- 将 JAR 部署到 AWS Lambda

克隆开发工具包并准备开发环境

在开始之前，请确保使用 `sudo yum install git -y` 在您的系统上安装 git。

安装 AWS 查询联合开发工具包

- 在命令行输入以下内容以克隆软件开发工具包存储库。此存储库包括软件开发工具包、示例和一套数据源连接器。有关数据源连接器的更多信息，请参阅使用 Amazon Athena 联合查询。

```
git clone https://github.com/awslabs/aws-athena-query-federation.git
```

此程序的安装先决条件

如果您正在使用已安装 Apache Maven、AWS CLI 和 AWS Serverless Application Model 构建工具的开发计算机，则可以跳过此步骤。

1. 从克隆时创建的 `aws-athena-query-federation` 目录的根目录下，运行准备开发环境的 prepare_dev_env.sh 脚本。
2. 更新您的 shell 以获取在安装过程中创建的新变量或重新启动终端会话。

```
source ~/.profile
```

> **Important**
>
> 如果您跳过此步骤，稍后将收到关于 AWS CLI 或 AWS SAM 构建工具无法发布 Lambda 函数的错误。

创建您的 Maven 项目

运行以下命令以创建您的 Maven 项目。将 *groupId* 替换为组织的唯一 ID，并将 *my-athena-udf* 替换为您的应用程序的名称。有关详细信息，请参阅 Apache Maven 文档中的如何创建我的第一个 Maven 项目？。

```
mvn -B archetype:generate \
-DarchetypeGroupId=org.apache.maven.archetypes \
-DgroupId=groupId \
-DartifactId=my-athena-udfs
```

将依赖项和插件添加到您的 Maven 项目中

将以下配置添加到 Maven 项目 pom.xml 文件中。要查看示例，请参阅 GitHub 中的 pom.xml 文件。

```
<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>
```

为 UDF 编写 Java 代码

通过扩展 UserDefinedFunctionHandler.java 来创建一个新类。在类中编写您的 UDF。

在以下示例中，已在 MyUserDefinedFunctions 类中创建了两个用于 UDF、compress() 和 decompress() 的 Java 方法。

```
*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);
    }
}
```

构建 JAR 文件

运行 `mvn clean install` 以构建您的项目。成功构建后，将在名为 *`artifactId`*`-`*`version`*`.jar` 的项目的 `target` 文件夹中创建一个 JAR 文件，其中 *`artifactId`* 是您在 Maven 项目中提供的名称，例如 `my-athena-udfs`。

将 JAR 部署到 AWS Lambda

有两个选项可以将代码部署到 Lambda：

- 使用 AWS Serverless Application Repository 部署（推荐）
- 从 JAR 文件创建 Lambda 函数

选项 1：部署到 AWS Serverless Application Repository

将 JAR 文件部署到 AWS Serverless Application Repository 时，您将创建一个代表应用程序体系结构的 AWS SAM 模板 YAML 文件。然后，您可以指定此 YAML 文件和一个 Amazon S3 存储桶，在其中上载应用程序的构件并使其对 AWS Serverless Application Repository 可用。下面的过程使用 publish.sh 脚本，该脚本位于您之前克隆的 Athena Query Federation SDK 的 `athena-query-federation/tools` 目录中。

有关更多信息和要求，请参阅《AWS Serverless Application Repository 开发人员指南》中的 发布应用程序、《AWS Serverless Application Model 开发人员指南》中的 AWS SAM 模板概念和 使用 AWS SAM CLI 发布无服务器应用程序。

以下示例演示了 YAML 文件中的参数。将类似的参数添加到 YAML 文件并将其保存在项目目录中。有关完整示例，请参阅 GitHub 中的 athena-udf.yaml。

```
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
```

将 publish.sh 脚本复制到保存 YAML 文件的项目目录中，然后运行以下命令：

```
./publish.sh MyS3Location MyYamlFile
```

例如，如果您的存储桶位置为 s3://DOC-EXAMPLE-BUCKET/mysarapps/athenaudf 并且您的 YAML 文件已另存为 my-athena-udfs.yaml：

```
./publish.sh DOC-EXAMPLE-BUCKET/mysarapps/athenaudf my-athena-udfs
```

创建 Lambda 函数

1. 在以下位置打开 Lambda 控制台：https://console.aws.amazon.com/lambda/，选择 Create function（创建函数），然后选择 Browse serverless app repository（浏览无服务器应用程序存储库）
2. 选择 Private applications (私有应用程序)，在列表中找到您的应用程序，或使用关键词搜索它，然后选择它。
3. 查看并提供应用程序详细信息，然后选择 Deploy (部署)。

   您现在可以使用在 Lambda 函数 JAR 文件中定义的方法名称作为 Athena 中的 UDF。

选项 2：直接创建 Lambda 函数

您也可以直接使用控制台或 AWS CLI 创建 Lambda 函数。以下示例演示如何使用 create-function Lambda CLI 命令。

```
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
```

跨区域查询

Athena 支持在 AWS 区域 中查询 Amazon S3 数据，此区域与您使用 Athena 的区域不同。若移动数据不可行或没有权限时，或者您想跨多个区域查询数据，则可以选择跨区域查询。即使 Athena 在特定区域不可用，也可以从另一个可使用 Athena 的区域查询该地区的数据。

若要查询某个区域中的数据，即使 Amazon S3 数据不属于您的账户，也必须在该区域启用您的账户。对于某些区域 [例如美国东部（俄亥俄）]，创建账户时会自动启用您对该区域的访问权限。其他区域要求您的账户在使用之前必须“选择加入”该区域。有关要求选择加入的区域列表，请参阅《适用于 Linux 实例的 Amazon EC2 用户指南》中的可用区域。有关选择加入区域的特定说明，请参阅 Amazon Web Services 一般参考 中的管理 AWS 区域。

注意事项和限制

- 数据访问权限 – 若要跨地区成功查询 Athena 的 Amazon S3 数据，您的账户必须具有读取数据的权限。若您要查询的数据属于另一个账户，则此其他账户必须授予您对包含该数据的 Amazon S3 位置的访问权限。
- 数据传输费 – 跨区域查询可能会收取 Amazon S3 数据传输费。运行查询可能导致传输的数据超过数据集的大小。我们建议您首先测试对数据子集的查询，然后查看 AWS Cost Explorer 中的成本。
- AWS Glue – 您可以跨区域使用 AWS Glue。跨区域 AWS Glue 流量可能收取额外费用。有关更多信息，请参阅 AWS Glue 大数据博客中的创建跨账户和跨区域 AWS 连接。
- Amazon S3 加密选项 – SSE-S3 和 SSE-KMS 加密选项支持跨区域查询；CSE-KMS 不支持跨区域查询。有关更多信息，请参阅 支持的 Amazon S3 加密选项。
- 联合查询 - 不支持跨 AWS 区域 使用联合查询。
- 中国区域 – 中国区域不支持跨区域查询。

只要满足上述条件，您就可以创建一个指向自己指定的 `LOCATION` 值的 Athena 表，并透明地查询数据。不需要特殊语法。有关创建 Athena 表的信息，请参阅在 Athena 中创建表。

查询 AWS Glue Data Catalog

由于许多 AWS 服务 都将 AWS Glue Data Catalog 用作其中央元数据存储库，因此您可能需要查询 Data Catalog 元数据。为此，您可以在 Athena 中使用 SQL 查询。您可以使用 Athena 查询 AWS Glue 目录元数据，如数据库、表、分区和列等。

要获取 AWS Glue Catalog 元数据，请查询 Athena 后端上的 `information_schema` 数据库。本主题中的示例查询显示如何使用 Athena 查询 AWS Glue Catalog 元数据以搜索常用案例。

主题

- 列出特定表的分区
- 列出所有表的所有列
- 列出特定的表的共同列
- 列出或搜索指定表或视图的列

注意事项和限制

- 您可以不查询 `information_schema` 数据库，而是使用单独的 Apache Hive DDL 命令从 Athena 中提取特定数据库、表、视图、分区和列的元数据信息。但输出将为非表格格式。
- 如果您的 AWS Glue 元数据较少或中等，则查询 `information_schema` 的性能会最佳。如果您有大量的元数据，则可能会出现错误。
- 不能使用 `CREATE VIEW` 在 `information_schema` 数据库上创建视图。

列出数据库并搜索指定的数据库

本节中的示例演示如何按模式名称列出元数据中的数据库。

Example – 列出数据库

以下示例查询列出 `information_schema.schemata` 表中的数据库。

```
SELECT schema_name
FROM   information_schema.schemata
LIMIT  10;
```

下表显示了示例结果。

6	alb-databas1
7	alb_original_cust
8	alblogsdatabase
9	athena_db_test
10	athena_ddl_db

Example – 搜索指定的数据库

在以下示例查询中，rdspostgresql 是示例数据库。

```
SELECT schema_name
FROM   information_schema.schemata
WHERE  schema_name = 'rdspostgresql'
```

下表显示了示例结果。

	schema_name
1	rdspostgresql

列出指定数据库中的表并按名称搜索表

要列出表的元数据，您可以按表架构或表名进行查询。

Example – 按架构列出表

以下查询列出了使用 rdspostgresql 表架构的表。

```
SELECT table_schema,
       table_name,
       table_type
FROM   information_schema.tables
WHERE  table_schema = 'rdspostgresql'
```

下图显示了一个示例结果。

	table_schema	table_name	table_type
1	rdspostgresql	rdspostgresqldb1_public_account	BASE TABLE

Example – 按名称搜索表

以下查询获取表 athena1 的元数据信息。

```
SELECT table_schema,
       table_name,
       table_type
FROM   information_schema.tables
WHERE  table_name = 'athena1'
```

下图显示了一个示例结果。

	table_schema	table_name	table_type
1	默认值	athena1	BASE TABLE

列出特定表的分区

您可以使用 SHOW PARTITIONS *table_name* 以列出指定表的分区，如以下示例所示。

```
SHOW PARTITIONS cloudtrail_logs_test2
```

您还可以使用 $partitions 元数据查询列出特定表的分区号和分区值。

Example – 使用 $partitions 语法查询表的分区

以下示例查询使用 $partitions 语法列出了表 cloudtrail_logs_test2 的分区。

```
SELECT * FROM default."cloudtrail_logs_test2$partitions" ORDER BY partition_number
```

下表显示了示例结果。

	table_catalog	table_schema	table_name	年	月	天
1	awsdatacatalog	default	cloudtrail_logs_test2	2020	08	10
2	awsdatacatalog	default	cloudtrail_logs_test2	2020	08	11

	table_catalog	table_schema	table_name	年	月	天
3	awsdatacatalog	default	cloudtrail_logs_test2	2020	08	12

列出所有表的所有列

您可以列出 AwsDataCatalog 中所有表的所有列，或者 AwsDataCatalog 中特定数据库中所有表的所有列。

- 要列出 AwsDataCatalog 中所有数据库的所有列，请使用查询 SELECT * FROM information_schema.columns。
- 要将结果限制在特定数据库中，请使用 WHERE 子句中的 table_schema='*database_name*'。

Example — 列出特定数据库中所有表的所有列

以下示例查询列出了数据库 webdata 中所有表的所有列。

```
SELECT * FROM information_schema.columns WHERE table_schema = 'webdata'
```

列出特定的表的共同列

可以列出数据库中的特定的表的共同列。

- 使用语法 SELECT column_name FROM information_schema.columns。
- 对于 WHERE 子句，请使用语法 WHERE table_name IN ('table1', 'table2')。

Example – 列出同一个数据库中的两个表的共同列

如下示例查询列出了表 table1 和表 table2 的共同列。

```
SELECT column_name
FROM information_schema.columns
WHERE table_name IN ('table1', 'table2')
GROUP BY column_name
HAVING COUNT(*) > 1;
```

列出或搜索指定表或视图的列

您可以列出表的所有列、视图的所有列，或者在指定的数据库和表中按名称搜索列。

要列出列，请使用 SELECT * 查询。在 FROM 子句中，指定 information_schema.columns。在 WHERE 子句中，使用 table_schema='*database_name*' 以指定数据库，使用 table_name = '*table_name*' 指定包含要列出的列的表或视图。

Example – 列出指定表的所有列

以下示例查询列出了 rdspostgresqldb1_public_account 表的所有列。

```
SELECT *
FROM   information_schema.columns
WHERE  table_schema = 'rdspostgresql'
       AND table_name = 'rdspostgresqldb1_public_account'
```

下表显示了示例结果。

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_t	comm	extra_info
1	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	passwor	1		是	varcha		
2	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	user_id	2		是	整数		
3	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	created_n	3		是	时间戳		
4	awsdatacatalog	rdspostgresql	rdspostgresqldb1_p	last_login	4		是	时间戳		

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_type	comment	extra_info
			ublic_account							
5	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	email	5		是	varcha		
6	awsdatacatalog	rdspostgresql	rdspostgresqldb1_public_account	usernam	6		是	varcha		

Example – 列出指定视图的列

以下示例查询列出了 `default` 数据库中用于 `arrayview` 视图的所有列。

```
SELECT *
FROM   information_schema.columns
WHERE  table_schema = 'default'
       AND table_name = 'arrayview'
```

下表显示了示例结果。

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_type	comment	extra_info
1	awsdatacatalog	default	arrayvie	searchdate	1		是	varchar		
2	awsdatacatalog	default	arrayvie	sid	2		是	varchar		
3	awsdatacatalog	default	arrayvie	btid	3		是	varchar		

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_type	comment	extra_info
4	awsdatacatalog	default	arrayview	p	4		是	varchar		
5	awsdatacatalog	default	arrayview	infantprice	5		是	varchar		
6	awsdatacatalog	default	arrayview	sump	6		是	varchar		
7	awsdatacatalog	default	arrayview	journeymaparray	7		是	array(varchar)		

Example – 在指定数据库和表中按名称搜索列

以下示例查询在 `default` 数据库的 `arrayview` 视图中搜索 `sid` 列的元数据。

```
SELECT *
FROM   information_schema.columns
WHERE  table_schema = 'default'
       AND table_name = 'arrayview'
       AND column_name='sid'
```

下图显示了一个示例结果。

	table_catalog	table_schema	table_name	column_name	ordinal_position	column_default	is_nullable	data_type	comment	extra_info
1	awsdatacatalog	default	arrayview	sid	2		是	varchar		

查询 AWS 服务 日志

本部分包含使用 Amazon Athena 查询常见数据集的几个过程，例如 AWS CloudTrail 日志、Amazon CloudFront 日志、经典负载均衡器日志、Application Load Balancer 日志、Amazon VPC 流日志和网络负载均衡器日志。

本节中的任务使用 Athena 控制台，但您也可以使用其他工具，例如 Athena JDBC 驱动程序、AWS CLI 或 Amazon Athena API 参考。

有关使用 AWS CloudFormation 自动创建 AWS 服务 日志表、分区和 Athena 中示例查询的信息，请参阅 AWS 大数据博客中的《使用 Amazon Athena 实现自动化 AWS 服务 日志表创建并使用 Amazon Athena 进行查询》。要了解如何将 Python 库用于 AWS Glue 以创建一个用于处理 AWS 服务 日志的常用框架，并在 Athena 中查询它们，请参阅 使用 Amazon Athena 轻松查询 AWS 服务 日志。

本节中的主题假定您已配置适当权限来访问 Athena 和要查询的数据所在的 Amazon S3 存储桶。有关更多信息，请参阅 设置和 开始使用。

主题

查询 Application Load Balancer 日志

Application Load Balancer 是 Elastic Load Balancing 的负载均衡选项，它允许使用容器在微服务部署中实现流量分配。通过查询 Application Load Balancer 日志，您可以查看进出 Elastic Load Balancing 实例和后端应用程序的流量来源、延迟和传输字节。有关更多信息，请参阅《User Guide for Application Load Balancers》中的 Access logs for your Application Load Balancer 和 Connection logs for your Application Load Balancer。

主题

先决条件

- 启用访问日志记录功能或连接日志记录功能，以便将应用程序负载均衡器日志保存到 Amazon S3 存储桶。
- 用于保存您将为 Athena 创建的表的数据库。要创建数据库，你可以使用 Athena 或 AWS Glue 控制台。有关更多信息，请参阅本指南中的 在 Athena 中创建数据库 或《AWS Glue 开发人员指南》中的 通过 AWS Glue 控制台使用数据库。

为 ALB 访问日志创建表

1. 在 Athena 控制台中将以下 CREATE TABLE 语句复制并粘贴到查询编辑器中。有关 Amazon 控制台入门的更多信息，请参阅 开始使用。将 LOCATION 子句中的路径替换为 Amazon S3 访问日志文件夹位置。有关访问日志文件位置的更多信息，请参阅《User Guide for Application Load Balancers》中的 Access log files。有关每个日志文件字段的信息，请参阅 Access log entries。

> **Note**
>
> 以下 CREATE TABLE 语句包含最近添加的 classification、classification_reason 和 traceability_id 列。要为不包含这些条目的应用程序负载均衡器访问日志创建表，请从 CREATE TABLE 语句中删除相应的列，并相应地修改正则表达式。

```
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. 在 Athena 控制台中运行查询。查询完成后，Athena 将注册 `alb_access_logs` 表，使其中的数据可以供您发出查询。

使用分区投影功能在 Athena 中为 ALB 访问日志创建表

由于 ALB 访问日志具有一个可以预先指定其分区方案的已知结构，所以您可以使用 Athena 分区投影功能，以此来减少查询运行时间并自动管理分区。当添加新数据时，分区投影会自动添加新分区。这样就不必使用 `ALTER TABLE ADD PARTITION` 手动添加分区了。

从指定日期开始到当前日期为止，以下示例 `CREATE TABLE` 语句会自动在 ALB 访问日志上为单个 AWS 区域使用分区投影。该语句以上一部分中的示例为基础，但添加了 `PARTITIONED BY` 和 `TBLPROPERTIES` 子句以启用分区投影。在 `LOCATION` 和 `storage.location.template` 子句中，将占位符替换为标识 ALB 访问日志在 Amazon S3 存储桶中位置的值。有关访问日志文件位置的更多信息，请参阅《User Guide for Application Load Balancers》中的 Access log files。对于 `projection.day.range`，将 *2022/01/01* 替换为要使用的开始日期。成功运行查询后，您可以查询表。您无需运行 `ALTER TABLE ADD PARTITION` 来加载分区。有关每个日志文件字段的信息，请参阅 Access log entries。

```
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}"
            )
```

更多有关分区投影的信息，请参阅 使用 Amazon Athena 分区投影。

ALB 访问日志的示例查询

以下查询计算按客户端 IP 地址分组的负载均衡器接收到的 HTTP GET 请求的数目：

```
SELECT COUNT(request_verb) AS
 count,
 request_verb,
 client_ip
FROM alb_logs
GROUP BY request_verb, client_ip
LIMIT 100;
```

另一个查询显示 Safari 浏览器用户访问的 URL：

```
SELECT request_url
FROM alb_logs
WHERE user_agent LIKE '%Safari%'
LIMIT 10;
```

以下查询显示了 ELB 状态代码值大于或等于 500 的记录。

```
SELECT * FROM alb_logs
WHERE elb_status_code >= 500
```

以下示例演示了如何根据 datetime 解析日志：

```
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;
```

以下查询会查询在指定日期内对所有 ALB 日志使用分区投影的表。

```
SELECT *
FROM alb_logs
WHERE day = '2022/02/12'
```

为 ALB 连接日志创建表

1. 在 Athena 控制台中将以下 CREATE TABLE 语句复制并粘贴到查询编辑器中。有关 Amazon 控制台入门的更多信息，请参阅 开始使用。将 LOCATION 子句中的路径替换为 Amazon S3 连接日志文件夹位置。有关连接日志文件位置的更多信息，请参阅《User Guide for Application Load Balancers》中的 Connection log files。有关每个日志文件字段的信息，请参阅 Connection log entries。

```
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. 在 Athena 控制台中运行查询。查询完成后，Athena 将注册 alb_connection_logs 表，使其中的数据可以供您发出查询。

使用分区投影功能在 Athena 中为 ALB 连接日志创建表

由于 ALB 日志具有一个可以预先指定其分区方案的已知结构，您可以使用 Athena 分区投影功能，以此来减少查询运行时间并自动管理分区。当添加新数据时，分区投影会自动添加新分区。这样就不必使用 ALTER TABLE ADD PARTITION 手动添加分区了。

从指定日期开始到当前日期为止，以下示例 CREATE TABLE 语句会自动在 ALB 连接日志上为单个 AWS 区域使用分区投影。该语句以上一部分中的示例为基础，但添加了 PARTITIONED BY 和

TBLPROPERTIES 子句以启用分区投影。在 LOCATION 和 storage.location.template 子句中，将占位符替换为标识 ALB 连接日志在 Amazon S3 存储桶中位置的值。有关连接日志文件位置的更多信息，请参阅《User Guide for Application Load Balancers》中的 Connection log files。对于 projection.day.range，将 *2023/01/01* 替换为要使用的开始日期。成功运行查询后，您可以查询表。您无需运行 ALTER TABLE ADD PARTITION 来加载分区。有关每个日志文件字段的信息，请参阅 Connection log entries。

```
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}"
```

```
            )
```

更多有关分区投影的信息，请参阅 使用 Amazon Athena 分区投影。

ALB 连接日志的示例查询

以下查询对 `tls_verify_status` 值不为 `'Success'` 的情况进行计数，结果按客户端 IP 地址分组：

```
SELECT DISTINCT client_ip, count() AS count FROM alb_connection_logs
WHERE tls_verify_status != 'Success'
GROUP BY client_ip
ORDER BY count() DESC;
```

以下查询搜索指定时间范围内 `tls_handshake_latency` 值超过 2 秒的情况：

```
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);
```

其他 资源

- AWS 知识中心中的如何使用 Amazon Athena 分析应用程序负载均衡器访问日志。
- 有关弹性负载均衡 HTTP 状态代码的信息，请参阅《应用程序负载均衡器用户指南》中的对应用程序负载均衡器进行问题排查。
- 在 AWS 大数据博客中的使用 AWS Glue 自定义分类器和 Amazon Athena 更有效地编目和分析应用程序负载均衡器日志。

查询经典负载均衡器日志

使用经典负载均衡器日志，分析和了解传入和传出 Elastic Load Balancing 实例和后端应用程序的流量模式。您可以查看流量来源、延迟和已传输的字节。

在分析 Elastic Load Balancing 日志之前，请对其进行配置以保存在目标 Amazon S3 存储桶中。有关更多信息，请参阅为经典负载均衡器启用访问日志。

- 为 Elastic Load Balancing 日志创建表
- Elastic Load Balancing 示例查询

为 Elastic Load Balancing 日志创建表

1. 将以下 DDL 语句复制并粘贴到 Athena 控制台中。检查 Elastic Load Balancing 日志的语法。您可能需要更新以下查询以包含列和最新版本的记录的 Regex 语法。

```
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. 修改 LOCATION Amazon S3 存储桶以指定您的 Elastic Load Balancing 日志的目标。
3. 在 Athena 控制台中运行查询。查询完成后，Athena 将注册 elb_logs 表，使其中的数据可以供查询。有关更多信息，请参阅 Elastic Load Balancing 示例查询。

Elastic Load Balancing 示例查询

使用类似于以下示例的查询。它列出返回 4XX 或 5XX 错误响应代码的后端应用程序服务器。使用 LIMIT 运算符可限制每次查询的日志数目。

```
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;
```

使用后续查询对按后端 IP 地址和 Elastic Load Balancing 实例名称分组的所有事务的响应时间进行求和。

```
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;
```

有关更多信息，请参阅使用 Athena 分析 S3 中的数据。

查询 Amazon CloudFront 日志

您可以配置 Amazon CloudFront CDN 以将 Web 分配访问日志导出到 Amazon Simple Storage Service。使用这些日志可跨 CloudFront 提供的 Web 属性浏览用户的网上冲浪模式。

在开始查询日志之前，请在您的首选 CloudFront 分配上启用 Web 分配访问登录。有关信息，请参阅《Amazon CloudFront 开发人员指南》中的 访问日志。记录保存这些日志的 Amazon S3 存储桶。

- 为 CloudFront 标准日志创建表

- 为 CloudFront 实时日志创建表
- 标准 CloudFront 日志的示例查询

为 CloudFront 标准日志创建表

> **Note**
>
> 此过程适用于 CloudFront 中的 Web 分配访问日志。它不适用于 RTMP 分配中的流日志。

为 CloudFront 标准日志文件字段创建表

1. 将以下示例 DDL 语句复制并粘贴到 Athena 控制台的查询编辑器中。该示例语句使用《Amazon CloudFront 开发人员指南》的标准日志文件字段部分中记录的日志文件字段。修改用于存储日志的 Amazon S3 存储桶的 `LOCATION`。有关使用查询编辑器的信息，请参阅 开始使用。

   此查询指定了 `ROW FORMAT DELIMITED` 和 `FIELDS TERMINATED BY '\t'`，表示字段由制表符分隔。对于 `ROW FORMAT DELIMITED`，Athena 默认使用 LazySimpleSerDe。列 `date` 使用反引号 (`) 转义，因为它是 Athena 中的保留字。有关信息，请参阅保留关键字。

```
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. 在 Athena 控制台中运行查询。查询完成后，Athena 将注册 `cloudfront_standard_logs` 表，使其中的数据可以供您发出查询。

为 CloudFront 实时日志创建表

为 CloudFront 实时日志文件字段创建表

1. 将以下示例 DDL 语句复制并粘贴到 Athena 控制台的查询编辑器中。该示例语句使用了《Amazon CloudFront Developer Guide》的 Real-time logs 章节中记录的日志文件字段。修改用于存储日志的 Amazon S3 存储桶的 `LOCATION`。有关使用查询编辑器的信息，请参阅 开始使用。

   此查询指定了 `ROW FORMAT DELIMITED` 和 `FIELDS TERMINATED BY '\t'`，表示字段由制表符分隔。对于 `ROW FORMAT DELIMITED`，Athena 默认使用 LazySimpleSerDe。列 `timestamp` 使用反引号 (`) 转义，因为它是 Athena 中的保留字。有关信息，请参阅保留关键字。

   以下示例包含所有可用字段。若有不需要的字段，可将相应字段注释掉或者删除掉。

```
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. 在 Athena 控制台中运行查询。查询完成后，Athena 将注册 `cloudfront_real_time_logs` 表，使其中的数据可以供您发出查询。

标准 CloudFront 日志的示例查询

以下查询将累计 2018 年 6 月 9 日到 6 月 11 日之间由 CloudFront 提供的字节的数量。将日期列名称用双引号引起来，因为它是保留字。

```
SELECT SUM(bytes) AS total_bytes
FROM cloudfront_standard_logs
WHERE "date" BETWEEN DATE '2018-06-09' AND DATE '2018-06-11'
LIMIT 100;
```

要从查询结果中消除重复的行（例如，重复的空行），您可以使用 `SELECT DISTINCT` 语句，如以下示例所示。

```
SELECT DISTINCT *
FROM cloudfront_standard_logs
LIMIT 10;
```

其他 资源

有关使用 Athena 查询 CloudFront 日志的详细信息，请参阅 AWS 大数据博客中的以下文章。

使用 Amazon Athena 轻松查询 AWS 服务 日志（2019 年 5 月 29 日）。

大规模分析 Amazon CloudFront 访问日志（2018 年 12 月 21 日）。

使用 AWS Lambda、Amazon Athena 和适用于 Apache Flink 的亚马逊托管服务构建无服务器架构来分析 Amazon CloudFront 访问日志（2017 年 5 月 26 日）。

查询 AWS CloudTrail 日志

AWS CloudTrail 是一项服务，可记录 AWS API 调用和 Amazon Web Services 账户的事件。

CloudTrail 日志包含有关对 AWS 服务 进行的任何 API 调用的详细信息，包括控制台。CloudTrail 生成加密的日志文件并将其存储在 Amazon S3 中。有关更多信息，请参阅《AWS CloudTrail 用户指南》。

> **Note**
>
> 如果您需要跨账户、区域和日期对 CloudTrail 事件信息执行 SQL 查询，请考虑使用 CloudTrail Lake。CloudTrail Lake 是一种可用于创建跟踪的替代 AWS 服务，可将来自企业的信息聚合到单个可搜索的事件数据存储中。它不使用 Amazon S3 存储桶存储，而是将事件存储在某个数据湖中，从而支持更丰富、更快的查询。您可以使用它创建 SQL 查询，以便在自定义时间范围内跨组织和区域搜索事件。由于 CloudTrail Lake 查询是在 CloudTrail 控制台中执行，因此使用 CloudTrail Lake 不需要 Athena。有关更多信息，请参阅 CloudTrail Lake 文档。

将 Athena 与 CloudTrail 日志结合使用是加强对 AWS 服务 活动进行分析的强有力方法。例如，您可以使用查询来确定趋势，并根据属性 (如源 IP 地址或用户) 进一步隔离活动。

常见的应用程序是使用 CloudTrail 日志分析运营活动的安全性和合规性。有关详细示例的信息，请参阅 AWS 大数据博客文章：使用 AWS CloudTrail 和 Amazon Athena 分析安全性、合规性和运营活动。

您可以使用 Athena 通过指定日志文件的 `LOCATION` 直接从 Amazon S3 查询这些日志文件。您可以通过两种方式之一来执行此操作：

- 通过直接从 CloudTrail 控制台为 CloudTrail 日志文件创建表。
- 通过在 Athena 控制台中手动为 CloudTrail 日志文件创建表。

主题

了解 CloudTrail 日志和 Athena 表

在开始创建表之前，应了解有关 CloudTrail 以及它如何存储数据的更多信息。这有助于创建所需的表，无论您从 CloudTrail 控制台或从 Athena 创建它们都是如此。

CloudTrail 将日志另存为采用压缩 gzip 格式的 JSON 文本文件 (*.json.gzip)。日志文件的位置取决于您如何设置跟踪、您正在记录的一个 AWS 区域 或多个区域以及其他因素。

有关日志存储位置、JSON 结构和记录文件内容的更多信息，请参阅《AWS CloudTrail 用户指南》中的以下主题：

- 查找 CloudTrail 日志文件
- CloudTrail 日志文件示例
- CloudTrail 记录内容
- CloudTrail 事件参考

要收集日志并将其保存到 Amazon S3，请从 AWS Management Console 启用 CloudTrail。有关更多信息，请参阅《AWS CloudTrail 用户指南》中的 创建跟踪记录。

请记下您在其中保存日志的目标 Amazon S3 存储桶。将 LOCATION 子句替换为至 CloudTrail 日志位置的路径和要使用的一组对象。该示例使用特定账户的日志的 LOCATION 值，但您可以使用最适合您的应用程序的明确度程度。

例如：

- 要分析多个账户中的数据，您可以通过使用 `LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/'`，回滚 LOCATION 说明符来指示所有 AWSLogs。
- 要分析特定日期、账户和区域中的数据，请使用 `LOCATION 's3://DOC-EXAMPLE-BUCKET/123456789012/CloudTrail/us-east-1/2016/03/14/'.`

如果使用对象层次结构中的最高级别，则可以在使用 Athena 查询时获得最高程度的灵活度。

使用 CloudTrail 控制台为 CloudTrail 日志创建 Athena 表

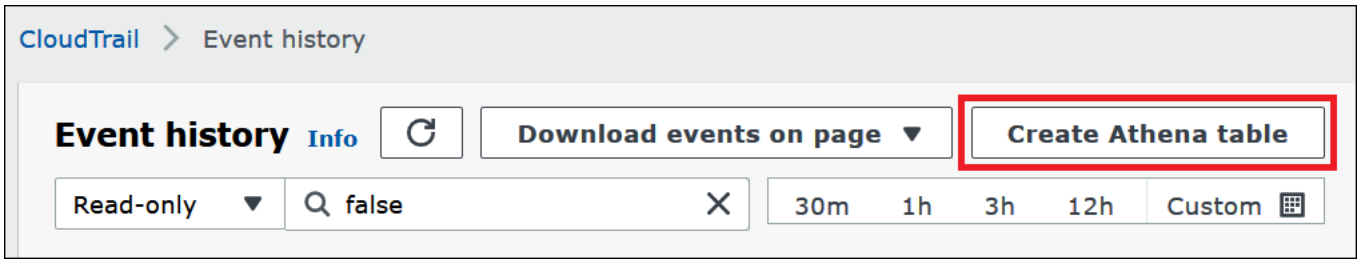
您可以创建未分区的 Athena 表，来直接从较旧的 CloudTrail 控制台查询 CloudTrail 日志。若要从 CloudTrail 控制台创建 Athena 表，您需要使用具有在 Athena 中创建表的足够权限的角色登录。

> **Note**
>
> 您不能使用 CloudTrail 控制台为组织跟踪记录日志创建 Athena 表。相反，请使用 Athena 控制台手动创建表，以便指定正确的存储位置。有关组织跟踪的信息，请参阅《AWS CloudTrail 用户指南》中的为组织创建跟踪记录。

- 有关设置 Athena 权限的信息，请参阅 设置。
- 有关创建带分区的表的信息，请参阅使用手动分区在 Athena 中为 CloudTrail 日志创建表。

使用 CloudTrail 控制台为 CloudTrail 跟踪记录创建 Athena 表

1. 访问 https://console.aws.amazon.com/cloudtrail/，打开 CloudTrail 控制台。
2. 在导航窗格中，选择事件历史记录。
3. 选择 Create Athena table（创建 Athena 表）。



4. 对于 Storage location（存储位置），使用向下箭头选择其中存储了供跟踪查询的日志文件的 Amazon S3 存储桶。

> **Note**
>
> 要查找与跟踪记录关联的存储桶的名称，请选择 CloudTrail 导航窗格中的 Trails（跟踪记录），然后查看跟踪记录的 S3 bucket（S3 存储桶）列。要查看 Amazon S3 中存储桶的位置，请在 S3 bucket（S3 存储桶）列中选择存储桶的链接。这样，就可以将 Amazon S3 控制台打开到 CloudTrail 存储桶位置。

5. 选择创建表。将使用包含 Amazon S3 存储桶名称的默认名称创建表。

使用手动分区在 Athena 中为 CloudTrail 日志创建表

可以在 Athena 控制台中为 CloudTrail 日志文件手动创建表，然后在 Athena 中运行查询。

使用 Athena 控制台为 CloudTrail 跟踪记录创建 Athena 表

1. 将以下 DDL 语句复制并粘贴到 Athena 控制台查询编辑器中。

```
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**
>
> 建议使用示例中所示的 `org.apache.hive.hcatalog.data.JsonSerDe`。虽然存在 `com.amazon.emr.hive.serde.CloudTrailSerde`，但目前还无法处理一些较新的 CloudTrail 字段。

2. （可选）删除表格中所有非必填字段。如果您只需要读取一组特定的列，则您的表定义可以排除其他列。
3. 修改 `s3://DOC-EXAMPLE-BUCKET/AWSLogs/Account_ID/CloudTrail/` 以指向包含您的日志数据的 Amazon S3 存储桶。
4. 验证字段是否正确列出。有关 CloudTrail 记录中的完整字段列表的更多信息，请参阅 CloudTrail 记录内容。

   下面的示例使用了 Hive JSON SerDe。在此示例中，字段 `requestparameters`、`responseelements` 和 `additionaleventdata` 作为查询中的 STRING 类型列出，但在 JSON 中使用 STRUCT 数据类型。因此，要将数据移出这些字段，请使

用 JSON_EXTRACT 函数。有关更多信息，请参阅 the section called “从 JSON 中提取数据”。为了提高性能，此示例按 AWS 区域、年份、月份和日期对数据进行分区。

5. 在 Athena 控制台中运行 CREATE TABLE 语句。
6. 使用 ALTER TABLE ADD PARTITION 命令加载分区，以便您可以查询它们，如以下示例所示。

```
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/'
```

使用手动分区为整个组织范围的跟踪创建表

要在 Athena 中为整个组织的 CloudTrail 日志文件创建表，请按照 使用手动分区在 Athena 中为 CloudTrail 日志创建表 中的步骤操作，但需要按照以下过程中的说明进行修改。

为整个组织的 CloudTrail 日志记录创建 Athena 表

1. 在 CREATE TABLE 语句中，修改 LOCATION 子句以包含组织 ID，如下例所示：

```
LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/organization_id/Account_ID/CloudTrail/'
```

2. 在 PARTITIONED BY 子句中，为账户 ID 添加一个字符串条目，如下例所示：

```
PARTITIONED BY (account string, region string, year string, month string, day
 string)
```

以下示例显示的是综合结果：

```
...

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. ALTER TABLE 语句的 ADD PARTITION 子句包含账户 ID，如下例所示：

```
ALTER TABLE table_name ADD
PARTITION (account='111122223333',
region='us-east-1',
year='2022',
month='08',
day='08')
```

4. ALTER TABLE 语句的 LOCATION 子句包含组织 ID、账户 ID 以及您要添加的分区，如下例所示：

```
LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/organization_id/Account_ID/CloudTrail/us-
east-1/2022/08/08/'
```

以下示例 ALTER TABLE 语句显示的是综合结果：

```
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/'
```

使用分区投影在 Athena 中为 CloudTrail 日志创建表

由于 CloudTrail 日志具有一个已知结构，您可以预先指定该结构的分区方案，因此可以使用 Athena 分区投影功能减少查询运行时间并自动管理分区。当添加新数据时，分区投影会自动添加新分区。这样就不必使用 ALTER TABLE ADD PARTITION 手动添加分区了。

以下示例 CREATE TABLE 语句会自动在 CloudTrail 日志上从指定日期开始到当前为单个 AWS 区域 使用分区投影。在 LOCATION 和 storage.location.template 子句中，将*###*、*account-id* 和 *aws-region* 占位符替换为对应的相同值。对于 projection.timestamp.range，将 *2020*/*01*/*01* 替换为要使用的开始日期。成功运行查询后，您可以查询表。您无需运行 ALTER TABLE ADD PARTITION 来加载分区。

```
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}')
```

更多有关分区投影的信息，请参阅 使用 Amazon Athena 分区投影。

查询嵌套字段

由于 userIdentity 和 resources 字段是嵌套的数据类型，查询这些内容需要特殊处理。

userIdentity 对象由嵌套 STRUCT 类型组成。可以使用点分隔字段以分隔待查询的字段，如下例所示：

```
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
```

resources 字段是一个 STRUCT 对象数组。对于这些数组，请使用 CROSS JOIN UNNEST 来取消嵌套数组，以便您可以查询其对象。

下面的示例将返回资源 ARN 以 example/datafile.txt 结尾的所有行。为了便于读取，replace 函数将从 ARN 中删除初始 arn:aws:s3::: 子字符串。

```
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
```

以下是 DeleteBucket 事件的示例查询。查询将从 resources 对象中提取存储桶的名称以及存储桶所属的账户 ID。

```
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
```

有关取消嵌套的更多信息，请参阅 筛选数组。

示例查询

以下示例显示从在 CloudTrail 事件日志创建的表，返回所有匿名（未签名）请求的查询部分。此查询选择 useridentity.accountid 匿名并且 useridentity.arn 未指定的那些请求：

```
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 ]%';
```

有关更多信息，请参阅 AWS 大数据博客文章：使用 AWS CloudTrail 和 Amazon Athena 分析安全性、合规性和运营活动。

有关查询 CloudTrail 日志的提示

要浏览 CloudTrail 日志数据，请使用下列提示：

- 在查询日志之前，请验证您的日志表是否看似与the section called “使用手动分区在 Athena 中为 CloudTrail 日志创建表”中的表一样。如果不是第一个表，请使用以下命令删除现有表：DROP TABLE cloudtrail_logs。
- 删除现有表后，重新创建它。有关更多信息，请参阅 使用手动分区在 Athena 中为 CloudTrail 日志创建表。

  确认正确列出了 Athena 查询中的字段。有关 CloudTrail 记录中的完整字段列表的信息，请参阅 CloudTrail 记录内容。

  如果您的查询包含 JSON 格式的字段，例如 STRUCT，请从 JSON 中提取数据。有关更多信息，请参阅 从 JSON 中提取数据。

  关于针对 CloudTrail 表发布查询的一些建议如下：

- 首先查看哪些用户调用了哪些 API 操作以及来自哪些源 IP 地址。
- 将以下基本 SQL 查询用作您的模板。将查询粘贴到 Athena 控制台并运行它。

```
SELECT
 useridentity.arn,
 eventname,
 sourceipaddress,
 eventtime
FROM cloudtrail_logs
LIMIT 100;
```

- 修改查询以进一步探索您的数据。
- 为了提高性能，请包含 LIMIT 子句以返回指定的行子集。

查询 Amazon EMR 日志

Amazon EMR 和在 Amazon EMR 上运行的大数据应用程序会生成日志文件。日志文件写入主节点(master node)，您还可以配置 Amazon EMR 将日志文件自动归档到 Amazon S3。您可以使用 Amazon Athena 查询这些日志，以确定应用程序和集群的事件和趋势。有关 Amazon EMR 中日志文件类型以及将其保存到 Simple Storage Service (Amazon S3) 的更多信息，请参阅《Amazon EMR 管理指南》中的查看日志文件。

基于 Amazon EMR 日志文件创建和查询基本表

以下示例基于保存到的 s3://aws-logs-*123456789012*-*us-west-2*/elasticmapreduce/*j-2ABCDE34F5GH6*/elasticmapreduce/ 日志文件创建基本表 myemrlogs。以下示例中使用的 Amazon S3 位置反映了 Amazon Web Services 账户 *123456789012* 在区域 *us-west-2* 中创建的 EMR 集群的默认日志位置的模式。如果使用自定义位置，则模式为 s3://DOC-EXAMPLE-BUCKET/*ClusterID*。

有关创建分区表以潜在改善查询性能和减少数据传输的信息，请参阅 基于 Amazon EMR 日志创建和查询分区表。

```
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'
```

以下示例查询可以在上一示例创建的 myemrlogs 表上运行。

Example – 查询步骤日志以查找出现的 ERROR、WARN、INFO、EXCEPTION、FATAL 或 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 – 查询特定实例日志 i-00b3c0a839ece0a9c 以查找 ERROR、WARN、INFO、EXCEPTION、FATAL 或 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 – 查询 Presto 应用程序日志以查找 ERROR、WARN、INFO、EXCEPTION、FATAL 或 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 – 查询 Namenode 应用程序日志以查找 ERROR、WARN、INFO、EXCEPTION、FATAL 或 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 – 按日期和小时查询所有日志以查找 ERROR、WARN、INFO、EXCEPTION、FATAL 或 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;
```

基于 Amazon EMR 日志创建和查询分区表

这些示例使用相同的日志位置创建 Athena 表，但对表进行了分区，然后为每个日志位置创建一个分区。有关更多信息，请参阅 在 Athena 中对数据进行分区。

以下查询将创建名为 mypartitionedemrlogs 的分区表：

```
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'
```

然后，以下查询语句基于 Amazon EMR 在 Amazon S3: 中创建的不同日志类型的子目录创建表分区：

```
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/'
```

创建分区后，您可以在表上运行 SHOW PARTITIONS 查询以确认：

```
SHOW PARTITIONS mypartitionedemrlogs;
```

以下示例演示对特定日志条目的查询使用由上述示例创建的表和分区。

Example – 查询容器分区中的应用程序 application_1561661818238_0002 日志以查找 ERROR 或 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 – 查询 hadoop-Mapreduce 分区以查找任务 job_1561661818238_0004 和失败的减少操作

```
SELECT data,
        "$PATH"
FROM "default"."mypartitionedemrlogs"
WHERE logtype='hadoop-mapreduce'
        AND regexp_like(data,'job_1561661818238_0004|Failed Reduces') limit 100;
```

Example – 查询节点分区中的 Hive 日志以查找查询 ID 056e0609-33e1-4611-956c-7a31b42d2663

```
SELECT data,
        "$PATH"
FROM "default"."mypartitionedemrlogs"
WHERE logtype='node'
        AND regexp_like("$PATH",'hive')
        AND regexp_like(data,'056e0609-33e1-4611-956c-7a31b42d2663') limit 100;
```

Example – 查询节点分区中的 resourcemanager 日志以查找应用程序 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
```

查询 AWS Global Accelerator 流日志

可以使用 AWS Global Accelerator 创建加速器，从而将网络流量定向到 AWS 全局网络上的最佳终端节点。有关 Global Accelerator 的更多信息，请参阅什么是 AWS Global Accelerator。

Global Accelerator 流日志允许您捕获有关进出加速器中的网络接口的 IP 地址流量的信息。流日志数据将发布到 Amazon S3，您可以在其中检索和查看您的数据。有关更多信息，请参阅 AWS Global Accelerator 中的流日志。

可以使用 Athena 查询 Global Accelerator 流日志，方式是创建一个指定流日志在 Amazon S3 中的位置的表。

为 Global Accelerator 流日志创建表

1. 将以下 DDL 语句复制并粘贴到 Athena 控制台中。此查询指定 ROW FORMAT DELIMITED 并且不再指定 SerDe，这意味着查询将使用 LazySimpleSerDe。在此查询中，字段由一个空格终止。

```
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. 修改 LOCATION 值以指向包含您的日志数据的 Amazon S3 存储桶。

```
's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/account_id/globalaccelerator/region_code/'
```

3. 在 Athena 控制台中运行查询。查询完成后，Athena 注册 aga_flow_logs 表，使其中的数据可用于查询。
4. 创建分区可读取数据，如以下示例查询中所示。此查询创建指定日期的单个分区。替换日期和位置的占位符。

```
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';
```

AWS Global Accelerator 流日志的示例查询

Example – 列出通过特定边缘站点的请求

以下示例查询列出了已通过 LHR 边缘站点的请求。使用 LIMIT 运算符可限制一次查询的日志数目。

```
SELECT
  clientip,
  agaregion,
```

```
  protocol,
  action
FROM
  aga_flow_logs
WHERE
  agaregion LIKE 'LHR%'
LIMIT
  100;
```

Example – 列出接收大多数 HTTPS 请求的端点 IP 地址

要查看哪些终端节点 IP 地址收到的 HTTPS 请求最多，请使用以下查询。此查询计算在 HTTPS 端口 443 上接收的数据包数，按目标 IP 地址对它们进行分组，并返回前 10 个 IP 地址。

```
SELECT
  SUM(numpackets) AS packetcount,
  endpointip
FROM
  aga_flow_logs
WHERE
  endpointport = 443
GROUP BY
  endpointip
ORDER BY
  packetcount DESC
LIMIT
  10;
```

查询 Amazon GuardDuty 调查结果

Amazon GuardDuty 是一项安全监控服务，用于帮助识别您 AWS 环境中的意外活动和潜在的未经授权或恶意活动。当检测到意外和潜在恶意活动时，GuardDuty 会生成安全调查结果，您可以将其导出到 Amazon S3 进行存储和分析。将调查结果导出到 Amazon S3 后，您可以使用 Athena 进行查询。本文介绍如何在 Athena 中为 GuardDuty 调查结果创建表并查询它们。

有关 Amazon GuardDuty 更多信息，请参阅《Amazon GuardDuty 用户指南》。

先决条件

- 启用 GuardDuty 功能以将调查结果导出到 Amazon S3。有关步骤，请参阅《Amazon GuardDuty 用户指南》中导出调查结果。

在 Athena 中为 GuardDuty 调查结果创建表

要从 Athena 查询您的 GuardDuty 调查结果，您必须为它们创建一个表。

要在 Athena 中为 GuardDuty 调查结果创建表

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 将以下 DDL 语句粘贴到 Athena 控制台中。修改 `LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/account-id/GuardDuty/'` 中的值以指向您在 Amazon S3 中的 GuardDuty 调查结果。

```
CREATE EXTERNAL TABLE `gd_logs` (
  `schemaversion` string,
  `accountid` string,
  `region` string,
  `partition` string,
  `id` string,
  `arn` string,
  `type` string,
  `resource` string,
  `service` string,
  `severity` string,
  `createdat` string,
  `updatedat` string,
  `title` string,
  `description` string)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
 LOCATION 's3://DOC-EXAMPLE-BUCKET/AWSLogs/account-id/GuardDuty/'
 TBLPROPERTIES ('has_encrypted_data'='true')
```

> **Note**
>
> SerDe 期望每个 JSON 文档都位于单行文本中，并且不使用行终止字符分隔记录中的字段。如果 JSON 文本采用美观的打印格式，当您在创建表后尝试对其进行查询时，可能会收到类似以下内容的错误消息：HIVE_CURSOR_ERROR: Row is not a valid JSON Object（HIVE_CURSOR_ERROR：行不是有效的 JSON 对象）或 HIVE_CURSOR_ERROR: JsonParseException: Unexpected end-of-input: expected close marker for OBJECT（HIVE_CURSOR_ERROR：JsonParseException：意外的输入结束：对象的预期关闭标记）。有关更多信息，请参阅 GitHub 上 OpenX SerDe 文档中的 JSON 数据文件。

3. 在 Athena 控制台中运行查询以注册 gd_logs 表。查询完成后，调查结果准备就绪，可供您从 Athena 查询。

示例查询

以下示例演示了如何从 Athena 查询 GuardDuty 调查结果。

Example – DNS 数据泄露

以下查询返回可能会通过 DNS 查询泄露数据的 Amazon EC2 实例的相关信息。

```
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 – 未授权的 IAM 用户访问

以下查询返回所有区域中 IAM 委托人的所有 UnauthorizedAccess:IAMUser 调查结果类型。

```
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;
```

有关查询 GuardDuty 调查结果的提示

创建查询时，请记住以下几点。

- 要从嵌套 JSON 字段中提取数据，请使用 Presto `json_extract` 或 `json_extract_scalar` 函数。有关更多信息，请参阅 从 JSON 中提取数据。
- 请确保 JSON 字段中的所有字符均为小写字符。
- 有关下载查询结果的信息，请参阅使用 Athena 控制台下载查询结果文件。

查询 AWS Network Firewall 日志

AWS Network Firewall 是一种托管式服务，您可以使用它为您的 Amazon Virtual Private Cloud 实例部署必要的网络保护。AWS Network Firewall 与 AWS Firewall Manager 结合使用，以便您可以基于 AWS Network Firewall 规则构建策略，然后在您的 VPC 和账户中集中应用这些策略。有关 AWS Network Firewall 的更多信息，请参阅 AWS Network Firewall。

您可以为您转发到防火墙有状态规则引擎的流量配置 AWS Network Firewall 日志记录。日志记录为您提供有关网络流量的详细信息，包括有状态引擎接收数据包的时间、有关数据包的详细信息以及针对数据包采取的任何有状态规则操作。日志将发布到您配置的日志目标，您可以在其中检索和查看日志。有关更多信息，请参阅《AWS Network Firewall 开发人员指南》中的录入来自 AWS Network Firewall 的网络流量。

为警报日志创建表

1. 修改下面的 DDL 语句示例，使其符合警报日志的结构。您可能需要更新语句以包含最新版本日志的列。有关更多信息，请参阅《AWS Network Firewall 开发人员指南》中的防火墙日志的内容。

```
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. 修改 LOCATION 子句以指定您在 Amazon S3 中的日志文件夹。
3. 在 Athena 查询编辑器中运行 CREATE TABLE 查询。查询完成后，Athena 将注册 network_firewall_alert_logs 表，使其指向的数据可以进行查询。

警报日志示例查询

本节中的示例警报日志查询会筛选执行了 TLS 检查且警报严重性级别为 2 或更高的事件。

查询使用别名来创建输出列标题，以显示列所属的 struct。例如，event.alert.category 字段的列标题是 event_alert_category，而不只是 category。若要进一步自定义列名，可以根据自己的喜好修改别名。例如，可以使用下划线或其他分隔符来分隔 struct 名称和字段名称。

请记得根据表定义和查询结果中所需的字段修改列名和 struct 引用。

```
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;
```

为 netflow 日志创建表

1. 修改下面的 DDL 语句示例，使其符合 netflow 日志的结构。您可能需要更新语句以包含最新版本日志的列。有关更多信息，请参阅《AWS Network Firewall 开发人员指南》中的防火墙日志的内容。

```
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. 修改 LOCATION 子句以指定您在 Amazon S3 中的日志文件夹。
3. 在 Athena 查询编辑器中运行 CREATE TABLE 查询。查询完成后，Athena 将注册 network_firewall_netflow_logs 表，使其指向的数据可以进行查询。

Netflow 日志示例查询

本节中的 netflow 日志查询示例会筛选执行了 TLS 检查的事件。

查询使用别名来创建输出列标题，以显示列所属的 struct。例如，event.netflow.bytes 字段的列标题是 event_netflow_bytes，而不只是 bytes。若要进一步自定义列名，可以根据自己的喜好修改别名。例如，可以使用下划线或其他分隔符来分隔 struct 名称和字段名称。

请记得根据表定义和查询结果中所需的字段修改列名和 struct 引用。

```
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
```

查询 Network Load Balancer 日志

使用 Athena 分析和处理来自 Network Load Balancer 的日志。这些日志将接收有关发送到 Network Load Balancer 的传输层安全性 (TLS) 请求的详细信息。您可以使用这些访问日志分析流量模式并解决问题。

在分析 Network Load Balancer 访问日志之前，请启用并配置它们以保存在目标 Amazon S3 存储桶中。有关更多信息以及有关每个网络负载均衡器访问日志条目的信息，请参阅网络负载均衡器的访问日志。

- 为 Network Load Balancer 日志创建表
- Network Load Balancer 示例查询

为 Network Load Balancer 日志创建表

1. 将以下 DDL 语句复制并粘贴到 Athena 控制台中。检查 Network Load Balancer 日志记录的语法。您可能需要更新以下查询以包含列和最新版本的记录的 Regex 语法。

```
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. 修改 LOCATION Amazon S3 存储桶以指定您的 Network Load Balancer 日志的目标。
3. 在 Athena 控制台中运行查询。查询完成后，Athena 将注册 `nlb_tls_logs` 表，使其中的数据可以供查询。

Network Load Balancer 示例查询

要查看证书的使用次数，请使用与以下示例类似的查询：

```
SELECT count(*) AS
         ct,
         cert_arn
FROM "nlb_tls_logs"
GROUP BY  cert_arn;
```

以下查询显示了所用 TLS 版本低于 1.3 的用户数：

```
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;
```

使用以下查询可确定需要较长的 TLS 握手时间的连接：

```
SELECT *
FROM "nlb_tls_logs"
ORDER BY  tls_handshake_time_ms DESC
LIMIT 10;
```

使用以下查询可识别和统计过去 30 天内协商的 TLS 协议版本和密码套件。

```
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;
```

查询 Amazon Route 53 Resolver 查询日志

您可以为 Amazon Route 53 Resolver 查询日志创建 Athena 表，并从 Athena 查询这些日志。

Route 53 解析程序查询日志记录用于记录 VPC 中的资源提出的 DNS 查询、使用入站解析程序端点的本地部署资源、使用出站解析程序端点进行递归 DNS 解析的查询以及使用 Route 53 解析程序 DNS 防火墙规则阻止、允许或监控域列表的查询。有关解析程序查询日志记录的更多信息，请参阅《Amazon Route 53 开发人员指南》中的解析程序查询日志记录。有关日志中每个字段的信息，请参阅《Amazon Route 53 开发人员指南》中的 显示在解析程序查询日志中的值。

为解析程序查询日志创建表

您可以使用 Athena 控制台中的查询编辑器为 Route 53 解析程序查询日志创建和查询表。

为 Route 53 解析程序查询日志创建和查询 Athena 表

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 在 Athena 查询编辑器中，输入以下 `CREATE TABLE` 语句。将 `LOCATION` 子句值替换为与您的解析程序日志在 Amazon S3 中的位置相对应的子句值。

```
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}/'
```

由于解析程序查询日志数据是 JSON 格式的，因此 CREATE TABLE 语句将使用 JSON SerDe 库来分析数据。

> **Note**
>
> SerDe 期望每个 JSON 文档都位于单行文本中，并且不使用行终止字符分隔记录中的字段。如果 JSON 文本采用美观的打印格式，当您在创建表后尝试对其进行查询时，可能会收到类似以下内容的错误消息：HIVE_CURSOR_ERROR: Row is not a valid JSON Object（HIVE_CURSOR_ERROR：行不是有效的 JSON 对象）或 HIVE_CURSOR_ERROR: JsonParseException: Unexpected end-of-input: expected close marker for OBJECT（HIVE_CURSOR_ERROR：JsonParseException：意外的输入结束：对象的预期关闭标记）。有关更多信息，请参阅 GitHub 上 OpenX SerDe 文档中的 JSON 数据文件。

3. 选择运行查询。该语句创建名为 `r53_rlogs` 的 Athena 表，其中的列表示解析程序日志数据中的每个字段。
4. 在 Athena 控制台查询编辑器中，运行以下查询以验证您的表是否已创建。

```
SELECT * FROM "r53_rlogs" LIMIT 10
```

分区示例

以下示例显示了 Resolver 查询日志的 CREATE TABLE 语句，该语句使用分区投影并按 VPC 和日期进行分区。更多有关分区投影的信息，请参阅 使用 Amazon Athena 分区投影。

```
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}/'
)
```

示例查询

以下示例显示了您可以从 Athena 对解析程序查询日志执行的一些查询。

示例 1 – 按降序 query_timestamp 顺序查询日志

以下查询以降序 query_timestamp 顺序显示日志结果。

```
SELECT * FROM "r53_rlogs"
ORDER BY query_timestamp DESC
```

示例 2 – 指定的开始时间和结束时间内的查询日志

以下查询查询在 2020 年 9 月 24 日午夜和上午 8 点之间日志。根据您自己的要求替换开始时间和结束时间。

```
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
```

示例 3 – 基于指定 DNS 查询名称模式的查询日志

以下查询选择其查询名称包含字符串“example.com”的记录。

```
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
```

示例 4 – 没有答案的查询日志请求

以下查询选择请求未收到任何答案的日志条目。

```
SELECT query_timestamp, srcids.instance, srcaddr, srcport, query_name, rcode, answers
FROM "r53_rlogs"
WHERE cardinality(answers) = 0
```

示例 5 – 查询具有特定答案的日志

以下查询显示了 `answer.Rdata` 值具有指定的 IP 地址的日志。

```
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';
```

查询 Amazon SES 事件日志

您可以使用 Amazon Athena 查询Amazon Simple Email Service（Amazon SES）事件日志。

Amazon SES 是一个电子邮件平台，让您能够使用自己的电子邮件地址和域，经济高效且方便地收发电子邮件。您可以使用事件、指标和统计信息更加细致地监控 Amazon SES 发送活动。

您可以根据自己定义的特征，将 Amazon SES 事件发布到 Amazon CloudWatch、Amazon Data Firehose 或 Amazon Simple Notification Service。将信息存储到 Amazon S3 以后，您可以从 Amazon Athena 进行查询。

有关如何使用 Firehose、Amazon Athena 和 Amazon QuickSight 分析 Amazon SES 电子邮件事件的更多信息，请参阅 AWS Messaging and Targeting Blog 中的 Analyzing Amazon SES event data with AWS Analytics Services。

查询 Amazon VPC 流日志

Amazon Virtual Private Cloud 流日志捕获有关在 VPC 中传入和传出网络接口的 IP 流量的信息。可使用日志来调查网络流量模式，并识别 VPC 网络中的威胁和风险。

若要查询 Amazon VPC 流日志，您有两种选择：

- Amazon VPC 控制台 – 使用 Amazon VPC 控制台中的 Athena 集成功能生成 AWS CloudFormation 模板，用于创建 Athena 数据库、工作组和流日志表并为您进行分区。该模板还会创建一组预定义的流日志查询，可用于获取有关流经 VPC 的流量的洞察。

  有关此方法更多信息，请参阅《Amazon VPC 用户指南》中的使用 Amazon Athena 查询流日志。
- Amazon Athena 控制台 – 直接在 Athena 控制台中创建表和查询。有关更多信息，请继续阅读此页面。

创建和查询自定义 VPC 流日志表

当您开始在 Athena 中查询日志之前，启用 VPC 流日志，并将其配置为保存到您的 Amazon S3 存储桶。在您创建日志后，让它们运行几分钟以收集一些数据。这些日志是采用 Athena 允许您直接查询的 GZIP 压缩格式创建的。

创建 VPC 流日志时，当您希望指定在流日志中返回的字段以及这些字段的显示顺序时，请使用自定义格式。有关流日志记录的更多信息，请参阅《Amazon VPC 用户指南》中的流日志记录。

常见注意事项

在 Athena 中为 Amazon VPC 流日志创建表时，请记住以下几点：

- 默认情况下，在 Athena 中，Parquet 将按名称访问列。有关更多信息，请参阅 处理架构更新。
- 使用流日志记录中的名称作为 Athena 中列的名称。Athena 架构中列的名称应与 Amazon VPC 流日志中的字段名称完全匹配，但有以下不同之处：
  - 将 Amazon VPC 日志字段名称中的连字符替换为 Athena 列名称中的下划线。在 Athena 中，数据库名称、表名称和列名称仅可接受小写字母、数字和下划线字符。有关更多信息，请参阅 数据库、表和列名称。
  - 在 Athena 中，使用反引号将保留关键字中的流日志记录名称括起来，将其转义。
- VPC 流日志特定于 AWS 账户。当您将日志文件发布到 Amazon S3 时，在 Amazon S3 中创建的 Amazon VPC 的路径包含用于创建流日志的 AWS 账户 ID。有关更多信息，请参阅 Amazon VPC 用户指南中的将流日志发布到 Amazon S3。

适用于 Amazon VPC 流日志的 CREATE TABLE 语句

以下过程将为 Amazon VPC 流日志创建 Amazon VPC 表。使用自定义格式创建流日志时，可以创建一个表，其字段与您在创建流日志时指定的字段相匹配，且字段顺序与您指定的字段顺序相同。

为 Amazon VPC 流日志创建 Athena 表

1. 按照 常见注意事项 部分中的准则，在 Athena 控制台查询编辑器中输入类似以下内容的 DDL 语句。此示例语句创建一个表，其中包含 Amazon VPC 流日志版本 2 到 5 的列，如流日志记录中所示。如果使用不同的列集或列顺序，请相应地修改语句。

```
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");
```

请注意以下几点：

- 此查询指定 ROW FORMAT DELIMITED 并省略指定 SerDe。这意味着查询使用用于 CSV、TSV 和自定义分隔文件的 LazySimpleSerDe。在此查询中，字段由一个空格终止。
- PARTITIONED BY 子句使用 date 类型。这样，就可以在查询中使用数学运算符来选择比特定日期更早或更新的内容。

> **Note**
>
> 因为 date 在 DDL 语句中是保留关键字，所以用反引号字符对其进行转义。有关更多信息，请参阅 保留关键字。

- 对于具有不同自定义格式的 VPC 流日志，请修改字段以与您创建流日志时指定的字段相匹配。

2. 修改 LOCATION 's3://DOC-EXAMPLE-BUCKET/*prefix*/AWSLogs/*{account_id}*/vpcflowlogs/*{region_code}*/' 以指向包含您的日志数据的 Amazon S3 存储桶。
3. 在 Athena 控制台中运行查询。查询完成后，Athena 将注册 vpc_flow_logs 表，使其中的数据可以供您发出查询。
4. 创建分区以便能够读取数据，如以下示例查询中所示。此示例查询创建指定日期的单个分区。根据需要替换日期和位置的占位符。

> **Note**
>
> 此查询仅为您指定的日期创建单个分区。若要自动执行此过程，请使用运行此查询并以这种方式为 year/month/day 创建分区的脚本，或使用 CREATE TABLE 语句指定分区投影。

```
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';
```

vpc_flow_logs 表的查询示例

使用 Athena 控制台中的查询编辑器在创建的表上运行 SQL 语句。您可以保存查询、查看之前的查询或下载 CSV 格式的查询结果。在以下示例中，将 `vpc_flow_logs` 替换为表名称。根据您的要求修改列值和其他变量。

以下示例查询列出了指定日期的最多 100 个流日志。

```
SELECT *
FROM vpc_flow_logs
WHERE date = DATE('2020-05-04')
LIMIT 100;
```

以下查询列出所有被拒绝的 TCP 连接并使用新创建的日期分区列 `date` 来从中提取这些事件发生的星期几。

```
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;
```

若要查看哪个服务器收到最大数量的 HTTPS 请求，请使用以下查询。它计算在 HTTPS 端口 443 上接收的数据包数，按目标 IP 地址对它们进行分组，并返回上一周的前 10 个。

```
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;
```

以 Apache Parquet 格式为流日志创建表

以下过程将以 Apache Parquet 格式为 Amazon VPC 流日志创建 Amazon VPC 表。

以 Parquet 格式为 Amazon VPC 流日志创建 Athena 表

1. 按照 常见注意事项 部分中的准则，在 Athena 控制台查询编辑器中输入类似以下内容的 DDL 语句。以下示例语句创建一个表，其中包含 Amazon VPC 流日志版本 2 到 5 的列，如 Parquet 格式的流日志记录中所示，Hive 按小时分区。如果您没有按小时分区，请删除 PARTITIONED BY 子句中的 hour。

```
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. 修改示例 `LOCATION 's3://DOC-EXAMPLE-BUCKET/prefix/AWSLogs/'` 以指向包含您的日志数据的 Amazon S3 路径。
3. 在 Athena 控制台中运行查询。
4. 如果数据采用 Hive 兼容格式，请在 Athena 控制台中运行以下命令，更新并加载元存储中的 Hive 分区。查询完成后，可以查询 `vpc_flow_logs_parquet` 表中的数据。

   ```
   MSCK REPAIR TABLE vpc_flow_logs_parquet
   ```

   如果您使用的不是 Hive 兼容数据，请运行 ALTER TABLE ADD PARTITION 以加载分区。

有关使用 Athena 查询 Parquet 格式 Amazon VPC 流日志的更多信息，请参阅 AWS 大数据博客中的文章 使用 Apache Pparquet 格式的 VPC 流日志优化性能并降低网络分析成本。

使用分区投影创建和查询 Amazon VPC 流日志表

使用如下所示的 `CREATE TABLE` 语句创建表、对表进行分区并使用分区投影自动填充分区。将示例中的表名称 `test_table_vpclogs` 替换为您的表名称。编辑 `LOCATION` 子句以指定包含 Amazon VPC 日志数据的 Amazon S3 存储桶。

以下 `CREATE TABLE` 语句适用于以非 Hive 样式的分区格式传送的 VPC 流日志。该示例支持多账户聚合。要将来自多个账户的 VPC 流日志集中到一个 Amazon S3 存储桶中，必须在 Amazon S3 路径中输入账户 ID。

```
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}"
)
```

test_table_vpclogs 表的查询示例

以下示例查询将查询之前 CREATE TABLE 语句创建的 test_table_vpclogs。将查询中的 test_table_vpclogs 替换为您自己的表名称。根据您的要求修改列值和其他变量。

若要在指定时间段内按时间顺序返回前 100 个访问日志条目，请运行如下所示的查询。

```
SELECT *
FROM test_table_vpclogs
WHERE day >= '2021/02/01' AND day < '2021/02/28'
ORDER BY day ASC
LIMIT 100
```

若要查看哪个服务器在指定时间段内接收前十个 HTTP 数据包，请运行如下所示的查询。该查询计算在 HTTPS 端口 443 上接收的数据包数，按目标 IP 地址对其进行分组，并返回上一周的前 10 个条目。

```
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
```

若要返回在指定时间段内创建的日志，请运行如下所示的查询。

```
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'
```

若要返回指定时间段内源 IP 地址的访问日志，请运行如下所示的查询。

```
SELECT *
FROM test_table_vpclogs
WHERE srcaddr = '10.117.1.22'
  AND day >= '2021/02/01'
  AND day < '2021/02/28'
```

若要列出已被拒绝的 TCP 连接，请运行如下所示的查询。

```
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
```

若要返回以 10.117 开头的 IP 地址范围的访问日志，请运行如下所示的查询。

```
SELECT *
FROM test_table_vpclogs
WHERE split_part(srcaddr,'.', 1)='10'
  AND split_part(srcaddr,'.', 2) ='117'
```

若要返回某个时间范围内目标 IP 地址的访问日志，请运行如下所示的查询。

```
SELECT *
FROM test_table_vpclogs
WHERE dstaddr = '10.0.1.14'
  AND day >= '2021/01/01'
  AND day < '2021/01/31'
```

使用分区投影并以 Apache Parquet 格式为流日志创建表

以下用于 VPC 流日志的分区投影 CREATE TABLE 语句采用 Apache Parquet 格式，不兼容 Hive，按小时和日期而不是按天分区。将示例中的表名称 `test_table_vpclogs_parquet` 替换为您的表名称。编辑 `LOCATION` 子句以指定包含 Amazon VPC 日志数据的 Amazon S3 存储桶。

```
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}"
)
```

其他 资源

有关使用 Athena 分析 VPC 流日志的更多信息，请参阅以下 AWS 大数据博客文章：

- 通过点击式 Amazon Athena 集成分析 VPC 流日志
- 使用 Amazon Athena 和 Amazon QuickSight 分析 VPC 流日志
- 使用 Apache Parquet 格式的 VPC 流日志优化性能并降低网络分析成本

查询 AWS WAF 日志

AWS WAF 是一个 Web 应用程序防火墙，可以监视和控制受保护的 Web 应用程序从客户端收到的 HTTP 和 HTTPS 请求。您可以通过在 AWS WAF Web 访问控制列表（ACL）中配置规则来定义如何处理 Web 请求。然后，您可以通过将 Web ACL 关联到 Web 应用程序来保护该应用程序。您可以使用 AWS WAF 保护的 Web 应用程序资源的示例包括 Amazon CloudFront 分配、Amazon API Gateway REST API 和应用程序负载均衡器。有关 AWS WAF 的更多信息，请参阅《AWS WAF Developer Guide》中的 AWS WAF。

AWS WAF 日志包含您的 Web ACL 所分析的流量相关信息，例如 AWS WAF 从 AWS 资源收到请求的时间，有关请求的详细信息，以及每个请求所匹配的规则的操作。

您可以配置 AWS WAF Web ACL 将日志发布到多个目标中的一个目标，即您可以查询和查看这些日志的地方。有关配置 Web ACL 日志记录和 AWS WAF 日志内容的更多信息，请参阅《AWS WAF Developer Guide》中的 Logging AWS WAF web ACL traffic。

有关如何将 AWS WAF 日志聚合到中央数据湖存储库并使用 Athena 进行查询的示例，请参阅 AWS 大数据博客文章使用 OpenSearch 服务、Amazon Athena 和 Amazon QuickSight 分析 AWS WAF 日志。

本主题提供两个示例 CREATE TABLE 语句：一个使用分区，另一个不使用分区。

> **Note**
>
> 本主题中的 CREATE TABLE 语句可以用于 v1 和 v2 AWS WAF 日志。在 v1 中，webaclid 字段包含一个 ID。在 v2 中，webaclid 字段包含完整的 ARN。这里的 CREATE TABLE 语句通过使用 string 数据类型未知地处理此内容。

主题

使用分区投影为 Athena 中的 AWS WAF S3 日志创建表

由于 AWS WAF 日志具有您可以预先指定其分区方案的已知结构，因此您可以使用 Athena 分区投影功能减少查询运行时间并自动管理分区。当添加新数据时，分区投影会自动添加新分区。这样就不必使用 ALTER TABLE ADD PARTITION 手动添加分区了。

以下示例 CREATE TABLE 语句会自动在 AWS WAF 日志上从指定日期开始到当前日期为止，为四个不同 AWS 区域使用分区投影。本示例中的 PARTITION BY 子句按区域和日期进行分区，但您可以根据自己的要求修改此子句。根据需要修改字段以匹配您的日志输出。在 LOCATION 和 storage.location.template 子句中，将 *bucket* 和 *accountID* 占位符替换为值，该值标识 AWS WAF 日志在 Amazon S3 存储桶中的位置。对于 projection.day.range，将 *2021*/*01*/*01* 替换为要使用的开始日期。成功运行查询后，您可以查询表。您无需运行 ALTER TABLE ADD PARTITION 来加载分区。

```
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,
```

```
                                                                matchedata: 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

示例中 LOCATION 子句中的路径格式是标准格式，但可能因所实施的 AWS WAF 配置而异。例如，以下示例 AWS WAF 日志路径适用于 CloudFront 分配：

```
s3://DOC-EXAMPLE-BUCKET/AWSLogs/12345678910/WAFLogs/cloudfront/
cloudfronyt/2022/08/08/17/55/
```

如果您在创建或查询 AWS WAF 日志表时遇到问题，请确认日志数据位置或联系 AWS Support。

更多有关分区投影的信息，请参阅 使用 Amazon Athena 分区投影。

创建不进行分区的 AWS WAF 日志表

本节介绍如何创建不进行分区或分区投影的 AWS WAF 日志表。

> **Note**
>
> 出于性能和成本原因，不建议使用非分区架构进行查询。有关更多信息，请参阅 AWS 大数据博客中的 Top 10 Performance Tuning Tips for Amazon Athena（Amazon Athena 的十大性能优化技巧）。

创建 AWS WAF 表

1. 将以下 DDL 语句复制并粘贴到 Athena 控制台中。根据需要修改字段以匹配您的日志输出。修改 Amazon S3 存储桶的 `LOCATION` 以对应用于存储日志的存储桶。

   此查询使用 OpenX JSON SerDe。

   > **Note**
   >
   > SerDe 期望每个 JSON 文档都位于单行文本中，并且不使用行终止字符分隔记录中的字段。如果 JSON 文本采用美观的打印格式，当您在创建表后尝试对其进行查询时，可能会收到类似以下内容的错误消息：HIVE_CURSOR_ERROR: Row is not a valid JSON Object（HIVE_CURSOR_ERROR：行不是有效的 JSON 对象）或 HIVE_CURSOR_ERROR: JsonParseException: Unexpected end-of-input: expected close marker for OBJECT（HIVE_CURSOR_ERROR：JsonParseException：意外的输入结束：对象的预期关闭标记）。有关更多信息，请参阅 GitHub 上 OpenX SerDe 文档中的 JSON 数据文件。

```
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. 在 Athena 控制台查询编辑器中运行 CREATE EXTERNAL TABLE 语句。这将注册 waf_logs 表，并使其中的数据可用于来自 Athena 的查询。

AWS WAF 日志的示例查询

以下许多示例查询使用之前在此文档中创建的分区投影表。根据您的要求修改示例中的表名称、列值和其他变量。若要提高查询的性能并降低成本，请在筛选条件中添加分区列。

- 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

使用日期和时间

- 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

使用阻止的请求和地址

- 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 – 统计包含指定术语的引用站点数量

以下查询计算指定日期范围内包含“amazon”一词的引用者数量。

```
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 – 统计过去 10 天内与排除规则匹配的所有匹配 IP 地址

以下查询计算过去 10 天内 IP 地址与规则组中排除规则匹配的次数。

```
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 - 按匹配次数对所有已计数的托管规则进行分组

如果您在 2022 年 10 月 27 日之前在 Web ACL 配置中将规则组规则操作设置为“计数”，AWS WAF 在 Web ACL JSON 中将覆盖内容保存为 `excludedRules`。现在，用于将规则替换为“计数”的 JSON 设置位于 `ruleActionOverrides` 设置中。有关更多信息，请参阅《AWS WAF 开发人员指南》中的规则组中的操作覆盖。要从新的日志结构中提取计数模式下的托管规则，请在 `ruleGroupList` 部分而不是 `excludedRules` 字段中查询 `nonTerminatingMatchingRules`，如下例所示。

```
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 - 按匹配次数对所有已计数的自定义规则进行分组

以下查询按匹配次数对所有已计数的自定义规则进行分组。

```
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
```

有关自定义规则和托管规则组的日志位置的信息，请参阅《AWS WAF 开发人员指南》中的监控和调整。

使用日期和时间

Example – 以人类可读 ISO 8601 格式返回时间戳字段

以下查询使用 from_unixtime 和 to_iso8601 函数以人类可读的 ISO 8601 格式返回 timestamp 字段（例如 2019-12-13T23:40:12.000Z 而不是 1576280412771）。该查询还返回 HTTP 源名称、源 ID 和请求。

```
SELECT to_iso8601(from_unixtime(timestamp / 1000)) as time_ISO_8601,
       httpsourcename,
       httpsourceid,
       httprequest
FROM waf_logs
LIMIT 10;
```

Example – 返回过去 24 小时的记录

以下查询使用 WHERE 子句中的筛选条件返回过去 24 小时内记录的 HTTP 源名称、HTTP 源 ID 和 HTTP 请求字段。

```
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 – 返回指定日期范围和 IP 地址的记录

以下查询列出了指定的客户端 IP 地址在指定日期范围内的记录。

```
SELECT *
```

```
FROM waf_logs
WHERE httprequest.clientip='53.21.198.66' AND "date" >= '2021/03/01' AND "date" <
 '2021/03/31'
```

Example – 对于指定的日期范围，计算在五分钟间隔内的 IP 地址数

对于特定日期范围，以下查询计算在五分钟间隔内的 IP 地址数。

```
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 – 计算过去 10 天内 X-Forwarded-For IP 的数量

以下查询将筛选请求标头，并统计过去 10 天内 X-Forwarded-For IP 的数量。

```
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
```

有关日期和时间函数的更多信息，请参阅 Trino 文档中的 日期与时间函数和运算符。

使用阻止的请求和地址

Example – 提取被指定规则类型阻止的前 100 个 IP 地址

下面的查询将提取并统计在指定的日期范围内被 RATE_BASED 终止规则阻止的前 100 个 IP 地址。

```
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 – 计算来自指定国家/地区的请求被阻止的次数

以下查询针对来自属于爱尔兰 (IE) IP 地址的请求，计算请求到达但被 RATE_BASED 终止规则阻止的次数。

```
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 – 计算请求被阻止的次数，按特定属性分组

以下查询计算请求被阻止的次数，并按照 WebACL、RuleId、ClientIP 和 HTTP 请求 URI 对结果分组。

```
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 – 计算特定终止规则 ID 匹配的次数

以下查询计算特定终止规则 ID 匹配的次数 (WHERE terminatingruleid='e9dd190d-7a43-4c06-bcea-409613d9506e')。然后，查询按照 WebACL、操作、ClientIP 和 HTTP 请求 URI 对结果分组。

```
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 – 检索指定日期范围内被阻止的前 100 个 IP 地址

以下查询将提取在指定日期范围内被阻止的前 100 个 IP 地址。该查询还列出了 IP 地址被阻止的次数。

```
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
```

有关查询 Amazon S3 日志的更多信息，请参阅以下主题：

- AWS 知识中心中的如何使用 Athena 分析 Amazon S3 服务器访问日志？
- 《Amazon Simple Storage Service 用户指南》中的使用 Amazon Athena 查询请求的 Amazon S3 访问日志
- 《Amazon Simple Storage Service 用户指南》中的使用 AWS CloudTrail 识别 Amazon S3 请求

查询 Amazon S3 中存储的 Web 服务器日志

您可以使用 Athena 查询 Amazon S3 中存储的 Web 服务器日志。本部分中的主题介绍如何在 Athena 中创建表，以查询各种格式的 Web 服务器日志。

主题

查询存储在 Amazon S3 中的 Apache 日志

您可以使用 Amazon Athena 以查询存储在 Amazon S3 账户中的 Apache HTTP 服务器日志文件。本主题介绍如何创建表架构以查询常见日志格式的 Apache 访问日志文件。

常见日志格式中的字段包括客户端 IP 地址、客户端 ID、用户 ID、接收请求的时间戳、客户端请求的文本、服务器状态代码以及返回给客户端的对象的大小。

以下示例数据显示了 Apache 的常见日志格式。

```
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
```

在 Athena 中为 Apache 日志创建表

必须先为 Athena 创建一个表架构，以便它能够读取日志数据，然后才能查询存储在 Amazon S3 中的 Apache 日志。若要为 Apache 日志创建 Athena 表，可以使用 Grok SerDe。有关使用 Grok SerDe 的更多信息，请参阅《AWS Glue 开发人员指南https://docs.aws.amazon.com/glue/latest/dg/custom-classifier.html#custom-classifier-grok》中的 编写 Grok 自定义分类器。

要在 Athena 中为 Apache Web 服务器日志创建表

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 将以下 DDL 语句粘贴到 Athena 查询编辑器中。修改 `LOCATION 's3://DOC-EXAMPLE-BUCKET/apache-log-folder/'` 中的值以指向 Amazon S3 中的 Apache 日志。

```
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. 在 Athena 控制台中运行查询以注册 apache_logs 表。查询完成后，调查结果准备就绪，可供您从 Athena 查询。

为 Apache 日志选择查询的示例

Example – 筛选 404 错误

以下的示例查询从 apache_logs 表中选择了请求接收时间、客户端请求的文本以及服务器状态代码。HTTP 状态代码 404（未找到页面）的 WHERE 子句筛选条件。

```
SELECT request_received_time, client_request, server_status
FROM apache_logs
WHERE server_status = '404'
```

下图显示了 Athena 查询编辑器中的查询结果。

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 – 筛选成功的请求

以下的示例查询从 apache_logs 表中选择了用户 ID、请求接收时间、客户端请求的文本以及服务器状态代码。HTTP 状态代码 200（成功）的子句筛选条件 WHERE。

```
SELECT user_id, request_received_time, client_request, server_status
FROM apache_logs
WHERE server_status = '200'
```

下图显示了 Athena 查询编辑器中的查询结果。

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 – 按时间戳筛选

如下示例将查询请求接收时间大于指定时间戳的记录。

```
SELECT * FROM apache_logs WHERE request_received_time > 10/Oct/2023:00:00:00
```

查询存储在 Amazon S3 中的互联网信息服务器（IIS）日志

您可以使用 Amazon Athena 查询存储在您 Amazon S3 账户中的 Microsoft 互联网信息服务 (IIS) Web 服务器日志。虽然 IIS 使用各种不同的日志文件格式，本主题将介绍如何创建表架构以从 Athena 查询 W3C 扩展日志和 IIS 日志文件格式日志。

由于 W3C 扩展和 IIS 日志文件格式使用单字符分隔符（分别为空格和逗号），并且没有位于引号中的值，因此您可以使用 LazySimpleSerDe 为其创建 Athena 表。

W3C 扩展日志文件格式

W3C 扩展日志文件数据格式具有空格分隔的字段。W3C 扩展日志中显示的字段由 Web 服务器管理员决定，后者将选择要包含哪些日志字段。以下示例日志数据具有 `date`、`time`、`c-ip`、`s-ip`、`cs-method`、`cs-uri-stem`、`sc-status`、`sc-bytes`、`cs-bytes`、`time-taken` 和 `cs-version` 字段。

```
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
```

在 Athena 中为 W3C 扩展日志创建表

在查询 W3C 扩展日志之前，必须先创建表架构，以便 Athena 可以读取日志数据。

要在 Athena 中为 W3C 扩展日志创建表

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 将类似以下内容的 DDL 语句粘贴到 Athena 控制台中，并注意以下几点：
   a. 在示例中添加或删除列，以便与要查询的日志中的字段对应。
   b. W3C 扩展日志文件格式的列名称包含连字符 (-)。然而，根据 Athena 命名约定，示例 `CREATE TABLE` 语句将用下划线 (_) 替换连字符。
   c. 要指定空格分隔符，请使用 `FIELDS TERMINATED BY ' '`。
   d. 修改 `LOCATION 's3://DOC-EXAMPLE-BUCKET/w3c-log-folder/'` 中的值以指向您在 Amazon S3 中的 W3C 扩展日志。

```
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. 在 Athena 控制台中运行查询以注册 `iis_w3c_logs` 表。查询完成后，调查结果准备就绪，可供您从 Athena 查询。

示例 W3C 扩展日志选择查询

以下示例查询从表 `iis_w3c_logs` 中选择了请求的日期、时间、请求目标和用时。`WHERE` 子句筛选条件，用于 HTTP 方法为 `GET`，以及 HTTP 状态代码为 `200`（成功）的情况。

```
SELECT date_col, time_col, cs_uri_stem, time_taken
FROM iis_w3c_logs
WHERE cs_method = 'GET' AND sc_status = '200'
```

下图显示了 Athena 查询编辑器中的查询结果。

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

合并日期和时间字段

以空格分隔的 date 和 time 字段是日志源数据中的单独条目，但您可以根据需要将它们合并到时间戳中。在 SELECT 或者 CREATE TABLE AS SELECT 查询中使用 concat() 和 date_parse() 函数来连接日期和时间列并将其转换为时间戳格式。以下示例使用 CTAS 查询创建一个新表，其中包含 derived_timestamp 列。

```
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
```

创建表后，您可以直接查询新的时间戳列，如以下示例所示。

```
SELECT derived_timestamp, cs_uri_stem, time_taken
FROM iis_w3c_logs_w_timestamp
WHERE cs_method = 'GET' AND sc_status = '200'
```

下图显示了查询的结果。

Results

	derived_timestamp	cs_uri_stem	time_taken
1	2020-01-19 22:48:39.000	/default.html	157
2	2020-01-19 22:49:40.000	/index.html	164
3	2020-01-19 22:50:12.000	/image.gif	358
4	2020-01-19 22:51:44.000	/faq.html	288

IIS 日志文件格式

与 W3C 扩展格式不同，IIS 日志文件格式有一组固定的字段，并包含逗号作为分隔符。LazySimpleSerDe 将逗号视为分隔符，逗号后的空格视为下一个字段的开头。

以下示例以 IIS 日志文件格式显示示例数据。

```
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, -,
```

在 Athena 中为 IIS 日志文件创建表

要在 Amazon S3 中查询 IIS 日志文件格式日志，请首先创建一个表架构，以便 Athena 可以读取日志数据。

要在 Athena 中为 IIS 日志文件格式日志创建表

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 将以下 DDL 语句粘贴到 Athena 控制台中，并注意以下几点：
   a. 要指定逗号分隔符，请使用 `FIELDS TERMINATED BY ','`。
   b. 修改 LOCATION 's3://DOC-EXAMPLE-BUCKET/*iis-log-file-folder*/' 中的值以指向 Amazon S3 中的 IIS 日志格式日志文件。

```
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. 在 Athena 控制台中运行查询以注册 `iis_format_logs` 表。查询完成后，调查结果准备就绪，可供您从 Athena 查询。

IIS 日志格式选择查询示例

以下示例查询从表 `iis_format_logs` 中选择了请求日期、请求时间、请求目标和用时（以毫秒为单位）。WHERE 子句筛选条件，用于请求类型为 GET，以及 HTTP 状态代码为 200（成功）的情况。在查询中，请注意，在 `' GET'` 和 `' 200'` 中需要前导空白才能成功查询。

```
SELECT request_date, request_time, target_of_operation, time_taken_millisec
FROM iis_format_logs
WHERE request_type = ' GET' AND service_status_code = ' 200'
```

下图显示了示例数据查询的结果。

Results

	request_date	request_time	target_of_operation	time_taken_millisec
1	2020-02-24	22:48:38	/index.htm	254
2	2020-02-24	22:48:39	/about.html	147
3	2020-02-24	22:48:40	/image.png	170
4	2020-02-24	22:48:41	/intro.htm	125

NCSA 日志文件格式

IIS 还使用 NCSA 日志记录格式，该格式具有固定数量的 ASCII 文本格式的字段，以空格分隔。该结构与用于 Apache 访问日志的常用日志格式类似。NCSA 常用日志数据格式中的字段包括客户端 IP 地址、客户端 ID（通常不使用）、域\用户 ID、接收请求的时间戳、客户端请求的文本、服务器状态代码以及返回给客户端的对象的大小。

以下示例显示了 IIS 所记录的 NCSA 常用日志格式的数据。

```
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
```

在 Athena 中为 IIS NCSA 日志创建表

对于您的 `CREATE TABLE` 语句，则可以使用 Grok SerDe 和一个类似于 Apache Web 服务器日志模式的 grok 模式。与 Apache 日志不同，grok 模式使用 `%{DATA:user_id}` 而不是 `%{USERNAME:user_id}` 作为第三个字段来考虑 `domain\user_id` 中反斜杠的存在。有关使用 Grok SerDe 的更多信息，请参阅《AWS Glue 开发人员指南https://docs.aws.amazon.com/glue/latest/dg/custom-classifier.html#custom-classifier-grok》中的 编写 Grok 自定义分类器。

要在 Athena 中为 IIS NCSA Web 服务器日志创建表

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 将以下 DDL 语句粘贴到 Athena 查询编辑器中。修改 LOCATION 's3://DOC-EXAMPLE-BUCKET/*iis-ncsa-logs*/' 中的值以指向 Amazon S3 中的 IIS NCSA 日志。

```
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. 在 Athena 控制台中运行查询以注册 iis_ncsa_logs 表。查询完成后，调查结果准备就绪，可供您从 Athena 查询。

IIS NCSA 日志的选择查询示例

Example – 筛选 404 错误

以下的示例查询从 iis_ncsa_logs 表中选择了请求接收时间、客户端请求的文本以及服务器状态代码。HTTP 状态代码 404（未找到页面）的 WHERE 子句筛选条件。

```
SELECT request_received_time, client_request, server_status
FROM iis_ncsa_logs
```

```
WHERE server_status = '404'
```

下图显示了 Athena 查询编辑器中的查询结果。

Results

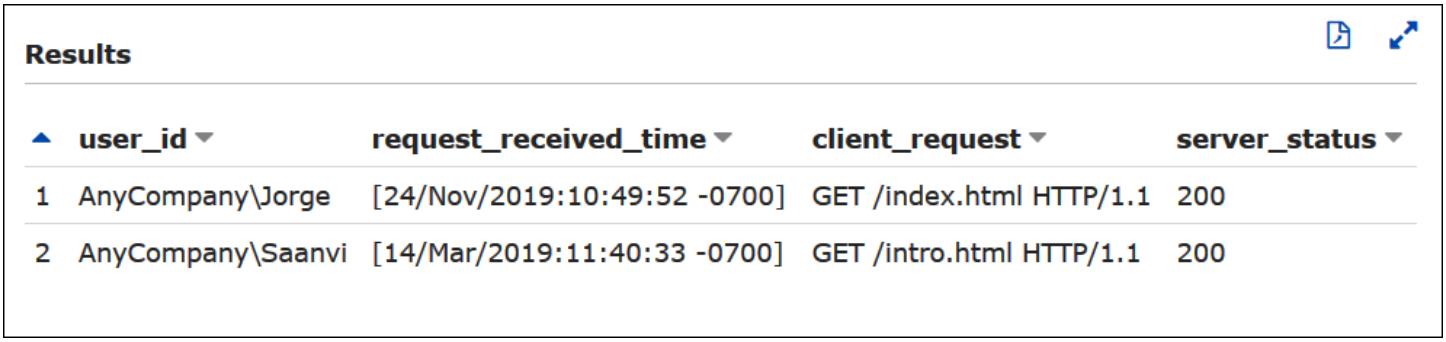
	request_received_time	client_request	server_status
1	[11/Jan/2020:11:40:11 -0700]	GET /image.png HTTP/1.1	404
2	[15/Feb/2019:10:12:22 -0700]	GET /favicon.ico HTTP/1.1	404

Example – 筛选来自特定域的成功请求

以下的示例查询从 `iis_ncsa_logs` 表中选择了用户 ID、请求接收时间、客户端请求的文本以及服务器状态代码。`WHERE` 子句筛选来自 AnyCompany 域中用户且具有 HTTP 状态代码 `200`（成功）的请求。

```
SELECT user_id, request_received_time, client_request, server_status
FROM iis_ncsa_logs
WHERE server_status = '200' AND user_id LIKE 'AnyCompany%'
```

下图显示了 Athena 查询编辑器中的查询结果。



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

使用 Athena ACID 事务

术语“ACID 事务”是指确保数据库事务中数据完整性的一组属性（原子性、一致性、隔离性和持久性）。ACID 事务使多个用户能够以原子方式并发可靠地添加和删除 Amazon S3 对象，同时通过维持针对数据湖查询的读取一致性，来隔离任何现有查询。Athena ACID 事务在 Athena SQL 数据操作语言（DML）中添加了对插入、删除、更新和时间旅行操作的单表支持。您和多个并发用户可以使用

Athena ACID 事务对 Amazon S3 数据进行可靠的行级修改。Athena 事务会自动管理锁定语义和协调，无需使用自定义记录锁定解决方案。

Athena ACID 事务和熟悉的 SQL 语法简化了对业务和监管数据的更新。例如，要响应数据擦除请求，您可以执行 SQL `DELETE` 操作。要手动更正记录，您可以使用单个 `UPDATE` 语句。要恢复最近删除的数据，您可以使用 `SELECT` 语句发出时间旅行查询。

由于它们都基于共享表格格式构建，因此 Athena ACID 事务与其它同样支持共享表格格式的服务和引擎兼容，例如 Amazon EMR 和 Apache Spark。

可以通过 Athena 控制台、API 操作以及 ODBC 和 JDBC 驱动程序访问 Athena 事务。

主题

查询 Linux Foundation Delta Lake 表

Linux Foundation Delta Lake 是一种用于大数据分析的表格格式。您可以使用 Amazon Athena 直接读取存储在 Amazon S3 中的 Delta Lake 表，无需生成清单文件或运行 `MSCK REPAIR` 语句。

Delta Lake 格式会存储每个数据文件每列的最小值和最大值。Athena 实施利用该信息在谓词上启用文件跳过，以消除不需要的文件。

注意事项和限制

Athena 中的 Delta Lake 支持具有以下注意事项和限制：

- 仅包含 AWS Glue 目录的表 – 仅通过注册到 AWS Glue 的表支持原生 Delta Lake。如果 Delta Lake 表已注册到其他元存储，您仍可保留该表并将其视为主元存储。由于 Delta Lake 元数据存储在文件系统（例如 Amazon S3）中而不是元存储中，因此 Athena 只需 AWS Glue 中的位置属性即可从 Delta Lake 表中进行读取。
- V3 engine only（仅限 V3 引擎）– 仅支持在 Athena 引擎版本 3 上进行 Delta Lake 查询。您必须确保创建的工作组配置为使用 Athena 引擎版本 3。
- Delta Lake 读取器版本 – 支持最高版本 3 的 Delta Lake 读取器协议。
- 列映射和 timestampNtz – Delta 列映射，它允许 Delta 表列和底层 Parquet 文件列使用不同的名称，并且支持不带时区的时间戳 (timestampNtz)。

- No time travel support（不支持时间旅行）– 不支持使用 Delta Lake 时间旅行功能的查询。
- Read only（只读）– 不支持编写 UPDATE、INSERT、或 DELETE 等 DML 语句。
- Lake Formation 支持 – Lake Formation 集成可用于架构与 AWS Glue 同步的 Delta Lake 表。有关更多信息，请参阅《AWS Lake Formation 开发人员指南》中的 AWS Lake Formation 与 Amazon Athena 一起使用和为 Delta Lake 表设置权限。
- Limited DDL support（有限的 DDL 支持）– 支持以下 DDL 语句：CREATE EXTERNAL TABLE、SHOW COLUMNS、SHOW TBLPROPERTIES、SHOW PARTITIONS、SHOW CREATE TABLE 和 DESCRIBE。有关使用 CREATE EXTERNAL TABLE 语句的信息，请参阅 开始使用 一节。
- 不支持跳过 S3 Glacier 对象 – 如果 Linux Foundation Delta Lake 表中的对象属于 Amazon S3 Glacier 存储类，将 `read_restored_glacier_objects` 表属性设置为 `false` 则无效。

  例如，假设发出以下命令：

  ```
  ALTER TABLE table_name SET TBLPROPERTIES ('read_restored_glacier_objects' = 'false')
  ```

  对于 Iceberg 和 Delta Lake 表，该命令会生成错误 Unsupported table property key: read_restored_glacier_objects。对于 Hudi 表，ALTER TABLE 命令不会产生错误，但是 Amazon S3 Glacier 对象仍无法跳过。在 ALTER TABLE 命令之后运行 SELECT 查询会继续返回所有对象。

支持的非分区列数据类型

对于非分区列，除 CHAR 外，支持 Athena 支持的所有数据类型（Delta Lake 协议本身不支持 CHAR）。支持的数据类型包括：

```
boolean
tinyint
smallint
integer
bigint
double
float
decimal
varchar
string
binary
date
timestamp
```

```
array
map
struct
```

支持的分区列数据类型

对于分区列，Athena 支持具有以下数据类型的表：

```
boolean
integer
smallint
tinyint
bigint
decimal
float
double
date
timestamp
varchar
```

有关 Athena 中数据类型的更多信息，请参阅 Amazon Athena 中的数据类型。

开始使用

要进行查询，Delta Lake 表必须存在于 AWS Glue 中。如果表位于 Amazon S3 但不在 AWS Glue 中，请使用以下语法运行 CREATE EXTERNAL TABLE 语句。如果表已存在于 AWS Glue 中（例如，因为您正在将 Apache Spark 或其他引擎与 AWS Glue 结合使用），则可以跳过此步骤。

```
CREATE EXTERNAL TABLE
  [db_name.]table_name
  LOCATION 's3://DOC-EXAMPLE-BUCKET/your-folder/'
  TBLPROPERTIES ('table_type' = 'DELTA')
```

请注意，列定义、SerDe 库和其他表属性会省略。与传统的 Hive 表不同，Delta Lake 表元数据是从 Delta Lake 事务日志中推断出来的，并直接同步到 AWS Glue。

> **Note**
>
> 对于 Delta Lake 表，不允许包含除 LOCATION 和 table_type 属性外的 CREATE TABLE 语句。

读取 Delta Lake 表

要查询 Delta Lake 表，请使用标准 SQL SELECT 语法：

```
[ 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 ] ]
```

有关 SELECT 语句的更多信息，请参阅 Athena 文档中的 SELECT。

Delta Lake 格式会存储每个数据文件每列的最小值和最大值。Athena 利用该信息在谓词上启用文件跳过，以消除不需要的文件。

同步 Delta Lake 元数据

如果您使用 Athena 创建 Delta Lake 表，Athena 会将表元数据（包括架构、分区列和表属性）同步到 AWS Glue。随着时间推移，该元数据可能会中断与事务日志中基础表元数据的同步。要让表保持最新状态，您可以选择以下选项之一：

- 为 Delta Lake 表使用 AWS Glue 爬网程序。有关更多信息，请参阅 AWS 大数据博客中的带 AWS Glue 爬网程序的原生 Delta Lake 表支持的简介和《AWS Glue 开发人员指南》中的计划 AWS Glue 爬网程序。
- 在 Athena 中删除并重新创建表格。
- 使用 SDK、CLI 或 AWS Glue 控制台手动更新 AWS Glue 中的架构。

请注意：以下功能要求您的 AWS Glue 架构始终具有与事务日志相同的架构：

- Lake Formation
- 视图
- 行和列筛选器

如果您的工作流程不需要任何此功能，并且您不希望保持这种兼容性，则可以在 Athena 中使用 CREATE TABLE DDL，然后将 Amazon S3 路径作为 SerDe 参数添加到 AWS Glue 中。

使用 Athena 和 AWS Glue 控制台创建 Delta Lake 表

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 在 Athena 查询编辑器中，使用以下 DDL 创建您的 Delta Lake 表。请注意：使用此方法时，TBLPROPERTIES 的值必须为 `'spark.sql.sources.provider' = 'delta'`，而不是 `'table_type' = 'delta'`。

   请注意：当您使用 Apache Spark（Athena for Apache Spark）或大多数其他引擎创建表时，会插入相同的架构（其中包含一个名为 `col` 的 `array<string>` 类型列）。

   ```
   CREATE EXTERNAL TABLE
      [db_name.]table_name(col array<string>)
      LOCATION 's3://DOC-EXAMPLE-BUCKET/your-folder/'
      TBLPROPERTIES ('spark.sql.sources.provider' = 'delta')
   ```
3. 通过 https://console.aws.amazon.com/glue/ 打开 AWS Glue 控制台。
4. 在导航窗格中，选择数据目录、表。
5. 在表的列表中，选择表的链接。
6. 在表格页面上，选择操作、编辑表格。
7. 在 Serde 参数部分，添加带有值 **s3://DOC-EXAMPLE-BUCKET/*your-folder*/** 的键 **path**。
8. 选择保存。

其他 资源

有关将 Delta Lake 表与 AWS Glue 一起使用以及使用 Athena 查询表的讨论，请参阅 AWS 大数据博客中的使用开源 Delta Lake 和 AWS Glue 处理 UPSERT 数据操作。

使用 Athena 查询 Apache Hudi 数据集

Apache Hudi是一个开源数据管理框架，可简化增量递增数据的处理。记录级别的插入、更新、加入和删除操作的处理更加精细，从而降低了开销。Upsert 指的是将记录插入到现有数据集中（如果它们不存在）或对数据集进行更新（如果它们存在）的功能。

Hudi 处理数据插入和更新事件，而不会创建许多可能导致分析性能问题的小文件。Apache Hudi 会自动跟踪更改并合并文件，以便它们保持最佳大小。这样就不需要构建自定义解决方案来监控许多小文件并将其重写为大文件以减少其数量。

Hudi 数据集适用于以下使用案例：

- 遵守隐私法规，如《一般数据保护条例》（GDPR）和《加州消费者隐私法》（CCPA），这些法规将强制执行人们删除个人信息的权力或更改其数据使用方式。
- 处理来自传感器和其它需要特定数据插入和更新事件的物联网 (IoT) 设备的流数据。
- 实施更改数据捕获 (CDC) 系统。

Hudi 管理的数据集使用开放存储格式存储在 Amazon S3 中。目前，Athena 可以读取压缩的 Hudi 数据集，但不能写 Hudi 数据。Athena 最高可支持将 Hudi 版本 0.8.0 与 Athena 引擎版本 2 结合使用，以及将 Hudi 版本 0.14.0与 Athena 引擎版本 3 结合使用。这可能随时更改。Athena 不保证与使用更高版本的 Hudi 创建的表具有读取兼容性。有关 Athena 引擎版本控制的更多信息，请参阅 Athena 引擎版本控制。有关 Hudi 功能和版本控制的更多信息，请参阅 Apache 网站上的 Hudi 文档。

Hudi 数据集表类型

Hudi 数据集可以采用以下类型之一：

- 写入时复制 (CoW) – 数据以列状格式存储 (Parquet)，并且每次更新都会在写入过程中创建一个新版本的文件。
- 读取时合并 (MOR) – 数据使用列式 (Parquet) 和基于行 (Avro) 的格式的组合进行存储。更新记录到基于行的 `delta` 文件中，并根据需要进行压缩以创建新版本的列式文件。

对于 CoW 数据集，每次更新记录时，包含该记录的文件都会使用更新后的值进行重写。对于 MoR 数据集，每次进行更新时，Hudi 仅写入已更改记录对应的行。MoR 更适合写入或更改繁重而读取量较少的工作负载。CoW 更适合更改频率较低但读取量繁重的工作负载。

Hudi 提供三个查询类型用于访问数据：

- 快照查询 – 该查询用于查看截至给定提交或压缩操作时表的最新快照。对于 MOR 表，快照查询通过在查询时合并最新文件切片的基本文件和增量文件来显示表的最新状态。
- 增量查询 – 查询只能看到自给定提交/压缩以来，写入表的新数据。这有效地提供了更改流以启用增量数据管道。
- 读取优化查询 – 对于 MOR 表，查询将看到最新压缩的数据。对于 CoW 表，查询将看到最新提交的数据。

下表显示了适用于每种表类型的可用 Hudi 查询类型。

表类型	可用的 Hudi 查询类型
写入时复制	快照、增量
读取时合并	快照、增量、读取优化

目前，Athena 支持快照查询和读取优化查询，但不支持增量查询。在 MOR 表上，所有在读取优化查询中显示的 数据均已压缩。这提供了良好的性能，但不包括最新的增量提交。快照查询包含最新的数据，但会但产生一些计算开销，这使得这些查询的性能降低。

有关在表和查询类型之间权衡的更多信息，请参阅 Apache Hudi 文档中的表和查询类型。

Hudi 术语变更：视图现在为查询

从发行版 0.5.1 开始，Apache Hudi 改变了一些术语。以前的视图在较新的版本中称为查询。下表总结了旧新术语之间的变更。

旧术语	新术语
CoW：读取优化视图 MoR：实时视图	快照查询
增量视图	增量查询
MoR 读取优化视图	读取优化的查询

引导启动操作中的表

从 Apache Hudi 版本 0.6.0 开始，引导启动操作功能可为现有的 Parquet 数据集提供更好的性能。引导操作只能生成元数据，使数据集保持原位，而不是重写数据集。

您可以使用 Athena 从引导启动操作中查询表，就像基于 Amazon S3 中数据的其他表一样。在您的 `CREATE TABLE` 语句中，在 `LOCATION` 子句指定 Hudi 表路径。

有关在 Amazon EMR 中使用引导启动操作创建 Hudi 表的更多信息，请参阅 AWS 大数据博客文章：New features from Apache Hudi available in Amazon EMR（在 Amazon EMR 中可用的 Apache Hudi 新功能）。

Hudi 元数据列表

Apache Hudi 有一个元数据表，其中包含用于提高性能的索引功能，例如文件列表、使用列统计数据跳过数据以及基于布隆筛选器的索引。

在这些功能中，Athena 目前仅支持文件列表索引。文件列表索引通过从维护分区到文件映射的索引中获取信息，从而消除了“列表文件”之类的文件系统调用。这样就无需以递归方式列出表路径下的每个分区即可查看文件系统。当您处理大型数据集时，这种索引可以大大减少写入和查询期间获取文件列表时出现的延迟。还可以避免诸如 Amazon S3 `LIST` 调用的请求限制节流之类的瓶颈。

> **Note**
>
> Athena 目前不支持数据跳过或布隆筛选器索引。

启用 Hudi 元数据表

默认情况下，禁用基于元数据表的文件列表。要启用 Hudi 元数据表和相关的文件列表功能，请将 `hudi.metadata-listing-enabled` 表格属性设置为 `TRUE`。

示例

以下 `ALTER TABLE SET TBLPROPERTIES` 示例在示例 `partition_cow` 表上启用元数据表。

```
ALTER TABLE partition_cow SET TBLPROPERTIES('hudi.metadata-listing-enabled'='TRUE')
```

注意事项和限制

- Athena 不支持增量查询。
- Athena 不支持对 Hudi 数据执行 CTAS 或者 INSERT INTO。如果您希望 Athena 支持编写 Hudi 数据集，请将反馈发送至 <athena-feedback@amazon.com>。

  有关编写 Hudi 数据的更多信息，请参阅以下资源：

  - 《Amazon EMR 版本指南》中的使用 Hudi 数据集。
  - Apache Hudi 文档中的写入数据。
- 不支持在 Athena 的 Hudi 表上使用 MSCK REPAIR TABLE。如果您需要加载未在 AWS Glue 中创建的 Hudi 表，请使用 ALTER TABLE ADD PARTITION。
- 不支持跳过 S3 Glacier 对象 – 如果 Apache Hudi 表中的对象属于 Amazon S3 Glacier 存储类，将 `read_restored_glacier_objects` 表属性设置为 `false` 则无效。

例如，假设发出以下命令：

```
ALTER TABLE table_name SET TBLPROPERTIES ('read_restored_glacier_objects' = 'false')
```

对于 Iceberg 和 Delta Lake 表，该命令会生成错误 Unsupported table property key: read_restored_glacier_objects。对于 Hudi 表，ALTER TABLE 命令不会产生错误，但是 Amazon S3 Glacier 对象仍无法跳过。在 ALTER TABLE 命令之后运行 SELECT 查询会继续返回所有对象。

视频

以下视频展示了如何使用 Amazon Athena 查询基于 Amazon S3 的数据湖中的读取优化 Apache Hudi 数据集。

使用 Amazon Athena 查询 Apache Hudi 数据集

创建 Hudi 表

本节提供了 Athena 中针对 Hudi 数据的分区表和非分区表的 CREATE TABLE 语句的示例。

如果您有已在 AWS Glue 中创建的 Hudi 表，您可以直接在 Athena 中查询它们。当您在 Athena 中创建分区 Hudi 表时，您必须运行 ALTER TABLE ADD PARTITION 以加载 Hudi 数据，然后再查询这些数据。

写入时复制（CoW）创建表示例

未分区 CoW 表

以下示例在 Athena 中创建了一个未分区的 CoW 表。

```
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/'
```

分区 CoW 表

以下示例在 Athena 中创建了一个已分区的 CoW 表。

```
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/'
```

以下 ALTER TABLE ADD PARTITION 示例将两个分区添加到了示例 partition_cow 表。

```
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/'
```

读取时合并（MoR）创建表示例

Hudi 在元数据仓中为 MoR 创建了两个表：一个用于快照查询的表，一个用于读取优化查询的表。两个表都可查询。在 0.5.1 之前的 Hudi 版本中，用于读优化查询的表具有您在创建表时指定的名称。从 Hudi 版本 0.5.1 开始，表名将默认以 `_ro` 为后缀。用于快照查询的表的名称是指定的名称，其中附加了 `_rt`。

未分区的读取时合并（MoR）表

以下示例在 Athena 中创建了一个未分区的 MoR 表用于读取优化查询。请注意，读取优化查询使用输入格式 `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/'
```

以下示例在 Athena 中创建了一个未分区的 MoR 表用于快照查询。对于快照查询，请使用输入格式 `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/'
```

已分区的读取时合并（MoR）表

以下示例在 Athena 中创建了一个已分区的 MoR 表用于读取优化查询。

```
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/'
```

以下 ALTER TABLE ADD PARTITION 示例将两个分区添加到了示例 partition_mor 表。

```
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/'
```

以下示例在 Athena 中创建了一个已分区的 MoR 表用于快照查询。

```
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/'
```

同样，以下 `ALTER TABLE ADD PARTITION` 示例将两个分区添加到了示例 `partition_mor_rt` 表。

```
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/'
```

其他 资源

- 要了解如何使用 AWS Glue 自定义连接器和 AWS Glue 2.0 任务来创建可以使用 Athena 查询的 Apache Hudi 表，请参阅 AWS 大数据博客文章使用 AWS Glue 自定义连接器写入 Apache Hudi 表。

- 有关使用 Apache Hudi、AWS Glue 和 Amazon Athena 为数据湖构建数据处理框架的文章，请参阅 AWS Big Data Blog 中的 Simplify operational data processing in data lakes using AWS Glue and Apache Hudi。

使用 Apache Iceberg 表

Athena 支持对 Apache Iceberg 表进行读取、时间旅行、写入和 DDL 查询，这些表对数据和元数据仓分别使用 Apache Parquet 格式和 AWS Glue 目录。

Apache Iceberg 是适用于超大型分析数据集的开放表格式。Iceberg 以表的形式管理大量文件，并支持现代分析数据湖操作，例如记录级插入、更新、删除和时间旅行查询。Iceberg 规范支持表无缝发展，例如架构和分区发展，且其设计针对在 Amazon S3 上的使用进行了优化。Iceberg 还有助于在并发写入场景下保证数据的正确性。

有关 Apache Iceberg 的更多信息，请参阅 https://iceberg.apache.org/。

注意事项和限制

Athena 对 Iceberg 表的支持具有以下考虑和限制：

- Iceberg 版本支持 – Athena 支持 Apache Iceberg 版本 1.4.2。
- 仅限包含 AWS Glue 目录的表 – Athena 仅支持根据由开源 Glue 目录实施所定义的规范在 AWS Glue 目录上创建的 Iceberg 表。
- AWS Glue 仅支持表锁定 – 与支持插件自定义锁的开源 Glue 目录实施不同，Athena 仅支持 AWS Glue 乐观锁。使用 Athena 修改具有任何其他锁实施的 Iceberg 表将丢失潜在数据并中断事务。
- 支持的文件格式 – Athena 中的 Iceberg 文件格式支持取决于 Athena 引擎版本，如下表所示。

Athena 引擎版本	Parquet	ORC	Avro
2	是	否	否
3	支持	是	是

- Iceberg v2 表 – Athena 仅在 Iceberg v2 表上进行创建和操作。有关 v1 和 v2 表之间的区别，请参阅 Apache Iceberg 文档中的格式版本更改。
- 显示没有时区的时间类型 – 没有时区类型的时间和时间戳以 UTC 格式显示。如果在时间列的筛选条件表达式中未指定时区，则使用 UTC。

- 时间戳相关的数据精度 – 虽然 Iceberg 对时间戳数据类型支持微秒精度，但 Athena 仅对读写操作中的时间戳支持毫秒精度。在与时间相关的列中，Athena 只对在手动压缩操作过程中重写的数据保留毫秒精度。
- 不支持的操作 – Iceberg 表格不支持以下 Athena 操作。
  - ALTER TABLE SET LOCATION
- 视图 – 使用 `CREATE VIEW` 创建 Athena 视图，如 使用视图 中所述。如果您想使用 Iceberg 视图规范创建视图，请联系 athena-feedback@amazon.com。
- AWS Lake Formation 中不支持 TTF 管理命令 — 尽管您可以使用 Lake Formation 管理 Apache Iceberg、Apache Hudi 和 Linux Foundation Delta Lake 等 TransactionTable 格式（TTF）的读取访问权限，但您不能使用 Lake Formation 管理诸如 `VACUUM`、`MERGE`、`UPDATE` 或 `OPTIMIZE` 使用这些表格格式之类的操作的权限。有关 Lake Formation 与 Athena 集成的更多信息，请参阅《AWS Lake Formation 开发人员指南》中的将 AWS Lake Formation 与 Amazon Athena 一起使用。
- 按嵌套字段分区 — 不支持按嵌套字段进行分区。尝试这样做会生成消息 NOT_SUPPORTED：不支持按嵌套字段进行分区：*`column_name.nested_field_name`*。
- 不支持跳过 S3 Glacier 对象 – 如果 Apache Iceberg 表中的对象属于 Amazon S3 Glacier 存储类，将 `read_restored_glacier_objects` 表属性设置为 `false` 则无效。

  例如，假设发出以下命令：

  ```
  ALTER TABLE table_name SET TBLPROPERTIES ('read_restored_glacier_objects' = 'false')
  ```

  对于 Iceberg 和 Delta Lake 表，该命令会生成错误 Unsupported table property key: read_restored_glacier_objects。对于 Hudi 表，`ALTER TABLE` 命令不会产生错误，但是 Amazon S3 Glacier 对象仍无法跳过。在 `ALTER TABLE` 命令之后运行 `SELECT` 查询会继续返回所有对象。

若您希望 Athena 支持特定功能，请将反馈发送至 athena-feedback@amazon.com。

主题

- 优化 Iceberg 表
- Athena 中支持的 Iceberg 表数据类型。
- Iceberg 表上的其他 Athena 操作
- 其他 资源

创建 Iceberg 表

要创建在 Athena 中使用的 Iceberg 表，您可以使用本页中记录的 CREATE TABLE 语句，也可以使用 AWS Glue 爬网程序。

使用 CREATE TABLE 语句

Athena 创建 Iceberg v2 表。有关 v1 和 v2 表之间的区别，请参阅 Apache Iceberg 文档中的格式版本更改。

Athena CREATE TABLE 创建没有数据的 Iceberg 表。若表使用 Iceberg 开源 Glue目录，则您可以直接从外部系统（例如 Apache Spark）查询表。您无需创建外部表。

> **Warning**
>
> 运行 CREATE EXTERNAL TABLE 会导致出现错误消息 ICEBERG 表类型不支持外部关键字。

要从 Athena 创建 Iceberg 表，请将 'table_type' 子句中的 'ICEBERG' 表属性设置为 TBLPROPERTIES，如下语法摘要所示。

```
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] )
```

有关可以在 Iceberg 表中查询的数据类型的信息，请参阅 Athena 中支持的 Iceberg 表数据类型。。

分区

要创建带有分区的 Iceberg 表，请使用 PARTITIONED BY 语法。必须首先在列声明中指定用于分区的列。PARTITIONED BY 子句中不得包括列类型。还可以在 CREATE TABLE 语法中定义分区转换。要指定多个列进行分区，请使用逗号 (,) 字符分隔列，如以下示例所示。

```
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' )
```

下表列出了可用的分区转换函数。

函数	描述	支持的类型
year(ts)	按年分区	date, timestamp
month(ts)	按月分区	date, timestamp
day(ts)	按天分区	date, timestamp
hour(ts)	按小时分区	timestamp
bucket(*N*, col)	按哈希值 mod *N* 存储桶分区。这与 Hive 表的哈希分桶概念相同。	int, long, decimal, date, timestamp ，string, binary
truncate(*L*, col)	按值截断为 *L* 分区	int, long, decimal, string

Athena 支持 Iceberg 的隐藏分区。有关更多信息，请参阅 Apache Iceberg 文档中的 Iceberg 的隐藏分区。

表属性

本部分介绍可以在 CREATE TABLE 语句的 TBLPROPERTIES 子句中指定为键值对的表属性。Athena 仅允许表属性中预定义的键值对列表用于创建或更改 Iceberg 表。下表列出了可以指定的表属性。有关压缩选项的更多信息，请参阅本文档中的 优化 Iceberg 表。如果您希望 Athena 支持特定的开源表配置属性，请将反馈发送至 athena-feedback@amazon.com。

format

描述	文件数据格式

允许的属性值	支持的文件格式和压缩组合因 Athena 引擎版本而异。有关更多信息，请参阅 不同文件格式支持的 Iceberg 表压缩。
默认值	parquet

write_compression

描述	文件压缩编码器
允许的属性值	支持的文件格式和压缩组合因 Athena 引擎版本而异。有关更多信息，请参阅 不同文件格式支持的 Iceberg 表压缩。
默认值	默认写入压缩因 Athena 引擎版本而异。有关更多信息，请参阅 不同文件格式支持的 Iceberg 表压缩。

optimize_rewrite_data_file_threshold

描述	数据优化的特定配置。若需要优化的数据文件少于给定阈值，则不会重写这些文件。这将允许积累更多的数据文件以生成更接近目标大小的文件，并跳过不必要的计算以节省成本。
允许的属性值	正数。必须小于 50。
默认值	5

optimize_rewrite_delete_file_threshold

描述	数据优化的特定配置。如果与数据文件关联的删除文件少于阈值，则不会重写该数据文件。这将允许为每个数据文件累积更多的删除文件，以节省成本。
允许的属性值	正数。必须小于 50。
默认值	2

vacuum_min_snapshots_to_keep

描述	在表的主分支上保留的最小快照数。 此值优先于 vacuum_max_snapshot_age_seconds 属性。如果剩余的最小快照比 vacuum_max_snapshot_age_seconds 指定的时间长，则会保留这些快照，并忽略 vacuum_max_snapshot_age_seconds 的值。
允许的属性值	正数。
默认值	1

vacuum_max_snapshot_age_seconds

描述	要在主分支上保留的最大快照期限。如果 vacuum_min_snapshots_to_keep 指定的剩余最小快照比指定的时间长，则忽略此值。此表行为属性对应于 Apache Iceberg 配置中的 history.expire.max-snapshot-age-ms 属性。
允许的属性值	正数。
默认值	432000 秒（5 天）

vacuum_max_metadata_files_to_keep

描述	要在表的主分支上保留的先前元数据文件的最大数量。
允许的属性值	正数。
默认值	100

CREATE TABLE 语句示例

以下示例将创建一个包含三列的 Iceberg 表。

```
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)

有关使用 CREATE TABLE AS 语句创建 Iceberg 表的信息，请参阅 CREATE TABLE AS，尤其是 CTAS 表属性 部分。

使用 AWS Glue 爬网程序

您可以使用 AWS Glue 爬网程序将您的 Iceberg 表自动注册到 AWS Glue Data Catalog。如果您想从其他 Iceberg 目录迁移，可以创建和安排 AWS Glue 爬网程序，并提供 Iceberg 表所在的 Amazon S3 路径。您可以指定 AWS Glue 爬网程序可以遍历的 Amazon S3 路径的最大深度。安排 AWS Glue 爬网程序后，爬网程序会提取架构信息，并在每次运行时使用架构更改更新 AWS Glue Data Catalog。AWS Glue 爬网程序支持跨快照合并架构，并更新 AWS Glue Data Catalog 中最新的元数据文件位置。有关更多信息，请参阅 AWS Glue 中的数据目录和爬网程序。

管理 Iceberg 表

Athena 支持对 Iceberg 表执行以下表 DDL 操作。

ALTER TABLE RENAME

重命名表。

由于 Iceberg 表的表元数据存储在 Amazon S3 中，因此您可以更新 Iceberg 托管式表的数据库和表名，而不会影响底层表信息。

摘要

```
ALTER TABLE [db_name.]table_name RENAME TO [new_db_name.]new_table_name
```

示例

```
ALTER TABLE my_db.my_table RENAME TO my_db2.my_table2
```

ALTER TABLE SET PROPERTIES

向 Iceberg 表中添加属性并设置它们的分配值。

根据 Iceberg 规范，表属性存储在 Iceberg 表元数据文件中，而不是存储在 AWS Glue 中。Athena 不接受自定义表属性。请参阅 表属性 部分，了解允许的键值对。如果您希望 Athena 支持特定的开源表配置属性，请将反馈发送至 athena-feedback@amazon.com。

摘要

```
ALTER TABLE [db_name.]table_name SET TBLPROPERTIES ('property_name' =
 'property_value' [ , ... ])
```

示例

```
ALTER TABLE iceberg_table SET TBLPROPERTIES (
  'format'='parquet',
  'write_compression'='snappy',
  'optimize_rewrite_delete_file_threshold'='10'
)
```

ALTER TABLE UNSET PROPERTIES

从 Iceberg 表中删除现有属性。

摘要

```
ALTER TABLE [db_name.]table_name UNSET TBLPROPERTIES ('property_name' [ , ... ])
```

示例

```
ALTER TABLE iceberg_table UNSET TBLPROPERTIES ('write_compression')
```

DESCRIBE TABLE

描述表格信息。

摘要

```
DESCRIBE [FORMATTED] [db_name.]table_name
```

指定 FORMATTED 选项后，输出将显示其他信息（如表位置和属性）。

示例

```
DESCRIBE iceberg_table
```

DROP TABLE

删除 Iceberg 表。

> **Warning**
> 由于 Iceberg 表在 Athena 中被视为托管式表，因此删除 Iceberg 表也会删除表中的所有数据。

摘要

```
DROP TABLE [IF EXISTS] [db_name.]table_name
```

示例

```
DROP TABLE iceberg_table
```

SHOW CREATE TABLE

显示可用于在 Athena 中重新创建 Iceberg 表的 CREATE TABLE DDL 语句。如果 Athena 无法重现表结构（例如，由于在表中指定了自定义表属性），则将引发不支持错误。

摘要

```
SHOW CREATE TABLE [db_name.]table_name
```

示例

```
SHOW CREATE TABLE iceberg_table
```

SHOW TABLE PROPERTIES

显示 Iceberg 表的一个或多个表属性。仅显示 Athena 支持的表属性。

摘要

```
SHOW TBLPROPERTIES [db_name.]table_name [('property_name')]
```

示例

```
SHOW TBLPROPERTIES iceberg_table
```

查询 Iceberg 表元数据

在 SELECT 查询中，您可以在 *table_name* 之后使用以下属性来查询 Iceberg 表元数据：

- $files – 显示表的当前数据文件。
- $manifests – 显示表的当前文件清单。
- $history – 显示表的历史记录。
- $partitions – 显示表的当前分区。
- $snapshots – 显示表的快照。
- $refs – 显示表的引用。

语法

以下语句列出了 Iceberg 表的文件。

```
SELECT * FROM "dbname"."tablename$files"
```

以下语句列出了 Iceberg 表的清单。

```
SELECT * FROM "dbname"."tablename$manifests"
```

以下语句显示 Iceberg 表的历史记录。

```
SELECT * FROM "dbname"."tablename$history"
```

以下示例显示 Iceberg 表的分区。

```
SELECT * FROM "dbname"."tablename$partitions"
```

以下示例列出了 Iceberg 表的快照。

```
SELECT * FROM "dbname"."tablename$snapshots"
```

以下示例显示 Iceberg 表的引用。

```
SELECT * FROM "dbname"."tablename$refs"
```

不断变化的 Iceberg 表架构

Iceberg 架构更新是仅对元数据进行更改。执行架构更新时，不会更改任何数据文件。

Iceberg 格式支持以下架构发展更改：

- 添加 – 将新列添加到表或嵌套 `struct`。
- 删除 – 删除表或嵌套 `struct` 中的现有列。
- 重命名 – 重命名嵌套 `struct` 中的现有列或字段。
- 重新排序 – 更改列的顺序。
- 类提升 – 扩大列、`struct` 字段、`map` 键、`map` 值或 `list` 元素的类型。目前，Iceberg 表支持以下案例：
  - 整数到大整数
  - 浮点到双精度
  - 提高十进制类型的精度

ALTER TABLE ADD COLUMNS

向现有 Iceberg 表添加一列或多列。

摘要

```
ALTER TABLE [db_name.]table_name ADD COLUMNS (col_name data_type [,...])
```

示例

以下示例将 `string` 类型的 `comment` 列添加到 Iceberg 表中。

```
ALTER TABLE iceberg_table ADD COLUMNS (comment string)
```

以下示例将 `struct` 类型的 `point` 列添加到 Iceberg 表中。

```
ALTER TABLE iceberg_table
ADD COLUMNS (point struct<x: double, y: double>)
```

以下示例将作为结构数组的 `points` 列添加到 Iceberg 表中。

```
ALTER TABLE iceberg_table
ADD COLUMNS (points array<struct<x: double, y: double>>)
```

ALTER TABLE DROP COLUMN

从现有 Iceberg 表中删除列。

摘要

```
ALTER TABLE [db_name.]table_name DROP COLUMN col_name
```

示例

```
ALTER TABLE iceberg_table DROP COLUMN userid
```

ALTER TABLE CHANGE COLUMN

更改列的名称、类型、顺序或注释。

> **Note**
>
> 不支持 `ALTER TABLE REPLACE COLUMNS`。由于 `REPLACE COLUMNS` 删除所有列并添加新列，因此 Iceberg 不支持该语句。`CHANGE COLUMN` 是架构发展的首选语法。

摘要

```
ALTER TABLE [db_name.]table_name
  CHANGE [COLUMN] col_old_name col_new_name column_type
```

```
  [COMMENT col_comment] [FIRST|AFTER column_name]
```

示例

```
ALTER TABLE iceberg_table CHANGE comment blog_comment string AFTER id
```

SHOW COLUMNS

显示表中的列。

摘要

```
SHOW COLUMNS (FROM|IN) [db_name.]table_name
```

示例

```
SHOW COLUMNS FROM iceberg_table
```

查询 Iceberg 表数据并执行时间旅行

若要查询 Iceberg 数据集，请使用标准 SELECT 语句，如下所示。查询遵循 Apache Iceberg 格式 v2 规范，并对位置和平等删除执行读取时合并。

```
SELECT * FROM [db_name.]table_name [WHERE predicate]
```

为了优化查询时间，所有谓词都会向下推送到数据所在的位置。

时间旅行和版本旅行查询

每个 Apache Iceberg 表都维护所包含 Amazon S3 对象的版本控制清单。清单的早期版本可用于时间旅行和版本旅行查询。

Athena 中的时间旅行查询在 Amazon S3 中查询一致快照中截至指定日期和时间的历史数据。Athena 中的版本旅行查询在 Amazon S3 中查询指定快照 ID 以前的历史数据。

时间旅行查询

要运行时间旅行查询，请在 SELECT 语句中的表名称之后使用 FOR TIMESTAMP AS OF *timestamp*，如下例所示。

```
SELECT * FROM iceberg_table FOR TIMESTAMP AS OF timestamp
```

为旅行指定的系统时间可以是时间戳，也可以是带时区的时间戳。如果未指定，Athena 将该值视为以 UTC 时间表示的时间戳。

以下示例时间旅行查询选择指定日期和时间的 CloudTrail 数据。

```
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)
```

版本旅行查询

要执行版本旅行查询（即，查看指定版本以前的一致快照），请在 SELECT 语句中的表名称之后使用 FOR VERSION AS OF *version*，如下例所示。

```
SELECT * FROM [db_name.]table_name FOR VERSION AS OF version
```

version 参数是与 Iceberg 表版本关联的 bigint 快照 ID。

以下示例版本旅行查询选择指定版本的数据。

```
SELECT * FROM iceberg_table FOR VERSION AS OF 949530903748831860
```

> **Note**
>
> Athena 引擎版本 2 中的 FOR SYSTEM_TIME AS OF 和 FOR SYSTEM_VERSION AS OF 子句已替换为 Athena 引擎版本 3 中的 FOR TIMESTAMP AS OF 和 FOR VERSION AS OF 子句。

检索快照 ID

使用 Iceberg 提供的 Java SnapshotUtil 类可以检索 Iceberg 快照 ID，如下例所示。

```
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);
```

结合使用时间旅行和版本旅行

您可以在同一查询中使用时间旅行和版本旅行语法来指定不同的时间和版本控制条件，如以下示例所示。

```
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)
```

使用 Iceberg 表创建和查询视图

要在 Iceberg 表上创建和查询 Athena 视图，请使用 使用视图 中所述的 `CREATE VIEW` 视图。

例如：

```
CREATE VIEW view1 AS SELECT * FROM iceberg_table
```

```
SELECT * FROM view1
```

如果您想使用 Iceberg 视图规范创建视图，请联系 athena-feedback@amazon.com。

使用 Lake Formation 精细访问控制

Athena 引擎版本 3 支持使用 Iceberg 表进行 Lake Formation 精细访问控制，包括列级和行级安全访问控制。这种访问控制适用于时间旅行查询和已执行架构发展的表。有关更多信息，请参阅 Lake Formation 精细访问控制和 Athena 工作组。

如果您在 Athena 外部创建了 Iceberg 表，请使用 Apache Iceberg SDK 版本 0.13.0 或更高版本，以便在 AWS Glue Data Catalog 中填充 Iceberg 表列信息。如果 Iceberg 表不包含 AWS Glue 中的列信息，则可以使用 Athena ALTER TABLE SET PROPERTIES 语句或最新的 Iceberg SDK 修复表并更新 AWS Glue 中的列信息。

更新 Iceberg 表数据

Iceberg 表数据可以直接在 Athena 上使用 INSERT、UPDATE 和 DELETE 查询进行管理。每个数据管理事务都会生成一个新的快照，可以使用时间旅行查询该快照。UPDATE 和 DELETE 语句遵循 Iceberg 格式 v2 行级位置删除规范并强制执行快照隔离。

使用以下命令对 Iceberg 表执行数据管理操作。

INSERT INTO

将数据插入 Iceberg 表。Athena Iceberg INSERT INTO 收费与当前对外部 Hive 表进行 INSERT INTO 查询的收费一致，皆按扫描的数据量收费。要将数据插入到 Iceberg 表中，请使用以下语法，其中 *query* 可以是 VALUES (val1, val2, ...) 或 SELECT (col1, col2, …) FROM [*db_name*.]*table_name* WHERE *predicate*。有关 SQL 语法和语义的详细信息，请参阅 INSERT INTO。

```
INSERT INTO [db_name.]table_name [(col1, col2, …)] query
```

以下示例将值插入到表 iceberg_table 中。

```
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
```

删除

Athena Iceberg DELETE 将 Iceberg 位置删除文件写入表中。这称为读取时合并删除。与写入时复制删除不同，读取时合并删除效率更高，因为其不会重写文件数据。当 Athena 读取 Iceberg 数据时，将合并 Iceberg 位置删除文件与数据文件，以生成表的最新视图。若要删除这些位置删除文件，您可以运行 REWRITE DATA 压缩操作。DELETE 操作按扫描的数据量收费。有关语法，请参阅 删除。

以下示例从 iceberg_table 中删除 c3 作为 category 的值的行。

```
DELETE FROM iceberg_table WHERE category='c3'
```

UPDATE

Athena Iceberg UPDATE 将 Iceberg 位置删除文件和新更新的行作为数据文件写入同一事务中。UPDATE 可视为 INSERT INTO 和 DELETE 的组合。UPDATE 操作按扫描的数据量收费。有关语法，请参阅 UPDATE。

以下示例更新表 iceberg_table 中的指定值。

```
UPDATE iceberg_table SET category='c2' WHERE category='c1'
```

MERGE INTO

有条件地更新行、删除行或将行插入到 Iceberg 表中。单个语句可以组合更新、删除和插入操作。有关语法，请参阅 MERGE INTO。

> **Note**
>
> MERGE INTO 是事务性的，仅支持用于 Athena 引擎版本 3 中的 Apache Iceberg 表。

以下示例从表 t 中删除源表 s 中的所有客户。

```
MERGE INTO accounts t USING monthly_accounts_update s
ON t.customer = s.customer
WHEN MATCHED
THEN DELETE
```

以下示例使用源表 s 中的客户信息更新目标表 t。如果表 t 与表 s 中的客户行匹配，则该示例会增加表 t 中的购买量。如果表 t 与表 s 中的客户行不匹配，则该示例会将表 s 的客户行插入到表 t 中。

```
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)
```

以下示例使用源表 s 中的信息有条件地更新目标表 t。该示例删除了源地址为 Centreville 的所有匹配的目标行。对于所有其他匹配的行，该示例会添加源采购并将目标地址设置为源地址。如果目标表中没有匹配项，该示例会插入源表中的行。

```
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)
```

优化 Iceberg 表

随着数据累积到 Iceberg 表中，由于打开文件所需的处理时间增加，查询效率逐渐降低。如果表包含删除文件，则会产生额外的计算成本。在 Iceberg 中，删除文件存储行级删除，引擎必须将删除的行应用于查询结果。

为了帮助优化 Iceberg 表的查询性能，Athena 支持手动压缩作为表维护命令。压缩可在不改变表内容的情况下优化表的结构布局。

OPTIMIZE

`OPTIMIZE` *`table`* `REWRITE DATA` 压缩操作根据数据文件的大小和相关删除文件的数量将数据文件重写为更优化的布局。有关语法和表属性的详细信息，请参见 OPTIMIZE。

示例

以下示例将删除文件合并到数据文件中，然后生成接近目标文件大小的文件，其中 `category` 的值为 `c1`。

```
OPTIMIZE iceberg_table REWRITE DATA USING BIN_PACK
  WHERE category = 'c1'
```

VACUUM

VACUUM 会执行快照过期和孤立文件删除。这些操作会减小元数据大小，并删除不处于当前表状态且早于为表指定的保留期的文件。有关语法的详细信息，请参阅 VACUUM。

示例

以下示例使用表属性将表 iceberg_table 配置为保留最近三天的数据，然后使用 VACUUM 使旧快照过期并从表中删除孤立文件。

```
ALTER TABLE iceberg_table SET TBLPROPERTIES (
  'vacuum_max_snapshot_age_seconds'='259200'
)

VACUUM iceberg_table
```

Athena 中支持的 Iceberg 表数据类型。

Athena 可以查询包含以下数据类型的 Iceberg 表：

```
binary
boolean
date
decimal
double
float
int
list
long
map
string
struct
timestamp without time zone
```

有关 Iceberg 表类型的更多信息，请参阅 Apache 文档中的 Iceberg 的架构页面。

下表显示了 Athena 数据类型与 Iceberg 表数据类型之间的关系。

Iceberg 类型	Athena 类型	注意
boolean	boolean	
-	tinyint	不支持 Athena 中的 Iceberg 表。
-	smallint	不支持 Athena 中的 Iceberg 表。
int	int	在 Athena DML 语句中，此类型是 INTEGER。
long	bigint	
double	double	
float	float	
decimal(P, S)	decimal(P, S)	P 表示精度，S 表示小数位数。
-	char	不支持 Athena 中的 Iceberg 表。
string	string	在 Athena DML 语句中，此类型是 VARCHAR。
binary	binary	
date	date	
time	-	CREATE TABLE 等 Athena Iceberg DDL 语句只支持 Iceberg 时间戳（不带时区），但是所有时间戳类型都可以通过 Athena 查询。
timestamp	timestamp	
timestamptz	timestamptz	
list<E>	array	
map<K,V>	map	

Iceberg 类型	Athena 类型	注意
`struct<...>`	`struct`	
`fixed(L)`	-	Athena 目前不支持 `fixed(L)` 类型。

有关 Athena 中数据类型的更多信息，请参阅 Amazon Athena 中的数据类型。

Iceberg 表上的其他 Athena 操作

数据库级别操作

DROP DATABASE 与 CASCADE 选项一起使用时，将删除所有 Iceberg 表数据。以下 DDL 操作不会对 Iceberg 表格产生影响。

- CREATE DATABASE
- ALTER DATABASE SET DBPROPERTIES
- SHOW DATABASES
- SHOW TABLES
- SHOW VIEWS

分区相关操作

由于 Iceberg 表使用隐藏分区，您无需直接使用物理分区。因此，Athena 中的 Iceberg 表不支持以下与分区相关的 DDL 操作：

- SHOW PARTITIONS
- ALTER TABLE ADD PARTITION
- ALTER TABLE DROP PARTITION
- ALTER TABLE RENAME PARTITION

如果您想在 Athena 中查看 Iceberg 分区发展，请将反馈发送至athena-feedback@amazon.com。

卸载 Iceberg 表

Iceberg 表可以卸载到 Amazon S3 上文件夹中的文件。有关信息，请参阅UNLOAD。

MSCK REPAIR

由于 Iceberg 表会跟踪表布局信息，因此像 Hive 表那样运行 MSCK REPAIR TABLE 是不必要的，也不受支持。

其他 资源

有关将 Athena 与 Apache Iceberg 表一起使用的深入探讨文章，请参阅 AWS 大数据博客。

- 将 Amazon Athena 与 Apache Iceberg 一起使用，加速交易数据湖上的数据科学特征工程
- 使用 Amazon Athena、Amazon EMR 和 AWS Glue 构建 Apache Iceberg 数据湖
- 使用 Amazon Athena 和 Apache Iceberg 在数据湖中执行更新插入
- 使用 Apache Iceberg、AWS Glue 构建交易数据湖，使用 AWS Lake Formation 和 Amazon Athena 构建跨账户数据共享
- 在数据湖中使用 Apache Iceberg 来支持增量数据处理
- 构建与 GDPR 一致的实时 Apache Iceberg 数据湖
- 使用 Apache Iceberg 和 AWS Glue 自动将关系源复制到交易数据湖中
- 与使用 Amazon Athena 的 Apache Iceberg 交互，以及与使用 AWS Lake Formation 的跨账户精细权限交互
- 使用 Apache Iceberg、Amazon EMR Serverless 和 Amazon Athena 构建无服务器交易数据湖

Amazon Athena 安全性

AWS 十分重视云安全性。作为 AWS 客户，您将从专为满足大多数安全敏感型企业的要求而打造的数据中心和网络架构中受益。

安全性是 AWS 和您的共同责任。责任共担模式将其描述为云的 安全性和云中 的安全性：

- 云的安全性 – AWS 负责保护在 AWS 云中运行 AWS 服务 的基础设施。AWS 还向您提供可安全使用的服务。作为 AWS 合规性计划的一部分，我们的安全措施的有效性定期由第三方审计员进行测试和验证。要了解适用于 Athena 的合规性计划，请参阅 合规性计划范围内的 AWS 服务。
- 云中的安全性 – 您的责任由您使用的 AWS 服务 决定。您还需要对其他因素负责，包括您的数据的敏感性、您组织的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 Amazon Athena 时应用责任共担模型。以下主题说明如何配置 Athena 以实现您的安全性和合规性目标。您还将了解如何使用其他 AWS 服务 来帮助您监控和保护您的 Athena 资源。

主题

Athena 的数据保护

AWS责任共担模式适用于 Amazon Athena 中的数据保护。如该模式中所述，AWS 负责保护运行所有 AWS Cloud 的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS 安全性博客 上的 AWS 责任共担模式和 GDPR 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management（IAM）设置单个用户。这样，每个用户只获得履行其工作职责所需的权限。我们还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证 (MFA)。
- 使用 SSL/TLS 与 AWS 资源进行通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用 AWS CloudTrail 设置 API 和用户活动日记账记录。
- 使用 AWS 加密解决方案以及 AWS 服务 中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果在通过命令行界面或 API 访问 AWS 时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-2 版》。

我们强烈建议您切勿将机密信息或敏感信息（例如您客户的电子邮件地址）放入标签或自由格式字段[例如 Name（名称）字段]。这包括通过控制台、API、AWS CLI 或 AWS SDK 使用 Athena 或其他 AWS 服务时。在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供网址，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

作为额外的安全步骤，您可以使用 aws:CalledVia 全局条件上下文键将请求限制为仅从 Athena 发出的请求。有关更多信息，请参阅 将 Athena 与 CalledVia 上下文键结合使用。

保护多种类型的数据

当您使用 Athena 创建数据库和表时，涉及多种类型的数据。这些数据类型包括 Amazon S3 中存储的源数据、在运行查询或 AWS Glue Crawler 以发现数据时创建的数据库和表的元数据、查询结果数据，以及查询历史记录。本部分介绍每种类型的数据并提供有关保护数据的指导。

- 源数据 – 您在 Amazon S3 中存储数据库和表的数据，并且 Athena 不修改这些数据。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 Amazon S3 中的数据保护。您可以控制对您的源数据的访问并在 Amazon S3 中加密这些数据。您可以使用 Athena，在 Amazon S3 中根据加密的数据集创建表。
- 数据库和表元数据（架构）– Athena 使用基于读取的架构技术，这意味着当 Athena 运行查询时，表定义将应用于 Amazon S3 中的数据。您定义的任何架构都会自动保存，除非明确将其删除。在 Athena 中，您可以使用 DDL 语句修改 Data Catalog 元数据。您还可以删除表定义和架构，这不会影响存储在 Amazon S3 上的基础数据。在 Athena 中使用的数据库和表元数据存储在 AWS Glue Data Catalog 中。

  您可以使用 AWS Identity and Access Management (IAM) 定义针对在 AWS Glue Data Catalog 中注册的数据库和表的精细访问策略。您还可以对 AWS Glue Data Catalog 中的元数据进行加密。如果您加密元数据，请使用对加密元数据的权限进行访问。
- 查询结果和查询历史记录，包括保存的查询 - 查询结果存储在 Amazon S3 位置中，您可以选择全局指定该位置，或者为每个工作组指定该位置。如果未指定，Athena 在每个案例中使用默认位置。您可以控制对 Amazon S3 存储桶的访问，您在这些存储桶中存储查询结果和保存的查询。此外，您可以选择加密在 Amazon S3 中存储的查询结果。您的用户必须拥有适当的权限，才能访问 Amazon S3 位置并解密文件。有关更多信息，请参阅本文档中的加密在 Amazon S3 中存储的 Athena 查询结果。

  Athena 保留查询历史记录 45 天。您可以在控制台中使用 Athena API 查看查询历史记录，或者使用 AWS CLI 查看。要将查询存储超过 45 天时间，请保存查询。要保护对已保存查询的访问，在 Athena 中使用工作组，限制仅有权查看已保存查询的用户才能访问。

主题

静态加密

您可以在 Amazon Athena 中针对同一区域或有限数量的区域中 Amazon S3 中的加密数据运行查询。您还可以加密 Amazon S3 中的查询结果以及 AWS Glue Data Catalog 中的数据。

您可以对 Athena 中的以下资产进行加密：

- Amazon S3 中所有查询的结果，Athena 将这些结果存储在称为 Amazon S3 结果位置的位置中。您可以加密在 Amazon S3 中存储的查询结果，无论底层数据集是否在 Amazon S3 中加密都是如此。有关信息，请参阅加密在 Amazon S3 中存储的 Athena 查询结果。
- AWS Glue 数据目录中的数据。有关信息，请参阅针对 AWS Glue 数据目录中加密元数据的权限。

> **Note**
>
> 使用 Athena 读取加密的表时，Athena 使用为表数据指定的加密选项，而不是查询结果的加密选项。如果为查询结果和表数据配置了单独的加密方法或密钥，Athena 将读取表数据，而不使用加密选项和用于加密或解密查询结果的密钥。
>
> 但是，如果使用 Athena 将数据插入到具有加密数据的表中，则 Athena 将使用为查询结果指定的加密配置来加密插入的数据。例如，如果您为查询结果指定 CSE_KMS 加密，则 Athena 将使用与用于查询结果加密的相同 AWS KMS 密钥 ID，通过 CSE_KMS 对插入的表数据进行加密。

主题

支持的 Amazon S3 加密选项

Athena 支持 Amazon S3 中数据集和查询结果的以下加密选项。

加密类型	描述	跨区域支持
SSE-S3	使用 Amazon S3 托管式密钥进行服务器端加密 (SSE)。	是
SSE-KMS	使用 AWS Key Management Service 客户管理的密钥进行服务器端加密 (SSE)。 **Note** 使用此加密类型，Athena 不需要您在创建表时指示数据是加密的。	是
CSE-KMS	使用 AWS KMS 客户管理的密钥进行客户端加密 (CSE)。在 Athena 中，此选项要求您使用带有 `TBLPROPERTIES` 子句的 `CREATE TABLE` 语句，指定 `'has_encrypted_data'='true'` 。有关更多信息，请参阅 根据 Amazon S3 中的加密数据集创建表。	否

有关使用 Amazon S3 进行 AWS KMS 加密的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的 什么是 AWS Key Management Service 和 Amazon Simple Storage Service (Amazon S3) 如何使用 AWS KMS。有关使用 Athena 操作 SSE-KMS 或 CSE-KMS 的更多信息，请参阅 AWS 大数据博客的 启动：Amazon Athena 增加了对查询加密数据的支持。

不支持的选项

不支持以下加密选项：

- 客户提供密钥 SSE (SSE-C)。
- 使用客户端托管式密钥进行客户端加密。
- 非对称密钥。

若要比较 Amazon S3 加密选项，请参阅《Amazon Simple Storage Service 用户指南》中的 使用加密保护数据。

客户端加密的工具

对于客户端加密，请注意有两种工具可用：

- Amazon S3 加密客户端 – 此操作工具仅加密 Amazon S3 的数据，并且受 Athena 支持。
- AWS Encryption SDK – 软件开发工具包可在 AWS 的任何地方加密数据但并不直接受 Athena 支持。

这两种工具不兼容，使用一种工具加密的数据不能被另一种工具解密。Athena 仅能直接支持 Amazon S3 加密客户端。如果您使用软件开发工具包对数据进行加密，则可以从 Athena 运行查询，但数据将作为加密文本返回。

如果要使用 Athena 查询已使用 AWS 加密软件开发工具包加密的数据，您必须下载并解密您的数据，然后使用 Amazon S3 加密客户端再次对其加密。

Amazon S3 中加密数据的权限

根据在 Amazon S3 中使用的加密类型，可能需要向 Athena 中使用的策略添加权限（也称为“允许”操作）：

- SSE-S3 - 如果使用 SSE-S3 加密，则 Athena 用户无需在其策略中添加权限。对相应的 Amazon S3 位置和 Athena 操作具有适当的 Amazon S3 权限就足够了。要详细了解允许适当 Athena 和 Amazon S3 权限的策略，请参阅Amazon Athena 的 AWS 托管策略和 对 Amazon S3 的访问权限。
- AWS KMS - 如果使用 AWS KMS 加密，则除了 Athena 和 Amazon S3 权限之外，还必须允许 Athena 用户执行特定 AWS KMS 操作。通过编辑对 Amazon S3 中数据加密使用的 AWS KMS 客户管理的密钥 (CMK) 密钥策略，您可以允许这些操作。要将密钥用户添加到相应的 AWS KMS 密钥策略，则可以使用 AWS KMS 控制台，网址：https://console.aws.amazon.com/kms。有关如何将用户添加到 AWS KMS 密钥策略，请参阅《AWS Key Management Service 开发人员指南》中的允许密钥用户使用 CMK。

  > **Note**
  >
  > 高级密钥策略管理员可以调整密钥策略。要处理加密数据集，最少应允许 Athena 用户执行 `kms:Decrypt` 操作。要使用加密的查询结果，则最小允许操作是 `kms:GenerateDataKey` 和 `kms:Decrypt`。

  当使用 Athena 在 Amazon S3 中查询数据集时，如果有大量使用 AWS KMS 加密的对象，则 AWS KMS 可能会节流查询结果。当有大量的小对象时，这种可能性更大。Athena 已停止重试请求，但可

能仍会发生节流错误。如果您在处理大量加密对象时遇到此问题，则可以选择启用 Amazon S3 存储桶密钥来减少对 KMS 的调用次数。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 使用 Amazon S3 存储桶密钥降低 SSE-KMS 的成本。另一种选择是增加 AWS KMS 的服务配额。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的配额。

有关将 Amazon S3 与 Athena 结合使用时的权限疑难解答信息，请参阅 在 Athena 中进行故障排除 主题的 权限 部分。

针对 AWS Glue 数据目录中加密元数据的权限

如果加密 AWS Glue Data Catalog 中的元数据，则必须向用于访问 Athena 的策略添加 `"kms:GenerateDataKey"`、`"kms:Decrypt"` 和 `"kms:Encrypt"` 操作。有关信息，请参阅从 Athena 访问 AWS Glue Data Catalog 中的加密元数据。

加密在 Amazon S3 中存储的 Athena 查询结果

使用 Athena 控制台或使用 JDBC 或 ODBC 时设置查询结果加密。使用工作组可以强制对查询结果实施加密。

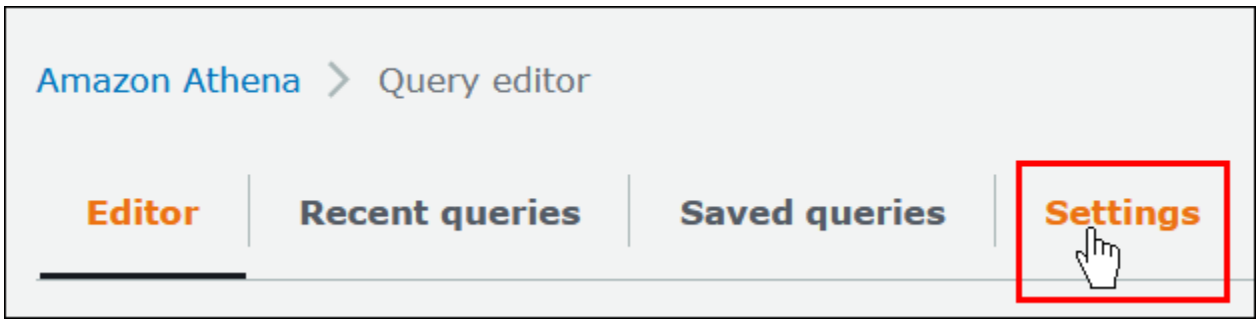
在控制台中，您可以通过两种方式配置对查询结果加密的设置：

- 客户端设置 - 当您在控制台中使用 Settings（设置）或者 API 操作来指示您希望加密查询结果时，这称为使用客户端设置。客户端设置包括查询结果位置和加密。如果您指定这些内容，则除非工作组设置进行覆盖，否则将使用它们。
- Workgroup settings（工作组设置）– 当您创建或编辑工作组并选择 Override client-side settings（覆盖客户端设置）字段时，此工作组中运行的所有查询均使用工作组设置和查询结果位置设置。有关更多信息，请参阅 工作组设置覆盖客户端设置。

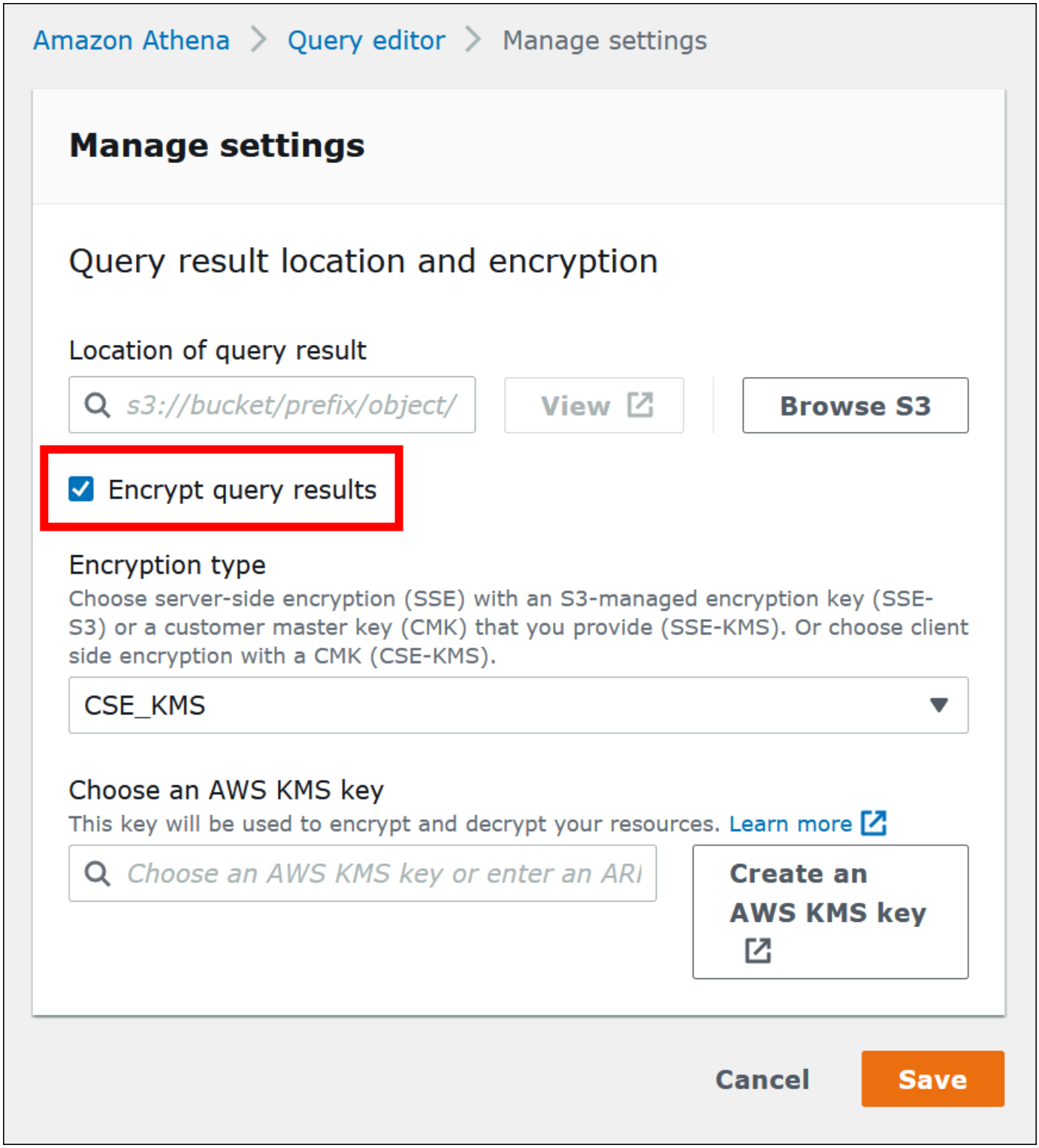
使用控制台加密在 Amazon S3 中存储的查询结果

> **Important**
>
> 如果工作组中的 Override client-side settings（覆盖客户端设置）字段已被选中，则工作组中的所有查询均使用工作组设置。不使用在 Athena 控制台中的 Settings（设置）选项卡上通过 API 操作及通过 JDBC 和 ODBC 驱动程序指定的加密配置和查询结果位置。有关更多信息，请参阅 工作组设置覆盖客户端设置。

1. 在 Athena 控制台中，选择 Settings（设置）。



2. 选择管理。
3. 在 Location of query result（查询结果位置）中，输入或选择 Amazon S3 路径。这是存储查询结果的 Amazon S3 位置。
4. 选择 Encrypt query results。



5. 对于 Encryption type，选择 CSE-KMS、SSE-KMS 或 SSE-S3。在这三者中，CSE-KMS 提供的加密级别最高，SSE-S3 提供的加密级别最低。
6. 如果您选择了 SSE-KMS 或 CSE-KMS，则指定 AWS KMS 密钥。

- 对于选择 AWS KMS 键，如果您的账户有权访问某个现有的 AWS KMS 客户自主管理型密钥（CMK），则选择其别名，或输入一个 AWS KMS 键 ARN。
- 如果您的账户无权访问任何现有的客户自主管理型密钥（CMK），则选择创建 AWS KMS 键，然后打开 AWS KMS 控制台。有关更多信息，请参阅 AWS Key Management Service 开发人员指南中的创建密钥。

> **Note**
>
> Athena 仅支持读取和写入数据的对称密钥。

7. 返回 Athena 控制台，选择您通过别名或 ARN 创建的密钥。
8. 选择保存。

使用 JDBC 或 ODBC 时对 Athena 查询结果进行加密

如果您使用 JDBC 或 ODBC 驱动程序进行连接，请配置驱动程序选项来指定要使用的加密类型以及 Amazon S3 暂存目录位置。要配置 JDBC 或 ODBC 驱动程序以使用 Athena 支持的任何加密协议加密查询结果，请参阅 通过 ODBC 和 JDBC 驱动程序连接到 Amazon Athena。

根据 Amazon S3 中的加密数据集创建表

创建表时，向 Athena 指示已在 Amazon S3 中加密数据集。使用 SSE-KMS 时不需要。对于 SSE-S3 和 AWS KMS 加密，Athena 将确定如何解密数据集和创建表，因此您无需提供密钥信息。

运行查询的用户（包括创建表的用户）必须具有本主题前面所述的权限。

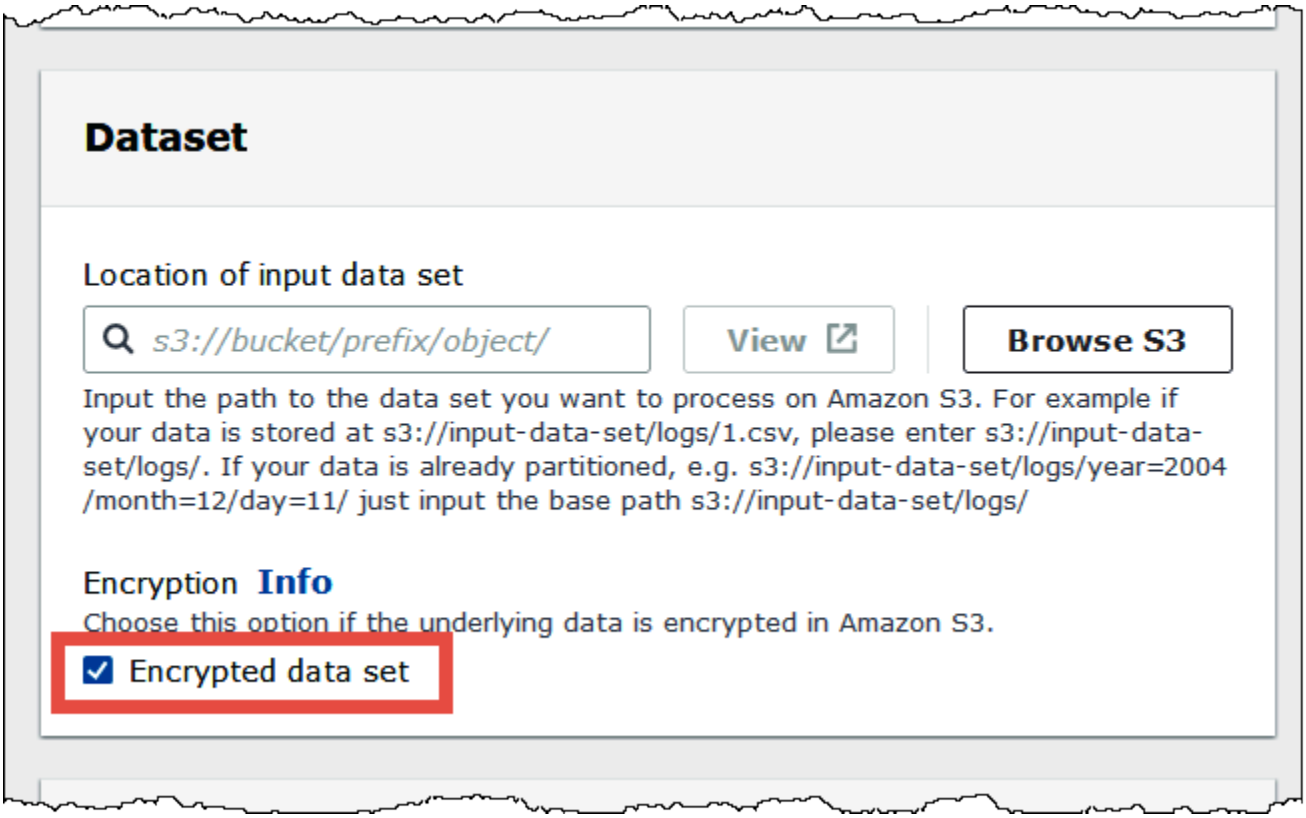
> **Important**
>
> 如果使用 Amazon EMR 以及 EMRFS 上载加密的 Parquet 文件，则必须通过将 `fs.s3n.multipart.uploads.enabled` 设置为 `false` 来禁用分段上载。如果您未执行此操作，则 Athena 无法确定 Parquet 文件长度，将出现 HIVE_CANNOT_OPEN_SPLIT 错误。有关更多信息，请参阅《Amazon EMR 管理指南》中的为 Amazon S3 配置分段上传。

要指示 Amazon S3 中的数据集已加密，请执行以下步骤之一。如果使用 SSE-KMS，无需执行此步骤。

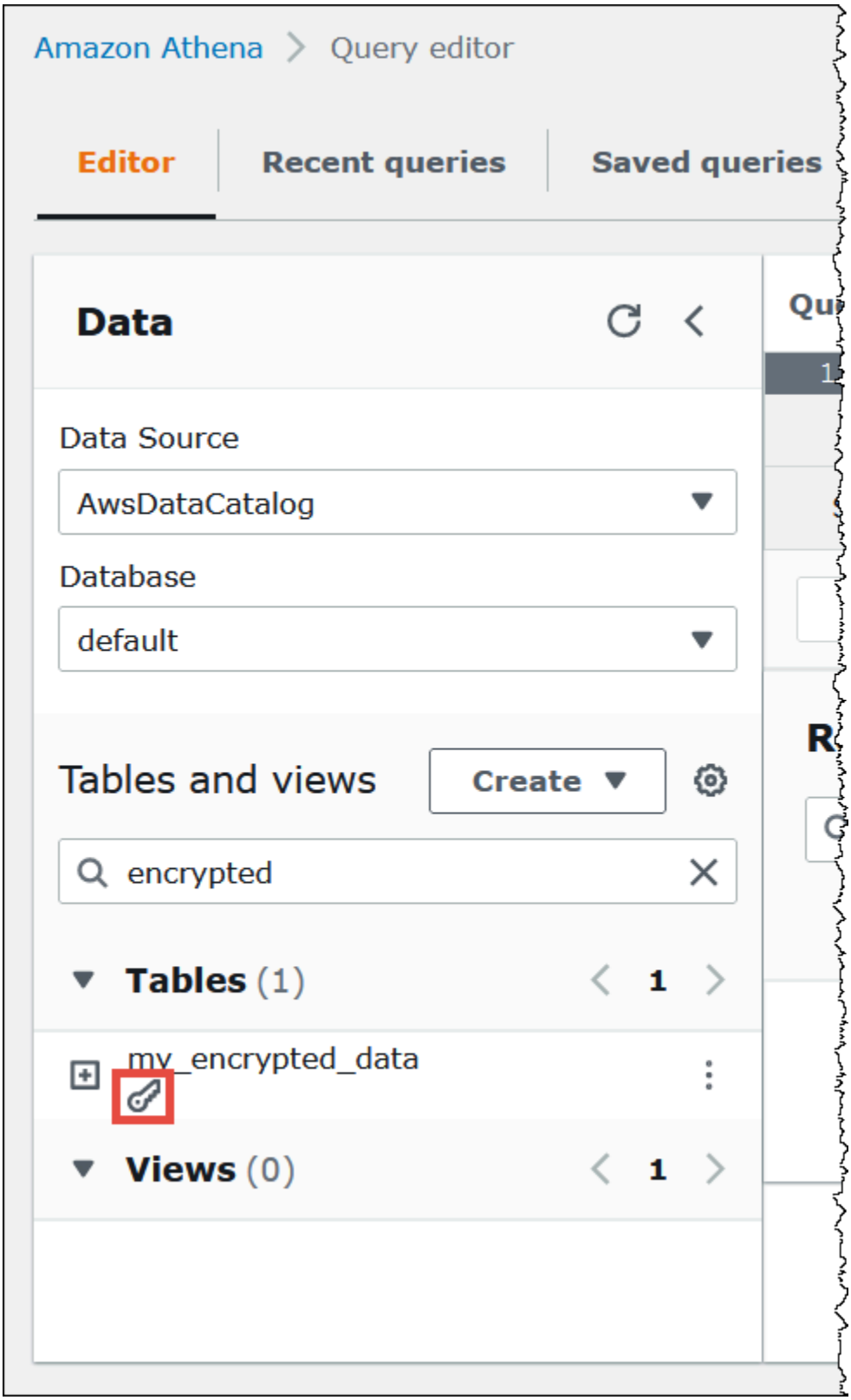
- 在 CREATE TABLE 语句，请使用 TBLPROPERTIES 子句指定 `'has_encrypted_data'='true'`，如以下示例所示。

```
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')
```

- 使用 JDBC 驱动程序 并设置 TBLPROPERTIES 值，如上一个示例中当您使用 `statement.executeQuery()` 运行 CREATE TABLE 语句时所示。
- 使用 Athena 控制台使用表单创建表并指定表位置时，请选择 Encrypted data set（加密的数据集）选项。



在 Athena 控制台的表列表中，加密的表显示钥匙形图标。



传输中加密

除了对 Amazon S3 中的数据静态加密，Amazon Athena 还对 Athena 与 Amazon S3 之间、以及 Athena 与访问它的客户应用程序之间的传输中数据使用传输层安全性 (TLS) 加密。

您应该在 Amazon S3 存储桶 IAM 策略上使用 `aws:SecureTransport condition`，以只允许通过 HTTPS (TLS) 的加密连接。

流式传输到 JDBC 或 ODBC 客户端的查询结果使用 TLS 进行加密。有关最新版本的 JDBC 和 ODBC 驱动程序及其文档的信息，请参阅通过 JDBC 连接到 Amazon Athena和通过 ODBC 连接到 Amazon Athena。

对于 Athena 联合数据来源连接器，是否支持使用 TLS 进行传输中加密取决于各个连接器。有关信息，请参阅各个数据来源连接器的文档。

密钥管理

Amazon Athena 支持使用 AWS Key Management Service (AWS KMS) 来加密 Amazon S3 和 Athena 查询结果中的数据集。AWS KMS 使用客户主密钥 (CMK) 加密您的 Amazon S3 对象，并依赖于信封加密。

在 AWS KMS 中，您可以执行以下操作：

- 创建密钥
- 为新 CMK 导入自己的密钥材料

> **Note**
>
> Athena 仅支持读取和写入数据的对称密钥。

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的什么是 AWS Key Management Service 和 Amazon Simple Storage Service 如何使用 AWS KMS。要查看账户中 AWS 为您创建和管理的密钥，请在导航窗格中选择 AWS 托管式密钥。

如果您要上载或访问使用 SSE-KMS 加密的对象，请使用 AWS 签名版本 4 来提高安全性。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 在请求身份验证中指定签名版本。

如果您的 Athena 工作负载加密大量数据，您可以使用 Amazon S3 存储桶密钥来降低成本。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 使用 Amazon S3 存储桶密钥降低 SSE-KMS 的成本。

互联网络流量隐私保护

对 Athena 与本地应用程序之间的流量以及 Athena 与 Amazon S3 之间的流量进行保护。Athena 与其他服务（如 AWS Glue 和 AWS Key Management Service）之间的流量预设情况下使用 HTTPS。

- 对于 Athena 与本地客户端和应用程序之间的流量，流式传输到 JDBC 或 ODBC 客户端的查询结果使用传输层安全性 (TLS) 进行加密。

  您可以在私有网络与 AWS 之间使用以下连接选项之一：

  - 一个 Site-to-Site VPN AWS VPN 连接。有关更多信息，请参阅《AWS Site-to-Site VPN 用户指南》中的什么是 Site-to-Site VPN AWS VPN。
  - AWS Direct Connect 连接。有关更多信息，请参阅《AWS Direct Connect 用户指南》中的什么是 AWS Direct Connect。
- 对于 Athena 与 Amazon S3 存储桶之间的流量，传输层安全性 (TLS) 会加密 Athena 和 Amazon S3 之间以及在 Athena 和访问该对象的客户应用程序之间的传输对象，您应该只允许Amazon S3 存储桶 IAM 策略上 `aws:SecureTransport condition` 通过 HTTPS (TLS) 的加密连接。尽管 Athena 目前使用公有端点访问 Amazon S3 存储桶中的数据，但这并不意味着数据会遍历公有互联网。Athena 和 Amazon S3 之间的所有流量都通过 AWS 网络路由，并使用 TLS 进行加密。
- 合规性计划 – Amazon Athena 遵守多个 AWS 合规性计划，其中包括 SOC、PCI、FedRAMP 及其他计划。有关更多信息，请参阅合规性计划范围内的 AWS 服务。

Athena 中的 Identity and Access Management

Amazon Athena 使用 AWS Identity and Access Management (IAM) 策略来限制对 Athena 操作的访问。有关 Athena 的完整权限列表，请参阅《服务授权参考》中的 Amazon Athena 的操作、资源和条件键。

每当您使用 IAM policy 时，请确保遵循 IAM 最佳实践。有关更多信息，请参阅《IAM 用户指南》中的 IAM 安全最佳实践。

运行 Athena 查询所需的权限包含以下内容：

- 存储查询底层数据的 Amazon S3 位置。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 Amazon S3 中的身份和访问管理。

- 在 AWS Glue Data Catalog 中存储的元数据和资源，例如数据库和表，包括加密元数据的其他操作。有关更多信息，请参阅《AWS Glue 开发人员指南》中的 为 AWS Glue 设置 IAM 权限 和 在 AWS Glue 中设置加密。
- Athena API 操作。有关 Athena 中 API 操作的完整列表，请参阅 Amazon Athena API 参考中的操作。

以下主题提供了有关 Athena 的特定区域权限的更多信息。

主题

Amazon Athena 的 AWS 托管策略

AWS 托管策略是由 AWS 创建和管理的独立策略。AWS 托管策略旨在为许多常见用例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住，AWS 托管式策略可能不会为您的特定使用场景授予最低权限，因为它们可供所有 AWS 客户使用。我们建议通过定义特定于您的使用场景的客户管理型策略来进一步减少权限。

您无法更改 AWS 托管策略中定义的权限。如果 AWS 更新在 AWS 托管式策略中定义的权限，则更新会影响该策略所附加到的所有主体身份（用户、组和角色）。当新的 AWS 服务启动或新的 API 操作可用于现有服务时，AWS 最有可能更新 AWS 托管式策略。

有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略。

在 Athena 中使用托管策略时的注意事项

托管策略易于使用，并且随着服务的发展而自动使用必需的操作进行更新。将托管策略与 Athena 结合使用时，请记住以下几点：

- 要使用 AWS Identity and Access Management (IAM) 为您自己或其他用户允许或拒绝 Amazon Athena 服务操作，请将基于身份的策略附加到委托人，例如用户或组。
- 每个基于身份的策略均包含用于定义允许或拒绝的操作的语句。有关将策略附加到用户的更多信息和分步说明，请参阅 IAM 用户指南中的 附加托管式策略。有关操作列表，请参阅 Amazon Athena API 参考。
- 客户托管策略和基于身份的内联策略允许您在策略内指定更详细的 Athena 操作来精调访问权限。我们建议您使用 `AmazonAthenaFullAccess` 策略作为起始点，然后允许或拒绝 Amazon Athena API 参考中所列的特定操作。有关内联策略的更多信息，请参阅《IAM 用户指南》中的 托管式策略与内联策略。
- 如果您还具有使用 JDBC 连接的委托人，则必须为您的应用程序提供 JDBC 驱动程序凭证。有关更多信息，请参阅 通过 JDBC 和 ODBC 连接访问。
- 如果您已加密 AWS Glue Data Catalog，则必须在基于身份的 IAM 策略中为 Athena 指定其他操作。有关更多信息，请参阅 从 Athena 访问 AWS Glue Data Catalog 中的加密元数据。
- 如果您创建和使用工作组，请确保您的策略包括对工作组操作的相关访问权限。有关详细信息，请参阅the section called “ 用于访问工作组的 IAM policy”和the section called “工作组策略示例”。

AWS 托管策略：AmazonAthenaFullAccess

`AmazonAthenaFullAccess` 托管策略授予对 Athena 的完全访问权限。

要提供访问权限，请为您的用户、组或角色添加权限：

- AWS IAM Identity Center 中的用户和群组：

  创建权限集合。按照《AWS IAM Identity Center 用户指南》中 创建权限集 的说明进行操作。
- 通过身份提供商在 IAM 中托管的用户：

创建适用于身份联合验证的角色。按照《IAM 用户指南》中为第三方身份提供商创建角色（联合身份验证）的说明进行操作。

- IAM 用户：
  - 创建您的用户可以担任的角色。按照《IAM 用户指南》中为 IAM 用户创建角色的说明进行操作。
  - （不推荐使用）将策略直接附加到用户或将用户添加到用户组。按照《IAM 用户指南》中向用户添加权限（控制台）中的说明进行操作。

权限组

`AmazonAthenaFullAccess` 策略被分组为以下权限集。

- **`athena`** – 允许委托人访问 Athena 资源。
- **`glue`** – 允许委托人访问 AWS Glue 数据库、表和分区。这是必需的，以便委托人可以将 AWS Glue Data Catalog 和 Athena 搭配使用。
- **`s3`** – 允许委托人从 Amazon S3 编写和读取查询结果，读取驻留在 Amazon S3 中的公开可用的 Athena 数据示例，并列出存储桶。这是必需的，以便委托人可以将 Athena 与 Amazon S3 搭配使用。
- **`sns`** – 允许委托人列出 Amazon SNS 主题并获取主题属性。这使委托人能够将 Amazon SNS 主题与 Athena 结合使用，以进行监控和提示。
- **`cloudwatch`** – 允许委托人创建、读取和删除 CloudWatch 告警。有关更多信息，请参阅 使用 CloudWatch 指标和事件控制成本和监控查询。
- **`lakeformation`** – 允许委托人请求临时证书以访问已注册到 Lake Formation 的数据湖位置中的数据。有关更多信息，请参阅《AWS Lake Formation 开发人员指南https://docs.aws.amazon.com/lake-formation/latest/dg/access-control-underlying-data.html》中的 底层数据访问控制。
- **`datazone`** – 允许主体列出 Amazon DataZone 项目、域和环境。有关如何在 Athena 中使用 DataZone 的信息，请参阅 在 Athena 中使用 Amazon DataZone。
- **`pricing`** – 提供对 AWS Billing and Cost Management 的访问权限。有关更多信息，请参阅《AWS Billing and Cost Management API 参考》中的 GetProducts。

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

```
            "Effect": "Allow",
            "Action": [
                "athena:*"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseGluePermissions",
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:DeleteDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:CreateTable",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:StartColumnStatisticsTaskRun",
                "glue:GetColumnStatisticsTaskRun",
                "glue:GetColumnStatisticsTaskRuns"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseQueryResultsPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
```

```
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:PutObject",
                "s3:PutBucketPublicAccessBlock"
            ],
            "Resource": [
                "arn:aws:s3:::aws-athena-query-results-*"
            ]
        },
        {
            "Sid": "BaseAthenaExamplesPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::athena-examples*"
            ]
        },
        {
            "Sid": "BaseS3BucketPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:ListAllMyBuckets"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseSNSPermissions",
            "Effect": "Allow",
            "Action": [
                "sns:ListTopics",
                "sns:GetTopicAttributes"
            ],
            "Resource": [
```

```
                "*"
            ]
        },
        {
            "Sid": "BaseCloudWatchPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:GetMetricData"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseLakeFormationPermissions",
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BaseDataZonePermissions",
            "Effect": "Allow",
            "Action": [
                "datazone:ListDomains",
                "datazone:ListProjects",
                "datazone:ListAccountEnvironments"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "BasePricingPermissions",
            "Effect": "Allow",
            "Action": [
                "pricing:GetProducts"
            ],
```

```
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AWS 托管策略：AWSQuicksightAthenaAccess

AWSQuicksightAthenaAccess 授予对 Amazon QuickSight 与 Athena 集成所需操作的访问权限。您可以将 AWSQuicksightAthenaAccess 策略附加到 IAM 身份。仅将此策略附加到将 Amazon QuickSight 与 Athena 结合使用的委托人。此策略包括 Athena 的一些操作，这些操作已弃用且未包含在当前公有 API 中或仅与 JDBC 和 ODBC 驱动程序一起使用。

权限组

AWSQuicksightAthenaAccess 策略被分组为以下权限集。

- **athena** – 允许委托人对 Athena 资源运行查询。
- **glue** – 允许委托人访问 AWS Glue 数据库、表和分区。这是必需的，以便委托人可以将 AWS Glue Data Catalog 和 Athena 搭配使用。
- **s3** – 允许委托人从 Amazon S3 中写入和读取查询结果。
- **lakeformation** – 允许委托人请求临时证书以访问已注册到 Lake Formation 的数据湖位置中的数据。有关更多信息，请参阅《AWS Lake Formation 开发人员指南https://docs.aws.amazon.com/lake-formation/latest/dg/access-control-underlying-data.html》中的 底层数据访问控制。

```
{
    "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": [
                "*"
            ]
        }
    ]
}
```

AWS 托管策略的 Athena 更新

查看有关 Athena AWS 托管策略更新的详细信息（从该服务开始跟踪这些更改开始）。

更改	描述	日期
AmazonAthenaFullAccess – 对现有政策的更新	添加了 `datazone:ListDomains` 、`datazone:ListProjects` 和 `datazone:ListAccountEnvironments` 权限，以使 Athena 用户能够使用 Amazon DataZone 域、项目和环境。有关更多信息，请参阅 在 Athena 中使用 Amazon DataZone。	2024 年 1 月 3 日

更改	描述	日期
AmazonAthenaFullAccess – 对现有政策的更新	添加了 `glue:StartColumnStatisticsTaskRun` 、`glue:GetColumnStatisticsTaskRun` 和 `glue:GetColumnStatisticsTaskRuns` 权限，以使 Athena 有权调用 AWS Glue，以检索基于成本的优化器功能的统计数据。有关更多信息，请参阅 使用成本型优化器。	2024 年 1 月 3 日
AmazonAthenaFullAccess – 对现有政策的更新	Athena 增加了 `pricing:GetProducts` 以提供对 AWS Billing and Cost Management 的访问权限。有关更多信息，请参阅《AWS Billing and Cost Management API 参考》中的 GetProducts。	2023 年 1 月 25 日
AmazonAthenaFullAccess – 对现有政策的更新	Athena 添加了 `cloudwatch:GetMetricData` ，用以检索 CloudWatch 指标值。有关更多信息，请参阅 Amazon CloudWatch API 参考中的 GetMetricData。	2022 年 11 月 14 日
AmazonAthenaFullAccess 和 AWSQuicksightAthenaAccess – 对现有策略的更新	将 Athena 添加到 `s3:PutBucketPublicAccessBlock` 以允许阻止对 Athena 创建的存储桶的公共访问。	2021 年 7 月 7 日
Athena 开始跟踪变更	Athena 为其 AWS 托管策略开启了跟踪更改。	2021 年 7 月 7 日

通过 JDBC 和 ODBC 连接访问

若要获得 AWS 服务 和资源（如 Athena 和 Amazon S3 存储桶）的访问权限，请向您的应用程序提供 JDBC 或 ODBC 驱动程序凭证。如果您使用的是 JDBC 或 ODBC 驱动程序，请确保 IAM 权限策略包括 AWS 托管策略：AWSQuicksightAthenaAccess 中列出的所有操作。

每当您使用 IAM policy 时，请确保遵循 IAM 最佳实践。有关更多信息，请参阅《IAM 用户指南》中的 IAM 安全最佳实践。

身份验证方法

Athena JDBC 和 ODBC 驱动程序支持基于 SAML 2.0 的身份验证，包括以下身份提供程序：

- Active Directory 联合身份验证服务 (AD FS)
- Azure Active Directory (AD)
- Okta
- PingFederate

有关详细信息，请参阅相应驱动程序的安装和配置指南，可从 JDBC 和 ODBC 驱动程序页面下载 PDF 格式文件。有关更多信息，请参阅以下内容：

- 启用对 Athena API 的联合身份访问
- 将 Lake Formation 和 Athena JDBC 和 ODBC 驱动程序用于对 Athena 进行联合访问
- 使用 ODBC、SAML 2.0 和 Okta 身份提供商配置单点登录

有关最新版本的 JDBC 和 ODBC 驱动程序及其文档的信息，请参阅通过 JDBC 连接到 Amazon Athena和通过 ODBC 连接到 Amazon Athena。

对 Amazon S3 的访问权限

您可以使用基于身份的策略、存储桶资源策略、访问点策略或上述内容的任意组合授予对 Amazon S3 位置的访问权限。当参与者与 Athena 交互时，其权限会通过 Athena 传递，以确定 Athena 可以访问的内容。这意味着用户必须具有 Amazon S3 存储桶的访问权限，才能使用 Athena 进行查询。

每当您使用 IAM policy 时，请确保遵循 IAM 最佳实践。有关更多信息，请参阅《IAM 用户指南》中的 IAM 安全最佳实践。

当您在策略中配置 `aws:SourceIp` 时，Athena 将使用您指定的 IP 地址访问 Amazon S3 存储桶。您无法根据 `aws:SourceVpc` 或 `aws:SourceVpce` 条件键限制或允许对 Amazon S3 资源的访问。

> **Note**
>
> 使用 IAM Identity Center 身份验证的 Athena 工作组要求将 S3 Access Grants 配置为使用可信身份传播身份。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 S3 Access Grants and directory identities。

Amazon S3 访问点和访问点别名

如果您在 Amazon S3 存储桶中拥有共享数据集，则维护单个存储桶策略以管理数百个使用案例的访问权限可能会非常困难。

Amazon S3 存储桶访问点有助于解决此问题。一个存储桶可以有多个访问点，每个访问点都有一个策略，能够以不同方式控制对存储桶的访问。

对于您创建的每个访问点，Amazon S3 都会生成一个代表访问点的别名。由于别名采用 Amazon S3 存储桶名称格式，因此您可以在 Athena 中 `CREATE TABLE` 语句的 `LOCATION` 子句中使用别名。然后，Athena 对存储桶的访问由别名所代表的访问点策略控制。

有关更多信息，请参阅《Amazon S3 用户指南》中的 Amazon S3 中的表位置 和使用访问点。

使用 CalledVia 下文键

为了增加安全性，您可以使用 aws:CalledVia 全局条件上下文键。`aws:CalledVia` 键包含链中代表主体发出请求的每个服务的有序列表。通过为 `aws:CalledVia` 上下文键指定 Athena 服务主体名称 `athena.amazonaws.com`，您可以将请求限制为仅从 Athena 发出的请求。有关更多信息，请参阅 将 Athena 与 CalledVia 上下文键结合使用。

其他 资源

有关如何授予 Amazon S3 访问权限的详细信息和示例，请参阅以下资源：

- 《Amazon S3 用户指南》中的 示例演练：管理访问。
- AWS 知识中心中的如何提供对 Amazon S3 存储桶中对象的跨账户访问权限？。
- Athena 中对 Amazon S3 存储桶的跨账户访问.

Athena 中对 Amazon S3 存储桶的跨账户访问

一个常见的 Amazon Athena 方案是向账户中与存储桶拥有者不同的用户授予访问权限，以便他们可以执行查询。在这种情况下，使用存储桶策略来授予访问权限。

> **Note**
>
> 有关从 Athena 跨账户访问 AWS Glue 数据目录的信息，请参阅 授予 AWS Glue 数据目录跨账户访问权限。

以下示例存储桶策略 (由存储桶所有者创建并应用到存储桶 s3://DOC-EXAMPLE-BUCKET) 向账户 123456789123 (它是不同账户) 中的所有用户授予访问权限。

```
{
   "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/*"
          ]
       }
    ]
 }
```

要向账户中的特定用户授予访问权限，请将 Principal 密钥替换为用户指定的密钥，而不是 root。例如，对于用户配置文件 Dave，请使用 arn:aws:iam::123456789123:user/Dave。

跨账户访问使用自定义 AWS KMS 密钥加密的存储桶

如果您的 Amazon S3 存储桶使用自定义 AWS Key Management Service (AWS KMS) 密钥加密，则可能需要向其他 Amazon Web Services 账户中的用户授予访问该存储桶的权限。

向账户 B 中的用户授予对账户 A 中 AWS KMS 加密存储桶的访问权限需要以下权限：

- 账户 A 中的存储桶策略必须授予对账户 B 所担任角色的访问权限。
- 账户 A 中的 AWS KMS 密钥策略必须授予对账户 B 中用户所担任角色的访问权限。
- 账户 B 担任的 AWS Identity and Access Management（IAM）角色必须同时授予对账户 A 中存储桶和密钥的访问权限。

以下过程描述如何授予这些权限中的每个权限。

向账户 b 中的用户授予对账户 a 中存储桶的访问权限

- 在账户 A 中，查看 S3 存储桶策略并确认存在允许从账户 B 的账户 ID 访问的语句。

  例如，以下存储桶策略允许 `s3:GetObject` 访问账户 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"
        ]
      }
    }
  ]
}
```

从账户 a 中的 AWS KMS 密钥策略向账户 b 中的用户授予访问权限

1. 在账户 A 的 AWS KMS 密钥策略中，向账户 B 所担任的角色授予执行以下操作的权限：

   - `kms:Encrypt`

- kms:Decrypt
- kms:ReEncrypt*
- kms:GenerateDataKey*
- kms:DescribeKey

以下示例仅向一个 IAM 角色授予密钥访问权限。

```
{
    "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. 在账户 A 中，使用 AWS Management Console 策略视图查看密钥策略。
3. 在密钥策略中，验证以下语句是否将账户 B 列为委托人。

```
"Sid": "Allow use of the key"
```

4. 如果 "Sid": "Allow use of the key" 语句不存在，请执行以下步骤：
   a. 切换到使用控制台默认视图查看密钥策略。
   b. 将账户 B 的账户 ID 添加为可访问密钥的外部账户。

从账户 b 担任的 IAM 角色中授予对账户 a 中存储桶和密钥的访问权限

1. 从账户 B 中，在 https://console.aws.amazon.com/iam/ 开启 IAM 控制台。
2. 打开与账户 B 中用户关联的 IAM 角色。
3. 查看应用于 IAM 角色的权限策略列表。
4. 确保应用授予对存储桶的访问权限的策略。

   以下示例语句授予 IAM 角色对存储桶 `DOC-EXAMPLE-BUCKET` 执行 `s3:GetObject` 和 `s3:PutObject` 操作的访问权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ExampleStmt2",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
    }
  ]
}
```

5. 确保应用授予对密钥的访问权限的策略。

   > **Note**
   >
   > 如果账户 B 所担任的 IAM 角色已经具有管理员访问权限，则您无需从用户的 IAM policy 授予对密钥的访问权限。

   以下示例语句授予 IAM 角色使用密钥 `arn:aws:kms:us-west-2:123456789098:key/111aa2bb-333c-4d44-5555-a111bb2c33dd` 的权限。

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

跨账户访问存储桶对象

由存储桶拥有账户（账户 A）以外的账户（账户 C）上载的对象可能需要明确的对象级 ACL，以向查询账户（账户 B）授予读取访问权限。为避免此要求，账户 C 在将对象放入账户 A 的存储桶之前应代入账户 A 中的角色。有关更多信息，请参阅如何提供对 Amazon S3 存储桶中的对象的跨账户访问权限？。

针对 AWS Glue Data Catalog 中数据库和表的精细访问权限

如果您将 AWS Glue Data Catalog 与 Amazon Athena 结合使用，则可以为 Athena 中使用的数据库和表 Data Catalog 对象定义资源级策略。

> **Note**
>
> 此处的“精细访问控制”一词是指数据库和表级别的安全性。有关列级别、行级别和单元格级别安全性的信息，请参阅 Lake Formation 中的数据筛选和单元格级安全性。

在 IAM 基于身份的策略中定义资源级别权限。

> **Important**
>
> 本部分介绍 IAM 基于身份的策略中的资源级权限。这些权限与基于资源的策略不同。有关差异的更多信息，请参阅 IAM 用户指南 中的 基于身份的策略与基于资源的策略。

对于这些任务，请参阅以下主题：

执行此任务	请参阅以下主题
创建定义针对资源的精细访问权限的 IAM policy	《IAM 用户指南》中的 创建 IAM policy 。
了解 AWS Glue 中使用 IAM 基于身份的策略	AWS Glue 开发人员指南 中的 基于身份的策略（IAM policy）。

本节内容

限制

在对 AWS Glue Data Catalog 和 Athena 使用精细访问控制时，需要考虑以下限制：

- 启用了 IAM Identity Center 的 Athena 工作组要求将 Lake Formation 配置为使用 IAM Identity Center 身份。有关更多信息，请参阅《AWS Lake Formation 用户指南》中的 Integrating IAM Identity Center。
- 您只能限制对数据库和表的访问。精细访问控制在表级别上应用，并且您无法限制对表中单独分区的访问。有关更多信息，请参阅 AWS Glue 中的表分区和版本。
- AWS Glue Data Catalog包含以下资源：`CATALOG`、`DATABASE`、`TABLE` 和 `FUNCTION`。

> **Note**
>
> 对于此列表，各个账户的在 Athena 与 AWS Glue Data Catalog 之间公用的资源包括 `TABLE`、`DATABASE` 和 `CATALOG`，而 `Function` 特定于 AWS Glue。对于 Athena 中的删除操作，您必须包含对 AWS Glue 操作的权限。请参阅 针对表和数据库的精细权限的示例。

层次结构如下所示：CATALOG 是每个账户中所有 DATABASES 的原级，DATABASE 是其所有 TABLES 和 FUNCTIONS 的原级。例如，在您账户的目录中，对于属于 db 数据库的名为 table_test 的表，其原级为您账户中的 db 和目录。对于 db 数据库，其原级是您的账户中的目录，其后代是表和函数。有关资源的层次结构的更多信息，请参阅《AWS Glue 开发人员指南》中的 Data Catalog 中的 ARN 列表。

- 对于资源上的任何非删除 Athena 操作，例如 CREATE DATABASE、CREATE TABLE、SHOW DATABASE、SHOW TABLE 或 ALTER TABLE，您需要在资源（表或数据库）上以及该资源在 Data Catalog 中的所有原级上调用该操作的权限。例如，对于表来说，其原级是所属的数据库以及账户的目录。对于数据库来说，其原级是账户的目录。请参阅 针对表和数据库的精细权限的示例。
- 对于 Athena 中的删除操作（例如 DROP DATABASE 或 DROP TABLE），您还需要拥有在该数据目录中资源的所有祖先级和子孙级上调用删除操作的权限。例如，要删除数据库，您需要数据库、目录（作为数据库的原级）以及所有表和用户定义的函数（作为数据库的子代）的权限。表没有子代。要运行 DROP TABLE，您需要对该表、该表所属的数据库以及目录执行该操作的权限。请参阅 针对表和数据库的精细权限的示例。

每个 AWS 区域 对目录和数据库的 AWS Glue 访问

为了让 Athena 使用 AWS Glue，需要制定一项策略，授予访问您的数据库和您账户中每个 AWS 区域的 AWS Glue Data Catalog 的相应权限。要创建数据库，还需要 CreateDatabase 权限。在以下示例策略中，将 AWS 区域、AWS 账户 ID 和数据库名称替换为您自己的名称。

```
{
   "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"
   ]
}
```

AWS Glue 中的表分区和版本

在 AWS Glue 中，表可以具有分区和版本。表版本和分区在 AWS Glue 中不视为独立的资源。通过授予对表及表的原级资源的访问权限，授予对该表版本和分区的访问权限。

出于精细访问控制的目的，以下访问权限适用：

- 精细访问控制在表级别应用。您只能限制对数据库和表的访问。例如，如果您允许访问某个分区表，此访问权限应用到该表的所有分区。您无法限制对表内各个分区的访问。

  > **Important**
  >
  > 要在分区的 AWS Glue 上运行操作，需要目录、数据库和表级别的分区操作权限。仅仅访问表中的分区是不够的。例如，要在数据库 `myDB` 中的表 `myTable` 上运行 `GetPartitions`，必须授予 `glue:GetPartitions` 对目录、`myDB` 数据库和 `myTable` 资源的权限。

- 精细访问控制不适用于表版本。对于分区，对表以前版本的访问权限通过在 AWS Glue 中，在表上对表版本 API 的访问权限以及表原级的访问权限来授予。

有关对 AWS Glue 操作的权限信息，请参阅《AWS Glue 开发人员指南》中的 AWS Glue API 权限：操作和资源参考。

针对表和数据库的精细权限的示例

下表列出了 IAM 基于身份的策略的示例，这些策略允许对 Athena 中的数据库和表的精细访问。建议您从这些示例开始，根据您的需求，调整它们以允许或拒绝对特定数据库和表的特定操作。

这些示例包括访问数据库和目录，以便 Athena 和 AWS Glue 可以协同工作。对于多个 AWS 区域，请为每个数据库和目录包括类似策略，每个区域一行。

在示例中，将 `example_db` 数据库名称和 `test` 表名称替换为您的数据库名称和表名称。

DDL 语句	授予对资源的访问权限的 IAM 访问策略的示例

ALTER DATABASE	允许修改 `example_db` 数据库的属性。
	`{`
	`"Effect": "Allow",`
	`"Action": [`

DDL 语句	授予对资源的访问权限的 IAM 访问策略的示例

	<pre>"glue:GetDatabase",
	"glue:UpdateDatabase"
	],
	"Resource": [
	"arn:aws:glue: *us-east-1* :*123456789012* :catalog",
	"arn:aws:glue: *us-east-1* :*123456789012* :database
	/ *example_db* "
	]
	}</pre>

CREATE DATABASE	允许创建名为 example_db 的数据库。
	<pre>{
	"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* "
	]
	}</pre>

DDL 语句	授予对资源的访问权限的 IAM 访问策略的示例
CREATE TABLE	允许在 example_db 数据库中创建名为 test 的表。

```
{
   "Sid": "DatabasePermissions",
   "Effect": "Allow",
   "Action": [
      "glue:GetDatabase",
      "glue:GetDatabases"
   ],
   "Resource": [
     "arn:aws:glue: us-east-1 :123456789012 :catalog",
     "arn:aws:glue: us-east-1 :123456789012 :database
/ example_db "
   ]
},
{
   "Sid": "TablePermissions",
   "Effect": "Allow",
   "Action": [
      "glue:GetTables",
      "glue:GetTable",
      "glue:GetPartitions",
      "glue:CreateTable"
   ],
   "Resource": [
     "arn:aws:glue: us-east-1 :123456789012 :catalog",
     "arn:aws:glue: us-east-1 :123456789012 :database
/ example_db ",
     "arn:aws:glue: us-east-1 :123456789012 :table/example_d
b /test"
   ]
}
```

DDL 语句	授予对资源的访问权限的 IAM 访问策略的示例

DROP DATABASE	允许删除 example_db 数据库（包括其中的所有表）。
	
	{
	"Effect": "Allow",
	"Action": [
	"glue:GetDatabase",
	"glue:DeleteDatabase",
	"glue:GetTables",
	"glue:GetTable",
	"glue:DeleteTable"
	],
	"Resource": [
	"arn:aws:glue: *us-east-1* :*123456789012* :catalog",
	"arn:aws:glue: *us-east-1* :*123456789012* :database/ *example_db* ",
	"arn:aws:glue: *us-east-1* :*123456789012* :table/*example_db* /*",
	"arn:aws:glue: *us-east-1* :*123456789012* :userDefinedFunction/ *example_db* /*"
	]
	}

DDL 语句	授予对资源的访问权限的 IAM 访问策略的示例

DROP TABLE	允许删除 example_db 数据库中名为 test 的分区表。如果您的表没有分区，请勿包含分区操作。
	
	{
	"Effect": "Allow",
	"Action": [
	"glue:GetDatabase",
	"glue:GetTable",
	"glue:DeleteTable",
	"glue:GetPartitions",
	"glue:GetPartition",
	"glue:DeletePartition"
	],
	"Resource": [
	"arn:aws:glue: *us-east-1* :*123456789012* :catalog",
	"arn:aws:glue: *us-east-1* :*123456789012* :database / *example_db* ",
	"arn:aws:glue: *us-east-1* :*123456789012* :table/*example_db* /*test*"
	]
	}

DDL 语句	授予对资源的访问权限的 IAM 访问策略的示例
MSCK REPAIR TABLE	将 Hive 兼容分区添加到名为 `test` 的表之后，允许更新目录元数据中的 `example_db` 数据库。 `{ "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	允许列出 AWS Glue Data Catalog 中的所有数据库。 `{ "Effect": "Allow", "Action": [ "glue:GetDatabase", "glue:GetDatabases" ], "Resource": [ "arn:aws:glue: us-east-1 :123456789012 :catalog", "arn:aws:glue: us-east-1 :123456789012 :database/*" ] }`

DDL 语句	授予对资源的访问权限的 IAM 访问策略的示例

SHOW TABLES	允许列出 example_db 数据库中的所有表。
	
	{
	"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_d*
	b /*"
	]
	}

授予 AWS Glue 数据目录跨账户访问权限

您可以使用 Athena 的跨账户 AWS Glue 目录功能来从您自己的账户以外的账户注册 AWS Glue 目录。为 AWS Glue 配置所需 IAM 权限并将目录注册为 Athena DataCatalog 资源后，您可以使用 Athena 运行跨账户查询。要了解如何使用 Athena 控制台注册来自其他账户的目录，请参阅 从另一个账户注册 AWS Glue Data Catalog。

有关 AWS Glue 中跨账户访问的更多信息，请参阅《AWS Glue 开发人员指南》中的 授予跨账户访问权限。

开始之前

因为此功能使用现有的 Athena DataCatalog 资源 API 和功能来启用跨账户访问，我们建议您先阅读以下资源：

- 连接到数据源 - 包含有关将 Athena 与 AWS Glue、Hive 或 Lambda 数据目录源搭配使用的主题。
- 数据目录示例策略 - 介绍如何编写控制对数据目录访问权限的策略。
- 将 AWS CLI 与 Hive 元存储结合使用 - 介绍如何使用带有 Hive 元存储仓的 AWS CLI，但包含适用于其他数据源的使用案例。

注意事项和限制

当前，Athena 跨账户 AWS Glue 目录访问具有以下限制：

- 此功能仅在支持 Athena 引擎版本 2 或更高版本的 AWS 区域可用。有关 Athena 引擎版本的更多信息，请参阅 Athena 引擎版本控制。要升级工作组的引擎版本，请参阅 更改 Athena 引擎版本。
- 当您在账户中注册另一个账户的 AWS Glue Data Catalog 时，您可以创建一个区域 `DataCatalog` 资源，该资源将仅链接到该特定区域中的其他账户数据。
- 当前，包含跨账户 AWS Glue 目录的 `CREATE VIEW` 语句不受支持。
- 无法跨账户查询使用 AWS 托管密钥进行加密的目录。对于希望跨账户查询的目录，请改用客户托管密钥 (KMS_CMK)。有关客户托管密钥和 AWS 托管密钥之间差异的信息，请参阅《AWS Key Management Service Developer Guide》中的 Customer keys and AWS keys。

开始使用

在下列应用场景中，“借入者”账户（666666666666）想要运行 `SELECT` 查询，该查询引用的 AWS Glue 目录属于“拥有者”账户（999999999999），如下例所示：

```
SELECT * FROM ownerCatalog.tpch1000.customer
```

在以下过程中，步骤 1a 和 1b 显示了如何授予借入者账户对拥有者账户 AWS Glue 资源的访问权限，从借入者和拥有者这两个角度均进行了展示。该示例授予对数据库 `tpch1000` 和表 `customer` 的访问权限。更改这些示例名称以满足您的要求。

步骤 1a：创建借入者角色以及可访问拥有者 AWS Glue 资源的策略

要创建借入者账户角色以及可访问拥有者账户 AWS Glue 资源的策略，可以使用 AWS Identity and Access Management（IAM）控制台或 IAM API。以下过程使用了 IAM 控制台。

创建借入者角色以及可访问拥有者账户 AWS Glue 资源的策略

1. 通过借入者账户从 https://console.aws.amazon.com/iam/ 登录 IAM 控制台。
2. 在导航窗格中，展开访问管理，然后选择策略。
3. 选择创建策略。
4. 对于策略编辑器，选择 JSON。
5. 在策略编辑器中，输入以下策略，然后根据您的要求对其进行修改：

```
{
```

```
    "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. 选择下一步。
7. 在查看并创建页面上，输入策略的名称（例如 **CrossGluePolicyForBorrowerRole**）作为策略名称。
8. 选择创建策略。
9. 在导航窗格中，选择角色。
10. 选择 Create role（创建角色）。
11. 在选择信任的实体页面上，选择 AWS 账户，然后选择下一步。
12. 在添加权限页面上，在搜索框中输入所创建策略的名称（例如 **CrossGluePolicyForBorrowerRole**）。
13. 选择策略名称旁的复选框，然后选择下一步。
14. 在 Name, review, and create（命名、检查和创建）页面上，对于 Role name（角色名称），输入角色的名称（例如 **CrossGlueBorrowerRole**）。
15. 选择 Create role(创建角色)。

步骤 1b：创建要向借入者授予 AWS Glue 访问权限的拥有者策略

要从拥有者账户（999999999999）向借入者角色授予 AWS Glue 访问权限，可以使用 AWS Glue 控制台或 AWS Glue PutResourcePolicy API 操作。以下步骤使用了 AWS Glue 控制台。

从拥有者账户向借入者账户授予 AWS Glue 访问权限

1. 通过拥有者账户从 https://console.aws.amazon.com/glue/ 登录 AWS Glue 控制台。
2. 在导航窗格中，展开数据目录，然后选择目录设置。

3. 在 Permissions（权限）框中，输入类似如下的策略。对于 *rolename*，输入借入者在步骤 1a 中创建的角色（例如 **CrossGlueBorrowerRole**）。如果要增加权限范围，您可以将通配符 * 用于数据库和表资源类型。

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

完成后，建议使用 AWS Glue API 进行一些测试跨账户调用，确认权限是否按预期进行配置。

第 2 步：借入者注册属于拥有者账户的 AWS Glue Data Catalog

以下过程说明了如何使用 Athena 控制台将拥有者的 Amazon Web Services 账户中的 AWS Glue Data Catalog 配置为数据来源。要了解如何使用 API 操作而不是通过控制台来注册目录，请参阅 使用 API 注册属于拥有者账户的 Athena 数据目录。

注册属于其他账户的 AWS Glue Data Catalog

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。

3. 展开管理，然后选择数据来源。
4. 选择右上角的 Connect data source（连接数据来源）。
5. 在选择数据来源页面上，对于数据来源，选择 S3 - AWS Glue Data Catalog，然后选择下一步。
6. 在输入数据来源详细信息页面的 AWS Glue Data Catalog 部分中，对于选择 AWS Glue Data Catalog，选择 AWS Glue Data Catalog 中的其他账户。
7. 对于 Data source details（数据源详细信息），提供以下信息：
   - Data source name（数据源名称）– 输入要在 SQL 查询中使用的名称，以引用其他账户中的数据目录。
   - Description（描述）–（可选）输入其他账户中数据目录的描述。
   - Catalog ID（目录编号）– 输入数据目录所属账户的 12 位 Amazon Web Services 账户 ID。Amazon Web Services 账户 ID 即是目录 ID。
8. （可选）展开标签，输入要与数据来源关联的键-值对。有关标签的更多信息，请参阅 为 Athena 资源添加标签。
9. 选择下一步。
10. 在 Review and create（审核和创建）页面中，检查数据来源的详细信息，然后选择 Create data source（创建数据来源）。Data source details（数据来源详细信息）页面列出了所注册数据目录的数据库和标签。
11. 选择 Data Source（数据源）。您注册的数据目录在 Data source name（数据来源名称）列中列出。
12. 要查看或编辑有关数据目录的信息，请选择该目录，然后选择Actions（操作）、Edit（编辑）。
13. 要删除新数据目录，请选择该目录，然后选择 Actions（操作）、Delete（删除）。

步骤 3：借入者提交查询

借入者提交一个查询，该查询使用 *catalog.database.table* 语法引用目录，如下例所示：

```
SELECT * FROM ownerCatalog.tpch1000.customer
```

借入者如不使用完全限定的语法，也可以通过 QueryExecutionContext 传入目录，从而根据上下文指定目录。

其他 Amazon S3 权限

- 如果借入者账户使用 Athena 查询将新数据写入拥有者账户中的表，即使该表存在于拥有者的账户中，拥有者也将无法自动获取对 Amazon S3 中此数据的访问权限。这是因为借入者就是 Amazon S3 中信息的对象拥有者，另有配置的情况除外。要授予拥有者对数据的访问权限，请作为附加步骤相应地设置对象的权限。
- 某些跨账户 DDL 操作，如 MSCK REPAIR TABLE 需要 Amazon S3 权限。例如，如果借入者账户正在根据拥有者账户中的表（该表的数据位于拥有者账户 S3 存储桶中）执行跨账户 `MSCK REPAIR` 操作，则该存储桶必须向借入者所担任的角色授予权限才能成功查询。

有关授予存储桶权限的信息，请参阅《Amazon Simple Storage Service 用户指南》中的如何设置 ACL 存储桶权限？。

动态使用目录

在某些情况下，您可能需要快速对跨账户 AWS Glue 目录执行测试，而无需执行注册它的先决步骤。如果您已按照本文档前述说明正确配置了所需的 IAM 和 Amazon S3 权限，则您可以动态执行跨账户查询，而无需创建 `DataCatalog` 资源对象。

要在不注册的情况下显式引用目录，请使用以下示例中的语法：

```
SELECT * FROM "glue:arn:aws:glue:us-east-1:999999999999:catalog".tpch1000.customer
```

使用格式“`glue:`*<arn>*”，其中 *<arn>* 是您想要使用的 AWS Glue Data Catalog ARN。在该示例中，Athena 使用此语法动态指向账户 999999999999 的 AWS Glue 数据目录，如同您已为其单独创建了 `DataCatalog` 对象。

使用动态目录的注意事项

使用动态目录时，请记住以下几点。

- 使用动态目录需要您通常用于 Athena Data Catalog API 操作的 IAM 权限。主要区别在于，Data Catalog 资源名称遵循 `glue:*` 命名约定。
- 目录 ARN 必须属于正在运行查询的同一区域。
- 在 DML 查询或视图中使用动态目录时，请使用转义双引号 (\")。在 DDL 查询中使用动态目录时，请使用反引号字符 (`)。

使用 API 注册属于拥有者账户的 Athena 数据目录

您可以使用 API 操作来注册属于拥有者账户的数据目录，而不是如第 2 步中所述使用 Athena 控制台。

Athena DataCatalog 资源的创建者必须具有运行 Athena CreateDataCatalog API 操作的必要权限。根据您的要求，可能需要访问其他 API 操作。有关更多信息，请参阅 数据目录示例策略。

以下 `CreateDataCatalog` 请求正文为跨账户访问注册了 AWS Glue 目录：

```
# 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"
}
```

以下示例代码使用 Java 客户端创建 `DataCatalog` 对象。

```
# 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);
```

完成这些步骤后，借入者应会在调用 ListDataCatalogs API 操作时看到 `ownerCatalog`。

其他 资源

- 从另一个账户注册 AWS Glue Data Catalog

- AWS 规范性指导模式指南中的使用 Amazon Athena 配置对共享 AWS Glue Data Catalog 的跨账户访问权限。
- AWS 大数据博客中的使用 Amazon Athena 查询跨账户 AWS Glue Data Catalog
- 在《AWS Glue 开发人员指南》中的授予跨账户访问权限

从 Athena 访问 AWS Glue Data Catalog 中的加密元数据

如果您将 AWS Glue Data Catalog 与 Amazon Athena 一起使用，您可以使用 AWS Glue 控制台或 API在 AWS Glue Data Catalog 中启用加密。想要了解有关信息，请参阅《AWS Glue 开发人员指南》中的 加密数据目录。

如果对 AWS Glue Data Catalog 加密，则必须将以下操作添加到用于访问 Athena 的所有策略：

```
{
 "Version": "2012-10-17",
 "Statement": {
 "Effect": "Allow",
     "Action": [
           "kms:GenerateDataKey",
           "kms:Decrypt",
           "kms:Encrypt"
      ],
     "Resource": "(arn of the key used to encrypt the catalog)"
   }
}
```

每当您使用 IAM 策略时，请确保遵循 IAM 最佳实践。有关更多信息，请参阅《IAM 用户指南》中的 IAM 安全最佳实践。

对工作组和标签的访问

工作组是由 Athena 管理的资源。因此，如果您的工作组策略使用获取 `workgroup` 作为输入的操作，您必须如下所示指定工作组的 ARN，其中 *`workgroup-name`* 是您的工作组的名称：

```
"Resource": [arn:aws:athena:region:AWSAcctID:workgroup/workgroup-name]
```

例如，对于 `us-west-2` 区域中用于亚马逊云科技账户 `123456789012` 的名为 `test_workgroup` 的工作组，请使用以下 ARN 指定工作组作为资源：

```
"Resource":["arn:aws:athena:us-east-2:123456789012:workgroup/test_workgroup"]
```

要访问启用了可信身份传播（TIP）的工作组，必须将 IAM Identity Center 用户分配给由 Athena GetWorkGroup API 操作的响应返回的 `IdentityCenterApplicationArn`。

- 有关工作组策略的列表，请参阅the section called “工作组策略示例”。
- 有关工作组的基于标签的策略列表，请参阅基于标签的 IAM 访问控制策略。
- 有关为工作组创建 IAM policy 的更多信息，请参阅 用于访问工作组的 IAM policy。
- 有关 Amazon Athena 操作的完整列表，请参阅 Amazon Athena API 参考中的 API 操作名称。
- 有关 IAM policy 的更多信息，请参阅《IAM 用户指南》中的使用可视化编辑器创建策略。

每当您使用 IAM 策略时，请确保遵循 IAM 最佳实践。有关更多信息，请参阅《IAM 用户指南》中的 IAM 安全最佳实践。

允许访问预编译语句

本主题介绍了 Amazon Athena 中可用于预编译语句的 IAM 权限。每当您使用 IAM 策略时，请确保遵循 IAM 最佳实践。有关更多信息，请参阅 IAM 用户指南中的 IAM 安全最佳实践。

有关预编译语句的更多信息，请参阅 使用参数化查询。

创建、管理和执行预编译语句需要以下 IAM 权限。

```
athena:CreatePreparedStatement
athena:UpdatePreparedStatement
athena:GetPreparedStatement
athena:ListPreparedStatements
athena:DeletePreparedStatement
```

如下表所示，使用这些权限。

要执行的操作	使用这些权限
运行 PREPARE 查询	`athena:StartQueryExecution` `athena:CreatePreparedStatement`
重新运行 PREPARE 查询以更新现有预编译语句	`athena:StartQueryExecution` `athena:UpdatePreparedStatement`

要执行的操作	使用这些权限
运行 EXECUTE 查询	athena:StartQueryExecution athena:GetPreparedStatement
运行 DEALLOCATE PREPARE 查询	athena:StartQueryExecution athena:DeletePreparedStatement

示例

在以下示例中，IAM 策略授予对指定账户 ID 和工作组管理和运行预编译语句的权限。

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

将 Athena 与 CalledVia 上下文键结合使用

主体 向 AWS 提交请求时，AWS 会将请求信息收集到请求上下文，以评估并授权该请求。您可以使用 JSON 策略的 Condition 元素将请求上下文中的键与您在策略中指定的键值进行比较。全局条件上下文键是带 aws: 前缀的条件键。

aws:CalledVia 上下文键

您可以使用 aws:CalledVia 全局条件上下文键将策略中的服务与代表 IAM 委托人（用户或角色）发出请求的服务进行比较。主体向 AWS 服务 发出请求时，该服务可能会使用主体的凭证向其他服务发出后续请求。aws:CalledVia 键包含链中代表主体发出请求的每个服务的有序列表。

通过指定 aws:CalledVia 上下文键的服务主体名称，您可以将上下文键设置为特定于 AWS 服务。例如，您可以使用 aws:CalledVia 条件键将请求限制为仅从 Athena 发出的请求。要借助 Athena 在策略中使用 aws:CalledVia 条件键，您可以指定 Athena 服务委托人名称 athena.amazonaws.com，如以下示例所示。

```
 ...
    "Condition": {
        "ForAnyValue:StringEquals": {
            "aws:CalledVia": "athena.amazonaws.com"
        }
    }
...
```

您可以使用 aws:CalledVia 上下文键来确保调用者只有在从 Athena 调用资源时才能访问资源（如 Lambda 函数）。

> **Note**
>
> aws:CalledVia 上下文键与可信身份传播功能不兼容。

添加一个可选的 CalledVia 上下文密钥，用于对 Lambda 函数进行细化访问

Athena 要求调用者有 lambda:InvokeFunction 权限，以便调用与查询关联的 Lambda 函数。以下语句允许对 Lambda 函数进行细粒度访问，以便用户只能使用 Athena 调用 Lambda 函数。

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

下例说明了将前一语句添加到策略中以允许用户运行和读取联合查询。允许执行这些操作的委托人可以运行指定与联合数据源关联的 Athena 目录的查询。但是，除非通过 Athena 调用该函数，否则委托人无法访问关联的 Lambda 函数。

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

有关 CalledVia 条件键的更多信息，请参阅《IAM 用户指南》中的 AWS 全局条件上下文键。

允许访问适用于外部配置 Hive 元存储的 Athena 数据连接器

本主题中的权限策略示例演示了需要允许的操作以及允许执行这些操作的资源。在将类似的权限策略附加到 IAM 身份之前，请仔细检查这些策略并根据您的需求修改它们。

- 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 – 允许 IAM 委托人使用适用于外部 Hive 元存储的 Athena 数据连接器查询数据

除了对 Athena 操作授予完全访问权限的 AWS 托管策略：AmazonAthenaFullAccess 之外，还将以下策略附加到 IAM 委托人。

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

权限说明

允许的操作	说明
`"s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket", "s3:PutObject", "s3:ListMultipartUploadParts",`	s3 操作允许从指定为 "arn:aws:s3::: *MyLambdaSpillBucket* /*MyLambdaSpillLocation* " 的资源读取和写入到该资源，其中 *MyLambdaSpillLocation* 标识在调用的一个或多

允许的操作	说明
`"s3:AbortMultipartUpload"`	个 Lambda 函数的配置中指定的溢出存储桶。仅当您使用 Lambda 层创建自定义运行时依赖关系以减少部署时函数构件大小时，才需要 *arn:aws:lambda:*: MyAWSAcctId :layer:MyAthenaLambdaLayer :** 资源标识符。最后一个位置的 * 是图层版本的通配符。

`"lambda:GetFunction",`
`"lambda:GetLayerVersion",`
`"lambda:InvokeFunction"`
	允许查询调用 Resource 块中指定的 AWS Lambda 函数。例如 arn:aws:lambda:*: *MyAWSAcctId* :function: *MyAthenaLambdaFunction* ，其中 *MyAthenaLambdaFunction* 指定要调用的 Lambda 函数的名称。如示例中所示，可以指定多个函数。

Example – 允许 IAM 委托人创建适用于外部 Hive 元存储的 Athena 数据连接器

除了对 Athena 操作授予完全访问权限的 AWS 托管策略：AmazonAthenaFullAccess 之外，还将以下策略附加到 IAM 委托人。

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

权限说明

允许查询为 Resource 块中指定的 AWS Lambda 函数调用 AWS Lambda 函数。例如 arn:aws:lambda:*:*MyAWSAcctId*:function:*MyAthenaLambdaFunction*，其中 *MyAthenaLambdaFunction* 指定要调用的 Lambda 函数的名称。如示例中所示，可以指定多个函数。

允许 Lambda 函数访问外部 Hive 元存储

要在账户中调用 Lambda 函数，您必须创建具有以下权限的角色：

- AWSLambdaVPCAccessExecutionRole – AWS Lambda 执行角色权限，用于管理将函数连接到 VPC 的弹性网络接口。确保您有足够数量的可用网络接口和 IP 地址。
- AmazonAthenaFullAccess – AmazonAthenaFullAccess 托管式策略授予对 Athena 的完全访问权限。
- 一个 Amazon S3 策略，该策略允许 Lambda 函数对 S3 进行写入并允许 Athena 从 S3 中进行读取。

例如，以下策略定义溢出位置 s3:\\mybucket\spill 的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
```

```
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/spill"
            ]
        }
    ]
}
```

每当您使用 IAM 策略时，请确保遵循 IAM 最佳实践。有关更多信息，请参阅 IAM 用户指南中的 IAM 安全最佳实践。

创建 Lambda 函数

要在您的账户中创建 Lambda 函数，需要函数开发权限或 AWSLambdaFullAccess 角色。有关更多信息，请参阅 AWS Lambda 的基于身份的 IAM 策略。

由于 Athena 使用 AWS Serverless Application Repository 创建 Lambda 函数，因此，创建 Lambda 函数的超级用户或管理员也应具有 IAM 策略以允许 Athena 联合查询。

目录注册和元数据 API 操作

要访问目录注册 API 和元数据 API 操作，请使用 AmazonAthenaFullAccess 托管策略。如果您不使用此策略，请将以下 API 操作添加到您的 Athena 策略：

```
{
    "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": [
                "*"
            ]
        }
    ]
```

```
}
```

跨区域 Lambda 调用

要在运行 Athena 查询的区域以外的区域中调用 Lambda 函数，请使用 Lambdaq 函数的完整 ARN。预设情况下，Athena 调用在同一区域中定义的 Lambda 函数。如果需要调用 Lambda 函数来访问运行 Athena 查询的区域以外的区域中的 Hive 元存储，您必须提供 Lambda 函数的完整 ARN。

例如，假设您在欧洲（法兰克福）区域 `eu-central-1` 定义目录 `ehms` 以便在美国东部（弗吉尼亚北部）区域使用以下 Lambda 函数。

```
arn:aws:lambda:us-east-1:111122223333:function:external-hms-service-new
```

当您通过此方式指定完整 ARN 时，Athena 会对 `us-east-1` 调用 `external-hms-service-new` Lambda 函数以从 `eu-central-1` 中提取 Hive 元存储数据。

> **Note**
>
> 应在运行 Athena 查询的相同区域中注册目录 `ehms`。

跨账户 Lambda 调用

有时，您可能需要从其他账户访问 Hive 元存储。例如，要运行 Hive 元存储，您可以从用于 Athena 查询的账户之外的其他账户启动 EMR 集群。不同的组或团队可能会在其 VPC 内使用不同的账户运行 Hive 元存储。或者，您可能希望访问来自不同的组或团队中的各个 Hive 元存储的元数据。

Athena 使用 AWS Lambda 对跨账户访问的支持来启用对 Hive 元存储的跨账户访问。

> **Note**
>
> 请注意，对 Athena 的跨账户访问通常意味着对 Amazon S3 中的元数据和数据的跨账户访问。

设想以下场景：

- 账户 `111122223333` 在 Athena 中的 us-east-1 上设置 Lambda 函数 `external-hms-service-new` 以访问 EMR 集群上运行的 Hive 元存储。
- 账户 `111122223333` 希望允许账户 `444455556666` 访问 Hive 元存储数据。

为了向账户 444455556666 授予对 Lambda 函数 external-hms-service-new 的访问权限，账户 111122223333 使用以下 AWS CLI add-permission 命令。该命令设置了格式以便于阅读。

```
$ 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\"}"
}
```

要检查 Lambda 权限，请使用 get-policy 命令，如以下示例中所示。该命令设置了格式以便于阅读。

```
$ 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\"}]}"
}
```

添加权限后，您可以在定义目录 ehms 时对 us-east-1 使用 Lambda 函数的完整 ARN，如下所示：

```
arn:aws:lambda:us-east-1:111122223333:function:external-hms-service-new
```

有关跨区域调用的信息，请参阅本主题前面的 跨区域 Lambda 调用。

授予对数据的跨账户访问权限

您必须先授予对 Amazon S3 中数据的跨账户访问权限，然后才能运行 Athena 查询。您可以通过下列方式之一来执行该操作：

- 使用规范用户 ID 更新 Amazon S3 存储桶的访问控制列表策略。
- 添加对 Amazon S3 存储桶策略的跨账户访问权限。

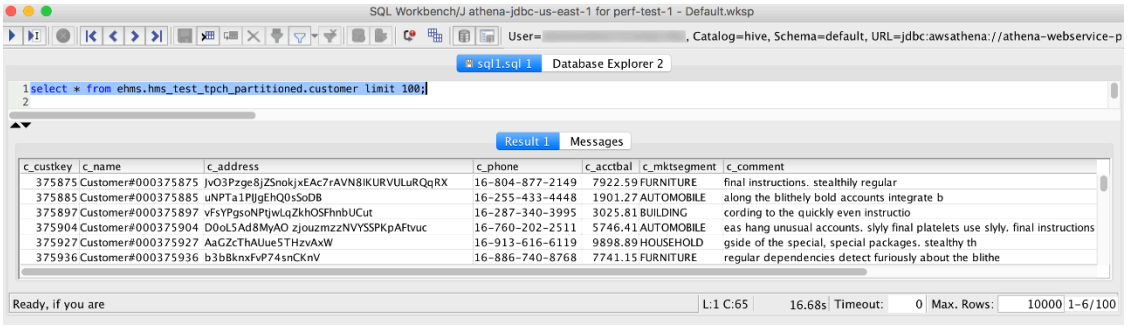
例如，将以下策略添加到账户 111122223333 中的 Amazon S3 存储桶策略以允许账户 444455556666 从指定的 Amazon S3 位置读取数据。

```
{
    "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**
>
> 您可能需要向 Amazon S3 授予对数据以及 Amazon S3 溢出位置的跨账户访问权限。当响应对象的大小超过给定阈值时，Lambda 函数会使额外数据流出到溢出位置。有关示例策略，请参阅本主题的开头部分。

在当前示例中，在向 444455556666，授予跨账户访问权限后，444455556666 可在其自己的 account 中使用目录 ehms 查询账户 111122223333 中定义的表。

在以下示例中，SQL Workbench 配置文件 perf-test-1 适用于账户 444455556666。查询使用目录 ehms 访问 Hive 元存储以及账户 111122223333 中的 Amazon S3 数据。



允许 Athena 联合查询的 IAM 权限策略示例

本主题中的权限策略示例演示了需要允许的操作以及允许执行这些操作的资源。仔细检查这些策略并根据您的需求修改它们，然后再将它们附加到 IAM 身份。

有关将 IAM 策略添加到用户的信息，请参阅 IAM 用户指南 中的 添加和删除 IAM 身份权限。

- 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 – 允许 IAM 委托人运行并使用 Athena 联合查询返回结果

以下基于身份的权限策略允许用户或其他 IAM 委托人执行使用 Athena 联合查询所需的操作。允许执行这些操作的委托人可以运行指定与联合数据源关联的 Athena 目录的查询。

```
{
    "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/*"
            ]
        }
    ]
}
```

权限说明

允许的操作	说明
`"athena:GetQueryExecution", "athena:GetQueryResults",`	运行联合查询所需的 Athena 权限。

允许的操作	说明
"athena:GetWorkGroup", "athena:StartQueryExecution", "athena:StopQueryExecution"	
"athena:GetDataCatalog", "athena:GetQueryExecution," "athena:GetQueryResults", "athena:GetWorkGroup", "athena:StartQueryExecution", "athena:StopQueryExecution"	运行联合视图查询所需的 Athena 权限。视图需要 GetDataCatalog 操作。
"lambda:InvokeFunction"	允许查询为 Resource 块中指定的 AWS Lambda 函数调用 AWS Lambda 函数。例如 arn:aws:lambda:*: *MyAWSAcctId* :function: *MyAthenaLambdaFunction* ，其中 *MyAthenaLambdaFunction* 指定要调用的 Lambda 函数的名称。如示例中所示，可以指定多个函数。

允许的操作	说明
"s3:AbortMultipartUpload", "s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket", "s3:ListMultipartUploadParts", "s3:PutObject"	必须具备 s3:ListBucket 和 s3:GetBucketLocation 权限，才能访问运行 StartQueryExecution 的 IAM 委托人的查询输出存储桶。 s3:PutObject 、s3:ListMultipartUploadParts 和 s3:AbortMultipartUpload 允许将查询结果写入由 arn:aws:s3::: *MyQueryResultsBucket* /* 资源标识符指定的查询结果存储桶的所有子文件夹，其中 *MyQueryResultsBucket* 为 Athena 查询结果存储桶。有关更多信息，请参阅 使用查询结果、最近查询和输出文件。 s3:GetObject 允许读取指定为 arn:aws:s3::: *MyQueryResultsBucket* 的资源的查询结果和查询历史记录，其中 *MyQueryResultsBucket* 是 Athena 查询结果存储桶。 s3:GetObject 还允许从指定为 "arn:aws:s3::: *MyLambdaSpillBucket* /*MyLambdaSpillPrefix* *" 的资源中读取，其中 *MyLambdaSpillPrefix* 在被调用的一个或多个 Lambda 函数的配置中指定。

Example – 允许 IAM 委托人创建数据源连接器

```
{
    "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/*"
        }
    ]
}
```

权限说明

允许的操作	说明
`"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",`	允许创建和管理列为资源的 Lambda 函数。在此示例中，资源标识符 `arn:aws:lambda:*:` *MyAWSAcctId* `:function:` *MyAthenaLambdaFunctionsPrefix* `*` 中使用名称前缀，其中 *MyAthenaLambdaFunctionsPrefix* 是一组 Lambda 函数的名称中使用的共享前缀，因此它们不需要单独指定为资源。您可以指定一个或多个 Lambda 函数资源。
`"s3:GetObject"`	允许读取 AWS Serverless Application Repository 所需的存储桶，如资源标识符 `arn:aws:s3:::awsserverlessrepo-changesets-` *1iiv3xa62ln3m* `/*` 所指定。此存储桶可能特定于您的账户。
`"cloudformation:*"`	允许由资源 *MyCFStackPrefix* 指定的 AWS CloudFormation 堆栈的创建和管理。这些堆栈和堆栈集是 AWS Serverless Application Repository 部署连接器和 UDF 的方式。
`"serverlessrepo:*"`	允许在由资源标识符 `arn:aws:serverlessrepo:*:*:applications/*` 指定的 AWS Serverless Application Repository 中搜索、查看、发布和更新应用程序。

允许 Amazon Athena 用户定义函数（UDF）的 IAM 权限策略示例

本主题中的权限策略示例演示了需要允许的操作以及允许执行这些操作的资源。在将类似的权限策略附加到 IAM 身份之前，请仔细检查这些策略并根据您的需求修改它们。

- 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 – 允许 IAM 委托人运行并返回包含 Athena UDF 语句的查询

以下基于身份的权限策略允许用户或其他 IAM 委托人执行使用 Athena UDF 语句运行查询所需的操作。

```
{
    "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": "*"
        }
    ]
}
```

权限说明

允许的操作	说明
`"athena:StartQueryExecution", "athena:GetQueryResults", "athena:GetWorkGroup", "athena:StopQueryExecution", "athena:GetQueryExecution",`	在 MyAthenaWorkGroup 工作组中运行查询所需的 Athena 权限。
`"s3:PutObject", "s3:GetObject", "s3:AbortMultipartUpload"`	s3:PutObject 和 s3:AbortMultipartUpload 允许将查询结果写入由 arn:aws:s3::: *MyQueryResultsBucket* /* 资源标识符指定的查询结果存储桶的所有子文件夹，其中 *MyQueryResultsBucket* 是 Athena 查询结果存储桶。有关更多信息，请参阅 使用查询结果、最近查询和输出文件。 s3:GetObject 允许读取指定为 arn:aws:s3::: *MyQueryResultsBucket* 的资源的查询结果和查询历史记录，其中 *MyQueryResultsBucket* 是 Athena 查询结果存储桶。有关更多信息，请参阅 使用查询结果、最近查询和输出文件。 s3:GetObject 还允许从指定为 "arn:aws:s3::: *MyLambdaSpillBucket* /*MyLambdaSpillPrefix* *" 的资源中读取，其中 *MyLambdaSpillPrefix* 在被调用的一个或多个 Lambda 函数的配置中指定。

允许的操作	说明
`"lambda:InvokeFunction"`	允许查询调用 `Resource` 块中指定的 AWS Lambda 函数。例如 `arn:aws:lambda:*:` *MyAWSAcctId* `:function:` *MyAthenaLambdaFunction* ，其中 *MyAthenaLambdaFunction* 指定要调用的 Lambda 函数的名称。如示例中所示，可以指定多个函数。

Example – 允许 IAM 委托人创建 Athena UDF

```
{
    "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/*"
        }
    ]
}
```

权限说明

允许的操作	说明
`"lambda:CreateFunction", "lambda:ListVersionsByFunction", "lambda:GetFunctionConfiguration", "lambda:PutFunctionConcurrency", "lambda:ListTags", "lambda:DeleteFunction", "lambda:GetAlias", "lambda:InvokeFunction", "lambda:GetFunction", "lambda:ListAliases", "lambda:UpdateFunctionConfigur ation", "lambda:UpdateFunctionCode", "lambda:AddPermission", "lambda:DeleteFunctionConcurrency", "lambda:RemovePermission", "lambda:GetPolicy" "lambda:GetAccountSettings",`	允许创建和管理列为资源的 Lambda 函数。在此示例中，资源标识符 `arn:aws:lambda:*:` *MyAWSAcctId* `:function:` *MyAthenaLambdaFunctionsPrefix* `*` 中使用名称前缀，其中 *MyAthenaLambdaFunctionsPrefix* 是一组 Lambda 函数的名称中使用的共享前缀，因此它们不需要单独指定为资源。您可以指定一个或多个 Lambda 函数资源。

允许的操作	说明
"lambda:ListFunctions", "lambda:ListEventSourceMappings",	
"s3:GetObject"	允许读取 AWS Serverless Application Repository 所需的存储桶，如资源标识符 arn:aws:s3:::awsserverlessrepo-changesets- *1iiv3xa62ln3m* /* 所指定。
"cloudformation:*"	允许由资源 *MyCFStackPrefix* 指定的 AWS CloudFormation 堆栈的创建和管理。这些堆栈和堆栈集是 AWS Serverless Application Repository 部署连接器和 UDF 的方式。
"serverlessrepo:*"	允许在由资源标识符 arn:aws:serverlessrepo:*:*:applications/* 指定的 AWS Serverless Application Repository 中搜索、查看、发布和更新应用程序。

允许使用 Athena 访问机器学习

必须允许运行 Athena ML 查询的 IAM 委托人对其使用的 Sagemaker 终端节点执行 sagemaker:invokeEndpoint 操作。在附加到用户身份且基于身份的权限策略中包括类似于下面的策略语句。此外，附加对 Athena 操作授予完全访问权限的 AWS 托管策略：AmazonAthenaFullAccess，或者附加允许操作子集的已修改内联策略。

将示例中的 arn:aws:sagemaker:*region*:*AWSAcctID*:*ModelEndpoint* 替换为要在查询中使用的模型终端节点的 ARN。有关更多信息，请参阅《服务授权参考》中的 SageMaker 的操作、资源和条件键。

```
{
            "Effect": "Allow",
            "Action": [
                "sagemaker:invokeEndpoint"
            ],
            "Resource": "arn:aws:sagemaker:us-west-2:123456789012:workteam/public-crowd/default"
}
```

每当您使用 IAM 策略时，请确保遵循 IAM 最佳实践。有关更多信息，请参阅《IAM 用户指南》中的 IAM 安全最佳实践。

启用对 Athena API 的联合身份访问

此部分介绍允许组织中的用户或客户端应用程序调用 Amazon Athena API 操作的联合身份访问。在这种情况下，组织的用户没有 Athena 的直接访问权限。相反，您在 AWS 外部的 Microsoft Active Directory 中管理用户凭证。Active Directory 支持 SAML 2.0（安全断言标记语言 2.0）。

要在此方案中对用户进行身份验证，请使用支持 SAML.2.0 的 JDBC 或 ODBC 驱动程序，访问 Active Directory 联合身份验证服务 (AD FS) 3.0 并允许客户端应用程序调用 Athena API 操作。

有关更多信息对于 AWS 的 SAML 2.0 支持，请参阅《IAM 用户指南》中的 关于 SAML 2.0 联合身份验证。

> **Note**
>
> 特定类型的身份提供商 (IdP)，即 Windows Server 中包括的 Active Directory 联合身份验证服务 (AD FS 3.0) 支持对 Athena API 的联合访问。联合访问与 IAM Identity Center 可信身份传播功能不兼容。访问通过支持 SAML 2.0 的 JDBC 或 ODBC 驱动程序版本建立。有关信息，请参阅通过 JDBC 连接到 Amazon Athena和通过 ODBC 连接到 Amazon Athena。

主题

- 开始前的准备工作
- 架构示意图
- 过程：对 Athena API 的基于 SAML 的联合访问

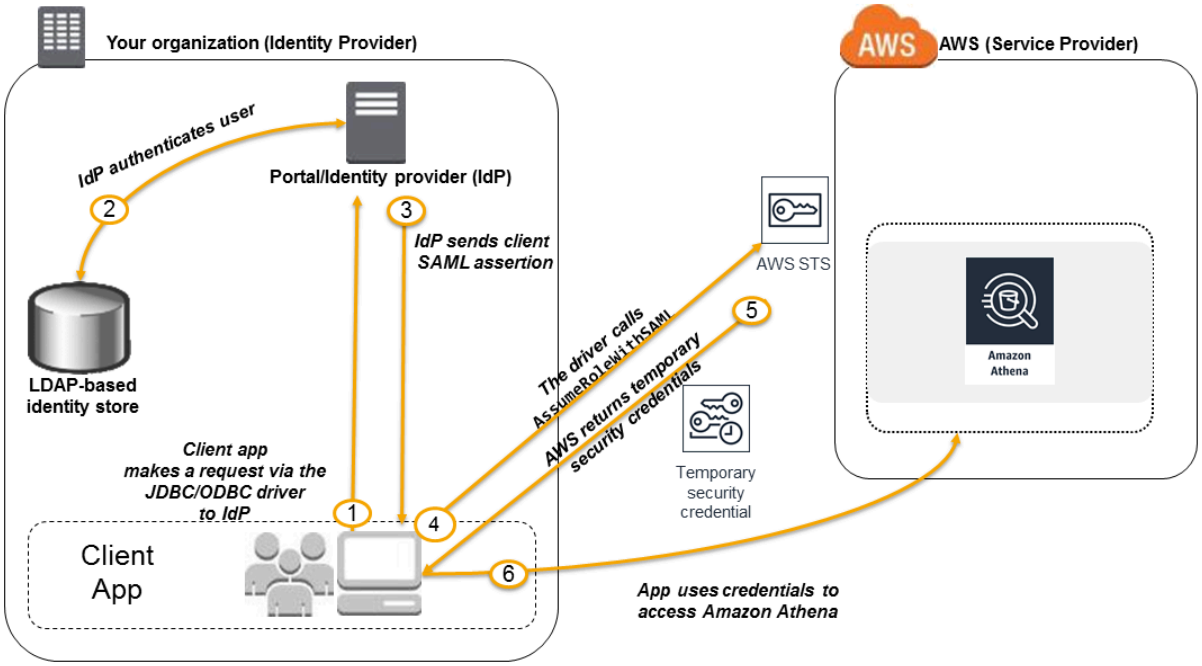
开始前的准备工作

在开始之前，请满足以下先决条件：

- 在组织中，安装并配置 ADFS 3.0 作为您的 IdP。
- 在用于访问 Athena 的客户端上，安装和配置最新可用版本的 JDBC 或 ODBC 驱动程序。驱动程序必须包括对与 SAML 2.0 兼容的联合身份访问的支持。有关信息，请参阅通过 JDBC 连接到 Amazon Athena和通过 ODBC 连接到 Amazon Athena。

架构示意图

下图阐明了此过程。



1. 您组织中的用户使用具有 JDBC 或 ODBC 驱动程序的客户端应用程序，请求组织的 IdP 进行身份验证。该 IdP 是 ADFS 3.0。
2. IdP 根据 Active Directory（组织的身份存储）对用户进行身份验证。
3. IdP 构建一个具有用户相关信息的 SAML 断言，并将此断言通过 JDBC 或 ODBC 驱动程序发送到客户端应用程序。
4. JDBC 或 ODBC 驱动程序调用 AWS Security Token Service AssumeRoleWithSAML API 操作，将其传递给以下参数：
   - SAML 提供商的 ARN
   - 要代入的角色的 ARN
   - 来自 IdP 的 SAML 断言

   有关更多信息，请参阅 AWS Security Token Service API 参考中的 AssumeRoleWithSAML。
5. 通过 JDBC 或 ODBC 驱动程序发送到客户端应用程序的 API 相应包含临时安全凭证。
6. 客户端应用程序使用临时安全凭证调用 Athena API 操作，从而允许您的用户访问 Athena API 操作。

过程：对 Athena API 的基于 SAML 的联合访问

该过程在组织的 IdP 与 AWS 账户之间建立信任关系，以实现对 Amazon Athena API 操作的基于 SAML 的联合访问。

要启用对 Athena API 的联合访问，请执行以下操作：

1. 在组织中，将 AWS 注册为您 IdP 中的服务提供商 (SP)。此过程称为信赖方信任。有关更多信息，请参阅《IAM 用户指南》中的 使用信赖方信任配置您的 SAML 2.0 IdP。作为此任务的一部分，请执行以下步骤：

   a. 从以下 URL 获取示例 SAML 元数据文档：https://signin.aws.amazon.com/static/saml-metadata.xml。

   b. 在您组织的 IdP (ADFS) 中，生成一个等同元数据 XML 文件，将您的 IdP 描述为 AWS 的身份提供商。您的元数据文件必须包括发布者名称、创建日期、过期日期以及 AWS 用来验证来自您组织的身份验证响应（断言）的密钥。

2. 在 IAM 控制台中，创建一个 SAML 身份提供程序实体。有关更多信息，请参阅《IAM 用户指南》中的 创建 SAML 身份提供商。作为此步骤的一部分，请执行以下操作：

   a. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。

   b. 上载此过程的第 1 步中由 IdP (ADFS) 生成的 SAML 元数据文档。

3. 在 IAM 控制台中，为您的 IdP 创建一个或多个 IAM 角色。有关更多信息，请参阅《IAM 用户指南》中的 为第三方身份提供商创建角色（联合访问）。作为此步骤的一部分，请执行以下操作：

   - 在角色的权限策略中，列出允许您组织的用户在 AWS 中执行的操作。
   - 在角色的信任策略中，将在此过程第 2 步中创建的 SAML 提供商实体设置为委托人。

   这将在您的组织与 AWS 之间建立信任关系。

4. 在您组织的 IdP (ADFS) 中，定义可将您组织中的用户或组映射到 IAM 角色的断言。将用户和组映射到 IAM 角色也称为断言规则。请注意，您的组织中不同的用户和组可能映射到不同的 IAM 角色。

   有关配置 ADFS 中的映射的信息，请参阅博客文章：使用 Windows Active Directory、ADFS 和 SAML 2.0 启用 AWS 的联合身份验证。

5. 安装和配置具有 SAML 2.0 支持的 JDBC 或 ODBC 驱动程序。有关信息，请参阅通过 JDBC 连接到 Amazon Athena和通过 ODBC 连接到 Amazon Athena。

6. 指定从您应用程序到 JDBC 或 ODBC 驱动程序的连接字符串。有关应用程序应使用的连接字符串的信息，请参阅《JDBC 驱动程序安装和配置指南》中的“使用 Active Directory 联合身份验证服务 (ADFS) 凭证提供程序”或《ODBC 驱动程序安装和配置指南》中的类似主题，可从 通过 JDBC 连接到 Amazon Athena 和 通过 ODBC 连接到 Amazon Athena 主题以 PDF 格式下载。

   下面大致概述了配置驱动程序的连接字符串：

   1. 在 `AwsCredentialsProviderClass configuration` 中，设置 `com.simba.athena.iamsupport.plugin.AdfsCredentialsProvider`，指示您希望通过 ADFS IdP 使用基于 SAML 2.0 的身份验证。
   2. 对于 `idp_host`，请提供 ADFS IdP 服务器的主机名。
   3. 对于 `idp_port`，请提供 ADFS IdP 侦听 SAML 断言请求的端口号。
   4. 对于 `UID` 和 `PWD`，请提供 AD 域用户凭证。在 Windows 上，如果使用驱动程序时未提供 `UID` 和 `PWD`，则驱动程序尝试获取登录到 Windows 计算机用户的用户凭证。
   5. （可选）将 `ssl_insecure` 设置为 `true`。在这种情况下，驱动程序不会检查 ADFS IdP 服务器的 SSL 证书的真实性。如果没有将 ADFS IdP 的 SSL 证书配置为受驱动程序信任，则需要设置为 `true`。
   6. 要启用 Active Directory 域用户或组与一个或多个 IAM 角色的映射（如此过程中第 4 步所述），请在 JDBC 或 ODBC 连接的 `preferred_role` 中，为驱动程序连接指定要代入的 IAM 角色 (ARN)。指定 `preferred_role` 是可选的，在角色不是断言规则中列出的第一个角色时非常有用。

   作为此过程的结果，将发生以下操作：

   1. JDBC 或 ODBC 驱动程序调用 AWS STS AssumeRoleWithSAML API，并向其传递断言，如架构示意图中的第 4 步所示。
   2. AWS 确保代入角色的请求来自 SAML 提供商实体中引用的 IdP。
   3. 如果请求成功，AWS STS AssumeRoleWithSAML API 操作会返回一组临时安全凭证，您的客户端应用程序即可用其向 Athena 发出已签名的请求。

      现在，您的应用程序拥有当前用户的相关信息，并且可以通过编程方式访问 Athena。

Athena 中的日志记录和监控

要检测事件、在发生事件时接收警报并进行响应，请针对 Amazon Athena 使用以下选项：

- 使用 AWS CloudTrail 监控 Athena – AWS CloudTrail 提供了 Athena 中的用户、角色或 AWS 服务的操作记录。它捕获来自 Athena 控制台的调用以及以事件方式对 Athena API 操作的代码调用。您可以确定向 Athena 发出的请求、从中发出请求的 IP 地址、发出请求的用户、发出请求的时间以及其他详细信息。有关更多信息，请参阅 使用 AWS CloudTrail 记录 Amazon Athena API 调用。

  您还可以使用 Athena 查询 CloudTrail 日志文件，不仅可用于 Athena，也可用于其他 AWS 服务。有关更多信息，请参阅 查询 AWS CloudTrail 日志。
- 通过 CloudTrail 和 Amazon QuickSight 监控 Athena 的使用情况 – Amazon QuickSight是一项完全托管式、基于云的业务情报服务，可让您创建组织可以从任何设备访问的交互式控制面板。有关使用 CloudTrail 和 Amazon QuickSight 监控 Athena 使用情况的解决方案的示例，请参阅 AWS 大数据博客文章 Realtor.com 如何使用 AWS CloudTrail 和 Amazon QuickSight 监控 Amazon Athena 使用情况。
- 将 EventBridge 与 Athena 一起使用 - Amazon EventBridge 提供近乎实时的系统事件流，这些系统事件描述 AWS 资源中的更改。EventBridge 注意到在他们发生时显示的操作更改、响应这些操作更改并在必要时采取纠正措施，方式是发送消息以响应环境、激活函数、进行更改并捕获状态信息。尽最大努力发出事件。有关更多信息，请参阅 Amazon EventBridge 用户指南中的 Amazon EventBridge 入门。
- 使用工作组分隔用户、团队、应用程序或工作负载，并设置查询限制和控制查询成本 – 您可以在 Amazon CloudWatch 中查看与查询相关的指标、通过配置扫描的数据量限制来控制查询成本、创建阈值以及在突破这些阈值时触发操作，例如 Amazon SNS 告警。有关简要过程，请参阅设置工作组。使用资源级别的 IAM 权限来控制对特定工作组的访问。有关更多信息，请参阅 使用工作组运行查询 和 使用 CloudWatch 指标和事件控制成本和监控查询。

主题

使用 AWS CloudTrail 记录 Amazon Athena API 调用

Athena 已与 AWS CloudTrail 集成，后者是一项提供 Athena 中由用户、角色或 AWS 服务 所采取操作的记录的服务。

CloudTrail 将 Athena 的API 调用作为事件捕获。捕获的调用包括来自 Athena 控制台的调用和对 Athena API 操作的代码调用。如果您创建跟踪，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶（包括 Athena 的事件）。如果您不配置跟踪，则仍可在 CloudTrail 控制台中的 事件历史记录 中查看最新事件。

使用 CloudTrail 收集的信息，您可以确定向 Athena 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息，请参阅《AWS CloudTrail 用户指南》。

您可以使用 Athena 从 Athena 本身和其他 AWS 服务 查询 CloudTrail 日志文件。有关更多信息，请参阅 查询 AWS CloudTrail 日志、Hive JSON SerDe，以及 AWS 大数据博客文章将 CTAS 语句与 Amazon Athena 结合使用以降低成本并提高性能，其中介绍了使用 CloudTrail 提供对 Athena 使用情况的洞察。

CloudTrail 中的 Athena 信息

在您创建 Amazon Web Services 账户时，将在该账户上启用 CloudTrail。当 Athena 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其他 AWS 服务事件一同保存在 Event history（事件历史记录）中。您可以在 Amazon Web Services 账户中查看、搜索和下载最新事件。有关更多信息，请参阅使用 CloudTrail 事件历史记录查看事件。

要持续记录 Amazon Web Services 账户中的事件（包括 Athena 的事件），请创建跟踪记录。通过跟踪记录，CloudTrail 可将日志文件传送至 Simple Storage Service (Amazon S3) 存储桶。预设情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有AWS 区域。此跟踪记录在 AWS 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Simple Storage Service（Amazon S3）存储桶。此外，您可以配置其他 AWS 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取行动。有关更多信息，请参阅下列内容：

- 创建跟踪记录概述
- CloudTrail supported services and integrations
- 为 CloudTrail 配置 Amazon SNS 通知
- 从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

所有 Athena 操作均由 CloudTrail 记录下来并记载到 Amazon Athena API 参考中。例如，对 StartQueryExecution 和 GetQueryResults 操作的调用会在 CloudTrail 日志文件中生成条目。

每个事件或日记账条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 AWS Identity and Access Management（IAM）用户凭证发出的。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务 发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

了解 Athena 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 存储桶。CloudTrail 日志文件包含一个或多个日记账条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

> **Note**
>
> 为防止意外泄露敏感信息，`StartQueryExecution` 和 `CreateNamedQuery` 日志中的 `queryString` 条目的值均为 `***OMITTED***`。这是设计使然。要访问实际查询字符串，可以使用 Athena GetQueryExecution API 并传入 CloudTrail 日志中的 `responseElements.queryExecutionId` 值。

以下示例展示了 CloudTrail 日志条目：

- StartQueryExecution (成功)
- StartQueryExecution (失败)
- CreateNamedQuery

StartQueryExecution（成功）

```
{
 "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（失败）

```
{
 "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"
},
```

Amazon Athena 的合规性验证

作为多个 AWS 合规性计划的一部分，第三方审计员将评估 Amazon Athena 的安全性和合规性。这些合规性计划包括 SOC、PCI、FedRAMP 等。

有关特定合规性计划范围内的 AWS 服务 的列表，请参阅合规性计划范围内的 AWS 服务。有关一般信息，请参阅 AWS 合规性计划。

您可以使用 AWS Artifact 下载第三方审计报告。有关更多信息，请参阅在 AWS Artifact 中下载报告。

您使用 Athena 的合规性责任取决于您数据的敏感度、贵公司的合规性目标以及适用的法律法规。AWS 提供以下资源来帮助满足合规性：

- 安全性与合规性快速入门指南 – 这些部署指南讨论了架构注意事项，并提供了在 AWS 上部署基于安全性和合规性的基准环境的步骤。
- Amazon Web Services 上的 HIPAA 安全性和合规性架构设计– 此白皮书介绍了公司如何使用 AWS 创建符合 HIPAA 标准的应用程序。
- AWS 合规性资源 – 此业务手册和指南集合可能适用于您的行业和位置。
- AWS Config：此 AWS 服务 评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- AWS Security Hub：此 AWS 服务 提供了 AWS 中安全状态的全面视图，可帮助您检查是否符合安全行业标准和最佳实践规范。

Athena 中的故障恢复能力

AWS全球基础设施围绕AWS 区域和可用区构建。AWS 区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域 和可用区的更多信息，请参阅 AWS 全球基础设施。

除了 AWS 全球基础设施之外，Athena 还提供了多种功能，以帮助支持您的数据弹性和备份需求。

Athena 是无服务器服务，因此无需设置或管理基础设施。Athena 具备高可用性，并使用多个可用区中的计算资源来运行查询，如果特定可用区无法访问，则会自动适当地路由查询。Athena 使用 Amazon S3 作为其底层数据存储，使您的数据具备高可用性和持久性。Amazon S3 提供持久性基础设施来存储重要数据，并且旨在为对象提供 99.999999999% 的持久性。您的数据以冗余方式存储在多个设施中，以及各个设施内的多个设备上。

Athena 中的基础设施安全性

作为一项托管式服务，Amazon Athena 受 AWS 全球网络安全保护。有关 AWS 安全服务以及 AWS 如何保护基础设施的信息，请参阅 AWS 云安全。要按照基础设施安全最佳实践设计您的 AWS 环境，请参阅《安全性支柱 AWS Well-Architected Framework》中的 基础设施保护。

您可以使用 AWS 发布的 API 调用通过网络访问 Athena。客户端必须支持以下内容：

- 传输层安全性协议 (TLS) 我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密 (PFS) 的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service（AWS STS）生成临时安全凭证来对请求进行签名。

使用 IAM policy 来限制对 Athena 操作的访问。每当您使用 IAM policy 时，请确保遵循 IAM 最佳实践。有关更多信息，请参阅《IAM 用户指南》中的 IAM 安全最佳实践。

Athena 托管策略易于使用，并且随着服务的发展而自动用必需操作进行更新。利用客户托管策略和内联策略，您可以通过在策略中指定更精细的 Athena 操作来微调策略。授予对数据的 Amazon S3 位置的适当访问权限。有关详细信息和如何授予 Amazon S3 访问权限的方案，请参阅 Amazon Simple Storage Service 开发人员指南 中的 示例演练：管理访问权限。有关更多信息以及允许哪些 Amazon S3 操作的示例，请参阅跨账户访问中的示例存储桶策略。

主题

- 使用接口 VPC 终端节点连接到 Amazon Athena

使用接口 VPC 终端节点连接到 Amazon Athena

您可以通过使用 接口 VPC 端点（AWS PrivateLink）和虚拟私有云（VPC）中的 AWS Glue VPC 端点来提高 VPC 的安全性。接口 VPC 端点允许您控制可以从 VPC 内部连接到的目的地，从而提高安全性。每个 VPC 端点都由您的 VPC 子网中一个或多个使用私有 IP 地址的弹性网络接口（ENI）代表。

接口 VPC 终端节点将您的 VPC 直接连接到 Athena，而无需互联网网关、NAT 设备、VPN 连接或 AWS Direct Connect 连接。VPC 中的实例不需要公有 IP 地址便可与 Athena API 进行通信。

要通过您的 VPC 使用 Athena，您必须从位于 VPC 中的实例进行连接，或者使用 Amazon 虚拟专用网络 (VPN) 或 AWS Direct Connect 将您的专用网络连接到 VPC。有关 Amazon VPN 的信息，请参阅《Amazon Virtual Private Cloud 用户指南》中的 VPN 连接。有关 AWS Direct Connect 连接的更多信息，请参阅在《AWS Direct Connect 用户指南》中的创建连接。

在 Amazon VPC 和 Athena 均可用的所有 AWS 区域 中，Athena 支持 VPC 终端节点。

您可以创建接口 VPC 终端节点以使用 AWS Management Console 控制台或 AWS Command Line Interface (AWS CLI) 命令连接到 Athena。有关更多信息，请参阅创建接口端点。

在创建接口 VPC 终端节点后，如果您为终端节点启用私有 DNS 主机名，则默认 Athena 端点 (https://athena.*Region*.amazonaws.com) 将解析为您的 VPC 终端节点。

如果您尚未启用私有 DNS 主机名，则 Amazon VPC 将提供一个您可以使用的 DNS 端点名称，格式如下：

```
VPC_Endpoint_ID.athena.Region.vpce.amazonaws.com
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的接口 VPC 端点 (AWS PrivateLink)。

Athena 支持调用您的 VPC 中的所有 API 操作。

为 Athena 创建 VPC 终端节点策略

您可以为 Athena 的 Amazon VPC 端点创建一个策略，指定如下限制：

- 主体 – 可执行操作的主体。
- 操作 – 可执行的操作。
- 资源 – 可对其执行操作的资源。
- 仅限可信身份 - 使用 `aws:PrincipalOrgId` 条件将访问权限限制为仅限 AWS 组织中的凭证。这可以帮助防止主体非预期访问。
- 仅限可信资源 - 使用 `aws:ResourceOrgId` 条件可防止访问非预期资源。
- 仅限可信身份和资源 - 为 VPC 端点创建组合策略，以帮助防止访问非预期主体和资源。

有关更多信息，请参阅《Amazon VPC 用户指南》中的使用 VPC 端点控制对服务的访问以及 AWS 白皮书在 AWS 上构建数据周界中的附录 2 - VPC 端点策略示例。

Example – VPC 端点策略

以下示例允许按组织身份向组织资源发出请求，并允许 AWS 服务主体发出请求。

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

每当您使用 IAM policy 时，请确保遵循 IAM 最佳实践。有关更多信息，请参阅《IAM 用户指南》中的 IAM 安全最佳实践。

共享子网

您无法在与您共享的子网中创建、描述、修改或删除 VPC 端点。但是，您可以在与您共享的子网中使用 VPC 端点。有关 VPC 共享的信息，请参阅《Amazon VPC 用户指南》中的与其他账户共享 VPC。

Athena 中的配置和漏洞分析

Athena 采用无服务器架构，因此没有基础设施要设置或管理。AWS 负责处理来宾操作系统 (OS) 和数据库补丁、防火墙配置和灾难恢复等基本安全工作。这些流程已通过相应第三方审核和认证。有关更多详细信息，请参阅以下 AWS 资源：

- 责任共担模式
- 安全性、身份和合规性最佳实践

使用 Athena 查询向 AWS Lake Formation 注册的数据

AWS Lake Formation 允许您在使用 Athena 查询来读取存储在 Amazon S3 中的数据时定义和实施数据库、表和列级访问策略。Lake Formation 为存储在 Amazon S3 中的数据提供授权和监管层。您可以使用 Lake Formation 中的权限层次结构来授予或撤消读取数据目录对象（如数据库、表和列）的权限。Lake Formation 简化了权限管理，并允许您为数据实现精细的访问控制 (FGAC)。

您可以使用 Athena 同时查询向 Lake Formation 注册的数据和未向 Lake Formation 注册的数据。

使用 Athena 从向 Lake Formation 注册的 Amazon S3 位置查询源数据时，将适用 Lake Formation 权限。Lake Formation 权限也适用于您创建指向已注册的 Amazon S3 数据位置的数据库和表的操作。要将 Athena 与使用 Lake Formation 注册的数据结合使用，必须将 Athena 配置为使用 AWS Glue Data Catalog。

Lake Formation 权限不适用于将对象写入 Amazon S3 时，也不适用于在查询存储在 Amazon S3 的数据或未向 Lake Formation 注册的元数据时。对于 Amazon S3 中的源数据和未向 Lake Formation 注册的元数据，访问权限由 Amazon S3 和 AWS Glue 操作的 IAM 权限策略决定。Amazon S3 中的 Athena 查询结果位置无法向 Lake Formation 注册，而 Amazon S3 的 IAM 权限策略将控制访问权限。此外，Lake Formation 权限不适用于 Athena 查询历史记录。您可以使用 Athena 工作组来控制对查询历史记录的访问。

有关 Lake Formation 的更多信息，请参阅 Lake Formation 常见问题和《AWS Lake Formation 开发人员指南》。

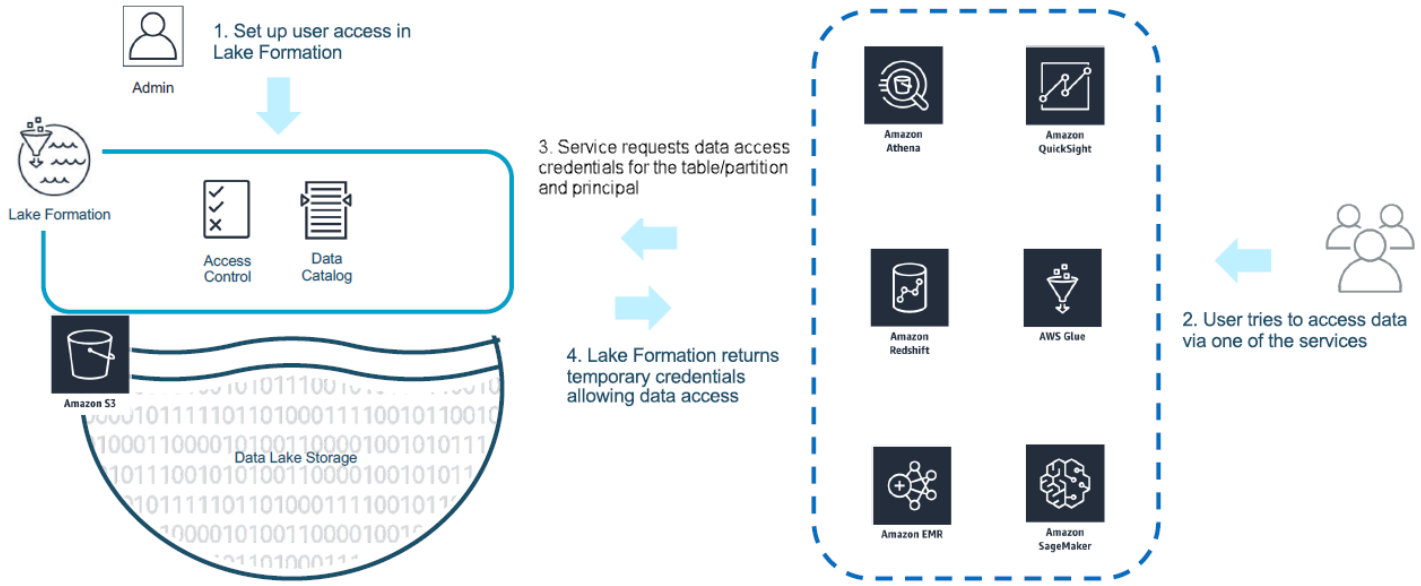
主题

Athena 如何访问已向 Lake Formation 注册的数据

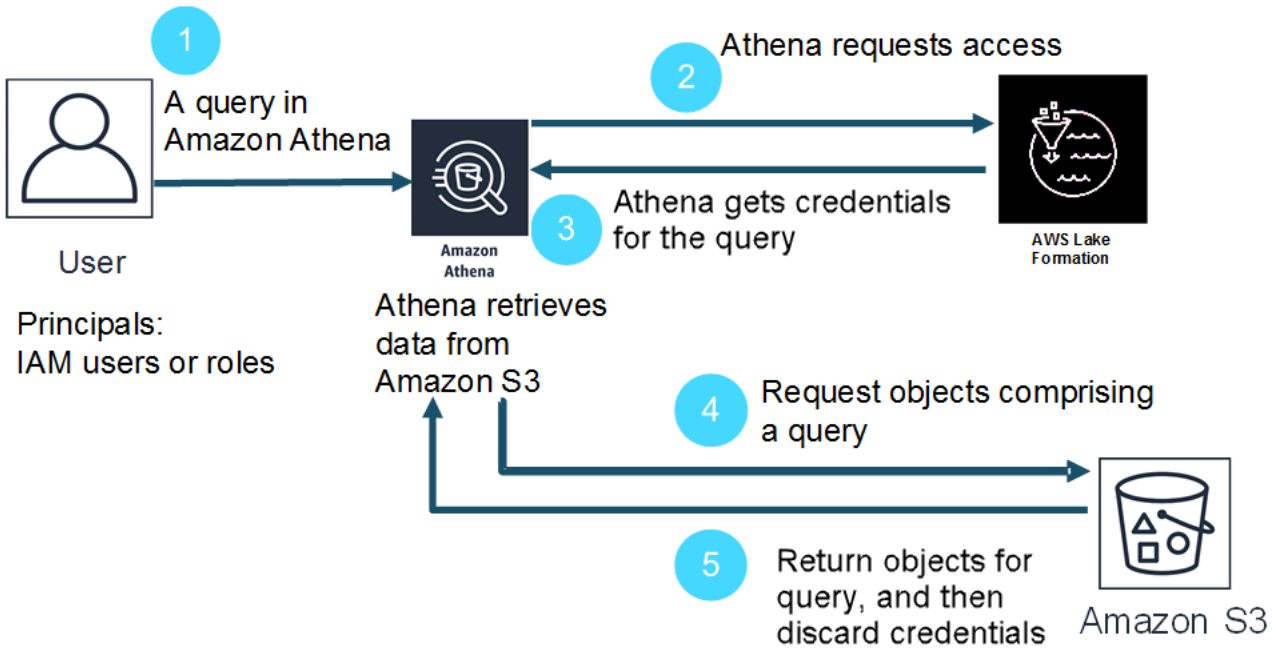
本部分中描述的访问工作流仅当对已向 Lake Formation 注册的 Amazon S3 位置和元数据对象运行 Athena 查询时才适用。有关更多信息，请参阅《AWS Lake Formation 开发人员指南》中的 注册数据湖。除了注册数据之外，Lake Formation 管理员还应用以下 Lake Formation 权限：这些权限授予或撤消对 Data Catalog 中的元数据和 Amazon S3 中的数据位置的访问权限。有关更多信息，请参阅《AWS Lake Formation 开发人员指南》中的 元数据和数据的安全性和访问控制。

当 Athena 委托人（用户、组或角色）每次对使用 Lake Formation 注册的数据运行查询时，Lake Formation 都会验证委托人是否对数据库、表和 Amazon S3 位置具有适合的 Lake Formation 权限来进行相应的查询。如果委托人具有访问权限，Lake Formation 会将临时凭据发送给 Athena，然后查询运行。

下图阐明了上述流程。



下图显示凭证发送过程如何在 Athena 中在逐个查询的基础上，对具有在 Lake Formation 中注册的 Amazon S3 位置的表进行假设的 SELECT 查询：



1. 委托人在 Athena 中运行 SELECT 查询。
2. Athena 分析查询并检查 Lake Formation 权限，以查看该委托人是否已被授予对该表和表列的访问权限。
3. 如果委托人具有访问权限，Athena 将从 Lake Formation 请求凭证。如果委托人没有访问权限，则 Athena 发出拒绝访问错误。
4. Lake Formation 将向 Athena 颁发凭据，以便在从 Amazon S3 读取数据时使用，同时还将发送允许的列列表。
5. Athena 将使用 Lake Formation 临时凭证从 Amazon S3 查询数据。在查询完成后，Athena 将丢弃凭证。

使用 Athena 查询向 Lake Formation 注册的数据时的注意事项和限制

使用 Athena 查询在 Lake Formation 中注册的数据时，请考虑以下因素。有关更多信息，请参阅《AWS Lake Formation 开发人员指南》中的 AWS Lake Formation 的已知问题。

注意事项和限制

- 在使用 Avro 和自定义 SerDe 的某些情况下，未经授权的用户可以看到列元数据
- 使用 Lake Formation 权限查看视图
- Lake Formation 精细访问控制和 Athena 工作组
- Amazon S3 中的 Athena 查询结果位置未向 Lake Formation 注册
- 使用 Athena 工作组限制对查询历史记录的访问
- 跨账户数据目录访问
- 向 Lake Formation 注册的 CSE-KMS 加密的 Amazon S3 位置
- 向 Lake Formation 注册的分区数据位置必须在表子目录中
- Create Table As Select (CTAS) 查询需要 Amazon S3 写入权限
- 默认数据库需要 DESCRIBE 权限

在使用 Avro 和自定义 SerDe 的某些情况下，未经授权的用户可以看到列元数据

Lake Formation 列级授权可防止用户访问自己没有 Lake Formation 权限的列中的数据。但是，在某些情况下，用户可以访问描述表中所有列的元数据，包括他们对其中的数据没有权限的列。

当列元数据使用 Avro 存储格式或使用自定义序列化程序/反序列化程序（SerDe）存储在表的表属性中（其中表架构在表属性中与 SerDe 定义一起定义）时，会发生这种情况。在将 Athena 与 Lake

Formation 一起使用时，我们建议您查看向 Lake Formation 注册的表属性的内容，并在可能的情况下限制存储在表属性中的信息，以防止任何敏感元数据对用户可见。

使用 Lake Formation 权限查看视图

对于向 Lake Formation 注册的数据，Athena 用户只有在对 VIEW 所基于的表、列和源 Amazon S3 数据位置具有 Lake Formation 权限时，才能创建 VIEW。在 Athena 中创建 VIEW 后，Lake Formation 权限可以应用于 VIEW。列级权限不可用于 VIEW。对 VIEW 具有 Lake Formation 权限但对视图所基于的表和列没有权限的用户无法使用 VIEW 查询数据。但是，具有此权限组合的用户可以使用语句（如 DESCRIBE VIEW、SHOW CREATE VIEW 和 SHOW COLUMNS）查看 VIEW 元数据。因此，请确保对于每个 VIEW 的 Lake Formation 权限与基础表权限一致。表中定义的单元格筛选条件不适用于该表的 VIEW。资源链接名称必须与原始账户中的资源名称相同。在跨账户设置中使用视图时还存在其他限制。有关跨账户设置共享视图的权限的更多信息，请参阅 跨账户数据目录访问。

Lake Formation 精细访问控制和 Athena 工作组

同一 Athena 工作组中的用户可以查看由 Lake Formation 精细访问控制配置为可供工作组访问的数据。有关在 Lake Formation 中使用精细访问控制的更多信息，请参阅 AWS 大数据博客中的使用 AWS Lake Formation 管理精细访问控制。

Amazon S3 中的 Athena 查询结果位置未向 Lake Formation 注册

Amazon S3 中针对 Athena 的查询结果位置无法向 Lake Formation 注册。Lake Formation 权限不限制对这些位置的访问。除非您限制访问权限，否则，如果 Athena 用户拥有数据的 Lake Formation 权限，则可以访问查询结果文件和元数据。为避免这种情况，我们建议您使用工作组指定查询结果的位置，并将工作组成员身份与 Lake Formation 权限对齐。然后，您可以使用 IAM 权限策略限制对查询结果位置的访问。有关查询结果的更多信息，请参阅 使用查询结果、最近查询和输出文件。

使用 Athena 工作组限制对查询历史记录的访问

Athena 查询历史记录显示已保存查询和完整查询字符串的列表。除非您使用工作组来分离查询历史记录的访问权限，否则即使 Athena 用户没有获得授权查询 Lake Formation 中的数据，仍可以查看对该数据运行的查询字符串，包括列名、选择条件等。我们建议您使用工作组来分隔查询历史记录，并将 Athena 工作组成员资格与 Lake Formation 权限对齐以限制访问。有关更多信息，请参阅 使用工作组控制查询访问和成本。

跨账户数据目录访问

要访问其他账户中的数据目录，您可以使用 Athena 的跨账户 AWS Glue 功能或在 Lake Formation 中设置跨账户访问。

Athena 跨账户数据目录访问

您可以使用 Athena 的跨账户 AWS Glue 目录功能在您的账户中注册目录。此功能仅在 Athena 引擎版本 2 及更高版本中可用，并且仅限于在账户之间使用同一区域时。有关更多信息，请参阅 从另一个账户注册 AWS Glue Data Catalog。

如果要共享的数据目录在 AWS Glue 中配置了资源策略，必须更新该策略才能允许对 AWS Resource Access Manager 的访问，并向账户 B 授予使用账户 A 的数据目录的权限，如以下示例所示。

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

有关更多信息，请参阅 授予 AWS Glue 数据目录跨账户访问权限。

在 Lake Formation 中设置跨账户访问权限

AWS Lake Formation 允许您使用单个账户管理中央 Data Catalog。您可以使用此功能来对 Data Catalog 元数据和底层数据实施跨账户访问。例如，拥有者账户可以授予另一个（收件人）账户对于表的 SELECT 权限。

要在 Athena 查询编辑器中显示共享数据库或表，您可以在 Lake Formation 中创建资源链接来访问共享数据库或表。当 Lake Formation 中的收件人账户查询拥有者的表格时，CloudTrail 会将数据访问事件添加到收件人账户和拥有者账户的日志中。

对于共享视图，请记住以下几点：

- 查询在目标资源链接上运行，而不是在源表或视图上运行，然后系统会将输出共享到目标账户。
- 仅共享视图是不够的。创建视图涉及的所有表都必须是跨账户共享的一部分。
- 基于共享资源创建的资源链接的名称必须与拥有者账户中资源的名称相匹配。如果名称不匹配，系统会显示与以下内容类似的错误消息：Failed analyzing stored view 'awsdatacatalog.*my-lf-resource-link*.*my-lf-view*': line 3:3: Schema *schema_name* does not exist（分析存储的视图“awsdatacatalog.my-lf-resource-link.my-lf-view”时失败：line 3:3: 架构 schema_name 不存在）。

有关 Lake Formation 中跨账户访问的更多信息，请参阅《AWS Lake Formation 开发人员指南》中的以下资源：

跨账户访问

资源链接在 Lake Formation 中的工作原理

跨账户 CloudTrail 日志记录

向 Lake Formation 注册的 CSE-KMS 加密的 Amazon S3 位置

Athena 无法查询 Apache Iceberg 等具有以下特征的 Open Table Format（OTF）表：

- 这些表以向 Lake Formation 注册的 Amazon S3 数据位置为基础。
- 使用客户端加密（CSE）对 Amazon S3 中的对象进行加密。
- 使用 AWS KMS 客户托管密钥 (CSE_KMS) 进行加密。

要查询使用 CSE_KMS 密钥加密的非 OTF 表，请将以下块添加到用于 CSE 加密的 AWS KMS 密钥策略中。*<KMS_KEY_ARN>* 是加密数据的 AWS KMS 密钥的 ARN。*<IAM-ROLE-ARN>* 是在 Lake Formation 中注册 Amazon S3 位置的 IAM 角色的 ARN。

```
{
    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal": {
        "AWS": "*"
    },
    "Action": "kms:Decrypt",
    "Resource": "<KMS-KEY-ARN>",
    "Condition": {
        "ArnLike": {
            "aws:PrincipalArn": "<IAM-ROLE-ARN>"
        }
    }
}
```

向 Lake Formation 注册的分区数据位置必须在表子目录中

向 Lake Formation 注册的分区表必须在作为 Amazon S3 中该表的子目录的目录中具有分区数据。例如，可以向 Lake Formation 注册并使用 Athena 查询具有位置 `s3://DOC-EXAMPLE-BUCKET/mytable` 和分区 `s3://DOC-EXAMPLE-BUCKET/mytable/dt=2019-07-11`、`s3://DOC-EXAMPLE-BUCKET/mytable/dt=2019-07-12` 等的表。另一方面，无法向 Lake Formation 注册具有位置 `s3://DOC-EXAMPLE-BUCKET/mytable` 和分区（位于`s3://DOC-EXAMPLE-BUCKET/dt=2019-07-11`、`s3://DOC-EXAMPLE-BUCKET/dt=2019-07-12` 等等中）的表。因为这样的分区不是 `s3://DOC-EXAMPLE-BUCKET/mytable` 的子目录，它们也无法从 Athena 读取。

Create Table As Select (CTAS) 查询需要 Amazon S3 写入权限

Create Table As Statements (CTAS) 需要对表的 Amazon S3 位置进行写入访问。要对向 Lake Formation 注册的数据运行 CTAS 查询，Athena 用户除了具有相应的 Lake Formation 权限以读取数据位置以外，还必须具有写入表 Amazon S3 位置的 IAM 权限。有关更多信息，请参阅 从查询结果创建表（CTAS）。

默认数据库需要 DESCRIBE 权限

对于 `default` 数据库，Lake Formation DESCRIBE 权限是必需的。以下示例 AWS CLI 命令对 AWS 账户 `111122223333` 中的用户 `datalake_user1` 授予对 `default` 数据库的 DESCRIBE 权限。

```
aws lakeformation grant-permissions --principal
 DataLakePrincipalIdentifier=arn:aws:iam::111122223333:user/datalake_user1 --
permissions "DESCRIBE" --resource '{ "Database": {"Name":"default"}}
```

有关更多信息，请参阅《AWS Lake Formation 开发人员指南》中的 Lake Formation 权限参考。

管理 Lake Formation 和 Athena 用户权限

Lake Formation 发送凭证以查询向 Lake Formation 注册的 Amazon S3 数据存储。如果您以前使用过 IAM policy 来允许或拒绝读取 Amazon S3 中的数据位置的权限，则可以改用 Lake Formation 权限。但是，其他 IAM 权限仍然是必需的。

每当您使用 IAM policy 时，请确保遵循 IAM 最佳实践。有关更多信息，请参阅《IAM 用户指南》中的 IAM 安全最佳实践。

以下各节总结了使用 Athena 查询在 Lake Formation 中注册的数据所需的权限。有关更多信息，请参阅《AWS Lake Formation 开发人员指南》中的 AWS Lake Formation 的安全性。

权限摘要

- Lake Formation 和 Athena 基于身份的权限
- Amazon S3 对 Athena 查询结果位置的权限
- 针对查询历史记录的 Athena 工作组成员身份
- Lake Formation 对数据的权限
- 写入 Amazon S3 位置的 IAM 权限
- 针对加密数据、元数据和 Athena 查询结果的权限
- 外部账户中针对 Amazon S3 存储桶的基于资源的权限（可选）

Lake Formation 和 Athena 基于身份的权限

使用 Athena 查询向 Lake Formation 注册的数据的任何人都必须有一个 IAM 权限策略，此权限允许执行 `lakeformation:GetDataAccess` 操作。AWS 托管策略：AmazonAthenaFullAccess 允许此操作。如果您使用内联策略，请务必更新权限策略以允许此操作。

在 Lake Formation 中，数据湖管理员有权创建元数据对象（如数据库和表），向其他用户授予 Lake Formation 权限以及注册新的 Amazon S3 位置。要注册新位置，需要具有 Lake Formation 的服务相关角色的权限。有关更多信息，请参阅《AWS Lake Formation 开发人员指南》中的 创建数据湖管理员和 Lake Formation 的服务相关角色权限。

Lake Formation 用户可以使用 Athena，通过数据湖管理员授予他们的 Lake Formation 权限，查询数据库、表、表列和基础 Amazon S3 数据存储。用户不能创建数据库或表，也不能向 Lake Formation 注册新的 Amazon S3 位置。有关更多信息，请参阅《AWS Lake Formation 开发人员指南》中的 创建数据湖用户。

在 Athena 中，基于身份的权限策略（包括 Athena 工作组的权限策略）仍然控制对 Amazon Web Services 账户用户的 Athena 操作的访问。此外，可以通过 Athena 驱动程序提供的基于 SAML 的身份验证来提供联合访问。有关更多信息，请参阅 使用工作组控制查询访问和成本、用于访问工作组的 IAM policy 和 启用对 Athena API 的联合身份访问。

有关更多信息，请参阅《AWS Lake Formation 开发人员指南》中的 授予 Lake Formation 权限。

Amazon S3 对 Athena 查询结果位置的权限

Amazon S3 中针对 Athena 的查询结果位置无法向 Lake Formation 注册。Lake Formation 权限不限制对这些位置的访问。除非您限制访问权限，否则，如果 Athena 用户拥有数据的 Lake Formation 权限，则可以访问查询结果文件和元数据。为避免这种情况，我们建议您使用工作组指定查询结果的位置，并将工作组成员身份与 Lake Formation 权限对齐。然后，您可以使用 IAM 权限策略限制对查询结果位置的访问。有关查询结果的更多信息，请参阅 使用查询结果、最近查询和输出文件。

针对查询历史记录的 Athena 工作组成员身份

Athena 查询历史记录显示已保存查询和完整查询字符串的列表。除非您使用工作组来分离查询历史记录的访问权限，否则即使 Athena 用户没有获得授权查询 Lake Formation 中的数据，仍可以查看对该数据运行的查询字符串，包括列名、选择条件等。我们建议您使用工作组来分隔查询历史记录，并将 Athena 工作组成员资格与 Lake Formation 权限对齐以限制访问。有关更多信息，请参阅 使用工作组控制查询访问和成本。

Lake Formation 对数据的权限

除了使用 Lake Formation 的基准权限之外，Athena 用户还必须具有访问他们查询的资源的 Lake Formation 权限。这些权限由 Lake Formation 管理员授予和管理。有关更多信息，请参阅《AWS Lake Formation 开发人员指南》中的 元数据和数据的安全性和访问控制。

写入 Amazon S3 位置的 IAM 权限

Amazon S3 的 Lake Formation 权限不包括写入 Amazon S3 的权限。Create Table As Statements (CTAS) 需要对表的 Amazon S3 位置进行写入访问。要对向 Lake Formation 注册的数据运行 CTAS 查询，Athena 用户除了具有相应的 Lake Formation 权限以读取数据位置以外，还必须具有写入表 Amazon S3 位置的 IAM 权限。有关更多信息，请参阅 从查询结果创建表（CTAS）。

针对加密数据、元数据和 Athena 查询结果的权限

可以对向 Lake Formation 注册的 Amazon S3 中的基础源数据和 Data Catalog 中的元数据进行加密。在使用 Athena 查询向 Lake Formation 注册的数据时，Athena 处理查询结果加密的方式没有变化。有关更多信息，请参阅 加密在 Amazon S3 中存储的 Athena 查询结果。

- 加密源数据 – 支持对 Amazon S3 数据位置源数据进行加密。查询已向 Lake Formation 注册的加密 Amazon S3 位置的 Athena 用户需要加密和解密数据的权限。有关要求的更多信息，请参阅 支持的 Amazon S3 加密选项 和 Amazon S3 中加密数据的权限。
- 加密元数据 – 支持对 Data Catalog 中的元数据进行加密。对于使用 Athena 的委托人，基于身份的策略必须允许对用于加密元数据的密钥执行 `"kms:GenerateDataKey"`、`"kms:Decrypt"` 和 `"kms:Encrypt"` 操作。有关更多信息，请参阅《AWS Glue 开发人员指南》中的加密数据目录和从 Athena 访问 AWS Glue Data Catalog 中的加密元数据。

外部账户中针对 Amazon S3 存储桶的基于资源的权限（可选）

要查询其他账户中的 Amazon S3 数据位置，基于资源的 IAM policy（存储桶策略）必须允许访问该位置。有关更多信息，请参阅 Athena 中对 Amazon S3 存储桶的跨账户访问。

有关访问其他账户中的 Data Catalog 的信息，请参阅 Athena 跨账户数据目录访问。

将 Lake Formation 权限应用于现有数据库和表

如果您是首次使用 Athena 并且使用 Lake Formation 配置查询数据的访问权限，则无需配置 IAM policy 以使用户可以读取 Amazon S3 数据和创建元数据。您可以使用 Lake Formation 管理权限。

不需要向 Lake Formation 注册数据和更新 IAM 权限策略。如果数据未向 Lake Formation 注册，则在 Amazon S3 中具有相应权限的 Athena 用户和 AWS Glue（如果适用）可以继续查询未向 Lake Formation 注册的数据。

如果您的现有 Athena 用户查询未向 Lake Formation 注册的数据，则您可以更新 Amazon S3以及 AWS Glue Data Catalog（如果适用）的 IAM 权限，以便您可以使用 Lake Formation 权限集中管理用户访问权限。要获得读取 Amazon S3 数据位置的权限，您可以更新基于资源和基于身份的策略以修改 Amazon S3 权限。要访问元数据，如果您使用 AWS Glue 配置了用于精细访问控制的资源级策略，则可以改用 Lake Formation 权限来管理访问。

有关更多信息，请参阅《AWS Lake Formation 开发人员指南》中的 针对 AWS Glue Data Catalog 中数据库和表的精细访问权限 和 升级 AWS Lake Formation 模型的 AWS Glue 数据权限。

将 Lake Formation 和 Athena JDBC 和 ODBC 驱动程序用于对 Athena 进行联合访问

Athena JDBC 和 ODBC 驱动程序使用 Okta 和 Microsoft Active Directory 联合访问服务 (AD FS) 身份提供程序支持对 Athena 进行基于 SAML 2.0 的联合访问。通过将 Amazon Athena 与 AWS Lake Formation 整合，您可以使用公司凭证对 Athena 启用基于 SAML 的身份验证。借助 Lake Formation

和 AWS Identity and Access Management (IAM)，您可以保持对 SAML 用户可用数据的精细列级访问控制。使用 Athena JDBC 和 ODBC 驱动程序，联合访问可用于工具或编程访问。

要使用 Athena 来访问由 Lake Formation 控制的数据源，您需要通过配置身份提供程序 (IdP) 和 AWS Identity and Access Management (IAM) 角色来启用基于 SAML 2.0 的联合访问。有关详细步骤，请参阅教程：使用 Lake Formation 和 JDBC 配置 Okta 用户对 Athena 的联合访问。

先决条件

要使用 Amazon Athena 和 Lake Formation 进行联合访问，您必须满足以下要求：

- 使用现有基于 SAML 的身份提供程序（例如 Okta 或 Microsoft Active Directory Federation Services (AD FS)）管理您的公司身份。
- 您可以将 AWS Glue Data Catalog 用作元数据存储。
- 在 Lake Formation 中定义和管理权限以访问 AWS Glue Data Catalog 中的数据库、表和列。有关更多信息，请参见AWS Lake Formation 开发人员指南。
- 您可以使用版本 2.0.14 或更高版本的 Athena JDBC 驱动程序或版本 1.1.3 或更高版本的 Athena ODBC 驱动程序。

注意事项和限制

使用 Athena JDBC 或 ODBC 驱动程序和 Lake Formation 配置对 Athena 的联合访问时，请记住以下几点：

- 目前，Athena JDBC 驱动程序和 ODBC 驱动程序支持 Okta、Microsoft Active Directory Federation Services（AD FS）和 Azure AD 身份提供商。尽管 Athena JDBC 驱动程序具有可扩展为使用其他身份提供程序的通用 SAML 类，但对允许其他身份提供程序 (IdPs) 与 Athena 一起使用的自定义扩展的支持可能会受到限制。
- 使用 JDBC 和 ODBC 驱动程序的联合访问与 IAM Identity Center 可信身份传播功能不兼容。
- 目前，您无法使用 Athena 控制台配置对 IdP 和 SAML 与 Athena 一起使用的支持。要配置此支持，请使用第三方身份提供程序、Lake Formation 和 IAM 管理控制台以及 JDBC 或 ODBC 驱动程序客户端。
- 您应该了解 SAML 2.0 规范以及它如何与身份提供程序相互合作，然后才嗯呢该配置您的身份提供程序和 SAML 用于 Lake Formation 和 Athena。
- SAML 提供商和 Athena JDBC 和 ODBC 驱动程序由第三方提供，因此通过 AWS 对与其使用有关问题的支持可能是有限的。

主题

教程：使用 Lake Formation 和 JDBC 配置 Okta 用户对 Athena 的联合访问

本教程介绍如何配置 Okta、AWS Lake Formation、AWS Identity and Access Management 权限和 Athena JDBC 驱动程序，以启用基于 SAML 的 Athena 联合使用。Lake Formation 为基于 SAML 的用户提供了对 Athena 中可用数据的细粒度访问控制。为设置此配置，本教程使用 Okta 开发人员控制台、AWS IAM 和 Lake Formation 控制台以及 SQL Workbench/J 工具。

先决条件

本教程假定您已执行以下操作：

- 已创建 Amazon Web Services 账户。要创建帐户，请访问 Amazon Web Services 主页。
- 在 Amazon S3 中为 Athena 设置查询结果位置。
- 向 Lake Formation 注册了 Amazon S3 数据存储桶位置。
- 在 AWS Glue Data Catalog 中定义了指向您在 Amazon S3 中数据的数据库和表。
  - 如果尚未定义表，则对您要访问的数据运行 AWS Glue 爬网程序或者使用 Athena 定义数据库和一个或多个表。
  - 本教程使用基于 AWS 开放数据存储库中的纽约出租车旅行数据集。本教程使用数据库名称 `tripdb` 和表名称 `nyctaxi`。

教程步骤

步骤 1：创建 Okta 账户

本教程使用 Okta 作为基于 SAML 的身份提供程序。如果您还没有 Okta 账户，您可以创建一个免费的账户。需要 Okta 帐户，以便您可以创建 Okta 应用程序进行 SAML 身份验证。

创建 Okta 账户

1. 要使用 Okta，请导航到 Okta 开发人员注册页并创建一个免费的 Okta 试用账户。开发人员版服务是免费的，不超过 Okta 在 developer.okta.com/pricing 总指定的限制。
2. 当您收到激活电子邮件时，请激活您的账户。

   Okta 域名将被分配给您。保存域名以供参考。稍后，您可以在连接到 Athena 的 JDBC 字符串中使用域名 (*<okta-idp-domain>*)。

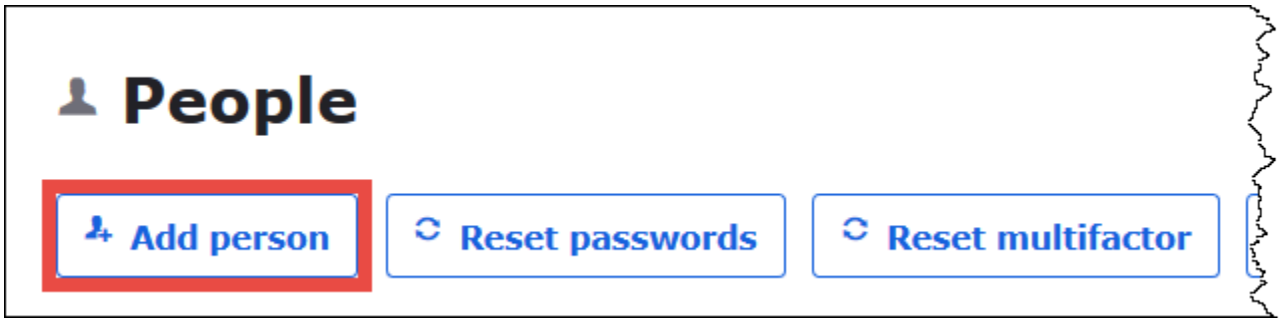
步骤 2：向 Okta 添加用户和组

在此步骤中，您使用 Okta 控制台执行以下任务：

- 创建两个 Okta 用户。
- 创建两个 Okta 组。
- 向每个 Okta 组添加一个 Okta 用户。

要向 Okta 添加用户

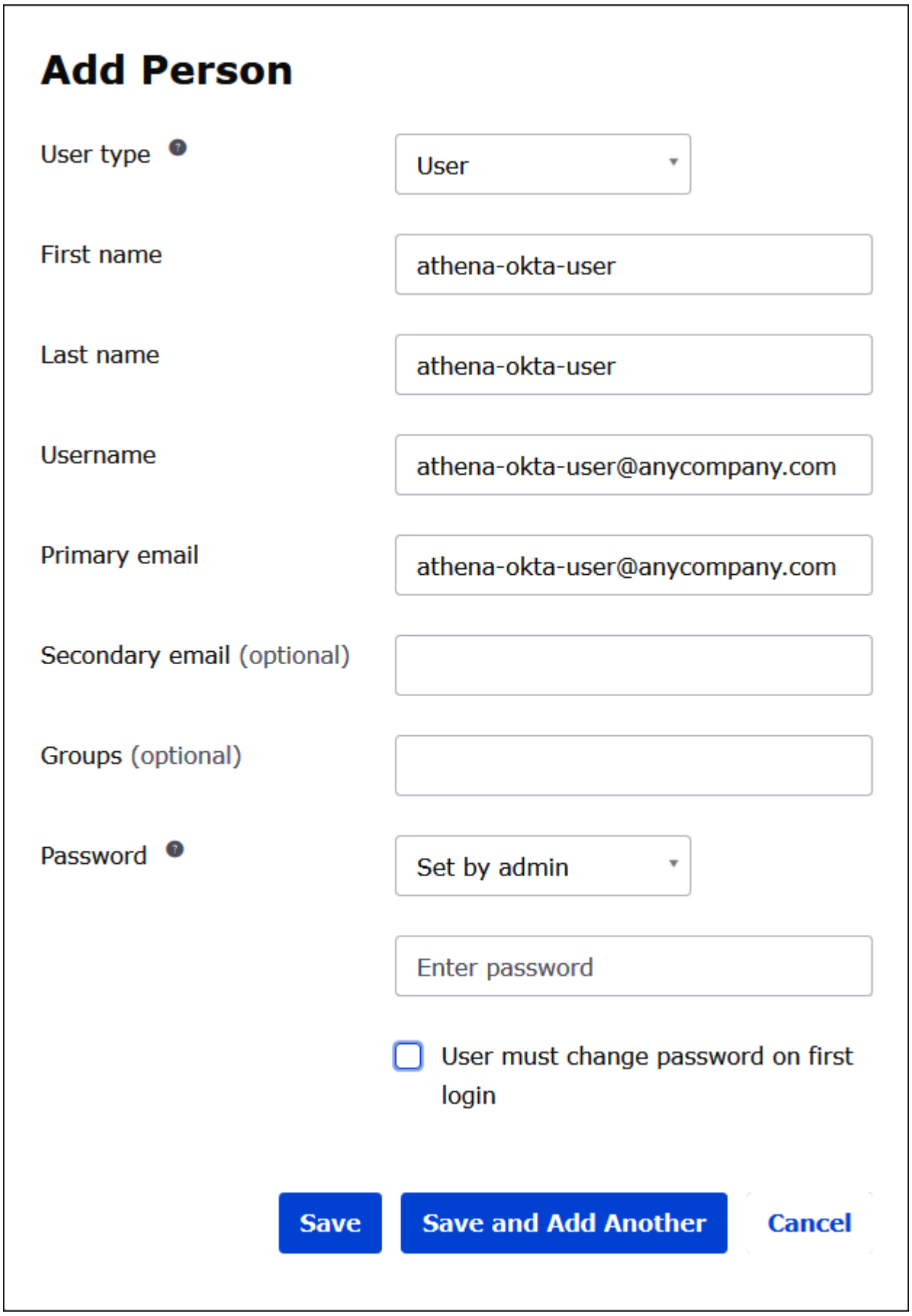
1. 激活 Okta 账户后，以管理用户身份登录到指定的 Okta 域。
2. 在左侧导航窗格中，选择 Directory（目录），然后选择 People（人员）。
3. 选择 Add Person（添加人员）添加将通过 JDBC 驱动程序访问 Athena 的新用户。



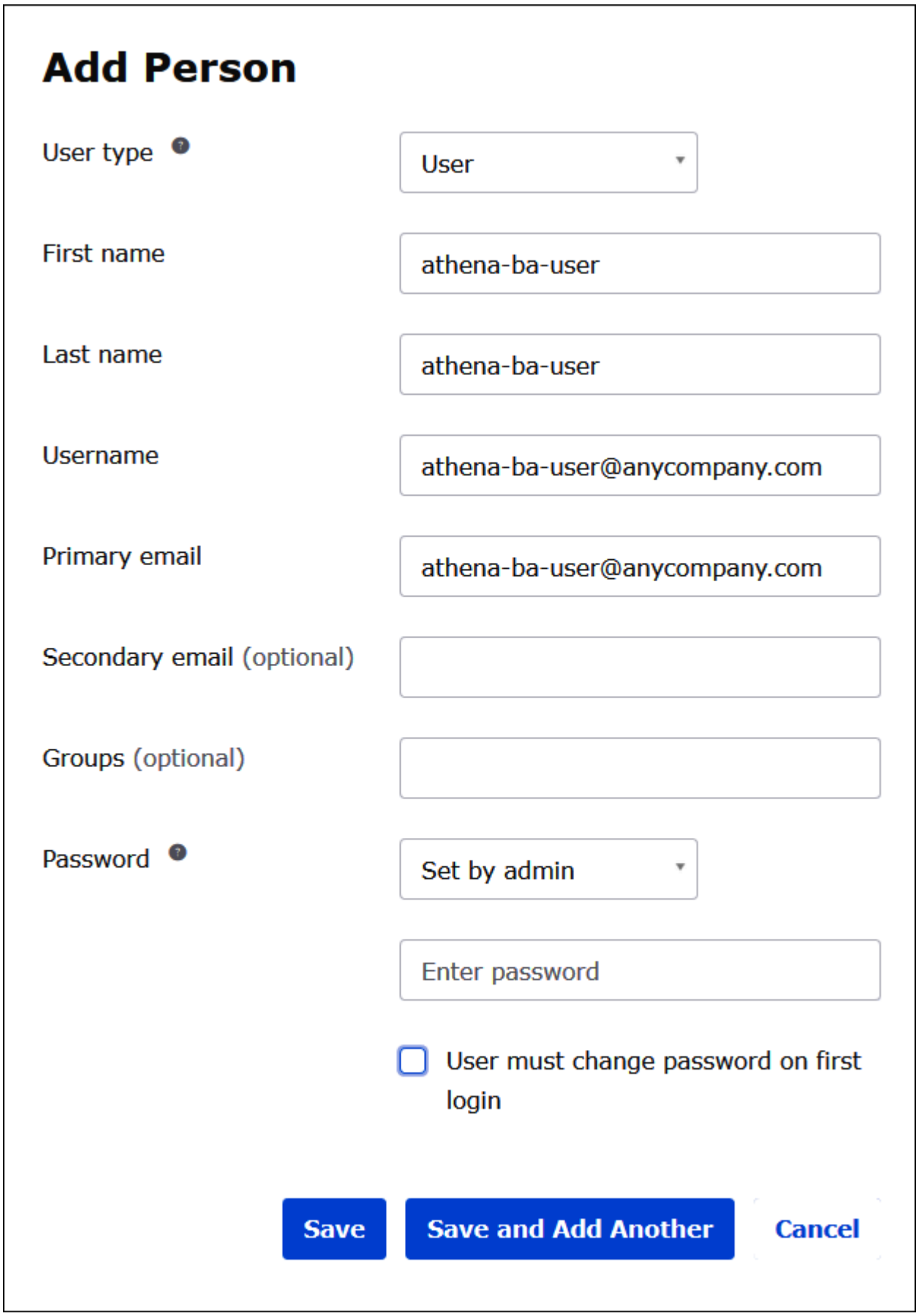
4. 在 Add Person（添加人员）对话框中，输入所需信息。

   - 对于 First name（名字）和 Last name（姓氏），输入值。本教程使用 *athena-okta-user*。

- 输入 Username（用户名）和 Primary email（主电子邮件）。本教程使用 *athena-okta-user@anycompany.com*。
- 对于 Password（密码），选择 Set by admin（由管理员设置），然后提供密码。本教程清除了 User must change password on the first login（用户必须在首次登录时更改密码）的选项；您的安全要求可能会有所不同。


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. 选择 Save and Add Another（保存并添加另一个）。
6. 输入其他用户的信息。此示例添加了业务分析师用户 *athena-ba-user@anycompany.com*。

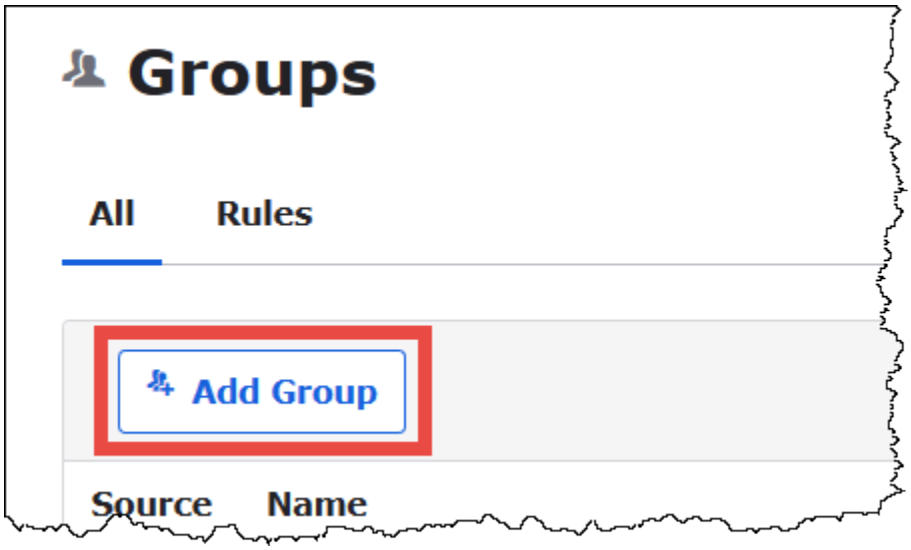

Add Person
User type
User
First name
athena-ba-user
Last name
athena-ba-user
Username
athena-ba-user@anycompany.com
Primary email
athena-ba-user@anycompany.com
Secondary email (optional)
Groups (optional)
Password
Set by admin
Enter password
User must change password on first login
Save
Save and Add Another
Cancel

7. 选择保存。

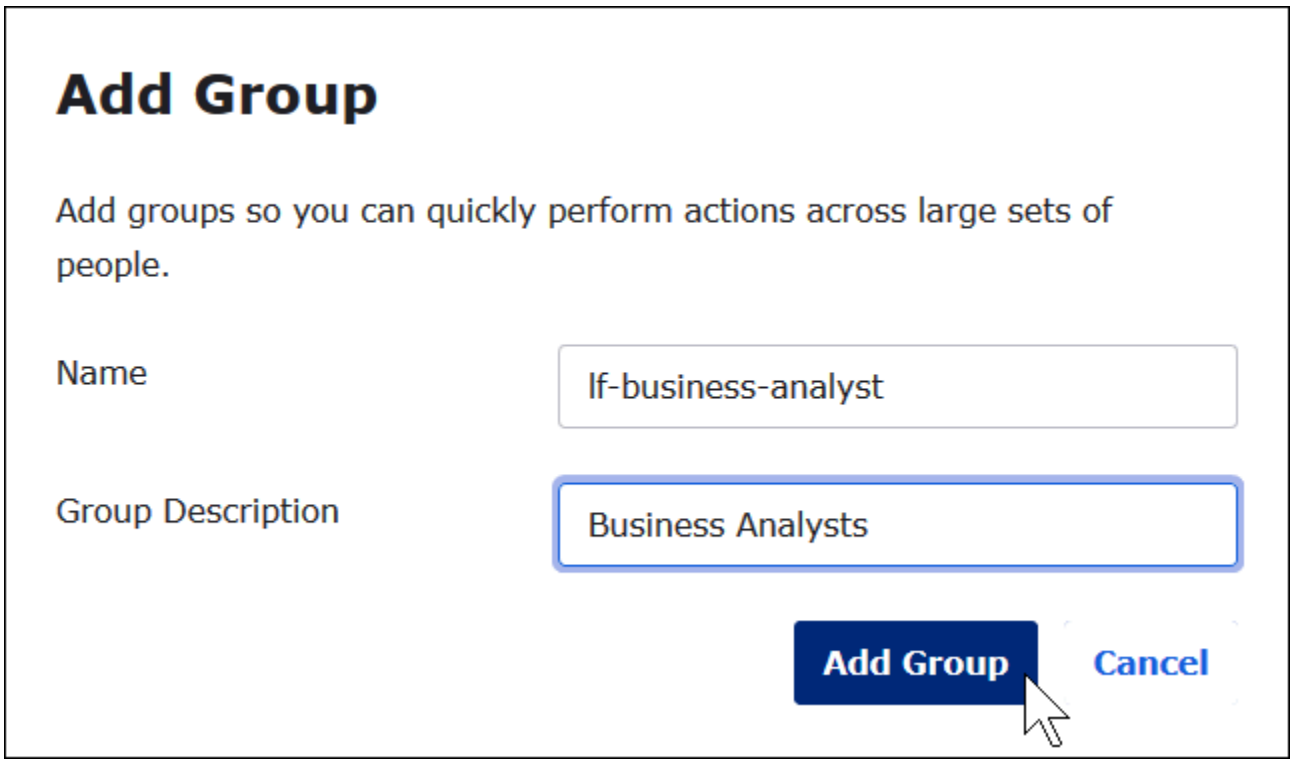
在以下过程中，您可以通过添 “业务分析师”组和“开发人员”组来通过 Athena JDBC 驱动程序提供两个 Okta 组的访问权限。

要添加 Okta 组

1. 在 Okta 导航窗格中，选择 Directory（目录），然后选择 Groups（组）。
2. 在 Groups（组）页面上，选择 Add Group（添加组）。



3. 在 Add Group（添加组）对话框中，输入所需信息。
   - 对于 Name（名称），输入 *lf-business-analyst*。
   - 对于 Group Description（组描述），输入######。

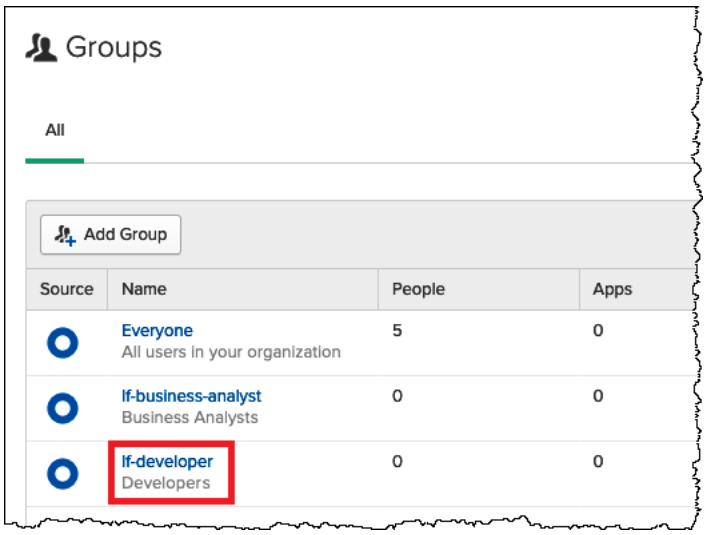


4. 选择添加组。
5. 在 Groups（组）页面上，再次选择 Add Group（添加组）。这一次，您将输入开发人员组的信息。
6. 输入所需的信息。
   - 对于 Name（名称），输入 *lf-developer*。
   - 对于 Group Description（组描述），输入####。
7. 选择添加组。

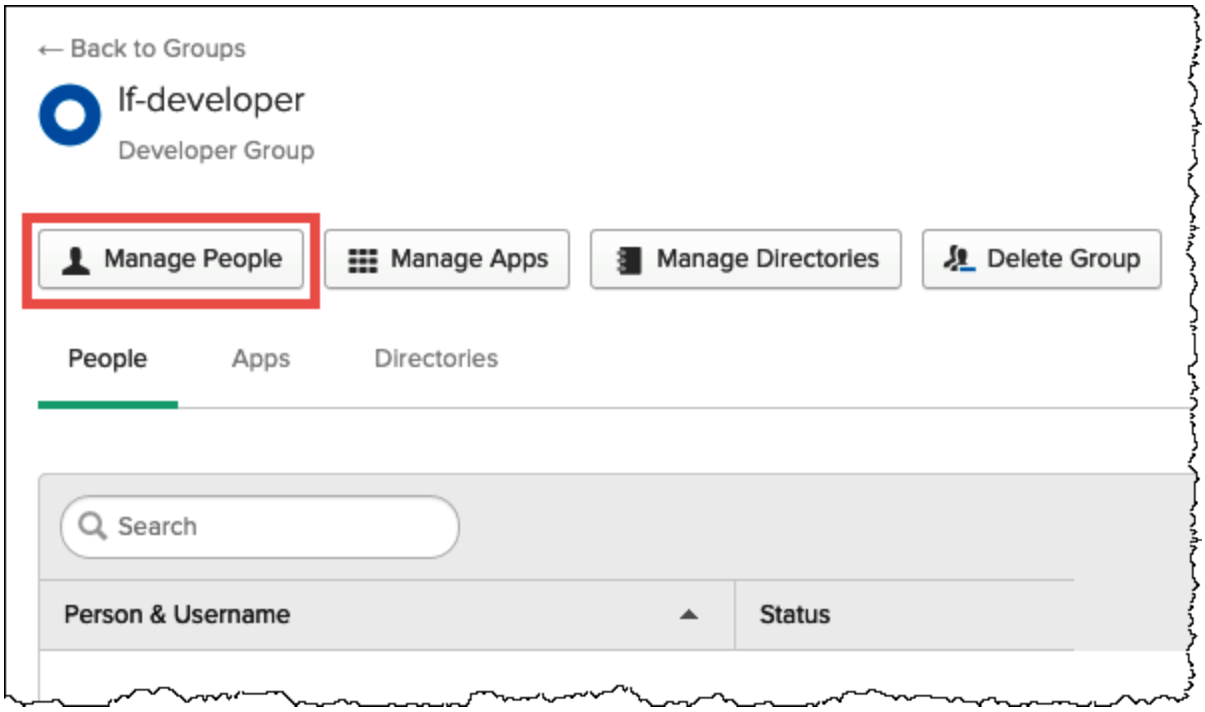
现在，您有两个用户和两个组，您已准备就绪，可以向每个组添加一个用户。

要将用户添加到组

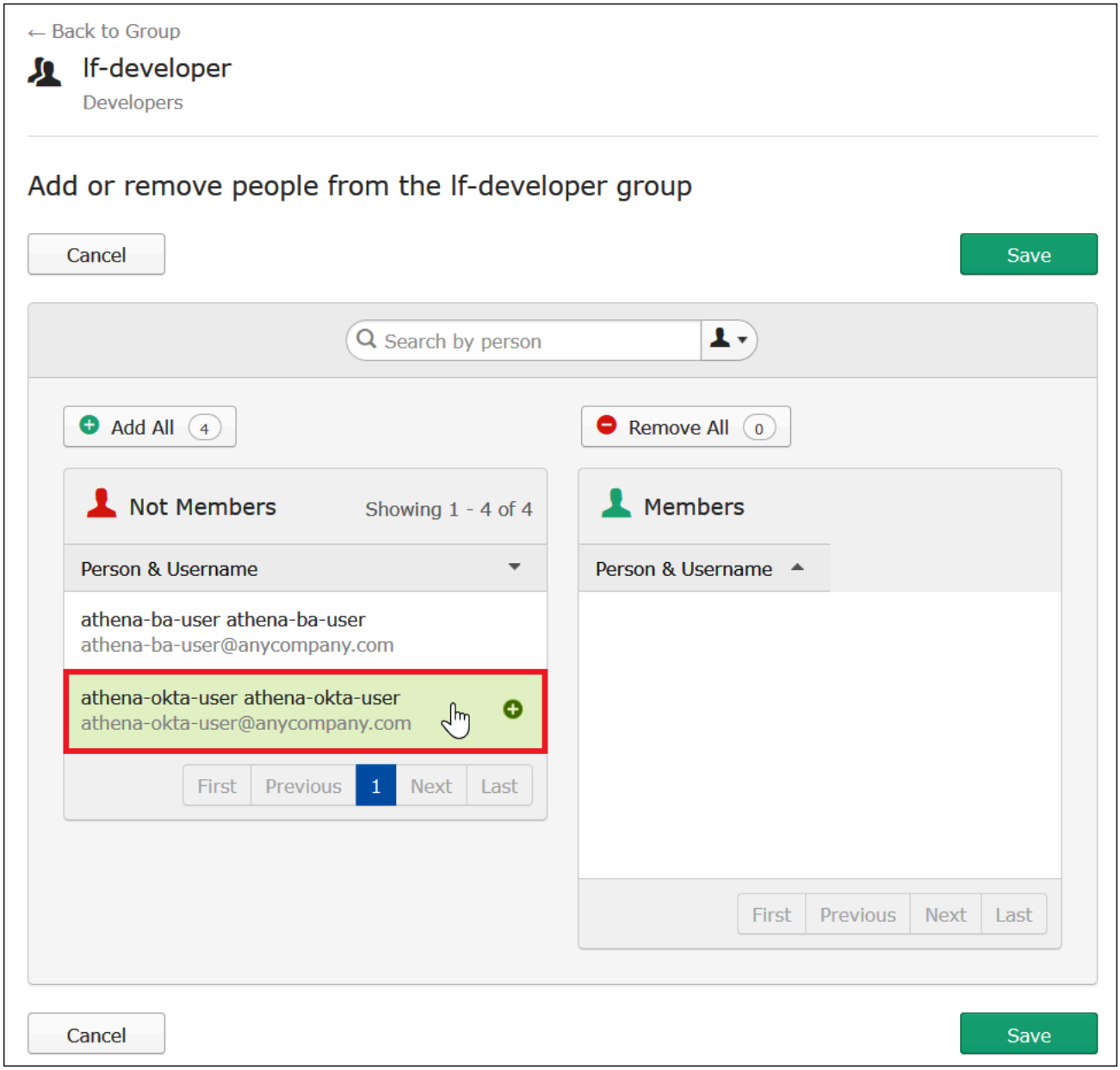
1. 在 Groups（组）页面上，选择您刚创建的 lf-developer 组。您需要将您作为开发人员创建的 Okta 用户之一添加到此组中。



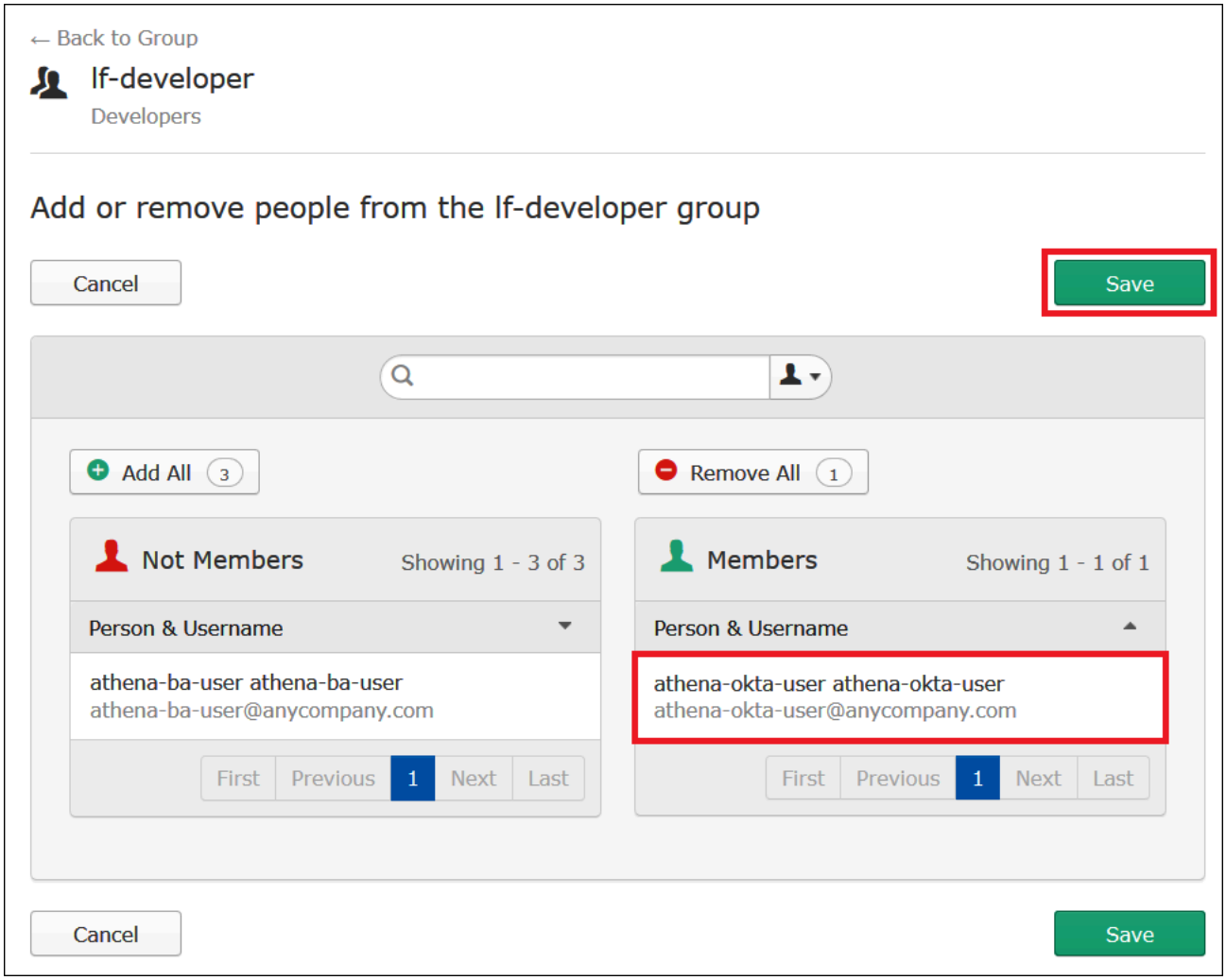
2. 选择 Manage People（管理人员）。



3. 从 Not Members（非成员）列表中，选择 athena-okta-user。

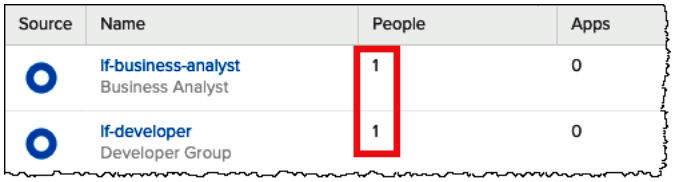


用户的条目将从左侧的非成员列表移动到右侧的成员列表。



4. 选择保存。
5. 选择 Back to Group（返回组），或选择 Directory（目录），然后选择 Groups（组）。
6. 选择 lf-business-analyst 组。
7. 选择 Manage People（管理人员）。
8. 向 lf-business-analyst 组的 Members（成员）列表添加 athena-ba-user，然后选择 Save（保存）。
9. 选择 Back to Group（返回组），或选择 Directory（目录），Groups（组）。

   Groups（组）页面现在显示每个组都有一个 Okta 用户。



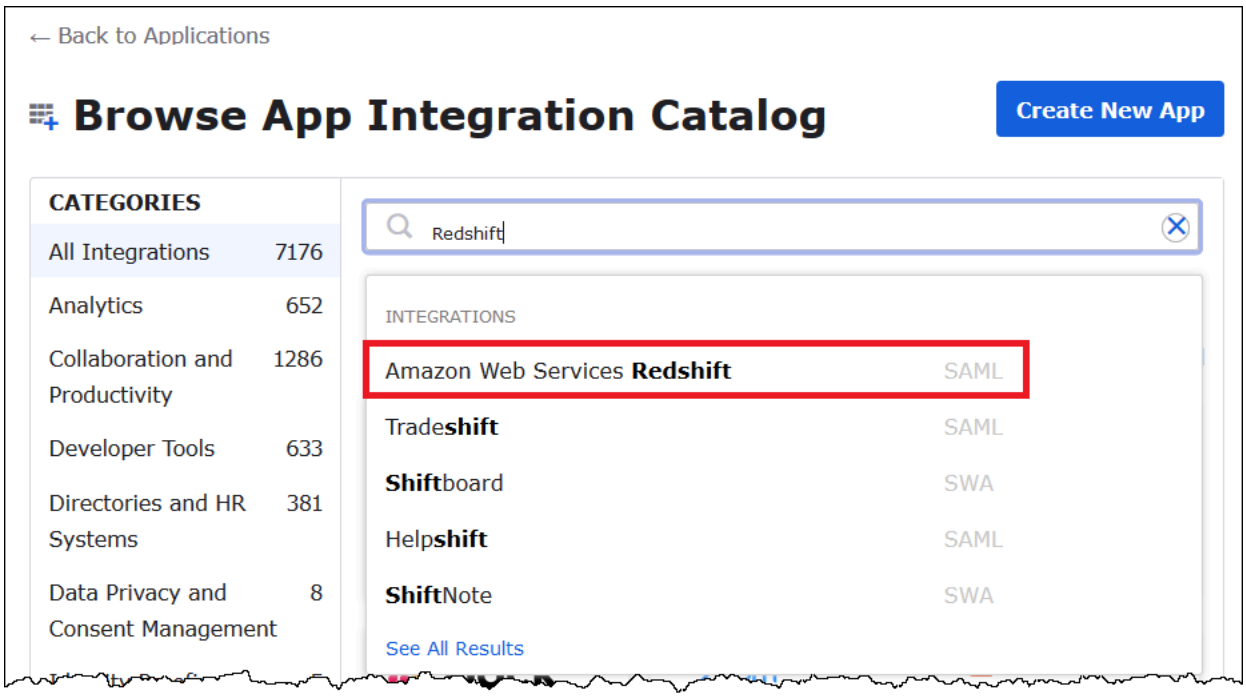
步骤 3：设置 Okta 应用程序以进行 SAML 身份验证

在此步骤中，您使用 Okta 开发人员控制台执行以下任务：

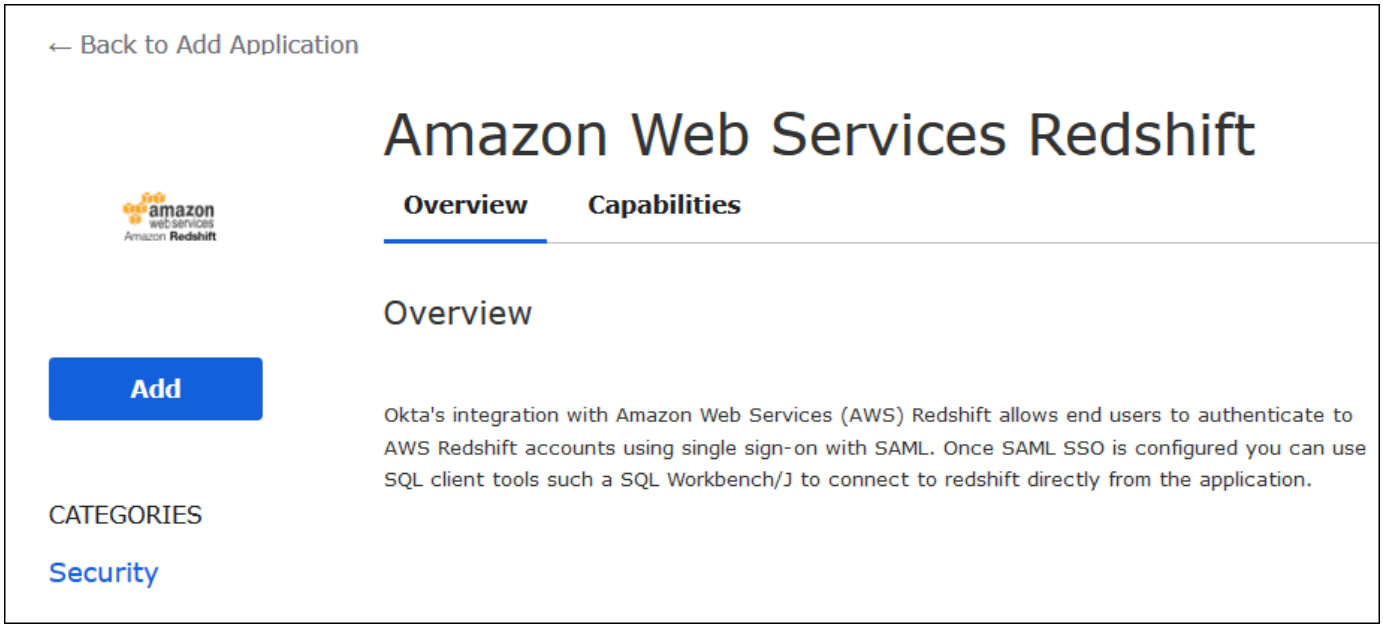
- 添加 SAML 应用程序以便与 AWS 结合使用。
- 将应用程序分配给 Okta 用户。
- 将应用程序分配给 Okta 组。
- 下载生成的身份提供程序元数据，以便日后与 AWS 结合使用。

要添加用于 SAML 身份验证的应用程序

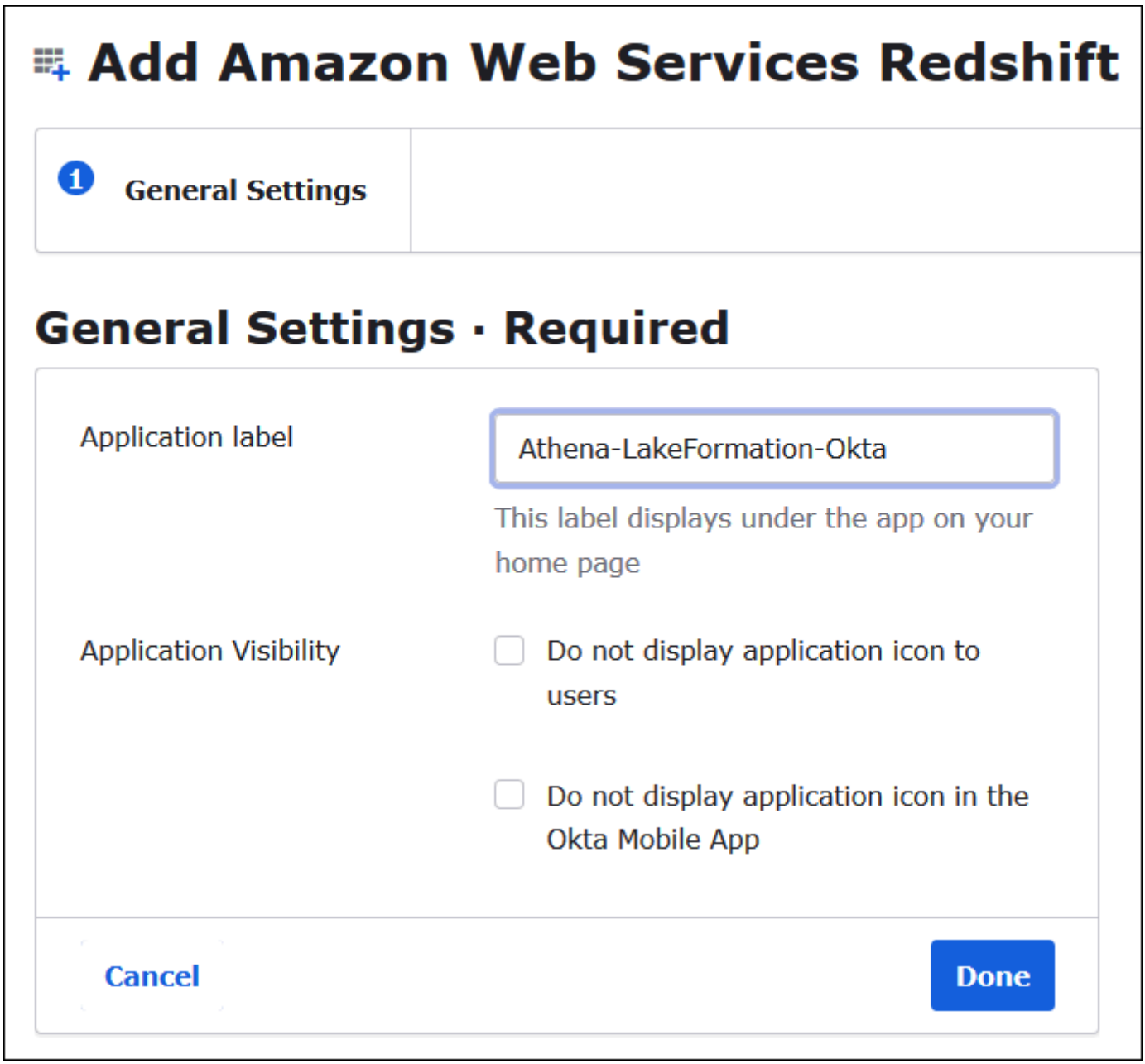
1. 在 Okta 导航窗格中，选择 Applications（应用程序）、Applications（应用程序），以便您可以为对 Athena 进行的 SAML 身份验证配置 Okta 应用程序。
2. 单击 Browse App Catalog（浏览应用程序目录）。
3. 在搜索框中，输入 **`Redshift`**。
4. 选择 Amazon Web Services Redshift。本教程中的 Okta 应用程序使用 Amazon Redshift 的现有 SAML 集成。



5. 在 Amazon Web Services Redshift 页面上，选择 Add（添加）为 Amazon Redshift 创建基于 SAML 的应用程序。



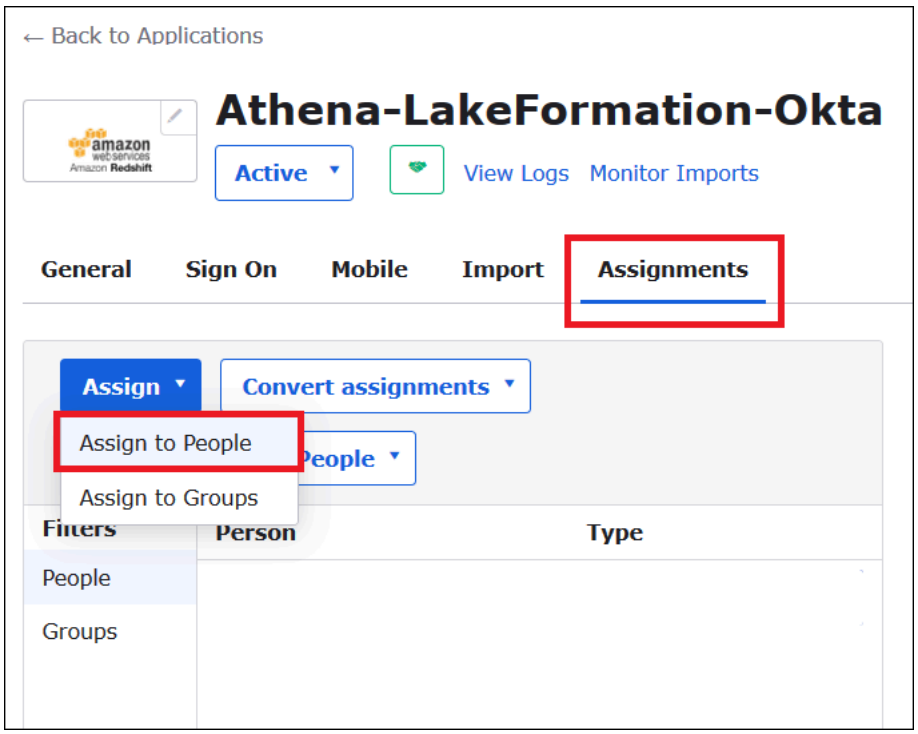
6. 对于 Application label（应用程序标签），输入 `Athena-LakeFormation-Okta`，然后选择 Done（完成）。



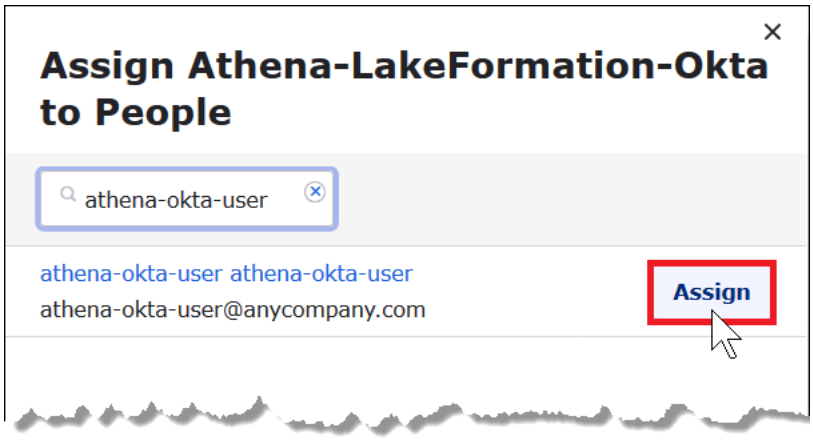
现在您已创建 Okta 应用程序，可以将其分配给您创建的用户和组。

要将应用程序分配给用户和组

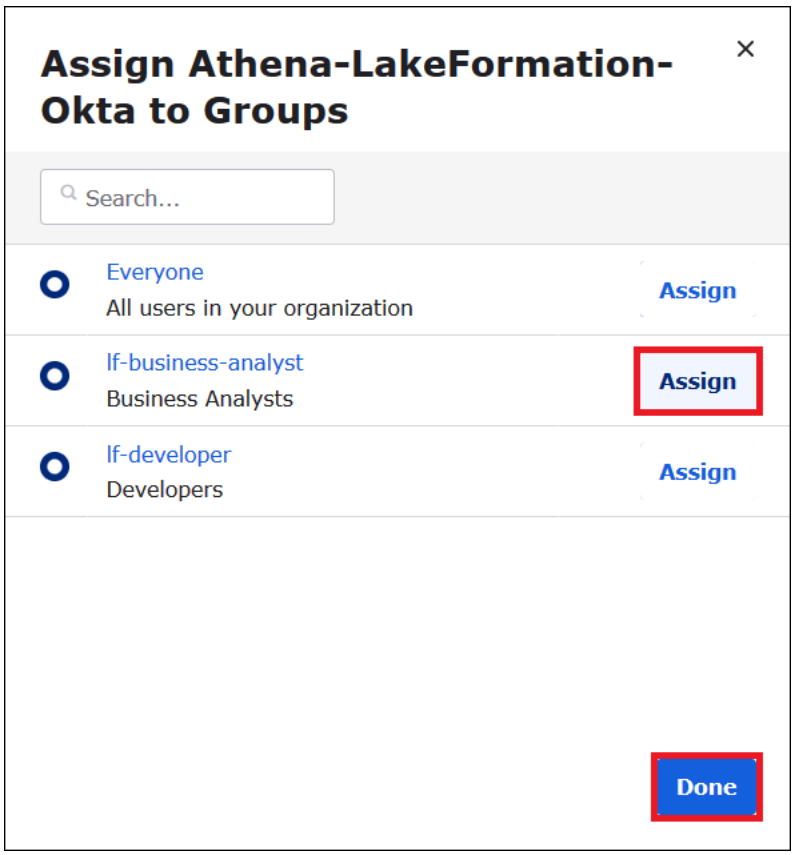
1. 在 Applications（应用程序）页面上，选择 Athena-LakeFormation-Okta 应用程序。
2. 在 Assignments（分配）选项卡上，选择 Assign（分配）、Assign to People（分配给人员）。



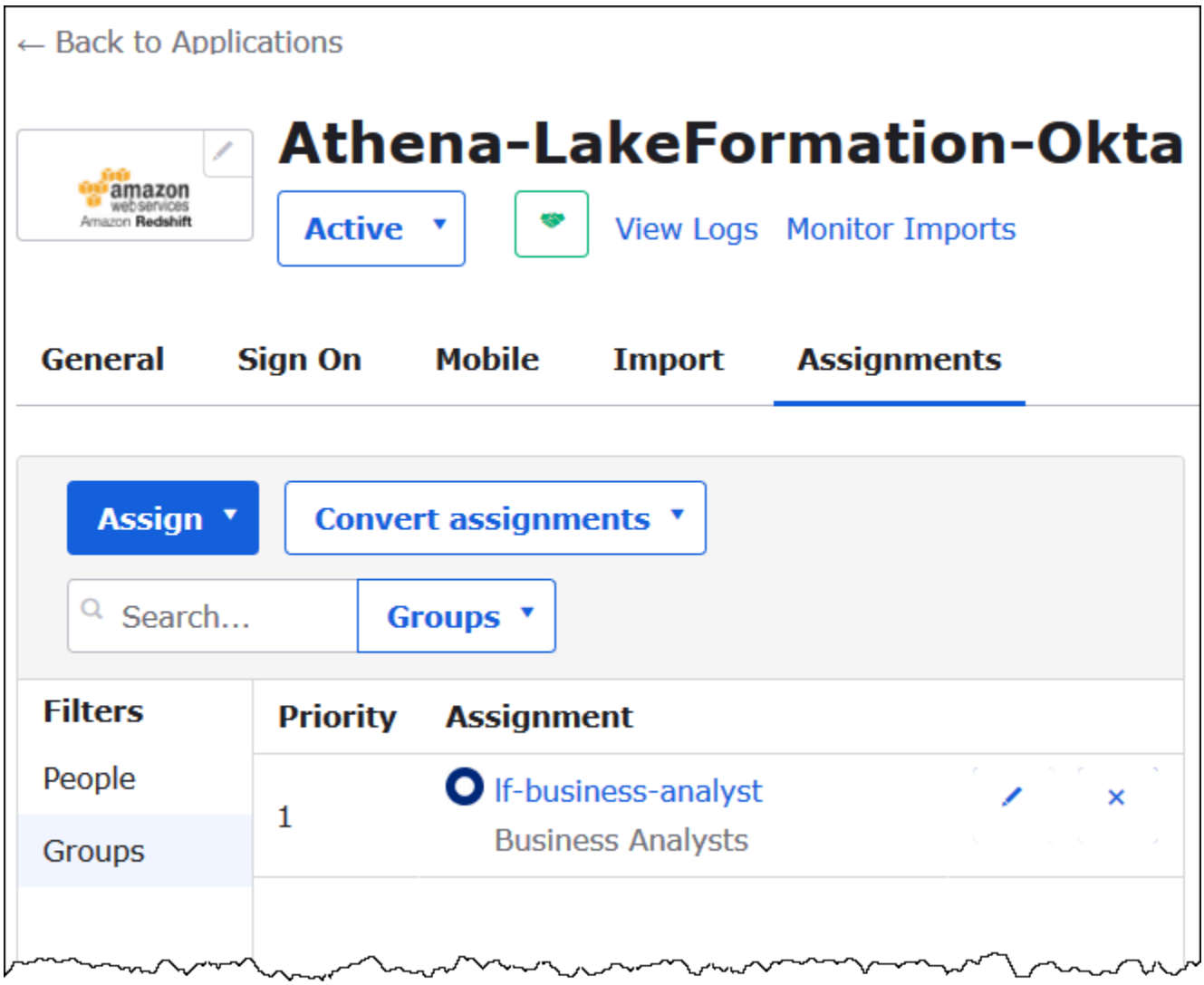
3. 在 Assign Athena-LakeFormation-Okta to People（将 Assign Athena-LakeFormation-Okta 分配给人员）对话框中，找到您之前创建的 athena-okta-user 用户。
4. 选择 Assign（分配）将用户分配给应用程序。



5. 选择 Save and Go Back（保存并返回）。
6. 选择完成。
7. 在 Athena-LakeFormation-Okta 应用程序的 Assignments（分配）选项卡上，选择 Assign（分配）、Assign to Groups（分配给组）。
8. 对于 lf-business-analyst，选择 Assign（分配）以将 Athena-LakeFormation-Okta 应用程序分配到 lf-business-analyst 组，然后选择Done（完成）。



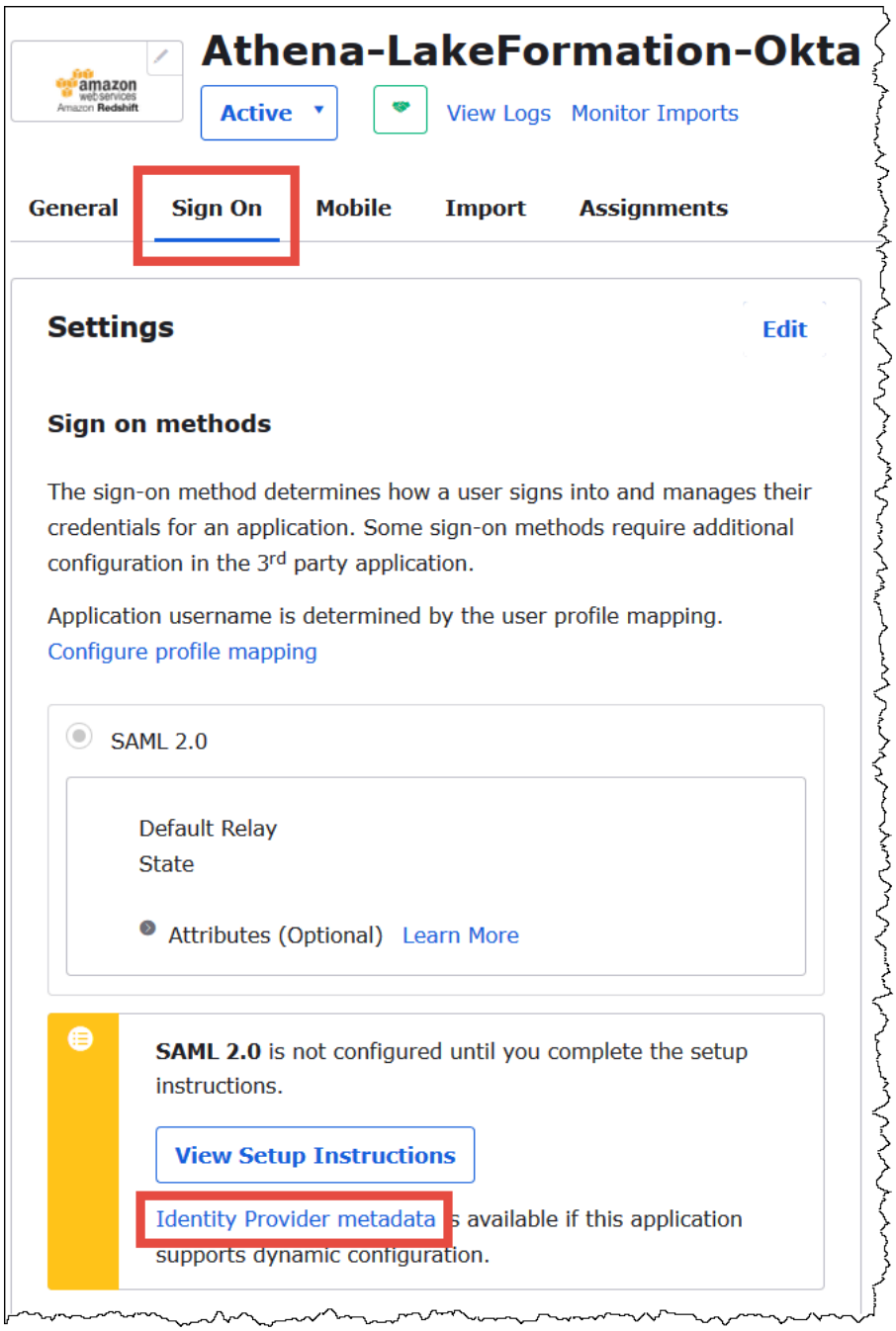
该组将显示在应用程序的组列表中。



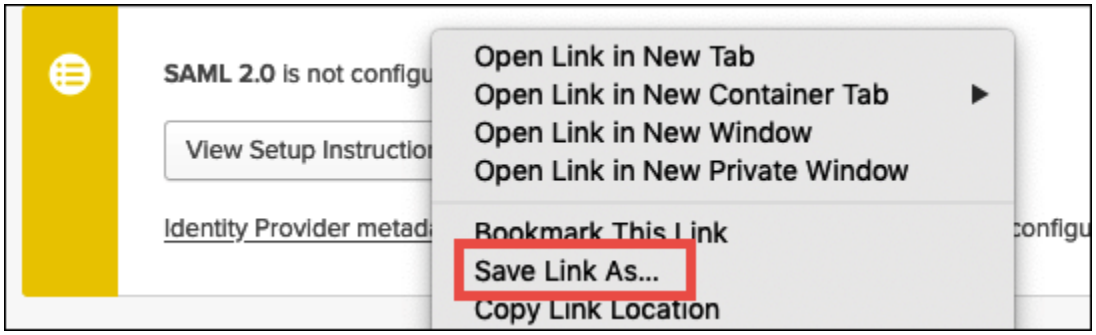
现在，您已准备就绪，可以下载身份提供程序应用程序元数据，以用于 AWS。

下载应用程序元数据

1. 选择 Okta 应用程序的 Sign On（登录）选项卡，然后右键单击 Identity Provider metadata（身份提供程序元数据）。



2. 选择 Save Link As（将链接另存为）以将身份提供程序元数据（XML 格式）保存到文件中。给它一个您可以识别的名称（例如，`Athena-LakeFormation-idp-metadata.xml`）。



步骤 4：创建 AWS SAML 身份提供程序和 Lake Formation 访问 IAM 角色

在此步骤中，您将使用 AWS Identity and Access Management (IAM) 控制台执行以下任务：

- 为 AWS 创建身份提供程序。
- 创建针对 Lake Formation 访问的 IAM 角色。
- 将 AmazonAthenaFullAccess 托管策略添加到角色。
- 向角色添加 Lake Formation 策略和 AWS Glue。
- 向角色添加 Athena 查询结果的策略。

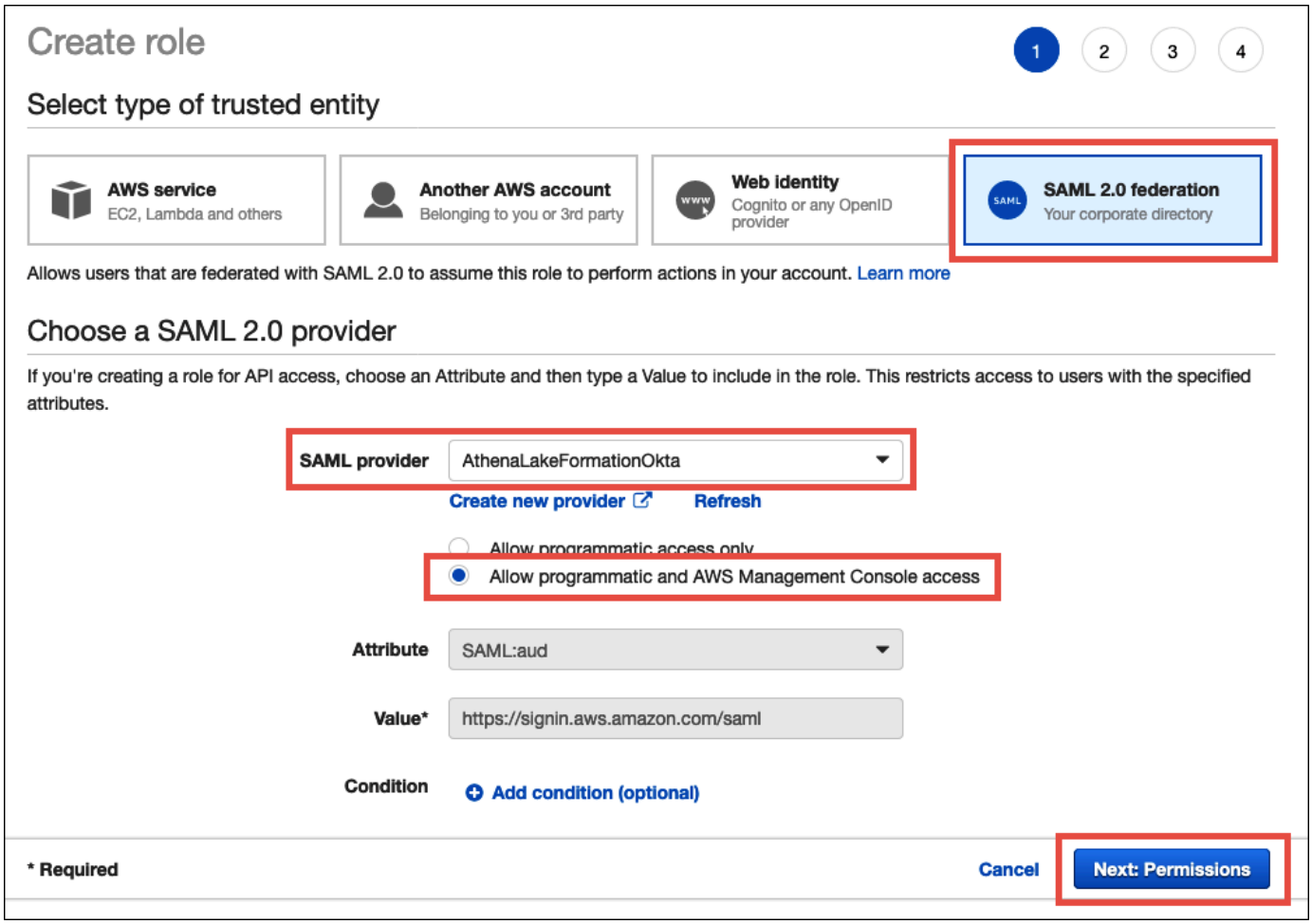
要创建 AWS SAML 身份提供程序

1. 以 Amazon Web Services 账户管理员身份登录 Amazon Web Services 账户控制台，并导航到 IAM 控制台（https://console.aws.amazon.com/iam/）。
2. 在导航窗格中，选择 Identity providers（身份提供程序），然后单击 Add provider（添加提供程序）。
3. 在 Define provider（定义提供程序）页面上，输入以下信息：
   - 对于 Provider type（提供程序类型），选择 SAML。
   - 对于 Provider name（提供程序名称），输入 `AthenaLakeFormationOkta`。
   - 对于 Metadata document（元数据文档），请使用 Select file（选择文件）选项上载您下载的身份提供程序 (IdP) 元数据 XML 文件。
4. 选择 Add provider（添加提供程序）。

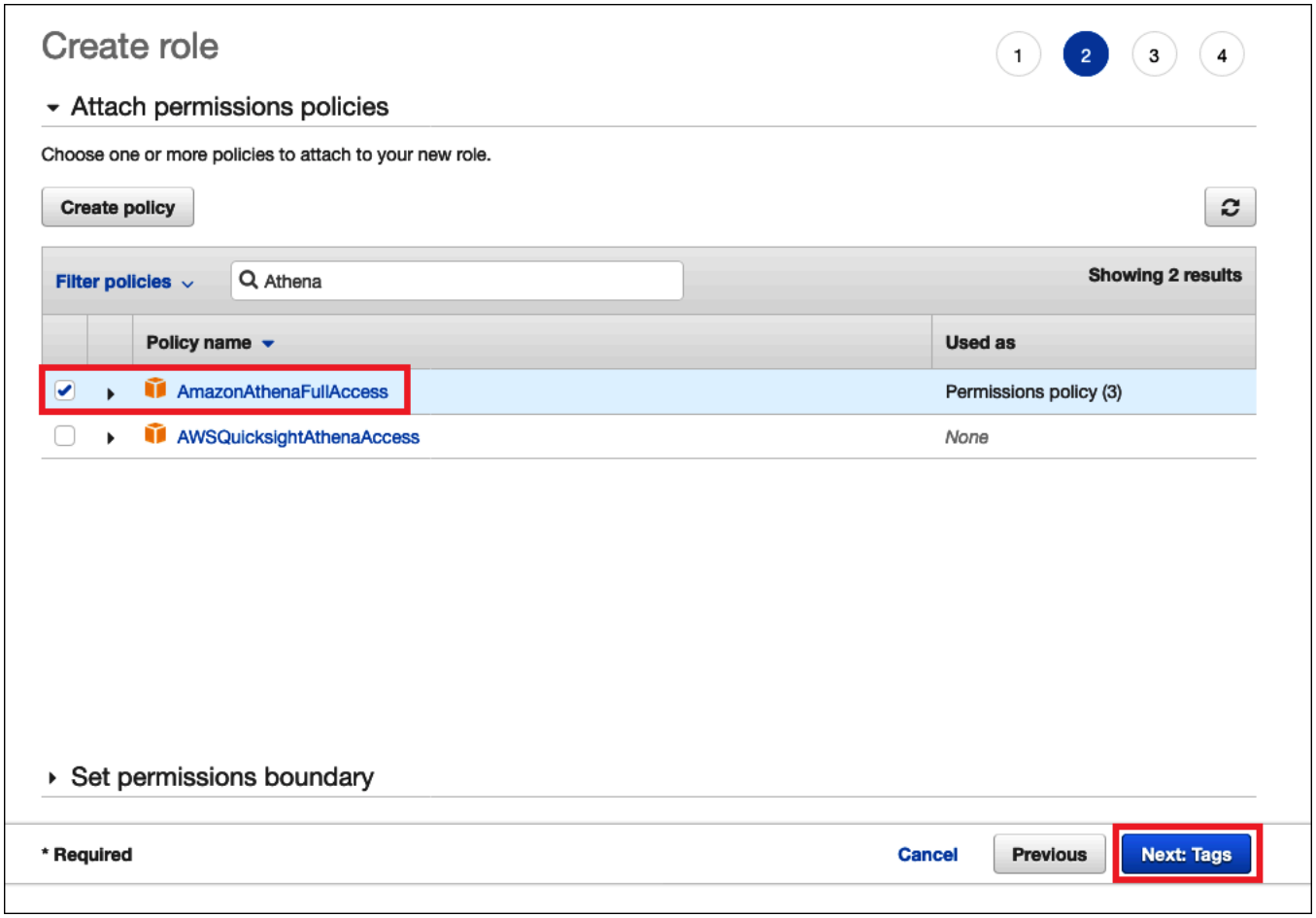
接下来，您将为 AWS Lake Formation 访问创建 IAM 角色。您将两个内联策略添加到角色。其中一个策略提供访问 Lake Formation 和 AWS Glue API 的权限。另一个策略提供了对 Amazon S3 中 Athena 和 Athena 查询结果位置的访问权限。

要为 AWS Lake Formation 创建 IAM 角色以进行访问

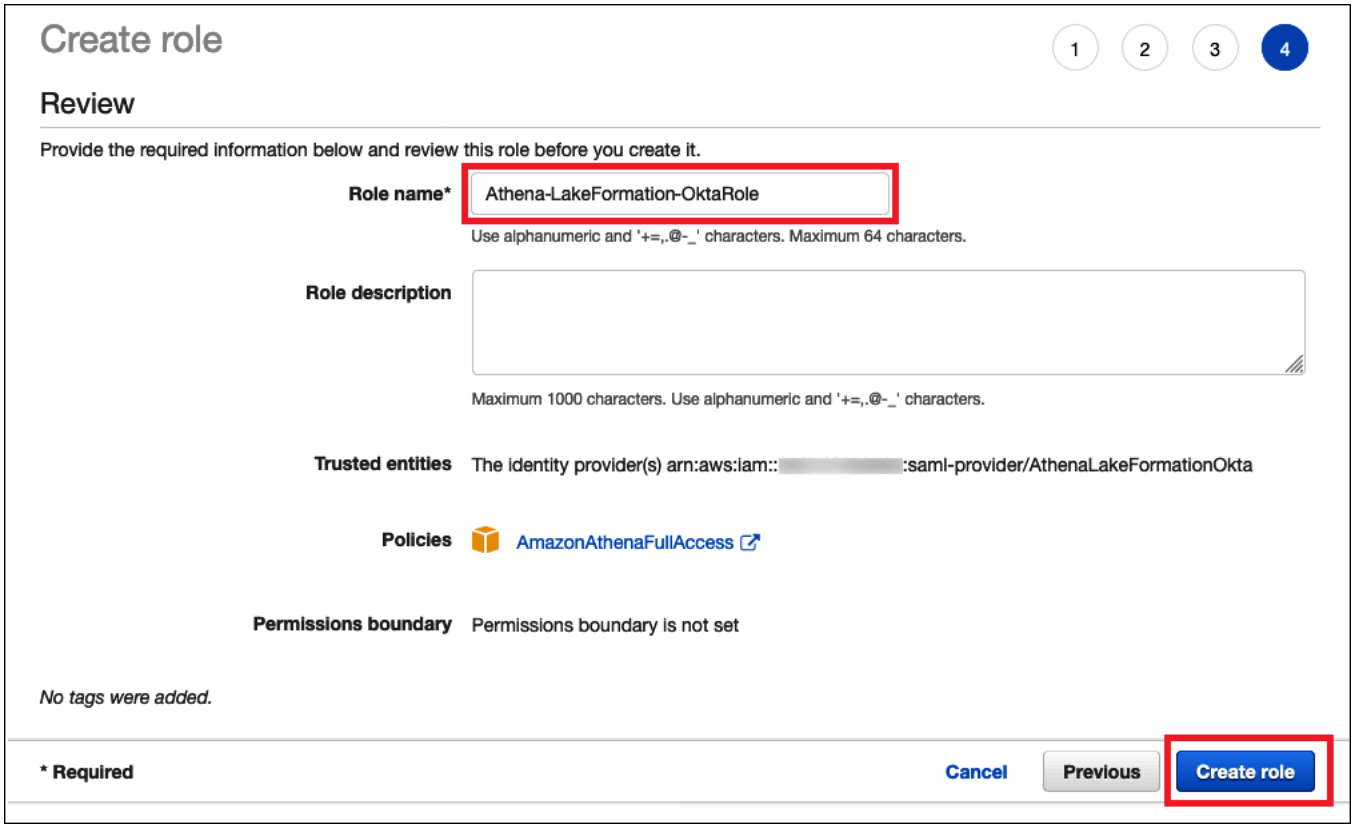
1. 在 IAM 控制台的导航窗格中，选择 Roles（角色），然后选择 Create role（创建角色）。
2. 在 Create role（创建角色）页面上，执行以下步骤：



   a. 对于 Select type of trusted entity（选择受信任实体的类型），选择 SAML 2.0 Federation。
   b. 对于 SAML provider（SAML 提供程序），选择 AthenaLakeFormationOkta。
   c. 对于 SAML provider（SAML 提供程序），选择选项 Allow programmatic and AWS Management Console access（允许编程和控制台访问）。
   d. 选择下一步：权限。
3. 在 Attach Permissions policies（附加权限策略）页面，对于 Filter pilicies（筛选策略），输入 **`Athena`**。
4. 选择名为 AmazonAthenaFullAccess 的托管式策略，然后选择 Next: Tags（下一步：标签）。



5. 在 Add tags（添加标签）页面上，选择 Next: Review（下一步：审核）。
6. 在 Review（审核）页面上，对于 Role name（角色名称），输入角色的名称（例如，*Athena-LakeFormation-OktaRole*），然后选择 Create role（创建角色）。



接下来，您可以添加允许访问 Lake Formation、AWS Glue API 和 Amazon S3 中 Athena 查询结果的内联策略。

每当您使用 IAM 策略时，请确保遵循 IAM 最佳实践。有关更多信息，请参阅《IAM 用户指南》中的 IAM 安全最佳实践。

要将内联策略添加到 Lake Formation 和 AWS Glue 的角色

1. 从 IAM 控制台中的角色列表中，选择新创建的 `Athena-LakeFormation-OktaRole`。
2. 在角色的 Summary（摘要）页面上，在 Permissions（权限）选项卡上，选择 Add inline policy（添加内联策略）。
3. 在创建策略页面上，选择 JSON。
4. 添加一个内联策略，如下所示，该策略可提供访问 Lake Formation 和 AWS Glue API 的权限。

```
{
    "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. 选择查看策略。
6. 对于 Name（名称），请为策略输入名称（例如，**LakeFormationGlueInlinePolicy**）。
7. 选择创建策略。

要将内联策略添加到 Athena 查询结果位置的角色

1. 在 Athena-LakeFormation-OktaRole 角色的 Summary（摘要）页面上，在 Permissions（权限）选项卡上，选择 Add inline policy（添加内联策略）。
2. 在创建策略页面上，选择 JSON。
3. 添加允许角色访问 Athena 查询结果位置的内联策略，如下所示。将示例中的 *<athena-query-results-bucket>* 占位符替换为您的 Amazon S3 存储桶的名称。

```
{
    "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. 选择查看策略。
5. 对于 Name（名称），请为策略输入名称（例如，**AthenaQueryResultsInlinePolicy**）。
6. 选择创建策略。

接下来，您需要复制 Lake Formation 访问角色的 ARN 和您创建的 SAML 提供程序的 ARN。当您在本教程的下一部分中配置 Okta SAML 应用程序时，这些都是必需的。

要复制角色 ARN 和 SAML 身份提供程序 ARN

1. 在 IAM 控制台中，在 `Athena-LakeFormation-OktaRole` 角色的 Summary（摘要）页面中，选择 Role ARN（角色 ARN）旁的 Copy to clipboard（复制到剪贴板）图标。ARN 有以下格式：

```
arn:aws:iam::<account-id>:role/Athena-LakeFormation-OktaRole
```

2. 安全地保存完整 ARN 以供日后参考。
3. 在 IAM 控制台导航窗格中，选择 Identity providers（身份提供程序）。
4. 选择 AthenaLakeFormationOkta 提供程序。
5. 在 Summary（摘要）页面上，选择 Provider ARN（提供程序 ARN）旁的 Copy to clipboard（复制到剪贴板）图标。ARN 应如下所示：

```
arn:aws:iam::<account-id>:saml-provider/AthenaLakeFormationOkta
```

6. 安全地保存完整 ARN 以供日后参考。

步骤 5：将 IAM 角色和 SAML 身份提供程序添加到 Okta 应用程序

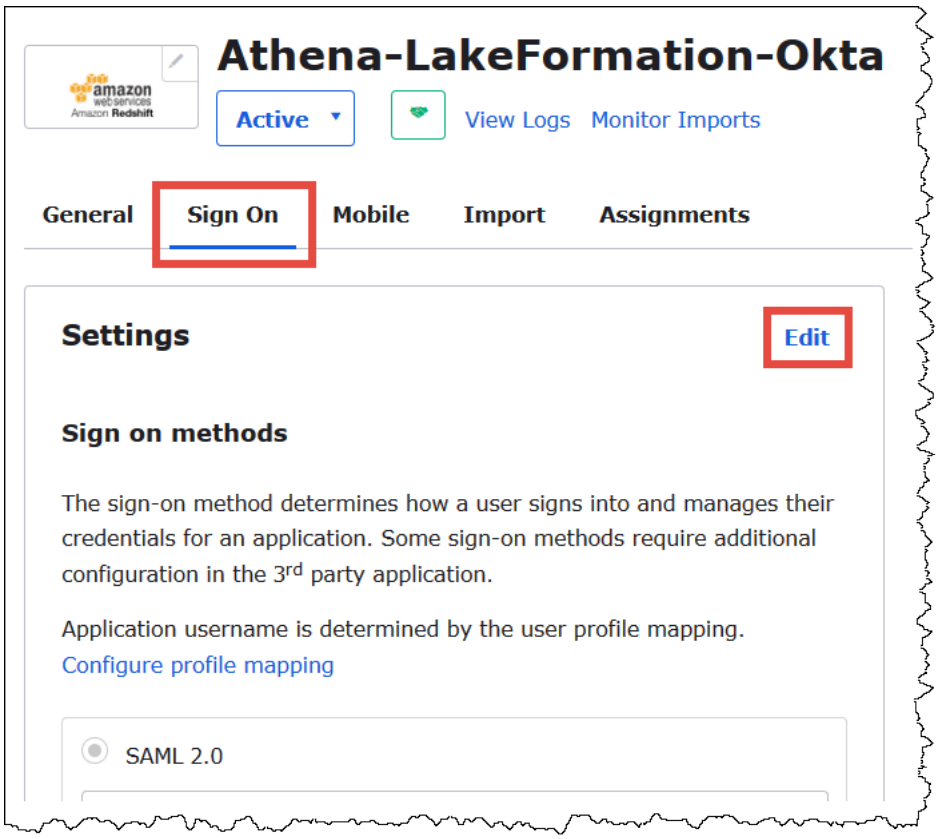
在此步骤中，您将返回到 Okta 开发人员控制台并执行以下任务：

- 将用户和组 Lake Formation URL 属性添加到 Okta 应用程序。
- 将身份提供程序的 ARN 和 IAM 角色的 ARN 添加到 Okta 应用程序。
- 复制 Okta 应用程序 ID。连接到 Athena 的 JDBC 配置文件中需要 Okta 应用程序 ID。

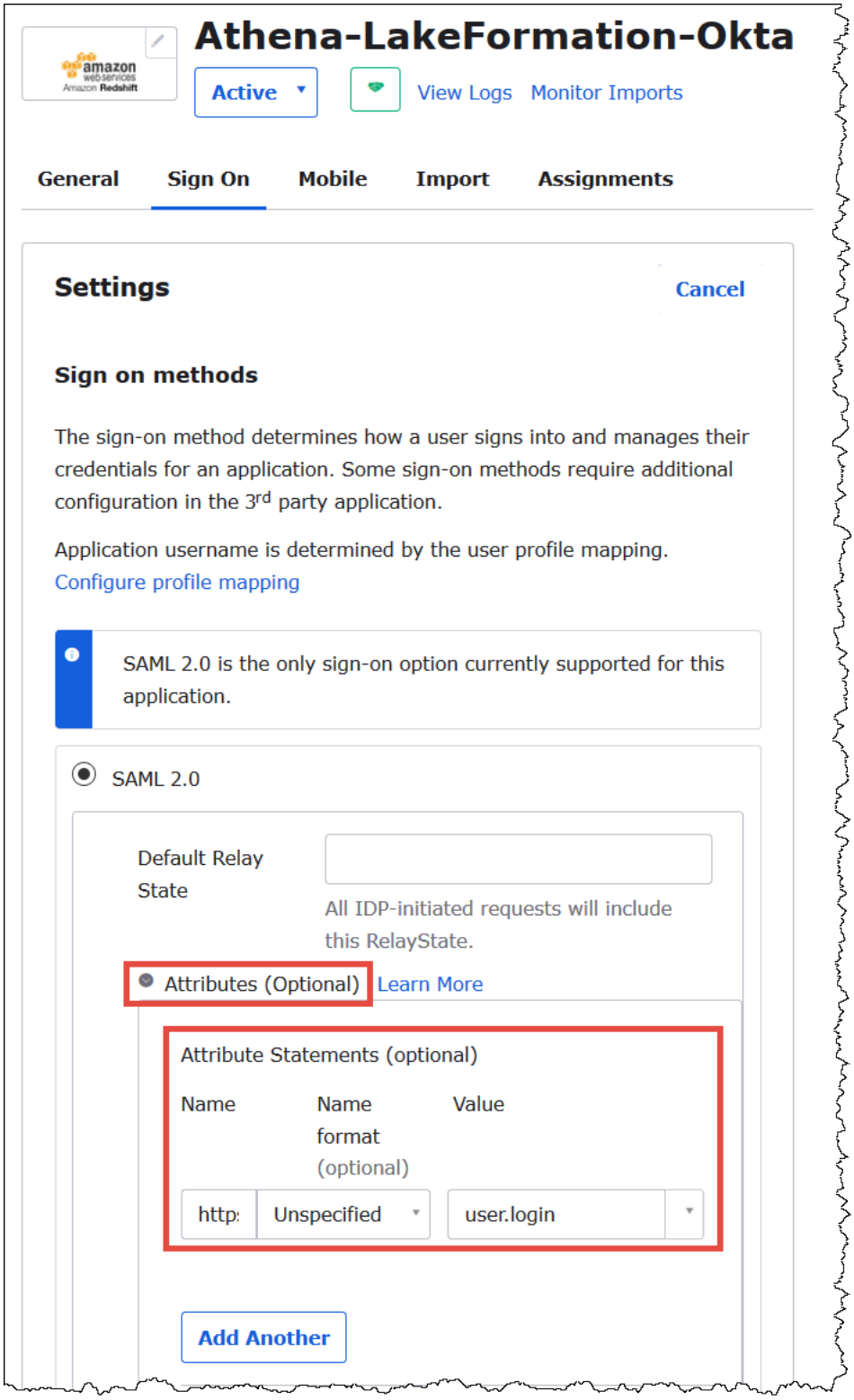
将用户和组 Lake Formation URL 属性添加到 Okta 应用程序

1. 登录 Okta 开发人员控制台。

2. 在 Applications（应用程序）选项卡上，然后选择 `Athena-LakeFormation-Okta` 应用程序。
3. 选择应用程序的 Sign On（登录）选项卡，然后选择 Edit（编辑）。



4. 选择 Attributes (optional)（属性（可选））来扩展它。



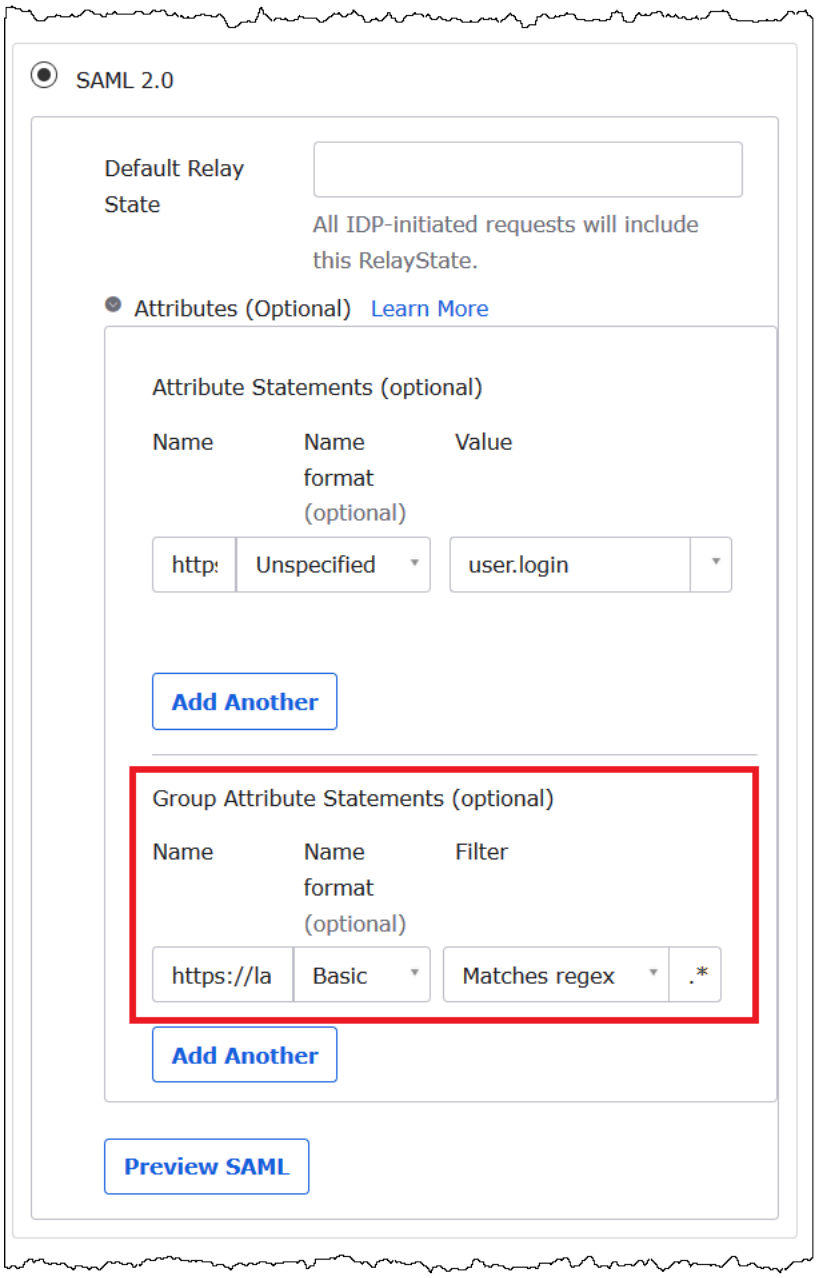
5. 对于 Attribute Statements (optional)（属性语句（可选）），添加以下属性：

   - 对于名称，请输入 **`https://lakeformation.amazon.com/SAML/Attributes/Username`**。

- 对于 Value（值），输入 **`user.login`**

6. 在 Group Attribute Statements (optional)（组属性语句（可选））项下，添加以下属性：

   - 对于名称，请输入 **`https://lakeformation.amazon.com/SAML/Attributes/Groups`**。
   - 对于 Name format（名称格式），输入 **`Basic`**
   - 对于 Filter（筛选条件），选择 Matches regex（匹配正则表达式），然后在筛选条件框中输入 **`.*`**。

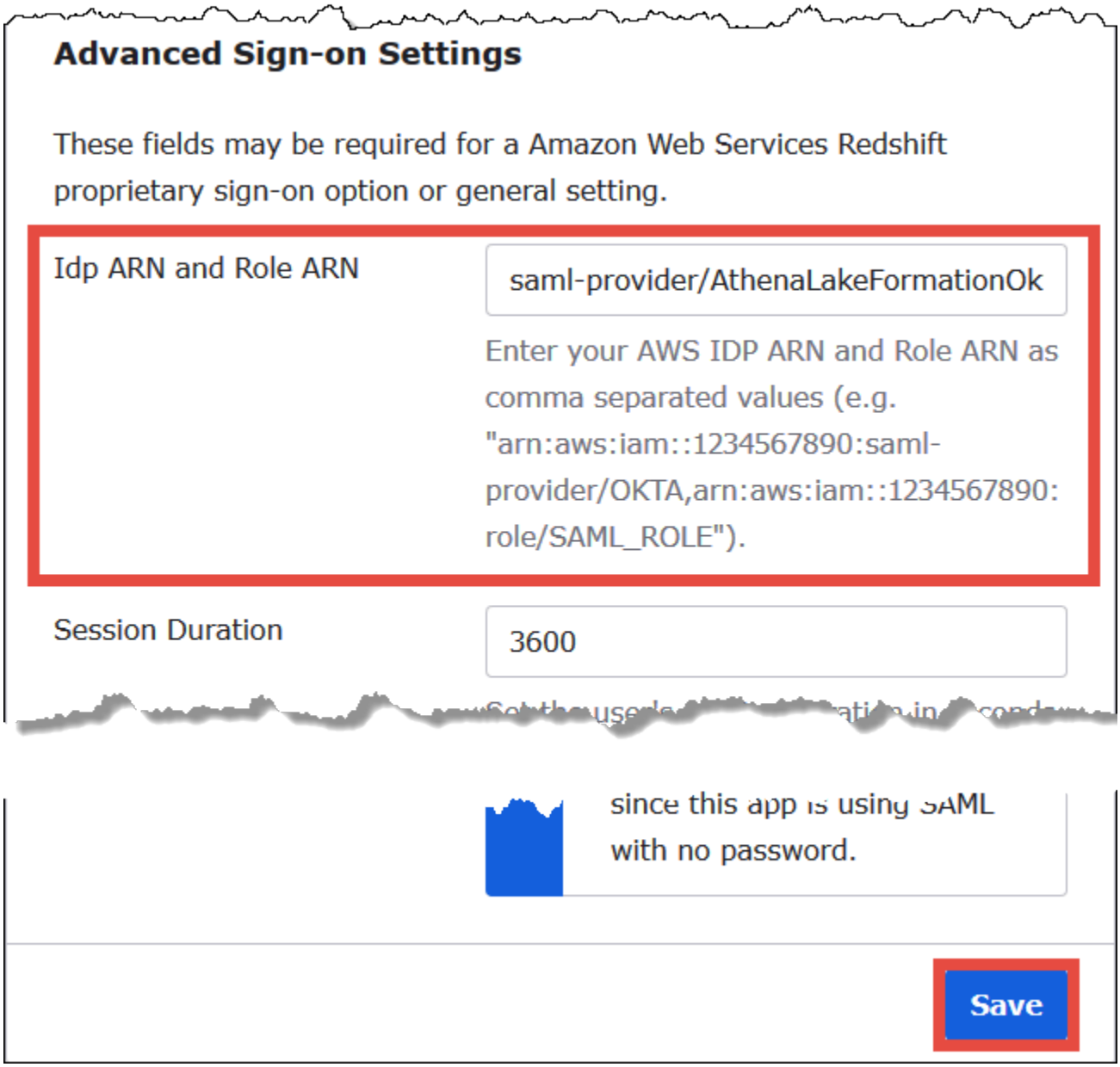


7. 向下滚动到 Advanced Sign-On Settings（高级登录设置）部分，您将在其中将身份提供程序和 IAM 角色 ARN 添加到 Okta 应用程序。

将身份提供程序和 IAM 角色的 ARN 添加到 Okta 应用程序

1. 对于 Idp ARN 和角色 ARN，输入 AWS 身份提供程序 ARN 和角色 ARN，作为逗号分隔的值，格式为 *<saml-arn>*、*<role-arn>*。完成字符串应与以下内容类似：

```
arn:aws:iam::<account-id>:saml-provider/
AthenaLakeFormationOkta,arn:aws:iam::<account-id>:role/Athena-LakeFormation-
OktaRole
```

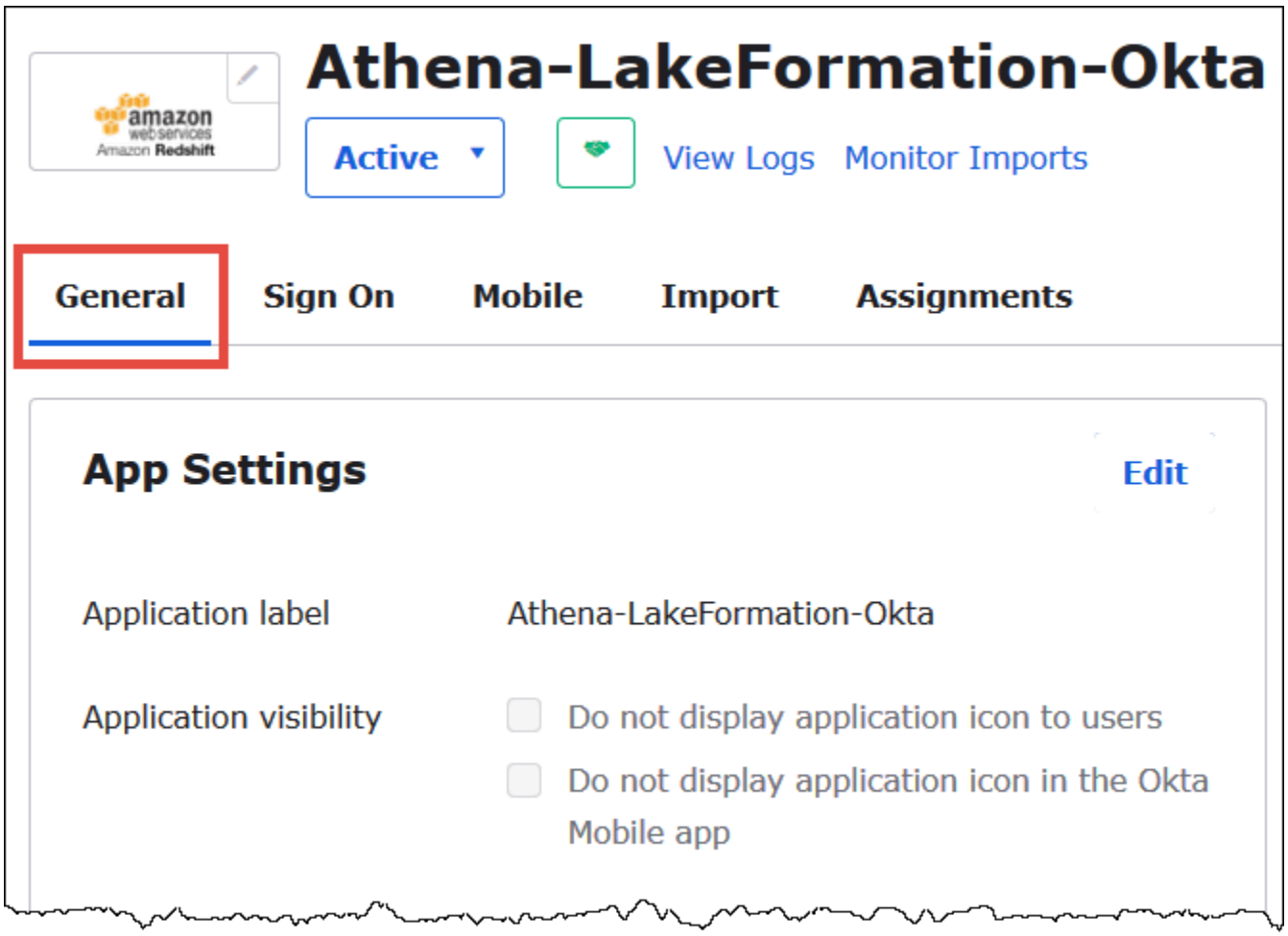


2. 选择保存。

接下来，您将复制 Okta 应用程序 ID。您稍后需要在连接到 Athena 的 JDBC 字符串中用到它。

要查找和复制 Okta 应用程序 ID

1. 选择 Okta 应用程序的 General（常规）选项卡。



2. 向下滚动到 App Embed Link（应用程序嵌入链接）部分。
3. 从 Embed Link（嵌入链接），复制并安全地保存 URL 的 Okta 应用程序 ID 部分。Okta 应用程序 ID 是 `amazon_aws_redshift/` 之后的 URL 部分内容，但在下一个正斜杠之前。例如，如果 URL 包含 `amazon_aws_redshift/aaa/bbb`，则应用程序 ID 为 `aaa`。

> **Note**
>
> 嵌入的链接不能用于直接登录 Athena 控制台以查看数据库。仅当您使用 JDBC 或 ODBC 驱动程序向 Athena 提交查询时，才认可 SAML 用户和组的 Lake Formation 权限。要查看数据库，可以使用 SQL WorkBench/J 工具，该工具会使用 JDBC 驱动程序连接到 Athena。有关 SQL Workbench/J 工具的介绍详见步骤 7：验证通过 Athena JDBC 客户端的访问。

步骤 6：通过 AWS Lake Formation 授予用户和组权限

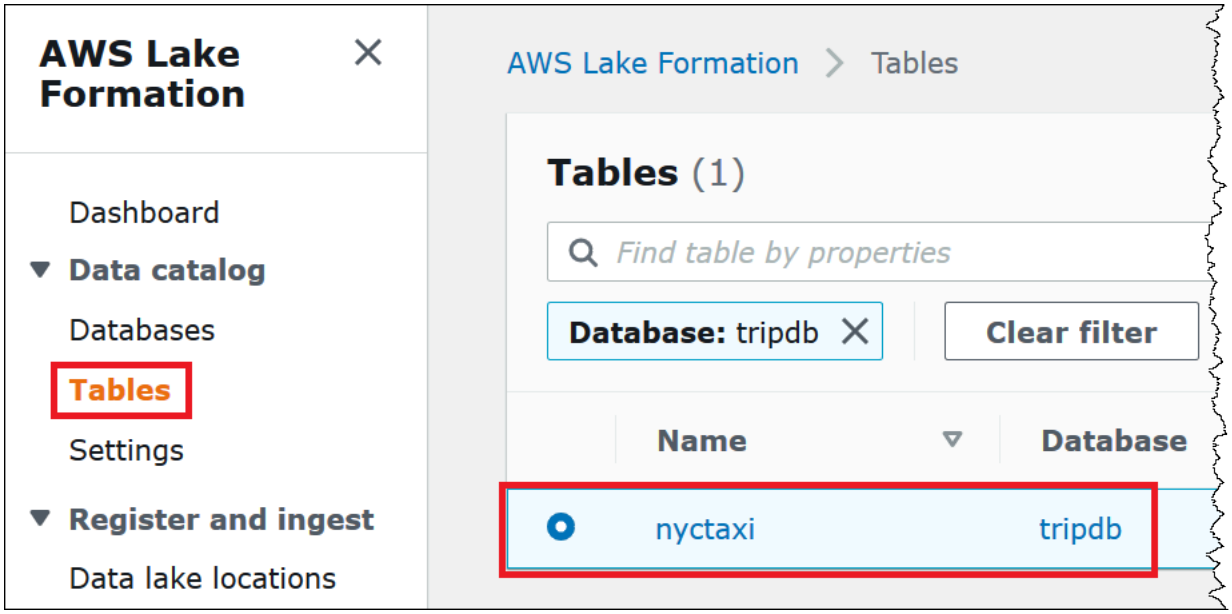
在此步骤中，您可以使用 Lake Formation 控制台向 SAML 用户和组授予对表的权限。您可以执行以下任务：

- 指定 Okta SAML 用户以及关联用户的 ARN 对表的权限。
- 指定 Okta SAML 组以及关联组的 ARN 对表的权限。
- 验证您授予的权限。

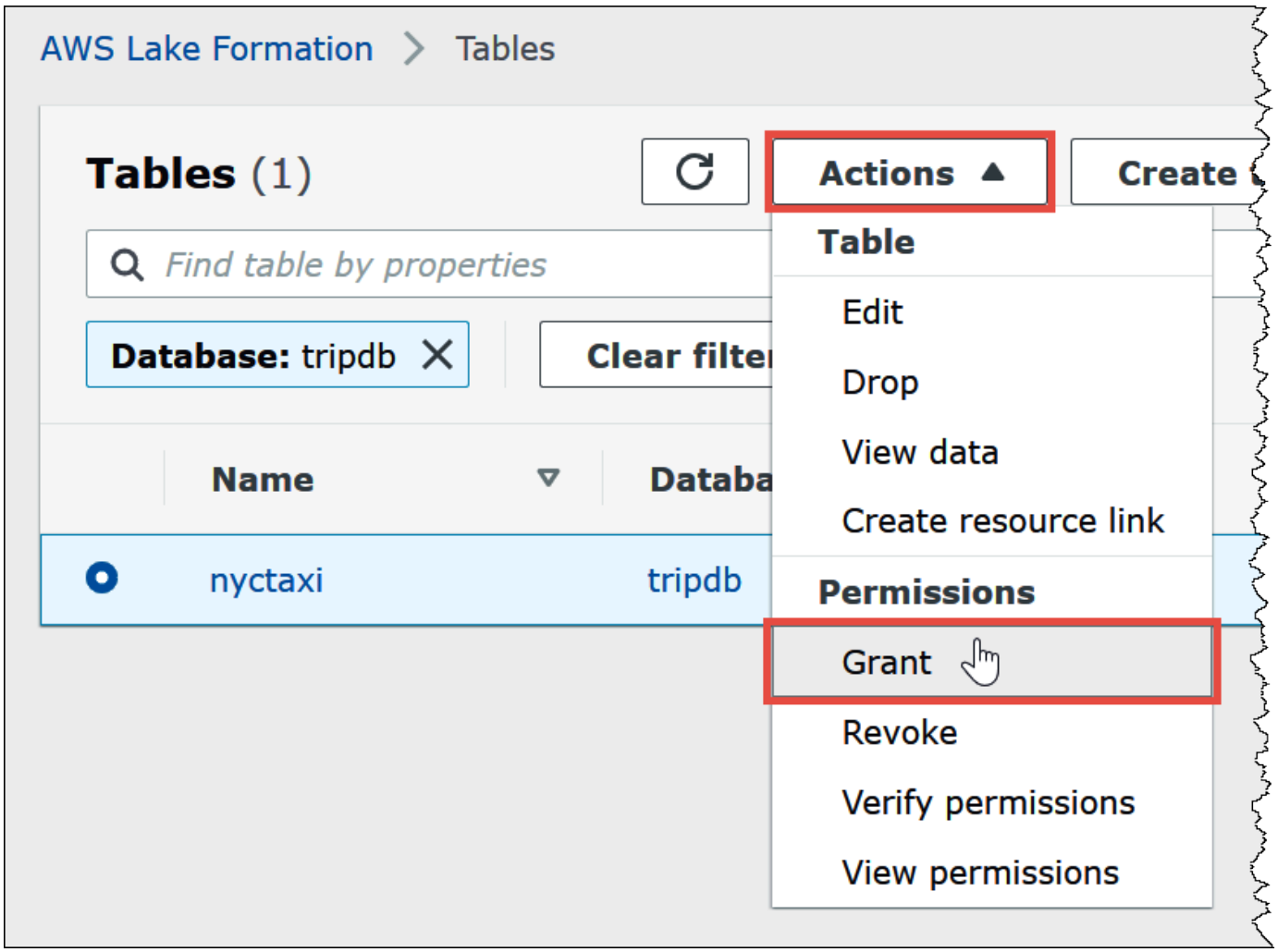
为 Okta 用户授予 Lake Formation 中的权限

1. 以数据湖管理员身份登录 AWS Management Console。

2. 打开 Lake Formation 控制台，网址为 https://console.aws.amazon.com/lakeformation/。
3. 从导航窗格中，选择 Tables（表），然后选择您要授予权限的表。本教程使用 `tripdb` 数据库中的 `nyctaxi` 表。



4. 从 Actions（操作）中，选择 Grant（授权）。



5. 在 Grant permissions（授予权限）对话框中，输入以下信息：

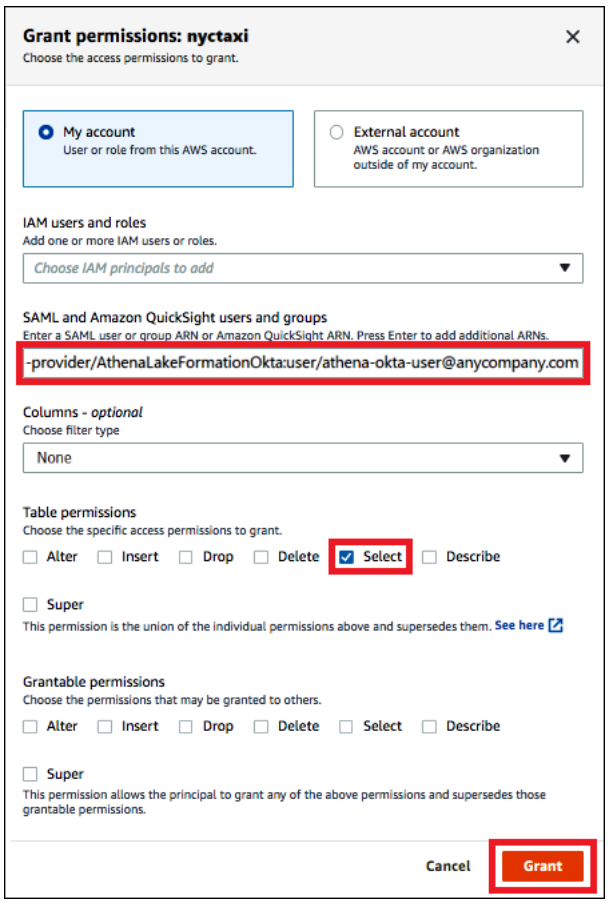
   a. 在 SAML and Amazon QuickSight users and groups（SAML 和 Amazon QuickSight 用户和组）下，按以下格式输入 Okta SAML 用户 ARN：

   ```
   arn:aws:iam::<account-id>:saml-provider/AthenaLakeFormationOkta:user/<athena-okta-user>@<anycompany.com>
   ```

   b. 对于 Columns（列），对于 Choose filter type（选择筛选条件类型），然后选择 Include columns（包括列）或者 Exclude columns（排除列）。

   c. 使用筛选条件下的 Choose one or more columns（选择一个或多个列）下拉列表指定要包含或排除的用户的列。

   d. 对于 Table permissions（表权限），选择 Select（选择）。本教程仅授予 SELECT 权限；您的要求可能会有所不同。



6. 选择 Grant（授权）。

现在，您需要对 Okta 组执行类似的步骤。

要授予 Okta 组在 Lake Formation 中的权限

1. 在 Lake Formation 控制台的 Tables（表）页面，请确保仍选择了 nyctaxi表。
2. 从 Actions（操作）中，选择 Grant（授权）。
3. 在 Grant permissions（授予权限）对话框中，输入以下信息：

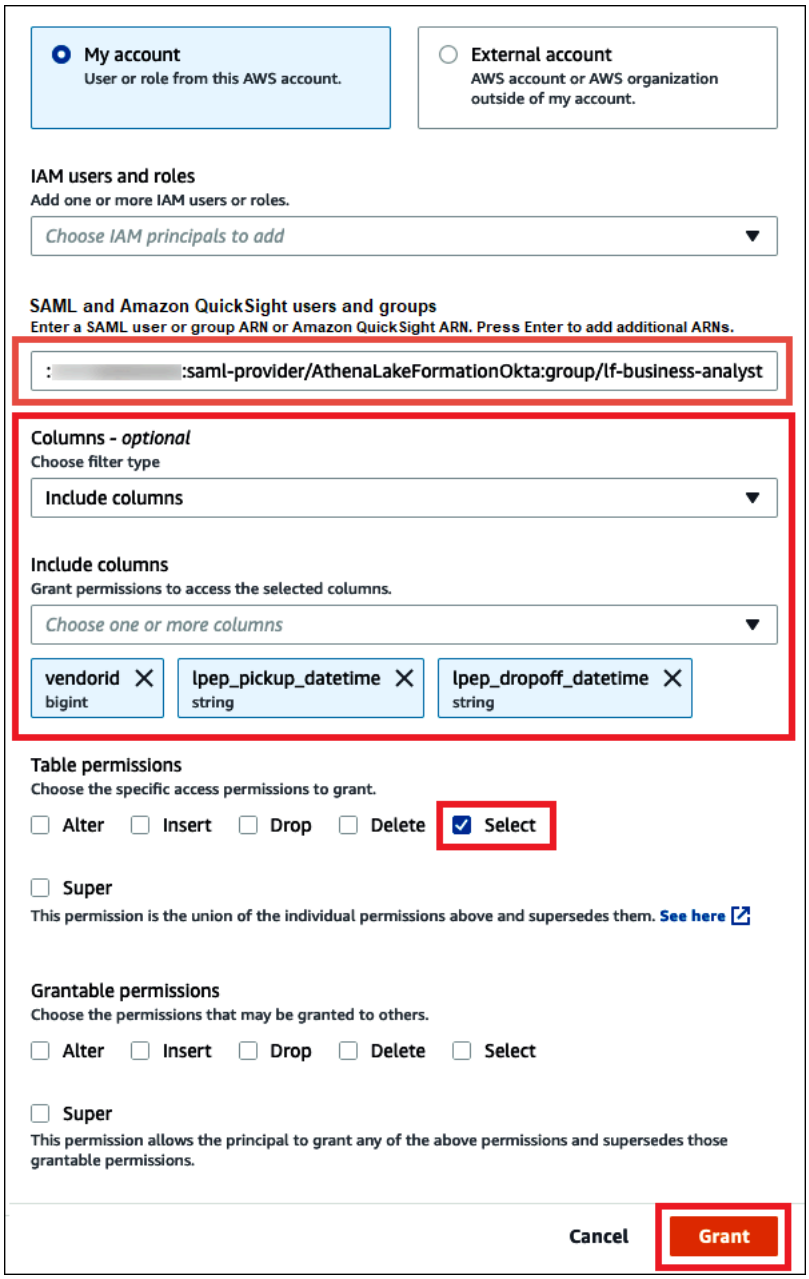
   a. 在 SAML and Amazon QuickSight users and groups（SAML 和 Amazon QuickSight 用户和组）下，按以下格式输入 Okta SAML 组 ARN：

   ```
   arn:aws:iam::<account-id>:saml-provider/AthenaLakeFormationOkta:group/lf-business-analyst
   ```

   b. 对于 Columns（列）、Choose filter type（选择筛选条件类型）中，选择 Include columns（包括列）。

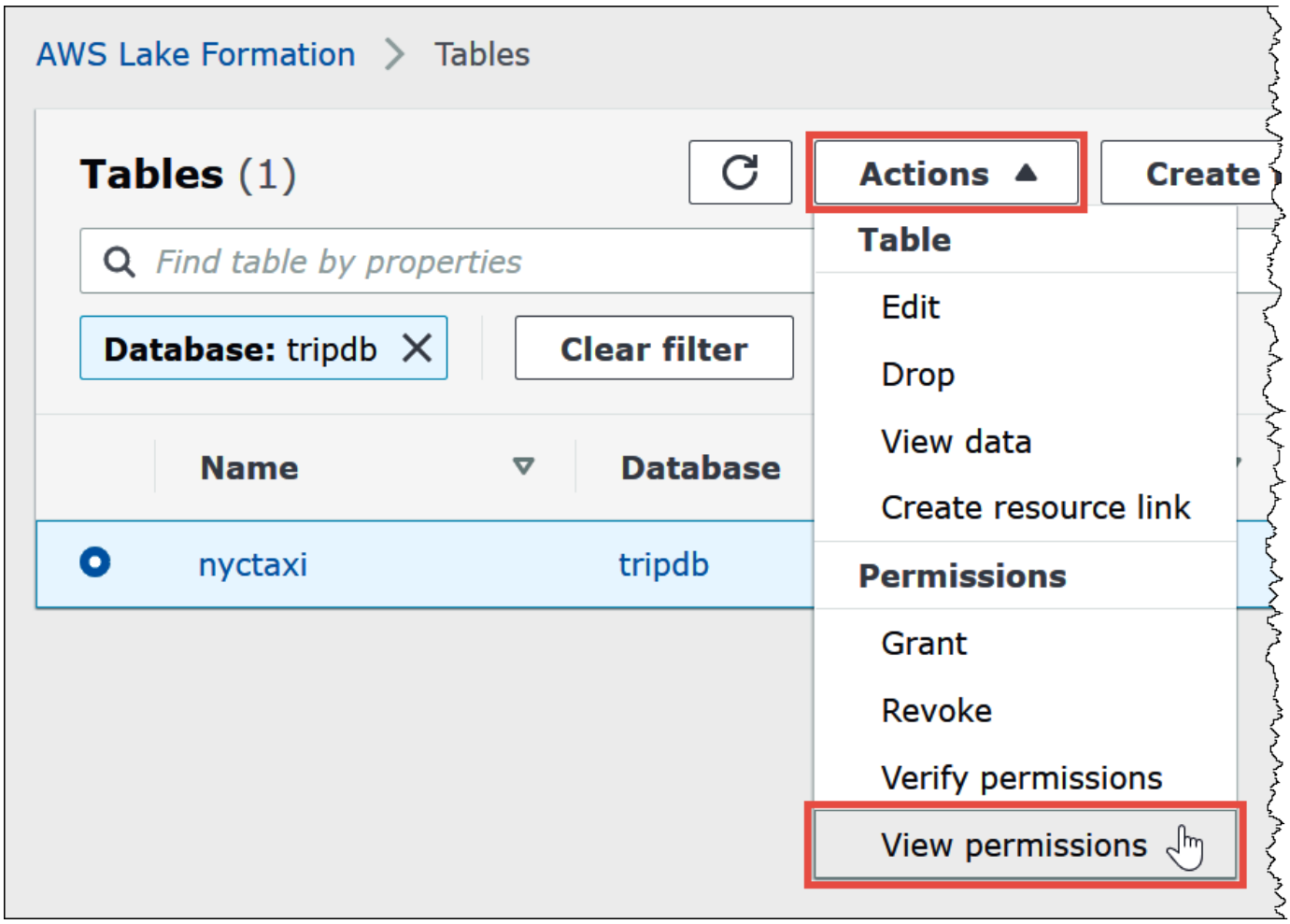
c. 对于 Choose one or more columns（选择一个或多个列），选择表的前三列。

d. 对于 Table permissions（表权限），选择要授予的特定访问权限。本教程仅授予 SELECT 权限；您的要求可能会有所不同。

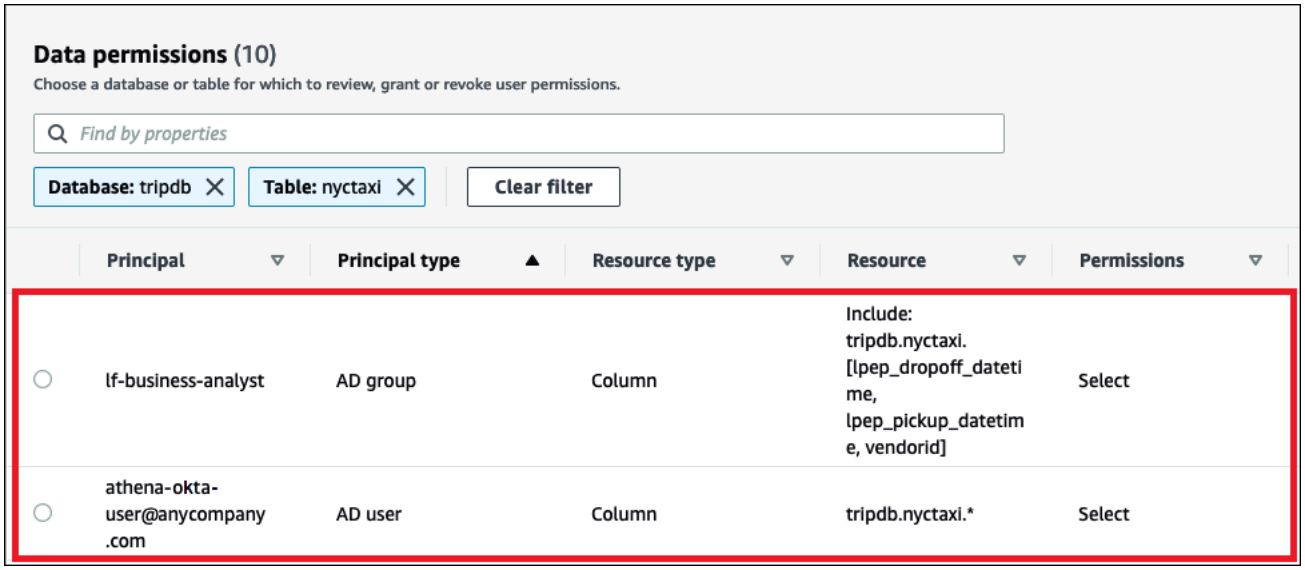


4. 选择 Grant（授权）。

5. 要验证您授予的权限，请选择 Actions（操作）、View permissions（查看权限）。



nyctaxi 表的 Data permissions（数据权限）页面上显示了 athena-okta-user 和 lf-business-analyst 组的权限。



步骤 7：验证通过 Athena JDBC 客户端的访问

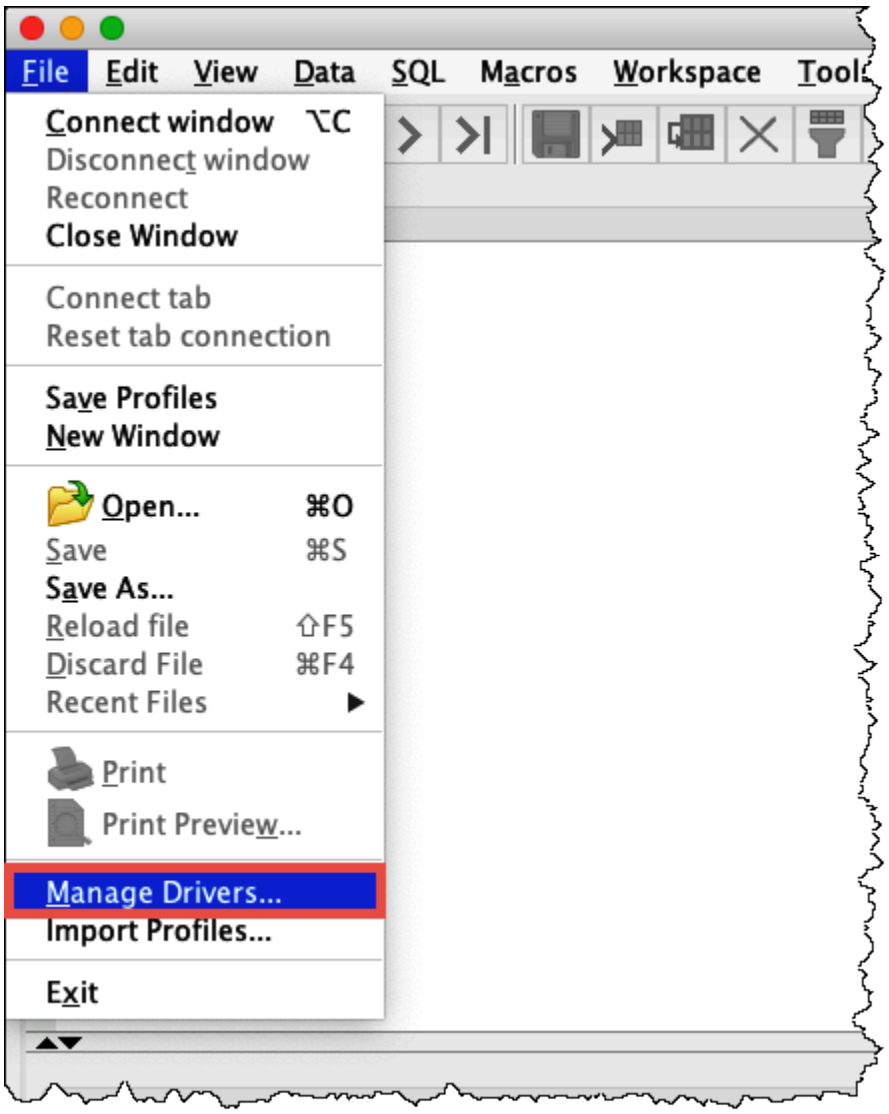
现在，您可以使用 JDBC 客户端以 Okta SAML 用户的身份执行与 Athena 的测试连接。

在本节中，您将执行以下任务：

- 准备测试客户端 – 下载 Athena JDBC 驱动程序，安装 SQL Workbench，然后将驱动程序添加到 Workbench。本教程使用 SQL Workbench 通过 Okta 身份验证访问 Athena，并验证 Lake Formation 权限。
- 在 SQL Workbench 中：
  - 为 Athena Okta 用户创建连接。
  - 以 Athena Okta 用户身份运行测试查询。
  - 为业务分析师用户创建和测试连接。
- 在 Okta 控制台中，将业务分析师用户添加到开发人员组。
- 在 Lake Formation 控制台中，为开发人员组配置表权限。
- 在 SQL Workbench 中，以业务分析师用户身份运行测试查询，并验证权限更改如何影响结果。

要准备测试客户端

1. 从 通过 JDBC 连接到 Amazon Athena 下载并提取 Lake Formation 兼容 Athena JDBC 驱动程序（2.0.14 或更高版本）。
2. 下载并安装免费的 SQL Workbench/J SQL 查询工具，在已修改的 Apache 2.0 许可证下可用。
3. 在 SQL Workbench 中，选择 File（文件），然后选择 Manage Drivers（管理驱动程序）。



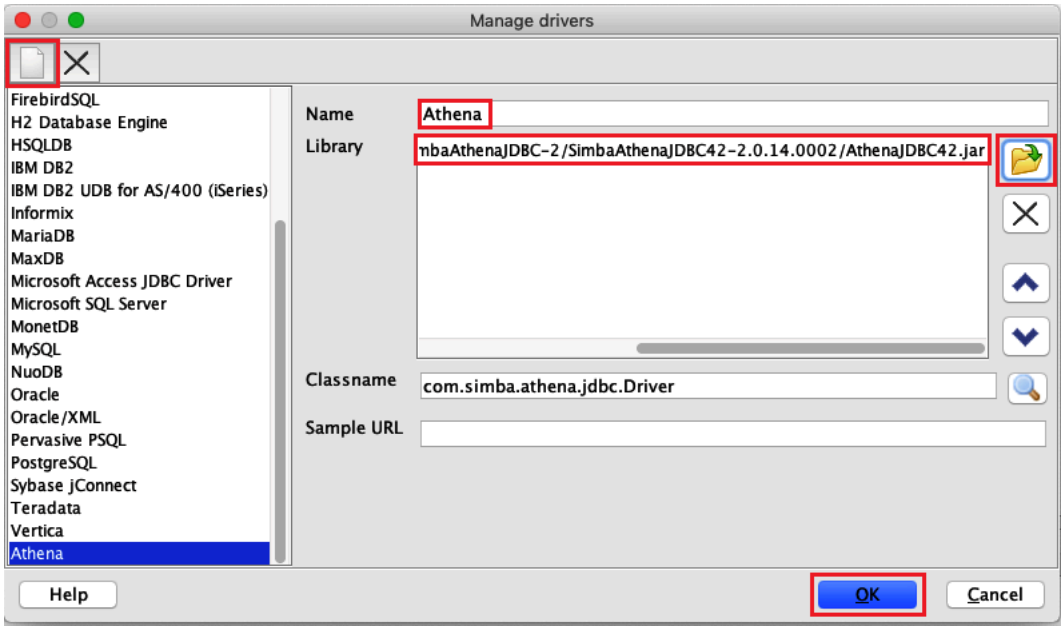
4. 在 Manage Drivers（管理驱动程序）对话框中，执行以下步骤：

   a. 选择新驱动程序图标。

   b. 对于名称，请输入 **`Athena`**。

   c. 对于 Library（库），浏览并选择您刚下载的 Simba Athena JDBC `.jar` 文件。

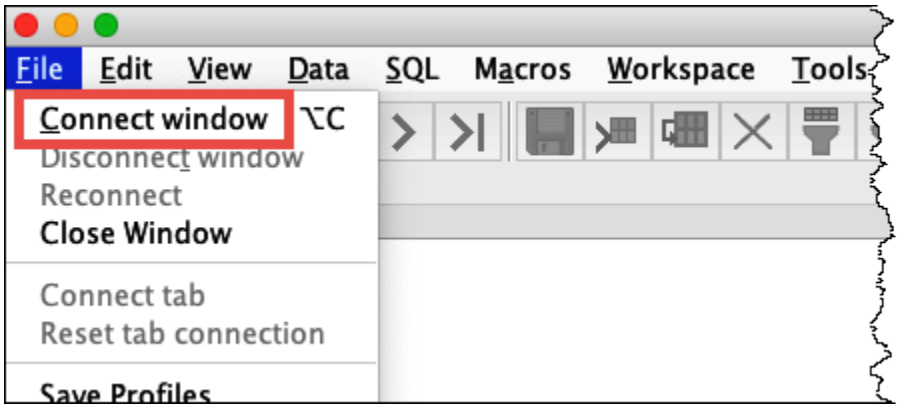
   d. 选择确定。



现在，您已准备就绪，可以为 Athena Okta 用户创建和测试连接。

要为 Okta 用户创建连接

1. 选择 File（文件）、Connect window（连接窗口）。



2. 在 Connection profile（连接配置文件）对话框中，通过输入以下信息创建连接：

   - 在名称框中，输入 **`Athena_Okta_User_Connection`**。
   - 对于 Driver（驱动程序），选择 Simba Athena JDBC 驱动程序。
   - 对于 URL，请执行以下操作之一：
     - 要使用连接 URL，请输入单行连接字符串。为便于阅读，以下示例添加了换行符。

```
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;
```

- 要使用基于 AWS 配置文件的 URL，请执行以下步骤：

  1. 配置具有 AWS 凭证文件的 AWS 配置文件，如下例所示。

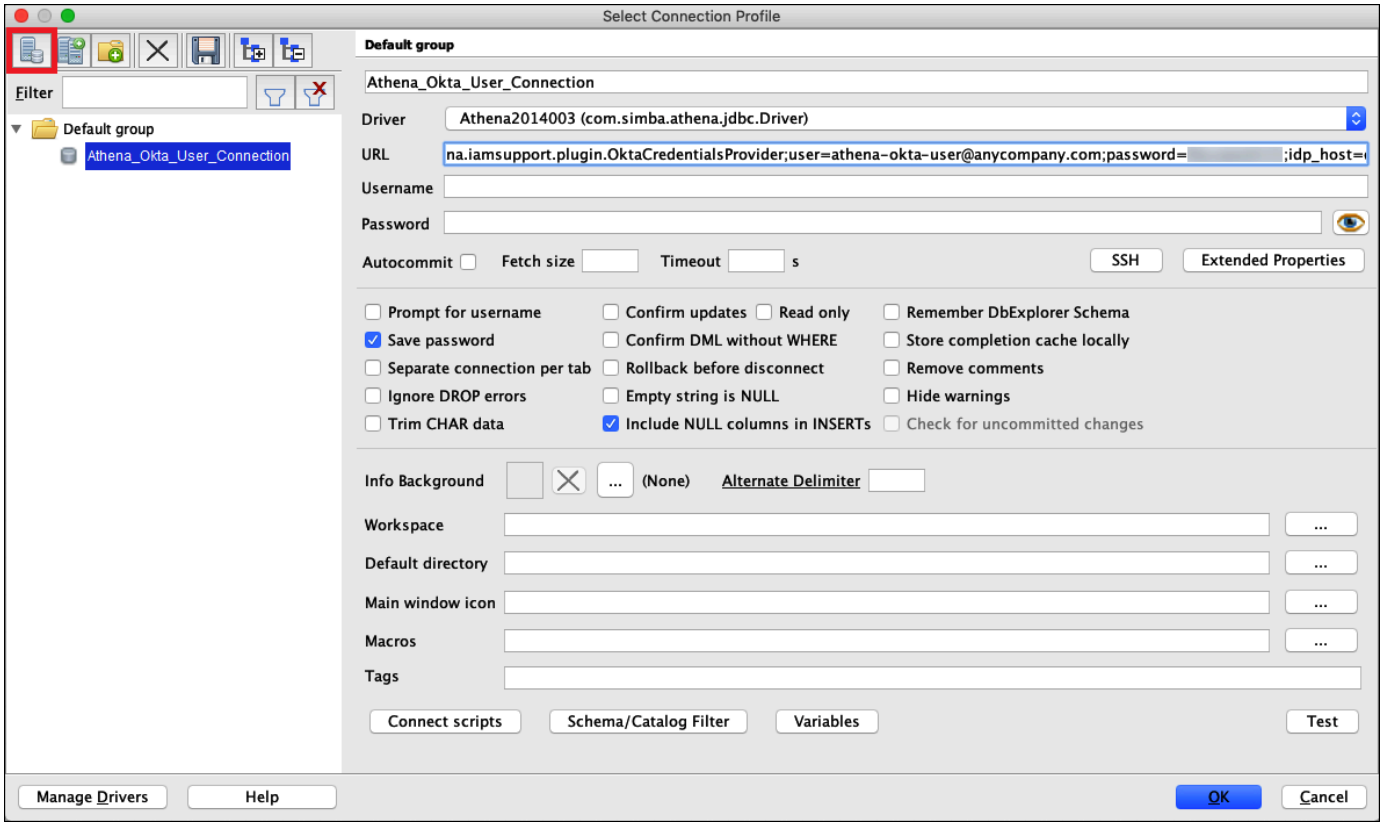
     ```
     [athena_lf_dev]
     plugin_name=com.simba.athena.iamsupport.plugin.OktaCredentialsProvider
     idp_host=okta-idp-domain
     app_id=okta-app-id
     uid=athena-okta-user@anycompany.com
     pwd=password
     ```

  2. 对于 URL，输入单行连接字符串，如下例所示。为便于阅读，这些示例添加了换行符。

     ```
     jdbc:awsathena://AwsRegion=region-id;
     S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/athena_results;
     profile=athena_lf_dev;
     SSL_Insecure=true;
     LakeFormationEnabled=true;
     ```

  请注意，这些示例是连接到 Athena 所需 URL 的基本表示形式。有关 URL 支持的参数的完整列表，请参阅 JDBC 文档。

下图显示了使用连接 URL 的 SQL Workbench 连接配置文件。

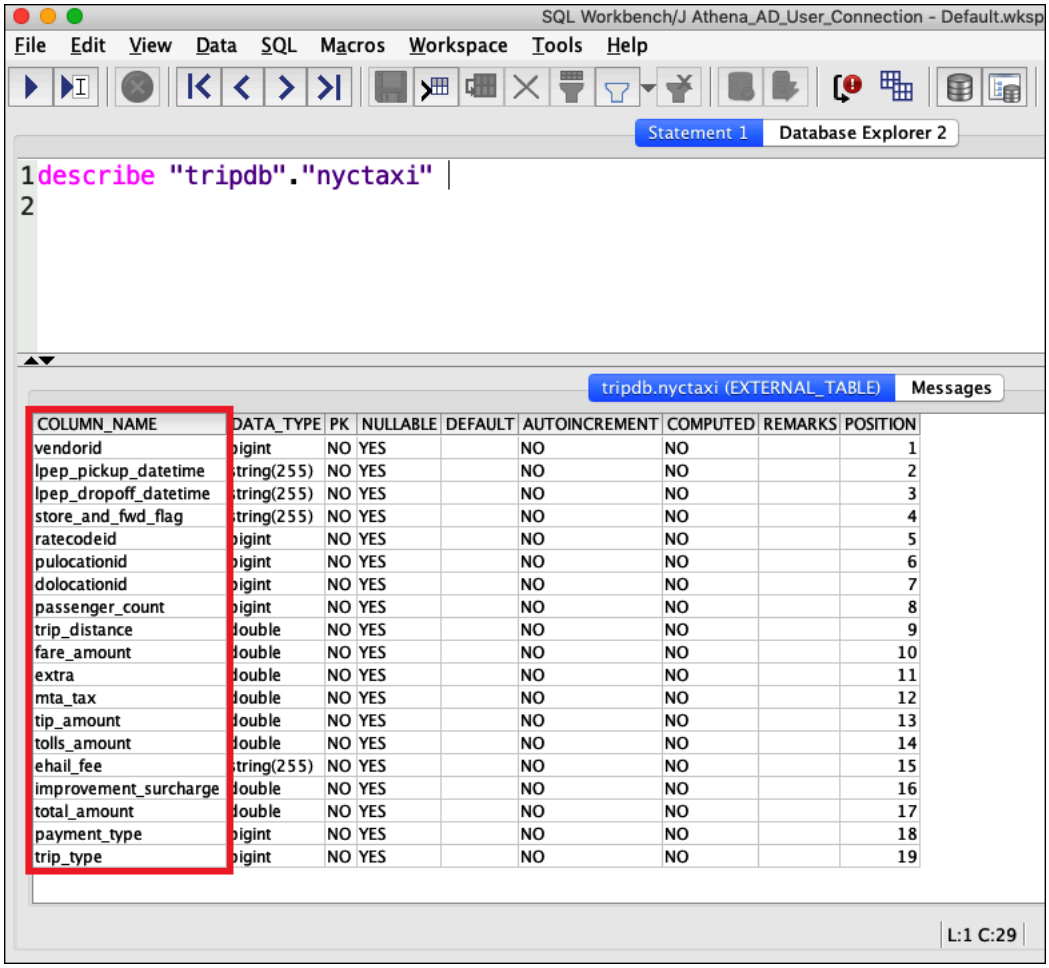


现在，您已为 Okta 用户建立了连接，您可以通过检索某些数据对其进行测试。

要测试 Okta 用户的连接

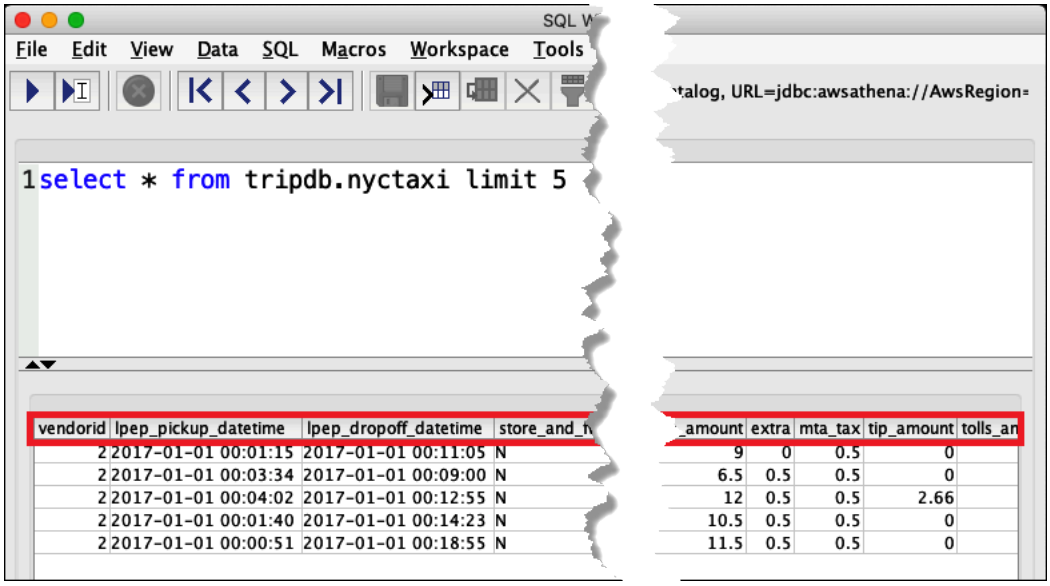
1. 选择 Test（测试），然后验证连接是否成功。
2. 从 SQL Workbench 的 Statement（语句）窗口中，运行以下 SQL DESCRIBE 命令。验证是否显示了所有列。

```
DESCRIBE "tripdb"."nyctaxi"
```



3. 从 SQL Workbench 的 Statement（语句）窗口中，运行以下 SQL SELECT 命令。验证是否显示了所有列。

```
SELECT * FROM tripdb.nyctaxi LIMIT 5
```



接下来，您将验证 athena-ba-user（作为 lf-business-analyst 组的一部分）是否只能访问您之前在 Lake Formation 成中指定的表的前三列。

要验证对 athena-ba-user 的访问

1. 在 SQL Workbench 中的 Connection profile（连接配置文件）对话框中，创建另一个连接配置文件。

   - 对于连接配置文件名称，输入 **Athena_Okta_Group_Connection**。
   - 对于 Driver（驱动程序），选择 Simba Athena JDBC 驱动程序。
   - 对于 URL，请执行以下操作之一：
     - 要使用连接 URL，请输入单行连接字符串。为便于阅读，以下示例添加了换行符。

```
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;
```

     - 要使用基于 AWS 配置文件的 URL，请执行以下步骤：

1. 配置 AWS 配置文件，其具有类似于以下示例的凭证文件。

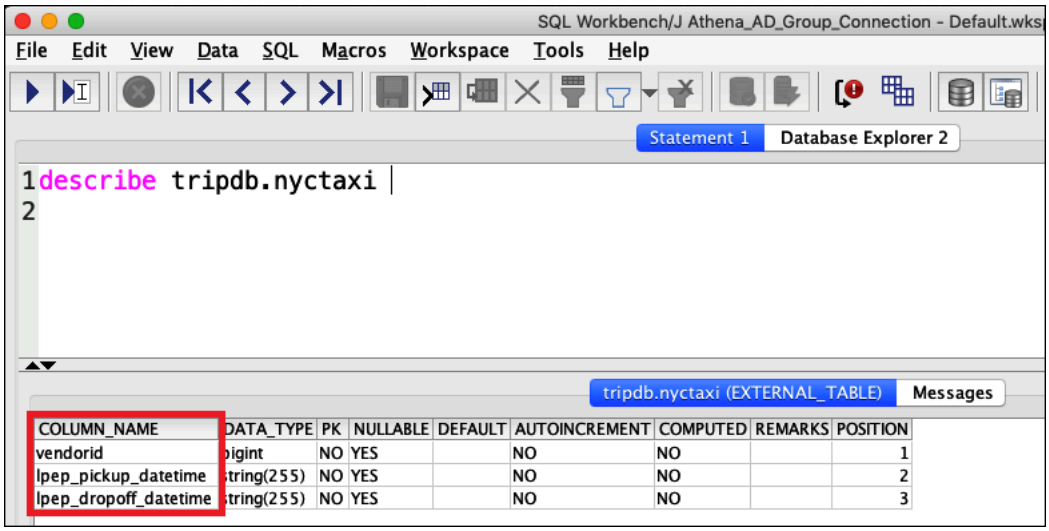
```
[athena_lf_ba]
plugin_name=com.simba.athena.iamsupport.plugin.OktaCredentialsProvider
idp_host=okta-idp-domain
app_id=okta-application-id
uid=athena-ba-user@anycompany.com
pwd=password
```

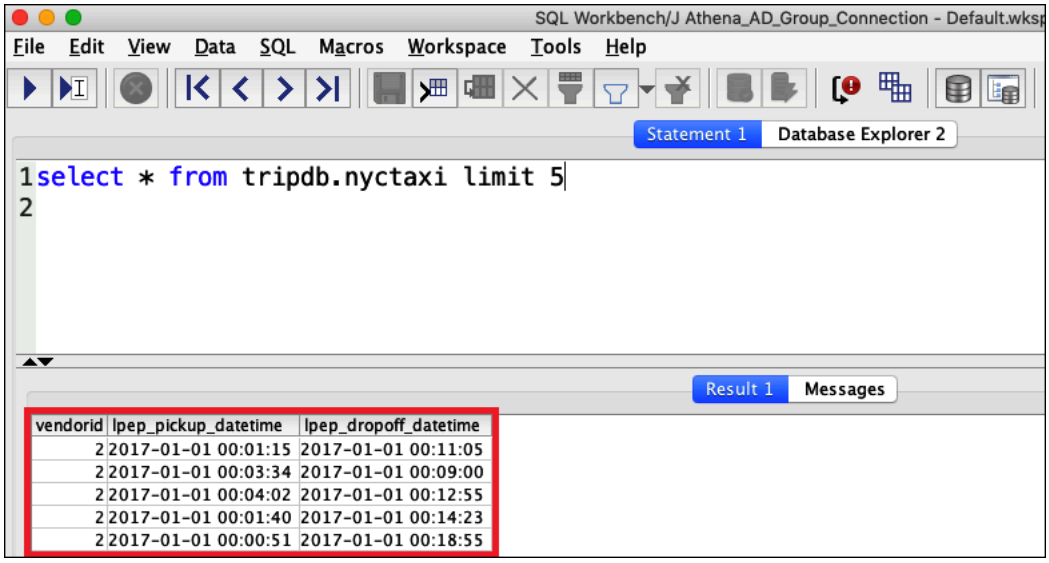
2. 对于 URL，输入如下所示的单行连接字符串。为便于阅读，这些示例添加了换行符。

```
jdbc:awsathena://AwsRegion=region-id;
S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/athena_results;
profile=athena_lf_ba;
SSL_Insecure=true;
LakeFormationEnabled=true;
```

2. 选择 Test（测试）以确认连接是否成功。
3. 从 SQL Statement（SQL 语句）窗口中，如您之前的操作那样运行相同的 DESCRIBE 和 SELECT SQL 命令并检查结果。

   由于 athena-ba-user 是 lf-business-analyst 组中的成员，将仅返回您在 Lake Formation 控制台中指定的前三列。

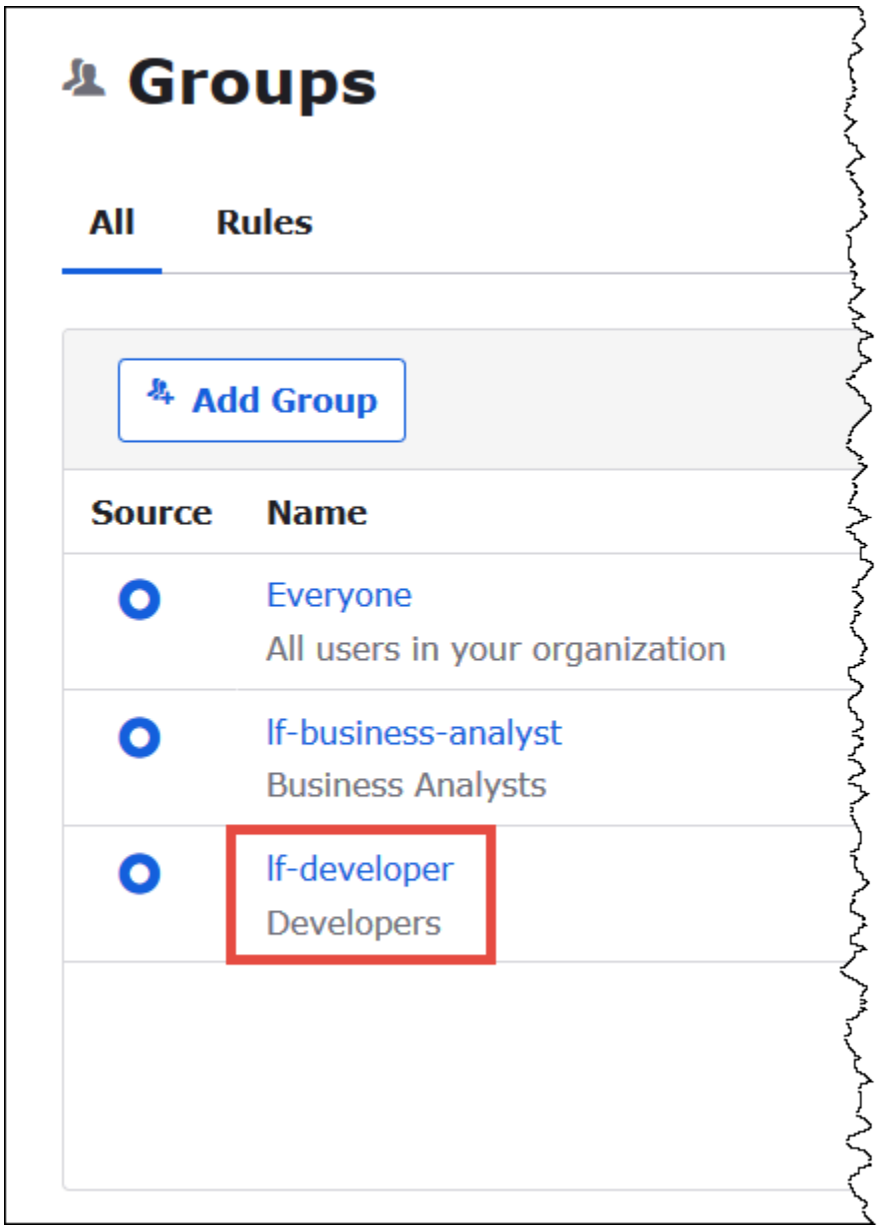




接下来，您需要返回到 Okta 控制台以将 `athena-ba-user` 添加到 `lf-developer` Okta 组。

要将 athena-ba-user 添加到 lf-developer 组

1. 以分配的 Okta 域的管理用户身份登录到 Okta 控制台。
2. 选择 Directory（目录），然后选择 Groups（组）。
3. 在组页面上，选择 lf-developer 组中。



4. 选择 Manage People（管理人员）。
5. 从 Not Members（非成员）列表中，选择 athena-ba-user，以将其添加到 lf-developer group（lf-developer 组）。
6. 选择保存。

现在，您将返回到 Lake Formation 控制台，以为 lf-developer 组配置表权限。

要为 lf-developer-group 配置表权限

1. 以数据湖管理员身份登录 Lake Formation 控制台。

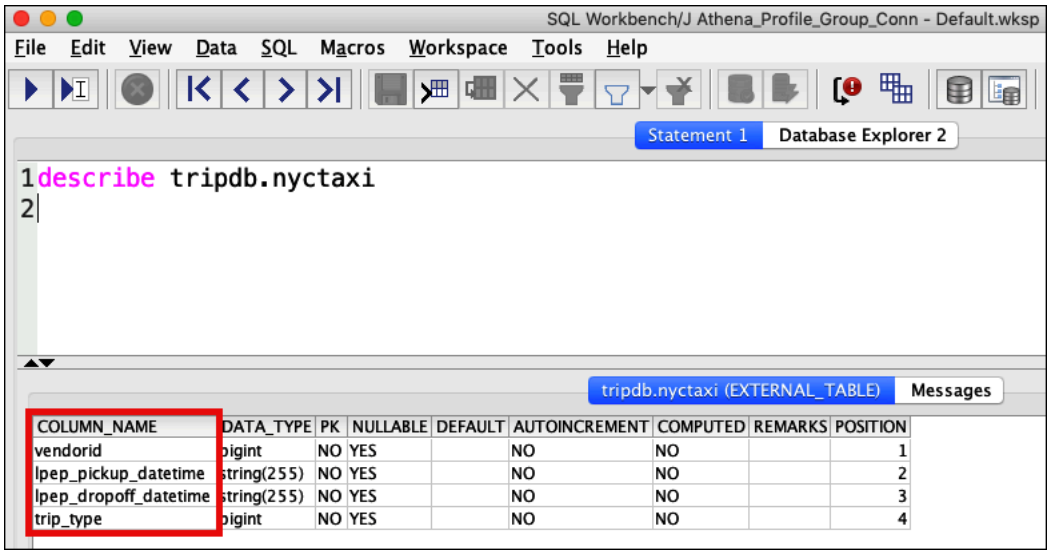
2. 在导航窗格中，选择表。
3. 选择 nyctaxi 表。
4. 选择 Actions（操作）、Grant（授权）。
5. 在 Grant Permissions（授予权限）对话框中，输入以下信息：
   - 对于 SAML and Amazon QuickSight users and groups（SAML 和 Amazon QuickSight 用户和组），按以下格式输入 Okta SAML lf-developer 组 ARN：
   - 对于 Columns（列）、Choose filter type（选择筛选条件类型）中，选择 Include columns（包括列）。
   - 选择 trip_type 列。
   - 对于 Table permissions（表权限），选择 SELECT（选择）。
6. 选择 Grant（授权）。

现在，您可以使用 SQL Workbench 验证 lf-developer 组的权限变更了。更改应反映在可用于 athena-ba-user 的数据中，该用户现在是 lf-developer 组的成员。

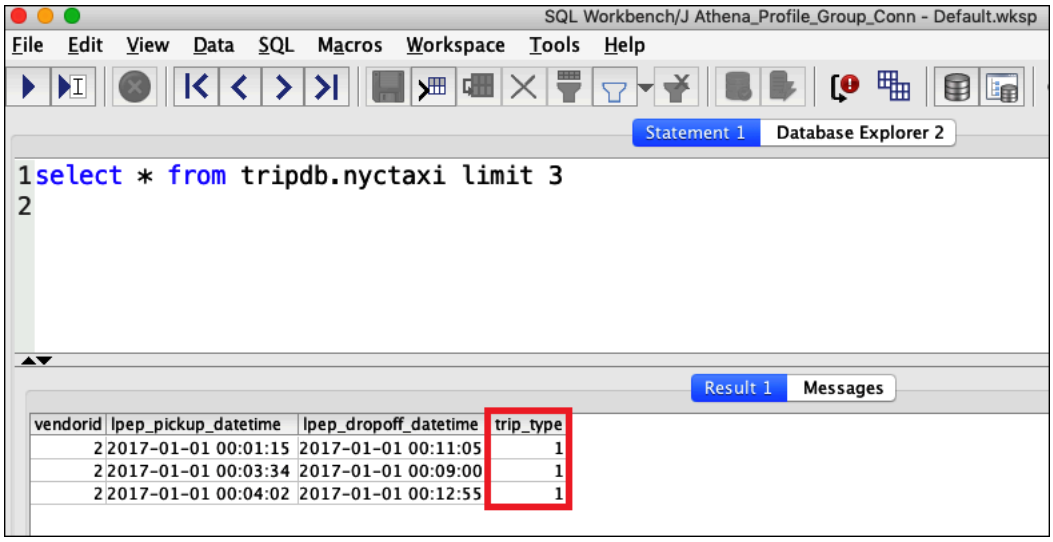
要验证 athena-ba-user 用户的权限更改

1. 关闭 SQL Workbench 程序，然后重新打开。
2. 连接到 athena-ba-user 的配置文件。
3. 从 Statement（语句）窗口中，发布与之前运行的相同 SQL 语句：

   这一次，将显示 trip_type 列。



由于 athena-ba-user 现在同时是 lf-developer 和 lf-business-analyst 组中的成员，这些组的 Lake Formation 权限组合将决定返回的列。



结论

在本教程中，您使用 Okta 作为 SAML 提供程序配置了 Athena 与 AWS Lake Formation 的整合。您使用 Lake Formation 和 IAM 控制数据湖 AWS Glue Data Catalog 中 SAML 用户可用的资源。

相关资源

有关信息，请参阅以下资源：

- 通过 JDBC 连接到 Amazon Athena
- 启用对 Athena API 的联合身份访问
- 《AWS Lake Formation 开发人员指南》
- 《AWS Lake Formation 开发人员指南》中的 授予和撤消数据目录权限。
- 《IAM 用户指南》中的 身份提供程序和联合服务。
- 《IAM 用户指南》中的 创建 IAM SAML 身份提供程序。
- AWS 安全博客上的 使用 Windows Active Directory、ADFS 和 SAML 2.0 启用对 AWS 的联合查询。

工作负载管理

您可以使用 Athena 的工作组、容量管理、性能调整、压缩支持、标签和服务限额功能来管理您的工作负载。

主题

使用工作组控制查询访问和成本

可使用工作组分隔用户、团队、应用程序或工作负载，对每个查询或整个工作组可处理的数据量设置限制，以及跟踪成本。由于工作组可作为资源，因此您可以使用基于资源级身份的策略来控制对特定工作组的访问。您还可以在 Amazon CloudWatch 中查看与查询相关的指标、通过配置扫描的数据量限制来控制成本、创建阈值以及在突破这些阈值时触发操作，例如 Amazon SNS。

为了进一步控制成本，您可以使用您指定的数据处理单元数量创建容量预留，然后向预留中添加一个或多个工作组。有关更多信息，请参阅 管理查询处理容量。

工作组与 IAM、CloudWatch、Amazon Simple Notification Service 和 AWS Cost and Usage Reports 集成，如下所示：

- 具有资源级权限的基于 IAM 身份的策略控制谁可以在工作组中运行查询。
- 如果您启用查询指标，Athena 会将工作组查询指标发送到 CloudWatch。
- 在 Amazon SNS 中，您可以创建 Amazon SNS 主题，用于在工作组中查询的数据使用控制超过您建立的阈值时向指定工作组用户发出警报。
- 当您在 Billing and Cost Management（账单和成本管理）控制台中使用配置为成本分配标签的标签来标记工作组时，与在该工作组中运行查询相关的费用会显示在带有该成本分配标签的“成本和使用情况报告”中。

主题

- 使用工作组运行查询
- 使用 CloudWatch 指标和事件控制成本和监控查询

另请参见 AWS 大数据博客文章：使用 Amazon Athena 工作组分离查询并管理成本，其中展示了如何使用工作组分离工作负载、控制用户访问权限以及管理查询使用情况和成本。

使用工作组运行查询

我们建议使用工作组来隔离团队、应用程序或不同工作负载的查询。例如，您可以为组织中的两个不同团队创建单独工作组。您还可以分隔工作负载。例如，您可以创建两个独立工作组，一个用于报告生成等自动计划的应用程序，另一个供分析师临时使用。您可以在工作组之间进行切换。

主题

- 使用工作组的优势
- 工作组的工作原理
- 设置工作组
- 用于访问工作组的 IAM policy
- 工作组设置
- 管理工作组
- 使用已启用 IAM Identity Center 的 Athena 工作组
- Athena 工作组 API
- 工作组故障排除

使用工作组的优势

使用工作组可以：

使用组隔离用户、团队、应用程序或工作负载。	每个工作组都有自己的不同查询历史记录和已保存查询的列表。有关更多信息，请参阅 工作组的工作原理。 您可以选择为工作组中的所有查询配置工作组设置。其中包括存储查询结果的 Amazon S3 位置、预期存储桶拥有者、加密，以及对写入查询结果存储桶的对象的控制权。您还可以强制实施工作组设置。有关更多信息，请参阅 工作组设置。

强制实施成本约束。	您可以为工作组中的查询设置两种类型的成本约束： • 各查询限制是为每个查询扫描的数据量设置的阈值。当查询超过指定阈值时，Athena 将取消查询。该限制适用于工作组中每一个正在运行的查询。您可以只设置一个查询限制，并在需要时更新它。 • Per-workgroup limit (工作组限制) 是一个您可以为工作组设置的阈值，用于限制为工作组中的查询扫描的数据量。超出阈值会激活 Amazon SNS 警报告警，它会触发您选择的操作，如给指定用户发送电子邮件。您可以为每个工作组设置多个工作组限制。 有关详细步骤，请参阅设置数据使用控制限制。
在 CloudWatch 中跟踪所有工作组查询的查询相关指标。	对于使用工作组运行的每个查询，如果您将工作组配置为发布指标，Athena 会将其发布到 CloudWatch。您可以查看查询指标，用于 Athena 控制台中的每个工作组。在 CloudWatch 中，您可以创建自定义控制面板，并为这些指标设置阈值和警报。

工作组的工作原理

Athena 中的工作组具有下列特征：

- 在默认情况下，每个账户都有一个主工作组，且默认权限为允许所有经过身份验证的用户访问此工作组。不能删除主工作组。
- 您创建的每个工作组仅为其中运行的查询显示已保存的查询和查询历史记录，而不会为账户中的所有查询显示这些信息。由此可以将您的查询与账户中的其他查询分离开，让您能够更高效地找到您自己的已保存查询和历史记录中的查询。
- 禁用工作组可防止在其中运行查询，直到您启用它。发送到已禁用工作组的查询会失败，直到您再次启用它。
- 如果您具有权限，则可以删除空工作组以及包含已保存查询的工作组。在这种情况下，在删除工作组之前，Athena 会警告您保存的查询将被删除。在删除其他用户有权访问的工作组之前，确保其用户有权访问可以从中继续运行查询的其他工作组。
- 您可以设置工作组范围的设置，并强制工作组中运行的所有查询使用它们。这些设置包括 Amazon S3 中的查询结果位置、预期存储桶拥有者、加密，以及对写入查询结果存储桶的对象的控制权。

> **Important**
>
> 在强制实施工作组范围的设置时，该工作组中运行的所有查询都使用工作组设置。即使它们的客户端设置可能与工作组设置不同，也仍然如此。有关信息，请参阅工作组设置覆盖客户端设置。

工作组限制

- 您在自己的账户中对每个区域最多可创建 1000 个工作组。
- 不能删除主工作组。
- 在每个工作组中，您可以打开最多十个查询选项卡。当您在工作组之间切换时，最多可将三个工作组中的查询选项卡保持打开。

设置工作组

设置工作组包括创建工作组并为其建立使用权限。首先确定您的组织需要哪些工作组，然后创建它们。然后，设置 IAM 工作组策略以控制 `workgroup` 资源的用户访问权限和操作。此时，可访问这些工作组的用户可以运行其中的查询。

> **Note**
>
> 在第一次开始使用工作组时，使用这些任务设置它们。如果您的 Athena 账户已使用工作组，则每个账户的用户需要相应权限，才能运行该账户中一个或多个工作组中的查询。在运行查询之前，检查 IAM policy 以查看您可以访问的工作组，在需要时调整策略并切换到您要使用的工作组。

默认情况下，如果您尚未创建任何工作组，则账户中的所有查询在主工作组中运行。

Athena 将在控制台右上角显示 Workgroup（工作组）选项中的当前工作组。您可以使用此选项切换工作组。运行查询时，其会在该工作组中运行。您可以通过 API 操作、命令行界面或使用 JDBC 或 ODBC 驱动程序通过客户端应用程序在控制台中运行该工作组上下文中的查询。有权访问工作组时，可以查看工作组的设置、指标和数据使用控制限制。通过其他权限，您可以编辑设置和数据使用控制限制。

设置工作组

1. 确定要创建的工作组。例如，您可以确定以下几点：

   - 哪些用户可以运行每个工作组中的查询，以及哪些用户拥有工作组配置。这决定了您创建的 IAM policy。有关更多信息，请参阅 用于访问工作组的 IAM policy。
   - Amazon S3 中的哪些位置用于各个工作组中运行的查询的查询结果。Amazon S3 中必须存在对应位置，然后您才能为工作组查询结果指定它。使用工作组的所有用户必须能够访问此位置。有关更多信息，请参阅 工作组设置。
   - Amazon S3 查询结果存储桶的拥有者对写入存储桶的新对象是否拥有完全控制权。例如，假设您的查询结果位置属于其他账户所有，则可以将所有权以及对查询结果的完全控制权授予该其他账户。有关更多信息，请参阅 AclConfiguration。
   - 指定您希望其成为输出位置存储桶拥有者的 AWS 账户 ID。这是可选的附加安全措施。如果存储桶拥有者的账户 ID 与您在此处指定的 ID 不匹配，则输出到存储桶的尝试将失败。有关更多信息，请参阅《Amazon S3 用户指南》中的使用存储桶拥有者条件验证存储桶所有权。此设置不适用于 CTAS、INSERT INTO 或 UNLOAD 语句。
   - 需要哪些加密设置，以及哪些工作组中包含必须加密的查询。我们建议您为加密查询和未加密查询创建单独的工作组。由此可以强制实施工作组加密，并应用于其中运行的所有查询。有关更多信息，请参阅 加密在 Amazon S3 中存储的 Athena 查询结果。

2. 根据需要创建工作组，并为它们添加标签。要查看步骤，请参阅 创建工作组。
3. 为您的用户、组或角色创建 IAM policy，使他们能够访问工作组。这些策略建立工作组成员资格，以及对`workgroup`资源相关操作的访问权限。有关详细步骤，请参阅 用于访问工作组的 IAM policy。有关示例 JSON 策略，请参阅对工作组和标签的访问。
4. 设置工作组设置。指定一个用于存储查询结果的 Amazon S3 位置，此外还可选择指定预期存储桶拥有者、加密设置以及对写入查询结果存储桶的对象的控制权。您可以强制实施工作组设置。有关更多信息，请参阅工作组设置。

> **Important**
>
> 如果您覆盖客户端设置，Athena 将使用工作组的设置。这会影响您在控制台中、使用驱动程序、命令行界面或 API 操作运行的查询。
>
> 虽然查询会继续运行，但基于特定 Amazon S3 存储桶中结果的可用性构建的自动化可能中断。我们建议您先通知您的用户，然后再进行覆盖。在设置要覆盖工作组设置后，您可以省略在驱动程序或 API 中指定客户端设置的操作。

5. 通知用户哪些工作组将用于运行查询。发送电子邮件。就您账户中用户可使用的工作组、需要的 IAM policy 和工作组设置通知对应用户。
6. 配置查询和工作组的成本控制限制，也称为数据使用控制限制。如要在超出阈值时收到通知，请创建 Amazon SNS 主题并配置订阅。有关详细步骤，请参阅设置数据使用控制限制和《Amazon Simple Notification Service 开发人员指南》中的 Amazon SNS 入门。
7. 切换到工作组，以便您可以运行查询。要运行查询，请切换到相应的工作组。有关详细步骤，请参阅the section called “指定从中运行查询的工作组”。

用于访问工作组的 IAM policy

要控制对工作组的访问，使用资源级 IAM 权限或基于身份的 IAM policy。每当您使用 IAM policy 时，请确保遵循 IAM 最佳实践。有关更多信息，请参阅《IAM 用户指南》中的 IAM 安全最佳实践。

> **Note**
>
> 要访问启用了可信身份传播的工作组，必须将 IAM Identity Center 用户分配给由 Athena GetWorkGroup API 操作的响应返回的 `IdentityCenterApplicationArn`。

以下是 Athena 的特定过程。

有关 IAM 的特定信息，请参阅本节末尾列出的链接。有关示例 JSON 工作组策略的信息，请参阅工作组策略示例。

要在 IAM 控制台中使用可视化编辑器创建工作组策略

1. 登录 AWS Management Console，然后使用以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧的导航窗格中，选择 Policies (策略)，然后选择 Create policy (创建策略)。
3. 在可视化编辑器选项卡上，选择选择服务。然后，选择要添加到策略的 Athena。
4. 选择 Select actions (选择操作)，然后选择要添加到策略的操作。可视化编辑器会显示 Athena 中可用的操作。有关更多信息，请参阅《服务授权参考》中的 Amazon Athena 的操作、资源和条件键。
5. 选择 add actions (添加操作) 以键入特定操作，或使用通配符 (*) 指定多个操作。

   预设情况下，您创建的策略允许执行选择的操作。如果您在 Athena 中选择对 `workgroup` 资源执行一个或多个支持资源级权限的操作，则编辑器会列出 `workgroup` 资源。

6. 选择 Resources (资源)，为您的策略指定特定的工作组。有关示例 JSON 工作组策略，请参阅工作组策略示例。
7. 指定 workgroup 资源，如下所示：

```
arn:aws:athena:<region>:<user-account>:workgroup/<workgroup-name>
```

8. 选择 Review policy (查看策略)，然后为您创建的策略键入 Name (名称) 和 Description (描述) (可选)。查看策略摘要以确保您已授予所需的权限。
9. 选择创建策略可保存您的新策略。
10. 将此基于身份的策略附加到用户、组或角色。

有关详细信息，请参阅服务授权参考和《IAM 用户指南》中的以下主题：

- Amazon Athena 的操作、资源和条件键
- 使用可视化编辑器创建策略
- 添加和移除 IAM policy
- 控制对资源的访问

有关示例 JSON 工作组策略，请参阅工作组策略示例。

有关 Amazon Athena 操作的完整列表，请参阅 Amazon Athena API 参考中的 API 操作名称。

工作组策略示例

本节包含您可以用于启用对工作组执行各种操作的策略示例。每当您使用 IAM policy 时，请确保遵循 IAM 最佳实践。有关更多信息，请参阅《IAM 用户指南》中的 IAM 安全最佳实践。

工作组是由 Athena 管理的 IAM 资源。因此，如果您的工作组策略使用获取 workgroup 作为输入的操作，您必须如下所示指定工作组的 ARN：

```
"Resource": [arn:aws:athena:<region>:<user-account>:workgroup/<workgroup-name>]
```

其中，*<workgroup-name>* 是工作组的名称。例如，对于名为 test_workgroup 的工作组，指定它作为资源，如下所示：

```
"Resource": ["arn:aws:athena:us-east-1:123456789012:workgroup/test_workgroup"]
```

有关 Amazon Athena 操作的完整列表，请参阅 Amazon Athena API 参考中的 API 操作名称。有关 IAM policy 的更多信息，请参阅《IAM 用户指南》中的 使用可视化编辑器创建策略。有关为工作组创建 IAM policy 的更多信息，请参阅 用于访问工作组的 IAM policy。

- 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 对所有工作组具有完全访问权限的策略示例

以下策略可实现对账户中可能存在的所有工作组资源具有完全访问权限。对于必须为其他所有用户监督和管理工作组的那些用户，我们建议您使用此策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Example 对指定工作组具有完全访问权限的策略示例

以下策略可实现对单个名为 workgroupA 的特定工作组资源具有完全访问权限。对于对特定工作组具有完全控制权限的用户，您可以使用此策略。

```
{
    "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 运行指定工作组中查询的策略示例

在以下策略中，允许用户运行指定 workgroupA 中的查询并查看它们。不允许用户对工作组本身执行管理任务，例如，更新或删除它。

```
{
    "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 运行主工作组中查询的策略示例

您可以修改上述示例以允许特定用户在主工作组中运行查询。

> **Note**
>
> 对于另行配置为运行指定工作组中查询的所有用户，我们建议您为其添加主工作组资源。在用户的指定工作组已被删除或禁用的情况下，在工作组用户策略中添加此策略资源有用。在这种情况下，他们可以继续运行主工作组中的查询。

要允许您账户中的用户运行主工作组中的查询，请将包含朱总做足 ARN 的行添加到 Example policy for running queries in a specified workgroup 的资源部分，如下例所示。

```
arn:aws:athena:us-east-1:123456789012:workgroup/primary"
```

Example 管理指定工作组操作的策略示例

在以下策略中，允许用户创建、删除、获取详细信息并更新工作组的 `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 列出工作组的策略示例

以下策略允许所有用户列出所有工作组：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListWorkGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

Example 运行和停止特定工作组中查询的策略示例

在此策略中，允许用户运行工作组中的查询：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:StartQueryExecution",
                "athena:StopQueryExecution"
            ],
            "Resource": [
                "arn:aws:athena:us-east-1:123456789012:workgroup/test_workgroup"
            ]
        }
    ]
}
```

Example 使用特定工作组中的命名查询的策略示例

在以下策略中，用户具有在指定工作组中创建和删除命名查询以及获取其相关信息的权限：

```
{
    "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 在 Athena 中使用 Spark 笔记本的示例策略

使用以下策略在 Athena 中使用 Spark 笔记本。

```
{
    "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": "*"
        }
    ]
}
```

工作组设置

每个工作组具有以下设置：

- 唯一名称。它可以包含 1 到 128 个字符，包括字母数字字符、短划线和下划线。当您创建工作组后，便无法再更改其名称。但是，您可以创建一个具有相同设置但名称不同的新工作组。
- 设置适用于工作组中运行的所有查询。其中包括：
  - Amazon S3 中用于存储查询结果的位置，它用于该工作组中运行的所有查询。必须存在该位置，然后才能在创建工作组时为其指定对应位置。有关创建 Amazon S3 存储桶的信息，请参阅 创建存储桶。
  - 存储桶拥有者对查询结果的控制权 – Amazon S3 查询结果存储桶的拥有者对写入存储桶的新对象是否拥有完全控制权。例如，假设您的查询结果位置属于其他账户所有，则可以将所有权以及对查询结果的完全控制权授予该其他账户。
  - 预期存储桶拥有者 – 此为您希望成为查询结果存储桶拥有者 AWS 账户 的 ID。这是附加安全措施。如果存储桶拥有者的账户 ID 与您在此处指定的 ID 不匹配，则输出到存储桶的尝试将失败。有关更多信息，请参阅《Amazon S3 用户指南》中的使用存储桶拥有者条件验证存储桶所有权。

    > **Note**
    >
    > 预期存储桶拥有者设置仅适用于您为 Athena 查询结果指定的 Amazon S3 输出位置。其不适用于其他 Amazon S3 位置，例如外部 Amazon S3 存储桶中的数据源位置、CTAS 和 INSERT INTO 目标表位置、UNLOAD 语句输出位置、为联合查询溢出存储桶的操作，或针对另一个账户中的表运行的 SELECT 查询。

  - 加密设置，前提是您为所有工作组查询使用加密。您只能将工作组的全部查询加密，而不能仅将其中一部分加密。最好创建单独的工作组，用于包含加密或未加密的查询。

另外，您的工作组可以覆盖客户端设置。在发布工作组之前，您可以将结果位置和加密选项指定为 JDBC 或 ODBC 驱动程序中的参数，或 Athena 控制台中的 Properties（属性）选项卡中设置。也可以通过 API 操作直接指定这些设置。这些设置称为“客户端设置”。对于工作组，您可以在工作组级别配置

这些设置，以控制客户端级别的可用选项。强制执行工作组级别的设置还可以使用户不必单独配置其客户端设置。如果您为工作组选择覆盖客户端设置选项，查询会使用工作组设置并忽略客户端设置。

如果已选择 Override Client-Side Settings (覆盖客户端设置)，控制台会通知用户他们的设置已更改。如果通过上述方式强制实施工作组设置，用户可忽略对应的客户端设置。然后，即使存在客户端设置，在控制台中运行的查询也会使用工作组的设置。此外，当通过 AWS CLI、API 操作或 JDBC 或 ODBC 驱动程序运行工作组中的查询时，工作组设置会覆盖查询结果位置和加密等客户端设置。要查看工作组的设置，请查看工作组详细信息。

您还可以为工作组中的查询设置查询限制。

工作组设置覆盖客户端设置

Create workgroup (创建工作组) 和 Edit workgroup (编辑工作组) 对话框中有一个标题为 Override client-side settings (覆盖客户端设置) 的字段。在默认情况下，该字段未被选中。根据您是否选择它，Athena 会执行以下操作：

- 如果覆盖客户端设置未被选中，则不会在客户端级别强制实施工作组设置。当未为工作组选择覆盖客户端设置选项时，Athena 会将客户端设置用于工作组中运行的所有查询，包括查询结果位置、预期存储桶所有者、加密和写入查询结果存储桶的对象控制的设置。每个用户都可以在控制台的设置菜单中指定自己的设置。如果未设置客户端设置，则会应用工作组范围设置。如果您使用 AWS CLI、API 操作或 JDBC 和 ODBC 驱动程序在不覆盖客户端设置的工作组中运行查询，则您的查询将使用您在查询中指定的设置。
- 如果覆盖客户端设置被选中，则会在工作组级别对工作组中的所有客户端强制实施工作组设置。当为工作组选择覆盖客户端设置选项时，Athena 会将工作组设置用于工作组中运行的所有查询，包括查询结果位置、预期存储桶所有者、加密和写入查询结果存储桶的对象控制的设置。工作组设置会覆盖您在使用控制台、API 操作或 JDBC 或 ODBC 驱动程序时为查询指定的任何客户端设置。

  如果您覆盖客户端设置，则您或任何工作组用户下一次打开 Athena 控制台时，Athena 会通知您该工作组中的查询使用工作组设置，并提示您确认此更改。

> **Important**
>
> 如果您使用 API 操作、AWS CLI 或 JDBC 和 ODBC 驱动程序在覆盖客户端设置的工作组中运行查询，请确保在查询中省略客户端设置或更新它们以匹配工作组的设置。如果您在查询中指定了客户端设置，但在覆盖设置的工作组中运行这些设置，则查询将运行，但将使用工作组设置。有关查看工作组设置的信息，请参阅 查看工作组详细信息。

管理工作组

在 https://console.aws.amazon.com/athena/，您可以执行以下任务：

语句	描述
创建工作组	创建新的工作组
编辑工作组	编辑工作组并更改其设置。您不能更改工作组的名称，但您可以创建一个具有相同设置但名称不同的新工作组。
查看工作组详细信息	查看工作组的详细信息，例如其名称、描述、数据使用限制、查询结果位置、预期查询结果存储桶拥有者、加密，以及对写入查询结果存储桶的对象的控制权。如果 Override client-side settings (覆盖客户端设置) 已被选中，您还可以验证该工作组是否已强制实施其设置。
删除工作组	删除工作组。如果您删除某个工作组，则查询历史记录、已保存查询、工作组设置和查询数据限制控制都将被删除。工作组范围的数据限制控制会保留在 CloudWatch 中，您可以单独删除它们。 不能删除主工作组。
切换工作组	在您有权访问的工作组之间切换。
在工作组之间复制已保存的查询	在工作组之间复制已保存的查询。例如，如果您在预览工作组中创建了查询，并且希望在非预览工作组中使用该查询，则可能需要执行此操作。
启用和禁用工作组	启用或禁用工作组。当工作组处于禁用状态，其用户无法运行查询或创建新的命名查询。如果您有权访问它，您仍然可以查看指标、数据使用限制控制、工作组设置，查询历史记录和已保存查询。
指定从中运行查询的工作组	您必须为 Athena 指定要使用的工作组，然后才能运行查询。您必须具有工作组的权限。
创建使用 IAM Identity Center 身份验证的 Athena 工作组	要在 Athena 中使用 IAM Identity Center 身份，必须创建一个启用了 IAM Identity Center 的工作组。创建工作组后，您可以使用 IAM Identity Center 控制台或 API 将 IAM Identity Center 用户或组分配给工作组。

创建工作组

创建工作组需要具有执行 CreateWorkgroup API 操作的权限。请参阅对工作组和标签的访问和 用于访问工作组的 IAM policy。如果要添加标签，您还需要添加对 TagResource 的权限。请参阅 工作组的标签策略示例。

在控制台中创建工作组

1. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。

2. 在 Athena 控制台导航窗格中，选择 Workgroups（工作组）。
3. 在 Workgroups（工作组）页面中，选择 Create workgroup（创建工作组）。
4. 在 Create workgroup（创建工作组）页面中，如下所示填写各字段：

字段	描述
Workgroup name (工作组名称)	必需。为工作组输入唯一的名称。使用 1 到 128 个字符。（A-Z、a-z、0-9、_、-、.）。此名称不能更改。
描述	可选。为工作组输入描述。最多可包含 1024 个字符。
Choose the type of engine（选择引擎类型）	如需对 Amazon S3 中的数据运行临时 SQL 查询，或者使用预构建的数据来源连接器对 Amazon S3 外部的各种数据来源运行联合查询，请选择 Athena SQL。您可以使用 Athena 查询编辑器、AWS CLI 或 Athena API 运行查询。

字段	描述
	如需使用 Python 和 Apache Spark 创建、编辑和运行 Jupyter notebook 应用程序，请选择 Apache Spark。Jupyter notebook 包含单元格列表，其中可以包括代码、文本、Markdown、数学运算、绘图和富媒体。在 Athena 的交互式笔记本会话中，单元格按计算顺序运行。有关创建和配置启用 Spark 的工作组的信息，请参阅 在 Athena 中创建启用 Spark 的工作组。 创建工作组后，您可以升级其分析引擎（例如，从 Athena 引擎版本 2 升级到 Athena 引擎版本 3），但无法更改其引擎类型。例如，无法将 Athena 引擎版本 3 工作组更改为 PySpark 引擎版本 3 工作组。
更新查询引擎	选择您希望在发布新的 Athena 引擎版本时更新工作组的方式。您可以让 Athena 决定何时更新您的工作组或手动选择引擎版本。有关更多信息，请参阅 Athena 引擎版本控制。
身份验证模式	选择 AWS Identity and Access Management（IAM），以对工作组使用 IAM 身份验证或联合身份验证。如果要支持员工身份（例如来自 Microsoft Active Directory 等 SAML 2.0 身份提供商的用户和组），请选择 IAM Identity Center。有关更多信息，请参阅《AWS IAM Identity Center 用户指南》中的 Trusted identity propagation across applications。
IAM Identity Center 访问权限的服务角色	Athena 需要 IAM 权限才能代表您访问 IAM Identity Center。有关 IAM 服务角色的更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。

字段	描述
Location of query result（CTAS 查询结果位置）	可选。输入指向 Amazon S3 存储桶或前缀的路径。此存储桶和前缀必须首先存在，然后才能指定它们。 **Note** 如果您在控制台中运行查询，则是否指定查询结果位置是可选的。如果您没有为工作组或者在Settings (设置) 中指定它，Athena 会使用默认的查询结果位置。如果您使用 API 或驱动程序运行查询，则必须 至少在这两个位置中的一个位置中指定查询结果位置：使用 OutputLocation 为单独查询指定，或使用 WorkGroupConfiguration 为工作组指定。
预期存储桶拥有者	可选。输入您希望其成为输出位置存储桶拥有者的 AWS 账户 ID。这是附加安全措施。如果存储桶拥有者的账户 ID 与您在此处指定的 ID 不匹配，则输出到存储桶的尝试将失败。有关更多信息，请参阅《Amazon S3 用户指南》中的使用存储桶拥有者条件验证存储桶所有权 **Note** 预期存储桶拥有者设置仅适用于您为 Athena 查询结果指定的 Amazon S3 输出位置。其不适用于其他 Amazon S3 位置，例如外部 Amazon S3 存储桶中的数据源位置、CTAS 和 INSERT INTO 目标表位置、UNLOAD 语句输出位置、为联合查询溢出存储桶的操作，或针对另一个账户中的表运行的 SELECT 查询。

字段	描述
Assign bucket owner full control over query results（为存储桶拥有者分配对查询结果的完全控制权）	在默认情况下，该字段未被选中。如果您选择了此选项并且为查询结果位置存储桶启用了 ACL，您将向存储桶拥有者授予对查询结果的完全控制权。例如，假设您的查询结果位置属于其他账户所有，则可以使用此选项将所有权以及对查询结果的完全控制权授予该其他账户。 如果存储桶的 S3 Object Ownership（S3 对象所有权）设置为 Bucket owner preferred（存储桶拥有者优先），则存储桶拥有者还拥有从此工作组写入的所有查询结果对象。例如，假设某个外部账户的工作组启用了此选项并将其查询结果位置设置为您账户的 Amazon S3 存储桶，并且该存储桶的 S3 Object Ownership（S3 对象所有权）设置为 Bucket owner preferred（存储桶拥有者优先），则您将拥有并完全控制该外部工作组的查询结果的访问权限。 如果查询结果存储桶的 S3 Object Ownership（S3 对象所有权）设置为 Bucket owner enforced（存储桶拥有者强制），则选择此选项将不具有效力。有关更多信息，请参阅《Amazon S3 用户指南》中的 对象所有权设置。
Encrypt query results (加密查询结果)	可选。加密结果存储在 Amazon S3 中。如果选中，则工作组中的所有查询都会加密。 如果已选中，您可以选择 Encryption type (加密类型) 和 Encryption key (加密密钥) 并输入 KMS Key ARN (KMS 密钥 ARN)。 如果您没有密钥，请打开 AWS KMS 控制台以创建它。有关更多信息，请参阅 AWS Key Management Service 开发人员指南中的创建密钥。
将 ***encryption_type*** 设置为最低加密	可选。选择此选项可对工作组的所有用户强制对查询结果进行最低限度的加密。选择此选项可显示一个包含加密类型层次结构的表。该表还显示当您将特定加密类型指定为最低加密类型时，允许工作组用户使用哪些加密类型。要使用此选项，请勿选择覆盖客户端设置。 有关更多信息，请参阅 为工作组配置最低加密。

字段	描述
启用 S3 Access Grants	当您选择 IAM Identity Center 作为身份验证模式时，会默认选择此字段。选中后，此选项会将基于 IAM Identity Center 用户或组的权限应用于 Amazon S3 位置。
创建基于用户身份的 S3 前缀	选择此选项后，Athena 会在存储查询结果时创建 Amazon S3 前缀。前缀基于用户的 IAM Identity Center 用户身份。
Publish query metrics to CloudWatch（将查询指标发布到 CloudWatch）	在默认情况下，该字段会被选中。将查询指标发布到 CloudWatch。请参阅 使用 CloudWatch 指标监控 Athena 查询。
Override client-side settings (覆盖客户端设置)	在默认情况下，该字段未被选中。如果选择该字段，工作组设置会应用于工作组中的所有查询并覆盖客户端设置。有关更多信息，请参阅工作组设置覆盖客户端设置。
Requester Pays S3 buckets (申请方付款 S3 存储桶)	可选。如果工作组用户将对存储在配置为“申请方付款”的 Amazon S3 存储桶中的数据运行查询，请选择 Turn on queries on requester pays buckets in Amazon S3（在 Amazon S3 中启用对申请方付款存储桶的查询）。运行查询的用户的账户需要针对适用的数据访问支付费用，并支付和查询有关的数据传输费用。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 申请方付款存储桶。
Per query data usage control（管理每个查询的数据使用控制）	可选。设置查询允许扫描的最大数据量限制。只能为工作组的每个查询设置一个限制。该限制适用于工作组中的所有查询，如果查询超过限制，则将取消查询。有关更多信息，请参阅 设置数据使用控制限制。
Workgroup data usage alerts（工作组数据使用提示）	可选。当在此工作组中运行的查询在特定时间段内扫描指定数量的数据时，请设置多个提示阈值。提示使用 Amazon CloudWatch 发出告警，并适用于工作组中的所有查询。有关更多信息，请参阅《Amazon CloudWatch 用户指南》中的使用 Amazon CloudWatch 告警。

字段	描述
标签	可选。将一个或多个标记添加到工作组。标记是指您为 Athena 工作组资源分配的标签。其中包含一个键和一个值。使用 AWS 标记最佳实践 创建一组一致的标签，并将工作组按用途、所有者或环境分类。您还可以在 IAM policy 中使用标记，以及用于控制账单成本。请勿为同一工作组使用重复的标签键。有关更多信息，请参阅 the section called “标记资源”。

5. 选择 Create workgroup (创建工作组)。工作组显示在 Workgroups （工作组）页面上的列表中。

您也可以使用 CreateWorkGroup API 操作创建工作组。

> **Important**
>
> 在创建工作组后，创建将允许您运行工作组相关操作的 用于访问工作组的 IAM policy IAM。

编辑工作组

编辑工作组需要具有执行 `UpdateWorkgroup` API 操作的权限。请参阅对工作组和标签的访问和 用于访问工作组的 IAM policy。如果要添加或编辑标记，您还需要具有对 `TagResource` 的权限。请参阅工作组的标签策略示例。

在控制台中编辑工作组

1. 在 Athena 控制台导航窗格中，选择 Workgroups（工作组）。
2. 在 Workgroups（工作组）页面上，选择要编辑的工作组的按钮。
3. 选择 Actions（操作）和 Edit（编辑）。
4. 根据需要更改字段。有关字段的列表，请参阅创建工作组。您可以更改除工作组名称以外的所有字段。如果您需要更改名称，请创建具有新名称和相同设置的另一工作组。
5. 选择 Save changes（保存更改）。已更新的工作组显示在 Workgroups （工作组）页面上的列表中。

查看工作组详细信息

您可以查看每个工作组的详细信息。这些详细信息包括工作组名称、描述、启用还是禁用以及用于在该工作组中运行的查询的设置，其中包括查询结果位置、预期存储桶拥有者、加密，以及对写入查询结果存储桶的对象的控制权。如果工作组具有数据使用限制，也会显示出来。

查看工作组详细信息

1. 在 Athena 控制台导航窗格中，选择 Workgroups（工作组）。
2. 在 Workgroups（工作组）页面上，选择要查看的工作组的链接。显示工作组的 Overview Details（概览详细信息）页面。

删除工作组

如果您具有相应权限，则可以删除工作组。不能删除主工作组。

如果您有权限，则可以随时删除空工作组。您也可以删除包含已保存查询的工作组。在这种情况下，Athena 会警告您保存的查询将被删除，然后再继续删除工作组。

如果您删除正在其中的工作组，控制台会将焦点切换到主工作组。如果您有权访问它，可以运行查询并查看其设置。

如果您删除某个工作组，其设置和查询数据限制控制都将被删除。工作组范围的数据限制控制保留在 CloudWatch 中，您可以在需要时从中删除它们。

> **Important**
>
> 在删除工作组之前，确保其用户同时属于其他工作组，从而可以从中继续运行查询。如果用户的 IAM policy 仅允许他们允许该工作组中的查询，而您删除了它，则他们将不再具有运行查询的权限。有关更多信息，请参阅 Example policy for running queries in the primary workgroup。

在控制台中删除工作组

1. 在 Athena 控制台导航窗格中，选择 Workgroups（工作组）。
2. 在 Workgroups（工作组）页面上，选择要删除的工作组的按钮。
3. 依次选择操作、删除。
4. 在 Delete workgroup（删除工作组）确认提示处，输入工作组名称，然后选择 Delete（删除）。

要使用 API 操作删除工作组，请使用 DeleteWorkGroup 操作。

切换工作组

如果您具有两个工作组的权限，则可以从一个工作组切换到另一个工作组。

在每个工作组中，您可以打开最多十个查询选项卡。当您在工作组之间切换时，最多可将三个工作组中的查询选项卡保持打开。

切换工作组

1. 在 Athena 控制台中，使用右上角的 Workgroup（工作组）选项来选择工作组。
2. 如果出现 Workgroup ***workgroup-name*** settings（工作组 workgroup-name 设置）对话框，请选择 Acknowledge（确认）。

Workgroup（工作组）选项将显示已切换到的工作组名称。现在，您可以运行该工作组中的查询。

在工作组之间复制已保存的查询

目前，Athena 控制台没有将保存的查询从一个工作组直接复制到另一个工作组的选项，但您可以使用以下过程手动执行相同的任务。

要在工作组之间复制保存的查询

1. 在 Athena 控制台中，从您想复制查询的工作组中，选择 Saved queries（保存的查询）选项卡。
2. 选择要复制的已保存查询的链接。Athena 会在查询编辑器中打开查询。
3. 在查询编辑器中，选择查询文本，然后按 **Ctrl+C** 以复制查询。
4. 切换到目标工作组，或创建工作组，然后切换到该工作组。
5. 在查询编辑器中打开一个新选项卡，然后按 **Ctrl+V** 将文本粘贴到新选项卡中。
6. 在查询编辑器中，选择 Save as（另存为）将查询保存到目标工作组中。
7. 在 Choose a name（选择一个名称）对话框中，输入查询名称和可选说明。
8. 选择保存。

启用和禁用工作组

如果您拥有相应的权限，您可以在控制台中、使用 API 操作或者使用 JDBC 和 ODBC 驱动程序启用或禁用工作组。

启用或禁用工作组

1. 在 Athena 控制台导航窗格中，选择 Workgroups（工作组）。
2. 在 Workgroups（工作组）页面上，选择工作组的链接。
3. 在右上角，选择 Enable workgroup（启用工作组）或 Disable workgroup（禁用工作组）。
4. 在确认提示处，选择 Enable（启用）或 Disable（禁用）。如果您禁用一个工作组，其用户无法运行其中的查询或创建新的命名查询。如果您启用一个工作组，用户可以使用它来运行查询。

指定从中运行查询的工作组

若要指定要使用的工作组，您必须具有该工作组的权限。

指定要使用的工作组

1. 确保您的权限允许您在要使用的工作组中运行查询。有关更多信息，请参阅 the section called “ 用于访问工作组的 IAM policy”。
2. 若要指定工作组，请使用以下选项之一：
   - 如果您正在使用 Athena 控制台，请通过切换工作组设置工作组。
   - 如果您正在使用 Athena API 操作，请在 API 操作中指定工作组名称。例如，您可以在 StartQueryExecution 中设置工作组名称，如下所示：

```
StartQueryExecutionRequest startQueryExecutionRequest = new
 StartQueryExecutionRequest()
              .withQueryString(ExampleConstants.ATHENA_SAMPLE_QUERY)
              .withQueryExecutionContext(queryExecutionContext)
              .withWorkGroup(WorkgroupName)
```

   - 如果您正在使用 JDBC 或 ODBC 驱动程序，使用 Workgroup 配置参数在连接字符串中设置工作组名称。驱动程序将工作组名称传递到 Athena。在连接字符串中指定工作组参数，如以下示例所示：

```
jdbc:awsathena://AwsRegion=<AWSREGION>;UID=<ACCESSKEY>;
PWD=<SECRETKEY>;S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/<athena-
output>-<AWSREGION>/;
Workgroup=<WORKGROUPNAME>;
```

为工作组配置最低加密

作为 Athena SQL 工作组的管理员，您可以在 Amazon S3 中对该工作组的所有查询结果强制执行最低级别的加密。您可以使用此功能确保查询结果永远不会以未加密状态存储在 Amazon S3 存储桶中。

当启用最低加密功能的工作组中的用户提交查询时，他们只能将加密设置为您配置的最低级别，或者如果有更高的加密级别，则只能设置为更高的级别。Athena 在用户运行查询时指定的级别或工作组中设置的级别对查询结果进行加密。

可用级别如下：

- 基础 – 使用 Amazon S3 托管式密钥（SSE_S3）的服务器端加密。
- 中级 – 使用 KMS 托管式密钥（SSE_KMS）的服务器端加密。
- 高级 - 使用 KMS 托管式密钥（CSE_KMS）的客户端加密。

注意事项和限制

- 最低加密功能不适用于启用 Apache Spark 的工作组。
- 只有当工作组未启用覆盖客户端设置选项时，最低加密功能才起作用。
- 如果工作组启用了覆盖客户端设置选项，则以工作组加密设置为准，最低加密设置无效。
- 启用此功能不收取任何费用。

为工作组启用最低加密

在创建或更新工作组时，您可以为 Athena SQL 工作组的查询结果启用最低加密级别。为此，您可以使用 Athena 控制台、Athena API 或 AWS CLI。

使用 Athena 控制台启用最低加密

要开始使用 Athena 控制台创建或编辑工作组，请参阅创建工作组或编辑工作组。配置工作组时，请使用以下步骤启用最低加密。

为工作组查询结果配置最低加密级别

1. 在其他配置部分中，展开设置。
2. 清除覆盖客户端设置选项，或者确认该选项未被选中。
3. 在其他配置部分中，展开查询结果配置。
4. 选择加密查询结果选项。

5. 对于加密类型，选择您希望 Athena 用于工作组查询结果的加密方法（SSE_S3、SSE_KMS 或 CSE_KMS）。这些加密类型对应于基础、中级和高级安全级别。
6. 要对所有用户强制使用您选择的最低加密级别的加密方法，请选择将 ***encryption_method*** 设置为最低加密。

   当您选择此选项时，系统将显示一个表格，显示在您选择的加密类型变为最低加密类型时允许用户使用的加密层次结构和加密级别。
7. 创建工作组或更新工作组配置后，选择创建工作组或保存更改。

使用 Athena API 或 AWS CLI 启用最低加密

当您使用 CreateWorkGroup 或 UpdateWorkGroup API 创建或更新 Athena SQL 工作组时，请将 EnforceWorkGroupConfiguration 设置为 `false`，将 EnableMinimumEncryptionConfiguration 设置为 `true`，并使用 EncryptionOption 指定加密类型。

在 AWS CLI 中，使用带有 `--configuration` 或 `--configuration-updates` 参数的 `create-work-group` 或 `update-work-group` 命令，并指定与 API 对应的选项。

使用已启用 IAM Identity Center 的 Athena 工作组

AWS IAM Identity Center 的可信身份传播功能支持跨 AWS 分析服务使用您的员工身份。可信身份传播使您无需执行特定于服务的身份提供商配置或 IAM 角色设置。

借助 IAM Identity Center，您可以管理员工身份（也称为员工用户）的登录安全性。您可以在 IAM Identity Center 中创建或连接员工用户，并集中管理他们对所有 AWS 账户和应用程序的访问权限。您可以使用多账户权限为这些用户分配 AWS 账户 的访问权限。您可以使用应用程序分配为用户分配对启用了 IAM Identity Center 的应用程序、云应用程序和客户安全断言标记语言（SAML 2.0）应用程序的访问权限。有关更多信息，请参阅《AWS IAM Identity Center 用户指南》中的 Trusted identity propagation across applications。

目前，Athena SQL 支持可信身份传播，允许您为 Amazon EMR Studio 和 EMR Studio 中的 Athena SQL 接口使用相同的身份。要在 EMR Studio 中将 IAM Identity Center 身份与 Athena SQL 一起使用，您必须在 Athena 中创建已启用 IAM Identity Center 的工作组。然后，您可以使用 IAM Identity Center 控制台或 API 将 IAM Identity Center 用户或组分配给已启用 IAM Identity Center 的 Athena 工作组。来自使用可信身份传播的 Athena 工作组的查询必须从已启用 IAM Identity Center 的 EMR Studio 中的 Athena SQL 接口运行。

注意事项和限制

在 Amazon Athena 中使用可信身份传播时，请考虑以下几点：

- 创建工作组后，您无法更改工作组的身份验证方法。
  - 无法修改现有的 Athena SQL 工作组来支持启用 IAM Identity Center 的工作组。
  - 无法将启用 IAM Identity Center 的工作组修改为支持资源级 IAM 权限或基于身份的 IAM policy。
- 要访问启用了可信身份传播的工作组，必须将 IAM Identity Center 用户分配给由 Athena GetWorkGroup API 操作的响应返回的 `IdentityCenterApplicationArn`。
- 必须将 Amazon S3 Access Grants 配置为使用可信身份传播身份。有关更多信息，请参阅《Amazon S3 用户指南》中的 S3 Access Grants and corporate directory identities。
- 启用 IAM Identity Center 的 Athena 工作组需要将 Lake Formation 配置为使用 IAM Identity Center 身份。有关配置信息，请参阅《AWS Lake Formation 开发人员指南》中的 Integrating IAM Identity Center。
- 默认情况下，在使用可信身份传播的工作组中，查询会在 30 分钟后超时。您可以请求延长查询超时时间，不过在可信身份传播工作组中可以运行的最长查询时间为一个小时。
- 可信身份传播工作组中的用户或组权限更改可能需要长达一个小时才能生效。
- 使用可信身份传播的 Athena 工作组中的查询不能直接从 Athena 控制台运行。它们必须通过已启用 IAM Identity Center 的 EMR Studio 中的 Athena 接口运行。有关在 EMR Studio 中使用 Athena 的更多信息，请参阅《Amazon EMR 管理指南》中的 Use the Amazon Athena SQL editor in EMR Studio。
- 可信身份传播与以下 Athena 功能不兼容。
  - `aws:CalledVia` 上下文键。
  - Athena for Spark 工作组。
  - 对 Athena API 的联合访问
  - 使用 Lake Formation 以及 Athena JDBC 和 ODBC 驱动程序对 Athena 进行联合访问。
- 您只能在以下 AWS 区域 对 Athena 使用可信身份传播：
  - `us-east-2` – 美国东部（俄亥俄州）
  - `us-east-1` – 美国东部（弗吉尼亚州北部）
  - `us-west-1` – 美国西部（北加利福尼亚）
  - `us-west-2` – 美国西部（俄勒冈州）
  - `af-south-1` – 非洲（开普敦）
  - `ap-east-1` – 亚太地区（香港）
  - `ap-southeast-3` – 亚太地区（雅加达）
  - `ap-south-1` – 亚太地区（孟买）
  - `ap-northeast-3` – 亚太地区（大阪）

- ap-northeast-2 – 亚太地区（首尔）
- ap-southeast-1 – 亚太地区（新加坡）
- ap-southeast-2 – 亚太地区（悉尼）
- ap-northeast-1 – 亚太地区（东京）
- ca-central-1 – 加拿大（中部）
- eu-central-1 – 欧洲地区（法兰克福）
- eu-west-1 – 欧洲地区（爱尔兰）
- eu-west-2 – 欧洲地区（伦敦）
- eu-south-1 – 欧洲地区（米兰）
- eu-west-3 – 欧洲地区（巴黎）
- eu-north-1 – 欧洲地区（斯德哥尔摩）
- me-south-1 – 中东（巴林）
- sa-east-1 – 南美洲（圣保罗）

所需的权限

在 Athena 控制台中创建启用 IAM Identity Center 的工作组的管理员 IAM 用户必须附加以下策略。

- AmazonAthenaFullAccess 托管策略。有关详细信息，请参阅AWS 托管策略：AmazonAthenaFullAccess。
- 以下支持 IAM 和 AM Identity Center 操作的内联策略：

```
{
    "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": [
                "*"
            ]
        }
    ]
}
```

创建启用 IAM Identity Center 的 Athena 工作组

以下过程介绍了创建启用 IAM Identity Center 的 Athena 工作组的相关步骤和选项。有关可用于 Athena 工作组的其他配置选项的说明，请参阅创建工作组。

在 Athena 控制台中创建启用 SSO 的工作组

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 在 Athena 控制台导航窗格中，选择 Workgroups（工作组）。
3. 在 Workgroups（工作组）页面中，选择 Create workgroup（创建工作组）。
4. 在创建工作组页面上，在工作组名称中输入一个工作组的名称。
5. 对于分析引擎，请使用 Athena SQL 默认项。
6. 对于身份验证，请选择 IAM Identity Center。
7. 在 IAM Identity Center 访问权限的服务角色中，选择一个现有服务角色或创建一个新的服务角色。

   Athena 需要相应权限才能访问 IAM Identity Center。Athena 需要服务角色才能执行此操作。服务角色是一个由您管理的 IAM 角色，它授权 AWS 服务代表您访问其他 AWS 服务。有关更多信息，请参阅 IAM 用户指南中的创建向AWS服务委派权限的角色。
8. 展开查询结果配置，然后在查询结果的位置中输入或选择一个 Amazon S3 路径。
9. （可选）选择加密查询结果。
10. （可选）选择创建基于用户身份的 S3 前缀。

    创建启用 IAM Identity Center 的工作组时，默认选中启用 S3 Access Grants 选项。您可以使用 Amazon S3 Access Grants 来控制对 Amazon S3 中 Athena 查询结果位置（前缀）的访问权限。有关 Amazon S3 Access Grants 的更多信息，请参阅 Managing access with Amazon S3 Access Grants。

    在使用 IAM Identity Center 身份验证的 Athena 工作组中，您可以创建基于身份的查询结果位置，这些位置由 Amazon S3 Access Grants 管理。这些基于用户身份的 Amazon S3 前缀可让 Athena 工作组中的用户将其查询结果与同一工作组中的其他用户隔离开来。

    启用用户前缀选项后，Athena 会将用户 ID 作为 Amazon S3 路径前缀附加到工作组的查询结果输出位置（例如 `s3://DOC-EXAMPLE-BUCKET/${user_id}`）。要使用此功能，您必须对 Access Grants 进行配置，仅允许用户访问带有 `user_id` 前缀的位置。有关限制对 Athena 查询结果访问权限的 Amazon S3 Acess Grants 位置策略示例，请参阅 示例角色策略。

    > **Note**
    >
    > 选择用户身份 S3 前缀选项会自动启用工作组的“覆盖客户端侧设置”选项，如下一步所述。“覆盖客户端侧设置”选项是用户身份前缀功能的一个必要条件。

11. 展开设置，然后确认已选中覆盖客户端侧设置。

    如果已选中覆盖客户端侧设置，则会在工作组级别对工作组中的所有客户端强制实施工作组设置。有关更多信息，请参阅 工作组设置覆盖客户端设置。

12. （可选）按照 创建工作组 中所述对所需的任何其他配置进行设置。
13. 选择 Create workgroup (创建工作组)。
14. 参照 Athena 控制台的工作组部分，将 IAM Identity Center 目录中的用户或群组分配给已启用 IAM Identity Center 的 Athena 工作组。

示例角色策略

以下示例显示了将角色附加到 Amazon S3 Access Grant 位置的策略，该位置限制了对 Athena 查询结果的访问。

```
{
    "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"
}
```

Athena 工作组 API

以下是用于 Athena 工作组的部分 REST API 操作。在下列所有操作（`ListWorkGroups` 除外）中，都必须指定一个工作组。在 `StartQueryExecution` 等其他操作中，工作组参数是可选的，且此处未列出对应操作。有关操作的完整列表，请参阅 Amazon Athena API 参考。

- CreateWorkGroup
- DeleteWorkGroup
- GetWorkGroup
- ListWorkGroups
- UpdateWorkGroup

工作组故障排除

按照以下提示进行工作组故障排除。

- 检查您账户中各个用户的权限。他们必须可访问查询结果的位置，以及要从中运行查询的工作组。如果他们要切换工作组，那么同时需要两个工作组的权限。有关信息，请参阅 用于访问工作组的 IAM policy。
- 请注意 Athena 控制台中的上下文，以查看您要在哪一工作组中运行查询。如果您使用驱动程序，请确保设置为您需要的工作组。有关信息，请参阅the section called “指定从中运行查询的工作组”。
- 如果您使用 API 或驱动程序运行查询，则必须使用以下方法之一指定查询结果位置：对于个人查询，使用 OutputLocation（客户端）。在工作组中，使用 WorkGroupConfiguration。如果没有以任何一种方法指定位置，Athena 将在查询运行时发布错误。
- 如果您使用工作组设置覆盖客户端设置，您可能会遇到有关查询结果位置的错误。例如，工作组用户可能没有相应权限，不能在 Amazon S3 中的工作组位置存储查询结果。在这种情况下，添加必需的权限。
- 工作组会导致 API 操作的行为发生变化。要调用以下现有的 API 操作，要求您账户中的用户在 IAM 中对从中进行调用的工作组具有基于资源的权限。如果没有对工作组和工作组操作的权限，则下列 API 操作会引发 `AccessDeniedException`：CreateNamedQuery、DeleteNamedQuery、GetNamedQuery、ListNamedQ

和 GetQueryResultsStream（此 API 操作仅可与驱动程序一起使用，且不得通过其他方式供公众使用）。有关更多信息，请参阅《服务授权参考》中的 Amazon Athena 的操作、资源和条件键。

调用 BatchGetQueryExecution 和 BatchGetNamedQuery API 操作时，仅会返回在用户可访问的工作组中运行的那些查询的相关信息。如果用户不可访问工作组，这些 API 操作会在未处理 ID 的列表中返回未授权的查询 ID。有关更多信息，请参阅 the section called “ Athena 工作组 API”。

- 如果将从中运行查询的工作组已使用强制实施的查询结果位置配置，不要为 CTAS 查询指定 `external_location`。在此例中，Athena 将使指定了 `external_location` 的查询失败并发布错误。例如，如果您覆盖查询结果位置的客户端设置，强制工作组使用其自己的位置，此查询失败：`CREATE TABLE <DB>.<TABLE1> WITH (format='Parquet', external_location='s3://DOC-EXAMPLE-BUCKET/test/') AS SELECT * FROM <DB>.<TABLE2> LIMIT 10;`

您可能会看到以下错误。下表给出与工作组相关的部分错误以及建议的解决方案的列表。

工作组错误

错误	出错情况...
query state CANCELED.（查询状态已取消。）Bytes scanned limit was exceeded.（已超出扫描字节数限制。）	查询超过查询数据限制并已取消。请考虑重写查询以使其读取更少的数据，或联系您的账户管理员。
User: *arn:aws:iam::123456789012:user/abc* is not authorized to perform: athena:StartQueryExecution on resource: *arn:aws:athena:us-east-1:123456789012:workgroup/workgroupname*（用户：arn:aws:iam::123456789012:user/abc 无权执行：资源上的 athena：StartQueryExecution：arn:aws:athena:us-east-1:123456789012:workgroup/workgroupname）	用户在一个工作组中运行查询，但不可访问该工作组。更新您的策略以便可访问该工作组。
INVALID_INPUT. WorkGroup <name> is disabled.（工作组 <name> 已禁用。）	用户在一个工作组中运行查询，但该工作组已被禁用。您的工作组可由您的管理员禁用。也有可能是您没有对其的访问权限。在这两种情况下，与有权访问以修改工作组的管理员联系。

错误	出错情况...
INVALID_INPUT. WorkGroup <name> is not found. (未找到工作组 <name>。)	用户在一个工作组中运行查询，但该工作组不存在。如果工作组已被删除，就会发生这种情况。切换到另一工作组以运行查询。
InvalidRequestException: when calling the StartQueryExecution operation: No output location provided. (InvalidRequestException：调用 StartQueryExecution 操作时：未提供输出位置。) An output location is required either through the Workgroup result configuration setting or as an API input. (需要通过工作组结果配置设置或作为 API 输入的输出位置。)	用户使用 API 运行查询，但未指定查询结果的位置。您必须使用以下两种方法之一为查询结果设置输出位置：对于单独查询，使用 OutputLocation (客户端) ；对于工作组，使用 WorkGroupConfiguration。
The Create Table As Select query failed because it was submitted with an 'external_location' property to an Athena Workgroup that enforces a centralized output location for all queries. (“根据选择创建表”查询失败，因为其与“external_location”属性一同提交给强制执行所有查询集中输出位置的 Athena 工作组。) Please remove the 'external_location' property and resubmit the query. (请删除“external_location”属性并重新提交查询。)	如果从中运行查询的工作组已使用强制实施的查询结果位置配置，而且您为 CTAS 查询指定 `external_location` 。在这种情况下，请删除 `external_location` 并重新运行查询。
Cannot create prepared statement *`prepared_statement_name`* . (无法创建预处理语句 prepared_statement_name。) The number of prepared statements in this workgroup exceeds the limit of 1000. (该工作组中的预处理语句数已超过 1000 的限制。)	工作组中的预处理语句数已超过 1000 的限制。为解决此问题，请使用 DEALLOCATE PREPARE 从工作组中删除一条或多条预处理语句。或者，创建新工作组。

使用 CloudWatch 指标和事件控制成本和监控查询

通过使用工作组，您可以为每个查询或工作组设置数据使用控制限制，在超出这些限制时发出警报，并将查询指标发布到 CloudWatch。

在每个工作组中，您可以执行以下操作：

- 为每个查询和每个工作组配置数据使用控制，并建立在查询超出阈值时将采取的操作。
- 查看和分析查询指标，并将它们发布到 CloudWatch。如果在控制台中创建工作组，那么将为您选中将指标发布到 CloudWatch 这一设置。如果使用 API 操作，那么必须启用发布指标。发布指标时，它们将显示在 Workgroups（工作组）面板的 Metrics（指标）选项卡中。默认情况下，将为主工作组禁用指标。

视频

以下视频演示了如何在 CloudWatch 中创建自定义控制面板并设置指标告警和触发器。您可以从 Athena 控制台直接使用预先填充的控制面板来使用这些查询指标。

使用 Amazon CloudWatch 监控 Amazon Athena 查询

主题

- 启用 CloudWatch 查询指标
- 使用 CloudWatch 指标监控 Athena 查询
- 使用 Amazon EventBridge 事件监控 Athena 查询
- 监控 Athena 用量指标
- 设置数据使用控制限制

启用 CloudWatch 查询指标

在控制台中创建工作组时，默认情况下将选中将查询指标发布到 CloudWatch 这一设置。

在 Athena 控制台中为工作组启用或禁用查询指标

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。

3. 在导航窗格中，选择 Workgroups（工作组）。
4. 选择要修改的工作组链接。
5. 在工作组的详细信息页面上，选择 Edit（编辑）。
6. 在设置部分中，选择或清除将查询指标发布到 AWS CloudWatch。

如果您使用 API 操作、命令行界面或带 JDBC 驱动程序的客户端应用程序来创建工作组，要启用查询指标发布，请在 WorkGroupConfiguration 中将 `PublishCloudWatchMetricsEnabled` 设置为 `true`。以下示例仅显示指标配置，省略了其他配置：

```
"WorkGroupConfiguration": {
      "PublishCloudWatchMetricsEnabled": "true"
     ....
     }
```

使用 CloudWatch 指标监控 Athena 查询

如果选择 publish query metrics to CloudWatch（将查询指标发布到 CloudWatch）选项，Athena 会将与查询相关的指标发布到 Amazon CloudWatch。您可以创建自定义控制面板，对 CloudWatch 中的指标设置警报和触发器，或从 Athena 控制台直接使用预先填充的控制面板。

为工作组中的查询启用查询指标时，这些指标将显示在 Workgroups（工作组）面板中的 Metrics（指标）选项卡，用于 Athena 控制台中的每个工作组。

Athena 会将以下指标发布到 CloudWatch 控制台：

- DPUAllocated – 在容量预留中预置的用于运行查询的 DPU（数据处理单元）总数。
- DPUConsumed – 在预留的给定时间内，处于 RUNNING 状态的查询主动消耗的 DPU 数量。只有当工作组与容量预留关联并且包括与预留关联的所有工作组时，才会发出指标。
- DPUCount – 查询消耗的最大 DPU 数量，仅在查询完成时发布一次。
- EngineExecutionTime – 运行查询所耗费的毫秒数。
- ProcessedBytes – Athena 为每个 DML 查询扫描的字节数。
- QueryPlanningTime – Athena 计划查询处理流程所耗费的毫秒数。
- QueryQueueTime – 查询在查询队列中等待资源所耗费的毫秒数。
- ServicePreProcessingTime – 在将查询提交给查询引擎之前，Athena 用于预处理查询的毫秒数。
- ServiceProcessingTime – 查询引擎执行完查询后，Athena 处理查询结果所耗费的毫秒数。
- TotalExecutionTime – Athena 运行 DDL 或 DML 查询所耗费的毫秒数。

有关更完整的说明，请参阅本文档后面的 Athena 的 CloudWatch 指标与维度列表。

这些指标具有以下维度：

- CapacityReservation – 用于执行查询的容量预留的名称（如果适用）。
- QueryState – SUCCEEDED、FAILED 或 CANCELED
- QueryType – DML、DDL 或 UTILITY
- WorkGroup – 工作组的名称

Athena 会将以下指标发布到 AmazonAthenaForApacheSpark 命名空间下的 CloudWatch 控制台：

- DPUCount – 会话期间为执行计算而使用的 DPU 数量。

该指标具有以下维度：

- SessionId – 提交计算的会话 ID。
- WorkGroup – 工作组名称。

有关更多信息，请参阅本主题后面的 Athena 的 CloudWatch 指标与维度列表。有关 Athena 使用情况指标的信息，请参阅 监控 Athena 用量指标。

在控制台中查看工作组的查询指标

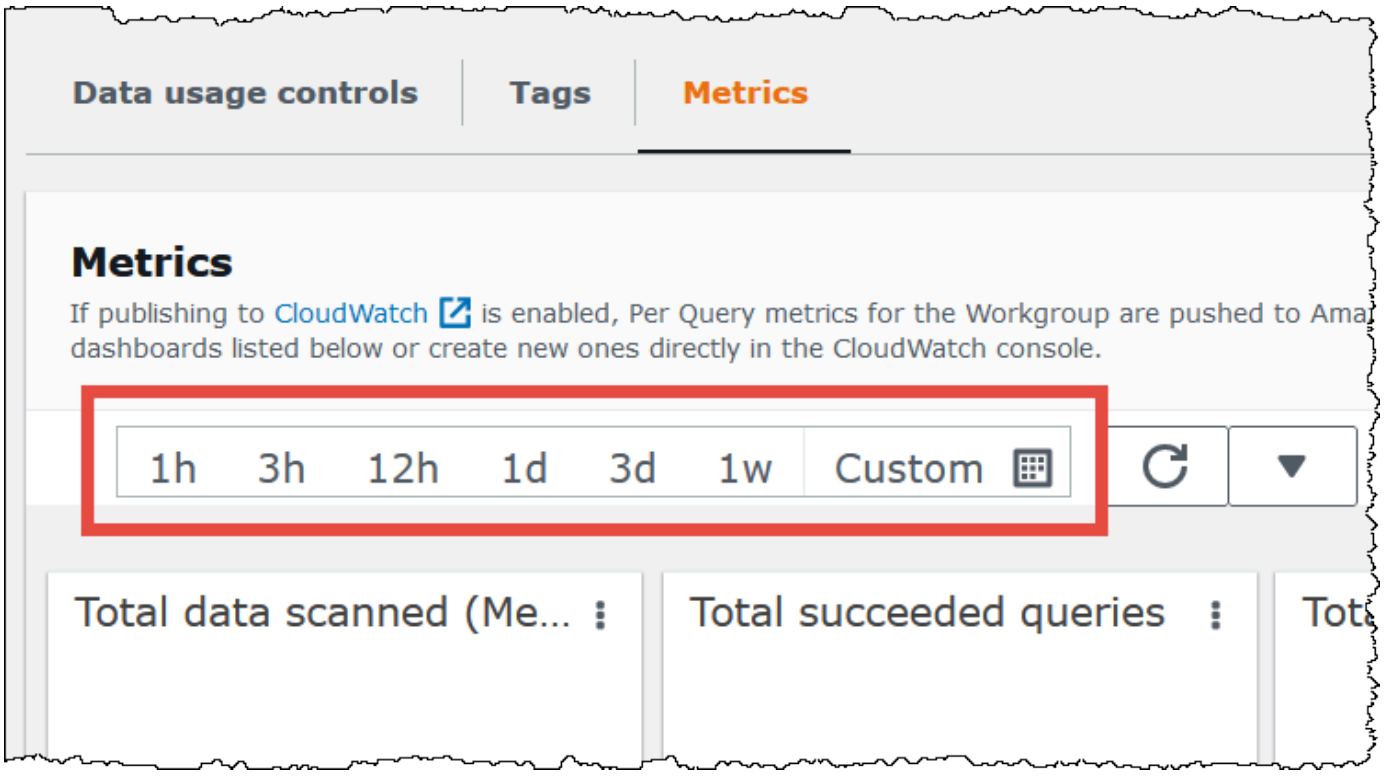
1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。

3. 在导航窗格中，选择 Workgroups（工作组）。
4. 从列表中选择所需的工作组，然后选择 Metrics（指标）选项卡。

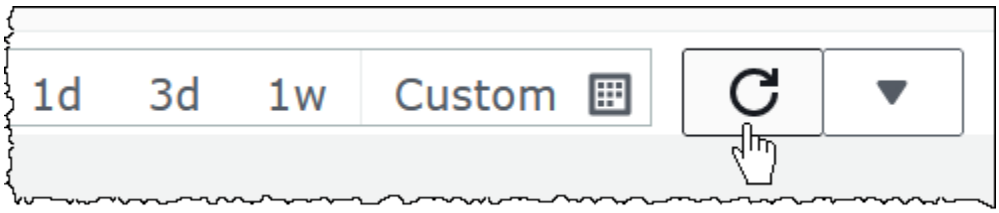
   此时将显示指标控制面板。

> **Note**
>
> 如果您最近刚为工作组启用了指标和/或没有最近的查询活动，则控制面板上的图形可能为空。根据您在下一步中指定的间隔从 CloudWatch 检索查询活动。

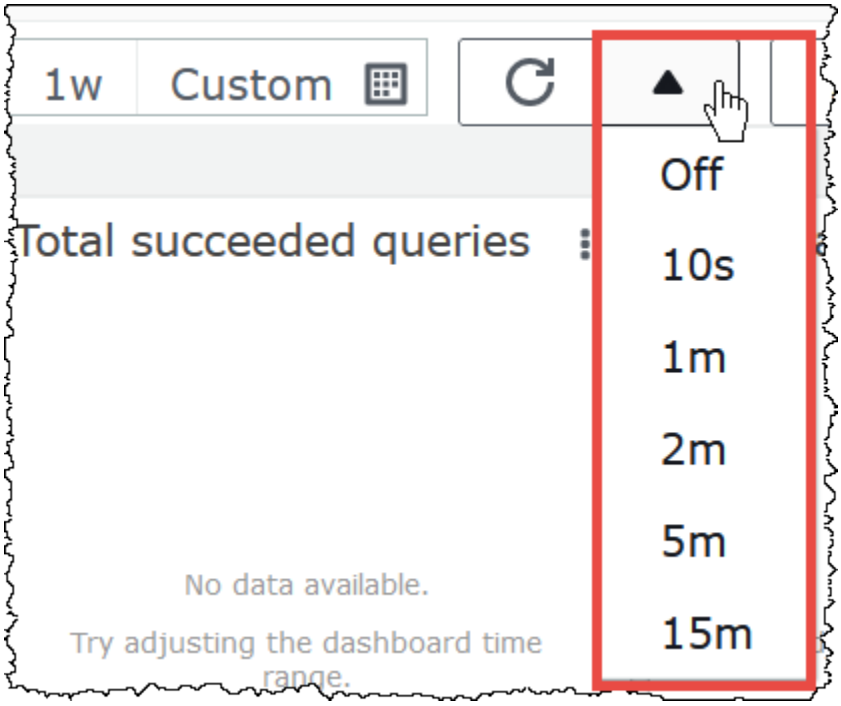
5. 在 Metrics（指标）部分中，选择 Athena 应该用于从 CloudWatch 中提取查询指标的指标间隔，或指定自定义间隔。



6. 要刷新显示的指标，请选择刷新图标。



7. 单击刷新图标旁边的箭头，选择指标显示的更新频率。



要使用 Amazon CloudWatch 控制台查看指标

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 控制台。
2. 在导航窗格中，依次选择 Metrics（指标）、All metrics（所有指标）。
3. 选择 AWS/Athena 命名空间。

借助 CLI 查看指标

- 请执行以下操作之一：
  - 要列出 Athena 的指标，请打开命令提示符，然后使用以下命令：

```
aws cloudwatch list-metrics --namespace "AWS/Athena"
```

  - 要列出所有可用的指标，请使用以下命令：

```
aws cloudwatch list-metrics"
```

Athena 的 CloudWatch 指标与维度列表

如果您在 Athena 中启用了 CloudWatch 指标，则它会按工作组将以下指标发送到 CloudWatch。以下指标使用 `AWS/Athena` 命名空间。

指标名称	描述
DPUAllocated	在容量预留中预置的用于运行查询的 DPU（数据处理单元）总数。
DPUConsumed	在预留的给定时间内，处于 `RUNNING` 状态的查询主动消耗的 DPU 数量。只有当工作组与容量预留关联并且包括与预留关联的所有工作组时，才会发出此指标。如果将工作组从一个预留转移到另一个预留，则指标将包含从该工作组属于第一个预留起的数据。有关容量预留的更多信息，请参阅管理查询处理容量。
DPUCount	查询消耗的最大 DPU 数量，仅在查询完成时发布一次。只有工作组附加到了容量预留，才会发出此指标。
EngineExecutionTime	运行查询所花费的毫秒数。
ProcessedBytes	Athena 为每个 DML 查询扫描的字节数。对于已取消的查询（由用户取消，或者因为达到限制而自动取消），此指标包含在取消时间之前扫描的数据量。对于 DDL 或 CTAS 查询，不会报告此指标。
QueryPlanningTime	Athena 计划查询处理流程所花费的毫秒数。这包括从数据源检索表分区所花费的时间。请注意，由于查询引擎执行查询计划，因此查询计划时间是 EngineExecutionTime 的子集。
QueryQueueTime	查询在查询队列中等待资源所花的毫秒数。请注意，如果发生临时错误，则可以自动将查询添加回队列。
ServicePreProcessingTime	在将查询提交给查询引擎之前，Athena 用于预处理查询的毫秒数。
ServiceProcessingTime	查询引擎执行完查询后，Athena 处理查询结果所耗费的毫秒数。

指标名称	描述
TotalExecutionTime	Athena 运行 DDL 或 DML 查询所花费的毫秒数。TotalExecutionTime 包括 QueryQueueTime、 QueryPlanningTime、EngineExecutionTime 和 ServiceProcessingTime。

Athena 的指标具有以下维度。

维度	描述
CapacityReservation	用于执行查询的容量预留的名称（如适用）。如果不使用容量预留，此维度不会返回任何数据。
QueryState	查询状态。 有效统计数据：SUCCEEDED（成功）、FAILED（失败）或 CANCELED（已取消）。
QueryType	查询类型。 有效统计数据：DDL、DML 或 UTILITY。运行的查询语句的类型。DDL 表示 DDL（数据定义语言）查询语句。DML 表示 DML（数据操作语言）查询语句，例如 `CREATE TABLE AS SELECT`。UTILITY 表示 DDL 和 DML 以外的查询语句，例如 `SHOW CREATE TABLE` 或 `DESCRIBE TABLE`。
工作组	工作组的名称。

使用 Amazon EventBridge 事件监控 Athena 查询

您可以将 Amazon Athena 与 Amazon EventBridge 结合使用来接收有关查询状态的实时通知。当您提交的查询转换了状态时，Athena 将事件发布到 EventBridge，其中包含有关该查询状态转换的信息。您可以针对感兴趣的事件编写简单规则，并在事件匹配规则时执行自动化操作。例如，您可以创建在查询到达终端状态时调用 AWS Lambda 函数的规则。尽最大努力发出事件。

在为 Athena 创建事件规则之前，您应该先执行以下操作：

- 熟悉 EventBridge 中的事件、规则和目标。有关更多信息，请参阅什么是 Amazon EventBridge？ 有关如何设置规则的更多信息，请参阅 Getting started with Amazon EventBridge。
- 创建要在您的事件规则中使用的目标。

> **Note**
>
> Athena 目前提供了一种事件类型，即 Athena 查询状态更改，但可能会添加其他事件类型和详细信息。如果您以编程方式对事件 JSON 数据反序列化，则在添加了其他属性时，请确保应用程序已准备好处理未知属性。

Athena 事件格式

以下是 Amazon Athena 事件的基本模式。

```
{
    "source":[
        "aws.athena"
    ],
    "detail-type":[
        "Athena Query State Change"
    ],
    "detail":{
        "currentState":[
            "SUCCEEDED"
        ]
    }
}
```

Athena 查询状态更改事件

以下示例显示 `currentState` 值为 SUCCEEDED 的 Athena 查询状态更改事件。

```
{
    "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"
    }
}
```

以下示例显示 currentState 值为 FAILED 的 Athena 查询状态更改事件。athenaError 数据块仅当 currentState 为 FAILED 时才会出现。有关 errorCategory 和 errorType 的值的信息，请参阅 Athena 错误目录。

```
{
    "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"
```

```
    }
}
```

输出属性

JSON 输出包括以下属性。

属性	描述
athenaError	仅当 currentState 为 FAILED 时才会出现。包含有关所发生错误的信息，例如错误类别、错误类型、错误消息以及是否可以重试导致错误的操作。每个字段的值取决于错误性质。有关 errorCategory 和 errorType 的值的信息，请参阅 Athena 错误目录。
versionId	详细对象的架构的版本号。
currentState	事件发生时查询转换到的状态。
previousState	事件发生时查询转换前的状态。
statementType	运行的查询语句的类型。
queryExecutionId	所运行查询的唯一标识符。
workgroupName	在其中运行查询的工作组的名称。
sequenceNumber	一个单调递增的数字，可用于对涉及单个查询运行的传入事件进行重复数据删除和排序。当针对同一个状态转换发布了重复的事件时，sequenceNumber 值相同。当查询多次经历状态转换（例如查询遇到极少发生的重新入队）时，您可以使用 sequenceNumber 对具有相同 currentState 和 previousState 值的事件进行排序。

示例

以下示例将事件发布到您订阅的 Amazon SNS 主题。查询 Athena 时，您会收到一封电子邮件。该示例假定 Amazon SNS 主题存在，并且您已订阅该主题。

将 Athena 事件发布到 Amazon SNS 主题

1. 为 Amazon SNS 主题创建目标。向 EventBridge 事件的服务主体授予 `events.amazonaws.com` 权限以发布到您的 Amazon SNS 主题，如下例所示。

```
{
    "Effect":"Allow",
    "Principal":{
        "Service":"events.amazonaws.com"
    },
    "Action":"sns:Publish",
    "Resource":"arn:aws:sns:us-east-1:111111111111:your-sns-topic"
}
```

2. 使用 AWS CLI `events put-rule` 命令为 Athena 事件创建规则，如以下示例所示。

```
aws events put-rule --name {ruleName} --event-pattern '{"source": ["aws.athena"]}'
```

3. 使用 AWS CLI `events put-targets` 命令将 Amazon SNS 主题目标附加到规则，如以下示例所示。

```
aws events put-targets --rule {ruleName} --targets Id=1,Arn=arn:aws:sns:us-
east-1:111111111111:your-sns-topic
```

4. 查询 Athena 并观察所调用的目标。您应该收到来自 Amazon SNS 主题的对应电子邮件。

将 AWS 用户通知服务 和 Amazon Athena 搭配使用

您可以使用 AWS 用户通知服务 来设置交付渠道，以获得有关 Amazon Athena 事件的通知。当事件与指定的规则匹配时，会收到通知。可以通过多个渠道接收事件通知，包括电子邮件、AWS Chatbot聊天通知或AWS Console Mobile Application推送通知。您还可以在控制台通知中心查看通知。用户通知服务 支持聚合，这可以减少在具体事件期间收到的通知数量。

有关更多信息，请参阅 AWS 用户通知服务 用户指南。

监控 Athena 用量指标

您可以使用 CloudWatch 用量指标在 CloudWatch 图表和控制面板上显示当前服务使用情况，以展示您的账户对资源的使用情况。

对于 Athena，用量可用性指标对应于 Athena 的 AWS 服务 配额。您可以配置警报，以在用量接近服务限额时向您发出警报。有关 Athena 服务配额的更多信息，请参阅 服务限额。有关 AWS 用量指标的更多信息，请参阅《Amazon CloudWatch 用户指南》中的 AWS 用量指标。

Athena 在 AWS/Usage 命名空间中发布以下指标。

指标名称	描述
ResourceCount	每个账户每个 AWS 区域 的所有已排队和正在执行的查询的总和，按查询类型（DML 或 DDL）分隔。最大值是此指标唯一有用的统计数据。 此指标每分钟定期发布。如果您没有运行任何查询，指标不会报告任何内容（甚至连 0 也不报告）。只有在获取指标时正在运行活动查询时才会发布该指标。

以下维度用于优化由 Athena 发布的用量指标。

维度	描述
Service	包含该资源的 AWS 服务 的名称。对于 Athena，此维度的值为 Athena。
Resource	正在运行的资源的类型。Athena 查询用量的资源值为 ActiveQueryCount 。
Type	正在报告的实体的类型。目前，Athena 用量指标的唯一有效值为 Resource。
Class	所跟踪的资源的类。对于 Athena，Class 可以是 DML 或 DDL。

在 CloudWatch 控制台中查看 Athena 资源用量指标

您可以使用 CloudWatch 控制台查看 Athena 用量指标图表，并配置告警以便在您的用量接近服务配额时提醒您。

查看 Athena 资源用量指标

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 控制台。
2. 在导航窗格中，依次选择 Metrics（指标）、All metrics（所有指标）。
3. 选择使用情况，然后选择按 AWS 资源。

   这将显示服务配额用量指标的列表。
4. 选中 Athena 和 ActiveQueryCount 旁边的复选框。
5. 选择绘成图表的指标选项卡。

   以上图表显示 AWS 资源的当前用量。

要了解如何向图表添加服务限额，以及设置警报以在接近服务限额时向您发送通知，请参阅《Amazon CloudWatch 用户指南》中的 可视化服务限额并设置警报。有关设置每个工作组的用量限制的信息，请参阅 设置数据使用控制限制。

设置数据使用控制限制

Athena 允许您设置两种类型的成本控制：每个查询限制和每个工作组限制。对于每个工作组，您可以设置一个每个查询限制和多个每个工作组限制。

- 每个查询控制限制用于指定扫描的每个查询的数据总量。如果工作组中运行的任何查询超出限制值，则该查询将取消。在工作组中您只能创建一个每个查询控制限制，该限制适用于工作组中运行的每个查询。如果您需要进行更改，可编辑限制。有关详细步骤，请参阅创建每个查询数据使用控制。
- 工作组范围内数据使用控制限制用于指定在指定时间段内，针对该工作组中运行的所有查询扫描的数据总量。您可以创建多个每个工作组限制。通过工作组范围内查询限制，您可以对工作组中运行的查询所扫描的数据设置每小时或每日聚合的多个阈值。

  如果扫描的数据聚合量超出阈值，那么您可以将通知推送到 Amazon SNS 主题。要如此做，您需要在 Athena 控制台中配置 Amazon SNS 警报和操作，以便当达到限制值时通知管理员。有关详细步骤，请参阅创建每个工作组数据使用控制。您还可以从 CloudWatch 控制台，对 Athena 发布的任何指标创建告警和操作。例如，您可以对失败的查询次数设置警报。如果数量超出特定阈值，此警报可以触发向管理员发送电子邮件。如果超出限制值，操作将向指定用户发送 Amazon SNS 警报通知。

  您可以采取的其他操作：

  - 调用 Lambda 函数。有关更多信息，请参阅《Amazon Simple Notification Service 开发人员指南》中的使用 Amazon SNS 通知调用 Lambda 函数。
  - 禁用工作组，以停止运行任何后续查询。要查看步骤，请参阅 启用和禁用工作组。

每个查询限制和每个工作组限制是相互独立的。只要超出任一限制，就会执行指定操作。如果两个或多个用户在同一工作组中同时运行查询，那么有可能每个查询未超出任何指定的限制，但扫描的数据总量超出了每个工作组的数据使用限制。在这种情况下，会向用户发送 Amazon SNS 告警。

创建每个查询数据使用控制

每个查询控制限制用于指定扫描的每个查询的数据总量。如果工作组中运行的任何查询超出限制值，则该查询将取消。取消的查询将按照 Amazon Athena 定价进行计费。

> **Note**
>
> 当查询被取消或失败时，Athena 可能已将部分结果写入 Amazon S3。在这种情况下，Athena 不会从存储结果的 Amazon S3 前缀中删除部分结果。您必须删除带有部分结果的 Amazon S3 前缀。Athena 使用 Amazon S3 分段上传来写入数据 Amazon S3。我们建议您设置存储桶生命周期策略，以便当查询失败时终止分段上传。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 使用存储桶生命周期策略中止未完成的分段上传。

在工作组中您只能创建一个每个查询控制限制，该限制适用于工作组中运行的每个查询。如果您需要进行更改，可编辑限制。

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。

3. 在导航窗格中，选择 Workgroups（工作组）。
4. 从列表中选择工作组的名称。

5. 打开 Data usage controls（数据使用控制）选项卡，在 Per query data usage control（每个查询数据使用控制）部分，选择 Manage（管理）。
6. 在 Manage per query data usage control（管理每个查询数据使用控制）页面上，指定以下值：

   - 对于 Data limit（数据限制），指定一个介于 10MB（最小值）与 7EB（最大值）之间的值。

     > **Note**
     >
     > 这些是控制台对工作组内的数据使用控制施加的限制。不表示 Athena 中的任何查询限制。

   - 对于单位，从下拉列表中选择单位值 [例如，Kilobytes KB（千字节 KB）或 Exabytes EB（艾字节 EB）]。

   如果超出限制，默认操作为取消查询。此设置不可更改。
7. 选择保存。

创建或编辑每个工作组的数据使用提示

当工作组中运行的查询在特定时间段内扫描指定数量的数据时，可以设置多个提示阈值。提示使用 Amazon CloudWatch 发出告警，并适用于工作组中的所有查询。达到阈值后，您可以让 Amazon SNS 向指定的用户发送电子邮件。达到阈值时，查询不会自动取消。

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。
3. 在导航窗格中，选择 Workgroups（工作组）。
4. 从列表中选择工作组的名称。
5. 选择 Edit（编辑）以编辑工作组设置。
6. 向下滚动到并展开 Workgroup data usage alerts - optional（工作组数据使用提示 - 可选）。
7. 选择 Add alert（添加提示）。
8. 对于 Data usage threshold configuration（数据使用阈值配置），指定以下值：

   - 对于 Data threshold（数据阈值），指定数字，然后从下拉列表中选择单位值。
   - 对于 Time period（时间段），从下拉列表中选择时间段。
   - 对于 SNS topic selection（SNS 主题选择），从下拉列表中选择 Amazon SNS 主题。或者，选择 Create SNS topic（创建 SNS 主题）直接转至 Amazon SNS 控制台，创建 Amazon SNS 主

题，并为 Athena 账户中的用户之一设置该主题的订阅。有关更多信息，请参阅 Amazon Simple Notification Service 开发人员指南中的 Amazon SNS 入门。

9. 如果正在创建新提示，则选择 Add alert（添加提示）；如果要保存现有的提示，则选择 Save（保存）。

管理查询处理容量

可以使用容量预留为您在 Athena 中运行的查询指定专用处理容量。可通过容量预留，利用工作负载管理功能，帮助您控制和扩展最重要的交互式工作负载并确定其优先级。例如，您可以随时添加容量以增加可并发运行的查询数量，控制可以使用容量的工作负载，并在工作负载之间共享容量。容量完全由 Athena 托管，只要您需要，可以保留任意长的时间。设置简单，无需更改 SQL 语句。

要获取查询的处理容量，可创建容量预留，指定所需的数据处理单元（DPU）数，并为预留分配一个或多个工作组。

在您使用容量预留时，工作组至关重要。工作组可用于将查询分为不同的逻辑分组。可通过容量预留，有选择地将容量分配给工作组，从而控制每个工作组的查询行为方式和计费方式。有关工作组的更多信息，请参阅 使用工作组控制查询访问和成本。

可通过将工作组分配给预留，优先处理提交给已分配工作组的查询。例如，您可以向工作组分配为用于时间敏感型财务报告查询的容量，以将这些查询与另一个工作组中不太重要的查询隔离开来。这样可以为关键工作负载提供一致的查询执行，同时允许其他工作负载独立运行。

您可以同时使用容量预留和工作组来满足不同的需求。下面是一些示例方案：

- 隔离 - 要隔离重要工作负载，您可以将单一工作组分配给一个预留。只有来自已分配工作组的查询才使用所选预留的处理能力。
- 共享 – 多个工作负载可以共享一个预留中的容量。例如，如果您需要一组特定工作负载的可预测每月成本，则可以将多个工作组分配给单一预留。分配的工作组共享预留容量。
- 混合模型 – 您可以在同一账户中同时使用容量预留和按查询计费。例如，为了确保可靠地执行支持生产应用程序的查询，可以将这些查询的一个工作组分配给容量预留。要先制定查询再将其移至生产工作组，您可以使用与预留无关联且使用按查询计费的单独工作组。

了解 DPU

容量以数据处理单元（DPU）数进行度量。DPU 表示 Athena 代表您访问和处理数据时所使用的计算和内存资源。一个 DPU 提供 4 个 vCPU 和 16GB 内存。所指定的 DPU 数会影响可以并发运行的查询数。例如，具有 256 个 DPU 的预留所支持的并发查询数大约是具有 128 个 DPU 的预留的两倍。

您最多可以创建 100 个容量预留，每个账户和区域最多可以具有 1000 个 DPU。您至少可以请求 24 个 DPU。如果您的使用案例需要超过 1000 个 DPU，请联系 athena-feedback@amazon.com。

有关估算容量需求的信息，请参阅 确定容量要求。有关定价信息，请参阅 Amazon Athena 定价。

注意事项和限制

- 该功能需要 Athena 引擎版本 3。
- 一次最多可以将一个工作组分配给一个预留，并且您最多可以向一个预留添加 20 个工作组。
- 您不能将启用 Spark 的工作组添加至容量预留。
- 要删除已分配给预留的工作组，请先将该工作组从预留中移除。
- 您至少可以预置 24 个 DPU。
- 您最多可以创建 100 个容量预留，每个账户和区域最多可以具有 1000 个 DPU。
- 容量请求无法保证，并且可能需要花费 30 分钟时间完成。
- 每个预留的计费周期至少为 1 小时。超出 1 小时，则按分钟对容量计费。有关定价信息，请参阅 Amazon Athena 定价。
- 预留容量无法转移到另一个容量预留 AWS 账户 或 AWS 区域。
- 对容量预留的 DDL 查询会消耗 DPU。
- 在预置容量上运行的查询不计入 DDL 和 DML 的活动查询限制。
- 如果所有 DPU 都在使用中，则已提交查询将排队。此类查询不会遭拒，也不会进入按需容量。
- DPUConsumed CloudWatch 指标是按工作组计算，而不是按预留量计算。因此，如果将工作组从一个预留转移到另一个预留，则 DPUConsumed 指标将包含从该工作组属于第一个预留起的数据。有关在 Athena 中使用 CloudWatch 指标的更多信息，请参阅使用 CloudWatch 指标监控 Athena 查询。
- 目前，该功能适用于以下 AWS 区域：
  - 美国东部（俄亥俄）
  - 美国东部（弗吉尼亚州北部）
  - US West（Oregon）
  - 亚太地区（新加坡）
  - 亚太地区（悉尼）
  - 亚太地区（东京）
  - 欧洲地区（爱尔兰）
  - 欧洲地区（斯德哥尔摩）

主题

确定容量要求

在创建容量预留之前，您可以估算所需的容量，以向其分配正确的 DPU 数。在使用预留后，您可能需要检查预留的容量是不足还是过剩。本主题介绍了可用于进行这些估算的方法，同时介绍了一些用于评估使用量和成本的 AWS 工具。

主题

估算所需容量

在估算容量需求时，建议考虑两个方面：特定查询可能需要的容量数量以及大体上可能需要的容量数量。

估算每个查询的容量需求

要确定查询可能需要的 DPU 数，可遵循以下准则：

- DDL 查询消耗 4 个 DPU。
- DML 查询会消耗 4 到 124 个 DPU。

Athena 会在提交查询时确定 DML 查询所需的 DPU 数。该数量取决于数据大小、存储格式、查询构造和其他因素。通常，Athena 会尝试选择最低且最高效的 DPU 数。如果 Athena 确定查询需要更高的计算能力才能成功完成，则它会增加分配给查询的 DPU 数。

估算特定于工作负载的容量需求

要确定同时运行多个查询可能需要的容量数量，请遵循下表中的常规准则：

并发查询	所需 DPU 数量
10	大于等于 40
20	大于等于 96
大于等于 30	大于等于 240

请注意，所需的实际 DPU 数量取决于您的目标和分析模式。例如，如果您希望查询立即开始而不排队，确定您的并发查询需求峰值，然后相应地预置 DPU 数量。

您可以配置低于峰值需求的 DPU 数，但是在出现需求峰值时，可能会导致排队。出现排队时，Athena 会将您的查询搁置在队列中，并在容量可用时运行这些查询。

如果您的目标是在固定预算内运行查询，则可以使用 AWS 定价计算器来确定适合自己预算的 DPU 数。

最后，请记住，数据大小、存储格式和查询写入方式都会影响查询所需的 DPU 数量。要提高查询性能，可对数据进行压缩或分区，或者将其转换为列格式。有关更多信息，请参阅 Athena 中的性能优化。

表明需要更多容量的迹象

容量不足错误消息和查询排队均表明您分配的容量不足。

如果您的查询失败并显示容量不足错误消息，则容量预留的 DPU 数过低，无法满足查询工作。例如，如果您的预留包含 24 个 DPU，并且运行的查询需要超过 24 个 DPU，则查询将失败。要监控此查询错误，可以使用 Athena 的 EventBridge 事件。尝试添加更多 DPU，并重新运行查询。

如果许多查询在排队，这意味着您的容量已被其他查询完全占据。要减少排队时间，请执行以下操作之一：

- 向预留添加 DPU 以提高查询并发性。
- 从预留中移除工作组，以释放容量供其他查询使用。

要检查查询排队是否过多，使用容量预留中工作组的 Athena 查询队列时间 CloudWatch 指标。如果该值高于您的首选阈值，则可以将 DPU 添加至容量预留。

检查是否存在空闲容量

要检查是否存在空闲容量，您可以减少预留中的 DPU 数或增加其工作负载，然后观察结果。

检查是否存在空闲容量

1. 请执行以下操作之一：
   - 减少预留中的 DPU 数（减少可用资源）
   - 向预留添加工作组（增加工作负载）
2. 使用 CloudWatch 测量查询队列时间。
3. 如果队列时间超过所需水平，执行以下任一操作
   - 移除工作组
   - 将 DPU 添加至容量预留
4. 在每次更改后，都要检查性能和查询队列时间。
5. 继续调整工作负载和/或 DPU 数以达到所需的平衡。

如果不希望在首选时间段范围之外保持容量，则可以取消预留并稍后创建另一个预留。但是，即使您最近取消了另一个预留的容量，也无法保证会请求新的容量，并且创建新的预留需要花费一些时间。

用于评估容量需求和成本的工具

您可以使用 AWS 中的以下服务和功能测量 Athena 的使用情况和成本。

CloudWatch 指标

您可以将 Athena 配置为在工作组级别下向 Amazon CloudWatch 发布查询相关指标。为工作组启用指标后，工作组查询的指标将在 Athena 控制台的工作组详细信息页面中显示。

有关发布到 CloudWatch 的 Athena 指标及其维度的信息，请参阅 使用 CloudWatch 指标监控 Athena 查询。

CloudWatch 使用情况指标

您可以使用 CloudWatch 用量指标在 CloudWatch 图表和控制面板上显示当前服务使用情况，以展示您的账户对资源的使用情况。对于 Athena，用量可用性指标对应于 AWS 服务限额。您可以配置警报，以在用量接近服务限额时向您发出警报。

有关更多信息，请参阅 监控 Athena 用量指标。

Amazon EventBridge 事件

您可以将 Amazon Athena 与 Amazon EventBridge 结合使用来接收有关查询状态的实时通知。当已提交查询的状态发生变化时，Athena 将向 EventBridge 发布一个包含有关查询状态转换的信息的事件。您可以针对感兴趣的事件编写简单规则，并在事件匹配规则时执行自动化操作。

有关详细信息，请参阅以下资源：

- 使用 Amazon EventBridge 事件监控 Athena 查询
- 什么是 Amazon EventBridge？
- Amazon EventBridge 事件

标签

在 Athena 中，容量预留支持标签。每个标签均包含一个键和一个值。要在 Athena 中跟踪成本，您可以使用 AWS 生成的成本分配标签。AWS 使用成本分配标签对成本和使用率报告上的资源成本进行组织。这样您可以更轻松地对 AWS 成本进行分类和跟踪。要激活 Athena 的成本分配标签，您可以使用 AWS Billing and Cost Management 控制台。

有关详细信息，请参阅以下资源：

- 为 Athena 资源添加标签
- 激活 AWS 生成的成本分配标签
- 使用 AWS 成本分配标签

创建容量预留

首先，创建一个具有所需 DPU 数的容量预留，然后分配一个或多个将使用该容量进行查询的工作组。您可以稍后根据需要调整容量，以实现更一致的性能或优化管理成本。有关估算容量需求的信息，请参阅 确定容量要求。

> **Important**
>
> 容量请求无法保证，并且可能需要花费 30 分钟时间完成。

创建容量预留

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。
3. 选择管理、容量预留。
4. 选择创建容量预留。
5. 在创建容量预留页面的容量预留名称中，输入相应的名称。名称必须唯一，长度介于 1 到 128 个字符之间，并且只能使用字符 a-z、A-Z、0-9、_（下划线）、.（句点）和 -（连字符）。在创建预留后无法更改其名称。
6. 对于 DPU，选择或输入所需的数据处理单元（DPU）数，增量为 4。有关更多信息，请参阅 了解 DPU。
7. （可选）展开标签选项，然后选择添加新标签添加一个或多个要与容量预留资源相关联的自定义键/值对。有关更多信息，请参阅 为 Athena 资源添加标签。
8. 选择审核。
9. 在确认创建容量预留提示中，确认 DPU 数、AWS 区域 和其他信息。如果您接受这些内容，请选择提交。

   在详细信息页面上，您的容量预留状态显示为待处理。当您的预留容量可供运行查询时，其状态将显示为活动。

此时，您可以随时向预留添加一个或多个工作组。要查看步骤，请参阅 向预留添加工作组。

管理预留

您可以在容量预留页面上查看和管理您的容量预留。您可以执行诸多管理任务，例如添加或减少 DPU、修改工作组分配以及标记或取消预留。

查看和管理容量预留

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。

3. 选择管理、容量预留。
4. 在容量预留页面上，可以执行以下任务：

- 要创建容量预留，请选择创建容量预留。
- 使用搜索框按 DPU 的名称或数量筛选预留。
- 选择状态下拉菜单以按容量预留状态（例如，活动或已取消）进行筛选。有关预留状态的更多信息，请参阅 了解预留状态。
- 要查看容量预留的详细信息，请选择预留链接。预留的详细信息页面包含用于编辑容量、添加工作组、删除工作组和取消预留的选项。
- 要编辑预留（例如，通过添加或移除 DPU），请选择预留对应的按钮，然后选择编辑。
- 要取消预订，请选择预留对应的按钮，然后选择取消。

了解预留状态

下表描述了容量预留的可能状态值。

Status	描述
待处理	Athena 正在处理您的容量请求。容量未就绪，无法运行查询。
处于活动状态	容量可用于运行查询。
已失败	您的容量请求未成功完成。请注意，无法保证完成容量请求。失败的预留将计入您的账户 DPU 限制。要释放用量，必须取消预留。
待更新	Athena 正在处理预留更改。例如，在编辑预留以添加或删除 DPU 之后，就会出现该状态。
正在取消	Athena 正在处理取消预留请求。允许正在使用预留的工作组中仍在运行的查询完成，但工作组中的其他查询将使用按需（未预置）容量。
已取消	容量预留取消已完成。已取消预留将在控制台中保留 45 天。45 天后，Athena 将删除该预留。在 45 天内，您不能重新利用或重复使用该预留，但是可以引用其标签并查看其详细信息以供历史参考。 无法保证已取消容量在未来可以重新预留。容量无法转移到另一个预留 AWS 账户 或 AWS 区域。

了解活动 DPU 和目标 DPU

在 Athena 控制台的容量预留列表中，您的预留会显示两个 DPU 值：活动 DPU 和目标 DPU。

- 活动 DPU - 您的预留中可用于运行查询的 DPU 数。例如，如果您请求 100 个 DPU，并且您的请求已完成，则活动 DPU 会显示 100。
- 目标 DPU - 您的预留正在移至的 DPU 数。在创建预留时或者在等待增加或减少 DPU 数时，目标 DPU 显示的值与活动 DPU 不同。

例如，在您提交创建具有 24 个 DPU 的预留请求后，预留状态将为待处理，活动 DPU 将为 0，目标 DPU 将为 24。

如果您的预留具有 100 个 DPU，并且编辑您的预留以请求增加 20 个 DPU，则状态将为待更新，活动 DPU 将为 100，目标 DPU 将为 120。

如果您的预留具有 100 个 DPU，并且编辑您的预留以请求减少 20 个 DPU，则状态将为待更新，活动 DPU 将为 100，目标 DPU 将为 80。

在此类转换期间，Athena 会根据您的要求主动获取或减少 DPU 数。当活动 DPU 等于目标 DPU 时，表示已达到目标数字，并且没有待处理的更改。

要以编程的方式检索这些值，可以调用 GetCapacityReservation API 操作。API 将活动 DPU 和目标 DPU 称为 `AllocatedDpus` 和 `TargetDpus`。

主题

编辑容量预留

在创建容量预留后，您可以调整其 DPU 数并添加或移除其自定义标签。

编辑容量预留

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。

2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。
3. 选择管理、容量预留。
4. 在容量预留列表中，执行以下操作之一：
   - 选择预留旁的按钮，然后选择编辑。
   - 选择预留链接，然后选择编辑。
5. 对于 DPU，选择或输入所需的数据处理单元数，增量为 4。您至少可以具有 24 个 DPU。有关更多信息，请参阅 了解 DPU。

   > **Note**
   >
   > 您可以随时向现有容量预留添加 DPU。但是，在创建预留或向其添加 DPU 后的 1 小时内，无法减少 DPU 数。

6. （可选）对于标签，选择移除移除标签，或选择添加新标签添加新标签。
7. 选择提交。预留的详细信息页面将显示更新后的配置。

向预留添加工作组

在创建容量预留后，最多可以向预留添加 20 个工作组。向预留添加工作组即会告知 Athena 应对预留容量执行的查询。来自与预留无关的工作组的查询将继续使用默认的每 TB 扫描定价模型运行。

当预留具有两个或多个工作组时，来自这些工作组的查询可以使用预留容量。您可以随时添加和移除工作组。在添加或移除工作组时，正在运行的查询不会中断。

当您的预留处于待处理状态时，来自您添加的工作组的查询将继续使用默认每 TB 扫描定价模型运行，直到预留变为活动状态为止。

向容量预留添加一个或多个工作组

1. 在容量预留的详细信息页面中，选择添加工作组。
2. 在添加工作组页面中，选择要添加的工作组，然后选择添加工作组。无法将一个或多个工作组分配给预留。

   容量预留的详细信息页面列出了已添加的工作组。在这些工作组中运行的查询将在预留处于活动状态时使用预留的容量。

从预留中移除工作组

如果您不再需要工作组的专用容量或想要将工作组移至其自己的预留，则可以随时移除该工作组。将工作组从预留中移除这一过程很简单。在从预留中移除工作组后，已移除工作组中的查询默认为使用按需（非预置）容量，并根据扫描的 TB 进行计费。

移除预留中的一个或多个工作组

1. 在容量预留的详细信息页面中，选择要移除的工作组。
2. 选择移除工作组。是否移除工作组？提示会通知您在将工作组从预留中移除之前将完成所有当前处于活动状态的查询。
3. 选择移除。容量预留的详细信息页面将不再显示已移除工作组。

取消容量预留

如果您不再需要容量预留，则可以将其取消。允许正在使用预留的工作组中仍在运行的查询完成，但工作组中的其他查询将不再使用该预留。

> **Note**
>
> 无法保证已取消容量在未来可以重新预留。容量无法转移到另一个预留 AWS 账户 或 AWS 区域。

取消容量预留

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。
3. 选择管理、容量预留。
4. 在容量预留列表中，执行以下操作之一：
   - 选择预留旁的按钮，然后选择取消。
   - 选择预留链接，然后选择创建容量预留。
5. 在是否取消容量预订？提示，输入取消，然后选择取消容量预留。

   预订状态将更改为正在取消，并显示一个进度横幅，通知您正在进行取消。

在取消完成后，容量预留仍存在，但其状态显示为已取消。将在取消 45 天后删除该预留。在 45 天内，您不能重新利用或重复使用已取消预留，但是可以引用其标签并查看该预留以供历史参考。

删除容量预留

如果要移除所有对已取消容量预留的引用，可以删除该预留。必须取消预留，然后才能将其删除。已删除预留将立即从您的账户中移除，且无法再进行引用，包括其 ARN。

删除容量预留

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。
3. 选择管理、容量预留。
4. 在容量预留列表中，执行以下操作之一：
   - 选择已取消预留旁的按钮，然后选择操作、删除。
   - 选择预留链接，然后选择删除。
5. 在是否删除容量预留？提示时，选择删除。

   将显示一个横幅，通知您容量预留已成功删除。已删除预留将不再出现在容量预留列表中。

容量预留的 IAM policy

要控制对容量预留的访问，使用资源级 IAM 权限或基于身份的 IAM policy。每当您使用 IAM policy 时，请确保遵循 IAM 最佳实践。有关更多信息，请参阅《IAM 用户指南》中的 IAM 安全最佳实践。

以下是 Athena 的特定过程。

有关 IAM 的特定信息，请参阅本节末尾列出的链接。有关示例 JSON 容量预留策略的信息，请参阅 容量预留示例政策。

在 IAM 控制台中使用可视化编辑器创建容量预留策略

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧的导航窗格中，选择 Policies (策略)，然后选择 Create policy (创建策略)。
3. 在可视化编辑器选项卡上，选择选择服务。然后，选择要添加到策略的 Athena。

4. 选择 Select actions (选择操作)，然后选择要添加到策略的操作。可视化编辑器会显示 Athena 中可用的操作。有关更多信息，请参阅《服务授权参考》中的 Amazon Athena 的操作、资源和条件键。
5. 选择添加操作输入特定操作，或使用通配符 (*) 指定多个操作。

   预设情况下，您创建的策略允许执行选择的操作。如果您在 Athena 中选择对 `capacity-reservation` 资源执行一个或多个支持资源级权限的操作，则编辑器会列出 `capacity-reservation` 资源。
6. 选择资源指定特定于您的策略的容量预留。有关示例 JSON 容量预留策略的信息，请参阅 容量预留示例政策。
7. 指定 `capacity-reservation` 资源，如下所示：

   ```
   arn:aws:athena:<region>:<user-account>:capacity-reservation/<capacity-reservation-name>
   ```
8. 选择 Review policy (查看策略)，然后为您创建的策略键入 Name (名称) 和 Description (描述) (可选)。查看策略摘要以确保您已授予所需的权限。
9. 选择创建策略可保存您的新策略。
10. 将此基于身份的策略附加到用户、组或角色。

有关详细信息，请参阅服务授权参考和《IAM 用户指南》中的以下主题：

- Amazon Athena 的操作、资源和条件键
- 使用可视化编辑器创建策略
- 添加和移除 IAM policy
- 控制对资源的访问

有关示例 JSON 容量预留策略的信息，请参阅 容量预留示例政策。

有关 Amazon Athena 操作的完整列表，请参阅 Amazon Athena API 参考中的 API 操作名称。

容量预留示例政策

本节包含可用于启用对容量预留执行的各种操作的策略示例。每当您使用 IAM policy 时，请确保遵循 IAM 最佳实践。有关更多信息，请参阅《IAM 用户指南》中的 IAM 安全最佳实践。

容量预留是由 Athena 托管的 IAM 资源。因此，如果容量预留策略使用将 `capacity-reservation` 用作输入的操作，则必须指定容量预留 ARN，如下所示：

```
"Resource": [arn:aws:athena:<region>:<user-account>:capacity-reservation/<capacity-reservation-name>]
```

其中，*<capacity-reservation-name>* 为容量预留的名称。例如，对于名为 `test_capacity_reservation` 的容量预留，将其指定为资源，如下所示：

```
"Resource": ["arn:aws:athena:us-east-1:123456789012:capacity-reservation/
test_capacity_reservation"]
```

有关 Amazon Athena 操作的完整列表，请参阅 Amazon Athena API 参考中的 API 操作名称。有关 IAM policy 的更多信息，请参阅《IAM 用户指南》中的使用可视化编辑器创建策略。

- Example policy to list capacity reservations
- Example policy for management operations

Example 用于列出容量预留的策略示例

以下策略允许所有用户列出所有容量预留。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListCapacityReservations"
            ],
            "Resource": "*"
        }
    ]
}
```

Example 适用于管理操作的策略示例

以下策略允许用户创建、取消和更新容量预留 `test_capacity_reservation` 以及获取其详细信息。该策略还允许用户将 `workgroupA` 和 `workgroupB` 分配给 `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"
         ]
      }
   ]
}
```

Athena 容量预留 API

以下列表包含指向 Athena 容量预留 API 操作的参考链接。有关数据结构及其他 Athena API 操作，请参阅 Amazon Athena API Reference（Amazon Athena API 参考）。

- CancelCapacityReservation
- CreateCapacityReservation
- GetCapacityAssignmentConfiguration
- GetCapacityReservation
- ListCapacityReservations
- PutCapacityAssignmentConfiguration
- UpdateCapacityReservation

Athena 中的性能优化

本主题提供了有关提高 Athena 查询性能的一般信息和具体建议，以及如何解决与限制和资源使用情况相关的错误。

服务限额

Athena 对查询运行时间、账户中的并发查询数量和 API 请求速率等指标强制实施限额。有关限额的更多信息，请参阅 服务限额。超过这些限额会导致查询失败，无论是在提交时还是在查询执行过程中。

本页上的许多性能优化提示可以帮助缩短查询的运行时间。优化可以释放容量，这样您就可以在并发限额内运行更多查询，并防止查询因运行时间过长而被取消。

并发查询和 API 请求数量的限额按 AWS 账户 和 AWS 区域 计算。我们建议每个 AWS 账户 运行一个工作负载（或使用单独的预调容量预留），以防止工作负载争夺相同的限额。

如果您在同一个账户中运行两个工作负载，则其中一个工作负载可能会运行大量查询。这可能会导致剩余的工作负载受到限制或阻止，无法运行查询。为避免这种情况，您可以将工作负载转移到不同的账户，为每个工作负载提供自己的并发限额。为其中一个或两个工作负载创建预调配容量预留可以实现相同的目标。

其他服务的限额

当 Athena 运行查询时，它可以调用其他强制实施限额的服务。在查询执行期间，Athena 可以对 AWS Glue Data Catalog、Amazon S3 以及其他 AWS 服务（如 IAM 和 AWS KMS）进行 API 调用。如果您使用联合查询，Athena 也会调用 AWS Lambda。所有这些服务都有自己的限制和限额，可以超出。当查询执行遇到来自这些服务的错误时，查询会失败并包括来自源服务的错误。系统会重试可恢复的错误；但如果问题不能及时解决，查询仍可能失败。请务必仔细阅读错误消息，以确定它们是来自 Athena 还是来自其他服务。本文档介绍了一些相关错误。

有关如何解决由 Amazon S3 服务限额导致的错误的更多信息，请参阅本文档后面的 避免文件过多。有关 Amazon S3 性能优化的更多信息，请参阅《Amazon S3 用户指南》中的最佳实践设计模式：优化 Amazon S3 性能。

资源限制

Athena 在分布式查询引擎中运行查询。当您提交查询时，Athena 引擎查询计划程序会估计运行查询所需的计算容量，并相应地准备计算节点集群。有些查询（例如 DDL 查询）只能在一个节点上运行。对大型数据集的复杂查询在更大的集群上运行。节点是统一的，具有相同的内存、CPU 和磁盘配置。Athena 横向扩展，而非纵向扩展，以处理要求更高的查询。

有时，查询的需求会超过运行查询的集群可用的资源。发生这种情况时，查询会失败，并显示错误查询在此缩放系数耗尽了资源。

最常耗尽的资源是内存，但在极少数情况下，也可能是磁盘空间。内存错误通常发生在引擎执行联接或窗口函数时，但也可能发生在不同的计数和聚合中。

即使查询因出现“资源不足”错误而失败一次，当您再次运行它时，它也可能会成功。查询执行的确定性不定。加载数据需要多长时间以及中间数据集在节点上的分布方式等因素可能会导致不同的资源使用情况。例如，假设一个查询连接两个表，并且在连接条件的值分布中存在严重偏差。这样的查询在大多数情况下都可以成功，但是当最常见的值最终由同一个节点处理时，偶尔会失败。

为防止您的查询超出可用资源，请使用本文档中提到的性能调整提示。特别是，有关如何优化耗尽可用资源的查询的提示，请参阅 优化连接、优化窗口函数 和 使用近似值优化查询。

查询优化技术

使用本节中介绍的查询优化技巧来加快查询的运行速度，或者将其作为超出 Athena 资源限制的查询的变通方法。

优化连接

在分布式查询引擎中执行联接有许多不同的策略。其中最常见的两种是分布式哈希联接和具有复杂联接条件的查询。

分布式哈希联接

最常见的连接类型使用等式比较作为连接条件。Athena 以分布式哈希联接的形式运行这种联接。

在分布式哈希联接中，引擎从联接的一端构建查找表（哈希表）。该端称为构建端。构建端的记录分布在各个节点上。每个节点都为其子集生成一个查找表。然后，联接的另一端（称为探测端）通过节点进行流式传输。探测端的记录以与构建端相同的方式分布在节点上。这使每个节点都能够通过在自己的查找表中查找匹配的记录来执行联接。

当从联接的构建端创建的查找表无法存入内存时，查询会失败。即使构建端的总大小小于可用内存，如果记录的分布存在明显偏差，查询也会失败。在极端情况下，所有记录的联接条件值可能相同，并且必须放入单个节点上的内存中。如果将一组值发送到同一个节点，并且这些值加起来超过可用内存，则即使是偏差较小的查询也会失败。节点确实能够将记录溢出到磁盘，但是溢出会减慢查询的执行速度，可能不足以防止查询失败。

Athena 尝试重新排序联接，使用较大的关系作为探测端，将较小的关系用作构建端。但是，由于 Athena 不管理表中的数据，因此它的信息有限，通常必须假设第一个表更大，第二个表更小。

使用基于等式的联接条件编写联接时，假设 JOIN 关键字左侧的表是探测端，右侧的表是构建端。确保右侧的表（即构建端）是表格中较小的一个。如果无法将联接的构建端缩小到足以放入内存，请考虑运行多个查询来联接构建表的子集。

其他联接类型

具有复杂联接条件的查询（例如，使用 LIKE、> 或其他运算符的查询）通常对计算要求很高。在最糟糕的情况下，必须将联接一侧的每条记录与联接另一侧的每条记录进行比较。由于执行时间随记录数的平方而增长，因此此类查询有可能超过最大执行时间。

要提前了解 Athena 将如何执行您的查询，您可以使用 EXPLAIN 语句。有关更多信息，请参阅 在 Athena 中使用 EXPLAIN 和 EXPLAIN ANALYZE和 了解 Athena EXPLAIN 语句结果。

优化窗口函数

由于窗口函数是资源密集型操作，因此它们可能会使查询运行缓慢甚至失败，并显示消息查询在此缩放系数耗尽了资源。窗口函数将它们操作的所有记录保存在内存中，以便计算其结果。当窗口非常大时，窗口函数可能会耗尽内存。

要确保查询在可用内存限制内运行，请减小窗口函数操作的窗口的大小。为此，您可以添加 PARTITIONED BY 子句或缩小现有分区子句的范围。

改用非窗口函数

有时，使用窗口函数的查询可以在没有窗口函数的情况下重写。例如，您可以使用 ORDER BY 和 LIMIT，而不是使用 row_number 来查找前 N 条记录。您可以使用 max_by、min_by 和任意聚合函数，而不是使用 row_number 或 rank 删除重复记录。

例如，假设您有一个包含来自传感器的更新的数据集。传感器会定期报告其电池状态，并包含一些元数据（例如位置）。如果您想知道每个传感器的上次电池状态及其位置，可以使用以下查询：

```
SELECT sensor_id,
       arbitrary(location) AS location,
       max_by(battery_status, updated_at) AS battery_status
FROM sensor_readings
GROUP BY sensor_id
```

由于每条记录的位置等元数据都相同，因此您可以使用 arbitrary 函数从组中选取任何值。

要获取最后电池状态，您可以使用 max_by 函数。max_by 函数从发现另一列的最大值的记录中选取一列的值。在这种情况下，它将返回记录的电池状态以及组内最后一次更新时间。与使用窗口函数的等效查询相比，此查询运行速度更快，占用的内存也更少。

优化聚合

当 Athena 执行聚合时，它会使用 GROUP BY 子句中的列在工作节点之间分配记录。为了尽可能高效地将记录与组进行匹配，节点会尝试将记录保存在内存中，但必要时会将其溢出到磁盘。

避免在 GROUP BY 子句中包含多余的列也是个不错的做法。由于较少的列需要更少的内存，因此使用较少的列描述组的查询效率更高。数字列使用的内存也比字符串少。例如，当您聚合同时具有数字类别 ID 和类别名称的数据集时，在 GROUP BY 子句中仅使用类别 ID 列。

有时，查询会在 GROUP BY 子句中包含列，以避免列必须是 GROUP BY 子句的一部分或是聚合表达式。如果不遵守此规则，您可能会收到如下错误消息：

EXPRESSION_NOT_AGGREGATE：第 1:8 行：'category' 必须是聚合表达式或出现在 GROUP BY 子句中

为避免在 GROUP BY 子句中添加冗余列，可以使用任意函数，如下例所示。

```
SELECT country_id,
       arbitrary(country_name) AS country_name,
       COUNT(*) AS city_count
FROM world_cities
GROUP BY country_id
```

ARBITRARY 函数从组中返回任意值。当您知道组中所有记录的某列值相同，但该值不能标识该组时，该函数很有用。

优化前 N 个查询

ORDER BY 子句按排序顺序返回查询结果。Athena 使用分布式排序在多个节点上并行运行排序操作。

如果您不严格要求对结果进行排序，请避免添加 ORDER BY 子句。此外，如果不是严格必需的，请避免将 ORDER BY 添加到内部查询中。在许多情况下，查询计划程序可以删除冗余排序，但这并不能保证。此规则的一个例外情况是，如果内部查询正在执行前 N 个操作，例如查找 N 个最新或 N 个最常见的值。

当 Athena 看到 LIMIT 与 ORDER BY 一起使用时，它会知道您正在运行前 N 个查询，因此会相应地使用专用操作。

> **Note**
>
> 尽管 Athena 也可以经常使用前 N 个来检测 `row_number` 等窗口函数，但我们建议使用 `ORDER BY` 和 `LIMIT` 的简单版本。有关更多信息，请参阅 优化窗口函数。

仅包含必需列

如果您并不严格需要某一列，请不要将其包含在查询中。查询需要处理的数据越少，运行速度就越快。这样既可以减少所需的内存量，也减少了必须在节点之间发送的数据量。如果您使用的是列式文件格式，则减少列数也会减少从 Amazon S3 读取的数据量。

Athena 对结果中的列数没有具体限制，但是查询的执行方式限制了可能的列组合大小。列的组合大小包括其名称和类型。

例如，以下错误是由超出关系描述符大小限制的关系引起的：

GENERIC_INTERNAL_ERROR: io.airlift.bytecode.CompilationException

要解决此问题，请减少查询中的列数，或创建子查询并使用可降低检索数据量的 `JOIN`。如果您的查询在最外层的查询中执行 `SELECT *`，则应将 `*` 更改为仅包含所需列的列表。

使用近似值优化查询

Athena 支持近似聚合函数，用于计算不同值、最常见的值、百分位数（包括近似中位数）和创建直方图。当不需要精确值时，请使用这些函数。

与 `COUNT(DISTINCT col)` 操作不同，approx_distinct 使用的内存要少得多，运行速度更快。同样，使用 numeric_histogram 代替直方图会使用近似法，因此内存更少。

优化 LIKE

您可以使用 `LIKE` 来查找匹配的字符串，但是对于长字符串，这将占用大量计算资源。在大多数情况下，regexp_like 函数是一种更快的替代方案，而且还提供了更大的灵活性。

通常，您可以通过锚定要查找的子字符串来优化搜索。例如，如果您正在寻找前缀，最好使用 '*substr*%' 而不是 '%*substr*%'。或者，如果您使用的是 `regexp_like`，请使用 '^*substr*'。

使用 UNION ALL 代替 UNION

`UNION ALL` 和 `UNION` 还有两种方法可以将两个查询的结果合并为一个结果。`UNION ALL` 将第一个查询的记录与第二个查询的记录连接起来，`UNION` 执行相同的操作，同时也会删除重复的记录。`UNION`

需要处理所有记录并找到重复项，这需要占用大量内存和计算，但 UNION ALL 操作速度相对较快。除非需要对记录进行重复数据删除，否则请使用 UNION ALL 以获得最佳性能。

对大型结果集使用 UNLOAD

当查询的结果预计会很大（例如，成千上万行或更多）时，请使用 UNLOAD 导出结果。在大多数情况下，这比运行常规查询要快，而且使用 UNLOAD 还可以让您更好地控制输出。

查询执行完毕后，Athena 会将结果作为单个未压缩的 CSV 文件存储在 Amazon S3 上。这需要比 UNLOAD 更长的时间，这不仅是因为结果未压缩，还因为操作无法并行化。相比之下，UNLOAD 直接从 Worker 节点写入结果，并充分利用计算集群的并行度。此外，您可以配置 UNLOAD 为以压缩格式和其他文件格式（例如 JSON 和 Parquet）写入结果。

有关更多信息，请参阅 UNLOAD。

使用 CTAS 或 Glue ETL 来具体化常用的聚合

'Materializing' 查询是一种通过存储预先计算的复杂查询结果（例如聚合和联接），以便在后续查询中重复使用，从而提高查询性能的方法。

如果您的许多查询包含相同的联接和聚合，则可以将常见子查询具体化为新表，然后对该表运行查询。您可以使用 从查询结果创建表（CTAS） 或专用 ETL 工具（如 Glue ETL）创建新表。

例如，假设您有一个控制面板，其中包含显示订单数据集不同方面的小部件。每个小部件都有自己的查询，但所有查询都共享相同的联接和筛选器。订单表与订单项目表联接，并且有一个筛选器仅显示最近三个月。如果您确定了这些查询的常用功能，则可以创建小部件可以使用的新表。这样可以减少重复并提高性能。缺点是必须使新表保持最新状态。

重复使用查询结果

同一个查询通常会在短时间内多次运行。例如，当多人打开同一个数据控制面板时，可能会发生这种情况。运行查询时，您可以让 Athena 重复使用之前计算的结果。您可以指定要重复使用的结果的最大期限。如果之前在该时间范围内运行过相同的查询，Athena 将返回这些结果，而不会再次运行查询。有关更多信息，请参阅《Amazon Athena 用户指南》中的 重用查询结果，以及 AWS 大数据博客中的通过 Amazon Athena 查询结果重复使用来降低成本并提高查询性能。

查询优化技术

性能不仅取决于查询，还取决于数据集的组织方式及其使用的文件格式和压缩。

对您的数据进行分区

分区可将您的表格分成多个部分，并根据日期、国家或地区等属性将相关数据保存在一起。分区键充当虚拟列。您可以在创建表时定义分区键，并使用它们来筛选查询。对分区键列进行筛选时，只读取来自匹配分区的数据。例如，如果您的数据集按日期分区，并且您的查询的筛选条件仅与上周匹配，则只读取上周的数据。有关分区的更多信息，请参阅 在 Athena 中对数据进行分区。

选择支持您的查询的分区键

由于分区会对查询性能产生重大影响，因此在设计数据集和表时，请务必考虑如何谨慎分区。分区键过多会导致数据集碎片化，文件过多或过小。相反，分区键太少或根本没有分区会导致查询扫描的数据超出必要的范围。

避免针对罕见的查询进行优化

一个好的策略是针对最常见的查询进行优化，避免针对罕见的查询进行优化。例如，如果您的查询以天为单位的时间跨度，即使某些查询筛选到该级别，也不要按小时进行分区。如果您的数据具有精细的时间戳列，则按小时筛选的罕见查询可以使用时间戳列。即使罕见案例扫描的数据比必要的要多一点，但为了极少数情况而降低整体性能通常也不是一个很好的权衡方案。

要减少查询必须扫描的数据量，从而提高性能，请使用列式文件格式并对记录进行排序。与其按小时分区，不如按时间戳对记录进行排序。对于时间范围较短的查询，按时间戳排序几乎与按小时分区一样有效。此外，按时间戳排序通常不会损害以天为单位的时间窗口的查询性能。有关更多信息，请参阅 使用列式文件格式。

请注意，如果所有分区键上都有谓词，则对包含数万个分区的表进行查询的性能会更好。这是为最常见的查询设计分区方案的另一个原因。有关更多信息，请参阅 按等式查询分区。

使用分区投影

分区投影是 Athena 的一项功能，它不是将分区信息存储在 AWS Glue Data Catalog 中，而是作为规则存储在 AWS Glue 中表的属性中。当 Athena 计划对配置了分区投影的表进行查询时，它会读取该表的分区投影规则。Athena 根据查询和规则计算要在内存中读取的分区，而不是在 AWS Glue Data Catalog 中查找分区。

除了简化分区管理外，分区投影还可以提高具有大量分区的数据集的性能。当查询包含范围而不是分区键的特定值时，分区越多，在目录中查找匹配的分区所需的时间就越长。使用分区投影，无需进入目录即可在内存中计算筛选器，而且速度可以快得多。

在某些情况下，分区投影可能会导致性能降低。例如，当表为“稀疏表”时。稀疏表不包含分区投影配置所描述的分区键值的每个排列的数据。对于稀疏表，通过查询计算得出的分区集和分区投影配置都会在 Amazon S3 上列出，即使它们没有数据。

使用分区投影时，请确保在所有分区键上都包含谓词。缩小可能值的范围，以避免不必要的 Amazon S3 列表。假设一个分区键的范围为一百万个值，而一个查询对该分区键没有任何筛选器。要运行查询，Athena 必须执行至少一百万个 Amazon S3 列表操作。无论您是使用分区投影还是在目录中存储分区信息，查询特定值时查询速度最快。有关更多信息，请参阅 按等式查询分区。

在为分区投影配置表时，请确保您指定的范围合理。如果查询不包含分区键的谓词，则使用该键范围内的所有值。如果您的数据集是在特定日期创建的，请使用该日期作为任何日期范围的起点。使用 NOW 作为日期范围的结束日期。避免使用具有大量值的数值范围，并考虑改用注入类型。

更多有关分区投影的信息，请参阅 使用 Amazon Athena 分区投影。

使用分区索引

分区索引是 AWS Glue Data Catalog 中的一项功能，它可以提高具有大量分区的表的分区查找性能。

目录中的分区列表就像关系数据库中的表。该表包含用于分区键的列，还有一列用于分区位置。查询分区表时，将通过扫描该表来查找分区位置。

与关系数据库一样，您可以通过添加索引来提高查询性能。您可以添加多个索引以支持不同的查询模式。AWS Glue Data Catalog 分区索引同时支持等式和比较运算符，例如 >、>= 和 <，并与 AND 运算符组合使用。有关更多信息，请参阅《AWS Glue 开发人员指南》中的在 AWS Glue 中使用分区索引和 AWS 大数据博客中的使用 AWS Glue Data Catalog 分区索引提高 Amazon Athena 查询性能。

始终使用 STRING 作为分区键的类型

在查询分区键时，请记住：Athena 要求分区键的类型必须为 STRING 类型才能下推分区筛选到 AWS Glue。如果分区数量不小，则使用其他类型可能会导致性能降低。如果您的分区键值类似于日期或类似数字，请在查询中将其转换为相应的类型。

删除旧的和空的分区

如果您从 Amazon S3 上的分区中移除数据（例如，使用 Amazon S3 生命周期），则还应从 AWS Glue Data Catalog 中删除该分区条目。在查询计划期间，与查询匹配的任何分区都会在 Amazon S3 上列出。如果您有许多空分区，则列出这些分区的开销可能会造成不利影响。

另外，如果您有成千上万个分区，可以考虑删除旧数据不再相关的分区元数据。例如，如果查询从不查看超过一年的数据，则可以定期删除旧分区的分区元数据。如果分区的数量增加到数万个，则移除未使用的分区可以加快所有分区键上都不包含谓词的查询速度。有关在查询中包含所有分区键的谓词的信息，请参阅 按等式查询分区。

按等式查询分区

由于可以直接加载分区元数据，因此在所有分区键上包含等式谓词的查询运行得更快。避免查询中一个或多个分区键没有谓词，或者谓词选择一定范围的值。对于此类查询，必须筛选所有分区的列表以找到匹配的值。对于大多数表来说，开销很小，但是对于具有成千上万或更多分区的表，开销可能会变得很大。

如果无法重写查询以按等式筛选分区，则可以尝试分区投影。有关更多信息，请参阅 使用分区投影。

避免使用 MSCK REPAIR TABLE 进行分区维护

由于 MSCK REPAIR TABLE 可能需要很长时间才能运行，只能添加新分区，而不会删除旧分区，因此这不是管理分区的有效方法（请参阅 注意事项和限制）。

使用 AWS Glue Data Catalog API、ALTER TABLE ADD PARTITION 或 AWS Glue 爬网程序以更好地手动管理分区。作为替代方案，您可以使用分区投影，这样就无需管理分区。有关更多信息，请参阅使用 Amazon Athena 分区投影。

验证您的查询是否与分区方案兼容

您可以使用 EXPLAIN 语句提前检查查询将扫描哪些分区。在查询前面加上 EXPLAIN 关键字，然后在 EXPLAIN 输出底部附近查找每个表的源片段（例如 Fragment 2 [SOURCE]）。查找右侧被定义为分区键的分配。下面的行包括运行查询时将扫描的该分区键的所有值的列表。

例如，假设您在表上有一个 dt 分区键的查询，并在查询前面加上 EXPLAIN。如果查询中的值是日期，并且筛选器选择了三天的范围，则 EXPLAIN 输出可能如下所示：

```
dt := dt:string:PARTITION_KEY
    :: [[2023-06-11], [2023-06-12], [2023-06-13]]
```

EXPLAIN 输出显示计划程序为该分区键找到了三个与查询相匹配的值。同时还会显示这些值是什么。有关使用 EXPLAIN 的更多信息，请参阅 在 Athena 中使用 EXPLAIN 和 EXPLAIN ANALYZE 和 了解 Athena EXPLAIN 语句结果。

使用列式文件格式

Parquet 和 ORC 等列式文件格式专为分布式分析工作负载而设计。它们按列而不是按行组织数据。以列式格式组织数据具有以下优点：

- 仅加载查询所需的列

- 减少了需要加载的总数据量
- 列值存储在一起，因此可以有效地压缩数据
- 文件可以包含允许引擎跳过加载不需要的数据的元数据

举例说明如何使用文件元数据，文件元数据可以包含有关数据页面中最小值和最大值的信息。如果查询的值不在元数据中注明的范围内，则可以跳过该页面。

使用此元数据来提高性能的一种方法是确保对文件中的数据进行排序。例如，假设您有查询来查找短时间段内 `created_at` 条目所在的记录。如果您的数据按 `created_at` 列排序，Athena 可以使用文件元数据中的最小值和最大值来跳过数据文件中不需要的部分。

使用列式文件格式时，请确保文件不要太小。如 避免文件过多 中所述，包含许多小文件的数据集会导致性能问题。对于列式文件格式尤其如此。对于小文件，列式文件格式的开销大于好处。

请注意，Parquet 和 ORC 在内部按行组（Parquet）和条带（ORC）进行组织。行组的默认大小为 128MB，条带的默认大小为 64MB。如果您有许多列，则可以增加行组和条带大小以提高性能。不建议将行组或条带大小减小到其默认值以下。

要将其他数据格式转换为 Parquet 或 ORC，您可以使用 AWS Glue ETL 或 Athena。有关使用 Athena for ETL 的更多信息，请参阅 将 CTAS 和 INSERT INTO 用于 ETL 和数据分析。

压缩数据

Athena 支持多种压缩格式。查询压缩数据更快，也更便宜，因为您需要为解压缩前扫描的字节数付费。

gzip 格式提供了良好的压缩比，并且在其他工具和服务中具有广泛的支持。zstd（Zstandard）格式是一种较新的压缩格式，在性能和压缩比之间取得了很好的平衡。

压缩 JSON 和 CSV 数据等文本文件时，请尽量在文件数量和文件大小之间取得平衡。大多数压缩格式都要求读取器从一开始就读取文件。这意味着通常不能并行处理压缩的文本文件。在查询处理过程中，大型未压缩文件通常在工作线程之间拆分，以实现更高的并行度；但是对于大多数压缩格式来说，这是不可能的。

如 避免文件过多 中所述，文件最好不要太多也不要太少。由于文件数量是可以处理查询的工作线程数量的限制，因此对于压缩的文件尤其如此。

有关在 Athena 中使用压缩的更多信息，请参阅 Athena 压缩支持。

使用存储桶查找具有高基数的键

存储桶是一种根据其中一列的值将记录分发到单独文件中的技术。这样可以确保所有具有相同值的记录都位于同一个文件中。当您的键具有高基数并且您的许多查询都查找该键的特定值时，存储桶很有用。

例如，假设您查询一组记录寻求特定用户。如果数据按用户 ID 划分存储桶，Athena 会事先知道哪些文件包含特定 ID 的记录，哪些文件没有。这使得 Athena 能够只读取可能包含该 ID 的文件，从而大大减少了读取的数据量。它还缩短了在数据中搜索特定 ID 所需的计算时间。

存储桶的缺点

当查询经常在存储数据的列中搜索多个值时，存储桶的价值就会降低。查询的值越多，必须读取所有或大多数文件的可能性就越大。例如，如果您有三个存储桶，并且查询查找三个不同的值，则可能必须读取所有文件。当查询查找单个值时，存储桶效果最好。

有关更多信息，请参阅 在 Athena 中进行分区和分桶。

避免文件过多

由许多小文件组成的数据集会导致整体查询性能不佳。当 Athena 计划查询时，它会列出所有分区位置，这需要时间。处理和请求每个文件也会产生计算开销。因此，从 Amazon S3 加载单个更大的文件要比从许多较小的文件加载相同记录要快。

在极端情况下，您可能会遇到 Amazon S3 服务限制。Amazon S3 每秒最多支持向单个索引分区发出 5500 个请求。最初，存储桶被视为单个索引分区，但是随着请求负载的增加，可以将其拆分为多个索引分区。

Amazon S3 着眼于请求模式，并根据键前缀进行拆分。如果您的数据集包含成千上万个文件，则来自 Athena 的请求可能会超过请求限额。即使文件较少，如果对同一个数据集进行多个并发查询，也可能超过限额。访问相同文件的其他应用程序可能会占请求总数。

超过请求速率 `limit` 时，Amazon S3 会返回以下错误。此错误包含在 Athena 中查询的状态信息中。

减速：请降低请求速率

要排除故障，首先要确定错误是由单个查询引起的，还是由读取相同文件的多个查询引起的。如果是后者，请协调查询的运行，这样它们就不会同时运行。为此，请在应用程序中添加排队机制甚至重试。

如果运行单个查询触发错误，请尝试合并数据文件或修改查询以减少读取的文件。合并小文件的最佳时机是在写入它们之前。为此，请考虑以下技术：

- 将写入文件的过程更改为写入更大的文件。例如，可以在写入记录之前将其缓冲更长时间。

- 将文件放在 Amazon S3 上的某个位置，然后使用像 Glue ETL 这样的工具将它们合并成更大的文件。然后，将较大的文件移动到表格所指向的位置。有关更多信息，请参阅《AWS Glue 开发人员指南》中的读取较大组中的输入文件和 AWS re:Post 知识中心中的如何配置 AWS Glue ETL 任务以输出较大文件？。
- 减少分区键的数量。当分区键过多时，每个分区可能只有几个记录，从而导致小文件数量过多。有关决定创建哪些分区的信息，请参阅 选择支持您的查询的分区键。

避免在分区之外设置额外的存储层次结构

为避免查询计划开销，请在每个分区位置以扁平结构存储文件。请勿使用任何其他目录层次结构。

当 Athena 计划查询时，它会列出与该查询匹配的所有分区中的所有文件。尽管 Amazon S3 本身没有目录，但惯例是将 / 正斜杠解释为目录分隔符。当 Athena 列出分区位置时，它会递归列出找到的任何目录。当分区内的文件按层次结构组织时，会出现多轮列表。

当所有文件都直接位于分区位置时，大多数时候只需要执行一个列表操作。但是，如果分区中有超过 1000 个文件，则需要进行多次顺序列表操作，因为 Amazon S3 每次列表操作仅返回 1000 个对象。一个分区中有 1000 个以上的文件还会造成其他更严重的性能问题。有关更多信息，请参阅 避免文件过多。

仅在必要时使用 SymlinkTextInputFormat

使用 SymlinkTextInputFormat 技术可以解决表的文件没有整齐地组织到分区中的情况。例如，当所有文件都使用相同前缀或具有不同架构的文件位于同一位置时，符号链接可能很有用。

但是，使用符号链接会增加查询执行的间接级别。这些间接级别会影响整体性能。必须读取符号链接文件，并且必须列出它们定义的位置。这会增加到 Amazon S3 的多次往返行程，而通常的 Hive 表不需要这些往返行程。总之，只有在没有更好的选项（例如重组文件）时，才应使用 `SymlinkTextInputFormat`。

其他资源

有关 Athena 中性能优化的更多信息，请参考以下资源：

- 阅读 AWS 大数据博客贴文 Amazon Athena 的十大性能优化技巧
- 有关使用谓词下推来提高联合查询性能的文章，请参阅 AWS 大数据博客中的在 Amazon Athena 中使用谓词下推改善联合查询。
- 有关 Athena 查询引擎性能优化的文章，请参阅 AWS Big Data Blog 中的 Run queries 3x faster with up to 70% cost savings on the latest Amazon Athena engine。

- 阅读 AWS 大数据博客中的其他 Athena 相关贴文
- 使用 Amazon Athena 标签在 AWS re:Post 上提问
- 在 AWS 知识中心中查询 Athena 主题
- 联系 AWS Support (在 AWS Management Console 中，单击 Support(支持)、Support Center(支持中心))

防止 Amazon S3 节流

节流指限制服务、应用程序或系统使用速率的过程。在 AWS 中，您可以使用节流防止过度使用 Amazon S3 服务，并提高 Amazon S3 对所有用户的可用性和响应能力。但是，由于节流会限制在 Amazon S3 中传入或传出数据的速率，因此必须考虑防止您的交互受到节流。

减少服务级别节流

为了避免服务级别下的 Amazon S3 节流，您可以监控使用情况并调整服务限额，也可以使用分区等特定方法。下面是一些可能导致节流的一些情况：

- 超过账户的 API 请求限制 - Amazon S3 具有基于账户类型和使用情况的默认 API 请求限制。如果超过单个对象每秒的最大请求数，则可能会对您的请求进行节流，以防 Amazon S3 服务过载。
- 数据分区不足 - 如果您没有正确分区数据并传输大量数据，Amazon S3 可能会对您的请求进行节流。有关分区的更多信息，请参阅本文档中的 使用分区 节。
- 大量小对象 - 如有可能，避免生成大量小文件。Amazon S3 的每个分区前缀每秒最多 5500 个 GET 请求，您的 Athena 查询也具有同样的限制。如果您需要在单一查询中扫描数百万个小对象，则 Amazon S3 可能会对您的查询进行节流。

要避免过度扫描，可使用 AWS Glue ETL 定期压缩文件，或者对表进行分区并添加分区键筛选条件。有关详细信息，请参阅以下资源：

- 如何配置 AWS Glue ETL 作业以输出较大的文件？（AWS 知识中心）
- 以较大的组读取输入文件（《AWS Glue 开发人员指南》）

优化表

如果您遇到节流问题，则需要对数据进行结构化。尽管 Amazon S3 可以处理大量数据，但由于数据的结构方式，有时也会发生节流。

以下各节就如何在 Amazon S3 中结构化数据以免出现节流问题提供了一些建议。

使用分区

您可以使用分区通过限制在任何给定时间必须访问的数据量来减少节流。通过对特定列中的数据进行分区，您可以将请求均匀地分配到多个对象，并减少单一对象的请求数量。可通过减少扫描的数据量提高查询性能并降低成本。

创建表时，您可以定义分区，来充当虚拟列。要在 CREATE TABLE 语句中创建包含分区的表，可使用 PARTITIONED BY (*column_name data_type*) 子句定义用于对数据进行分区的键。

要限制查询扫描的分区，可以在查询的 WHERE 子句中将其指定为谓词。因此，经常用作筛选条件的列是用于分区的理想候选项。常见做法是根据时间间隔对数据进行分区，这可能会导致多层分区方案。

请注意，分区也有成本。当您增加表中的分区数时，检索和处理分区元数据所需的时间也会增加。因此，过度分区可能会抵消以更明智的方式进行分区带来的好处。如果您的数据严重偏向于一个分区值，并且大多数查询都使用该值，则可能会产生额外开销。

有关在 Athena 中进行分区的更多信息，请参阅 什么是分区？。

对数据进行分桶

对数据进行分区的另一种方法是对单一分区中的数据进行分桶。进行分桶时，可指定包含要分组到一起的行的一列或多列。然后，将这些行放入多个存储桶中。这样，只需查询必须读取的存储桶即可，从而减少必须扫描的数据行数。

在选择要用于分桶的列时，选择具有高基数（即具有许多不同值）、分布均匀且经常用于筛选数据的列。例如，ID 列等主键就是用于分桶的理想列。

有关在 Athena 中分桶的更多信息，请参阅 什么是分桶？。

使用 AWS Glue 分区索引

您可以使用 AWS Glue 分区索引根据一个或多个分区值对表中的数据进行组织。AWS Glue 分区索引可以减少数据传输次数、数据处理量和查询处理时间。

AWS Glue 分区索引是一个元数据文件，其中包含有关表中分区的信息，包括分区键及其值。分区索引存储在 Amazon S3 存储桶中，并在向表中添加新分区时由 AWS Glue 自动更新。

当存在 AWS Glue 分区索引时，查询会尝试获取一部分的分区，而不是加载表中的所有分区。将仅对与查询有关的这部分数据运行查询。

在 AWS Glue 中创建表时，可以基于表中定义的任意分区键组合创建分区索引。在表中创建了一个或多个分区索引后，必须向表中添加用于启用分区筛选的属性。然后，您可以从 Athena 查询表。

有关在 AWS Glue 中创建分区索引的信息，请参阅《AWS Glue 开发人员指南》中的在 AWS Glue 中使用分区索引。有关添加表属性以启用分区筛选的信息，请参阅 AWS Glue 分区索引和筛选。

使用数据压缩和文件拆分

如果文件大小理想或者可以按逻辑组对其进行拆分，则数据压缩可以显著加快查询速度。通常，较高的压缩比需要更多的 CPU 周期才能压缩和解压缩数据。对于 Athena，建议使用 Apache Parquet 或 Apache ORC，以默认压缩数据。有关 Athena 中数据压缩的信息，请参阅 Athena 压缩支持。

可通过拆分文件允许 Athena 将读取单个文件的任务分配给多个读取器，从而提高并行度。如果单个文件不可拆分，则只有一个读取器可以读取该文件，而其他读取器处于空闲状态。Apache Parquet 和 Apache ORC 也支持可拆分文件。

使用经过优化的列式数据存储

如果将数据转换为列式格式，则会显著提高 Athena 查询性能。生成列式文件时，要考虑的一种优化方法是根据分区键对数据进行排序。

Apache Parquet 和 Apache ORC 是常用的开源列式数据存储。有关将现有 Amazon S3 数据来源转换为其中一种格式的信息，请参阅 转换为列式格式。

使用较大的 Parquet 块大小或 ORC 条带大小

Parquet 和 ORC 具有诸多数据存储参数，您可以调整这些参数以进行优化。在 Parquet 中，您可以针对块大小进行优化。在 ORC 中，您可以针对条带大小进行优化。数据块或条带越大，可在其中存储的行越多。默认情况下，Parquet 块大小为 128MB，ORC 条带大小为 64MB。

如果 ORC 条带小于 8MB（`hive.orc.max_buffer_size` 的默认值），Athena 会读取整个 ORC 条带。这是 Athena 在列选择性和针对较小条带的每秒输入 / 输出操作之间做出的权衡。

如果表的列数非常多，则较小的数据块或条带大小可能会导致扫描的数据量超出必要范围。在此类情况下，较大的块大小可能更高效。

使用 ORC 处理复杂类型

目前，当您查询在 Parquet 中存储的具有复杂数据类型（例如，`array`、`map` 或 `struct`）的列时，Athena 将读取整行数据，而不是有选择性地仅读取指定列。这是 Athena 中的已知问题。要解决该问题，考虑使用 ORC。

选择压缩算法

可配置的另一个参数是数据块的压缩算法。有关 Athena 中 Parquet 和 ORC 支持的压缩算法的信息，请参阅 Athena 压缩支持。

有关优化 Athena 列式存储格式的更多信息，请参阅 AWS 大数据博客文章 Amazon Athena 的十大性能调整技巧中的“优化列式数据存储生成”一节。

使用 Iceberg 表

Apache Iceberg 是一种适用于超大型分析数据集的开放表格式，专为优化 Amazon S3 使用而设计。可使用 Iceberg 表帮助减少 Amazon S3 节流。

Iceberg 表具有以下优势：

- 可基于一列或多列对 Iceberg 表进行分区。这将优化数据访问并减少查询必须扫描的数据量。
- 由于 Iceberg 对象存储模式可优化 Iceberg 表以与 Amazon S3 配合使用，因此它可以处理大量数据和繁重的查询工作负载。
- 对象存储模式下的 Iceberg 表具有可扩展性、容错性和耐用性，这有助于减少节流。
- ACID 事务支持意味着多个用户可以原子方式添加和删除 Amazon S3 对象。

有关 Apache Iceberg 的更多信息，请参阅 Apache Iceberg。有关在 Athena 中使用 Apache Iceberg 表的更多信息，请参阅使用 Iceberg 表。

优化查询

使用本节中的建议优化 Athena SQL 查询。

在 ORDER BY 子句中使用 LIMIT

`ORDER BY` 子句按排序顺序返回数据。这要求 Athena 将所有数据行发送到单一 Worker 节点，然后对这些行进行排序。此类查询可能会运行很长时间，甚至会失败。

为了提高查询效率，查看前或后 *N* 个值，然后还要使用 `LIMIT` 子句。这将显著降低排序成本，因为会将排序和限制推送到各个 Worker 节点而不是单一 Worker。

优化 JOIN 子句

当您联接两个表时，Athena 会将右侧的表分配给 Worker 节点，然后流式处理左侧的表以执行联接。

因此，在联接的左侧指定较大的表，在联接的右侧指定较小的表。这样可减少 Athena 所用的内存并降低查询运行延迟。

同时注意以下几点：

- 使用多个 `JOIN` 命令时，按从大到小顺序指定表。
- 除非查询需要，否则避免使用交叉联接。

优化 GROUP BY 子句

GROUP BY 运算符根据 GROUP BY 列将行分配给 Worker 节点。这些列将在内存中引用，并在载入行时对值进行比较。当 GROUP BY 列匹配时，这些值会聚合在一起。考虑到此过程的工作方式，建议按基数从高到低对列进行排序。

使用数字代替字符串

由于与字符串相比，数字所需的内存更少且处理速度更快，因此尽可能使用数字代替字符串。

限制列数

要减少存储数据所需的总内存量，限制在 SELECT 语句中指定的列数。

使用正则表达式代替 LIKE

在大型字符串中包含 `LIKE '%string%'` 等子句的查询的计算量可能非常大。在字符串列中筛选多个值时，改用 regexp_like() 函数和正则表达式。这在您比较一长串值时特别有用。

使用 LIMIT 子句

在运行查询时，使用 LIMIT 子句仅返回所需的列，而不是选择所有列。这减小了通过查询执行管道处理的数据集的大小。当查询包含大量基于字符串的列的表时，建议使用 LIMIT 子句。当您对任何查询执行多个联接或聚合时，也建议使用 LIMIT 子句。

其他 资源

《Amazon Simple Storage Service 用户指南》中的最佳实践设计模式：优化 Amazon S3 性能。

Athena 中的性能优化

Athena 压缩支持

主题

Athena 支持多种用于读取和写入数据的压缩格式，例如从使用多种压缩格式的表中进行读取。例如，当某些 Parquet 文件使用 Snappy 压缩而其他 Parquet 文件使用 GZIP 压缩时，Athena 可以成功读取使用 Parquet 文件格式的表中的数据。同样的原则适用于 ORC、文本文件和 JSON 存储格式。

Athena 支持以下压缩格式：

- BZIP2 – 使用 Burrows-Wheeler 算法的格式。
- DEFLATE – 基于 LZSS 和 Huffman 编码的压缩算法。Deflate 仅与 Avro 文件格式相关。
- GZIP – 基于 Deflate 的压缩算法。对于 Athena 引擎版本 2 和 3 中的 Hive 表，以及 Athena 引擎版本 2 中的 Iceberg 表，GZIP 是 Parquet 和文本文件存储格式文件的默认写入压缩格式。不支持 `tar.gz` 格式的文件。
- LZ4 – 属于 Lempel-Ziv 77 (LZ7) 系列，并且也侧重于压缩和解压缩速度，而非对数据的最大限度压缩。LZ4 具有以下成帧格式：
  - LZ4 Raw/Unframed – LZ4 数据块压缩格式的标准无帧实现。有关更多信息，请参阅 GitHub 上的 LZ4 数据块格式说明。
  - LZ4 Framed – 常见的 LZ4 成帧实现。有关更多信息，请参阅 GitHub 上的 LZ4 帧格式说明。
  - LZ4 Hadoop-Compatible – LZ4 的 Apache Hadoop 实现。此实现使用 BlockCompressorStream.java 类封装 LZ4 压缩。
- LZO – 使用 Lempel–Ziv–Oberhumer 算法的格式，该算法侧重于高速压缩和解压缩，而非对数据的最大限度压缩。LZO 具有两个实现：
  - Standard LZO – 有关更多信息，请参阅 Oberhumer 网站上的 LZO 摘要。
  - LZO Hadoop-Compatible – 此实现使用 BlockCompressorStream.java 类封装 LZO 算法。
- SNAPPY – 属于 Lempel-Ziv 77 (LZ7) 系列的压缩算法。Snappy 侧重于高速压缩和解压速度，而非对数据的最大限度压缩。
- ZLIB – 基于 Deflate，ZLIB 是 ORC 数据存储格式文件的默认写入压缩格式。有关更多信息，请参阅 GitHub 上的 zlib 页面。
- ZSTD – Zstandard 实时数据压缩算法是一种具有高压缩率的快速压缩算法。Zstandard（ZSTD）库作为使用 BSD 许可证的开源软件提供。ZSTD 是 Iceberg 表的默认压缩格式。在写入 ZSTD 压缩数据时，Athena 默认使用 ZSTD 压缩级别 3。有关在 Athena 中使用 ZSTD 压缩级别的更多信息，请参阅 在 Athena 中使用 ZSTD 压缩级别。

指定压缩格式

写入 CREATE TABLE 或 CTAS 语句时，您可以指定压缩属性，该属性可指定 Athena 写入这些表时要使用的压缩类型。

- 对于 CTAS，请参阅 CTAS 表属性。有关示例，请参阅CTAS 查询的示例。
- 对于 CREATE TABLE，请参阅 ALTER TABLE SET TBLPROPERTIES 以获取压缩表属性列表。

指定无压缩

CREATE TABLE 语句支持写入未压缩的文件。要写入未压缩的文件，请使用以下语法：

- CREATE TABLE（文本文件或 JSON）– 在 TBLPROPERTIES 中，请指定 `write.compression = NONE`。
- CREATE TABLE (Parquet) – 在 TBLPROPERTIES 中，请指定 `parquet.compression = UNCOMPRESSED`。
- CREATE TABLE (ORC) – 在 TBLPROPERTIES 中，请指定 `orc.compress = NONE`。

注释和资源

- 目前，Athena 无法识别大写文件扩展名，例如 `.GZ` 或 `.BZIP2`。避免使用包含大写文件扩展名的数据集，或将数据文件扩展名重命名为小写。
- 对于 CSV、TSV 和 JSON 格式的数据，Athena 根据文件扩展名确定压缩类型。如果不存在文件扩展名，则 Athena 将数据视为未压缩的纯文本。如果您的数据已压缩，请确保文件名包含压缩扩展名，例如 `gz`。
- 不支持 ZIP 文件格式。
- 对于从 Athena 查询 Amazon Data Firehose 日志，支持的格式包括 GZIP 压缩文件或采用 SNAPPY 压缩的 ORC 文件。
- 有关使用压缩的更多信息，请参阅 AWS 大数据博客文章 Top 10 performance tuning tips for Amazon Athena（Amazon Athena 的 10 大性能优化技巧）中的第 3 部分（“压缩和拆分文件”）。

不同文件格式支持的 Hive 表压缩

Athena 中的 Hive 压缩支持取决于引擎版本。

Athena 引擎版本 3 中的 Hive 压缩支持

下表总结了对于 Apache Hive 存储文件格式，Athena 引擎版本 3 中支持的压缩格式。文本文件格式包括 TSV、CSV、JSON 和用于文本的自定义 SerDes。除非另有说明，否则单元格中的“是”或“否”同样适用于读取和写入操作。就本表而言，CREATE TABLE、CTAS 和 INSERT INTO 视为写入操作。有关在 Athena 中使用 ZSTD 压缩级别的更多信息，请参阅 在 Athena 中使用 ZSTD 压缩级别。

	Avro	Ion	ORC	Parquet	文本文件
BZIP2	支持	是	否	否	支持
DEFLATE	支持	否	否	否	不支持
GZIP	不支持	是	否	是	支持
LZ4	不支持	是	是	是	支持
LZO	不支持	写入 – 否 读取 – 是	不支持	支持	写入 – 否 读取 – 是
SNAPPY	支持	是	是	是	支持
ZLIB	不支持	否	是	否	不支持
ZSTD	支持	是	是	是	支持
NONE	支持	是	是	是	支持

Athena 引擎版本 2 中的 Hive 压缩支持

下表总结了对于 Apache Hive，Athena 引擎版本 2 中支持的压缩格式。文本文件格式包括 TSV、CSV、JSON 和用于文本的自定义 SerDes。除非另有说明，否则单元格中的“是”或“否”同样适用于读取和写入操作。就本表而言，CREATE TABLE、CTAS 和 INSERT INTO 视为写入操作。

	Avro	Ion	ORC	Parquet	文本文件
BZIP2	支持	是	否	否	支持
DEFLATE	支持	否	否	否	不支持
GZIP	不支持	是	否	是	支持
LZ4	不支持	否	支持	写入 – 是 读取 – 否	写入 – 否 读取 – 是

	Avro	Ion	ORC	Parquet	文本文件
LZO	不支持	写入 – 否 读取 – 是	不支持	支持	写入 – 否 读取 – 是
SNAPPY	支持	是	是	是	支持
ZLIB	不支持	否	是	否	不支持
ZSTD	不支持	是	是	是	支持
NONE	支持	是	是	是	支持

不同文件格式支持的 Iceberg 表压缩

Athena 中的 Apache Iceberg 压缩支持取决于引擎版本。

Athena 引擎版本 3 中的 Iceberg 压缩支持

下表总结了对于 Apache Iceberg 存储文件格式，Athena 引擎版本 3 中支持的压缩格式。除非另有说明，否则单元格中的“是”或“否”同样适用于读取和写入操作。就本表而言，CREATE TABLE、CTAS 和 INSERT INTO 视为写入操作。在 Athena 引擎版本 3 中，适用于 Iceberg 的默认存储格式为 Parquet。在 Athena 引擎版本 3 中，适用于 Iceberg 的默认压缩格式为 ZSTD。有关在 Athena 中使用 ZSTD 压缩级别的更多信息，请参阅 在 Athena 中使用 ZSTD 压缩级别。

	Avro	ORC	Parquet（默认）
BZIP2	不支持	否	不支持
GZIP	支持	否	支持
LZ4	不支持	是	不支持
SNAPPY	支持	是	支持
ZLIB	不支持	是	不支持
ZSTD	支持	支持	是（默认）

	Avro	ORC	Parquet（默认）
NONE	是（请指定 None 或 Deflate）	支持	是（请指定 None 或 Uncompressed ）

Athena 引擎版本 2 中的 Iceberg 压缩支持

下表总结了对于 Apache Iceberg，Athena 引擎版本 2 中支持的压缩格式。除非另有说明，否则单元格中的“是”或“否”同样适用于读取和写入操作。就本表而言，CREATE TABLE、CTAS 和 INSERT INTO 视为写入操作。在 Athena 引擎版本 2 中，适用于 Iceberg 的默认存储格式为 Parquet。在 Athena 引擎版本 2 中，适用于 Iceberg 的默认压缩格式为 GZIP。

	Avro （不支持）	ORC （不支持）	Parquet（默认）
BZIP2	不支持	否	不支持
GZIP	不支持	不支持	是（默认）
LZ4	不支持	否	不支持
SNAPPY	不支持	否	支持
ZLIB	不支持	否	不支持
ZSTD	不支持	否	支持
NONE	不支持	否	支持

在 Athena 中使用 ZSTD 压缩级别

Zstandard 实时数据压缩算法 是一种具有高压缩比的快速压缩算法。Zstandard（ZSTD）库是一种开源软件，使用 BSD 许可证。Athena 支持读取和写入 ZSTD 压缩的 ORC、Parquet 和文本文件数据。

您可以根据自己的需求，使用 ZSTD 压缩级别来调整压缩比和速度。ZSTD 库支持的压缩级别为 1 到 22。Athena 默认使用的 ZSTD 压缩级别为 3。

通过压缩级别可以在压缩速度和要达到的压缩量之间进行精细平衡控制。压缩级别越低，速度越快，但文件越大。例如，在速度最为重要时，可以使用级别 1；在大小最为重要时，可以使用级别 22。默认设置为级别 3，适用于许多应用场景。使用大于 19 的级别时应当谨慎，因为这将需要更多内存。ZSTD 库还提供负压缩级别，从而扩展压缩速度和压缩比的范围。有关更多信息，请参阅 Zstandard 压缩 RFC。

由于提供了多种压缩级别，因此可以充分进行微调优化。但在决定压缩级别时，请务必衡量您的数据量并相应权衡。我们建议使用默认级别 3 或介于 6 到 9 之间的级别，以便合理平衡压缩速度和压缩数据大小。20 及以上的级别应专用于大小最为重要且无需关注压缩速度的场景。

注意事项和限制

在 Athena 中使用 ZSTD 压缩级别时，请注意以下几点。

- 仅 Athena 引擎版本 3 支持 ZSTD `compression_level` 属性。
- `ALTER TABLE`、`CREATE TABLE`、`CREATE TABLE AS`（CTAS）和 `UNLOAD` 语句支持 ZSTD`compression_level` 属性。
- `compression_level` 属性是可选的。
- 仅 ZSTD 压缩支持 `compression_level` 属性。
- 可能的压缩级别为 1 到 22。
- 默认压缩级别为 3。

有关 Athena 中的 Apache Hive ZSTD 压缩支持的信息，请参阅 不同文件格式支持的 Hive 表压缩。有关 Athena 中的 Apache Iceberg ZSTD 压缩支持的信息，请参阅 不同文件格式支持的 Iceberg 表压缩。

指定 ZSTD 压缩级别

要为 `ALTER TABLE`、`CREATE TABLE`、`CREATE TABLE AS` 和 `UNLOAD` 语句指定 ZSTD 压缩级别，请使用 `compression_level` 属性。要指定 ZSTD 压缩本身，您必须使用该语句的语法所用的单个压缩属性。

ALTER TABLE SET TBLPROPERTIES

在 ALTER TABLE SET TBLPROPERTIES 语句的 `SET TBLPROPERTIES` 子句中，使用 `'write.compression' = ' ZSTD'` 或 `'parquet.compression' = 'ZSTD'` 指定 ZSTD 压缩。然后使用 `compression_level` 属性指定一个介于 1 到 22 之间的值（例如 `'compression_level' = 5`）。如果您未指定压缩级别属性，则压缩级别默认为 3。

示例

以下示例将修改表 existing_table 以使用 Parquet 文件格式以及 ZSTD 压缩和 ZSTD 压缩级别 4。请注意，压缩级别值必须以字符串而不是整数形式输入。

```
ALTER TABLE existing_table
SET TBLPROPERTIES ('parquet.compression' = 'ZSTD', 'compression_level' = 4)
```

CREATE TABLE

在 CREATE TABLE 语句的 TBLPROPERTIES 子句中，指定 'write.compression' = 'ZSTD' 或 'parquet.compression' = 'ZSTD'，然后使用 compression_level = *compression_level* 并指定一个介于 1 到 22 之间的值。如果未指定 compression_level 属性，则默认压缩级别为 3。

示例

以下示例将以 Parquet 文件格式创建一个表，并使用 ZSTD 压缩和 ZSTD 压缩级别 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)

在 CREATE TABLE AS 语句的 WITH 子句中，指定 write_compression = 'ZSTD' 或 parquet_compression = 'ZSTD'，然后使用 compression_level = *compression_level* 并指定一个介于 1 到 22 之间的值。如果未指定 compression_level 属性，则默认压缩级别为 3。

示例

以下 CTAS 示例将指定使用 Parquet 文件格式以及 ZSTD 压缩和 ZSTD 压缩级别 4。

```
CREATE TABLE new_table
WITH ( format = 'PARQUET', write_compression = 'ZSTD', compression_level = 4)
AS SELECT * FROM old_table
```

UNLOAD

在 UNLOAD 语句的 WITH 子句中，指定 `compression = 'ZSTD'`，然后使用 `compression_level = compression_level` 并指定一个介于 1 到 22 之间的值。如果未指定 `compression_level` 属性，则默认压缩级别为 3。

示例

以下示例将使用 Parquet 文件格式、ZSTD 压缩和 ZSTD 压缩级别 4 将查询结果卸载到指定位置。

```
UNLOAD (SELECT * FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/'
WITH (format = 'PARQUET', compression = 'ZSTD', compression_level = 4)
```

为 Athena 资源添加标签

标签包含您定义的一个键和一个值。您在标记 Athena 资源时，将为该资源分配自定义元数据。您可以使用标签按照不同方式（例如按用途、所有者或环境）对 AWS 资源进行分类。在 Athena 中，工作组、数据目录和容量预留等资源是可标记的资源。例如，您可以在账户中为工作组创建一组标签，以帮助您跟踪工作组所有者，或按用途识别工作组。如果您还在 Billing and Cost Management（账单和成本管理）控制台中启用标签作为成本分配标签，则与运行查询相关的费用将显示在“成本和使用情况报告”中，并带有该成本分配标签。我们建议您使用 AWS 标记最佳实践 来创建一组一致的标签，以满足您组织的要求。

您可以通过 Athena 控制台或 API 操作来使用标签。

主题

- 标签基本知识
- 标签限制
- 在控制台中的工作组上使用标签
- 使用标签操作
- 基于标签的 IAM 访问控制策略

标签基本知识

标签是为 Athena 资源分配的标记。每个标签都包含定义的一个密钥和一个可选值。

标签可让您按不同方式对 AWS 资源进行分类。例如，您可以为账户的工作组定义一组标签，以帮助您跟踪每个工作组所有者或用途。

您可以在创建新的 Athena 工作组或数据目录时添加标签，也可以在其中添加、编辑或删除标签。您可以在控制台中编辑标签。要使用 API 操作编辑标签，请删除旧标签并添加新标签。如果删除资源，资源的所有标签也会被删除。

Athena 不会自动向您的资源分配标签。您可以修改标签的密钥和值，还可以随时删除资源的标签。您可以将标签的值设为空的字符串，但是不能将其设为空值。请不要将重复的标签键添加到同一资源中。如果这样操作，Athena 会发布一条错误消息。如果通过 TagResource 操作用现有标签键标记资源，则新的标签值将覆盖旧值。

在 IAM 中，您可以控制 Amazon Web Services 账户中的哪些用户有权创建、编辑、删除或列出标签。有关更多信息，请参阅 基于标签的 IAM 访问控制策略。

有关 Amazon Athena 标记操作的完整列表，请参阅 Amazon Athena API 参考中的 API 操作名称。

您可以使用标签记账。有关更多信息，请参阅《AWS Billing and Cost Management 用户指南》中的使用账单标签。

有关更多信息，请参阅 标签限制。

标签限制

标签具有以下限制：

- 在 Athena 中，您可以标记工作组和数据目录。但不能标记查询。
- 每个资源的最大标签数是 50。要不超出限制，请检查并删除未使用的标签。
- 对于每个资源，每个标签键都必须是唯一的，每个标签键只能有一个值。请不要同时将重复的标签键添加到同一资源中。如果这样操作，Athena 会发布一条错误消息。如果在单独的 `TagResource` 操作中使用现有标签键标记工作组，则新的标签值将覆盖旧值。
- 标签键长度为 1-128 个 Unicode 字符（UTF-8 格式）。
- 标签值长度为 0-256 个 Unicode 字符（UTF-8 格式）。

  标记操作（如添加、编辑、删除或列出标签），需要您为工作组资源指定一个 ARN。
- Athena 允许使用 UTF-8 格式的字母、数字和空格，以及以下字符：+ - = . _ : / @。
- 标签键和值区分大小写。
- 标记键中的 `"aws:"` 前缀将保留以供 AWS 使用。不能编辑或删除带此前缀的标签键。具有此前缀的标签不计入每个资源的标签数限制。
- 您分配的标签仅可用于您的 Amazon Web Services 账户。

在控制台中的工作组上使用标签

通过 Athena 控制台，您可以查看您账户中每个工作组所使用的标签。您只能按工作组查看标签。您还可以通过 Athena 控制台，每次在一个工作组中应用、编辑或删除标签。

您可以使用创建的标签搜索工作组。

主题

显示单个工作组的标签

在 Athena 控制台中显示单个工作组的标签

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。

3. 在导航菜单上，选择 Workgroups（工作组），然后选择您希望使用的工作组。
4. 请执行以下操作之一：
   - 选择标签选项卡。如果标签列表太长，请使用搜索框。
   - 选择 Edit（编辑），然后向下滚动到 Tags（标签）部分。

对单个工作组添加和删除标签

您可以从 Workgroups (工作组) 选项卡直接管理单个工作组的标签。

> **Note**
>
> 如果您希望用户在控制台中创建工作组时添加标签，或者在使用 CreateWorkGroup 操作时传入标签，请确保向用户授予 IAM 权限以执行 TagResource 和 CreateWorkGroup 操作。

在创建新工作组时添加标签

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 在导航菜单上，选择 Workgroups（工作组）。
3. 选择 Create workgroup（创建工作组）并根据需要填入值。有关详细步骤，请参阅创建工作组。
4. 在 Tags（标签）部分，通过指定密钥和值，添加一个或多个标签。请勿同时将重复的标签键添加到同一个工作组。如果这样操作，Athena 会发布一条错误消息。有关更多信息，请参阅 标签限制。
5. 完成后，选择 Create workgroup（创建工作组）。

对现有工作组添加或编辑标签

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 在导航窗格中，选择 Workgroups（工作组）。
3. 选择您想要修改的工作组。
4. 请执行以下操作之一：
   - 选择标签选项卡，然后选择管理标签。
   - 选择 Edit（编辑），然后向下滚动到 Tags（标签）部分。
5. 为每个标签指定密钥和值。有关更多信息，请参阅 标签限制。
6. 选择保存。

删除单个工作组的标签

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 在导航窗格中，选择 Workgroups（工作组）。

3. 选择您想要修改的工作组。
4. 请执行以下操作之一：
   - 选择标签选项卡，然后选择管理标签。
   - 选择 Edit（编辑），然后向下滚动到 Tags（标签）部分。
5. 在标签列表中，为要删除的标签选择 Remove（删除），然后选择 Save（保存）。

使用标签操作

使用以下标签操作在资源中添加、删除或列出标签。

API	CLI	操作描述
TagResource	tag-resource	在具有指定 ARN 的资源上添加或覆盖一个或多个标签。
UntagResource	untag-resource	从具有指定 ARN 的资源中删除一个或多个标签。
ListTagsForResource	list-tags-for-resource	列出具有指定 ARN 的资源的一个或多个标签。

在创建资源时添加标签

要在创建工作组或数据目录时添加标签，请将 `tags` 参数与 `CreateWorkGroup` 或 `CreateDataCatalog` API 操作结合使用，或者将该参数与 AWS CLI `create-work-group` 或 `create-data-catalog` 命令结合使用。

使用 API 操作管理标签

此部分中的示例说明如何使用标签 API 操作来管理工作组和数据目录上的标签。这些示例采用的是 Java 编程语言。

Example TagResource

以下示例将两个标签添加到工作组 `workgroupA` 中：

```
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);
```

以下示例将两个标签添加到数据目录 datacatalogA 中：

```
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**
>
> 请不要将重复的标签键添加到同一资源中。如果这样操作，Athena 会发布一条错误消息。如果在单独的 TagResource 操作中使用现有标签键标记工作组，则新的标签值将覆盖旧值。

Example UntagResource

以下示例从工作组 workgroupA 中删除 tagKey2：

```
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);
```

以下示例从数据目录 datacatalogA 中删除 tagKey2：

```
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

以下示例列出了工作组 workgroupA 的标签：

```
ListTagsForResourceRequest request = new ListTagsForResourceRequest()
    .withResourceARN("arn:aws:athena:us-east-1:123456789012:workgroup/workgroupA");

ListTagsForResourceResult result = client.listTagsForResource(request);

List<Tag> resultTags = result.getTags();
```

以下示例列出了数据目录 datacatalogA 的标签：

```
ListTagsForResourceRequest request = new ListTagsForResourceRequest()
    .withResourceARN("arn:aws:athena:us-east-1:123456789012:datacatalog/datacatalogA");

ListTagsForResourceResult result = client.listTagsForResource(request);

List<Tag> resultTags = result.getTags();
```

使用AWS CLI管理标签

以下部分说明如何使用 AWS CLI 创建和管理数据目录上的标签。

向资源添加标签：Tag-resource

tag-resource 命令可向指定资源添加一个或多个标签。

语法

```
aws athena tag-resource --resource-arn
arn:aws:athena:region:account_id:datacatalog/catalog_name --tags
Key=string,Value=string Key=string,Value=string
```

--resource-arn 参数指定要将标签添加到的资源。--tags 参数指定要作为标签添加到资源的用空格分隔的键/值对列表。

Example

以下示例将标签添加到 mydatacatalog 数据目录中。

```
aws athena tag-resource --resource-arn arn:aws:athena:us-
east-1:111122223333:datacatalog/mydatacatalog --tags Key=Color,Value=Orange
 Key=Time,Value=Now
```

要显示结果，请使用 list-tags-for-resource 命令。

有关在使用 create-data-catalog 命令时添加标签的信息，请参阅 注册目录：Create-data-catalog。

列出资源的标签：List-tags-for-resource

list-tags-for-resource 命令将列出指定资源的标签。

语法

aws athena list-tags-for-resource --resource-arn arn:aws:athena:*region*:*account_id*:datacatalog/*catalog_name*

--resource-arn 参数指定将列出其标签的资源。

以下示例列出了 mydatacatalog 数据目录的标签。

```
aws athena list-tags-for-resource --resource-arn arn:aws:athena:us-
east-1:111122223333:datacatalog/mydatacatalog
```

以下示例结果采用 JSON 格式。

```
{
    "Tags": [
        {
            "Key": "Time",
            "Value": "Now"
        },
        {
            "Key": "Color",
            "Value": "Orange"
```

```
        }
    ]
}
```

从资源中删除标签：Untag-resource

untag-resource 命令将从指定资源中删除指定的标签键及其关联的值。

语法

```
aws athena untag-resource --resource-arn
arn:aws:athena:region:account_id:datacatalog/catalog_name --tag-keys
key_name [key_name ...]
```

--resource-arn 参数指定从中删除标签的资源。--tag-keys 参数采用一组用空格分隔的键名称。对于指定的每个键名称，untag-resource 命令将同时删除键及其值。

以下示例从 mydatacatalog 目录资源中删除 Color 和 Time 键及其值。

```
aws athena untag-resource --resource-arn arn:aws:athena:us-
east-1:111122223333:datacatalog/mydatacatalog --tag-keys Color Time
```

基于标签的 IAM 访问控制策略

拥有标签后，您可以编写包含 Condition 块的 IAM policy，以便基于其标签来控制对资源的访问。

工作组的标签策略示例

Example 1. 基本标记策略

以下 IAM policy 允许您运行查询并与名为 workgroupA 的工作组的标签进行交互：

```
{
    "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：基于标签键和标签值对拒绝对工作组的操作的策略块

与资源（如工作组）关联的标签称作资源标签。利用资源标签，您可以编写如下所示的策略块，以拒绝对使用键-值对（如 stack、production）标记的任何工作组执行列出的操作。

```
{
    "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. 限制对指定标签执行标签更改操作请求的策略块

作为参数传入将更改标签的操作（例如，带标签的 TagResource、UntagResource 或 CreateWorkGroup）的标签称作请求标签。仅在传递的某个标签的键为 costcenter 且值为 1、2 或 3 时，以下示例策略块才允许 CreateWorkGroup 操作。

> **Note**
>
> 如果要允许 IAM 角色将标签作为 `CreateWorkGroup` 操作的一部分传入，请确保向角色授予执行 `TagResource` 和 `CreateWorkGroup` 操作的权限。

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

数据目录的标签策略示例

Example 1. 基本标记策略

以下 IAM policy 允许您与名为 `datacatalogA` 的数据目录的标签进行交互：

```
{
    "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：基于标签键和标签值对拒绝对数据目录的操作的策略块

您可以使用资源标签编写策略块，以拒绝对使用特定标签键/值对标记的数据目录执行特定操作。以下示例策略拒绝对具有标签键/值对 `stack production` 的数据目录执行操作。

```
{
    "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. 限制对指定标签执行标签更改操作请求的策略块

作为参数传入将更改标签的操作（例如，带标签的 TagResource、UntagResource 或 CreateDataCatalog）的标签称作请求标签。仅在传递的某个标签的键为 costcenter 且值为 1、2 或 3 时，以下示例策略块才允许 CreateDataCatalog 操作。

> **Note**
>
> 如果要允许 IAM 角色将标签作为 CreateDataCatalog 操作的一部分传入，请确保向角色授予执行 TagResource 和 CreateDataCatalog 操作的权限。

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

服务限额

> **Note**
>
> Service Quotas 控制台提供有关 Amazon Athena 配额的信息。对于可调整的限额，您可以使用服务限额控制台请求增加限额。有关与 AWS Glue 相关的架构限制，请参阅 AWS Glue 端

点和限额页面。有关 AWS 服务限额的一般性信息，请参阅 AWS 一般参考 中的 AWS 服务限额。

查询

您的账户对于 Amazon Athena 具有以下与查询相关的配额。有关详细信息，请参阅 AWS 一般参考 的 Amazon Athena 端点和限额页面。

- 活跃 DDL 查询 – 活跃 DDL 查询的数量。DDL 查询包括 `CREATE TABLE` 和 `ALTER TABLE ADD PARTITION` 查询。
- DDL 查询超时 – DDL 查询在取消之前可以运行的最长时间（以分钟为单位）。
- 活跃 DML 查询 – 活跃 DML 查询的数量。DML 查询包括 `SELECT`、`CREATE TABLE AS` (CTAS) 和 `INSERT INTO` 查询。具体配额因 AWS 区域而异。
- DML 查询超时 – DML 查询在取消之前可以运行的最长时间（以分钟为单位）。您可以申请延长此超时时间，但最长不得超过 240 分钟。

要请求增加限额，您可以使用 Athena 服务限额控制台。

Athena 会根据总体服务负载和传入请求的数量，通过分配资源来处理查询。您的查询可能会在运行之前临时排队。异步进程从队列中获取查询，并在资源可用时立即在物理资源上运行它们，只要您的账户配置允许。

DML 或 DDL 查询配额包括正在运行的查询和排队的查询。例如，如果您的 DML 配额为 25，并且正在运行和排队的查询总数为 26，则查询 26 将导致 TooManyRequestsException 错误。

> **Note**
>
> 如果您想直接控制在 Athena 中运行的查询的并发性，则可以使用容量预留。有关更多信息，请参阅 管理查询处理容量。

查询字符串长度

查询字符串允许的最大长度为 262144 字节，字符串采用 UTF-8 编码。这不是一个可调节的配额。但是，您可以通过将长查询拆分为多个较小的查询来解决此限制。有关更多信息，请参阅 AWS 知识中心中的如何增加 Athena 中的最大查询字符串长度？。

工作组

使用 Athena 工作组时，请记住以下几点：

- Athena 服务配额在账户中的所有工作组之间共享。
- 可为您账户中的每个区域创建的工作组的最大数量为 1000。
- 工作组中的最大预处理语句数为 1000。
- 每个工作组的最大标签数是 50。有关更多信息，请参阅 标签限制。

数据库、表和分区

- 如果您正在将 AWS Glue Data Catalog 与 Athena 搭配使用，请参阅 AWS Glue 端点和配额，获取有关表、数据库和分区的 Service Quotas 的信息，例如每个账户的最大数据库或表数。
  - 尽管 Athena 支持查询具有 1000 万个分区的 AWS Glue 表，但 Athena 的单次扫描读取量最多为 100 万个分区。
- 如果您未使用 AWS Glue Data Catalog，则每个表的分区数为 20000。您可以请求提高限额。

Amazon S3 存储桶

使用 Amazon S3 存储桶时，请记住以下几点：

- Amazon S3 的默认服务配额为每个账户 100 个存储桶。
- Athena 需要一个单独的存储桶来记录结果。
- 您可以请求每个 AWS 账户最多提高 1000 个 Amazon S3 存储桶的配额。

每个账户 API 调用配额

Athena API 对每个账户的 API 调用数量（不是每个查询）具有以下默认配额：

API 名称	每秒默认调用次数	容量爆增
`BatchGetNamedQuery` ，`ListNamedQueries` ，`ListQueryExecutions`	5	最多 10 个
`CreateNamedQuery` ，`DeleteNamedQuery` ，`GetNamedQuery`	5	最多 20 个

API 名称	每秒默认调用次数	容量爆增
BatchGetQueryExecution	20	最多 40 个
StartQueryExecution ，StopQueryExecution	20	最多 80 个
GetQueryExecution ，GetQueryResults	100	最多 200 个

例如，对于 StartQueryExecution，每秒最多可以调用 20 次。此外，如果此 API 在 4 秒内未被调用，则您账户的容量暴增将累积至最多 80 次调用。在这种情况下，应用程序在突增模式下最多可以对此 API 进行 80 次调用。

如果您使用这些 API 中的任何一个，并且超出了每秒调用次数的默认配额或您账户中的容量暴增，则 Athena API 会发出类似于以下内容的错误：“ClientError: An error occurred (ThrottlingException) when calling the <API_name> operation: Rate exceeded. (ClientError：调用操作时发生错误 (ThrottlingException)：超出速率。)” 减少每秒调用次数或此账户的 API 容量暴增。

无法在 Athena Service Quotas 控制台中更改每个账户 API 调用的 Athena 限额。要申请增加 Athena API 调用的限额，请导航到 AWS Support 服务限制提高页面，然后填写并提交表单。

Athena 引擎版本控制

Athena 偶尔会发布新的引擎版本，以提供更好的性能、功能并修复代码。当有新的引擎版本可用时，Athena 会通过 Athena 控制台和您的 AWS Health Dashboard 通知您。您的 AWS Health Dashboard 会通知您可能影响 AWS 服务或账户的事件。有关 AWS Health Dashboard 的更多信息，请参阅 AWS Health Dashboard 入门。

引擎版本控制是按工作组配置的。您可以使用工作组来控制查询使用哪个查询引擎，以及是否让 Athena 自动升级工作组。正在使用的查询引擎会显示在查询编辑器中，位于工作组的详细信息页面上，并可通过 Athena API 使用。

- 工作组默认配置为自动升级。工作组设置为自动升级时，Athena 会升级工作组，除非发现工作组不兼容。
- 如果将工作组配置为使用给定版本，Athena 就不会更改该工作组的版本。

在这两种情况下，当某个版本不再可用时，Athena 便会对工作组进行升级。如果某个引擎版本不再可用，Athena 会通过 AWS Health Dashboard 通知您。您的 AWS Health Dashboard 会通知您可能影响 AWS 服务或账户的事件。有关 AWS Health Dashboard 的更多信息，请参阅 AWS Health Dashboard 入门。

当您开始使用新的引擎版本时，由于不兼容，一小部分查询可能会中断。如果发布新的 Athena 版本，就会宣布重大更改。您应该在升级之前使用工作组来测试查询，方法是创建使用新引擎的测试工作组或测试升级现有工作组。有关更多信息，请参阅 在引擎版本升级之前测试查询。

主题

更改 Athena 引擎版本

Athena 偶尔会发布新的引擎版本，以提供更好的性能、功能并修复代码。当新的引擎版本可用时，Athena 会在控制台中通知您。您可以选择让 Athena 决定何时升级，也可以手动指定每个工作组的 Athena 引擎版本。

主题

查找工作组的查询引擎版本

您可以使用 Workgroups（工作组）页面查找任何工作组的当前引擎版本。

要查找任何工作组的当前引擎版本

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。

3. 在 Athena 控制台导航窗格中，选择 Workgroups（工作组）。
4. 在 Workgroups（工作组）页面上，找到所需的工作组。该工作组的 Query engine version（查询引擎版本）列显示查询引擎版本。

在 Athena 控制台中更改引擎版本

当新引擎版本可用时，您便可以选择让 Athena 决定何时升级工作组，或手动指定工作组使用的 Athena 引擎版本。如果当前只有一个版本可用，则无法手动指定其他版本。

> **Note**
>
> 要更改工作组的引擎版本，您必须有权限对工作组执行 `athena:ListEngineVersions` 操作。有关 IAM policy 示例，请参阅 工作组策略示例。

要让 Athena 决定何时升级工作组

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。
3. 在控制台导航窗格中，选择 Workgroups（工作组）。
4. 在工作组列表中，选择要配置的工作组链接。
5. 选择编辑。

6. 在 Query engine version（查询引擎版本）部分中，对于 Update query engine（更新查询引擎），选择 Automatic（自动），让 Athena 选择何时升级您的工作组。这是默认设置。
7. 选择 Save changes（保存更改）。

   在工作组列表中，工作组的 Query engine update status（查询引擎更新状态）显示 Automatic（自动）。

要手动选择引擎版本

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。
3. 在控制台导航窗格中，选择 Workgroups（工作组）。
4. 在工作组列表中，选择要配置的工作组链接。
5. 选择编辑。
6. 在 Query engine version（查询引擎版本）部分中，对于 Update query engine（更新查询引擎），选择 Manual（手动）以手动选择引擎版本。
7. 使用 Query engine version（查询引擎版本）选项，选择希望工作组使用的引擎版本。如果没有其他引擎版本可用，则无法指定其他引擎版本。
8. 选择 Save changes（保存更改）。

   在工作组列表中，工作组的 Query engine update status（查询引擎更新状态）显示 Manual（手动）。

通过 AWS CLI 更改引擎版本

要通过 AWS CLI 更改引擎版本，请使用以下示例中的语法。

```
aws athena update-work-group --work-group workgroup-name --configuration-updates
 EngineVersion={SelectedEngineVersion='Athena engine version 3'}
```

创建工作组时指定引擎版本

创建工作组时，可以指定工作组使用的引擎版本，或者让 Athena 决定何时升级工作组。如果有新的引擎版本可用，最佳实践是在升级其他工作组之前创建一个工作组以测试新引擎。要指定工作组的引擎版本，您必须具有工作组的 `athena:ListEngineVersions` 权限。有关 IAM policy 示例，请参阅 工作组策略示例。

要在创建工作组时指定引擎版本

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。
3. 在控制台导航窗格中，选择 Workgroups（工作组）。
4. 在 Workgroups（工作组）页面中，选择 Create workgroup（创建工作组）。
5. 在 Create workgroup（创建工作组）页面上的 Query engine version（查询引擎版本）部分中，请执行以下操作之一：
   - 选择 Automatic（自动），让 Athena 选择何时升级您的工作组。这是默认设置。
   - 选择 Manual（手动）以手动选择其他引擎版本（如果可用）。
6. 根据需要输入其他字段的信息。有关其他字段的信息，请参阅 创建工作组。
7. 选择 Create workgroup (创建工作组)。

在引擎版本升级之前测试查询

当工作组升级到新的引擎版本时，您的某些查询可能会因不兼容而中断。为了确保引擎版本升级顺利进行，您可以提前测试查询。

要在引擎版本升级之前测试查询

1. 验证您使用的工作组的引擎版本。您正在使用的引擎版本会显示在 Workgroups（工作组）页面上工作组的 Query engine version（查询引擎版本）列中。有关更多信息，请参阅 查找工作组的查询引擎版本。
2. 创建使用新引擎版本的测试工作组。有关更多信息，请参阅 创建工作组时指定引擎版本。
3. 使用新工作组运行要测试的查询。
4. 如果查询未能正常运行，请使用 Athena 引擎版本参考 来检查可能会影响查询的中断更改。某些更改可能需要您更新查询的语法。
5. 如果您的查询仍然失败，请联系 AWS Support 以获取帮助。在 AWS Management Console 中，选择 Support（支持）、Support Center（支持中心），或使用 Amazon Athena 标签在 AWS re:Post 上提问。

为运行失败的查询排查问题

如果查询在引擎版本升级后运行失败，请使用 Athena 引擎版本参考 来检查中断更改，包括可能影响查询中语法的更改。

如果您的查询仍然失败，请联系 AWS Support 以获取帮助。在 AWS Management Console 中，选择 Support（支持）、Support Center（支持中心），或使用 Amazon Athena 标签在 AWS re:Post 上提问。

Athena 引擎版本参考

这一部分列出了 Athena 查询引擎发生的更改。

主题

Athena 引擎版本 3

对于引擎版本 3，Athena 还引入了一种用于开源软件管理的持续集成方法，以提高与 Trino 和 Presto 项目并发的能力，从而让您能够更快地获得在 Athena 引擎中集成和优化的社区改进。

本次发布的 Athena 引擎版本 3 支持 Athena 引擎版本 2 的所有功能。本文档重点介绍了 Athena 引擎版本 2 和 Athena 引擎版本 3 之间的主要区别。有关更多信息，请参阅 AWS 大数据博客文章升级到 Athena 引擎版本 3 以提高查询性能并访问更多分析功能。

开始使用

首先，请创建一个使用 Athena 引擎版本 3 的新 Athena 工作组，或将现有工作组配置为使用版本 3。所有 Athena 工作组都可以在不中断查询提交能力的情况下，从引擎版本 2 升级到引擎版本 3。

有关更多信息，请参阅 更改 Athena 引擎版本。

改进功能和新功能

列出的功能和更新包括 Athena 本身的改进以及开源 Trino 项目集成功能的改进。有关 SQL 查询运算符和函数的详尽列表，请参阅 Trino 文档。

增加的功能

支持 Apache Spark 分桶算法

Athena 可以读取 Spark 哈希算法生成的桶。要指定数据最初是由 Spark 哈希算法编写的，请在 CREATE TABLE 语句的 TBLPROPERTIES 子句中输入 ('bucketing_format'='spark')。如果未指定此属性，则将使用 Hive 哈希算法。

```
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')
```

已添加的函数

这一部分介绍的函数是 Athena 引擎版本 3 新引入的函数。

聚合函数

listagg(x, separator) – 返回由分隔符字符串分隔的级联输入值。

```
SELECT listagg(value, ',') WITHIN GROUP (ORDER BY value) csv_value
FROM (VALUES 'a', 'c', 'b') t(value);
```

数组函数

contains_sequence(x, seq) – 如果数组 x 将所有数组序列作为顺序子集包含在内（所有值的连续顺序相同），则返回 true。

```
SELECT contains_sequence(ARRAY [1,2,3,4,5,6], ARRAY[1,2]);
```

二进制函数

murmur3(binary) – 计算二进制的 128 位 MurmurHash3 哈希值。

```
SELECT murmur3(from_base64('aaaaaa'));
```

转换函数

format_number(number) – 返回使用单位符号的格式化字符串。

```
SELECT format_number(123456); -- '123K'
```

```
SELECT format_number(1000000); -- '1M'
```

日期和时间函数

timezone_hour(timestamp) – 返回时区与时间戳偏移的小时数。

```
SELECT EXTRACT(TIMEZONE_HOUR FROM TIMESTAMP '2020-05-10 12:34:56 +08:35');
```

timezone_minute(timestamp) – 返回时区与时间戳偏移的分钟数。

```
SELECT EXTRACT(TIMEZONE_MINUTE FROM TIMESTAMP '2020-05-10 12:34:56 +08:35');
```

地理空间函数

to_encoded_polyline(Geometry) – 将线串或多点编码为折线。

```
SELECT to_encoded_polyline(ST_GeometryFromText(
   'LINESTRING (-120.2 38.5, -120.95 40.7, -126.453 43.252)'));
```

from_encoded_polyline(varchar) – 将折线解码为线串。

```
SELECT ST_AsText(from_encoded_polyline('_p~iF~ps|U_ulLnnqC_mqNvxq`@'));
```

to_geojson_geometry(SphericalGeography) – 以 GeoJSON 格式返回指定的球面地理位置。

```
SELECT to_geojson_geometry(to_spherical_geography(ST_GeometryFromText(
   'LINESTRING (0 0, 1 2, 3 4)')));
```

from_geojson_geometry(varchar) – 返回 GeoJSON 表示中的球面地理位置类型对象，去除非几何键/值。不支持 `Feature` 和 `FeatureCollection`。

```
SELECT
 from_geojson_geometry(to_geojson_geometry(to_spherical_geography(ST_GeometryFromText(
   'LINESTRING (0 0, 1 2, 3 4)'))));
```

geometry_nearest_points(Geometry, Geometry) – 返回每个几何体上彼此最近的点。如果任何一个几何体为空，则返回 NULL。否则，返回由几何体上任意两点间距离最小的两个 `Point` 对象组成的行。第一个点来自第一个 Geometry 参数，第二个点来自第二个 Geometry 参数。如果有多对点具有相同的最小距离，则任意选择一对。

```
SELECT geometry_nearest_points(ST_GeometryFromText(
   'LINESTRING (50 100, 50 200)'), ST_GeometryFromText(
   'LINESTRING (10 10, 20 20)'));
```

设置摘要函数

make_set_digest(x) – 将 x 的所有输入值合成一个 setdigest。

```
SELECT make_set_digest(value) FROM (VALUES 1, 2, 3) T(value);
```

字符串函数

soundex(char) – 返回一个包含字符语音表示的字符串。

```
SELECT name
FROM nation
WHERE SOUNDEX(name) = SOUNDEX('CHYNA'); -- CHINA
```

concat_ws(string0, string1, ..., stringN) – 返回以 `string0` 为分隔符的 `string1, string2, ..., stringN` 级联。如果 `string0` 为 null，则返回值为 null。参数中跳过分隔符后提供的任何空值。

```
SELECT concat_ws(',', 'def', 'pqr', 'mno');
```

窗口函数

GROUPS – 增加对基于组的窗口框架的支持。

```
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);
```

性能改进

Athena 引擎版本 3 中的性能改进包括以下方面。

- 更快的 AWS Glue 表元数据检索 – 涉及多个表的查询可缩短查询计划时间。
- RIGHT JOIN 动态筛选 – 现在可为具有相等连接条件的右连接启用动态筛选，如下例所示。

  ```
  SELECT *
  FROM lineitem RIGHT JOIN tpch.tiny.supplier
  ON lineitem.suppkey = supplier.suppkey
  WHERE supplier.name = 'abc';
  ```

- 大型预编译语句 – 将默认 HTTP 请求/响应标头的大小增加到 2 MB，以方便使用大型预编译语句。
- approx_percentile() - `approx_percentile` 函数现在使用 `tdigest` 而不是 `qdigest` 从分布中检索近似分位数值。这样可以提高性能并降低内存使用量。请注意，进行此更改后，该函数返回的结果与 Athena 引擎版本 2 中的结果不同。有关更多信息，请参阅 approx_percentile 函数将返回不同的结果。

可靠性增强

Athena 引擎版本 3 改进了整体引擎内存使用和跟踪性能。降低了大型查询因节点崩溃而失败的可能性。

查询语法增强

INTERSECT ALL – 增加了对 `INTERSECT ALL` 的支持。

```
SELECT * FROM (VALUES 1, 2, 3, 4) INTERSECT ALL SELECT * FROM (VALUES 3, 4);
```

EXCEPT ALL – 增加了对 EXCEPT ALL 的支持。

```
SELECT * FROM (VALUES 1, 2, 3, 4) EXCEPT ALL SELECT * FROM (VALUES 3, 4);
```

RANGE PRECEDING – 在窗口函数中增加了对 RANGE PRECEDING 的支持。

```
SELECT sum(x) over (order by x range 1 preceding)
FROM (values (1), (1), (2), (2)) t(x);
```

MATCH_RECOGNIZE – 增加了对行模式匹配的支持，如下例所示。

```
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;
```

数据格式和数据类型增强

Athena 引擎版本 3 提供了以下数据格式和数据类型增强。

- LZ4 和 ZSTD – 增加了对读取 LZ4 和 ZSTD 压缩格式的 Parquet 数据的支持。增加了对写入 ZSTD 压缩格式的 ORC 数据的支持。
- 基于符号链接的表 – 增加了对在 Avro 文件上创建基于符号链接的表的支持。下面是一个示例。

  ```
  CREATE TABLE test_avro_symlink
  ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
  ...
  INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
  ```

- SphericalGeography – SphericalGeography 类型为地理坐标（有时也称大地测量坐标、纬度/经度或经度/纬度）上表示的空间要素提供原生支持。地理坐标是以角度单位（度）表示的球面坐标。

  to_spherical_geography 函数将从几何（平面）坐标返回地理（球面）坐标，如下例所示。

```
SELECT to_spherical_geography(ST_GeometryFromText(
   'LINESTRING (-40.2 28.9, -40.2 31.9, -37.2 31.9)'));
```

重大更改

从 Athena 引擎版本 2 迁移到 Athena 引擎版本 3 时，某些更改可能会影响表 Schema、语法或数据类型的用法。这一部分列出了相关的错误消息并提供了解决方法建议。

查询语法更改

IGNORE NULLS 不能与非值窗口函数一起使用

错误消息：无法为 `bool_or` 函数指定空值处理子句。

原因：现在，`IGNORE NULLS` 只能与值函数 `first_value`、`last_value`、`nth_value`、`lead` 和 `lag` 一起使用。此更改是为了符合 ANSI SQL 规范。

建议的解决方案：从查询字符串的非值窗口函数中移除 `IGNORE NULLS`。

CONCAT 函数必须有两个或以上的参数

错误消息：INVALID_FUNCTION_ARGUMENT: There must be two or more concatenation arguments（INVALID_FUNCTION_ARGUMENT：必须有两个或更多串联参数）

原因：以前，`CONCAT` 字符串函数接受单个参数。在 Athena 引擎版本 3 中，`CONCAT` 函数至少需要两个参数。

建议的解决方法：将 `CONCAT(str)` 的出现次数更改为 `CONCAT(str, '')`。

在 Athena 引擎版本 3 中，函数的参数不能超过 127 个。有关更多信息，请参阅 函数调用的参数太多。

approx_percentile 函数将返回不同的结果

`approx_percentile` 函数在 Athena 引擎版本 3 中返回的结果与在 Athena 引擎版本 2 中返回的结果不同。

错误消息：无。

原因：`approx_percentile` 函数会受到版本变更的影响。

> **Important**
>
> 由于 approx_percentile 函数的输出是近似值，并且近似值可能会因版本而异，因此在关键应用中不应依赖 approx_percentile 函数。

建议的解决方案：要近似 approx_percentile 的 Athena 引擎版本 2 行为，可以在 Athena 引擎版本 3 中使用一组不同的函数。例如，假设您在 Athena 引擎版本 2 中具有以下查询：

```
SELECT approx_percentile(somecol, 2E-1)
```

要在 Athena 引擎版本 3 中获得近似相同的输出，可以尝试使用 qdigest_agg 和 value_at_quantile 函数，如以下示例所示。请注意，即使使用这种解决方法，也不能保证会出现相同的行为。

```
SELECT value_at_quantile(qdigest_agg(somecol, 1), 2E-1)
```

地理空间函数不支持 varbinary 输入

错误消息：FUNCTION_NOT_FOUND for st_XXX（未找到 st_XXX 的函数）

原因：一些地理空间函数不再支持遗留的 VARBINARY 输入类型或与文本相关的函数签名。

建议的解决方法：使用地理空间函数将输入类型转换为支持的类型。错误消息中指示了支持的输入类型。

在 GROUP BY 子句中，嵌套列必须包含在双引号中

错误消息："*column_name*"."*nested_column*" 必须是聚合表达式或者出现在 GROUP BY 子句中

原因：Athena 引擎版本 3 要求 GROUP BY 子句中的嵌套列名包含在双引号中。例如，以下查询会产生错误，因为 GROUP BY 子句中 user.name 没有包含在双引号中。

```
SELECT "user"."name" FROM dataset
GROUP BY user.name
```

建议的解决方案：在 GROUP BY 子句中将嵌套的列名称包含在双引号中，如以下示例所示。

```
SELECT "user"."name" FROM dataset
GROUP BY "user"."name"
```

在 Iceberg 表上使用 OPTIMIZE 时出现意外 FilterNode 错误

错误消息：计划中发现意外 FilterNode；连接器可能无法处理提供的 WHERE 表达式。

原因：在 Iceberg 表上运行的 `OPTIMIZE` 语句使用的 `WHERE` 子句在其筛选表达式中包含非分区列。

建议的解决方案：`OPTIMIZE` 语句仅支持按分区筛选。对分区表运行 `OPTIMIZE` 时，请在 `WHERE` 子句中仅包含分区列。如果对非分区表运行 `OPTIMIZE`，请不要指定 `WHERE` 子句。

Log() 函数的参数顺序

在 Athena 引擎版本 2 中，`log()` 函数的参数顺序为 `log(value, base)`。在 Athena 引擎版本 3 中，为符合 SQL 标准，参数顺序已更改 `log(base, value)`。

Minute() 函数不支持年和月间隔

错误消息：Unexpected parameters (interval year to month) for function minute. [函数分钟的意外参数（年到月的间隔）。] Expected: minute(timestamp with time zone) , minute(time with time zone) , minute(timestamp) , minute(time) , minute(interval day to second). [minute 函数参数不符合预期（年和月间隔）。预期参数：minute(时间戳及时区), minute(时间及时区), minute(时间戳), minute(时间), minute(天到秒间隔)。]

原因：在 Athena 引擎版本 3 中，根据 ANSI SQL 规范，`EXTRACT` 的类型检查变得更加精确。

建议的解决方法：更新查询，确保类型与建议的函数签名相匹配。

ORDER BY 表达式必须显示在 SELECT 列表中。

错误消息：For SELECT DISTINCT, ORDER BY expressions must appear in SELECT list（对于 SELECT DISTINCT，ORDER BY 表达式必须在 SELECT 列表中显示）

原因：`SELECT` 子句中使用了不正确的表别名。

建议的解决方法：仔细检查 `ORDER BY` 表达式中的所有列，确认其在 `SELECT DISTINCT` 子句中的引用是否正确。

比较子查询返回的多列时查询失败

错误消息示例：值表达式和子查询结果的类型必须相同：row(varchar, varchar) 与 row(row(varchar, varchar))

原因：由于 Athena 引擎版本 3 中的语法更新，当查询尝试比较子查询返回的多个值时，如果子查询 `SELECT` 语句将其列列表包含在括号内，就会出现此错误，如以下示例所示。

```
SELECT *
FROM table1
WHERE (t1_col1, t1_col2)
IN (SELECT (t2_col1, t2_col2) FROM table2)
```

解决方案：在 Athena 引擎版本 3 中，移除子查询 SELECT 语句中列列表两边的括号，如以下更新后的示例查询所示。

```
SELECT *
FROM table1
WHERE (t1_col1, t1_col2)
IN (SELECT t2_col1, t2_col2 FROM table2)
```

SKIP 是 DML 查询的保留字

SKIP 现在是 DML 查询的保留字，与 SELECT 一样。要在 DML 查询中将 SKIP 用作标识符，请用双引号将其括起来。

有关 Athena 中保留字的更多信息，请参阅 保留关键字。

时间旅行已弃用 SYSTEM_TIME 和 SYSTEM_VERSION 子句

错误消息：mismatched input 'SYSTEM_TIME'.（不匹配的输入“SYSTEM_TIME”。）Expecting: 'TIMESTAMP', 'VERSION'（预期输入：“TIMESTAMP”、“VERSION”）

原因：在 Athena 引擎版本 2 中，Iceberg 表使用 FOR SYSTEM_TIME AS OF 和 FOR SYSTEM_VERSION AS OF 子句来表示时间戳和版本时间旅行。Athena 引擎版本 3 使用 FOR TIMESTAMP AS OF 和 FOR VERSION AS OF 子句。

建议的解决方法：更新 SQL 查询以使用 TIMESTAMP AS OF 和 VERSION AS OF 子句进行时间旅行操作，如下例所示。

按时间戳划分的时间旅行：

```
SELECT * FROM TABLE FOR TIMESTAMP AS OF (current_timestamp - interval '1' day)
```

按版本划分的时间旅行：

```
SELECT * FROM TABLE FOR VERSION AS OF 949530903748831860
```

数组构造函数的参数过多

错误消息：TOO_MANY_ARGUMENTS：数组构造函数的参数过多。

原因：数组构造函数中的最大元素数现在设置为 254。

建议的解决方案：将元素分解为多个数组，每个数组的元素不超过 254 个，然后使用 CONCAT 函数连接这些数组，如以下示例所示。

```
CONCAT(
ARRAY[x1,x2,x3...x254],
ARRAY[y1,y2,y3...y254],
...
)
```

不允许使用零长度分隔标识符

错误消息：Zero–length delimited identifier not allowed.（不允许使用零长度分隔标识符。）

原因：查询使用空字符串作为列别名。

建议的解决方法：更新查询，为列使用非空别名。

数据处理更改

存储桶验证

错误消息：HIVE_INVALID_BUCKET_FILES：Hive 表已损坏。

原因：表可能已损坏。为了确保分桶表的查询正确性，Athena 引擎版本 3 支持对分桶表进行额外验证，以确保查询正确并避免在运行时出现意外故障。

建议的解决方案：使用 Athena 引擎版本 3 重新创建表。

将结构体转换为 JSON 现在会返回字段名称

当您在 Athena 引擎版本 3 的 SELECT 查询中将 `struct` 转换为 JSON 时，这种转换现在会返回字段名称和值（例如 `"useragent":null`），而不仅仅是值（例如 `null`）。

Iceberg 表列级安全执行更改

错误消息：Access Denied: Cannot select from columns（访问被拒绝：无法从列中选择）

原因：Iceberg 表是在 Athena 之外创建的，使用的是早于 0.13.0 的 Apache Iceberg SDK 版本。由于早期的 SDK 版本不填充 AWS Glue 中的列，因此 Lake Formation 无法确定已获访问授权的列。

建议的解决方法：使用 Athena ALTER TABLE SET PROPERTIES 语句执行更新，或者使用最新的 Iceberg SDK 修复该表并更新 AWS Glue 中的列信息。

List 数据类型中的空值现在会传播到 UDF

错误消息：Null Pointer Exception（空指针异常）

原因：如果您使用 UDF 连接器并实施了用户定义的 Lambda 函数，则可能出现此问题。

Athena 引擎版本 2 会过滤掉传递给用户定义函数的 List 数据类型中的空值。在 Athena 引擎版本 3 中，空值现在将被保留并传递到 UDF。如果 UDF 尝试在不检查的情况下取消引用空元素，则可能会导致空指针异常。

例如，假设您在 DynamoDB 等原始数据来源中有数据 `[null, 1, null, 2, 3, 4]`，则将传递给用户定义的 Lambda 函数的实际数据将会如下：

Athena 引擎版本 2：`[1,2,3,4]`

Athena 引擎版本 3：`[null, 1, null, 2, 3, 4]`

建议的解决方法：确保用户定义的 Lambda 函数会处理列表数据类型中的空元素。

字符数组中的子字符串不再包含填充空格

错误消息：No error is thrown, but the string returned no longer contains padded spaces.（不会触发错误，但返回的字符串不再包含填充空格。）例如，`substr(char[20],1,100)` 现在返回长度为 20 而不是 100 的字符串。

建议的解决方法：无需执行任何操作。

不支持的十进制列类型强制

错误消息：HIVE_CURSOR_ERROR：无法读取 Parquet 文件：s3://DOC-EXAMPLE-BUCKET/*path*/*file_name*.parquet 或 Parquet 列 ([*column_name*] 的列类型 (varchar) 不受支持

原因：Athena 引擎版本 2 在尝试将数据类型从 `varchar` 强制转换为十进制时偶尔会成功（但经常失败）。由于 Athena 引擎版本 3 具有在尝试读取值之前检查类型是否兼容的类型验证，因此这种尝试强制操作现在总是会失败。

建议的解决方案：对于 Athena 引擎版本 2 和 Athena 引擎版本 3，在 AWS Glue 中将架构修改为在 Parquet 文件的十进制列中使用数值数据类型，而不是 `varchar`。重新爬取数据，确保新列数据类型为十进制类型，或在 Athena 中手动重新创建表，然后使用语法 `decimal(precision, scale)` 为列指定 decimal 数据类型。

无法再将浮点型或双精度 NaN 值转换为 bigint

错误消息：INVALID_CAST_ARGUMENT：无法将实数/双精度 NaN 转换为 bigint

原因：在 Athena 引擎版本 3 中，`NaN` 无法转换为 0 (`bigint`)。

建议的解决方案：确保在转换为 `bigint` 时 `float` 或 `double` 列中不存在 `NaN` 值。

uuid() 函数返回类型更改

以下问题会影响表和视图。

错误消息：Hive 类型不受支持：uuid

原因：在 Athena 引擎版本 2 中，`uuid()` 函数返回一个字符串，但在 Athena 引擎版本 3 中，它返回一个随机生成的伪 UUID（类型 4）。由于 Athena 不支持 UUID 列数据类型，因此在 Athena 引擎版本 3 中，无法再在 CTAS 查询中直接使用 `uuid()` 函数来生成 UUID 列。

例如，以下 `CREATE TABLE` 语句可在 Athena 引擎版本 2 中成功完成，但在 Athena 引擎版本 3 中，将返回 NOT_SUPPORTED：Hive 类型不受支持：uuid：

```
CREATE TABLE uuid_table AS
   SELECT uuid() AS myuuid
```

同样，以下 `CREATE VIEW` 语句可在 Athena 引擎版本 2 中成功完成，但在 Athena 引擎版本 3 中，将返回列 myuuid 的列类型无效：Hive 类型不受支持：uuid：

```
CREATE VIEW uuid_view AS
   SELECT uuid() AS myuuid
```

在 Athena 引擎版本 3 中查询在 Athena 引擎版本 2 中如此创建的视图时，会出现如下所示的错误：

VIEW_IS_STALE：第 1:15 行：视图 'awsdatacatalog.mydatabase.uuid_view' 已过时或处于无效状态：从查询视图投影到位置 0 的 uuid 类型的列 [myuuid] 无法强制转换为视图定义中存储的 varchar 类型的列 [myuuid]

建议的解决方案：创建表或视图时，使用 cast() 函数将 uuid() 的输出转换为 varchar，如以下示例所示：

```
CREATE TABLE uuid_table AS
   SELECT CAST(uuid() AS VARCHAR) AS myuuid
```

```
CREATE VIEW uuid_view AS
   SELECT CAST(uuid() AS VARCHAR) AS myuuid
```

CHAR 和 VARCHAR 强制转换问题

如果您在 Athena 引擎版本 3 中遇到 varchar 和 char 强制转换问题，请使用本节中的解决方法。如果您无法使用这些解决方法，请联系 AWS Support。

采用混合 CHAR 和 VARCHAR 输入时 CONCAT 函数失败

问题：在 Athena 引擎版本 2 中，以下查询将成功。

```
SELECT concat(CAST('abc' AS VARCHAR(20)), '12', CAST('a' AS CHAR(1)))
```

但是，在 Athena 引擎版本 3 上，相同的查询会失败，并显示以下内容：

错误消息：FUNCTION_NOT_FOUND：第 1:8 行：函数 concat 包含非预期参数（varchar(20)、varchar(2)、char(1)）。预期参数：concat(char(x), char(y))、concat(array(E), E) E、concat(E, array(E)) E、concat(array(E)) E、concat(varchar)、concat(varbinary)

建议的解决方案：使用 concat 函数时，转换为 char 或 varchar，但不要转换为两者的组合。

使用 CHAR 和 VARCHAR 输入时 SQL || 联接失败

在 Athena 引擎版本 3 中，双竖线 **||** 联接运算符需要 varchar 作为输入。输入不能是 varchar 和 char 类型的组合。

错误消息：TYPE_NOT_FOUND：第 1:26 行：类型未知：char(65537)

原因：使用 **||** 联接 char 和 varchar 的查询可能会产生错误，如以下示例所示。

```
SELECT CAST('a' AS CHAR) || CAST('b' AS VARCHAR)
```

建议的解决方案：联接 varchar varchar，如以下示例所示。

```
SELECT CAST('a' AS VARCHAR) || CAST('b' AS VARCHAR)
```

CHAR 和 VARCHAR UNION 查询失败

错误消息：NOT_SUPPORTED：Hive 类型不受支持：char(65536)。支持的 CHAR 类型：CHAR(<=255)

原因：尝试组合 `char` 和 `varchar` 的查询，如以下示例所示：

```
CREATE TABLE t1 (c1) AS SELECT CAST('a' as CHAR) as c1 UNION ALL SELECT CAST('b' AS
 VARCHAR) AS c1
```

建议的解决方案：在示例查询中，将 `'a'` 转换为 `varchar` 而不是 `char`。

在 CHAR 或 VARCHAR 强制转换之后出现不必要的空格

在 Athena 引擎版本 3 中，如果 `char(X)` 和 `varchar` 数据在形成数组或单列时强制为单一类型，则 `char(65535)` 为目标类型，并且每个字段都包含许多不需要的尾随空格。

原因：Athena 引擎版本 3 将 `varchar` 和 `char(X)` 强制转换为 `char(65535)`，然后在数据后面填充空格。

建议的解决方案：将每个字段显式转换为 `varchar`。

时间戳更改

将带时区的 Timestamp 转换为 varchar 时的行为更改

在 Athena 引擎版本 2 中，如果将带时区的 `Timestamp` 投影到 `varchar`，则导致某些时区文字发生变化（例如，会将 `US/Eastern` 更改为 `America/New_York`）。在 Athena 引擎版本 3 中不会出现此行为。

日期时间戳溢出引发错误

错误消息：Millis overflow: XXX（Millis 溢出：XXX）

原因：由于 Athena 引擎版本 2 中不会检查 ISO 8601 日期是否溢出，因此部分日期会产生负时间戳。Athena 引擎版本 3 会检查此溢出并引发异常。

建议的解决方法：确保时间戳在范围内。

不支持带时间的政治时区

错误消息：INVALID LITERAL（文字无效）

原因：查询类似于 `SELECT TIME '13:21:32.424 America/Los_Angeles'`。

建议的解决方法：避免使用带 `TIME` 的政治时区。

Timestamp 列中的精度不匹配会导致序列化错误

错误消息：SERIALIZATION_ERROR: Could not serialize column '*COLUMNZ*' of type 'timestamp(3)' at position *X*:*Y*（序列化错误：无法序列化位置 X:Y 处类型为“timestamp(3)”的列“COLUMNZ”）

COLUMNZ 是导致问题的列的输出名称。数字 *X*:*Y* 表示该列在输出中的位置。

原因：Athena 引擎版本 3 会进行检查，以确保数据中时间戳的精度与表规范中为列数据类型指定的精度相同。目前，此精度始终为 3。如果数据的精度大于此值，则查询会失败并提示错误。

建议的解决方法：检查数据，确保您的时间戳精度为毫秒。

Iceberg 表的 UNLOAD 和 CTAS 查询中的时间戳精度错误

错误消息：时间戳（6）的时间戳精度错误；配置的精度为毫秒

原因：Athena 引擎版本 3 会进行检查，以确保数据中时间戳的精度与表规范中为列数据类型指定的精度相同。目前，此精度始终为 3。如果数据的精度大于此值（例如，使用微秒而不是毫秒），则查询可能失败并显示注明的错误。

解决方案：要解决此问题，首先通过 CAST 将时间戳精度转换为 6，如创建 Iceberg 表的以下 CTAS 示例所示。请注意，必须将精度指定为 6 而不是 3，以免出现错误时间戳精度（3）不适用于 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
```

然后，由于 Athena 不支持时间戳 6，因此再次将该值转换为时间戳（例如，在视图中）。下面的示例基于 `my_iceberg_ctas` 表创建了一个视图。

```
CREATE OR REPLACE VIEW my_iceberg_ctas_view AS
SELECT cast(dt AS timestamp) AS dt
```

```
FROM my_iceberg_ctas
```

现在，将 ORC 文件中的 Long 类型读取为 Timestamp 或将 Timestamp 读取为 Long 类型时，都将导致错误“ORC 文件格式错误”

错误消息：Error opening Hive split ‘FILE (SPLIT POSITION)’ Malformed ORC file. (打开 Hive split“FILE (SPLIT POSITION)”错误 ORC 文件时出错。) Cannot read SQL type timestamp from ORC stream .long_type of type LONG [打开 Hive 拆分文件“FILE (SPLIT POSITION)”时出错。ORC 文件格式不正确。无法从 LONG 类型的 .long_type ORC 流中读取 SQL 类型时间戳]

原因：Athena 引擎版本 3 现在拒绝在 Long 数据类型与 Timestamp 或 Timestamp 与 Long 之间进行隐式强制转换。以前，Long 值会被视作纪元毫秒，并隐式转换为时间戳。

建议的解决方法：使用 from_unixtime 函数显式转换列，或使用 from_unixtime 函数为未来的查询创建其他列。

不支持时间加年和月间隔

错误消息：TYPE MISMATCH (类型不匹配)

原因：Athena 引擎版本 3 不支持时间加年和月间隔 (例如，SELECT TIME '01:00' + INTERVAL '3' MONTH)。

int96 Parquet 格式的时间戳溢出

错误消息：Invalid timeOfDayNanos (timeOfDayNanos 无效)

原因：int96 Parquet 格式的时间戳溢出。

建议的解决方法：找出存在问题的特定文件。然后使用众所周知的最新 Parquet 库重新生成数据文件，或者使用 Athena CTAS。如果问题仍然存在，请联系 Athena 支持人员并告知我们数据文件是如何生成的。

将字符串转换为时间戳时，日期和时间值之间需要空间

错误消息：INVALID_CAST_ARGUMENT：无法将值转换为时间戳。

原因：Athena 引擎版本 3 不再接受连字符作为 cast 输入字符串中日期和时间值之间的有效分隔符。例如，以下查询在 Athena 引擎版本 2 中起作用，但在 Athena 引擎版本 3 中不起作用：

```
SELECT CAST('2021-06-06-23:38:46' AS timestamp) AS this_time
```

建议的解决方案：在 Athena 引擎版本 3 中，将日期和时间之间的连字符替换为空格，如以下示例所示。

```
SELECT CAST('2021-06-06 23:38:46' AS timestamp) AS this_time
```

to_iso8601() 时间戳返回值更改

错误消息：无

原因：在 Athena 引擎版本 2 中，即使传递给函数的值不包含时区，to_iso8601 函数也会返回包含时区的时间戳。在 Athena 引擎版本 3 中，仅当传递的参数包含时区时，to_iso8601 函数才会返回包含时区的时间戳。

例如，以下查询将当前日期传递给 to_iso8601 函数两次：第一次作为包含时区的时间戳，第二次作为时间戳。

```
SELECT TO_ISO8601(CAST(CURRENT_DATE AS TIMESTAMP WITH TIME ZONE)),
 TO_ISO8601(CAST(CURRENT_DATE AS TIMESTAMP))
```

以下输出显示了每个引擎中的查询结果。

Athena 引擎版本 2：

#	_col0	_col1
1	2023-02-24T00:00:00.000Z	2023-02-24T00:00:00.000Z

Athena 引擎版本 3：

#	_col0	_col1
1	2023-02-24T00:00:00.000Z	2023-02-24T00:00:00.000

建议的解决方案：要复制之前的行为，可以先将时间戳值传递给 with_timezone 函数，然后再将其传递给函数 to_iso8601，如以下示例所示：

```
SELECT to_iso8601(with_timezone(TIMESTAMP '2023-01-01 00:00:00.000', 'UTC'))
```

结果

#	_col0
1	2023-01-01T00:00:00.000Z

at_timezone() 第一个参数必须指定日期

问题：在 Athena 引擎版本 3 中，`at_timezone` 函数不能将 `time_with_timezone` 值作为第一个参数。

原因：如果没有日期信息，就无法确定传递的值是夏令时还是标准时间。例如，`at_timezone('12:00:00 UTC', 'America/Los_Angeles')` 存在歧义，因为无法确定传递的值是太平洋夏令时间（PDT）还是太平洋标准时间（PST）。

限制

Athena 引擎版本 3 具有以下限制。

- 查询性能 – 许多查询在 Athena 引擎版本 3 上运行会更快，但某些查询计划可能与 Athena 引擎版本 2 有所不同。因此，某些查询的延迟或成本也可能有差异。
- Trino 和 Presto 连接器 – 不支持 Trino 和 Presto 连接器。使用 Amazon Athena 联合查询连接数据来源。有关更多信息，请参阅 使用 Amazon Athena 联合查询。
- 容错执行 – 不支持 Trino 容错执行（Trino Tardigrade）。
- 函数参数限制 - 函数的参数不能超过 127 个。有关更多信息，请参阅 函数调用的参数太多。

Athena 引擎版本 2 中引入了以下限制，以确保查询不会因资源限制而失败。用户无法配置这些限制。

- 结果元素的数量 – 结果元素的数量 n 对于函数 `min(col, n)`、`max(col, n)`、`min_by(col1, col2, n)` 和 `max_by(col1, col2, n)` 将限制在 10,000 或更少。
- GROUPING SETS – 分组集中的最大切片数为 2048。
- 最大文本文件行长度 – 文本文件的默认最大行长为 200MB。
- 序列函数最大结果大小 – 序列函数的最大结果大小为 50000 个条目。例如，`SELECT sequence(0,45000,1)` 成功，但 `SELECT sequence(0,55000,1)` 会失败并显示错误消息 The result of the sequence function must not have more than 50000 entries（序列函数的结果不得超过 50000 个条目）。此限制适用于序列函数的所有输入类型，包括时间戳。

Athena 引擎版本 2

Athena 引擎版本 2 引入了以下更改。

改进功能和新功能

- EXPLAIN 和 EXPLAIN ANALYZE – 您可以使用 Athena 中的 `EXPLAIN` 语句查看 SQL 查询的执行计划。使用 `EXPLAIN ANALYZE` 查看 SQL 查询的分布式执行计划以及每项操作的成本。有关更多信息，请参阅 在 Athena 中使用 EXPLAIN 和 EXPLAIN ANALYZE。
- 联合查询 – Athena 引擎版本 2 支持联合查询。有关更多信息，请参阅 使用 Amazon Athena 联合查询。
- 地理空间函数 – 添加了超过 25 个地理空间函数。有关更多信息，请参阅 Athena 引擎版本 2 中新增的地理空间函数。
- 嵌套架构 – 添加了对读取嵌套架构的支持，从而降低了成本。
- 预编译语句 – 使用预编译语句重复执行具有不同查询参数的同一查询。预编译语句包含占位符参数，其值在运行时传递。预编译语句有助于防止 SQL 注入攻击。有关更多信息，请参阅 使用参数化查询。
- 架构演变支持 – 添加了 Parquet 格式数据的架构演进支持。
  - 添加了对从分区架构与表架构不同的分区读取数组、映射或行类型列的支持。创建分区后，更新表架构时可能会发生这种情况。更改后的列类型必须兼容。对于行类型，可以添加或删除尾随字段，但相应字段（按顺序）必须具有相同的名称。

  - ORC 文件现在可以具有缺少字段的结构列。这可在不重写 ORC 文件的情况下更改表架构。
  - ORC 结构列现在是按名称而不是顺序映射的。这可以正确处理 ORC 文件中缺少或多出的结构字段。
- SQL OFFSET – SELECT 语句现在支持 SQL OFFSET 子句。有关更多信息，请参阅 SELECT。
- UNLOAD 语句 – 您可以使用 UNLOAD 语句将 SELECT 查询的输出以 PARQUET、ORC、AVRO 和 JSON 格式写入。有关更多信息，请参阅 UNLOAD。

分组、联接和子查询改进

- 复杂分组 – 增加了对复杂分组操作的支持。
- 关联的子查询 – 添加了对 IN 谓词中相关子查询和需要强制转换的相关子查询的支持。
- CROSS JOIN – 添加了针对 LATERAL 的派生表对 CROSS JOIN 的支持。
- GROUPING SET – 添加了对使用 GROUPING SETS 的查询以聚合的形式对 ORDER BY 子句的支持。
- Lambda 表达式 – 添加了对在 Lambda 表达式中取消引用行字段的支持。
- 半连接中的空值 – 添加了对半连接左侧空值的支持（即带有 IN 谓词的子查询）。
- 空间联接 – 添加了对广播空间连接和空间左连接的支持。
- 溢出到磁盘 – 对于内存密集型 INNER JOIN 和 LEFT JOIN 操作，Athena 会将中间操作结果卸载到磁盘中。这样可以执行需要大量内存的查询。

数据类型增强功能

- INT for INTEGER – 添加了对 INT 的支持，作为 INTEGER 数据的别名。
- INTERVAL 类型 – 添加了对 INTERVAL 类型的转换支持。
- IPADDRESS – 添加了一个新的 IPADDRESS 类型来表示 DML 查询中的 IP 地址。增加了对在 VARBINARY 类型和 IPADDRESS 类型之间的转换。DDL 查询中无法识别 IPADDRESS 类型。
- IS DISTINCT FROM – 添加了对于 JSON 和 IPADDRESS 类型的 IS DISTINCT FROM 支持。
- 空值等效性检查 – ARRAY、MAP, 和 ROW 数据结构中的空值等效性检查现已得到支持。例如，表达式 ARRAY ['1', '3', null] = ARRAY ['1', '2', null] 返回 false。以前，空元素会返回错误消息 comparison not supported（不支持比较）。
- 行类型强制转换– 现在允许在行类型之间进行强制操作，而不考虑字段名称。以前，只有当源类型中的字段名与目标类型匹配时，或者当目标类型具有匿名字段名称时，才可强制转换为另一行类型。
- 时间减法 – 为所有 TIME 和 TIMESTAMP 类型执行减法。

- Unicode – 在字符串文字中添加了对转义 Unicode 序列的支持。
- VARBINARY 连接 – 添加了对 VARBINARY 值连接的支持。

  窗口值函数 – 窗口值函数现在支持 IGNORE NULLS 和 RESPECT NULLS。

函数的其他输入类型

以下函数现在接受其他输入类型。有关各个函数的更多信息，请访问 Presto 文档的相应链接。

- approx_distinct() – approx_distinct() 函数现在支持以下类型：INTEGER、SMALLINT、TINYINT、DECIMAL、REAL、DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONE、TIME、TIME WITH TIME ZONE、IPADDRESS 和 CHAR。
- Avg()、sum() – avg() 和 sum() 聚合函数现在支持 INTERVAL 数据类型。
- Lpad()、rpad() – lpad 和 rpad 函数现在可以执行 VARBINARY 输入。
- Min()、max() – min() 和 max() 聚合函数现在允许在查询分析时使用未知的输入类型，以便您可以将函数用于 NULL 文本。
- regexp_replace() – 添加了 regexp_replace() 函数的变体，可以为每个替换执行 Lambda 函数。
- Sequence() – 已将 DATE 变体添加到 sequence() 函数，包括带有隐式一天步骤增量的变体。
- ST_Area() – ST_Area() 地理空间函数现在支持所有几何类型。
- Substr() – substr 函数现在可以执行 VARBINARY 输入。
- zip_with() – 长度不匹配的数组现在可以与 zip_with() 共同使用。缺少的位置将用空值填充。以前，传递不同长度的数组时会引发错误。这种更改可能会使得难以区分最初为空的值和添加来填充相同长度数组的值。

已添加的函数

下面的列表包含在 Athena 引擎版本 2 中新启动的函数。该列表不包括地理空间函数。有关地理空间函数的列表，请参阅 Athena 引擎版本 2 中新增的地理空间函数。

有关各个函数的更多信息，请访问 Presto 文档的相应链接。

聚合函数

reduce_agg()

数组函数和运算符

array_sort() – 这个函数的变体添加了一个 Lambda 函数作为比较器。

ngrams()

二进制函数和运算符

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()

日期与时间函数和运算符

millisecond()

parse_duration()

to_milliseconds()

映射函数和运算符

multimap_from_entries()

数学函数和运算符

inverse_normal_cdf()

wilson_interval_lower()

wilson_interval_upper()

分位数摘要函数

分位数摘要函数和 qdigest 分位数摘要类型已添加。

字符串函数和运算符

hamming_distance()

split_to_multimap()

性能改进

在 Athena 引擎版本 2 中，以下功能的性能得到了改进。

查询性能

- 广播联接性能 – 通过在 Worker 节点中应用动态分区修剪来提高广播联接性能。
- 分桶表 – 在正在写入的数据已经适当分区（例如，当输出来自分桶连接时）时，改进了写入分桶表的性能。
- DISTINCT – 改进了一些使用 DISTINCT 的查询的性能。

  动态筛选和分区修剪 – 改进提高了性能，减少了查询中扫描的数据量。
- 筛选条件和投影操作 – 筛选条件和投影操作现在始终由列处理（如可能）。引擎在有效的情况下将自动采用字典编码。
- 收集交换 – 改进了通过收集交换查询的性能。
- 全局聚合 – 改进了执行筛选全局聚合的某些查询的性能。
- GROUPING SETS, CUBE, ROLLUP – 改进了涉及您可以用来在单个查询中聚合多组列的 GROUPING SETS、CUBE 或者 ROLLUP 的查询性能。
- 高度选择性的筛选条件 – 改进了具有高度选择性筛选条件的查询的性能。
- JOIN 和 AGGREGATE 操作 – JOIN 和 AGGREGATE 操作的性能得到了增强。
- LIKE – 改进了在 information_schema 表的列上使用 LIKE 谓词的查询性能。
- ORDER BY 和 LIMIT – 改进了涉及 ORDER BY 和 LIMIT 的查询的计划、性能和内存使用情况，以避免不必要的数据交换。

- ORDER BY – ORDER BY 操作现在默认分布，从而能够使用更大 ORDER BY 子句。
- ROW 类型转换 – 改进了在 ROW 类型之间转换时的性能。
- 结构类型 – 改进了处理结构类型并包含扫描、连接、聚合或表写入的查询的性能。
- 表扫描 – 引入了优化规则，以避免在某些情况下重复表扫描。
- UNION – 改进了 UNION 查询的性能。

查询计划性能

- 规划性能 – 改进了联接多个包含大量列的表的查询规划性能。
- 谓词评估 – 改进了规划中谓词下推期间的谓词评估性能。
- 转换的谓词下推支持 – 当值列表中的值要求进行转换以匹配列类型时，支持对 *<column>* IN *<values list>* 谓词的谓词下推。
- 谓词推理和下推 – 为使用 *<symbol>* IN *<subquery>* 谓词的查询扩展了谓词推理和下推。
- 超时 – 修复了在极少数情况下可能导致查询计划超时的错误。

联接性能

- 使用映射列连接 – 改进了包含映射列的连接和聚合的性能。
- 单独使用非相等条件连接 – 通过使用嵌套循环连接而不是散列连接来改进仅具有非相等条件的连接的性能。
- 外部联接 – 对于涉及外连接的查询，现在会自动选择连接分配类型。
- 函数连接范围 – 改进了连接的性能，其中条件是函数的范围（例如 `a JOIN b ON b.x < f(a.x) AND b.x > g(a.x)`）。
- 溢出到磁盘 – 修复了与溢出到磁盘相关的错误和内存问题，以提高性能并减少 JOIN 操作中的内存错误。

子查询性能

- 关联的 EXISTS 子查询 – 改进了关联的 EXISTS 子查询性能。
- 带有相等谓词的关联子查询 – 改进了对包含相等谓词的关联子查询的支持。
- 带有不等谓词的关联子查询 – 改进了包含不等谓词的关联子查询的性能。
- 子查询上的 count(*) 聚合 – 改进了具有已知常量基数的子查询的 `count(*)` 聚合性能。

- 外部查询筛选条件传播 – 在外部查询的筛选条件可以传播到子查询时，改进了相关子查询的性能。

函数性能

- 聚合窗口函数 – 改进了聚合窗口函数的性能。
- element_at() – 为映射改进了 `element_at()` 的性能，使其成为恒定时间，而非与地图大小成正比。
- Grouping() – 改进了涉及 `grouping()` 的查询的性能。
- JSON 转换 – 改进了从 JSON 转换到 ARRAY 或 MAP 类型的性能。
- 映射返回函数 – 改进了返回映射的函数性能。
- 映射到映射转换 – 改进了映射到映射转换的性能。
- Min() 和 max() – `min()` 和 `max()` 函数已经过优化，以避免不必要的对象创建，从而减少垃圾回收开销。
- row_number() – 改进了使用 `row_number()` 并生成了行号筛选条件的查询性能和内存使用情况。
- 窗口函数 – 改进了包含具有相同 `PARTITION BY` 和 `ORDER BY` 子句的窗口函数的查询性能。
- 窗口函数 – 改进了某些具有类似规格窗口函数（例如 LAG）的性能。

地理空间性能

- 几何体序列化 – 改进了几何体值的序列化性能。
- 地理空间函数 – 改进了 `ST_Intersects()`、`ST_Contains()`、`ST_Touches()`、`ST_Within()`、`ST_Overlaps()`、`ST_Dis` 和 `array_intersect()` 的性能。
- ST_Distance() – 改进了涉及 `ST_Distance()` 函数的连接查询的性能。
- ST_Intersection() – 为与坐标轴对齐的矩形优化了 `ST_Intersection()` 函数（例如，`ST_Envelope()` 和 `bing_tile_polygon()` 函数生成的多边形）。

JSON 相关改进

映射函数

- 改进了所有案例中从 `O(n)` 到 `O(1)` 映射下标的性能。以前，只有某些函数和读取器生成的映射才能利用这一改进。
- 增加了 `map_from_entries()` 和 `map_entries()` 函数。

转换

- 增加了从 REAL、TINYINT 或 SMALLINT 转换到 JSON 的能力。
- 现在，您可以将 JSON 转换为 ROW，即使 JSON 不包含 ROW 中的所有字段。
- 提高了 CAST(json_parse(...) AS ...) 的性能。
- 改进了从 JSON 转换到 ARRAY 或者 MAP 类型的性能。

新 JSON 函数

- is_json_scalar()

重大更改

突破性更改包括错误修复、对地理空间函数的更改、替换函数以及引入限制。ANSI SQL 合规性的改进可能会中断依赖于非标准行为的查询。

错误修复

以下更改修正了导致查询成功运行但结果不准确的行为问题。

- fixed_len_byte_array Parquet 列现在被接受为 DECIMAL – 如果查询在 Parquet Schema 中被注释为 DECIMAL，则对 fixed_len_byte_array 类型的 Parquet 列的查询成功并返回正确的值。不含 DECIMAL 注释的 fixed_len_byte_array 列查询将失败，并出现错误。以往，没有 DECIMAL 注释的 fixed_len_byte_array 列查询将会成功，但会返回令人难以理解的值。
- json_parse() 不再忽略尾随字符 – 以往，诸如 [1,2]abc 等输入将成功解析为 [1,2]。使用尾随字符现在会生成错误消息 Cannot convert '[1, 2]abc' to JSON（无法将 '[1, 2]abc' 转换为 JSON）。
- Round() 小数精度校正 – 若 x 是 DECIMAL，或 x 是小数位数 0 的 DECIMAL，且 d 是负整数，则 round(x, d) 现在会正确舍入 x。以往，在这些情况下没有进行四舍五入。
- round(x, d) 和 truncate(x, d) – 函数 round(x, d) 和 truncate(x, d) 的签名中的参数 d 现在的类型为 INTEGER。以往，d 的类型可能是 BIGINT。
- 带有重复键的 map() – map() 现在会在重复键上引发错误，而不是静默地生成损坏的映射。当前使用重复键构建映射值的查询将会在失败时显示错误。
- map_from_entries() 引发一个带有空条目的错误 – map_from_entries() 现在会在输入数组包含空条目时引发错误。通过传递 NULL 作为一个值来构建映射的查询现在将会失败。
- 表 – 不能再创建具有不受支持的分区类型的表。

- 改进统计函数的数值稳定性 – 统计函数 corr()、covar_samp()、regr_intercept() 和 regr_slope() 的数值稳定性已改进。
- Parquet 中定义的 TIMESTAMP 精度现将正确地实施 – Parquet 架构中 TIMESTAMP 值的精度以及为 TIMESTAMP 列定义的精度现在必须匹配。不匹配的精度会导致不正确的时间戳。
- 时区信息 – 现在将使用 Java 1.8 软件开发工具包的 java.time 软件包计算时区信息。
- INTERVAL_DAY_TO_SECOND 与 INTERVAL_YEAR_TO_MONTH 数据类型的 SUM – 您现在无法再直接使用 SUM(NULL)。为了使用 SUM(NULL)，将 NULL 强制转换为 BIGINT、DECIMAL、REAL、DOUBLE、INTERVAL_DAY_TO_SECOND 或者 INTERVAL_YEAR_TO_MONTH 等数据类型。

地理空间函数的更改

对地理空间函数所做的更改包括以下各项。

- 函数名更改 – 某些函数名称已更改。有关更多信息，请参阅 Athena 引擎版本 2 中的地理空间函数名称更改。
- VARBINARY 输入 – VARBINARY 类型不再直接支持地理空间函数的输入。例如，要直接计算几何的面积，现在必须将几何体以 VARCHAR 或者 GEOMETRY 格式输入。解决方法是使用转换函数，如以下示例所示。
  - 要使用 ST_area() 以计算采用 Well-Known Binary (WKB) 格式的 VARBINARY 输入的面积，请将输入传递给 ST_GeomFromBinary()，例如：

    ```
    ST_area(ST_GeomFromBinary(<wkb_varbinary_value>))
    ```

  - 要使用 ST_area() 来计算采用传统二进制格式的 VARBINARY 输入，请首先将相同的输入传递给 ST_GeomFromLegacyBinary() 函数，例如：

    ```
    ST_area(ST_GeomFromLegacyBinary(<legacy_varbinary_value>))
    ```

- ST_ExteriorRing() 和 ST_Polygon() – ST_ExteriorRing() 和 ST_Polygon() 现在仅接受多边形作为输入。以前，这些函数错误地接受了其他几何体。
- ST_Distance() – 根据 SQL/MM 规范的要求，如果其中一个输入为空几何体，则 ST_Distance() 函数现在会返回 NULL。以往，将会返回 NaN。

ANSI SQL 合规性

以下语法和行为问题已得到更正，以遵循 ANSI SQL 标准。

- Cast() 操作 – 从 REAL 或 DOUBLE 到 DECIMAL 的 Cast() 操作现在符合 SQL 标准。例如，`cast (double '10000000000000000000000000000000000' as decimal(38))` 以前返回 `10000000000000000053661622043934 72` 但现在返回 `10000000000000000000000000000000000`。
- JOIN ... USING – `JOIN ... USING` 现在符合标准 SQL 语义。以往，`JOIN ... USING` 需要限定列中的表名，而两个表中的列将出现在输出中。表限定现在无效，而列仅在输出中出现一次。
- 删除了 ROW 类型的文字 – ROW 类型的文字格式 `ROW<int, int>(1, 2)` 不再受支持。现在将使用语法 `ROW(1 int, 2 int)`。
- 分组聚合语义 – 分组聚合使用 `IS NOT DISTINCT FROM` 语义而不是等效语义。分组聚合现在会返回正确的结果，并在 NaN 浮点数值进行分组时展现出更好的性能。现在支持对包含空值的映射、列表和行类型进行分组。
- 不再允许带引号的类型 – 根据 ANSI SQL 标准，数据类型不能再用引号括起来。例如，`SELECT "date" '2020-02-02'` 不再是有效的查询。请改用语法 `SELECT date '2020-02-02'`。
- 匿名行字段访问 – 匿名行字段不能再使用语法 `[.field0, .field1, ...]` 访问。
- 复杂分组操作 – 复杂的分组操作 `GROUPING SETS`、`CUBE` 和 `ROLLUP` 不支持对由输入列组成的表达式进行分组。只允许使用列名。

替换函数

以下函数不再受支持，已由生成相同结果的语法替换。

- information_schema.__internal_partitions__ – Athena 引擎版本 2 不再支持使用 `__internal_partitions__`。如需等效的语法，请使用 `SELECT * FROM "<table_name>$partitions"` 或者 `SHOW PARTITIONS`。有关更多信息，请参阅 列出特定表的分区。
- 替换的地理空间函数 – 有关名称已更改的地理空间函数的列表，请参阅 Athena 引擎版本 2 中的地理空间函数名称更改。

限制

Athena 引擎版本 2 中引入了以下限制，以确保查询不会因资源限制而失败。用户无法配置这些限制。

- 结果元素的数量 – 结果元素的数量 n 对于函数 `min(col, n)`、`max(col, n)`、`min_by(col1, col2, n)` 和 `max_by(col1, col2, n)` 将限制在 10,000 或更少。
- GROUPING SETS – 分组集中的最大切片数为 2048。

- 最大文本文件行长度 – 文本文件的默认最大行长为 200MB。
- 序列函数最大结果大小 – 序列函数的最大结果大小为 50000 个条目。例如，`SELECT sequence(0,45000,1)` 成功，但 `SELECT sequence(0,55000,1)` 会失败并显示错误消息 The result of the sequence function must not have more than 50000 entries（序列函数的结果不得超过 50000 个条目）。此限制适用于序列函数的所有输入类型，包括时间戳。

Athena 的 SQL 参考

Amazon Athena 支持数据定义语言 (DDL) 和数据操作语言 (DML) 的语句、函数、运算符和数据类型的子集。除某些例外，Athena DDL 基于 HiveQL DDL，而 Athena DML 基于 Trino。有关 Athena 引擎版本的更多信息，请参阅 Athena 引擎版本控制。

主题

- Amazon Athena 中的数据类型
- DML 查询、函数和运算符
- DDL 语句
- Amazon Athena 中 SQL 查询的注意事项和限制

Amazon Athena 中的数据类型

运行 `CREATE TABLE` 时，您可以指定列名以及每列可以包含的数据类型。所创建的表存储在 AWS Glue Data Catalog 中。

为了促进与其他查询引擎的互操作性，Athena 对 DDL 语句（例如 `CREATE TABLE`）使用 Apache Hive 数据类型名称。对于像 `SELECT`、`CTAS` 和 `INSERT INTO` 这样的 DML 查询，Athena 则使用 Trino 数据类型名称。下表显示了 Athena 支持的数据类型。如果 DDL 和 DML 类型在名称、可用性或语法方面存在差异，则会显示在不同的列中。

DDL	DML	描述
BOOLEAN		值包括 `true` 和 `false`。
TINYINT		一个 8 位有符号的整数，采用二进制补码格式，最小值为 -2^7，最大值为 2^7-1。

DDL	DML	描述
SMALLINT		一个 16 位有符号的整数，采用二进制补码格式，最小值为 -2^{15}，最大值为 $2^{15}-1$。
INT，INTEGER		一个 32 位有符号的值，采用二进制补码格式，最小值为 -2^{31}，最大值为 $2^{31}-1$。
BIGINT		一个 64 位有符号的整数，采用二进制补码格式，最小值为 -2^{63}，最大值为 $2^{63}-1$。
FLOAT	REAL	一个 32 位有符号的单精度浮点数。范围为 1.40129846432481707e-45 至 3.40282346638528860e+38，可以为正数，也可以为负数。遵循 IEEE 浮点运算标准（IEEE 754）。
DOUBLE		一个 64 位有符号的双精度浮点数。范围为 4.94065645841246544e-324d 至 1.79769313486231570e+308d，可以为正数，也可以为负数。遵循 IEEE 浮点运算标准（IEEE 754）。
DECIMAL(*precision* ,*scale*)		*precision* 是总位数；*scale*（可选）是小数部分的位数，默认值为 0。例如，使用以下类型定义：decimal(11,5) 、decimal(15) 。最大##值为 38，最大##值为 38。
CHAR、CHAR(*length*)		固定长度字符数据，指定长度介于 1 到 255 之间，例如 char(10)。如果指定了 *length*，则读取字符串时，字符串将按指定长度进行截断。如果底层数据字符串较长，则底层数据字符串将保持不变。 有关更多信息，请参阅 CHAR Hive 数据类型。
string	VARCHAR	长度可变的字符数据。
VARCHAR(*length*)		具有最大读取长度的长度可变的字符数据。读取字符串时，字符串将按指定长度进行截断。如果底层数据字符串较长，则底层数据字符串将保持不变。

DDL	DML	描述
BINARY	VARBINARY	长度可变的二进制数据。
TIME		一天中的某个时间，精确到毫秒。
不可用	TIME(*precision*)	一天中的某个时间，具有具体精度。TIME(3) 等同于 TIME。
不可用	带时区的时间	某个时区一天中的某个时间。应将时区指定为 UTC 的偏移量。
DATE		包含年、月、日的某个日历日期。
TIMESTAMP	TIMESTAMP、TIMESTAMP WITHOUT TIME ZONE	某个日历日期一天中的某个时间，精确到毫秒。
不可用	TIMESTAMP(*precision*)、TIMESTAMP(*precision*) WITHOUT TIME ZONE	某个日历日期一天中的某个时间，具有具体精度。TIMESTAMP(3) 等同于 TIMESTAMP 。
不可用	TIMESTAMP（有时区）	在某个时区中，某个日历日期一天中的某个时间。可以将时区指定为 UTC 的偏移量、IANA 时区名称或使用 UTC、UT、Z 或 GMT。
不可用	TIMESTAMP(*precision*) WITH TIME ZONE	在某个时区中，某个日历日期一天中的某个时间，具有具体精度。
不可用	INTERVAL YEAR TO MONTH	间隔为一个月或数个整月

DDL	DML	描述
不可用	INTERVAL DAY TO SECOND	间隔为一或数秒/分钟/小时/日
ARRAY<*element_type* >	ARRAY[*element_type*]	值的数组。所有值的类型必须相同。
MAP<*key_type*, *value_type* >	MAP(*key_type*, *value_type*)	可以通过键查找值的映射。所有键的值必须相同，并且所有值必须相同。
STRUCT<*field_name_1* :*field_type_1* , *field_name_2* :*field_type_2* , ...>	ROW(*field_name_1* *field_type_1* , *field_name_2* *field_type_2* , ...)	包含命名字段及其值的数据结构。
不可用	JSON	JSON 值类型，可以是 JSON 对象、JSON 数组、JSON 数字、JSON 字符串、`true`、`false` 或 `null`。
不可用	UUID	UUID，全称通用唯一标识符。
不可用	IPADDRESS	IPv4 或 IPv6 地址。
不可用	HyperLogLog	这些数据类型支持近似的函数内部构成。有关各种类型的更多信息，请访问 Trino 文档中相应条目的链接。
	P4HyperLogLog	
	SetDigest	
	QDigest	
	TDigest	

数据类型示例

下表列示了 DML 数据类型的示例文字。

数据类型	示例
BOOLEAN	`true` `false`
TINYINT	`TINYINT '123'`
SMALLINT	`SMALLINT '123'`
INT，INTEGER	`123456790`
BIGINT	`BIGINT '1234567890'` `2147483648`
REAL	`'123456.78'`
DOUBLE	`1.234`
DECIMAL(*precision* , *scale*)	`DECIMAL '123.456'`
CHAR、CHAR(*length*)	`CHAR 'hello world'`, `CHAR 'hello ''world''!'`
VARCHAR、VARCHAR(*length*)	`VARCHAR 'hello world'`, `VARCHAR 'hello ''world''!'`
VARBINARY	`X'00 01 02'`
TIME、TIME(*precision*)	`TIME '10:11:12'` ，`TIME '10:11:12.345'`
带时区的时间	`TIME '10:11:12.345 -06:00'`
DATE	`DATE '2024-03-25'`
TIMESTAMP、TIMESTAMP WITHOUT TIME	`TIMESTAMP '2024-03-25 11:12:13'` ，`TIMESTAMP '2024-03-25 11:12:13.456'`

数据类型	示例
ZONE、TIME STAMP(*precision*)、TIMESTA MP(*precision*) WITHOUT TIME ZONE	
TIMESTAMP WITH TIME ZONE、TIME STAMP(*precision*) WITH TIME ZONE	`TIMESTAMP '2024-03-25 11:12:13.456 Europe/Berlin'`
INTERVAL YEAR TO MONTH	`INTERVAL '3' MONTH`
INTERVAL DAY TO SECOND	`INTERVAL '2' DAY`
ARRAY[*element_type*]	`ARRAY['one', 'two', 'three']`
MAP(*key_type*, *value_type*)	`MAP(ARRAY['one', 'two', 'three'], ARRAY[1, 2, 3])` 请注意，映射根据键的数组和值的数组创建。

ROW(*field_name_1 field_type_1* ,*field_name_2 field_type_2* ,...)	`ROW('one', 'two', 'three')`
	
	请注意，以这种方式创建的行并无列名。要添加列名，可以使用 CAST，如下例所示：
	
	`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'`

数据类型的注意事项

大小限制

对于没有指定大小限制的数据类型，请记住，对于单行中的所有数据，实际限制为 32MB。有关更多信息，请参阅Amazon Athena 中 SQL 查询的注意事项和限制中的Row or column size limitation。

CHAR 和 VARCHAR

CHAR(*n*) 值始终具有 *n* 个字符数。例如，若将 'abc' 强制转换为 CHAR(7)，则会添加 4 个尾随空格。

对 CHAR 值的比较包括前导空格和尾随空格。

如果指定了 CHAR 或 VARCHAR 的长度，则读取字符串时，字符串将按指定长度进行截断。如果底层数据字符串较长，则底层数据字符串将保持不变。

要对 CHAR 或 VARCHAR 中的单引号进行转义，请额外使用一个单引号。

要在 DML 查询中将非字符串数据类型转换为字符串，请强制转换为 VARCHAR 数据类型。

要使用 `substr` 函数从 CHAR 数据类型返回指定长度的子字符串，就必须先将 CHAR 值强制转换为 VARCHAR。在以下示例中，`col1` 使用了 CHAR 数据类型。

```
substr(CAST(col1 AS VARCHAR), 1, 4)
```

DECIMAL

要在 SELECT 查询中将十进制值指定为文字（例如选择具有特定十进制值的行时），您可以指定 DECIMAL 类型，并在查询中将该十进制值列为单引号中的文字，如下例所示：

```
SELECT * FROM my_table
WHERE decimal_value = DECIMAL '0.12'
```

```
SELECT DECIMAL '44.6' + DECIMAL '77.2'
```

处理时间戳数据

本节介绍在 Athena 中处理时间戳数据的一些注意事项。

> **Note**
>
> Athena 引擎版本 2 和 Athena 引擎版本 3 对时间戳的处理方式有所改变。有关 Athena 引擎版本 3 中可能出现的时间戳相关错误以及建议的解决方案的信息，请参阅 Athena 引擎版本 3 参考中的 时间戳更改。

向 Amazon S3 对象写入时间戳数据的格式

向 Amazon S3 对象写入时间戳数据的格式取决于列数据类型和您使用的 SerDe 库。

- 如果您的表列类型为 `DATE`，Athena 预计数据的相应列或属性是 ISO 格式 `YYYY-MM-DD` 的字符串，或者是内置的日期类型（例如 Parquet 或 ORC 的日期类型）。
- 如果您的表列类型为 `TIME`，Athena 预计数据的相应列或属性是 ISO 格式 `HH:MM:SS` 的字符串，或者是内置的时间类型（例如 Parquet 或 ORC 的日期类型）。
- 如果您的表列类型为 `TIMESTAMP`，Athena 预计数据的相应列或属性是 `YYYY-MM-DD HH:MM:SS.SSS` 格式的字符串（注意日期和时间之前的空格），或者是内置的时间类型（例如 Parquet、ORC 或 Ion 的时间类型）。

> **Note**
>
> OpenCSVSerDe 时间戳是个例外，必须编码为毫秒解析的 UNIX 纪元。

确保时间分区数据与记录中的时间戳字段相匹配

数据的创建者必须确保分区值与分区内的数据一致。例如，若数据具有 `timestamp` 属性，并且使用 Firehose 将数据加载到 Amazon S3 中，则您必须使用动态分区，因为 Firehose 的默认分区基于挂钟。

使用字符串作为分区键的数据类型

出于性能考虑，最好将 `STRING` 用作分区键的数据类型。尽管在您使用 `DATE` 类型时，Athena 会将 `YYYY-MM-DD` 格式的分区值识别为日期，但这可能会导致性能不佳。因此，我们建议您为分区键改用 `STRING` 数据类型。

如何为同时按时间分区的时间戳字段编写查询

如何为按时间分区的时间戳字段编写查询取决于要查询的表的类型。

Hive 表

由于 Athena 中最常用的 Hive 表，因此查询引擎对列和分区键之间的关系一无所知。因此，您必须始终在查询中为列和分区键添加谓词。

例如，假设您有一 `event_time` 列和一个 `event_date` 分区键，并且想要查询 23:00 到 03:00 之间的事件。在本例中，您必须在查询中同时包含列和分区键的谓词，如下例所示。

```
WHERE event_time BETWEEN start_time AND end_time
  AND event_date BETWEEN start_time_date AND end_time_date
```

Iceberg 表

使用 Iceberg 表，您可以使用计算出的分区值，从而简化查询。例如，假设您的 Iceberg 表是使用如下的 `PARTITIONED BY` 子句创建的：

```
PARTITIONED BY (event_date month(event_time))
```

在本例中，查询引擎会根据 `event_time` 谓词的值自动修剪分区。因此，您的查询只需要为 `event_time` 指定谓词，如以下示例所示。

```
WHERE event_time BETWEEN start_time AND end_time
```

有关更多信息，请参阅 创建 Iceberg 表。

DML 查询、函数和运算符

Athena DML 查询引擎全面支持 Trino 和 Presto 语法，并增加了自己的改进。Athena 并未支持所有 Trino 和 Presto 功能。有关更多信息，请参阅本部分中特定语句的主题和 注意事项和限制。有关函数的信息，请参阅 Amazon Athena 中的函数。有关 Athena 引擎版本的更多信息，请参阅 Athena 引擎版本控制。

有关 DDL 语句的信息，请参阅 DDL 语句。有关不支持的 DDL 语句列表，请参阅 不支持的 DDL。

SELECT

从零个或多个表中检索数据行。

> **Note**
>
> 本主题提供了摘要信息以供参考。本文档不包含有关使用 SELECT 和 SQL 语言的综合信息。有关使用特定于 Athena 的 SQL 的信息，请参阅 Amazon Athena 中 SQL 查询的注意事项和限制 和 使用 Amazon Athena 运行 SQL 查询。有关在 Athena 中创建数据库、创建表和在表上运行 SELECT 查询的示例，请参见 开始使用。

摘要

```
[ 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**
>
> SQL SELECT 语句中的保留字必须用双引号括起来。有关更多信息，请参阅 SQL SELECT 语句中保留关键字的列表。

参数

[WITH with_query [, ...]]

您可以使用 WITH 来展平嵌套查询或简化子查询。

从 Athena 引擎版本 3 开始支持使用 WITH 子句创建递归查询。最大递归深度为 10。

WITH 子句在查询中位于 SELECT 列表之前，定义一个或多个子查询，以便在 SELECT 查询中使用。

每个子查询均定义一个临时表，与可在 FROM 子句中引用的视图定义类似。只有在查询运行时才使用这些表。

with_query 语法为：

```
subquery_table_name [ ( column_name [, ...] ) ] AS (subquery)
```

其中：

- subquery_table_name 是临时表的唯一名称，该临时表用于定义 WITH 子句子查询的结果。每个 subquery 都必须具有一个可在 FROM 子句中引用的表名称。
- column_name [, ...] 是可选的输出列名称列表。列名称数目必须等于或小于 subquery 定义的列数。
- subquery 是任意查询语句。

[ALL | DISTINCT] select_expression

select_expression 确定要选择的行。select_expression 可以使用以下格式之一：

```
expression [ [ AS ] column_alias ] [, ...]
```

```
row_expression.* [ AS ( column_alias [, ...] ) ]
```

```
relation.*
```

```
*
```

- expression [[AS] column_alias] 语法指定输出列。可选 [AS] column_alias 语法指定要用于输出中的列的自定义标题名称。
- 对于 row_expression.* [AS (column_alias [, ...])]，row_expression 是数据类型为 ROW 的任意表达式。行的字段定义要包含在结果中的输出列。
- 对于 relation.*，relation 的列包含在结果中。此语法不允许使用列别名。
- 星号 * 指定所有列都包含在结果集中。
- 在结果集中，列的顺序与 select 表达式的规范顺序相同。如果 select 表达式返回多列，则列顺序遵循源关系或行类型表达式中使用的顺序。
- 指定列别名后，别名将覆盖先前存在的列或行字段名称。如果 select 表达式没有列名，则输出中将显示索引为零的匿名列名（_col0、_col1、_col2, ...）。
- 默认值为 ALL。使用 ALL 会被视为与省略它相同；将会选定所有列的所有行，并保留重复项。
- 当列包含重复值时，请使用 DISTINCT 仅返回不同的值。

FROM from_item [, ...]

指示要查询的输入，其中 from_item 可以是视图、联接构造或如下所述的子查询。

from_item 可以是：

- table_name [[AS] alias [(column_alias [, ...])]]

  其中，table_name 是要从中选择行的目标表的名称，alias 是要提供 SELECT 语句输出的名称，column_alias 定义指定 alias 的列。

- 或者 -

- join_type from_item [ON join_condition | USING (join_column [, ...])]

  其中，join_type 为以下值之一：

  - [INNER] JOIN
  - LEFT [OUTER] JOIN
  - RIGHT [OUTER] JOIN
  - FULL [OUTER] JOIN
  - CROSS JOIN
  - ON join_condition | USING (join_column [, ...])，其中，如果使用 join_condition，您可以为多个表中的联接键指定列名称；如果使用 join_column，则要求 join_column 在两个表中都存在。

[WHERE condition]

根据您指定的 condition 筛选结果，其中 condition 通常具有以下语法。

```
column_name operator value [[[AND | OR] column_name operator value] ...]
```

###可以是比较器 =、>、<、>=、<=、<>、!= 之一。

以下子查询表达式也可以用于 WHERE 子句。

- [NOT] BETWEEN *integer_A* AND *integer_B* – 指定两个整数之间的范围，如以下示例所示。如果列数据类型为 varchar，则必须先将列转换为整数。

```
SELECT DISTINCT processid FROM "webdata"."impressions"
WHERE cast(processid as int) BETWEEN 1500 and 1800
```

```
ORDER BY processid
```

- [NOT] LIKE *value* – 搜索指定的模式。使用百分号 (%) 作为通配符，如以下示例所示。

```
SELECT * FROM "webdata"."impressions"
WHERE referrer LIKE '%.org'
```

- [NOT] IN (*value*[, *value*[, ...]) – 指定列的可能值列表，如以下示例所示。

```
SELECT * FROM "webdata"."impressions"
WHERE referrer IN ('example.com','example.net','example.org')
```

[GROUP BY [ALL | DISTINCT] grouping_expressions [, ...]]

将 SELECT 语句的输出分成多个具有匹配值的行。

ALL 和 DISTINCT 确定重复的分组集是否各自产生不同的输出行。如果省略，则会采用 ALL。

grouping_expressions 允许您执行复杂的分组操作。您可以使用复杂分组操作在单个查询中执行需要聚合多个列集的分析。

grouping_expressions 元素可以是对输入列执行的任何函数，如 SUM、AVG 或者 COUNT。

GROUP BY 表达式可以按照不在 SELECT 语句的输出中显示的输入列名称对输出进行分组。

所有输出表达式都必须是存在于 GROUP BY 子句中的聚合函数或列。

您可以使用单个查询来执行需要聚合多个列集的分析。

Athena 支持使用 GROUPING SETS、CUBE 和 ROLLUP 的复杂聚合。GROUP BY GROUPING SETS 将指定要分组的多个列表。GROUP BY CUBE 为给定的列集生成所有可能的分组集。GROUP BY ROLLUP 为给定的列集生成所有可能的小计。复杂的分组操作不支持对由输入列组成的表达式进行分组。只允许使用列名。

通常，您可以使用 UNION ALL 实现和这些 GROUP BY 操作相同的结果，但使用 GROUP BY 的查询具有一次读取数据的优点，而 UNION ALL 三次读取底层数据，因此可能会在数据源发生变化时造成不一致的结果。

[HAVING condition]

与聚合函数和 GROUP BY 子句一起使用。控制哪些组处于选中状态，从而消除不满足 condition 的组。此筛选在计算组和聚合之后发生。

[{ UNION | INTERSECT | EXCEPT } [ALL | DISTINCT] union_query]]

UNION、INTERSECT 和 EXCEPT 将多个 SELECT 语句的结果合并到单个查询中。ALL 或 DISTINCT 控制包含在最终结果集中的行的唯一性。

UNION 将第一个查询生成的行与第二个查询生成的行组合在一起。为了消除重复，UNION 会构建一个散列表，这会消耗内存。为了更好的性能，如果您的查询不要求消除重复项，请考虑使用 UNION ALL。将会按从左到右的顺序处理多个 UNION 子句，除非您使用圆括号显式定义处理顺序。

INTERSECT 仅返回第一个和第二个查询的结果中存在的行。

EXCEPT 返回第一个查询结果中的行，不包括第二个查询找到的行。

ALL 会导致包含所有行，即使这些行是相同的。

DISTINCT 只导致唯一行包含在组合结果集中。

[ORDER BY expression [ASC | DESC] [NULLS FIRST | NULLS LAST] [, ...]]

按一个或多个输出 expression 对结果集进行排序。

当子句包含多个表达式时，将根据第一个 expression 对结果集进行排序。然后，第二个 expression 应用于具有第一个表达式中的匹配值的行，以此类推。

每个 expression 可以指定 SELECT 中的输出列或按位置指定输出列的序号（从 1 开始）。

在任何 GROUP BY 或 HAVING 子句之后，作为最后一个步骤，会计算 ORDER BY。ASC 和 DESC 确定是按升序还是按降序对结果进行排序。

默认 null 排序是 NULLS LAST，无论是按升序还是按降序排序。

[OFFSET count [ROW | ROWS]]

使用 OFFSET 子句丢弃结果集中的一些前导行。如果 ORDER BY 子句存在，则 OFFSET 子句将在排序的结果集上进行评估，并且在丢弃跳过的行后该集保持排序状态。如果查询没有 ORDER BY 子句，则将任意丢弃行。如果 OFFSET 指定的计数等于或超过结果集的大小，则最终结果为空。

LIMIT [count | ALL]

将结果集中的行数限制为 count。LIMIT ALL 与省略 LIMIT 子句相同。如果查询中没有 ORDER BY 子句，则结果是任意的。

TABLESAMPLE [BERNOULLI | SYSTEM] (percentage)

可选运算符，用于根据采样方法从表中选择行。

BERNOULLI 选择每个位于表样本中的行，概率为 percentage。将会扫描表的所有物理块，并根据样本 percentage 和在运行时计算的随机值之间的比较来跳过某些行。

借助 SYSTEM，表被划分成数据的逻辑段，而且表按此粒度采样。

将会选择特定段中的所有行，或者根据样本 percentage 和在运行时计算的随机值之间的比较来跳过该段。SYSTEM 采样取决于连接器。此方法不保证独立采样概率。

[UNNEST (array_or_map) [WITH ORDINALITY]]

展开数组或映射到关系。数组扩展到单个列中。映射将会扩展为两列（key、value）。

您可以将 UNNEST 与多个参数一起使用，这些参数将扩展到多个列，其中的行数与最高基数参数相同。

其他列将使用空值填补。

WITH ORDINALITY 子句将序数列添加到末尾。

UNNEST 通常和 JOIN 一起使用，可以在 JOIN 的左侧引用关系中的列。

获取 Amazon S3 中源数据的文件位置

要查看表行中数据的 Amazon S3 文件位置，可以在 SELECT 查询中使用 "$path"，如以下示例所示：

```
SELECT "$path" FROM "my_database"."my_table" WHERE year=2019;
```

这将返回以下结果：

```
s3://DOC-EXAMPLE-BUCKET/datasets_mytable/year=2019/data_file1.json
```

要返回表中数据的 S3 文件名路径的排序唯一列表，可以使用 SELECT DISTINCT 和 ORDER BY，如以下示例所示。

```
SELECT DISTINCT "$path" AS data_source_file
FROM sampledb.elb_logs
ORDER By data_source_file ASC
```

要仅返回没有路径的文件名，您可以将 "$path" 作为一个参数传递给 regexp_extract 函数，如以下示例所示。

```
SELECT DISTINCT regexp_extract("$path", '[^/]+$') AS data_source_file
FROM sampledb.elb_logs
ORDER By data_source_file ASC
```

要从特定文件返回数据，请在 WHERE 子句中指定文件，如以下示例所示。

```
SELECT *,"$path" FROM my_database.my_table WHERE "$path" = 's3://DOC-EXAMPLE-BUCKET/
my_table/my_partition/file-01.csv'
```

有关详细信息和示例，请参阅知识中心文章：如何在 Amazon S3 源文件中查找 Athena 表中的某行？。

> **Note**
>
> 在 Athena 中，视图不支持 Hive 或 Iceberg 隐藏的元数据列 `$bucket`、`$file_modified_time`、`$file_size` 和 `$partition`。

转义单引号

要转义单引号，请在其前面加上另一个单引号，如以下示例所示。不要将这种情况与双引号混淆。

```
Select 'O''Reilly'
```

结果

```
O'Reilly
```

其他 资源

有关在 Athena 中使用 SELECT 语句的更多信息，请参阅以下资源。

有关此内容的信息	请参阅此
在 Athena 中运行查询	使用 Amazon Athena 运行 SQL 查询
使用 SELECT 创建表	从查询结果创建表（CTAS）
从 SELECT 查询中将数据插入到另一个表	INSERT INTO

有关此内容的信息	请参阅此
在 SELECT 语句中使用内置函数	Amazon Athena 中的函数
在 SELECT 语句中使用用户定义的函数	使用用户定义函数进行查询
查询 Data Catalog 元数据	查询 AWS Glue Data Catalog

INSERT INTO

基于在源表上运行的 SELECT 查询语句或作为语句一部分提供的一组 VALUES，将新行插入到目标表中。如果源表基于一种格式（例如 CSV 或 JSON）的基础数据，而目标表基于另一种格式（例如 Parquet 或 ORC），则可以使用 INSERT INTO 查询将所选数据转换为目标表的格式。

注意事项和限制

在 Athena 中使用 INSERT 查询时注意以下事项。

- 在具有以 Amazon S3 加密的基础数据的表上运行 INSERT 查询时，INSERT 查询写入的输出文件预设情况下不加密。如果要插入具有加密数据的表中，建议您对 INSERT 查询结果进行加密。

  有关使用控制台加密查询结果的更多信息，请参阅 加密在 Amazon S3 中存储的 Athena 查询结果。要启用加密，请使用 AWS CLI 或 Athena API，使用 StartQueryExecution 操作的 EncryptionConfiguration 属性来根据您的要求指定 Amazon S3 加密选项。
- 对于 INSERT INTO 语句，预期存储桶拥有者设置不适用于 Amazon S3 中的目标表位置。预期存储桶拥有者设置仅适用于您为 Athena 查询结果指定的 Amazon S3 输出位置。有关更多信息，请参阅 使用 Athena 控制台指定查询结果位置。
- 有关符合 ACID 的 INSERT INTO 语句，请参阅 更新 Iceberg 表数据 的 INSERT INTO 一节。

支持的格式和 SerDes

针对使用以下格式和 SerDes 从数据创建的表，您可以运行 INSERT 查询。

Data format（数据格式）	SerDe
Avro	org.apache.hadoop.hive.serde2.avro.AvroSerDe

Data format（数据格式）	SerDe
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
文本文件	org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe **Note** 支持 CSV、TSV 和自定义分隔文件。

不支持分桶表

分桶表上不支持 `INSERT INTO`。有关更多信息，请参阅 在 Athena 中进行分区和分桶。

不支持联合查询

`INSERT INTO` 不支持联合查询。尝试这样做可能会导致出现错误消息 This operation is currently not supported for external catalogs（外部目录目前不支持此操作）。有关联合查询的信息，请参阅 使用 Amazon Athena 联合查询。

分区

使用 `INSERT INTO` 或者 `CREATE TABLE AS SELECT` 查询进行分区时，请考虑本节中的要点。

限制

`INSERT INTO` 语句支持向目标表写入最多 100 个分区。如果您在包含超过 100 个分区的表上运行 `SELECT` 子句，则查询将失败，除非 `SELECT` 查询限制为 100 个或更少的分区。

有关绕过此限制的信息，请参阅使用 CTAS 和 INSERT INTO 绕过 100 分区限制。

列排序

INSERT INTO 或者 CREATE TABLE AS SELECT 语句期望分区列是 SELECT 语句中投影列列表的最后一列。

如果源表未分区，或者与目标表相比在不同的列上进行分区，则类似 INSERT INTO *destination_table* SELECT * FROM *source_table* 的查询会将源表最后一列中的值视为目标表中某个分区列的值。在尝试从未分区的表创建已分区的表时，请记住这一点。

资源

有关使用带有分区功能的 INSERT INTO 的更多信息，请参阅以下资源。

- 有关将已分区数据插入已分区表中的信息，请参阅 使用 CTAS 和 INSERT INTO 绕过 100 分区限制。
- 有关将未分区数据插入已分区表中的信息，请参阅 将 CTAS 和 INSERT INTO 用于 ETL 和数据分析。

写入 Amazon S3 的文件

作为 INSERT 命令的结果，Athena 将文件写入 Amazon S3 中的源数据位置。每个 INSERT 操作都会创建一个新文件，而不是附加到现有文件。文件位置取决于表和 SELECT 查询的结构（如果存在）。Athena 为每个 INSERT 查询生成一个数据清单文件。清单跟踪查询写入的文件。它在 Amazon S3 中保存到 Athena 查询结果位置。有关更多信息，请参阅 识别查询输出文件。

避免高度事务性的更新

在使用 INSERT INTO 向 Amazon S3 中的表添加行时，Athena 不会重写或修改现有文件，而是会将这些行作为一个或多个新文件写入。由于这些表会因小文件过多而导致查询性能降低，而且 PutObject 和 GetObject 等写入和读取操作会导致 Amazon S3 的成本增加，所以请在使用 INSERT INTO 时考虑以下选项：

- 降低对较大批量行运行 INSERT INTO 操作的频率。
- 对于大量的数据摄取，可以考虑使用 Amazon Data Firehose 之类的服务。
- 完全避免使用 INSERT INTO，而是将行累积到更大的文件中，然后将这些文件直接上传到 Amazon S3，以便 Athena 在其中进行查询。

查找孤立文件

如果 CTAS 或 INSERT INTO 语句失败，孤立数据可以保留在数据位置，可能会在后续查询中进行读取。若要查找孤立文件以执行检查或删除操作，您可以使用 Athena 提供的数据清单文件跟踪要写入的文件列表。有关更多信息，请参阅识别查询输出文件和 DataManifestLocation。

INSERT INTO...SELECT

指定要在表 source_table 上运行的查询，该表确定要插入第二个表 destination_table 中的行。如果 SELECT 查询指定 source_table 中的列，则这些列必须与 destination_table 中的那些列精确匹配。

有关 SELECT 查询的更多信息，请参阅 SELECT。

摘要

```
INSERT INTO destination_table
SELECT select_query
FROM source_table_or_view
```

示例

选择 vancouver_pageviews 表中的所有行并将其插入 canada_pageviews 表中：

```
INSERT INTO canada_pageviews
SELECT *
FROM vancouver_pageviews;
```

只选择 vancouver_pageviews 表中 date 列的值介于 2019-07-01 和 2019-07-31 之间的那些行，然后将它们插入 canada_july_pageviews：

```
INSERT INTO canada_july_pageviews
SELECT *
FROM vancouver_pageviews
WHERE date
    BETWEEN date '2019-07-01'
        AND '2019-07-31';
```

仅从 country 列中值为 usa 的那些行中选择 cities_world 表中 city 和 state 列中的值，然后将它们插入 cities_usa 表中的 city 和 state 列中：

```
INSERT INTO cities_usa (city,state)
SELECT city,state
FROM cities_world
    WHERE country='usa'
```

INSERT INTO...VALUES

通过指定列和值将行插入到现有表中。指定的列和关联的数据类型必须与目标表中的列和数据类型精确匹配。

> **Important**
>
> 我们不建议使用 VALUES 插入行，因为 Athena 会为每个 INSERT 操作生成文件。这会导致创建许多小文件并降低表的查询性能。要标识 INSERT 查询创建的文件，请检查数据清单文件。有关更多信息，请参阅 使用查询结果、最近查询和输出文件。

摘要

```
INSERT INTO destination_table [(col1,col2,...)]
VALUES (col1value,col2value,...)[,
       (col1value,col2value,...)][,
       ...]
```

示例

在以下示例中，城市表包含三个列：id、city、state、state_motto。id 列为类型 INT，所有其他列均为类型 VARCHAR。

在 cities 表中插入一行，并指定所有列值：

```
INSERT INTO cities
VALUES (1,'Lansing','MI','Si quaeris peninsulam amoenam circumspice')
```

在 cities 表中插入两行：

```
INSERT INTO cities
VALUES (1,'Lansing','MI','Si quaeris peninsulam amoenam circumspice'),
```

```
        (3,'Boise','ID','Esto perpetua')
```

删除

删除 Apache Iceberg 表中的行。DELETE 是事务性的，仅支持用于 Apache Iceberg 表。

摘要

要从 Iceberg 表中删除行，请使用以下语法。

```
DELETE FROM [db_name.]table_name [WHERE predicate]
```

有关更多信息和示例，请参阅 更新 Iceberg 表数据 的 DELETE 一节。

UPDATE

更新 Apache Iceberg 表中的行。UPDATE 是事务性的，仅支持用于 Apache Iceberg 表。

摘要

要更新 Iceberg 表中的行，请使用以下语法。

```
UPDATE [db_name.]table_name SET xx=yy[,...] [WHERE predicate]
```

有关更多信息和示例，请参阅 更新 Iceberg 表数据 的 UPDATE 一节。

MERGE INTO

有条件地更新行、删除行或将行插入到 Apache Iceberg 表中。单个语句可以组合更新、删除和插入操作。

> **Note**
>
> MERGE INTO 是事务性的，仅支持用于 Athena 引擎版本 3 中的 Apache Iceberg 表。

摘要

要从 Iceberg 表中有条件地更新、删除或插入行，请使用以下语法。

```
MERGE INTO target_table [ [ AS ]  target_alias ]
USING { source_table | query } [ [ AS ] source_alias ]
ON search_condition
when_clause [...]
```

when_clause 是以下语法之一：

```
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 支持任意数量的具有不同 MATCHED 条件的 WHEN 子句。条件子句会执行 MATCHED 状态和匹配条件所选择的第一个 WHEN 子句中的 DELETE、UPDATE 或 INSERT 操作。

对于每个源行，WHEN 子句会按顺序处理。仅执行第一个匹配的 WHEN 子句。后续子句将忽略。如果单个目标表行与多个源行匹配，会引发用户错误。

如果源行与任何 WHEN 子句均不匹配且没有 WHEN NOT MATCHED 子句，则会忽略源行。

在具有 UPDATE 操作的 WHEN 子句中，列值表达式可以引用目标或源的任何字段。如果是 NOT MATCHED 字句，INSERT 表达式可以引用源的任何字段。

示例

如果第一个表中不存在行，则以下示例会将第二个表中的行合并到第一个表中。请注意，VALUES 子句中列出的列必须以源表别名为前缀。INSERT 子句中列出的目标列不得以此为前缀。

```
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)
```

有关更多 MERGE INTO 示例，请参阅 更新 Iceberg 表数据。

OPTIMIZE

根据数据文件的大小和相关删除文件的数量，将数据文件重写为更优化的布局，从而优化 Apache Iceberg 表中的行。

> **Note**
>
> OPTIMIZE 是事务性的，仅支持用于 Apache Iceberg 表。

语法

以下语法摘要显示了如何优化 Iceberg 表的数据布局。

```
OPTIMIZE [db_name.]table_name REWRITE DATA USING BIN_PACK
  [WHERE predicate]
```

> **Note**
>
> WHERE 子句##中只允许使用分区列。指定非分区列将导致查询失败。

压缩操作按重写过程中扫描的数据量收费。REWRITE DATA 操作使用谓词选择包含匹配行的文件。如果文件中的任何行与谓词匹配，则会选择该文件进行优化。因此，要控制受压缩操作影响的文件数量，可以指定 WHERE 子句。

配置压缩属性

要控制要选择以进行压缩的文件的大小以及压缩后生成的文件大小，可以使用表属性参数。您可以使用 ALTER TABLE SET PROPERTIES 命令配置相关的表属性。

其他 资源

优化 Iceberg 表

VACUUM

VACUUM 语句通过删除不再需要的数据文件来对 Apache Iceberg 表进行表维护。

> **Note**
>
> VACUUM 是事务性的，仅支持用于 Athena 引擎版本 3 中的 Apache Iceberg 表。

建议在 Iceberg 表上运行 VACUUM 语句，以删除不再相关的数据文件，并减少元数据大小和存储消耗。请注意，由于 VACUUM 语句会对 Amazon S3 进行 API 调用，因此对向 Amazon S3 发出的相关请求会产生费用。

> **Warning**
>
> 如果您运行快照过期操作，则无法对过期快照进行时间旅行操作。

摘要

要删除 Iceberg 表不再需要的数据文件，请使用以下语法。

```
VACUUM [database_name.]target_table
```

要在名称以下划线开头的表（例如 _mytable）上运行 VACUUM，请将表名括在反引号中，如以下示例所示。如果在表名前面加上数据库名称，请不要将数据库名称括在反引号中。请注意，双引号不能代替反引号。

这种行为是 VACUUM 特有的。对于以下划线开头的表名，CREATE 和 INSERT INTO 语句不要求使用反引号。

```
VACUUM `_mytable`
VACUUM my_database.`_mytable`
```

另请注意，VACUUM 希望 Iceberg 数据位于 Amazon S3 文件夹中，而不是 Amazon S3 存储桶中。例如，如果 Iceberg 数据位于 s3://DOC-EXAMPLE-BUCKET/ 而不是 s3://DOC-EXAMPLE-BUCKET/myicebergfolder/，则 VACUUM 语句会失败并显示错误消息 GENERIC_INTERNAL_ERROR：文件系统位置中缺少路径：s3://DOC-EXAMPLE-BUCKET。

执行的操作

VACUUM 执行以下操作：

- 删除超过 vacuum_max_snapshot_age_seconds 表属性指定时间的快照。默认情况下，该属性设置为 432000 秒（5 天）。
- 删除不在保留期限内且超过 vacuum_min_snapshots_to_keep 表属性指定数量的快照。默认 为 1。

  您可以在 CREATE TABLE 语句中指定这些表属性。创建表后，您可以使用 ALTER TABLE SET PROPERTIES 语句对其进行更新。
- 删除所有由于快照删除而无法访问的元数据和数据文件。您可以通过设置 vacuum_max_metadata_files_to_keep 表属性来配置要保留的旧元数据文件的数量。默认值是 100。
- 删除超过 vacuum_max_snapshot_age_seconds 表属性中指定时间的孤立文件。孤立文件是表数据目录中不属于表状态的文件。

有关在 Athena 中创建和管理 Apache Iceberg 表的更多信息，请参阅 创建 Iceberg 表 和 管理 Iceberg 表。

在 Athena 中使用 EXPLAIN 和 EXPLAIN ANALYZE

EXPLAIN 语句显示指定 SQL 语句的逻辑或分布式执行计划，或验证 SQL 语句。您可以以文本格式或数据格式输出结果，以便渲染到图形中。

> **Note**
>
> 您可以在 Athena 控制台中查看查询的逻辑和分布式计划的图形展示，而无需使用 EXPLAIN 语法。有关更多信息，请参阅 查看 SQL 查询的执行计划。

EXPLAIN ANALYZE 语句既显示了指定 SQL 语句的分布式执行计划，也显示了 SQL 查询中每个操作的计算成本。您可以以文本或 JSON 格式输出结果。

注意事项和限制

Athena 中的 EXPLAIN 和 EXPLAIN ANALYZE 语句具有以下限制。

- 由于 EXPLAIN 查询不扫描任何数据，因此 Athena 不会对其收取费用。然而，因为 EXPLAIN 查询会调用 AWS Glue 检索表元数据，如果呼叫超过 Glue 免费套餐限制，则可能会产生 Glue 费用。
- 由于执行 EXPLAIN ANALYZE 查询，它们会扫描数据，Athena 会根据扫描的数据量收费。

- 在 Lake Formation 中定义的行或单元格筛选信息以及查询统计信息未在 EXPLAIN 和 EXPLAIN ANALYZE 的输出中显示。

EXPLAIN 语法

```
EXPLAIN [ ( option [, ...]) ] statement
```

option 的值可以是以下值之一：

```
FORMAT { TEXT | GRAPHVIZ | JSON }
TYPE { LOGICAL | DISTRIBUTED | VALIDATE | IO }
```

如果未指定 FORMAT 选项，则输出默认为 TEXT 格式。IO 类型提供了有关查询读取的表和架构的信息。IO 仅在 Athena 引擎版本 2 中受支持，并且只能以 JSON 格式返回。

EXPLAIN ANALYZE 语法

除了 EXPLAIN 中包含的输出，EXPLAIN ANALYZE 输出还包括指定查询的运行时统计信息，例如 CPU 使用率、输入的行数和输出的行数。

```
EXPLAIN ANALYZE [ ( option [, ...]) ] statement
```

option 的值可以是以下值之一：

```
FORMAT { TEXT | JSON }
```

如果未指定 FORMAT 选项，则输出默认为 TEXT 格式。由于 EXPLAIN ANALYZE 的所有查询都是 DISTRIBUTED，所以 TYPE 选项不适用于 EXPLAIN ANALYZE。

*##*可以是以下值之一：

```
SELECT
CREATE TABLE AS SELECT
INSERT
UNLOAD
```

EXPLAIN 示例

以下 EXPLAIN 示例的复杂程度从简单到复杂依次递增。

EXPLAIN 示例 1：使用 EXPLAIN 语句以文本格式显示查询计划

在以下示例中，EXPLAIN 演示了针对 Elastic Load Balancing 日志的 SELECT 查询的执行计划。格式默认为文本输出。

```
EXPLAIN
SELECT
   request_timestamp,
   elb_name,
   request_ip
FROM sampledb.elb_logs;
```

结果

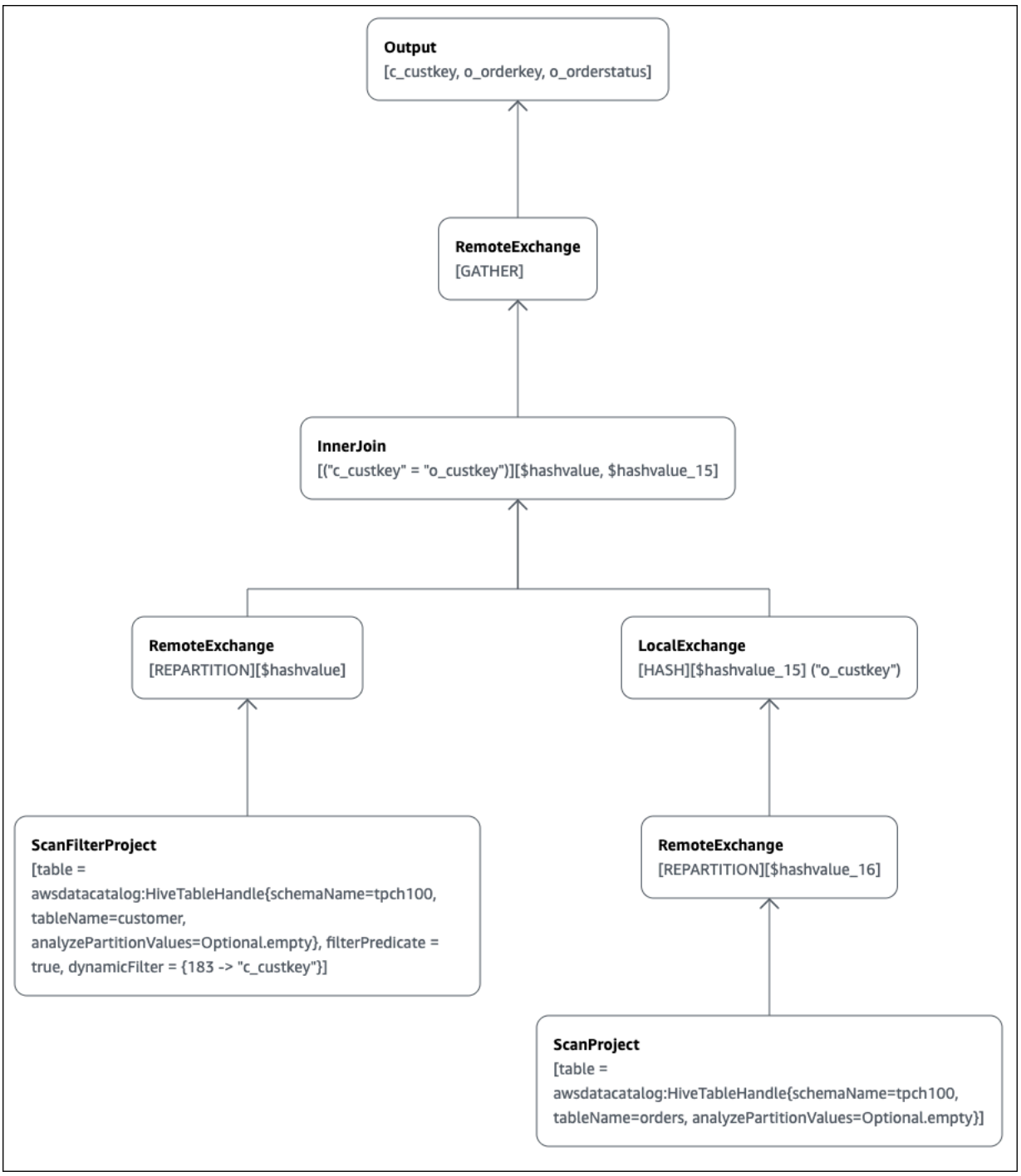
```
- Output[request_timestamp, elb_name, request_ip] => [[request_timestamp, elb_name,
 request_ip]]
    - RemoteExchange[GATHER] => [[request_timestamp, elb_name, request_ip]]
        - TableScan[awsdatacatalog:HiveTableHandle{schemaName=sampledb,
 tableName=elb_logs,
analyzePartitionValues=Optional.empty}] => [[request_timestamp, elb_name, request_ip]]
                LAYOUT: sampledb.elb_logs
                request_ip := request_ip:string:2:REGULAR
                request_timestamp := request_timestamp:string:0:REGULAR
                elb_name := elb_name:string:1:REGULAR
```

EXPLAIN 示例 2：绘制查询计划的图表

您可以使用 Athena 控制台为自己绘制查询计划图表。在 Athena 查询编辑器中输入类似以下内容的 SELECT 语句，然后选择 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
```

Athena 查询编辑器的 Explain（说明）页面将打开并显示查询的分布式计划和逻辑计划。下图显示了示例的逻辑计划。



Important

目前，某些分区筛选器可能在嵌套运算符树图表中不可见，即使 Athena 确实将其应用于您的查询。要验证此类筛选器的效果，请在您的查询中运行 `EXPLAIN` 或 `EXPLAIN ANALYZE` 并查看结果。

有关如何在 Athena 控制台中使用查询计划图表功能的更多信息，请参阅 查看 SQL 查询的执行计划。

EXPLAIN 示例 3：使用 EXPLAIN 语句验证分区修剪

在分区键上使用筛选谓词来查询分区表时，查询引擎会将谓词应用于分区键以减少读取的数据量。

下面的示例使用了 EXPLAIN 查询来验证分区表上 SELECT 查询的分区修剪。首先，一个 CREATE TABLE 语句将创建 tpch100.orders_partitioned 表。表将在 o_orderdate 列上进行分区。

```
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>/'
```

tpch100.orders_partitioned 表在 o_orderdate 上有几个分区，如 SHOW PARTITIONS 命令所示。

```
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
```

以下 EXPLAIN 查询验证指定 SELECT 语句的分区修剪。

```
EXPLAIN
SELECT
   o_orderkey,
   o_custkey,
   o_orderdate
FROM tpch100.orders_partitioned
WHERE o_orderdate = '1995'
```

结果

```
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
```

结果中的粗体文本显示谓词 o_orderdate = '1995' 被应用于 PARTITION_KEY。

EXPLAIN 示例 4：使用 EXPLAIN 查询检查联接顺序和联接类型

以下 EXPLAIN 查询检查 SELECT 语句的连接顺序和连接类型。使用这样的查询来检查查询内存使用情况，以便降低获取 EXCEEDED_LOCAL_MEMORY_LIMIT 错误消息的几率。

```
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
```

结果

```
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
```

示例查询经过优化为交叉连接以获得更好的性能。结果表明 tpch100.orders 将作为 BROADCAST 分配类型加以分配。这意味着 tpch100.orders 表将分发给执行联接操作的所有节点。BROADCAST 分布类型将要求 tpch100.orders 表的所有筛选结果适应执行连接操作的各节点内存。

然而，tpch100.customer 表较 tpch100.orders 更小。由于 tpch100.customer 需要的内存较少，因此可以将查询重写为 BROADCAST tpch100.customer 而不是 tpch100.orders。这会降低查询接收 EXCEEDED_LOCAL_MEMORY_LIMIT 错误消息的几率。此策略假定以下几点：

- tpch100.customer.c_custkey 是唯一的 tpch100.customer 表。
- 在 tpch100.customer 和 tpch100.orders 之间存在一对多映射关系。

以下示例显示了重写的查询：

```
SELECT
    c.c_custkey,
    o.o_orderkey,
    o.o_orderstatus
FROM tpch100.orders o
JOIN tpch100.customer c -- the filtered results of tpch100.customer are distributed to
 all nodes.
    ON c.c_custkey = o.o_custkey
WHERE c.c_custkey = 123
```

EXPLAIN 示例 5：使用 EXPLAIN 查询删除没有效果的谓词

您可以使用 EXPLAIN 查询来检查筛选谓词的有效性。您可以使用结果删除没有效果的谓词，如以下示例所示。

```
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
```

结果

```
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
```

结果中的 `filterPredicate` 显示优化程序将原来的三个谓词合并为两个谓词，并更改了它们的应用顺序。

```
filterPredicate = (("c_custkey" = 1500) AND ("c_custkey" = CAST(("random"() * 1E3) AS
 int)))
```

因为结果表明，谓词 `AND c.c_custkey BETWEEN 1000 AND 2000` 不起作用，所以可在不更改查询结果的情况下删除此谓词。

有关在 `EXPLAIN` 查询结果中使用的术语信息，请参阅 [了解 Athena EXPLAIN 语句结果](#)。

EXPLAIN ANALYZE 示例

以下示例演示示例 `EXPLAIN ANALYZE` 查询和输出。

EXPLAIN ANALYZE 示例 1：使用 EXPLAIN ANALYZE 以文本格式显示查询计划和计算成本

在以下示例中，`EXPLAIN ANALYZE` 演示了对 CloudFront 日志执行的 `SELECT` 查询的执行计划和计算成本。格式默认为文本输出。

```
EXPLAIN ANALYZE SELECT FROM cloudfront_logs LIMIT 10
```

结果

```
 Fragment 1
     CPU: 24.60ms, Input: 10 rows (1.48kB); per task: std.dev.: 0.00, Output: 10 rows
 (1.48kB)
     Output layout: [date, time, location, bytes, requestip, method, host, uri, status,
 referrer,\
       os, browser, browserversion]
Limit[10] => [[date, time, location, bytes, requestip, method, host, uri, status,
 referrer, os,\
  browser, browserversion]]
             CPU: 1.00ms (0.03%), Output: 10 rows (1.48kB)
             Input avg.: 10.00 rows, Input std.dev.: 0.00%
LocalExchange[SINGLE] () => [[date, time, location, bytes, requestip, method, host,
 uri, status, referrer, os,\
 browser, browserversion]]
```

```
                CPU: 0.00ns (0.00%), Output: 10 rows (1.48kB)
                Input avg.: 0.63 rows, Input std.dev.: 387.30%
RemoteSource[2] => [[date, time, location, bytes, requestip, method, host, uri, status,
 referrer, os,\
  browser, browserversion]]
                    CPU: 1.00ms (0.03%), Output: 10 rows (1.48kB)
                    Input avg.: 0.63 rows, Input std.dev.: 387.30%

 Fragment 2
     CPU: 3.83s, Input: 998 rows (147.21kB); per task: std.dev.: 0.00, Output: 20 rows
 (2.95kB)
     Output layout: [date, time, location, bytes, requestip, method, host, uri, status,
 referrer, os,\
       browser, browserversion]
LimitPartial[10] => [[date, time, location, bytes, requestip, method, host, uri,
 status, referrer, os,\
  browser, browserversion]]
            CPU: 5.00ms (0.13%), Output: 20 rows (2.95kB)
            Input avg.: 166.33 rows, Input std.dev.: 141.42%
TableScan[awsdatacatalog:HiveTableHandle{schemaName=default, tableName=cloudfront_logs,
\
  analyzePartitionValues=Optional.empty},
grouped = false] => [[date, time, location, bytes, requestip, method, host, uri, st
                CPU: 3.82s (99.82%), Output: 998 rows (147.21kB)
                Input avg.: 166.33 rows, Input std.dev.: 141.42%
                LAYOUT: default.cloudfront_logs
                date := date:date:0:REGULAR
                referrer := referrer:string:9:REGULAR
                os := os:string:10:REGULAR
                method := method:string:5:REGULAR
                bytes := bytes:int:3:REGULAR
                browser := browser:string:11:REGULAR
                host := host:string:6:REGULAR
                requestip := requestip:string:4:REGULAR
                location := location:string:2:REGULAR
                time := time:string:1:REGULAR
                uri := uri:string:7:REGULAR
                browserversion := browserversion:string:12:REGULAR
                status := status:int:8:REGULAR
```

EXPLAIN ANALYZE 示例 2：使用 EXPLAIN ANALYZE 以 JSON 格式显示查询计划

以下示例演示了对 CloudFront 日志执行的 SELECT 查询的执行计划和计算成本。该示例指定 JSON 作为输出格式。

```
EXPLAIN ANALYZE (FORMAT JSON) SELECT * FROM cloudfront_logs LIMIT 10
```

结果

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

其他 资源

有关更多信息，请参阅以下资源。

- 了解 Athena EXPLAIN 语句结果
- 查看 SQL 查询的执行计划
- 查看已完成查询的统计数据和执行详细信息
- Trino EXPLAIN 文档
- Trino EXPLAIN ANALYZE 文档
- AWS 大数据博客中的使用 Amazon Athena 中的 EXPLAIN 和 EXPLAIN ANALYZE 优化联合查询性能。

了解 Athena EXPLAIN 语句结果

本主题提供了 Athena EXPLAIN 语句结果中使用的操作术语的简要指南。

EXPLAIN 语句输出类型

EXPLAIN 语句输出可以是以下两种类型之一：

- 逻辑计划 – 显示 SQL 引擎用于执行语句的逻辑计划。该选项的语法为 EXPLAIN 或者 EXPLAIN (TYPE LOGICAL)。

- 分布式计划 – 显示分布式环境中的执行计划。输出显示片段，这些片段是处理阶段。每个计划片段由一个或多个节点进行处理。数据可以在处理片段的节点之间进行交换。该选项的语法为 EXPLAIN (TYPE DISTRIBUTED)。

  在分布式计划的输出中，片段（处理阶段）由 Fragment *number* [*fragment_type*] 表示，其中 *number* 是一个从零开始的整数，*fragment_type* 指定节点如何执行片段。下表介绍了片段类型，它们提供了解数据交换布局的洞察。

  分布式计划片段类型

片段类型	描述
SINGLE	片段在单个节点上执行。
HASH	片段在固定数量的节点上执行。输入数据使用散列函数进行分配。
ROUND_ROBIN	片段在固定数量的节点上执行。输入数据以轮询方式进行分配。
BROADCAST	片段在固定数量的节点上执行。输入数据将广播到所有节点。
SOURCE	片段在访问输入拆分的节点上执行。

Exchange

与交换相关的术语描述了如何在工件节点之间交换数据。传输可以是本地传输或远程传输。

LocalExchange [***exchange_type***]

针对查询的不同阶段，在工件节点内本地传输数据。*exchange_type* 的值可以是逻辑交换或分布式交换类型之一，如本部分后文所述。

RemoteExchange [***exchange_type***]

为查询的不同阶段在工件节点之间传输数据。*exchange_type* 的值可以是逻辑交换或分布式交换类型之一，如本部分后文所述。

逻辑交换类型

以下交换类型描述了在逻辑计划的交换阶段执行的操作。

- **GATHER** – 单个工件节点从所有其他工件节点收集输出。例如，特定查询的最后阶段从所有节点收集结果，并将结果写入 Amazon S3。
- **REPARTITION**– 根据应用于下一个运算符所需的分区方案，将行数据发送到特定的工件。
- **REPLICATE** – 将行数据复制到所有工件。

分布式交换类型

以下交换类型指示数据在分布式计划中的节点之间交换时的布局。

- **HASH** – 交换使用哈希函数将数据分发到多个目标。
- **SINGLE** – 交换将数据分发到单个目标。

扫描

以下术语描述了在查询过程中扫描数据的方式。

TableScan

从 Amazon S3 或 Apache Hive 连接器扫描表的源数据，并应用从筛选条件谓词生成的分区修剪。

ScanFilter

从 Amazon S3 或 Hive 连接器扫描表的源数据，并应用从筛选条件谓词和未通过分区修剪应用的其他筛选条件谓语生成的分区修剪。

ScanFilterProject

首先，从 Amazon S3 或 Hive 连接器扫描表的源数据，并应用从筛选条件谓词和未通过分区修剪应用的其他筛选条件谓词生成的分区修剪。然后，将输出数据的内存布局修改为新投影，以提高后续阶段的性能。

联接

在两个表之间连接数据。可以按连接类型和分配类型对连接进行分类。

联接类型

联接类型定义联接操作的发生方式。

CrossJoin – 生成连接的两个表的笛卡尔积。

InnerJoin – 选择两个表中具有匹配值的记录。

LeftJoin – 从左表中选择所有记录，从右表中选择匹配记录。如果没有匹配，则右侧的结果为 NULL。

RightJoin – 从右表中选择所有记录，从左表中选择匹配记录。如果没有匹配，则左侧的结果为 NULL。

FullJoin – 选择左表或右表记录中存在匹配项的所有记录。连接的表包含两侧表中的所有记录，以及任一侧缺少匹配项填充的空值。

> **Note**
>
> 出于性能考虑，查询引擎可以将连接查询重写为不同的连接类型以生成相同的结果。例如，在一个表上具有谓词的内部连接查询可以被重写为 `CrossJoin`。这会将谓词下推到表的扫描阶段，以减少扫描的数据量。

联接分配类型

分配类型定义执行连接操作时，如何在工件节点之间交换数据。

已分区 – 左表和右表在所有工件节点之间进行哈希分区。已分区分配在每个节点中消耗的内存较少。已分区分配可能比复制的连接慢得多。当您连接两个大型表时，适合使用已分区连接。

已复制 – 一个表在所有工件节点之间进行哈希分区，另一个表被复制到所有工件节点以执行连接操作。复制的分配比已分区的连接快得多，但它在每个工件节点中将占用更多的内存。如果复制的表太大，则工件节点可能会遇到内存不足的错误。复制的连接适用于其中一个连接表较小的情况。

PREPARE

创建名为 `statement_name` 的 SQL 语句以在稍后运行。该语句可以包含由问号表示的参数。要为参数提供值并运行预准备语句，请使用 EXECUTE。

摘要

```
PREPARE statement_name FROM statement
```

下表介绍了这些参数。

参数	描述
`statement_name`	要执行的预准备语句的名称。此名称在工作组范围内必须唯一。

参数	描述
statement	SELECT、CTAS 或 INSERT INTO 查询。

> **Note**
>
> 工作组中的最大预处理语句数为 1000。

示例

以下示例准备了不带参数的选择查询。

```
PREPARE my_select1 FROM
SELECT * FROM nation
```

以下示例准备了包含参数的选择查询。productid 和 quantity 的值将由 EXECUTE 语句的 USING 子句提供：

```
PREPARE my_select2 FROM
SELECT order FROM orders WHERE productid = ? and quantity < ?
```

以下示例准备了插入查询。

```
PREPARE my_insert FROM
INSERT INTO cities_usa (city, state)
SELECT city, state
FROM cities_world
WHERE country = ?
```

其他 资源

使用预准备语句进行查询

EXECUTE

DEALLOCATE PREPARE

INSERT INTO

EXECUTE

运行名称为 `statement_name` 的预准备语句。预准备语句中问号的参数值在逗号分隔列表的 `USING` 子句中定义。要创建预准备语句，请使用 PREPARE。

摘要

```
EXECUTE statement_name [ USING parameter1[, parameter2, ... ] ]
```

示例

以下示例准备并执行不带参数的查询。

```
PREPARE my_select1 FROM
SELECT name FROM nation
EXECUTE my_select1
```

以下示例准备并执行带单个参数的查询。

```
PREPARE my_select2 FROM
SELECT * FROM "my_database"."my_table" WHERE year = ?
EXECUTE my_select2 USING 2012
```

这等同于：

```
SELECT * FROM "my_database"."my_table" WHERE year = 2012
```

以下示例准备并执行带两个参数的查询。

```
PREPARE my_select3 FROM
SELECT order FROM orders WHERE productid = ? and quantity < ?
EXECUTE my_select3 USING 346078, 12
```

其他 资源

使用预准备语句进行查询

PREPARE

INSERT INTO

DEALLOCATE PREPARE

从当前工作组中的预准备语句中删除具有指定名称的预准备语句。

摘要

```
DEALLOCATE PREPARE statement_name
```

示例

以下示例从当前工作组中删除 my_select1 预准备语句。

```
DEALLOCATE PREPARE my_select1
```

其他 资源

使用预准备语句进行查询

PREPARE

UNLOAD

将查询结果从 SELECT 语句写为指定的数据格式。UNLOAD 支持的格式包括 Apache Parquet、ORC、Apache Avro 和 JSON。CSV 是 Athena SELECT 命令支持的唯一输出格式，但您可以使用支持多种输出格式的 UNLOAD 命令将 SELECT 查询括起来，并将其输出重写为 UNLOAD 支持的其中一种格式。

尽管您可以使用 CTAS 语句以 CSV 以外的格式输出数据，但这些语句还是需要在 Athena 中创建表。UNLOAD 语句在您想以非 CSV 的格式获取 SELECT 查询的输出结果，但不需要关联的表时很有用。例如，下游应用程序可能需要将 SELECT 查询的结果设置为 JSON 格式，而如果您打算使用 SELECT 查询的结果进行其他分析，则相较于 CSV，Parquet 或 ORC 可能会在性能上更有优势。

注意事项和限制

当您在 Athena 中使用 UNLOAD 语句时，请记住以下几点：

- 没有文件的全局排序 – UNLOAD 结果将并行写入多个文件。如果 UNLOAD 语句中的 SELECT 查询指定排序顺序，则每个文件的内容都将按顺序排序，但文件不相对于彼此排序。
- 孤立数据未删除 – 在出现故障的情况下，Athena 不会尝试删除孤立的数据。这种行为与 CTAS 和 INSERT INTO 语句相同。

- 最大分区 – UNLOAD 可以使用的最大分区数为 100。
- 元数据和清单文件 – Athena 为每个 UNLOAD 查询生成元数据文件和数据清单文件。清单跟踪查询写入的文件。这两个文件都会保存到 Amazon S3 中的 Athena 查询结果位置。有关更多信息，请参阅识别查询输出文件。
- 加密 – UNLOAD 输出文件将根据用于 Amazon S3 的加密配置进行加密。要设置加密配置以加密 UNLOAD 结果，则可以使用 EncryptionConfiguration API。
- 已准备好语句 – UNLOAD 可以与准备好的语句一起使用。有关 Athena 中准备语句的信息，请参阅使用参数化查询。
- 服务配额 – UNLOAD 使用 DML 查询配额。有关配额的信息，请参阅服务限额。
- 预期存储桶拥有者 – 预期存储桶拥有者设置不适用于在 UNLOAD 查询中指定的目标 Amazon S3 位置。预期存储桶拥有者设置仅适用于您为 Athena 查询结果指定的 Amazon S3 输出位置。有关更多信息，请参阅使用 Athena 控制台指定查询结果位置。

语法

UNLOAD 语句使用以下语法。

```
UNLOAD (SELECT col_name[, ...] FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/my_folder/'
WITH ( property_name = 'expression' [, ...] )
```

除了写入分区时，TO 目标必须在 Amazon S3 中指定一个没有数据的位置。在 UNLOAD 查询写入指定的位置前，它会验证存储桶位置是否为空。由于如果位置中已有数据，则 UNLOAD 不会向指定的位置写入数据，而 UNLOAD 不会覆盖现有数据。要重新使用存储桶位置作为 UNLOAD 的目标，请删除存储桶位置中的数据，然后再次运行查询。

请注意，当 UNLOAD 写入分区时，此行为会有所不同。如果多次运行具有相同 SELECT 语句、相同 TO 位置和相同分区的相同 UNLOAD 查询，则每个 UNLOAD 查询都会在指定的位置和分区将数据卸载到 Amazon S3 中。

参数

property_name 的可能值如下所示。

format = '***file_format***'

必需。指定输出的文件格式。*file_format* 可能的值为 ORC、PARQUET、AVRO、JSON 或 TEXTFILE。

compression = '*compression_format*'

可选。此选项特定于 ORC 和 Parquet 格式。对于 ORC，默认值为 `zlib`，对于 Parquet，默认为 `gzip`。有关支持的压缩格式的信息，请参阅 Athena 压缩支持。

> **Note**
>
> 此选项不适用于 `AVRO` 格式。Athena 将 `gzip` 用于 `JSON` 和 `TEXTFILE` 格式。

压缩级别 = *compression_level*

可选。要用于 ZSTD 压缩的压缩级别。此属性仅适用于 ZSTD 压缩。有关更多信息，请参阅 在 Athena 中使用 ZSTD 压缩级别。

field_delimiter = '*delimiter*'

可选。为 CSV、TSV 和其他文本格式文件指定单字符字段分隔符。下面的示例指定了逗号分隔符。

```
WITH (field_delimiter = ',')
```

目前，不支持多字符字段分隔符。如果您未指定字段分隔符，则会使用八进制字符 `\001` (^A)。

partitioned_by = ARRAY[*col_name*[,...]]

可选。输出进行分区所依据的列的数组列表。

> **Note**
>
> 在 `SELECT` 语句中，确保分区列的名称在列列表中最后列出。

示例

下面的示例将 `SELECT` 查询的输出写入了 Amazon S3 位置 `s3://DOC-EXAMPLE-BUCKET/unload_test_1/`，使用的是 JSON 格式。

```
UNLOAD (SELECT * FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/unload_test_1/'
```

```
WITH (format = 'JSON')
```

下面的示例使用 Snappy 压缩以 Parquet 格式写入了 SELECT 查询的输出。

```
UNLOAD (SELECT * FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/'
WITH (format = 'PARQUET',compression = 'SNAPPY')
```

以下示例以文本格式写入四列，按最后一列对输出进行分区。

```
UNLOAD (SELECT name1, address1, comment1, key1 FROM table1)
TO 's3://DOC-EXAMPLE-BUCKET/ partitioned/'
WITH (format = 'TEXTFILE', partitioned_by = ARRAY['key1'])
```

以下示例将使用 Parquet 文件格式、ZSTD 压缩和 ZSTD 压缩级别 4 将查询结果卸载到指定位置。

```
UNLOAD (SELECT * FROM old_table)
TO 's3://DOC-EXAMPLE-BUCKET/'
WITH (format = 'PARQUET', compression = 'ZSTD', compression_level = 4)
```

其他 资源

- AWS 大数据博客中的使用 Amazon Athena UNLOAD 功能简化您的 ETL 和 ML 管道。

Amazon Athena 中的函数

有关 Athena 引擎版本之间的函数变化，请参阅 Athena 引擎版本参考。有关可与 AT TIME ZONE 运算符共同使用的时区列表，请参阅 支持的时区。

Athena 引擎版本 3

Athena 引擎版本 3 中的函数是基于 Trino。有关 Trino 函数、运算符和表达式的信息，请参阅 函数和运算符 以及 Trino 文档中的以下子章节。

- 聚合
- 数组
- 二进制
- 按位运算

- Color (颜色)
- Comparison（比较）
- 条件
- 转换
- 日期和时间
- 十进制
- 地理空间
- HyperLogLog
- IP 地址
- JSON
- Lambda
- 逻辑
- 机器学习
- Map
- 数学
- 分位数摘要
- 正则表达式
- 会话
- 设置摘要
- String
- 表
- Teradata
- T-摘要
- URL
- UUID
- 窗口

Athena 引擎版本 2

Athena 引擎版本 2 中的函数是基于 Presto 0.217。有关 Athena 引擎版本 2 中的地理空间函数的信息，请参阅 Athena 引擎版本 2 中的地理空间函数。

> **Note**
>
> Presto 0.217 函数的特定版本文档不再可用。有关当前 Presto 函数、运算符和表达式的信息，请参阅 Presto 函数和运算符，或者访问本节中的子类别链接。

- 逻辑运算符
- 比较函数和运算符
- 条件表达式
- 转换函数
- 数学函数和运算符
- 按位函数
- 十进制函数和运算符
- 字符串函数和运算符
- 二进制函数
- 日期与时间函数和运算符
- 正则表达式函数
- JSON 函数和运算符
- URL 函数
- 聚合函数
- 窗口函数
- 颜色函数
- 数组函数和运算符
- 映射函数和运算符
- Lambda 表达式和函数
- Teradata 函数

支持的时区

您可以使用 `SELECT timestamp` 语句中的 `AT TIME ZONE` 运算符指定返回时间戳的时区，如以下示例所示：

```
SELECT timestamp '2012-10-31 01:00 UTC' AT TIME ZONE 'America/Los_Angeles' AS la_time;
```

结果

```
la_time

2012-10-30 18:00:00.000 America/Los_Angeles
```

下面的列表包含了可与 Athena 中 AT TIME ZONE 运算符共同使用的时区。有关其他时区相关函数和示例，请参阅 时区函数和示例。

```
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
```

时区函数和示例

以下是一些其他时区相关函数和示例。

- at_timezone (***###***、***##***) — 返回在相应##当地时间的###值。

  示例

  ```
  SELECT at_timezone(timestamp '2021-08-22 00:00 UTC', 'Canada/Newfoundland')
  ```

  结果

  ```
  2021-08-21 21:30:00.000 Canada/Newfoundland
  ```

- timezone_hour (***###***) — 返回时区与时间戳偏移的小时数作为 `bigint`。

  示例

```
SELECT timezone_hour(timestamp '2021-08-22 04:00 UTC' AT TIME ZONE 'Canada/
Newfoundland')
```

结果

```
-2
```

- timezone_minute（***###***）— 返回时区与###偏移的分钟数作为 `bigint`。

  示例

  ```
  SELECT timezone_minute(timestamp '2021-08-22 04:00 UTC' AT TIME ZONE 'Canada/
  Newfoundland')
  ```

  结果

  ```
  -30
  ```

- with_timezone（***###***、***##***）— 从指定的###和##值返回带时区的时间戳。

  示例

  ```
  SELECT with_timezone(timestamp '2021-08-22 04:00', 'Canada/Newfoundland')
  ```

  结果

  ```
  2021-08-22 04:00:00.000 Canada/Newfoundland
  ```

DDL 语句

直接在 Athena 中使用以下 DDL 语句。

Athena 查询引擎基于 HiveQL DDL。

Athena 并不支持所有 DDL 语句，并且 HiveQL DDL 和 Athena DDL 之间存在一些差异。有关更多信息，请参阅本部分中的参考主题和 不支持的 DDL。

主题

不支持的 DDL

Athena 不支持以下 DDL 语句：

- ALTER INDEX
- ALTER TABLE *table_name* ARCHIVE PARTITION
- ALTER TABLE *table_name* CLUSTERED BY
- ALTER TABLE *table_name* EXCHANGE PARTITION
- ALTER TABLE *table_name* NOT CLUSTERED
- ALTER TABLE *table_name* NOT SKEWED
- ALTER TABLE *table_name* NOT SORTED
- ALTER TABLE *table_name* NOT STORED AS DIRECTORIES
- ALTER TABLE *table_name* partitionSpec CHANGE COLUMNS
- ALTER TABLE *table_name* partitionSpec COMPACT
- ALTER TABLE *table_name* partitionSpec CONCATENATE
- ALTER TABLE *table_name* partitionSpec SET FILEFORMAT
- ALTER TABLE *table_name* SET SERDEPROPERTIES
- ALTER TABLE *table_name* SET SKEWED LOCATION
- ALTER TABLE *table_name* SKEWED BY
- ALTER TABLE *table_name* TOUCH
- ALTER TABLE *table_name* UNARCHIVE PARTITION
- COMMIT
- CREATE INDEX
- CREATE ROLE
- CREATE TABLE *table_name* LIKE *existing_table_name*
- CREATE TEMPORARY MACRO
- DELETE FROM
- DESCRIBE DATABASE
- DFS
- DROP INDEX
- DROP ROLE
- DROP TEMPORARY MACRO
- EXPORT TABLE
- GRANT ROLE

- IMPORT TABLE
- LOCK DATABASE
- LOCK TABLE
- REVOKE ROLE
- ROLLBACK
- SHOW COMPACTIONS
- SHOW CURRENT ROLES
- SHOW GRANT
- SHOW INDEXES
- SHOW LOCKS
- SHOW PRINCIPALS
- SHOW ROLE GRANT
- SHOW ROLES
- SHOW STATS
- SHOW TRANSACTIONS
- START TRANSACTION
- UNLOCK DATABASE
- UNLOCK TABLE

ALTER DATABASE SET DBPROPERTIES

为数据库创建一个或多个属性。DATABASE 和 SCHEMA 的使用是可互换的；它们具有相同的含义。

摘要

```
ALTER {DATABASE|SCHEMA} database_name
  SET DBPROPERTIES ('property_name'='property_value' [, ...] )
```

参数

SET DBPROPERTIES ('property_name'='property_value' [, ...]

为名为 property_name 的数据库指定一个或多个属性，并分别将每个属性的值设置为 property_value。如果 property_name 已存在，则会用 property_value 覆盖旧值。

示例

```
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

向现有表添加一个或多个列。使用可选 PARTITION 语法时，将更新分区元数据。

摘要

```
ALTER TABLE table_name
  [PARTITION
   (partition_col1_name = partition_col1_value
   [,partition_col2_name = partition_col2_value][,...])]
  ADD COLUMNS (col_name data_type)
```

参数

PARTITION (partition_col_name = partition_col_value [,...])

创建一个具有指定列名称/值组合的分区。仅当列的数据类型为字符串时，才将 partition_col_value 包含在引号中。

ADD COLUMNS (col_name data_type [,col_name data_type,...])

在现有列之后但在分区列之前添加列。

示例

```
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)
```

注意事项

- 要在运行 ALTER TABLE ADD COLUMNS 后在 Athena 查询编辑器导航窗格中查看新的表列，请手动刷新编辑器中的表列表，然后重新展开表。
- ALTER TABLE ADD COLUMNS 不适用于具有 date 数据类型的列。若要解决此问题，请使用 timestamp 数据类型。

ALTER TABLE ADD PARTITION

为表创建一个或多个分区列。每个分区包含一个或多个不同的列名称/值组合。将会为每个指定的组合创建一个单独的数据目录，这在某些情况下可提高查询性能。分区列在表数据本身中不存在，因此如果您使用的列名称与表本身的列同名，则会出现错误。有关更多信息，请参阅 在 Athena 中对数据进行分区。

在 Athena 中，一个表及其分区必须使用相同的数据格式，但它们的架构可以不同。有关更多信息，请参阅 包含分区的表中的更新。

有关 IAM 策略中需要的资源级权限（包括 glue:CreatePartition），请参阅 AWS Glue API 权限：操作和资源参考以及 针对 AWS Glue Data Catalog 中数据库和表的精细访问权限。有关使用 Athena 时权限相关的疑难解答信息，请参阅 在 Athena 中进行故障排除 主题的 权限 部分。

摘要

```
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']
  [,...]
```

参数

添加分区时，您可以为分区指定一个或多个列名/值对，以及该分区的数据文件所在的 Amazon S3 路径。

[IF NOT EXISTS]

如果已存在具有相同定义的分区，则会导致错误被隐藏。

PARTITION (partition_col_name = partition_col_value [,...])

创建一个具有指定列名称/值组合的分区。仅当列的数据类型为字符串时，才将 `partition_col_value` 包含在字符串字符中。

[LOCATION 'location']

指定存储前述语句所定义的分区的目录。当数据使用 Hive 样式分区 (pk1=v1/pk2=v2/pk3=v3) 时，LOCATION 子句为可选项。在 Hive 式分区中，完整的 Amazon S3 URI 是根据表的位置、分区键名称和分区键值自动构造的。有关更多信息，请参阅 在 Athena 中对数据进行分区。

注意事项

Amazon Athena 并未对可以在单个 `ALTER TABLE ADD PARTITION` DDL 语句中添加的分区数量施加具体限制。不过，若是需要添加大量分区，可考虑多分几次进行添加，避免性能存在潜在问题。以下示例使用连续命令来分别添加分区，同时使用 `IF NOT EXISTS` 来避免添加重复分区。

```
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')
```

在 Athena 中使用分区时，还请记住以下几点：

- 尽管 Athena 支持查询具有 1000 万个分区的 AWS Glue 表，但 Athena 的单次扫描读取量最多为 100 万个分区。
- 要优化查询并减少扫描的分区数量，可考虑分区修剪或使用分区索引等策略。
- 如果未使用 AWS Glue Data Catalog，则每个表的最大分区数为 20000。您可以请求提高限额。

有关在 Athena 中使用分区的其他注意事项，请参阅 在 Athena 中对数据进行分区。

示例

以下示例将单个分区添加至 Hive 样式分区数据表。

```
ALTER TABLE orders ADD
  PARTITION (dt = '2016-05-14', country = 'IN');
```

以下示例将多个分区添加至 Hive 样式分区数据表。

```
ALTER TABLE orders ADD
  PARTITION (dt = '2016-05-31', country = 'IN')
  PARTITION (dt = '2016-06-01', country = 'IN');
```

如果该表不适用于 Hive 样式分区数据，则 LOCATION 子句为必需项，并且应该是包含分区数据的前缀的完整 Amazon S3 URI。

```
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/';
```

要在已存在分区时忽略错误，使用 IF NOT EXISTS 子句，如以下示例所示。

```
ALTER TABLE orders ADD IF NOT EXISTS
  PARTITION (dt = '2016-05-14', country = 'IN');
```

零字节 _$folder$ 文件

如果您运行 ALTER TABLE ADD PARTITION 语句并错误地指定了已存在的分区和错误的 Simple Storage Service (Amazon S3) 位置，格式为 *partition_value*_$folder$ 的零字节占位符文件将在 Simple Storage Service (Amazon S3) 中创建。您必须手动移除这些文件。

为了防止这种情况发生，在 ALTER TABLE ADD PARTITION 语句中使用 ADD IF NOT EXISTS 语法，如以下示例所示。

```
ALTER TABLE table_name ADD IF NOT EXISTS PARTITION […]
```

ALTER TABLE DROP PARTITION

为命名的表删除一个或多个指定的分区。

摘要

```
ALTER TABLE table_name DROP [IF EXISTS] PARTITION (partition_spec) [, PARTITION
 (partition_spec)]
```

参数

[IF EXISTS]

如果指定的分区不存在，则会隐藏错误消息。

PARTITION (partition_spec)

每个 `partition_spec` 以形式 `partition_col_name = partition_col_value [,...]` 指定列名称/值组合。

示例

```
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');
```

注意事项

`ALTER TABLE DROP PARTITION` 语句不提供一次性删除所有分区的单一语法，也不支持用于指定要删除的分区范围的筛选条件。

作为解决方法，您可以在脚本中使用 AWS Glue API GetPartitions 和 BatchDeletePartition 操作。`GetPartitions` 操作支持复杂的筛选条件表达式，就像 SQL `WHERE` 表达式中的筛选条件表达式一样。使用 `GetPartitions` 创建要删除的分区的筛选列表后，您可以使用 `BatchDeletePartition` 操作批量删除 25 个分区。

Important

由于已知问题，如果为 `ALTER TABLE DROP PARTITION` 语句指定的分区无效，则将在 AWS Glue 中删除该表的所有分区。例如，以下语句将删除表 *my_table* 的所有分区，即使指定分区不存在也是如此。要解决该问题，确保在运行 `ALTER TABLE DROP PARTITION` 语句之前正确输入分区信息。

```
ALTER TABLE my_table DROP IF EXISTS PARTITION(zzz='');
```

ALTER TABLE RENAME PARTITION

重命名分区值。

> **Note**
>
> ALTER TABLE RENAME PARTITION 不重命名分区列。要更改分区列名称，您可以使用 AWS Glue 控制台。有关更多信息，请参阅本文后面的在 AWS Glue 中重命名分区列。

摘要

对于名为 `table_name` 的表，将 `partition_spec` 指定的分区值重命名为 `new_partition_spec` 指定的值。

```
ALTER TABLE table_name PARTITION (partition_spec) RENAME TO PARTITION
 (new_partition_spec)
```

参数

PARTITION (partition_spec)

每个 `partition_spec` 以形式 `partition_col_name = partition_col_value [,...]` 指定列名称/值组合。

示例

```
ALTER TABLE orders
PARTITION (dt = '2014-05-14', country = 'IN') RENAME TO PARTITION (dt = '2014-05-15',
 country = 'IN');
```

在 AWS Glue 中重命名分区列

使用以下过程在 AWS Glue 控制台中重命名分区列名。

在 AWS Glue 控制台中重命名表分区列

1. 登录 AWS Management Console，然后打开 AWS Glue 控制台，网址为：https://console.aws.amazon.com/glue/。
2. 在导航窗格中，选择表。

3. 在表页面上，使用筛选表搜索框查找要更改的表。
4. 在名称列中，选择要更改的表的链接。
5. 在表的详细信息页上的架构部分中，执行以下操作之一：
   - 要以 JSON 格式更改名称，请选择将架构编辑为 JSON。
   - 要直接更改名称，请选择编辑架构。此过程选择编辑架构。
6. 选中要重命名的分区列的复选框，然后选择编辑。
7. 在编辑架构条目对话框中，在名称中输入分区列的新名称。
8. 选择另存为新表版本。此操作会更新分区列名并保留架构演变历史记录，而不创建数据的单独物理副本。
9. 要比较表版本，请在表的详细信息页面上，选择操作，然后选择比较版本。

其他 资源

有关分区的更多信息，请参阅 在 Athena 中对数据进行分区。

ALTER TABLE REPLACE COLUMNS

从使用 LazySimpleSerDe 创建的表中移除所有现有列并用指定的列集替换它们。使用可选 PARTITION 语法时，将更新分区元数据。您还可以通过仅指定要保留的列，使用 ALTER TABLE REPLACE COLUMNS 来删除列。

摘要

```
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, ...])
```

参数

PARTITION (partition_col_name = partition_col_value [,...])

指定一个具有指定列名称/值组合的分区。仅当列的数据类型为字符串时，才将 partition_col_value 包含在引号中。

REPLACE COLUMNS (col_name data_type [,col_name data_type,...])

用指定的列名和数据类型替换现有列。

注意事项

- 要在运行 ALTER TABLE REPLACE COLUMNS 后在 Athena 查询编辑器导航窗格中查看表列的更改，您可以手动刷新编辑器中的表列表，然后重新展开表。
- ALTER TABLE REPLACE COLUMNS 不适用于具有 date 数据类型的列。若要解决此问题，请使用表中的 timestamp 数据类型。
- 请注意，即使您只替换单个列，语法也必须是 ALTER TABLE *table-name* REPLACE COLUMNS，columns（列）是复数形式。您不仅必须指定要替换的列，还必须指定要保留的列，否则会删除未指定的列。这种语法和行为源自 Apache Hive DDL。有关参考，请参阅 Apache 文档中的添加/替换列。

示例

在以下示例中，使用 LazySimpleSerDe 创建的表 names_cities 具有三个名为 col1、col2 和 col3 的列。所有列均为 string 类型。要显示表中的列，以下命令应使用 SHOW COLUMNS 语句。

```
SHOW COLUMNS IN names_cities
```

查询结果：

```
col1
col2
col3
```

以下 ALTER TABLE REPLACE COLUMNS 命令将列名替换为 first_name、last_name 和 city。底层源数据不受影响。

```
ALTER TABLE names_cities
REPLACE COLUMNS (first_name string, last_name string, city string)
```

要测试结果，请再次运行 SHOW COLUMNS。

```
SHOW COLUMNS IN names_cities
```

查询结果：

```
first_name
```

```
last_name
city
```

显示新列名的另一种方法是在 Athena 查询编辑器中预览表或运行您自己的 SELECT 查询。

ALTER TABLE SET LOCATION

更改名为 table_name 的表的位置，以及 (可选) 包含 partition_spec 的分区。

摘要

```
ALTER TABLE table_name [ PARTITION (partition_spec) ] SET LOCATION 'new location'
```

参数

PARTITION (partition_spec)

使用您要更改位置的参数 partition_spec 指定分区。partition_spec 以形式 partition_col_name = partition_col_value 指定列名称/值组合。

SET LOCATION '新位置'

指定新的位置，且必须是 Amazon S3 位置。有关语法的信息，请参阅 Amazon S3 中的表位置。

示例

```
ALTER TABLE customers PARTITION (zip='98040', state='WA') SET LOCATION 's3://DOC-
EXAMPLE-BUCKET/custdata/';
```

ALTER TABLE SET TBLPROPERTIES

向表中添加自定义或预定义元数据属性并设置其分配的值。要查看表中的属性，请使用 SHOW TBLPROPERTIES 命令。

Apache Hive 托管表不受支持，因此设置 'EXTERNAL'='FALSE' 没有效果。

摘要

```
ALTER TABLE table_name SET TBLPROPERTIES ('property_name' = 'property_value' [ , ... ])
```

参数

SET TBLPROPERTIES ('property_name' = 'property_value' [, ...])

将要添加的元数据属性指定为 `property_name`，将每个属性的值指定为 `property value`。如果 `property_name` 已存在，其值将设置为新指定的 `property_value`。

以下预定义表属性具有特殊用途。

预定义属性	描述
`classification`	为 AWS Glue 指示数据类型。可能的值为 `csv`、`parquet`、`orc`、`avro` 或 `json`。在 CloudTrail 控制台中为 Athena 创建的表将 `cloudtrail` 添加为 `classification` 属性的值。有关更多信息，请参阅 CREATE TABLE 的 TBLPROPERTIES 部分。
`has_encrypted_data`	指示由 `LOCATION` 指定的数据集是否已加密。有关更多信息，请参阅 CREATE TABLE 和 根据 Amazon S3 中的加密数据集创建表 的 TBLPROPERTIES 部分。
`orc.compress`	指定 ORC 格式的数据压缩格式。有关更多信息，请参阅 ORC SerDe。
`parquet.compression`	指定 Parquet 格式的数据压缩格式。有关更多信息，请参阅 Parquet SerDe。
`write.compression`	指定文本文件或 JSON 格式的数据压缩格式。对于 Parquet 和 ORC 格式，请分别使用 `parquet.compression` 和 `orc.compress` 属性。
`compression_level`	指定要使用的压缩级别。此属性仅适用于 ZSTD 压缩。可能的值介于 1 和 22 之间。默认值为 3。有关更多信息，请参阅 在 Athena 中使用 ZSTD 压缩级别。
`projection.*`	分区投影中使用的这些自定义属性可让 Athena 了解在对表运行查询时应使用哪种分区模式。有关更多信息，请参阅 使用 Amazon Athena 分区投影。
`skip.header.line.count`	定义表时忽略数据中的表头。有关更多信息，请参阅 忽略标题。

预定义属性	描述
storage.location.template	指定投影分区的自定义 Amazon S3 路径模板。有关更多信息，请参阅 设置分区投影。

示例

以下示例将为表属性添加一条注释。

```
ALTER TABLE orders
SET TBLPROPERTIES ('notes'="Please don't drop this table.");
```

以下示例将修改表 existing_table 以使用 Parquet 文件格式以及 ZSTD 压缩和 ZSTD 压缩级别 4。

```
ALTER TABLE existing_table
SET TBLPROPERTIES ('parquet.compression' = 'ZSTD', 'compression_level' = 4)
```

CREATE DATABASE

创建一个数据库。DATABASE 和 SCHEMA 的使用是可互换的。它们具有相同的含义。

> **Note**
>
> 有关在 Athena 中创建数据库、创建表和在表上运行 SELECT 查询的示例，请参阅 开始使用。

摘要

```
CREATE {DATABASE|SCHEMA} [IF NOT EXISTS] database_name
  [COMMENT 'database_comment']
  [LOCATION 'S3_loc']
  [WITH DBPROPERTIES ('property_name' = 'property_value') [, ...]]
```

参数

[IF NOT EXISTS]

如果已存在名为 database_name 的数据库，则会导致错误被隐藏。

[COMMENT database_comment]

为名为 comment 的内置元数据属性以及您为 database_comment 提供的值建立元数据值。在 AWS Glue 中，COMMENT 内容会写入数据库属性中的 Description 字段。

[LOCATION S3_loc]

将数据库文件和元存储将要存在的位置指定为 S3_loc。该位置必须是 Amazon S3 位置。

[WITH DBPROPERTIES ('property_name' = 'property_value') [, ...]]

允许您为数据库定义指定自定义元数据属性。

示例

```
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');
```

查看数据库属性

要查看您在 AWSDataCatalog 中使用 CREATE DATABASE 创建的数据库的属性，您可以使用 AWS CLI 命令 aws glue get-database，如以下示例所示：

```
aws glue get-database --name <your-database-name>
```

在 JSON 中，结果如下所示：

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

有关 AWS CLI 的更多信息，请参阅 AWS Command Line Interface 用户指南。

CREATE TABLE

使用您指定的名称和参数创建表。

> **Note**
>
> 此页面包含摘要参考信息。有关在 Athena 中创建表的更多信息和示例 CREATE TABLE 语句，请参阅在 Athena 中创建表。有关在 Athena 中创建数据库、创建表和在表上运行 SELECT 查询的示例，请参见 开始使用。

摘要

```
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 [, ...] ) ]
```

参数

EXTERNAL

指定表基于存储在 Amazon S3 中您所指定的 LOCATION 中的底层数据文件。除非是创建 Iceberg 表，否则请始终使用 EXTERNAL 关键字。如果您将没有 EXTERNAL 关键字的 CREATE TABLE 用于非 Iceberg 表，Athena 会发出错误。当您创建外部表时，引用的数据必须符合默认格式或您使用 ROW FORMAT、STORED AS 和 WITH SERDEPROPERTIES 子句指定的格式。

[IF NOT EXISTS]

此参数检查是否已存在名称相同的表。如果是，则参数返回 TRUE，然后 Amazon Athena 取消 CREATE TABLE 操作。由于取消发生在 Athena 调用数据目录之前，因此不会发出 AWS CloudTrail 事件。

[db_name.]table_name

指定要创建的表的名称。可选的 db_name 参数指定表所在的数据库。如果省略，则会采用当前数据库。如果表名称包含数字，请用引号将 table_name 引起来，例如 "table123"。如果 table_name 以下划线开头，请使用反引号，例如 `_mytable`。不支持特殊字符 (下划线除外)。

Athena 表名称不区分大小写；但是，如果您使用 Apache Spark，则 Spark 需要小写表名称。

[(col_name data_type [COMMENT col_comment] [, ...])]

为要创建的每个列指定名称，以及列的数据类型。列名称不允许下划线 (_) 以外的特殊字符。如果 col_name 以下划线开头，请将列名称放入反引号内，例如 `_mycolumn`。

data_type 值可能为以下任一值：

- boolean – 值为 true 和 false。
- tinyint – 一个 8 位有符号的整数，采用二进制补码格式，最小值为 -2^7，最大值为 2^7-1。
- smallint – 一个 16 位有符号的整数，采用二进制补码格式，最小值为 -2^15，最大值为 2^15-1。
- int – 在数据定义语言 (DDL) 查询（如 CREATE TABLE）中，使用 int 关键字来表示整数。在其他查询中，使用关键字 integer，其中 integer 以二进制补码格式表示为 32 位有符号值，最小值为 -2^31，最大值为 2^31-1。在 JDBC 驱动程序中，将返回 integer 以确保与业务分析应用程序兼容。
- bigint – 一个 64 位有符号的整数，采用二进制补码格式，最小值为 -2^63，最大值为 2^63-1。

- double – 64 位有符号的双精度浮点数。范围为 4.94065645841246544e-324d 至 1.79769313486231570e+308d，正或负。double 遵循 IEEE 浮点算法标准 (IEEE 754)。
- float – 32 位有符号的单精度浮点数。范围为 1.40129846432481707e-45 至 3.40282346638528860e+38，正或负。float 遵循 IEEE 浮点算法标准 (IEEE 754)。相当于 Presto 中的 real。在 Athena 中，在 CREATE TABLE 等 DDL 语句中使用 float，在 SELECT CAST 等 SQL 函数中使用 real。AWS Glue 爬网程序以 float 返回值, Athena 将内部翻译 real 和 float 类型（请参阅 2018 年 6 月 5 日 发布说明）。
- decimal [(*precision*, *scale*)]，其中 *precision* 是总位数，而 *scale*（可选）是小数部分的位数，默认值为 0。例如，使用以下类型定义：decimal(11,5)、decimal(15)。最大##值为 38，最大##值为 38。

  要将十进制值指定为文字（例如在查询 DDL 表达式中选择具有特定十进制值的行时），请指定 decimal 类型定义，并在查询中将十进制值列为文字（带单引号），如下例所示：decimal_value = decimal '0.12'。
- char – 固定长度字符数据，具有介于 1 和 255 之间的指定长度，例如 char(10)。有关更多信息，请参阅 CHAR Hive 数据类型。
- varchar – 可变长度字符数据，具有介于 1 和 65535 之间的指定长度，例如 varchar(10)。有关更多信息，请参阅 VARCHAR Hive 数据类型。
- string – 用单引号或双引号括起的字符串文本。

  > **Note**
  >
  > 在 Athena 中，无法将非字符串数据类型强制转换为 string；而是将这些数据类型强制转换为 varchar。

- binary –（适用于 Parquet 中的数据）
- date – ISO 格式的日期，例如 *YYYY*-*MM*-*DD*。例如，date '2008-09-15'。OpenCSVSerDe 是一个例外，它使用自 1970 年 1 月 1 日以来经过的天数。有关更多信息，请参阅 用于处理 CSV 的 OpenCSVSerDe。
- timestamp – 使用 java.sql.Timestamp 兼容格式的瞬间日期和时间最多可达毫秒级的最大分辨率，例如 *yyyy*-*MM*-*dd* *HH*:*mm*:*ss*[.f*...*]。例如，timestamp '2008-09-15 03:04:05.324'。OpenCSVSerDe 是一个例外，它使用UNIX 数字格式的 TIMESTAMP 数据（例如 1579059880000）。有关更多信息，请参阅 用于处理 CSV 的 OpenCSVSerDe。
- array < data_type >
- map < primitive_type, data_type >

- `struct<` col_name : data_type [comment col_comment] [, ...]>

[COMMENT table_comment]

创建 `comment` 表属性并用您指定的 `table_comment` 填充它。

[PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]

使用指定了 `col_name`、`data_type` 和 `col_comment` 的一个或多个分区列创建分区表。一个表可以有一个或多个分区，这些分区由多个不同的列名称和值组合组成。将会为每个指定的组合创建一个单独的数据目录，这在某些情况下可提高查询性能。已分区列在表数据本身中不存在。如果您将与表列相同的某个值用于 `col_name`，则会产生错误。有关更多信息，请参阅分区数据。

> **Note**
>
> 在您创建具有分区的表后，请运行后续查询，其中包含 MSCK REPAIR TABLE 子句用以刷新分区元数据，例如，`MSCK REPAIR TABLE cloudfront_logs;`。对于不兼容 Hive 的分区，请使用 ALTER TABLE ADD PARTITION 加载分区，以便您可以查询数据。

[CLUSTERED BY (col_name, col_name, ...) INTO num_buckets BUCKETS]

在使用分区或不使用分区的情况下，将指定 `col_name` 列中的数据分成名为存储桶的数据子集。`num_buckets` 参数指定要创建的存储桶数量。分桶可以提高对大型数据集的某些查询的性能。

[ROW FORMAT row_format]

指定表及其底层源数据 (如果适用) 的行格式。对于 `row_format`，您可以使用 `DELIMITED` 子句指定一个或多个分隔符，或者如下所述使用 `SERDE` 子句。如果省略 `ROW FORMAT` 或指定 `ROW FORMAT DELIMITED`，则会使用本机 SerDe。

- [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]

  仅当使用 Hive 0.13 且 STORED AS 文件格式为 `TEXTFILE` 时才可用。

--OR--

- SERDE 'serde_name' [WITH SERDEPROPERTIES ("property_name" = "property_value", "property_name" = "property_value" [, ...])]

  `serde_name` 指示要使用的 SerDe。`WITH SERDEPROPERTIES` 子句能让您提供 SerDe 允许的一个或多个自定义属性。

[STORED AS file_format]

为表数据指定文件格式。如果省略，则 `TEXTFILE` 是默认值。`file_format` 的选项是：

- SEQUENCEFILE
- TEXTFILE
- RCFILE
- ORC
- PARQUET
- AVRO
- ION
- INPUTFORMAT input_format_classname OUTPUTFORMAT output_format_classname

[LOCATION 's3://DOC-EXAMPLE-BUCKET/[folder]/']

指定从中创建表的底层数据在 Amazon S3 中的位置。位置路径必须是存储桶名称或存储桶名称以及一个或多个文件夹。如果您使用分区，请指定分区数据的根目录。有关表位置的更多信息，请参阅Amazon S3 中的表位置。有关数据格式和权限的信息，请参阅 对 Athena 中的数据和 Simple Storage Service (Amazon S3) 中的表的要求。

为您的文件夹或存储桶使用尾部斜杠。请勿使用文件名或 glob 字符。

使用:

```
s3://DOC-EXAMPLE-BUCKET/
```

```
s3://DOC-EXAMPLE-BUCKET/folder/
```

```
s3://DOC-EXAMPLE-BUCKET/folder/anotherfolder/
```

请勿使用：

```
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 [, ...])]

除预定义的表属性外，还指定表定义的自定义元数据键-值对，例如 "comment"。

has_encrypted_data – Athena 有一个内置属性 has_encrypted_data。将此属性设置为 true 以指示 LOCATION 指定的底层数据集是加密的。如果省略，并且如果工作组的设置不覆盖客户端设置，则假设 false。如果加密了底层数据时将它省略或设置为 false，则查询会导致错误。有关更多信息，请参阅 静态加密。

classification – 在 CloudTrail 控制台中为 Athena 创建的表将 cloudtrail 添加为 classification 属性的值。为运行 ETL 任务，AWS Glue 要求您创建一个具有 classification 属性的表来将 AWS Glue 的数据类型指示为 csv、parquet、orc、avro 或 json。例如，'classification'='csv'。如果您不指定此属性，则 ETL 任务将会失败。随后，您可以使用 AWS Glue 控制台、API 或 CLI 来指定它。有关更多信息，请参阅 将 AWS Glue ETL 任务与 Athena 结合使用和《AWS Glue 开发人员指南》中的在 AWS Glue 中编写任务。

compression_level – compression_level 属性指定了要使用的压缩级别。此属性仅适用于 ZSTD 压缩。可能的值介于 1 和 22 之间。默认值为 3。有关更多信息，请参阅 在 Athena 中使用 ZSTD 压缩级别。

有关其他表属性的更多信息，请参阅 ALTER TABLE SET TBLPROPERTIES。

示例

以下示例 CREATE TABLE 语句会基于存储在 Amazon S3 中的制表符分隔的行星数据创建一个表。

```
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/'
```

请注意以下几点：

- ROW FORMAT DELIMITED 子句指示数据由特定字符分隔。
- FIELDS TERMINATED BY '\t' 子句指定 TSV 数据中的字段由制表符 ('\t') 分隔。
- STORED AS TEXTFILE 子句指示数据以纯文本文件的形式存储在 Amazon S3 中。

要查询数据，可以使用如下所示的简单 SELECT 语句：

```
SELECT * FROM planet_data
```

要使用该示例在 Athena 中创建您自己的 TSV 表，请将表和列名替换为您自己的表和列的名称和数据类型，并更新 LOCATION 子句以指向存储 TSV 文件的 Amazon S3 路径。

有关创建表的更多信息，请参阅 在 Athena 中创建表。

CREATE TABLE AS

创建新表，使用 SELECT 查询的结果填充该表。要创建空表，使用 CREATE TABLE。CREATE TABLE AS 将 CREATE TABLE DDL 语句与 SELECT DML 语句结合在一起，因此严格地说同时包含 DDL 和 DML。请注意，尽管 CREATE TABLE AS 在这里与其他 DDL 语句分组在一起，但出于服务限额目的，Athena 中的 CTAS 查询仍被视为 DML。有关 Athena 服务限额的信息，请参阅 服务限额。

> **Note**
>
> 对于 CTAS 语句，预期存储桶拥有者设置不适用于 Amazon S3 中的目标表位置。预期存储桶拥有者设置仅适用于您为 Athena 查询结果指定的 Amazon S3 输出位置。有关更多信息，请参阅 使用 Athena 控制台指定查询结果位置。

有关超出此参考主题范围的 CREATE TABLE AS 其他信息，请参阅 从查询结果创建表（CTAS）。

主题

- 示例

摘要

```
CREATE TABLE table_name
[ WITH ( property_name = expression [, ...] ) ]
AS query
[ WITH [ NO ] DATA ]
```

其中：

WITH (property_name = expression [, ...])

可选 CTAS 表属性的列表，其中一些属性特定于数据存储格式。请参阅 CTAS 表属性。

查询

用于创建新表的 SELECT 查询。

> **Important**
>
> 如果您计划创建包含分区的查询，请指定 SELECT 语句的列列表中最后一个分区列的名称。

[WITH [NO] DATA]

如果使用 `WITH NO DATA`，则将创建与原始表具有相同架构的新空表。

> **Note**
>
> 若要在查询结果输出中包含列标题，可以使用简单的 SELECT 查询而不是 CTAS 查询。您可以从查询结果位置检索结果，也可以使用 Athena 控制台直接下载结果。有关更多信息，请参阅使用查询结果、最近查询和输出文件。

CTAS 表属性

Athena 中的每个 CTAS 表具有可选的 CTAS 表属性列表，您可以使用 `WITH ( property_name = expression [, ...] )` 指定。有关使用这些参数的信息，请参阅 CTAS 查询的示例。

WITH (property_name = expression [, ...],)

table_type = ['HIVE', 'ICEBERG']

可选。默认为 HIVE。指定结果表的表类型

例如：

```
WITH (table_type ='ICEBERG')
```

external_location = [location]

> **Note**
>
> 由于 Iceberg 表不是外部表，因此该属性不适用于 Iceberg 表。要在 CTAS 语句中定义 Iceberg 表的根位置，请使用本节后面所述的 `location` 属性。

可选。Athena 在 Amazon S3 中保存 CTAS 查询的位置。

例如：

```
WITH (external_location ='s3://DOC-EXAMPLE-BUCKET/tables/parquet_table/')
```

Athena 不会对查询结果使用同一路径两次。如果您手动指定了位置，请确保您指定的 Amazon S3 位置没有任何数据。Athena 从不尝试删除数据。如果您要再次使用相同的位置，请手动删除数据，否则 CTAS 查询将失败。

如果您运行一个 CTAS 查询，且该查询在强制执行查询结果位置的工作组中指定 `external_location`，则查询会失败并出现错误消息。要查看为工作组指定的查询结果位置，请参阅工作组的详细信息。

如果您的工作组覆盖查询结果位置的客户端设置，则 Athena 会在以下位置创建表：

```
s3://DOC-EXAMPLE-BUCKET/tables/query-id/
```

如果您未使用 `external_location` 属性指定位置，并且工作组未覆盖客户端设置，则 Athena 将使用查询结果位置的客户端设置在以下位置创建表：

```
s3://DOC-EXAMPLE-BUCKET/Unsaved-or-query-name/year/month/date/tables/query-id/
```

is_external = [boolean]

可选。表示此表是否为外部表。默认值为 true。对于 Iceberg 表，必须将其设置为 false。

例如：

```
WITH (is_external = false)
```

location = [location]

对于 Iceberg 表是必需的。指定要根据查询结果创建的 Iceberg 表的根位置。

例如：

```
WITH (location ='s3://DOC-EXAMPLE-BUCKET/tables/iceberg_table/')
```

field_delimiter = [delimiter]

（可选）特定于基于文本的数据存储格式。CSV、TSV 和文本文件中文件的单字符字段分隔符。例如，WITH (field_delimiter = ',')。目前，CTAS 查询不支持多字符字段分隔符。如果您未指定字段分隔符，则默认使用 \001。

format = [storage_format]

CTAS 查询结果的存储格式，例如 ORC、PARQUET、AVRO、JSON、ION 或 TEXTFILE。对于 Iceberg 表，允许使用 ORC、PARQUET 和 AVRO 格式。如果忽略，则默认使用 PARQUET。此参数的名称 format 必须以小写列出，否则 CTAS 查询将失败。

例如：

```
WITH (format = 'PARQUET')
```

bucketed_by = ARRAY[column_name[,…], bucket_count = [int]]

> **Note**
>
> 该属性不适用于 Iceberg 表。对于 Iceberg 表，使用带存储桶转换的分区。

用于分桶存储数据的存储桶数组列表。如果省略，则 Athena 在此查询中不会分桶存储数据。

bucket_count = [int]

> **Note**
>
> 该属性不适用于 Iceberg 表。对于 Iceberg 表，使用带存储桶转换的分区。

用于分桶存储数据的存储桶编号。如果省略，Athena 不分桶存储数据。例如：

```
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**
>
> 该属性不适用于 Iceberg 表。要将分区转换用于 Iceberg 表，请使用本节后面所述的 `partitioning` 属性。

可选。CTAS 表进行分区所依据的列的数组列表。确认分区列的名称在 SELECT 语句的列列表中最后列出。

partitioning = ARRAY[partition_transform, ...]

可选。指定要创建的 Iceberg 表的分区。Iceberg 支持多种分区转换和分区演化。下表总结了分区转换。

转换	描述
`year(ts)`	为每年创建一个分区。分区值是 1970 年 1 月 1 日与 `ts` 之间以年为单位的整数差。

转换	描述
month(ts)	为每年的每个月创建一个分区。分区值是 1970 年 1 月 1 日与 ts 之间以月为单位的整数差。
day(ts)	为每年的每一天创建一个分区。分区值是 1970 年 1 月 1 日与 ts 之间以天为单位的整数差。
hour(ts)	为每年的每个小时创建一个分区。分区值是一个时间戳，分和秒设置为零。
bucket(x, nbuckets)	将数据哈希处理到指定数量的存储桶中。分区值是 x 的整数哈希值，介于 0 到 nbuckets - 1（含）之间。
truncate(s, nchars)	将分区值设为 s 的第一个 nchars 字符。

例如：

```
WITH (partitioning = ARRAY['month(order_date)',
                           'bucket(account_number, 10)',
                           'country']))
```

optimize_rewrite_min_data_file_size_bytes = [long]

可选。数据优化的特定配置。包含小于指定值的文件以进行优化。默认值为 write_target_data_file_size_bytes 的 0.75 倍。该属性仅适用于 Iceberg 表。有关更多信息，请参阅 优化 Iceberg 表。

例如：

```
WITH (optimize_rewrite_min_data_file_size_bytes = 402653184)
```

optimize_rewrite_max_data_file_size_bytes = [long]

可选。数据优化的特定配置。包含大于指定值的文件以进行优化。默认值为 write_target_data_file_size_bytes 的 1.8 倍。该属性仅适用于 Iceberg 表。有关更多信息，请参阅 优化 Iceberg 表。

例如：

```
WITH (optimize_rewrite_max_data_file_size_bytes = 966367641)
```

optimize_rewrite_data_file_threshold = [int]

可选。数据优化的特定配置。若需要优化的数据文件少于给定阈值，则不会重写这些文件。这将允许积累更多的数据文件以生成更接近目标大小的文件，并跳过不必要的计算以节省成本。默认值为 5。该属性仅适用于 Iceberg 表。有关更多信息，请参阅 优化 Iceberg 表。

例如：

```
WITH (optimize_rewrite_data_file_threshold = 5)
```

optimize_rewrite_delete_file_threshold = [int]

可选。数据优化的特定配置。如果与数据文件关联的删除文件少于阈值，则不会重写该数据文件。这将允许为每个数据文件累积更多的删除文件，以节省成本。默认值为 2。该属性仅适用于 Iceberg 表。有关更多信息，请参阅 优化 Iceberg 表。

例如：

```
WITH (optimize_rewrite_delete_file_threshold = 2)
```

vacuum_min_snapshots_to_keep = [int]

可选。对特定配置执行 vacuum 操作。要保留的最新快照的最小数量。默认 为 1。该属性仅适用于 Iceberg 表。有关更多信息，请参阅 VACUUM。

> **Note**
>
> vacuum_min_snapshots_to_keep 属性需要 Athena 引擎版本 3。

例如：

```
WITH (vacuum_min_snapshots_to_keep = 1)
```

vacuum_max_snapshot_age_seconds = [long]

可选。对特定配置执行 vacuum 操作。以秒为单位的时间段，表示要保留的快照期限。默认值为 432000（5 天）。该属性仅适用于 Iceberg 表。有关更多信息，请参阅 VACUUM。

> **Note**
>
> vacuum_max_snapshot_age_seconds 属性需要 Athena 引擎版本 3。

例如：

```
WITH (vacuum_max_snapshot_age_seconds = 432000)
```

write_compression = [compression_format]

用于允许指定压缩的任何存储格式的压缩类型。compression_format 值指定将数据写入表时要使用的压缩。您可以为 TEXTFILE、JSON、PARQUET 和 ORC 文件格式指定压缩。

例如，如果 format 属性指定 PARQUET 作为存储格式，则 write_compression 的值指定 Parquet 的压缩格式。在这种情况下，为 write_compression 指定一个值等同于为 parquet_compression 指定一个值。

同样，如果 format 属性指定 ORC 作为存储格式，则 write_compression 的值指定 ORC 的压缩格式。在这种情况下，为 write_compression 指定一个值等同于为 orc_compression 指定一个值。

不能在同一 CTAS 查询中指定多个压缩格式表属性。例如，您无法在同一个查询中同时指定 write_compression 和 parquet_compression。这同样适用于 write_compression 和 orc_compression。有关每种文件格式支持的压缩类型的信息，请参阅 Athena 压缩支持。

orc_compression = [compression_format]

将 ORC 数据写入表时用于 ORC 文件格式的压缩类型。例如，WITH (orc_compression = 'ZLIB')。ORC 文件内的数据块（ORC Postscript 除外）使用您指定的压缩进行压缩。如果省略，则默认情况下对 ORC 使用 ZLIB 压缩。

> **Note**
>
> 为保持一致性，我们建议您使用 write_compression 属性而不是 orc_compression。使用 format 属性将存储格式指定为 ORC，然后使用 write_compression 属性指定 ORC 将使用的压缩格式。

`parquet_compression = [compression_format]`

将 Parquet 数据写入表时用于 Parquet 文件格式的压缩类型。例如，`WITH (parquet_compression = 'SNAPPY')`。此压缩应用于 Parquet 文件中的列块。如果省略，则默认情况下对 Parquet 使用 GZIP 压缩。

> **Note**
>
> 为保持一致性，我们建议您使用 `write_compression` 属性而不是 `parquet_compression`。使用 `format` 属性将存储格式指定为 `PARQUET`，然后使用 `write_compression` 属性指定 PARQUET 将使用的压缩格式。

`compression_level = [compression_level]`

要使用的压缩级别。此属性仅适用于 ZSTD 压缩。可能的值介于 1 和 22 之间。默认值为 3。有关更多信息，请参阅 在 Athena 中使用 ZSTD 压缩级别。

示例

有关 CTAS 查询的示例，请参阅以下资源。

- CTAS 查询的示例
- 将 CTAS 和 INSERT INTO 用于 ETL 和数据分析
- 将 CTAS 语句与 Amazon Athena 结合使用以降低成本并提高性能
- 使用 CTAS 和 INSERT INTO 绕过 100 分区限制

CREATE VIEW

从指定的 `SELECT` 查询创建新视图。该视图是一个逻辑表，可以被未来查询所引用。视图不包含任何数据，也不写入数据。相反，当您每次通过另一个查询引用该视图时，该视图指定的查询都会运行。

> **Note**
>
> 本主题提供了摘要信息以供参考。有关在 Athena 中使用视图的更多详细信息，请参阅 使用视图。有关视图限制的信息，请参阅 视图限制。

摘要

```
CREATE [ OR REPLACE ] VIEW view_name AS query
```

可选的 OR REPLACE 子句允许您通过替换来更新现有视图。有关更多信息，请参阅 创建视图。

示例

要根据表 orders 创建视图 test，请使用类似如下的查询：

```
CREATE VIEW test AS
SELECT
orderkey,
orderstatus,
totalprice / 2 AS half
FROM orders;
```

要根据表 orders 创建视图 orders_by_date，请使用以下查询：

```
CREATE VIEW orders_by_date AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate;
```

要更新现有视图，请使用类似于以下内容的示例：

```
CREATE OR REPLACE VIEW test AS
SELECT orderkey, orderstatus, totalprice / 4 AS quarter
FROM orders;
```

另请参阅 SHOW COLUMNS、SHOW CREATE VIEW、DESCRIBE VIEW 和 DROP VIEW。

DESCRIBE

为指定表显示一个或多个列，包括分区列。此命令对于检查复杂列的属性非常有用。

摘要

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

> **Important**
>
> 此语句的语法为 `DESCRIBE table_name` 而非 `DESCRIBE TABLE table_name`。使用后一种语法会导致出现错误消息 FAILED: SemanticException [Error 10001]: Table not found table（失败：SemanticException [错误 10001]：找不到表）。

参数

[EXTENDED | FORMATTED]

确定输出的格式。省略这些参数会以表格格式显示列名称及其对应的数据类型，包括分区列。指定 FORMATTED 不仅会以表格格式显示列名和数据类型，还会显示详细的表格和存储信息。EXTENDED 会以表格格式显示列和数据类型信息，并以 Thrift 序列化形式显示表的详细元数据。此格式的可读性较差，因而主要用于调试。

[PARTITION partition_spec]

如果包括在内，则列出由 `partition_spec` 指定的分区的元数据，其中 `partition_spec` 的格式为 `(partition_column = partition_col_value, partition_column = partition_col_value, ...)`。

[col_name ([.field_name] | [.'$elem$'] | [.'key'] | [.'$value$'])*]

指定要检查的列和属性。您可以为结构的元素指定 `.field_name`，为数组元素指定 `'$elem$'`，为映射键指定 `'$key$'`，为映射值指定 `'$value$'`。您可以以递归方式指定它，以便进一步探索复杂的列。

示例

```
DESCRIBE orders
```

```
DESCRIBE FORMATTED mydatabase.mytable PARTITION (part_col = 100) columnA;
```

下面的查询和输出显示了来自 `impressions` 表的列和数据类型信息，该表基于 Amazon EMR 示例数据。

```
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
```

下面的示例查询和输出显示了使用 FORMATTED 选项时相同表的结果。

```
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
```

下面的示例查询和输出显示了使用 EXTENDED 选项时相同表的结果。表的详细信息会输出到一行中，但是为了便于阅读，此处设置了格式。

```
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

显示命名视图的列列表。这能让您检查复杂视图的属性。

摘要

```
DESCRIBE [db_name.]view_name
```

示例

```
DESCRIBE orders;
```

另请参阅 SHOW COLUMNS、SHOW CREATE VIEW、SHOW VIEWS 和 DROP VIEW。

DROP DATABASE

从目录中删除命名的数据库。如果数据库包含表，则必须在运行 DROP DATABASE 之前删除这些表或使用 CASCADE 子句。DATABASE 和 SCHEMA 的使用是可互换的。它们具有相同的含义。

摘要

```
DROP {DATABASE | SCHEMA} [IF EXISTS] database_name [RESTRICT | CASCADE]
```

参数

[IF EXISTS]

如果 database_name 不存在，则会导致错误被隐藏。

[RESTRICT|CASCADE]

确定 database_name 中的表在 DROP 操作过程中如何被看待。如果您指定 RESTRICT，则不会删除包含表的数据库。这是默认行为。指定 CASCADE 将会导致数据库及其所有表被删除。

示例

```
DROP DATABASE clickstreams;
```

```
DROP SCHEMA IF EXISTS clickstreams CASCADE;
```

> **Note**
>
> 当您尝试删除名称中包含特殊字符（例如 my-database）的数据库时，可能会收到一条错误消息。要解决此问题，请尝试用反引号（`）字符将数据库名称括起来。有关 Athena 中数据库命名的信息，请参阅 表、数据库和列的名称。

DROP TABLE

删除名为 table_name 的表的元数据表定义。删除某个外部表时，底层数据将保持不变。

摘要

```
DROP TABLE [IF EXISTS] table_name
```

参数

[IF EXISTS]

如果 table_name 不存在，则会导致错误被隐藏。

示例

```
DROP TABLE fulfilled_orders
```

```
DROP TABLE IF EXISTS fulfilled_orders
```

使用 Athena 控制台查询编辑器删除具有除下划线 (_) 以外的特殊字符的表时，请使用反引号，如以下示例所示。

```
DROP TABLE `my-athena-database-01.my-athena-table`
```

使用 JDBC 连接器删除具有特殊字符的表时，不需要反引号字符。

```
DROP TABLE my-athena-database-01.my-athena-table
```

DROP VIEW

删除现有视图。如果该视图不存在，可选 `IF EXISTS` 子句将抑制错误出现。

有关更多信息，请参阅 使用视图。

摘要

```
DROP VIEW [ IF EXISTS ] view_name
```

示例

```
DROP VIEW orders_by_date
```

```
DROP VIEW IF EXISTS orders_by_date
```

另请参阅CREATE VIEW、SHOW COLUMNS、SHOW CREATE VIEW、SHOW VIEWS 和 DESCRIBE VIEW。

MSCK REPAIR TABLE

在添加与 Hive 兼容的分区后，使用 `MSCK REPAIR TABLE` 命令可更新目录中的元数据。

`MSCK REPAIR TABLE` 命令将扫描在创建表后被添加到文件系统且兼容 Hive 的分区文件系统(例如 Amazon S3)。`MSCK REPAIR TABLE` 将比较表元数据中的分区和 S3 中的分区。如果您在创建表时指定的 S3 位置中存在新分区，则表会将这些分区添加到元数据和 Athena 表中。

在添加物理分区时，目录中的元数据将变得与文件系统中的数据布局不一致，需要将有关新分区的信息添加到目录中。要更新元数据，请运行 `MSCK REPAIR TABLE` 以便从 Athena 查询新分区中的数据。

> **Note**
>
> MSCK REPAIR TABLE 仅向元数据添加分区；它不会删除它们。要在 Amazon S3 中手动删除分区后从元数据中删除分区，请运行命令 ALTER TABLE *table-name* DROP PARTITION。有关更多信息，请参阅 ALTER TABLE DROP PARTITION。

注意事项和限制

在使用 MSCK REPAIR TABLE 时，请记住以下几点：

- 添加所有分区可能需要一些时间。如果此操作超时，它将处于不完整的状态，其中只有少数分区会被添加到目录中。应在同一个表上运行 MSCK REPAIR TABLE 语句，直到添加所有分区。有关更多信息，请参阅 在 Athena 中对数据进行分区。
- 对于不兼容 Hive 的分区，请使用ALTER TABLE ADD PARTITION加载分区，以便您可以查询数据。
- 要与 Athena 结合使用的分区位置必须使用 s3 协议（例如，s3://DOC-EXAMPLE-BUCKET/*folder*/）。在 Athena 中，当对包含的表运行 MSCK REPAIR TABLE 查询时，使用其他协议的位置（例如，s3a://*bucket*/*folder*/）将导致查询失败。
- 由于 MSCK REPAIR TABLE 同时扫描文件夹及其子文件夹以查找匹配的分区方案，请确保在单独的文件夹层次结构中保留单独表的数据。例如，假设您在 s3://DOC-EXAMPLE-BUCKET1 中拥有表 1 的数据，在 s3://DOC-EXAMPLE-BUCKET1/table-2-data 中拥有表 2 的数据。如果两个表都是按字符串分区的，则 MSCK REPAIR TABLE 会将表 2 的分区添加到表 1 中。为了避免这种情况，请使用单独的文件夹结构，如 s3://DOC-EXAMPLE-BUCKET1 和 s3://DOC-EXAMPLE-BUCKET2。请注意，此行为与 Amazon EMR 和 Apache Hive 一致。
- 由于已知问题，当分区值包含冒号（:）字符时（例如，当分区值为一个时间戳时）MSCK REPAIR TABLE 会静默失败。一个解决方法是使用 ALTER TABLE ADD PARTITION。
- MSCK REPAIR TABLE 不会添加开头为下划线（_）的分区列名称。要绕过此限制，使用 ALTER TABLE ADD PARTITION。

摘要

```
MSCK REPAIR TABLE table_name
```

示例

```
MSCK REPAIR TABLE orders;
```

故障排除

运行 MSCK REPAIR TABLE 后，如果 Athena 未将分区添加到 AWS Glue Data Catalog 中的表，请检查以下内容：

- AWS Glue 访问权限 - 确保 AWS Identity and Access Management（IAM）角色具有允许执行 glue:BatchCreatePartition 操作的策略。有关更多信息，请参阅本文后面的在 IAM policy 中允许 glue:BatchCreatePartition。
- Amazon S3 访问权限 – 确保角色具有的策略的权限足以访问 Amazon S3，包括 s3:DescribeJob 操作。有关允许哪些 Amazon S3 操作的示例，请参阅 Athena 中对 Amazon S3 存储桶的跨账户访问 中的示例存储桶策略。
- Amazon S3 对象键大小写 - 确保 Amazon S3 路径为小写而不是驼峰式大小写（例如，userid 而不是 userId），或者使用 ALTER TABLE ADD PARTITION 指定对象键名称。有关更多信息，请参阅本文后面的更改或重新定义 Amazon S3 路径。
- 查询超时 – MSCK REPAIR TABLE 最适用于首次创建表或在数据和分区元数据之间存在奇偶校验不确定性的情况。如果您使用 MSCK REPAIR TABLE 以频繁添加新分区(例如，每天添加)并遇到查询超时，请考虑使用 ALTER TABLE ADD PARTITION。
- 文件系统中缺少分区 - 如果您在 Amazon S3 中手动删除分区，然后运行 MSCK REPAIR TABLE，您可能会收到错误消息文件系统中缺少分区。这是因为 MSCK REPAIR TABLE 不会从表元数据中删除过时的分区。要从表元数据中移除已删除的分区，请运行 ALTER TABLE DROP PARTITION。请注意，SHOW PARTITIONS 将类似地仅列出元数据中的分区，而不是文件系统中的分区。
- “NullPointerException name is null（NullPointerException 名称为空）”错误

  如果您将 AWS Glue CreateTable API 操作或 AWS CloudFormation AWS::Glue::Table 模板创建用于 Athena 的表，而不指定 TableType 属性，然后运行 DDL 查询，如 SHOW CREATE TABLE 或者 MSCK REPAIR TABLE，则您将收到错误消息 FAILED: NullPointerException Name is null（失败：NullPointerException 名称为空）。

  要纠正该错误，请为 TableInput TableType 属性指定值，使其作为 AWS Glue CreateTable API 调用或 AWS CloudFormation 模板的一部分。TableType 可能的值包括 EXTERNAL_TABLE 或 VIRTUAL_VIEW。

此要求仅适用于使用 AWS Glue CreateTable API 操作或 AWS::Glue::Table 模板创建表的情形。如果您适用 DDL 语句或 AWS Glue 爬网程序为 Athena 创建表，则 TableType 属性将自动定义。

以下各节提供了一些详细信息。

在 IAM policy 中允许 glue:BatchCreatePartition

审核附加到您用于执行 MSCK REPAIR TABLE 的角色的 IAM policy。当您将 AWS Glue Data Catalog 与 Athena 一起使用时，IAM policy 必须允许 glue:BatchCreatePartition 操作。有关允许 glue:BatchCreatePartition 操作的 IAM policy 的示例，请参阅 AWS 托管策略：AmazonAthenaFullAccess。

更改或重新定义 Amazon S3 路径

如果 Amazon S3 路径中的一个或多个对象键采用驼峰式大小写而不是小写，MSCK REPAIR TABLE 可能不会将分区添加到 AWS Glue Data Catalog。例如，如果您的 Amazon S3 路径包含对象键名称 userId，则可能不会将以下分区添加到 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/
```

要解决该问题，可以执行下列操作之一：

- 创建 Amazon S3 对象键时，请使用小写字母而不是驼峰式大小写：

```
s3://DOC-EXAMPLE-BUCKET/path/userid=1/

s3://DOC-EXAMPLE-BUCKET/path/userid=2/

s3://DOC-EXAMPLE-BUCKET/path/userid=3/
```

- 使用 ALTER TABLE ADD PARTITION 重新定义位置，如以下示例所示：

```
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/'
```

请注意，尽管 Amazon S3 对象键名称可以使用大写字母，但 Amazon S3 存储桶名称本身一定要使用小写字母。有关更多信息，请参阅《Amazon S3 用户指南》中的对象键命名指南和存储桶命名规则。

SHOW COLUMNS

仅显示单个指定表或视图的列名。要获取更多详细信息，请改为查询 AWS Glue Data Catalog。有关信息和示例，请参阅 查询 AWS Glue Data Catalog 主题中的以下章节：

- 要查看列元数据（如数据类型），请参阅 列出或搜索指定表或视图的列。
- 要查看 AwsDataCatalog 中特定数据库中所有表的所有列，请参阅 列出或搜索指定表或视图的列。
- 要查看 AwsDataCatalog 中所有数据库中所有表的所有列，请参阅 列出所有表的所有列。
- 要查看数据库中的特定表的共同列，请参阅 列出特定的表的共同列。

摘要

```
SHOW COLUMNS {FROM|IN} database_name.table_name
```

```
SHOW COLUMNS {FROM|IN} table_name [{FROM|IN} database_name]
```

FROM 和 IN 关键字可互换使用。如果 *table_name* 或者 *database_name* 具有诸如连字符之类的特殊字符，请用反引号将名称括起来（例如 `my-database`.`my-table`）。不要使用单引号或双引号将 *table_name* 或者 *database_name* 括起。目前，并不支持使用 LIKE 和模式匹配表达式。

示例

以下等效示例显示了 customers 数据库中的 orders 表。前两个示例假设 customers 是当前数据库。

```
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

分析名为 `table_name` 的现有表以生成创建它的查询。

摘要

```
SHOW CREATE TABLE [db_name.]table_name
```

参数

TABLE [db_name.]table_name

db_name 参数是可选的。如果省略，则上下文默认为当前数据库。

> **Note**
>
> 表名称是必需的。

示例

```
SHOW CREATE TABLE orderclickstoday;
```

```
SHOW CREATE TABLE `salesdata.orderclickstoday`;
```

故障排除

如果您将 AWS Glue CreateTable API 操作或 AWS CloudFormation `AWS::Glue::Table` 模板创建用于 Athena 的表，而不指定 `TableType` 属性，然后运行 DDL 查询，如 `SHOW CREATE TABLE` 或者 `MSCK REPAIR TABLE`，则您将收到错误消息失败：NullPointerException 名称为空。

要纠正该错误，请为 TableInput TableType 属性指定值，使其作为 AWS Glue CreateTable API 调用或 AWS CloudFormation 模板的一部分。TableType 可能的值包括 EXTERNAL_TABLE 或 VIRTUAL_VIEW。

此要求仅适用于使用 AWS Glue CreateTable API 操作或 AWS::Glue::Table 模板创建表的情形。如果您适用 DDL 语句或 AWS Glue 爬网程序为 Athena 创建表，则 TableType 属性将自动定义。

SHOW CREATE VIEW

显示创建指定视图的 SQL 语句。

摘要

```
SHOW CREATE VIEW view_name
```

示例

```
SHOW CREATE VIEW orders_by_date
```

另请参阅 CREATE VIEW 和 DROP VIEW。

SHOW DATABASES

列出元存储中定义的所有数据库。您可以使用 DATABASES 或 SCHEMAS。它们具有相同的含义。

SHOW DATABASES 的编程等效项是 ListDatabases Athena API 操作。AWS SDK for Python (Boto3) 中的等效方法是 list_databases。

摘要

```
SHOW {DATABASES | SCHEMAS} [LIKE 'regular_expression']
```

参数

[LIKE '*regular_expression*']

从数据库列表中筛选出那些与您指定的 *regular_expression* 匹配的数据库。对于通配符匹配，您可以使用组合 .*，它将任何字符匹配 0 到无限次。

示例

```
SHOW SCHEMAS;
```

```
SHOW DATABASES LIKE '.*analytics';
```

SHOW PARTITIONS

按未排序顺序列出 Athena 表中的所有分区。

摘要

```
SHOW PARTITIONS table_name
```

- 要显示表中的分区并按特定顺序列出分区，请参阅查询 AWS Glue Data Catalog页面上的列出特定表的分区部分。
- 要查看分区的内容，请参阅在 Athena 中对数据进行分区页面上的查询数据部分。
- SHOW PARTITIONS 不会列出由 Athena 投影但未在 AWS Glue 目录中注册的分区。有关分区投影的信息，请参阅使用 Amazon Athena 分区投影。
- SHOW PARTITIONS 将列出元数据中的分区，而不是实际文件系统中的分区。要在 Amazon S3 中手动删除分区后更新元数据，请运行 ALTER TABLE DROP PARTITION。

示例

以下示例查询显示了 flight_delays_csv 表的分区，其中显示了来自美国运输部的飞行表数据。有关使用示例 flight_delays_csv 表的更多信息，请参阅 用于 CSV、TSV 和自定义分隔文件的 LazySimpleSerDe。表按年份进行分区。

```
SHOW PARTITIONS flight_delays_csv
```

结果

```
year=2007
year=2015
year=1999
year=1993
year=1991
```

```
year=2003
year=1996
year=2014
year=2004
year=2011
...
```

以下示例查询显示了 impressions 表的分区，其中包含示例 Web 浏览数据。有关使用示例 impressions 表的更多信息，请参阅 在 Athena 中对数据进行分区。该表按照 dt(日期时间)列进行分区。

```
SHOW PARTITIONS impressions
```

结果

```
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
...
```

按排序顺序列出分区

要在结果列表中对分区进行排序，请使用以下 SELECT 语法，而不是 SHOW PARTITIONS。

```
SELECT * FROM database_name."table_name$partitions" ORDER BY column_name
```

以下查询显示了 flight_delays_csv 示例的分区列表，但按排序顺序排列。

```
SELECT * FROM "flight_delays_csv$partitions" ORDER BY year
```

结果

```
year
1987
```

```
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
...
```

有关更多信息，请参阅 查询 AWS Glue Data Catalog 页面上的 列出特定表的分区 部分。

SHOW TABLES

列出数据库中的所有基本表和视图。

摘要

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

参数

[IN database_name]

指定将列出其表的 database_name。如果省略，则会采用当前上下文中的数据库。

> **Note**
>
> 如果 database_name 使用不支持的字符（比如连字符），则 SHOW TABLES 可能失败。作为一种解决方法，请尝试用反引号将数据库名称括起来。

['regular_expression']

从表的列表中筛选出那些符合您指定的 regular_expression 的表。要指示 AWSDataCatalog 表中的任何字符，您可以使用 * 或者 .* 通配符表达式。对于 Apache Hive 数据库，请使用 .* 通配符表达式。要指示字符之间的选择，请使用 | 字符。

示例

Example – 显示数据库 **sampledb** 中的所有表

```
SHOW TABLES IN sampledb
```

Results

```
alb_logs
cloudfront_logs
elb_logs
flights_2016
flights_parquet
view_2016_flights_dfw
```

Example – 显示 **sampledb** 中包含单词“flights”的所有表的名称

```
SHOW TABLES IN sampledb '*flights*'
```

Results

```
flights_2016
flights_parquet
view_2016_flights_dfw
```

Example – 显示 **sampledb** 中以单词“log”结尾的所有表的名称

```
SHOW TABLES IN sampledb '*logs'
```

Results

```
alb_logs
cloudfront_logs
elb_logs
```

SHOW TBLPROPERTIES

列出命名表的表属性。

摘要

```
SHOW TBLPROPERTIES table_name [('property_name')]
```

参数

[('property_name')]

如果包含，则只会列出名为 property_name 的属性的值。

示例

```
SHOW TBLPROPERTIES orders;
```

```
SHOW TBLPROPERTIES orders('comment');
```

SHOW VIEWS

列出指定数据库或当前数据库（如果省略数据库的名称）中的视图。将可选 LIKE 子句与一个正则表达式结合使用来限制视图名称列表。

Athena 返回 STRING 类型值列表，其中每个值就是一个视图名称。

摘要

```
SHOW VIEWS [IN database_name] [LIKE 'regular_expression']
```

参数

[IN database_name]

指定将列出其视图的 database_name。如果省略，则会采用当前上下文中的数据库。

[LIKE 'regular_expression']

从视图列表中筛选出那些符合您指定的 regular_expression 的视图。只能使用表示任何字符的通配符 * 或表示在字符间进行选择的 |。

示例

```
SHOW VIEWS;
```

```
SHOW VIEWS IN marketing_analytics LIKE 'orders*'
```

另请参阅 SHOW COLUMNS、SHOW CREATE VIEW、DESCRIBE VIEW 和 DROP VIEW。

Amazon Athena 中 SQL 查询的注意事项和限制

在 Athena 中运行查询时，请记住以下注意事项和限制：

- 存储过程 – 不支持存储过程。
- 最大分区数 – 在 `CREATE TABLE AS SELECT` (CTAS)语句中可以创建的最大分区数为 100。有关信息，请参阅 CREATE TABLE AS。有关解决方法，请参阅使用 CTAS 和 INSERT INTO 绕过 100 分区限制。
- 不支持的语句 – 不支持以下语句：
  - 不支持 `CREATE TABLE LIKE`。
  - 不支持 `DESCRIBE INPUT` 和 `DESCRIBE OUTPUT`。
  - `MERGE` 语句仅支持事务表格式。有关更多信息，请参阅 MERGE INTO。
  - 不支持 `UPDATE` 语句。
- Trino 和 Presto 连接器 – 不支持 Trino 和 Presto 连接器。使用 Amazon Athena 联合查询连接数据来源。有关更多信息，请参阅 使用 Amazon Athena 联合查询。
- 对于具有多个分区的表超时 – 在查询具有数千个分区的表时，Athena 可能会超时。当表具有许多不属于类型 `string` 的分区时，可能会发生这种情况。使用类型 `string` 时，Athena 在元数据仓级别修剪分区。但是，当您使用其他数据类型时，Athena 会在服务器端修剪分区。您拥有的分区越多，此过程所花的时间越长，并且您的查询越可能超时。要解决此问题，请将您的分区类型设置为 `string`，以便 Athena 在元数据仓级别修剪分区。这可以减少开销并防止查询超时。
- S3 Glacier 支持 - 有关查询已还原 Amazon S3 Glacier 对象的信息，请参阅 查询还原的 Amazon S3 Glacier 对象。
- 将文件视为隐藏 – Athena 将以下划线(_)或句点(.)开头的源文件视为隐藏文件。要解决此限制，请重命名文件。
- 行或列大小限制 – 单行或其列的大小不能超过 32 兆字节。例如，当 CSV 或 JSON 文件中的行包含 300 兆字节的单列时，可能会超出此限制。超过此限制也会产生错误消息 Line too long in text life (文本文件中的行太长)。要解决此限制，请确保任何一行中列的数据总和小于 32 MB。

- LIMIT 子句最大值 – 可以为 `LIMIT` 子句指定的最大行数

  9223372036854775807。使用 `ORDER BY` 时，LIMIT 子句支持的最大行数为 2147483647。超过此限制会导致错误消息 NOT_SUPPORTED: ORDER BY LIMIT > 2147483647 is not supported（不支持：不支持大于 2147483647 限制的指令）。
- information_schema – 如果您的 AWS Glue 元数据较少或中等，则查询 `information_schema` 的性能会最佳。如果您有大量的元数据，则可能会出现错误。有关查询 `information_schema` 数据库的 AWS Glue 元数据的信息，请参阅查询 AWS Glue Data Catalog。
- 数组初始化 – 由于 Java 中的限制，无法在 Athena 中初始化具有超过 254 个参数的数组。
- 隐藏的元数据列 - 视图不支持 Hive 或 Iceberg 隐藏的元数据列 `$bucket`、`$file_modified_time`、`$file_size` 和 `$partition`。有关在 Athena 中使用 `$path` 元数据列的信息，请参阅 获取 Amazon S3 中源数据的文件位置 。

有关最大查询字符串长度、查询超时限额以及活动 DML 查询数限额的信息，请参阅 服务限额。

在 Athena 中进行故障排除

Athena 团队从客户问题中收集了以下故障排除信息。虽然并不全面，但它包括有关一些常见性能、超时和内存不足问题的建议。

主题

- CREATE TABLE AS SELECT (CTAS)
- 数据文件问题
- Linux Foundation Delta Lake 表
- 联合查询
- JSON 相关错误
- MSCK REPAIR TABLE
- 输出问题
- Parquet 问题
- 分区问题
- 权限
- 查询语法问题
- 查询超时问题
- 节流问题

- 视图
- 工作组
- 其他 资源
- Athena 错误目录

CREATE TABLE AS SELECT (CTAS)

重复的数据与并发 CTAS 语句一起出现

Athena 不维护 CTAS 的并发验证。确保同一位置不会同时出现重复的 CTAS 语句。即使 CTAS 或 INSERT INTO 语句失败，孤立数据也可以保留在语句中指定的数据位置。

HIVE_TOO_MANY_OPEN_PARTITIONS

当您使用 CTAS 语句来创建包含超过 100 个分区的表时，可能会收到错误消息则可能会收到错误消息 HIVE_TOO_MANY_OPEN_PARTITIONS: Exceeded limit of 100 open writers for partitions/buckets（HIVE_TOO_MANY_OPEN_PARTISS：分区/存储桶超过 100 个打开的写入程序限制）。要绕过此限制，您可以使用一个 CTAS 语句和一系列 `INSERT INTO` 语句，每个后一种语句将创建或插入不超过 100 个分区。有关更多信息，请参阅 使用 CTAS 和 INSERT INTO 绕过 100 分区限制。

数据文件问题

Athena 无法读取隐藏的文件

Athena 将以下划线 (_) 或句点 (.) 开头的源文件视为隐藏文件。要解决此限制，请重命名文件。

Athena 读取我从 AWS Glue 爬网程序中排除的文件

Athena 不承认您为 AWS Glue 爬网程序指定的排除模式。例如，如果您有一个 Amazon S3 存储桶，其中包含 `.csv` 和 `.json` 文件，并且您从爬网程序中排除了 `.json` 文件时，Athena 会查询两组文件。要避免这种情况，请将要排除的文件放置在其他位置。

HIVE_BAD_DATA：分析字段值时出错

以下情况下可能会出现此错误：

- 表中定义的数据类型与源数据不匹配，或者单个字段包含不同类型的数据。有关建议的分辨率，请参阅 AWS 知识中心的 我的 Amazon Athena 查询失败，出现错误“HIVE_BAD_DATA: Error parsing

field value for field X: For input string: "12312845691"（HIVE_BAD_DATA：分析 x 字段值时出错：输入字符串：“12312845691”）”。

- 空值存在于整数字段中。一种解决方法是创建带有空值为 `string` 的列，然后使用 `CAST` 转换查询中的字段，为空值提供 `0` 的默认值。有关更多信息，请参阅 AWS 知识中心的 当我在 Athena 中查询 CSV 数据时，出现错误“HIVE_BAD_DATA: Error parsing field value '' for field X: For input string: ""（HIVE_BAD_DATA：分析 X 字段''值时出错：输入字符串：“”）”。

HIVE_CANNOT_OPEN_SPLIT: Error opening Hive split s3://DOC-EXAMPLE-BUCKET

当您查询具有大量对象的 Amazon S3 存储桶前缀时，可能会发生此错误。有关更多信息，请参阅 AWS 知识中心的如何解决 Athena 中的“HIVE_CANNOT_OPEN_SPLIT：打开 Hive split s3://DOC-EXAMPLE-BUCKET/ 时出错：减速”错误？。

HIVE_CURSOR_ERROR: com.amazonaws.services.s3.model.AmazonS3Exception: The specified key does not exist

当查询正在运行而删除文件时，通常会发生此错误。您可重新运行查询，或检查您的工作流，以查看查询运行时是否有另一个任务或进程正在修改文件。

HIVE_CURSOR_ERROR：输入流式传输意外结束

此消息表示该文件已损坏或为空。检查文件的完整性，然后重启查询。

HIVE_FILESYSTEM_ERROR: Incorrect fileSize ***1234567*** for file（HIVE_FILESSTEM_ERROR：文件的文件大小错误 1234567）

文件在查询规划和查询执行之间更改时，可能会出现此消息。Amazon S3 上的文件被就地替换（例如，PUT 在对象已存在的密钥上执行）时，通常会出现此消息。运行查询时，Athena 不支持删除或替换文件内容。为避免此错误，请在查询不运行时安排覆盖或删除文件的任务，或者只将数据写入新文件或分区。

HIVE_UNKNOWN_ERROR：无法创建输入格式

此错误可能是由类似于以下的问题导致的：

- AWS Glue 爬网程序无法对数据格式进行分类
- 某些 AWS Glue 表定义属性为空
- Athena 不支持 Amazon S3 中文件的数据格式

有关更多信息，请参阅 AWS 知识中心的如何解决 Athena 中的“无法创建输入格式”错误？或观看知识中心视频。

为保存查询结果而提供的 S3 位置无效。

确保您为查询结果指定了有效的 S3 位置。有关更多信息，请参阅使用查询结果、最近查询和输出文件主题中的 指定查询结果位置。

Linux Foundation Delta Lake 表

Delta Lake 表架构不同步

如果您查询的 Delta Lake 表在 AWS Glue 中的架构已过时，可能会收到以下错误消息：

```
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.
```

如果在将架构添加到 Athena 之后在 AWS Glue 中对其进行了修改，架构可能会过时。要更新架构，请执行以下任一步骤：

- 在 AWS Glue 中运行 AWS Glue 爬网程序。
- 在 Athena 中，删除表并重新创建。
- 手动添加缺失的列，方法可以是在 Athena 中使用 ALTER TABLE ADD COLUMNS 语句，也可以是在 AWS Glue 中编辑表架构。

联合查询

调用 ListTableMetadata 时超时

如果数据来源中有很多表、数据来源运行缓慢或网络运行缓慢，则对 ListTableMetadata API 的调用可能会超时。要解决此问题，请执行以下步骤。

- 检查表的数量 – 如果您有 1000 多个表，请尝试减少表的数量。为了获得最快的 `ListTableMetadata` 响应，我们建议每个目录的表数少于 1000 个。
- 检查 Lambda 配置 - 监控 Lambda 函数行为至关重要。使用联合目录时，请务必检查 Lambda 函数的执行日志。根据结果，相应地调整内存和超时值。要确定任何潜在的超时问题，请重新访问您

的 Lambda 配置。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的配置函数超时（控制台）。

- 检查联合数据来源日志 - 检查来自联合数据来源的日志和错误消息，以查看是否存在任何问题或错误。这些日志可以提供有关超时原因的有用见解。
- 使用 **StartQueryExecution** 提取元数据 – 如果您有 1000 多个表，则使用联合连接器检索元数据所需的时间可能会超出预期。由于 StartQueryExecution 的异步性质可确保 Athena 以最佳方式运行查询，因此可以考虑使用 StartQueryExecution 作为 ListTableMetadata 的替代方案。以下 AWS CLI 示例说明如何使用 StartQueryExecution（而非 ListTableMetadata）来获取数据目录中表的所有元数据。

  首先，运行一个查询来获取所有表，如下例所示。

```
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"
```

  其次，检索单个表的元数据，如下例所示。

```
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"
```

  获取结果所需的时间取决于目录中表的数量。

有关联合查询故障排除的更多信息，请参阅 GitHub 上 awslabs/aws-athena-query-federation 部分中的常见问题，或参阅各个 Athena 数据来源连接器的文档。

JSON 相关错误

尝试读取 JSON 数据时出现 NULL 或不正确的数据错误

当您尝试读取 JSON 数据时，NULL 或不正确的数据错误可能是由于多种原因造成的。要识别在使用 OpenX SerDE 时导致错误的行，请将 ignore.malformed.json 设置为 true。格式错误的记录将返回为 NULL。有关更多信息，请参阅 AWS 知识中心的我在 Amazon Athena 中尝试读取 JSON 数据时收到错误观看知识中心视频。

HIVE_BAD_DATA: Error parsing field value for field 0: java.lang.String cannot be cast to org.openx.data.jsonserde.json.JSONObject

当 OpenX JSON SerDe 无法解析 Athena 查询中的列时，就会引发此错误。如果您将列定义为 `map` 或者 `struct`，但基础数据实际上是 `string`、`int` 或其他基本类型，就会出现这种状况。

HIVE_CURSOR_ERROR: Row is not a valid JSON Object - JSONException: Duplicate key（HIVE_CURSOR_ERROR：行不是有效的 JSON 对象 - JSONException：重复的键）

当您使用 Athena 查询拥有多个同名标签，但大小写不同的 AWS Config 资源时，会出现此错误。解决方案是使用 `WITH SERDEPROPERTIES 'case.insensitive'='false'` 运行 `CREATE TABLE` 并映射名称。有关 `case.insensitive` 和映射的更多信息，请参阅 JSON SerDe 库。有关更多信息，请参阅 AWS 知识中心中的 当在 Athena 中读取来自 AWS Config 的文件时，如何解决“HIVE_CURSOR_ERROR: Row is not a valid JSON Object - JSONException: Duplicate key（HIVE_CURSOR_ERROR：行不是有效的 JSON 对象 - JSONException：重复的键）”？

美观打印的 JSON 文本的 HIVE_CURSOR_ERROR 消息

Hive JSON SerDe 和 OpenX JSON SerDe 库期望每个 JSON 文档都位于单行文本中，并且不使用行终止字符分隔记录中的字段。如果 JSON 文本采用美观的打印格式，当您在创建表后尝试对其进行查询时，可能会收到类似以下内容的错误消息：HIVE_CURSOR_ERROR: Row is not a valid JSON Object（HIVE_CURSOR_ERROR：行不是有效的 JSON 对象）或 HIVE_CURSOR_ERROR: JsonParseException: Unexpected end-of-input: expected close marker for OBJECT（HIVE_CURSOR_ERROR：JsonParseException：意外的输入结束：对象的预期关闭标记）。有关更多信息，请参阅 GitHub 上 OpenX SerDe 文档中的 JSON data files（JSON 数据文件）。

多个 JSON 记录返回的 SELECT COUNT 为 1

如果您使用的是 OpenX JSON SerDe，请确保记录由换行符分隔。有关更多信息，请参阅 AWS 知识中心中的即使输入 JSON 文件有多条记录，Amazon Athena 中的 SELECT COUNT 查询也仅返回一条记录。

无法查询由使用自定义 JSON 分类器的 AWS Glue 爬网程序创建的表

Athena 引擎不支持自定义 JSON 分类器。要解决此问题，请创建不含自定义分类器的新表。要转换 JSON，您可以使用 CTAS 或创建视图。例如，如果您正在使用数组，则可以使用 UNNEST 选项来拼

合 JSON。另一个选择是使用支持自定义分类器的 AWS Glue ETL 任务，在 Amazon S3 中将数据转换为 Parquet，然后在 Athena 中查询它。

MSCK REPAIR TABLE

有关 MSCK REPAIR TABLE 相关问题的信息，请参阅 MSCK REPAIR TABLE 页面的 注意事项和限制 和 故障排除 的部分。

输出问题

无法验证/创建输出存储桶

如果指定的查询结果位置不存在或不存在适当的权限，则可能会发生此错误。有关更多信息，请参阅 AWS 知识中心的 如何解决 Amazon Athena 中的“unable to verify/create output bucket（无法验证/创建输出存储桶）”错误？

TIMESTAMP 结果为空

Athena 需要 Java TIMESTAMP 格式。有关更多信息，请参阅 AWS 知识中心中当我在 Amazon Athena 中查询表时，TIMESTAMP 结果为空。

以 CSV 以外的格式存储 Athena 查询输出

默认情况下，Athena 仅以 CSV 格式输出文件。若要输出采用其他格式的 SELECT 查询结果，可以使用 UNLOAD 语句。有关更多信息，请参阅 UNLOAD。您还可以使用 CTAS 查询，该查询使用 `format` 表属性配置输出格式。与 UNLOAD 不同，CTAS 技术需要创建表格。有关更多信息，请参阅 AWS 知识中心中的如何以 CSV 以外的格式（如压缩格式）存储 Athena 查询输出？。

为保存查询结果而提供的 S3 位置无效

如果您的输出存储桶位置与运行查询的区域不一致，则可能会收到此错误消息。要避免这种情况，请在运行查询的区域中指定查询结果位置。要查看步骤，请参阅 指定查询结果位置。

Parquet 问题

org.apache.parquet.io.GroupColumnIO cannot be cast to org.apache.parquet.io.PrimitiveColumnIO

此错误是由 parquet 架构不匹配引起的。具有非基本类型的列（例如，`array`）在 AWS Glue 中被声明为原始类型（例如，`string`）。若要解决此问题，请检查文件中的数据架构，并将其与 AWS Glue 中声明的架构相比较。

Parquet 统计数据问题

当读取 Parquet 数据时，您可能会收到如下错误消息：

```
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.
```

要解决此问题，请使用 CREATE TABLE 或 ALTER TABLE SET TBLPROPERTIES 语句将 Parquet Serde `parquet.ignore.statistics` 属性设置为 `true`，如以下示例所示。

CREATE TABLE 示例

```
...
ROW FORMAT SERDE
'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
WITH SERDEPROPERTIES ('parquet.ignore.statistics'='true')
STORED AS PARQUET
...
```

ALTER TABLE 示例

```
ALTER TABLE ... SET TBLPROPERTIES ('parquet.ignore.statistics'='true')
```

有关 Parquet Hive SerDe 的更多信息，请参阅 Parquet SerDe。

分区问题

MSCK REPAIR TABLE 不会删除过时的分区

如果您在 Amazon S3 中手动删除分区，然后运行 MSCK REPAIR TABLE，您可能会收到错误消息文件系统中缺少分区。发生这种情况的原因是 MSCK REPAIR TABLE 不会从表元数据中删除过时的分区。请使用 ALTER TABLE DROP PARTITION 手动删除过时的分区。有关更多信息，请参阅 MSCK REPAIR TABLE 主题的“疑难解答”部分。

MSCK REPAIR TABLE 故障

当大量的分区（例如，超过 100,000 个）与特定表相关联时，`MSCK REPAIR TABLE` 可能会因内存限制而故障。要解决此限制，请换成 ALTER TABLE ADD PARTITION。

MSCK REPAIR TABLE 检测到分区，但未将它们添加到 AWS Glue

如果 Amazon S3 路径混合使用大小写而不是全小写，或者 IAM policy 不允许 `glue:BatchCreatePartition` 操作，则会出现此问题。有关更多信息，请参阅 AWS 知识中心的 MSCK REPAIR TABLE 检测到 Athena 中的分区，但未将它们添加到 AWS Glue Data Catalog。

日期格式为 dd-MM-yyyy-HH-mm-ss 或 yyyy-MM-dd 的分区投影范围不起作用

要正常工作，必须将日期格式设置为 `yyyy-MM-dd HH:00:00`。有关更多信息，请参阅 Stack Overflow 文章：Athena partition projection not working as expected（Athena 分区投影无法按预期工作）。

PARTITION BY 不支持 BIGINT 类型

将数据类型转换为 `string`，然后重试。

没有可用的有意义分区

此错误消息通常意味着分区设置已损坏。若要解决此问题，请删除该表并创建具有新分区的表。

分区投影不能与范围分区结合使用

检查时间范围单位projection.*<columnName>*.interval.unit 是否匹配分区的分隔符。例如，如果分区按天分隔，则小时范围单位将不起作用。

用连字符指定范围时发生分区投影错误

使用连字符而不是逗号来指定 `range` 表属性会产生一个错误，比如 INVALID_TABLE_PROPERTY: For input string: "*number-number*"。确保使用逗号来分隔范围值，而不是使用连字符。有关更多信息，请参阅 整数类型。

HIVE_UNKNOWN_ERROR：无法创建输入格式

一个或多个 Glue 分区以不同的格式声明，因为每个 Glue 分区都有自己的特定输入格式。请检查您的分区是如何在 AWS Glue 中定义的。

HIVE_PARTITION_SCHEMA_MISMATCH

如果分区的架构与表的架构不同，则查询可能会失败，并显示错误消息：HIVE_PARTITION_SCHEMA_MISMATCH。有关更多信息，请参阅 同步分区架构以避免“HIVE_PARTITION_SCHEMA_MISMATCH”。

SemanticException 表未进行分区，但存在分区规范

如果 CREATE TABLE 语句中没有定义分区，则会发生此错误。有关更多信息，请参阅 AWS 知识中心的如何解决 Athena 中的错误“失败：SemanticException 表未进行分区，但存在分区规范”？。

零字节 _$folder$ 文件

如果您运行 ALTER TABLE ADD PARTITION 语句并错误地指定了已存在的分区和错误的 Simple Storage Service (Amazon S3) 位置，格式为 *partition_value*_$folder$ 的零字节占位符文件将在 Simple Storage Service (Amazon S3) 中创建。您必须手动移除这些文件。

为了防止这种情况发生，请在 ALTER TABLE ADD PARTITION 语句中使用 ADD IF NOT EXISTS 语法，像这样：

```
ALTER TABLE table_name ADD IF NOT EXISTS PARTITIION […]
```

从分区数据返回零条记录

此问题可能由于多个原因引发。有关可能的原因和解决方案，请参阅 AWS 知识中心的我在 Amazon Athena 中创建了一个具有定义分区的表，但是当我查询表时，返回零条记录。

另请参阅 HIVE_TOO_MANY_OPEN_PARTITIONS。

权限

查询 Amazon S3 时的访问被拒绝错误

当您没有读取存储桶中数据的权限、写入结果存储桶的权限或 Amazon S3 路径包含区域端点（如 us-east-1.amazonaws.com）时，则可能出现此错误。有关更多信息，请参阅 AWS 知识中心的 当我运行 Athena 查询时，出现“access denied（访问被拒绝）”错误。

对 Amazon S3 中的加密数据运行 DDL 查询时出现 Access denied with status code: 403（访问被拒绝，状态代码：403）错误

如果满足以下条件，则您可能会收到错误消息访问被拒绝（服务：Amazon S3；状态代码：403；错误代码：AccessDenied；请求 ID：*<request_id>*）：

1. 您运行了一个 DDL 查询，如 ALTER TABLE ADD PARTITION 或者 MSCK REPAIR TABLE。
2. 您的存储桶配置了默认加密，并使用 SSE-S3。

3. 存储桶还具有类似以下内容的存储桶策略，该策略强制 PutObject 请求指定 PUT 标题 "s3:x-amz-server-side-encryption": "true" 和 "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"
                }
            }
        }
    ]
}
```

在这种情况下，建议的解决方案是删除上述存储桶策略，因为存储桶的默认加密已经存在。

查询另一个账户中的 Amazon S3 存储桶时，出现 Access denied with status code: 403（访问被拒绝，状态代码：403）错误

尝试查询由另一个 AWS 服务 写入的日志，且第二个账户是存储桶拥有者，但不拥有存储桶中的对象时，可能会出现此错误。有关更多信息，请参阅 AWS 知识中心的 当我在另一个账户中查询存储桶时，我收到 Amazon Athena 中的 Amazon S3 异常“access denied with status code: 403（访问被拒绝，状态代码：403）”或观看知识中心视频。

使用 IAM 角色凭证连接到 Athena JDBC 驱动程序

您可以检索角色的临时凭证，以对 JDBC 与 Athena 的连接进行身份验证。临时凭证的最大生命周期为 12 小时。有关更多信息，请参阅 AWS 知识中心在使用 JDBC 驱动程序连接到 Athena 时，如何使用我的 IAM 角色凭证或切换到另一个 IAM 角色？。

查询语法问题

失败：NullPointerException 名称为空

如果您将 AWS Glue CreateTable API 操作或 AWS CloudFormation `AWS::Glue::Table` 模板创建用于 Athena 的表，而不指定 `TableType` 属性，然后运行 DDL 查询，如 `SHOW CREATE TABLE` 或者 `MSCK REPAIR TABLE`，则您将收到错误消息失败：NullPointerException 名称为空。

要纠正该错误，请为 TableInput `TableType` 属性指定值，使其作为 AWS Glue `CreateTable` API 调用或 AWS CloudFormation 模板的一部分。`TableType` 可能的值包括 `EXTERNAL_TABLE` 或 `VIRTUAL_VIEW`。

此要求仅适用于使用 AWS Glue `CreateTable` API 操作或 `AWS::Glue::Table` 模板创建表的情形。如果您适用 DDL 语句或 AWS Glue 爬网程序为 Athena 创建表，则 `TableType` 属性将自动定义。

函数未注册

当您尝试使用 Athena 不支持的函数时，会发生此错误。有关 Athena 支持的函数的列表，请参阅 Amazon Athena 中的函数 或在查询编辑器中运行 `SHOW FUNCTIONS` 语句。您还可以编写自己的用户定义函数 (UDF)。有关更多信息，请参阅 AWS 知识中心的如何解决 Athena 中的“函数未注册”语法错误？。

GENERIC_INTERNAL_ERROR 异常

`GENERIC_INTERNAL_ERROR` 异常可能有各种原因，包括以下原因：

- GENERIC_INTERNAL_ERROR: null – 在以下任一情况下，您可能会看到此异常：
  - 表定义中列的数据类型与数据集的实际数据类型之间存在架构不匹配。
  - 正在使用不正确的语法运行 `CREATE TABLE AS SELECT` (CTAS) 查询。
- GENERIC_INTERNAL_ERROR: Parent builder is null（GENERIC_INTERNAL_ERROR：父生成器为 null）– 查询其列数据类型为 `array` 的表，并且正在使用 OpenCSVSerDe 库时，则可能会看到此异常。OpenCSVSerde 格式不支持 `array` 数据类型。

- GENERIC_INTERNAL_ERROR：值超过 MAX_INT – 使用数据类型 INT 定义源数据列，并且源数据列中的某个数值大于 2147483647 时，您可能会看到此异常。
- GENERIC_INTERNAL_ERROR：值超过 MAX_BYTE – 当源数据列的某个数值超过数据类型 BYTE 允许的大小时，您可能会看到此异常。数据类型 BYTE 等同于 TINYINT。TINYINT 是一个有符号的 8 位整数，采用二进制补码格式，最小值为 -128，最大值为 127。
- GENERIC_INTERNAL_ERROR：分区值的数量与筛选条件的数量不匹配 – 如果 Amazon Simple Storage Service（Amazon S3）数据具有不一致的分区，您可能会看到此异常。在以下任一情况下，均可能存在不一致的分区：
  - Amazon S3 上的分区已更改（例如：添加了新分区）。
  - 表中的分区列数与分区元数据中的分区列数不匹配。

有关其中每个错误的更多详细信息，请参阅 AWS 知识中心中的《当我在 Amazon Athena 中查询表时，如何解决“GENERIC_INTERNAL_ERROR”错误？》。

匹配组的数量与列数不匹配

此错误当您在 CREATE TABLE 语句中使用 Regex SerDe，并且正则表达式匹配组的数量与您为表指定的列数不匹配时，会出现此错误。有关更多信息，请参阅 AWS 知识中心中的 如何解决 Amazon Athena 中的 RegexSerDe 错误“number of matching groups doesn't match the number of columns（匹配组的数量与列数不匹配）”？。

queryString 无法满足约束条件：成员的长度必须小于或等于 262144

Athena 中最大查询字符串长度（262,144 字节）不是可调配额。AWS Support 无法为您增加配额，但您可以通过将长查询拆分为较小的查询来解决此问题。有关更多信息，请参阅 AWS 知识中心的如何增加 Athena 中的最大查询字符串长度？。

SYNTAX_ERROR：无法解析列

当您查询由 AWS Glue 爬虫程序从具有字节顺序标记 (BOM) 的 UTF-8 编码 CSV 文件查询表时，会出现此问题。AWS Glue 不会识别 BOM 并会将其更改为问号，Amazon Athena 无法识别这些问号。解决方案是删除 Athena 或 AWS Glue 中的问号。

函数调用的参数太多

在 Athena 引擎版本 3 中，函数的参数不能超过 127 个。此限制是设计使然。如果您使用的函数参数超过 127 个，则会出现如下错误消息：

TOO_MANY_ARGUMENTS: line *nnn*:*nn*: 函数调用 *function_name*() 的参数太多了。

要解决此问题，请为每次函数调用使用较少的参数。

查询超时问题

如果 Athena 查询遇到超时错误，请查看您的 CloudTrail 日志。由于 AWS Glue 或 Lake Formation API 存在节流，查询可能会超时。出现这些错误时，相应的错误消息可能表明是查询超时问题，而不是节流问题。要排查问题，您可以在联系 AWS Support 之前查看自己的 CloudTrail 日志。有关更多信息，请参阅查询 AWS CloudTrail 日志 和使用 AWS CloudTrail 记录 Amazon Athena API 调用。

有关调用 `ListTableMetadata` API 时联合查询出现的查询超时问题的信息，请参阅调用 ListTableMetadata 时超时。

节流问题

如果您的查询超出了依赖服务（如 Amazon S3、AWS KMS、AWS Glue 或者 AWS Lambda）的限制，则可能会出现以下消息。若要解决这些问题，请减少来自同一账户的并发调用的数量。

服务	错误消息
AWS Glue	AWSGlueException: Rate exceeded.（AWSGlueException：数量超出限制。）
AWS KMS	您已超过可以调用 KMS 的数量。请降低您的调用频率。
AWS Lambda	Rate exceeded（数量超出限制） TooManyRequestsException
Amazon S3	AmazonS3Exception: Please reduce your request rate.（AmazonS3Exception：请降低您的请求率。）

有关在使用 Athena 时防止 Amazon S3 节流的方法的信息，请参阅 防止 Amazon S3 节流。

视图

在 Apache Hive shell 中创建的视图在 Athena 中不起作用

由于其实现完全不同，在 Apache Hive shell 中创建的视图与 Athena 不兼容。要解决此问题，请在 Athena 中重新创建视图。

视图已过时；必须重新创建

如果视图下方的表已更改或删除，则可能会收到此错误。解决方法是重新创建视图。有关更多信息，请参阅 AWS 知识中心的 我怎样才能解决 Athena 中的“view is stale; it must be re-created（视图已过时；必须重新创建）”错误？

工作组

有关工作组问题排查的更多信息，请参阅 工作组故障排除。

其他 资源

以下页面提供了有关对 Amazon Athena 问题进行故障排除的其他信息。

- Athena 错误目录
- 服务限额
- Amazon Athena 中 SQL 查询的注意事项和限制
- 不支持的 DDL
- 表、数据库和列的名称
- Amazon Athena 中的数据类型
- 支持的 SerDes 和数据格式
- Athena 压缩支持
- 保留关键字
- 工作组故障排除

以下 AWS 资源也会有所帮助：

- AWS 知识中心中的 Athena 主题
- AWS re:Post 上的 Amazon Athena 问题
- AWS 大数据博客中的 Athena 文章

故障排除通常需要由专家或多个帮助者进行迭代查询和发现。如果尝试此页面上的建议后仍然遇到问题，请联系 AWS Support [在 AWS Management Console 中，单击 Support（支持）、Support Center（支持中心）] 或使用 Amazon Athena 标签在 AWS re:Post 上提问。

Athena 错误目录

Athena 提供标准化的错误信息，以帮助您了解失败的查询，以及在查询失败发生后采取措施。AthenaError 功能包括一个 ErrorCategory 字段和一个 ErrorType 字段。ErrorCategory 说明导致查询失败的原因是系统错误、用户错误还是其他错误。ErrorType 提供了有关查询失败原因的更详细信息。结合这两个字段中的信息，可以更好地了解相关情况以及导致出现特定错误的原因。

错误类别

下表列出了 Athena 错误类别的值及其含义。

错误类别	来源
1	SYSTEM
2	USER
3	OTHER

错误类型参考

下表列出了 Athena 错误类型的值及其含义。

错误类型	描述
0	查询在此缩放系数耗尽了资源
1	查询在此缩放系数耗尽了资源
2	查询在此缩放系数耗尽了资源
3	查询在此缩放系数耗尽了资源
4	查询在此缩放系数耗尽了资源
5	查询在此缩放系数耗尽了资源
6	查询在此缩放系数耗尽了资源

错误类型	描述
7	查询在此缩放系数耗尽了资源
8	查询在此缩放系数耗尽了资源
100	内部服务错误
200	查询引擎出现内部错误
201	查询引擎出现内部错误
202	查询引擎出现内部错误
203	驱动程序错误
204	元数据仓出现错误
205	查询引擎出现内部错误
206	查询超时
207	查询引擎出现内部错误
208	查询引擎出现内部错误
209	未能取消查询
210	查询超时
211	查询引擎出现内部错误
212	查询引擎出现内部错误
213	查询引擎出现内部错误
214	查询引擎出现内部错误
215	查询引擎出现内部错误
216	查询引擎出现内部错误

错误类型	描述
217	查询引擎出现内部错误
218	查询引擎出现内部错误
219	查询引擎出现内部错误
220	查询引擎出现内部错误
221	查询引擎出现内部错误
222	查询引擎出现内部错误
223	查询引擎出现内部错误
224	查询引擎出现内部错误
225	查询引擎出现内部错误
226	查询引擎出现内部错误
227	查询引擎出现内部错误
228	查询引擎出现内部错误
229	查询引擎出现内部错误
230	查询引擎出现内部错误
231	查询引擎出现内部错误
232	查询引擎出现内部错误
233	Iceberg 错误
234	Lake Formation 错误
235	查询引擎出现内部错误
236	查询引擎出现内部错误

错误类型	描述
237	序列化错误
238	未能将元数据上载到 Amazon S3
239	常规持久性错误
240	未能提交查询
300	内部服务错误
301	内部服务错误
302	内部服务错误
303	内部服务错误
400	内部服务错误
401	未能将查询结果写入 Amazon S3
402	未能将查询结果写入 Amazon S3
1000	用户错误
1001	数据错误
1002	数据错误
1003	DDL 任务失败
1004	Schema 错误
1005	序列化错误
1006	语法错误
1007	数据错误
1008	查询被拒绝

错误类型	描述
1009	查询失败
1010	内部服务错误
1011	查询被用户取消
1012	查询引擎出现内部错误
1013	查询引擎出现内部错误
1014	查询被用户取消
1100	提供的参数无效
1101	提供的属性无效
1102	查询引擎出现内部错误
1103	提供的表属性无效
1104	查询引擎出现内部错误
1105	查询引擎出现内部错误
1106	提供的函数参数无效
1107	视图无效
1108	未能注册函数
1109	未找到提供的 Amazon S3 路径
1110	提供的表或视图不存在
1200	查询不受支持
1201	提供的解码器不受支持
1202	查询类型不受支持

错误类型	描述
1300	常规未找到错误
1301	常规未找到实体
1302	未找到文件
1303	未找到提供的函数或函数实现
1304	查询引擎出现内部错误
1305	查询引擎出现内部错误
1306	未找到 Amazon S3 存储桶
1307	未找到选定的引擎
1308	查询引擎出现内部错误
1400	节流错误
1401	由于 AWS Glue 节流，查询失败
1402	由于 AWS Glue 中的表版本太多，查询失败
1403	由于 Amazon S3 节流，查询失败
1404	由于 Amazon Athena 节流，查询失败
1405	由于 Amazon Athena 节流，查询失败
1406	由于 Amazon Athena 节流，查询失败
1500	权限错误
1501	Amazon S3 权限错误
1602	已超过预留容量限制。容量不足，无法执行此查询。
1700	由于 Lake Formation 内部异常，查询失败

错误类型	描述
1701	由于 AWS Glue 内部异常，查询失败
9999	内部服务错误

代码示例

本主题中的示例使用适用于 Java 2.x 的 SDK 作为编写 Athena 应用程序的起始点。

> **Note**
>
> 有关使用其他特定语言的 AWS SDK 编写 Athena 的信息，请参阅以下资源：
>
> - 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`)

有关运行本部分中的 Java 代码示例的更多信息，请参阅 GitHub 上 AWS 代码示例存储库上的 Amazon Athena Java 自述文件。有关 Athena 的 Java 编程参考，请参阅 AWS SDK for Java 2.x 中的 AthenaClient。

- Java 代码示例
  - 常量
  - 创建客户端以访问 Athena
  - 使用查询执行
    - 开始查询执行
    - 停止查询执行

        - 列出查询执行
    - 使用命名查询
        - 创建命名查询
        - 删除命名查询
        - 列出查询执行

> **Note**
>
> 这些示例对字符串使用常量（例如，ATHENA_SAMPLE_QUERY），它们是在 ExampleConstants.java 类声明中定义的。使用您自己的字符串或定义常量来替换这些常量。

常量

ExampleConstants.java 类演示了如何在 Athena 中查询由开始使用教程创建的表。

```
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

}
```

创建客户端以访问 Athena

`AthenaClientFactory.java` 类显示如何创建和配置 Amazon Athena 客户端。

```
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();
    }
}
```

开始查询执行

`StartQueryExample` 显示如何向 Athena 提交查询以供执行，等待结果可用，然后处理结果。

```
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());
            }
        }
    }
}
```

停止查询执行

StopQueryExecutionExample 运行示例查询，立即停止查询，并检查查询的状态以确保它已被取消。

```
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;

    }
}
```

列出查询执行

ListQueryExecutionsExample 显示如何获取查询执行 ID 的列表。

```
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);
        }
    }
}
```

创建命名查询

CreateNamedQueryExample 显示如何创建命名查询。

```
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);
        }
    }
}
```

删除命名查询

DeleteNamedQueryExample 显示如何使用命名查询 ID 删除命名查询。

```
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;
    }
}
```

列出命名查询

ListNamedQueryExample 显示如何获取命名查询 ID 的列表。

```
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);
        }
    }
}
```

在 Amazon Athena 中使用 Apache Spark

Amazon Athena 让您可以轻松使用 Apache Spark 以交互方式运行数据分析和探索，无需规划、配置或管理资源。在 Athena 上运行 Apache Spark 应用程序意味着，无需额外配置即可提交 Spark 代码进行处理和直接接收结果。您可以使用 Amazon Athena 控制台中简化的笔记本体验，以通过 Python 或 Athena notebook API 开发 Apache Spark 应用程序。Amazon Athena 上的 Apache Spark 无服务器，可通过提供即时计算实现自动按需扩展，从而满足不断变化的数据卷和处理要求。

Amazon Athena 提供以下功能：

- 控制台用法 – 从 Amazon Athena 控制台提交 Spark 应用程序。
- 脚本编写 – 在 Python 中以交互方式快速构建和调试 Apache Spark 应用程序。
- 动态扩展 – Amazon Athena 会自动确定运行任务所需的计算和内存资源，并不断相应地将这些资源扩展到您指定的最大值。这种动态扩展可以在不影响速度的情况下降低成本。
- 笔记本体验 – 使用 Athena 笔记本编辑器通过熟悉的界面创建、编辑和运行计算。Athena 笔记本与 Jupyter notebook 兼容，并且包含按顺序执行计算的单元格列表。单元格内容可以包括代码、文本、Markdown、数学运算、绘图和富媒体。

有关其他信息，请参阅 AWS 大数据博客中的使用 Amazon Athena for Apache Spark 探索您的数据湖。

注意事项和限制

- 目前，Amazon Athena for Apache Spark 可在以下 AWS 区域 中使用：
  - 亚太地区（孟买）
  - 亚太地区（新加坡）
  - 亚太地区（悉尼）
  - 亚太地区（东京）
  - 欧洲地区（法兰克福）
  - 欧洲（爱尔兰）
  - 美国东部（弗吉尼亚州北部）
  - 美国东部（俄亥俄州）
  - US West（Oregon）

- 不支持 AWS Lake Formation。
- 不支持使用分区投影的表。
- 启用 Apache Spark 的工作组可以使用 Athena 笔记本编辑器，但不能使用 Athena 查询编辑器。只有 Athena SQL 工作组可以使用 Athena 查询编辑器。
- 不支持跨引擎视图查询。Athena for Spark 无法查询 Athena SQL 创建的视图。由于这两个引擎的视图实现方式不同，因此它们与跨引擎使用不兼容。
- 不支持 MLlib（Apache Spark 机器学习库）和 `pyspark.ml` 包。有关支持的 Python 库列表，请参阅 预安装的 Python 库列表。
- 目前，Athena for Spark 会话不支持 `pip install`。
- 每个笔记本电脑实例只允许一个活动会话。
- 当多个用户使用控制台打开工作组中的一个现有会话时，他们将访问同一个笔记本电脑实例。为避免混淆，请仅打开您自己创建的会话。
- 可能与 Amazon Athena 配合使用的 Apache Spark 应用程序的托管域（例如 `analytics-gateway.us-east-1.amazonaws.com`）已注册到互联网公共后缀列表（PSL）。如果需要在域中设置敏感 Cookie，建议您使用带有 `__Host-` 前缀的 Cookie 来帮助保护自己的域免受跨站点请求伪造（CSRF）攻击。要了解更多信息，请参阅 Mozilla.org 开发人员文档中的 Set-Cookie 页面。
- 有关对 Athena 中 Spark 笔记本、会话和工作组问题进行故障排除的信息，请参阅 对 Athena for Spark 进行故障排除。

Amazon Athena 上的 Apache Spark 入门

要开始在 Amazon Athena 上使用 Apache Spark，您必须先创建一个支持 Spark 的工作组。切换到工作组后，您可以创建笔记本或打开现有笔记本。打开 Athena 中的笔记本后，会自动为其启动一个新会话，您可以直接在 Athena 笔记本编辑器中进行处理。

> **Note**
>
> 在尝试创建笔记本之前，请务必创建启用 Spark 的工作组。

在 Athena 中创建启用 Spark 的工作组

您可以在 Athena 中使用工作组对用户、团队、应用程序或工作负载进行分组，并跟踪成本。要在 Amazon Athena 中使用 Apache Spark，您需要创建一个使用 Spark 引擎的 Amazon Athena 工作组。

> **Note**
>
> 启用 Apache Spark 的工作组可以使用 Athena 笔记本编辑器，但不能使用 Athena 查询编辑器。只有 Athena SQL 工作组可以使用 Athena 查询编辑器。

在 Athena 中创建启用 Spark 的工作组

1. 打开 Athena 控制台，网址为 https://console.aws.amazon.com/athena/。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。

3. 在导航窗格中，选择 Workgroups（工作组）。
4. 在 Workgroups（工作组）页面中，选择 Create workgroup（创建工作组）。
5. 在 Workgroup name（工作组名称）中，输入您的 Apache Spark 工作组名称。
6. （可选）对于 Description（描述），输入工作组的描述。
7. 对于 Analytics engine（分析引擎），选择 Apache Spark。

> **Note**
>
> 创建工作组后，无法更改工作组的分析引擎类型。例如，无法将 Athena 引擎版本 3 工作组更改为 PySpark 引擎版本 3 工作组。

8. 在本教程中，请选择 Turn on example notebook（打开示例笔记本）。此可选功能将向工作组添加名为 `example-notebook-random_string` 的示例笔记本，并添加笔记本用于在账户中创

建、显示和删除特定数据库和表的 AWS Glue 相关权限，并在 Amazon S3 中添加示例数据集的读取权限。要查看添加的权限，请选择 View additional permissions details（查看其他权限详细信息）。

> **Note**
>
> 运行示例笔记本可能会产生一些额外费用。

9. 对于 Additional configuration（其他配置），执行以下操作之一：
   - 使用 Use defaults（使用默认值）设置。此选项为默认选项，可帮助您开始使用启用 Spark 的工作组。使用此选项，Athena 会在 Amazon S3 中为您创建 IAM 角色和计算结果位置。IAM 角色的名称和要创建的 S3 存储桶位置将在 Additional configurations（其他配置）标题下方的复选框中显示。
   - 禁用 Use defaults（使用默认值）设置，然后继续执行 指定您自己的工作组配置 一节中的步骤，以手动配置工作组。
10. （可选）Tags（标签）– 使用此选项可向工作组添加标签。有关更多信息，请参阅 为 Athena 资源添加标签。
11. 选择 Create workgroup (创建工作组)。系统将显示一条消息通知您工作组已成功创建，并且该工作组将在工作组列表中显示。

指定您自己的工作组配置

如果要为笔记本指定自己的 IAM 角色以及计算结果位置，请按照本节中的步骤操作。如果您为 Additional configurations（其他配置）选项选择了 Use defaults（使用默认值），请跳过本节并直接转至 打开笔记本资源管理器并切换工作组 。

以下过程假定您已完成上一节中 To create a Spark enabled workgroup in Athena（在 Athena 中创建启用 Spark 的工作组）过程的步骤 1 至 9。

指定您自己的工作组配置

1. 如果要创建或使用您自己的 IAM 角色或配置笔记本加密，请扩展 IAM role configuration（IAM 角色配置）。
   - 对于 Service Role（服务角色），选择以下选项之一：
     - 创建服务角色 – 选择此选项，Athena 将为您创建服务角色。要查看角色授予的权限，请选择 View permission details（查看权限详细信息）。

- 选择现有服务角色 – 从下拉菜单中选择一个现有角色。所选角色必须包含第一个选项中的权限。有关启用笔记本的工作组的权限的更多信息，请参阅 对启用 Spark 的工作组进行故障排除。
- 对于 Notebook and calculation code encryption key management（笔记本和计算代码加密密钥管理），选择下列选项之一：
  - 归 Amazon Athena 所有 – AWS KMS 密钥由 Amazon Athena 拥有和管理。使用此密钥不会产生额外的费用。
  - 对称密钥存储在账户中，由您拥有和管理 – 对于此选项，执行下列操作之一：
    - 要使用现有密钥，请使用搜索框选择 AWS KMS 或输入密钥 ARN。
    - 要在 AWS KMS 控制台中创建密钥，选择创建 AWS KMS 密钥。执行角色必须具有使用您创建的密钥的权限。

> **Important**
>
> 更改工作组的 AWS KMS key 后，在更新之前托管的笔记本仍会引用旧的 KMS 密钥，而更新之后托管的笔记本将使用新的 KMS 密钥。要更新旧笔记本以引用新的 KMS 密钥，请导出每个旧笔记本然后导入。如果在将旧笔记本引用更新为新 KMS 密钥之前删除了旧 KMS 密钥，则旧笔记本将无法解密且无法恢复。
>
> 此行为也适用于别名的更新，别名是 KMS 密钥的易记名称。更新 KMS 密钥别名以指向新的 KMS 密钥后，在别名更新之前托管的笔记本仍会引用旧的 KMS 密钥，而在别名更新之后托管的笔记本会使用新的 KMS 密钥。在更新 KMS 密钥或别名之前，请考虑以下几点。

2. 要指定您自己的计算结果设置，请展开 Calculation result settings（计算结果设置），然后从以下选项中进行选择。

- 创建新的 S3 存储桶 – 此选项会在账户中为计算结果创建 Amazon S3 存储桶。存储桶名称的格式为 *account_id*-*region*-athena-results-bucket-*alphanumeric_id*，使用的设置为：禁用 ACL、阻止公有访问、禁用版本控制和强制存储桶拥有者。
- 选择现有 S3 位置 – 对于此选项，请执行以下操作：
  - 在搜索框中输入一个现有位置的 S3 路径，或选择 Browse S3（浏览 S3）以从列表中选择一个存储桶。

> **Note**
>
> 在 Amazon S3 中选择现有位置时，请勿向该位置附加正斜杠 (/)。这样操作会导致指向 calculation details page（计算详细信息页面）上计算结果位置的链接指向错误的目录。如果出现这种情况，请编辑工作组的结果位置以删除尾部正斜杠。

- （可选）选择 View（查看）以打开 Amazon S3 控制台的 Buckets（存储桶）页面，您可以在其中查看有关所选现有存储桶的更多信息。
- （可选）在 Expected bucket owner（预期存储桶拥有者）中，输入您希望其成为查询结果输出位置存储桶拥有者的 AWS 账户 ID。建议您尽可能选择此选项作为额外的安全措施。如果存储桶拥有者的账户 ID 与您指定的 ID 不匹配，则输出到存储桶的尝试将会失败。有关更多信息，请参阅《Amazon S3 用户指南》中的使用存储桶拥有者条件验证存储桶所有权
- （可选）如果您的计算结果位置属于其他账户所有，并且您希望向其他账户授予对查询结果的完全控制权，请选择 Assign bucket owner full control over query results（为存储桶拥有者分配对查询结果的完全控制权）。

3. （可选）选择 Encrypt calculation results（加密计算结果），然后选择以下选项之一：
   - SSE_S3 – 这是 S3 托管的服务器端加密密钥。
   - SSE_KMS – 您提供的密钥。对于选择 AWS KMS 密钥，可以选择以下选项之一：
     - 使用 AWS 拥有的密钥 – 使用 AWS 拥有并为您管理的密钥。
     - 选择其他 AWS KMS 密钥（高级）– 选择或创建密钥。
       - 要使用现有密钥，请使用搜索框选择 AWS KMS 或输入密钥 ARN。
       - 要在 KMS 控制台中创建密钥，选择创建 AWS KMS 密钥。在 KMS 控制台中完成密钥创建后，返回 Athena 控制台中的创建工作组页面，然后使用选择 AWS KMS 密钥或输入 ARN 搜索框选择您刚刚创建的密钥。
4. （可选）Other settings（其他设置）– 扩展此选项可启用或禁用工作组的 Publish CloudWatch metrics（发布 CloudWatch 指标）选项。在默认情况下，该字段会被选中。有关更多信息，请参阅 使用 CloudWatch 指标监控 Apache Spark 计算。
5. （可选）Tags（标签）– 使用此选项可向工作组添加标签。有关更多信息，请参阅 为 Athena 资源添加标签。
6. 选择 Create workgroup (创建工作组)。系统将显示一条消息通知您工作组已成功创建，并且该工作组将在工作组列表中显示。

打开笔记本资源管理器并切换工作组

您必须先切换到该工作组，然后才能使用刚刚创建的启用 Spark 的工作组。要切换启用 Spark 的工作组，可以使用“笔记本资源管理器”或“笔记本编辑器”中的 Workgroup（工作组）选项。

> **Note**
>
> 开始之前，请确保您的浏览器不会阻止第三方 Cookie。浏览器的默认设置，或用户启用的设置阻止第三方 Cookie 时，都将阻止笔记本启动。有关管理 Cookie 的更多信息，请参阅：
>
> - Chrome
> - Firefox
> - Safari

打开笔记本资源管理器并切换工作组

1. 在导航窗格中，选择 Notebook explorer（笔记本资源管理器）。
2. 使用控制台右上角的 Workgroup（工作组）选项来选择您创建的启用 Spark 的工作组。示例笔记本将在笔记本列表中显示。

   您可以通过以下方式使用笔记本资源管理器：

   - 选择笔记本的链接名称，在新会话中打开该笔记本。
   - 要重命名、删除或导出笔记本，请使用 Actions（操作）菜单。
   - 要导入笔记本文件，请选择 Import file（导入文件）。
   - 要创建笔记本，请选择 Create notebook（创建笔记本）。

运行示例笔记本

示例笔记本从公开的纽约出租车旅行数据集查询数据。该笔记本包含展示了如何使用 Spark DataFrames、Spark SQL 和 AWS Glue Data Catalog 的示例。

运行示例笔记本

1. 在“笔记本资源管理器”中，选择示例笔记本的链接名称。

这将通过默认参数启动笔记本会话，并在笔记本编辑器中打开笔记本。系统将显示一条消息，通知您已使用默认参数（最大 DPU 数为 20）启动新的 Apache Spark 会话。

2. 要按顺序运行单元格并查看结果，请为笔记本中的每个单元格选择一次 Run（运行）按钮。

   - 向下滚动以查看结果以及新单元格。
   - 对于包含计算结果的单元格，进度栏会显示完成百分比、所用时间和剩余时间。
   - 示例笔记本将在您的账户中创建示例数据库和表。作为清除步骤，最后一个单元格会将其删除。

> **Note**
>
> 如果您在示例笔记本中更改文件夹、表或数据库名称，请确保这些更改会反映在您使用的 IAM 角色中。否则，笔记本可能因权限不足而无法运行。

编辑会话详细信息

在启动笔记本会话后，您可以编辑会话详细信息，例如表格式、加密、会话空闲超时以及要使用的数据处理单元（DPU）的最大并发数。DPU 是对处理能力的相对度量，它由 4 个 vCPU 的计算容量和 16GB 内存组成。

编辑会话详细信息

1. 在笔记本编辑器中，从右上角的 Session（会话）菜单中选择 Edit session（编辑会话）。
2. 在编辑会话详细信息对话框的 Spark 属性部分中，选择或输入以下选项的值：

   - 其他表格式 - 选择 Linux Foundation Delta Lake、Apache Hudi、Apache Iceberg 或自定义。
     - 对于 Delta、Hudi 或 Iceberg 表选项，对应表格式所需的表属性将自动在在表中编辑和在 JSON 中编辑选项中提供。有关使用这些表格式的更多信息，请参阅 在 Amazon Athena for Apache Spark 中使用非 Hive 表格式。
     - 要为自定义或其他表类型添加或移除表属性，使用在表中编辑和在 JSON 中编辑选项。
     - 对于在表中编辑选项，选择添加属性添加属性，或者选择移除移除属性。要输入属性名称及其值，使用键和值框。
     - 对于在 JSON 中编辑选项，使用 JSON 文本编辑器直接编辑配置。
       - 要将 JSON 文本复制到剪贴板，选择复制。
       - 要从 JSON 编辑器中移除所有文本，选择清除。

- 要配置换行或为 JSON 编辑器选择颜色主题，选择设置（齿轮）图标。
- 打开 Spark 加密 - 选择此选项可加密写入磁盘并通过 Spark 网络节点发送的数据。有关更多信息，请参阅 启用 Apache Spark 加密。

3. 在编辑参数部分中，选择或输入以下选项的值：

   - Session idle timeout（会话空闲超时）– 选择或输入一个介于 1 和 480 分钟之间的值。默认值为 20。
   - Coordinator size（协调器大小）– 协调器是一种特殊的执行程序，负责编排处理工作并管理笔记本会话中的其他执行程序。当前，1 DPU 是默认且唯一可能的值。
   - Executor size（执行程序大小）– 执行程序是笔记本会话可以向 Athena 请求的最小计算单位。当前，1 DPU 是默认且唯一可能的值。
   - Max concurrent value（最大并发值）– 可以同时运行的最大 DPU 数。默认值为 20，最小值为 3，最大值为 60。增加此值不会自动分配其他资源，但 Athena 会在计算负载需要以及资源可用时尝试分配指定的最大值。

4. 选择保存。
5. 出现 Confirm edit（确认编辑）提示时，选择 Confirm（确认）。

   Athena 会保存笔记本并使用您指定的参数开始新会话。笔记本编辑器中的横幅将通知您已通过修改后的参数开始新会话。

> **Note**
>
> Athena 会记住笔记本的会话设置。如果您编辑会话的参数然后终止会话，Athena 会在您下次启动笔记本会话时使用您配置的会话参数。

查看会话和计算的详细信息

运行笔记本后，您可以查看会话和计算的详细信息。

查看会话和计算的详细信息

1. 在右上角的 Session（会话）菜单中，选择 View details（查看详细信息）。

   - Current session（当前会话）选项卡会显示有关当前会话的信息，包括会话 ID、创建时间、状态和工作组。

- History（历史记录）选项卡会列出先前会话的会话 ID。要查看先前会话的详细信息，请选择 History（历史记录）选项卡，然后在列表中选择会话 ID。
- Calculations（计算）部分将在会话中运行的计算列表显示。

2. 要查看计算的详细信息，请选择计算 ID。
3. 在 Calculation details（计算详细信息）页面上，可以执行以下操作：

- 要查看计算代码，请参阅 Code（代码）部分。
- 要查看计算结果，请选择 Results（结果）选项卡。
- 要以文本格式下载显示的结果，请选择 Download results（下载结果）。
- 要在 Amazon S3 中查看有关计算结果的信息，请选择 View in S3（在 S3 中查看）。

终止会话

终止笔记本会话

1. 在笔记本编辑器中，从右上角的 Session（会话）菜单中选择 Terminate（终止）。
2. 出现 Confirm session termination（确认终止会话）提示时，选择 Confirm（确认）。笔记本已保存，您将返回到笔记本编辑器。

> **Note**
>
> 在笔记本编辑器中关闭笔记本选项卡并不会终止活动笔记本的会话。要确保会话已终止，请依次选择 Session（会话）、Terminate（终止）选项。

创建您自己的笔记本

创建启用 Spark 的 Athena 工作组后，您可以创建自己的笔记本。

创建笔记本

1. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。
2. 在 Athena 控制台导航窗格中，选择 Notebook explorer（笔记本资源管理器）或 Notebook editor（笔记本编辑器）。
3. 请执行以下操作之一：

- 在 Notebook explorer（笔记本资源管理器）中，选择 Create notebook（创建笔记本）。
- 在 Notebook editor（笔记本编辑器）中，选择 Create notebook（创建笔记本），或选择加号图标（+）添加笔记本。

4. 在 Create notebook（创建笔记本）对话框中，在 Notebook name（笔记本名称）中输入名称。
5. （可选）展开 Spark 属性，然后为以下选项选择或输入值：

- 其他表格式 - 选择 Linux Foundation Delta Lake、Apache Hudi、Apache Iceberg 或自定义。
  - 对于 Delta、Hudi 或 Iceberg 表选项，对应表格式所需的表属性将自动在在表中编辑和在 JSON 中编辑选项中提供。有关使用这些表格式的更多信息，请参阅 在 Amazon Athena for Apache Spark 中使用非 Hive 表格式。
  - 要为自定义或其他表类型添加或移除表属性，使用在表中编辑和在 JSON 中编辑选项。
  - 对于在表中编辑选项，选择添加属性添加属性，或者选择移除移除属性。要输入属性名称及其值，使用键和值框。
  - 对于在 JSON 中编辑选项，使用 JSON 文本编辑器直接编辑配置。
    - 要将 JSON 文本复制到剪贴板，选择复制。
    - 要从 JSON 编辑器中移除所有文本，选择清除。
    - 要配置换行或为 JSON 编辑器选择颜色主题，选择设置（齿轮）图标。
- 打开 Spark 加密 - 选择此选项可加密写入磁盘并通过 Spark 网络节点发送的数据。有关更多信息，请参阅 启用 Apache Spark 加密。

6. （可选）展开 Session parameters（会话参数），然后为以下选项选择或输入值：

- Session idle timeout（会话空闲超时）– 选择或输入一个介于 1 和 480 分钟之间的值。默认值为 20。
- Coordinator size（协调器大小）– 协调器是一种特殊的执行程序，负责编排处理工作并管理笔记本会话中的其他执行程序。当前，1 DPU 是默认且唯一可能的值。DPU（数据处理单元）是对处理能力的相对度量，它由 4 个 vCPU 的计算容量和 16 GB 的内存组成。
- Executor size（执行程序大小）– 执行程序是笔记本会话可以向 Athena 请求的最小计算单位。当前，1 DPU 是默认且唯一可能的值。
- Max concurrent value（最大并发值）– 可以同时运行的最大 DPU 数。默认值为 20，最大值为 60。增加此值不会自动分配其他资源，但 Athena 会在计算负载需要以及资源可用时尝试分配指定的最大值。

7. 选择创建。笔记本将在笔记本编辑器的新会话中打开。

打开先前创建的笔记本

打开先前创建的笔记本

1. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。
2. 在 Athena 控制台导航窗格中，选择 Notebook editor（笔记本编辑器）或 Notebook explorer（笔记本资源管理器）。
3. 请执行以下操作之一：

   - 在 Notebook editor（笔记本编辑器）中，在 Recent notebooks（最近使用的笔记本）或 Saved notebooks（保存的笔记本）列表中选择一个笔记本。笔记本将在新会话中打开。
   - 在 Notebook explorer（笔记本资源管理器）中，在列表中选择笔记本的名称。笔记本将在新会话中打开。

有关管理笔记本文件的更多信息，请参阅 管理笔记本文件。

使用笔记本

您可以在 Athena 笔记本资源管理器中管理笔记本，并使用 Athena 笔记本编辑器在会话中对其进行编辑和运行。您可以根据要求为笔记本会话配置 DPU 使用情况。

当您停止笔记本时，即会终止关联会话。将保存所有文件，但声明的变量、函数和类中正在进行的更改会丢失。重新启动笔记本后，Athena 会重新加载笔记本文件，您可以再次运行代码。

会话和计算

每个笔记本都与一个 Python 内核相关联并运行 Python 代码。笔记本可以有一个或多个包含命令的单元格。要运行笔记本中的单元格，首先要为笔记本创建会话。会话会跟踪笔记本的变量和状态。

运行笔记本中的单元格意味着运行当前会话中的计算。计算会推进笔记本的状态，并且可能执行诸如从 Amazon S3 读取或写入其他数据存储等任务。只要会话在运行，计算就会使用和修改为笔记本维护的状态。

如果不再需要状态，可以结束会话。结束会话后，笔记本会保留，但变量和其他状态信息会销毁。如果您需要同时处理多个项目，可以为每个项目创建一个会话，这些会话相互独立。

会话具有专用的计算容量，以 DPU 为单位。创建会话时，可以为会话分配多个 DPU。不同的会话可能具有不同的功能，具体取决于任务的要求。

使用 Athena 笔记本编辑器

Athena 笔记本编辑器是一个用于编写和运行代码的交互式环境。以下各节介绍该环境的各项功能。

命令模式与编辑模式

笔记本编辑器具有模态用户界面：一种用于在单元格中输入文本的编辑模式，以及一种用于向编辑器本身发出命令（例如复制、粘贴或运行）的命令模式。

要使用编辑模式和命令模式，您可以执行以下任务：

- 要进入编辑模式，请按 **ENTER** 或选择一个单元格。当单元格处于编辑模式时，单元格的左边距为绿色。
- 要进入命令模式，请按 **ESC**，或在单元格以外的位置单击。请注意，命令通常仅适用于当前选定的单元格，而不适用于所有单元格。当编辑器处于命令模式时，单元格的左边距为蓝色。
- 在命令模式下，您可以使用键盘快捷键和编辑器上方的菜单，但不能在单个单元格中输入文本。
- 要选择单元格，请选择该单元格。
- 要选择所有单元格，请按 **Ctrl+A**（Windows）或 **Cmd+A**（Mac）。

笔记本编辑器菜单

笔记本编辑器顶部菜单中的图标提供以下选项：

- 保存 – 保存笔记本的当前状态。
- 在下方插入单元格 – 在当前选定的单元格下方添加一个新的（空）单元格。
- 剪切所选单元格 – 将选定单元格从其当前位置移除并将其复制到内存。
- 复制所选单元格 – 将所选单元格复制到内存。
- 在下方粘贴单元格 – 将复制的单元格粘贴到当前单元格下方。
- 向上移动所选单元格 – 将当前单元格移至上方单元格上方。
- 向下移动所选单元格 – 将当前单元格移至下方单元格下方。
- 运行 – 运行当前（选定的）单元格。输出将立即在当前单元格下方显示。
- 全部运行 – 运行笔记本中的所有单元格。每个单元格的输出将立即在单元格下方显示。
- 停止（中断内核） – 通过中断内核来停止当前笔记本。
- 格式选项 – 选择单元格格式，可以是以下格式之一：
  - 代码 – 用于 Python 代码（默认）。

  - Markdown – 用于以 GitHub 风格的 markdown 格式输入文本。运行单元格以渲染 markdown。
  - Raw NBConvert – 用于以未修改的格式输入文本。标记为 Raw NBConvert 的单元格可通过 Jupyter nbconvert 命令行工具转换为其他格式，例如 HTML。
- 标题 – 用于更改单元格的标题级别。
- 命令面板 – 包含 Jupyter notebook 命令及其键盘快捷键。有关键盘快捷键的更多信息，请参阅本文档后续章节。
- 会话 – 使用此菜单中的选项查看会话的详细信息、编辑会话参数或终止会话。

命令模式键盘快捷键

以下是笔记本编辑器命令模式的一些常见键盘快捷键。按 **ESC** 进入命令模式后，可以使用这些快捷键。要查看编辑器中可用命令的完整列表，请按 **ESC + H**。

键	操作
1 - 6	将单元格类型更改为 markdown 并将标题级别设置为键入的数字
a	在当前单元格上方创建一个单元格
b	在当前单元格下方创建一个单元格
c	将当前单元格复制到内存
d d	删除当前单元格
h	显示键盘快捷键帮助屏幕
j	向下移动一个单元格
k	向上移动一个单元格
m	将当前单元格格式更改为 markdown
r	将当前单元格格式更改为 raw
s	保存笔记本
v	将内存内容粘贴到当前单元格下

键	操作
x	剪切选定的一个或多个单元格
y	将单元格格式更改为代码
z	撤消
Ctrl+Enter	运行当前单元格并进入命令模式
Shift+Enter 或 **Alt+Enter**	运行当前单元格并在输出下方创建一个新单元格，然后在编辑模式下输入新单元格
Space	向下翻页
Shift+Space	向上翻页
Shift + L	切换单元格中行号的可见性

编辑命令模式快捷键

笔记本编辑器可以选择自定义命令模式键盘快捷键。

编辑命令模式快捷键

1. 在笔记本编辑器菜单中选择 Command palette（命令面板）。
2. 在命令面板中选择 Edit command mode keyboard shortcuts（编辑命令模式键盘快捷键）命令。
3. 使用 Edit command mode shortcuts（编辑命令模式快捷键）界面，将所需的命令映射或重新映射到键盘。

   要查看编辑命令模式快捷键的说明，请滚动至 Edit command mode shortcuts（编辑命令模式快捷键）屏幕底部。

有关在 Athena for Apache Spark 中使用魔术命令的信息，请参阅 使用魔术命令。

使用魔术命令

魔术命令，又称魔术，是可在笔记本单元格中运行的特殊命令。例如，`%env` 可在笔记本会话中显示环境变量。Athena 支持 IPython 6.0.3 中的魔术命令函数。

本节显示了 Athena for Apache Spark 中的一些关键魔术命令。

- 要查看 Athena 中的魔术命令列表，在笔记本单元格中运行 **`%lsmagic`** 命令。
- 有关在 Athena 笔记本中使用魔术创建图表的信息，请参阅 用于创建数据图表的魔术命令。
- 有关其他魔术命令的更多信息，请参阅 IPython 文档中的内置魔术命令。

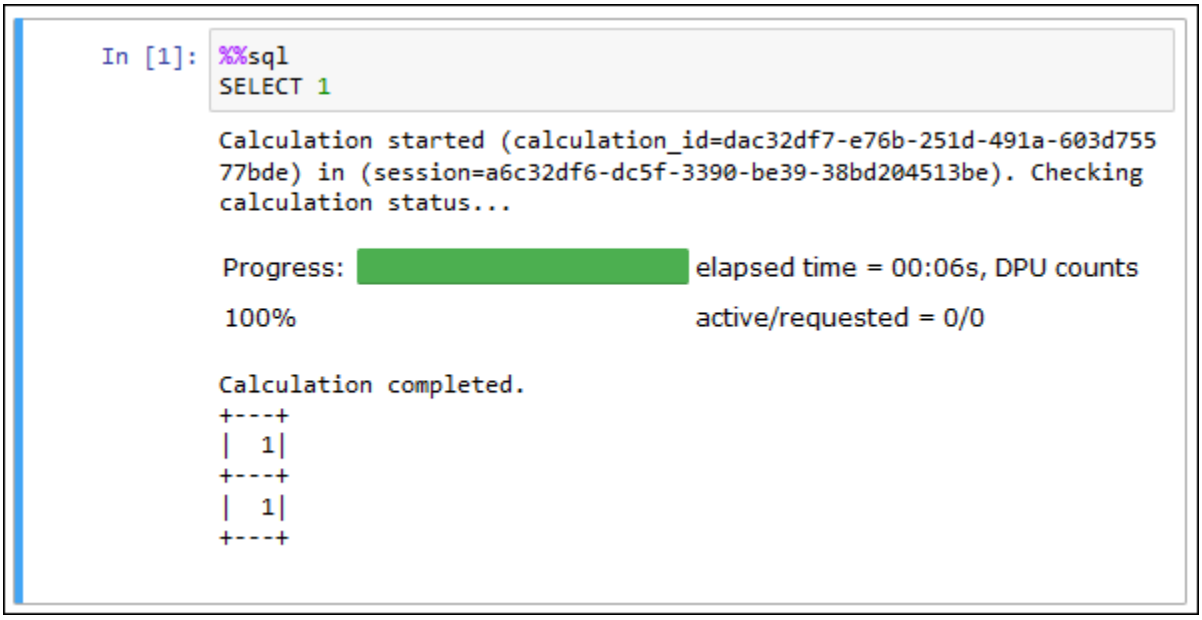
> **Note**
>
> 目前，`%pip` 命令无法执行。这是一个已知问题。

单元格魔术命令

多行书写的魔术命令前有双百分号（`%%`），称为单元格魔术命令函数或单元格魔术命令。

%%sql

此单元格魔术命令允许直接运行 SQL 语句，而无需使用 Spark SQL 语句对其进行装饰。该命令还通过隐式调用 `.show()` 在返回的数据框上显示输出。



%%sql 命令会自动将列输出截断为 20 个字符宽度。目前，无法配置此设置。要绕过此限制，使用以下完整语法并相应地修改 show 方法的参数。

```
spark.sql("""YOUR_SQL""").show(n=number, truncate=number, vertical=bool)
```

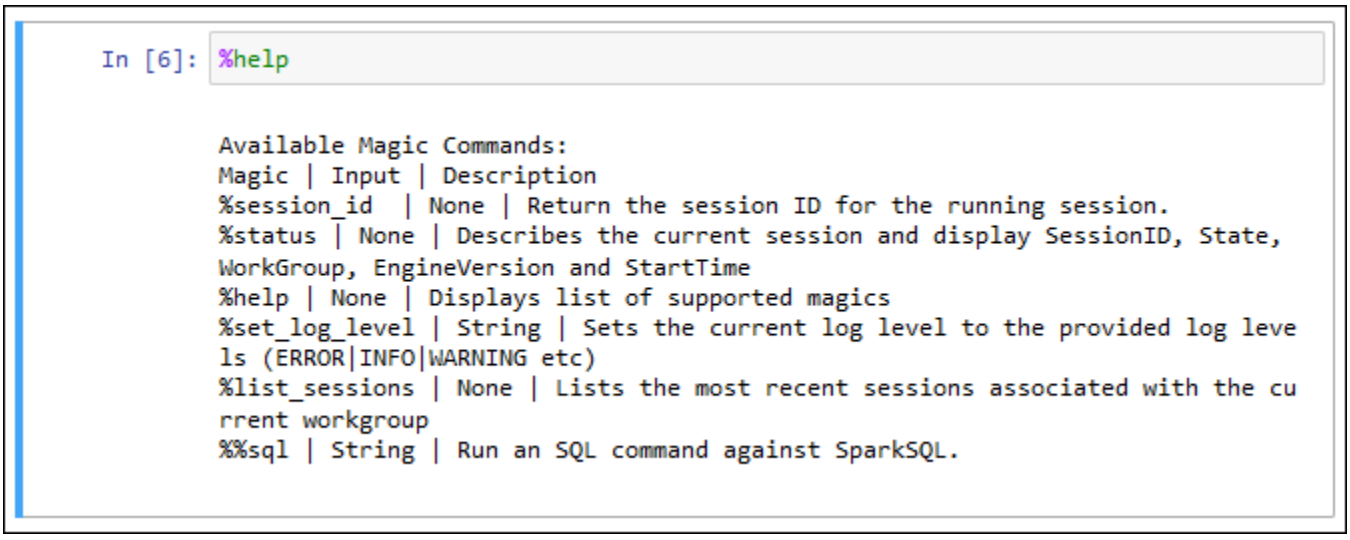
- n int（可选）。要显示的行数。
- 截断 - bool 或 int（可选）- 如果为 true，则截断的字符串长度超过 20 个字符。当设置为大于 1 的数字时，讲按指定长度截断长字符串，并使单元格右对齐。
- 垂直 - bool（可选）。如果为 true，则垂直发送输出行（每列值一行）。

行魔术命令

单行书写的魔术命令前有百分号（%），称为行魔术命令函数或行魔术命令。

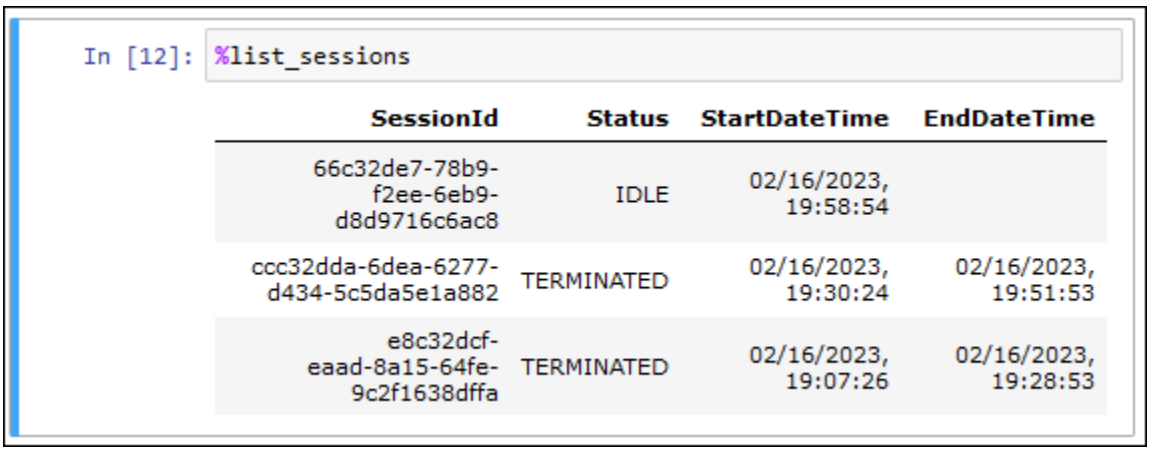
%help

显示可用魔术命令的描述。



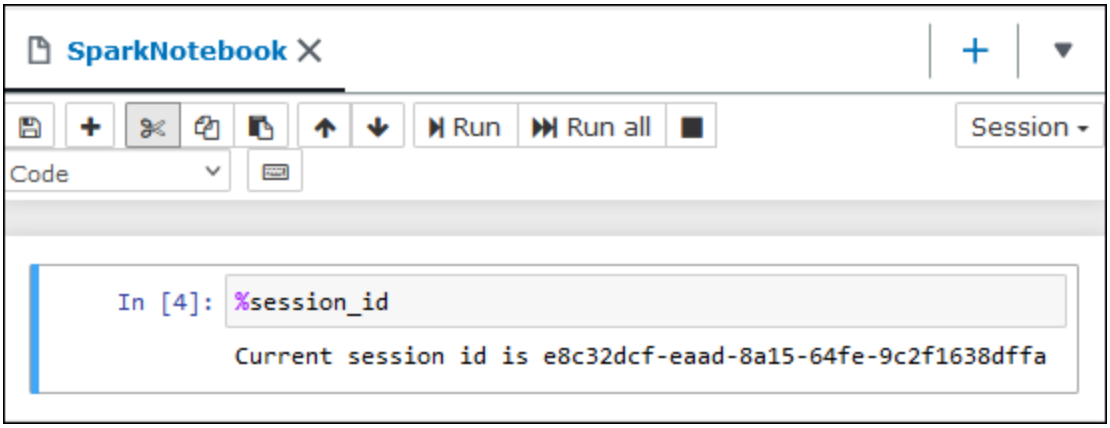
%list_session

列出与笔记本相关的会话。每个会话的信息包含会话 ID、会话状态以及会话的开始和结束日期和时间。



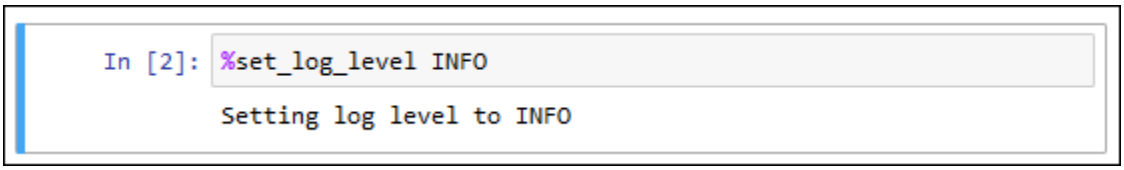
%session_id

检索当前会话 ID。



%set_log_level

设置或重置记录器以使用指定的日志级别。可能的值为 DEBUG、ERROR、FATAL、INFO 和 WARN 或 WARNING。值必须为大写且不得包含在单引号或双引号中。



%status

描述当前会话。输出包含会话 ID、会话状态、工作组名称、PySpark 引擎版本和会话开始时间。此魔术命令需要活动会话才能检索会话详细信息。

状态的可能值如下所示：

CREATING - 会话正在启动，包括获取资源。

CREATED - 会话已启动。

IDLE - 会话能够接受计算。

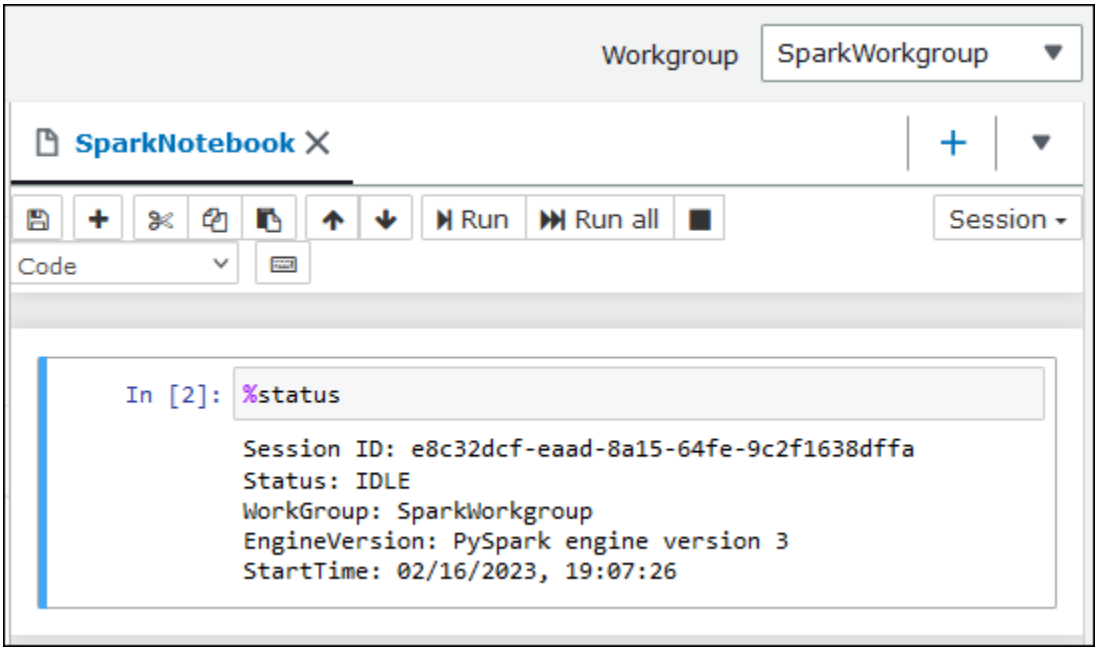
BUSY - 会话正在处理其他任务，无法接受计算。

TERMINATING - 会话正在关闭。

TERMINATED - 会话及其资源不再运行。

DEGRADED - 会话没有运行正常的协调器。

FAILED - 由于故障，会话及其资源不再运行。



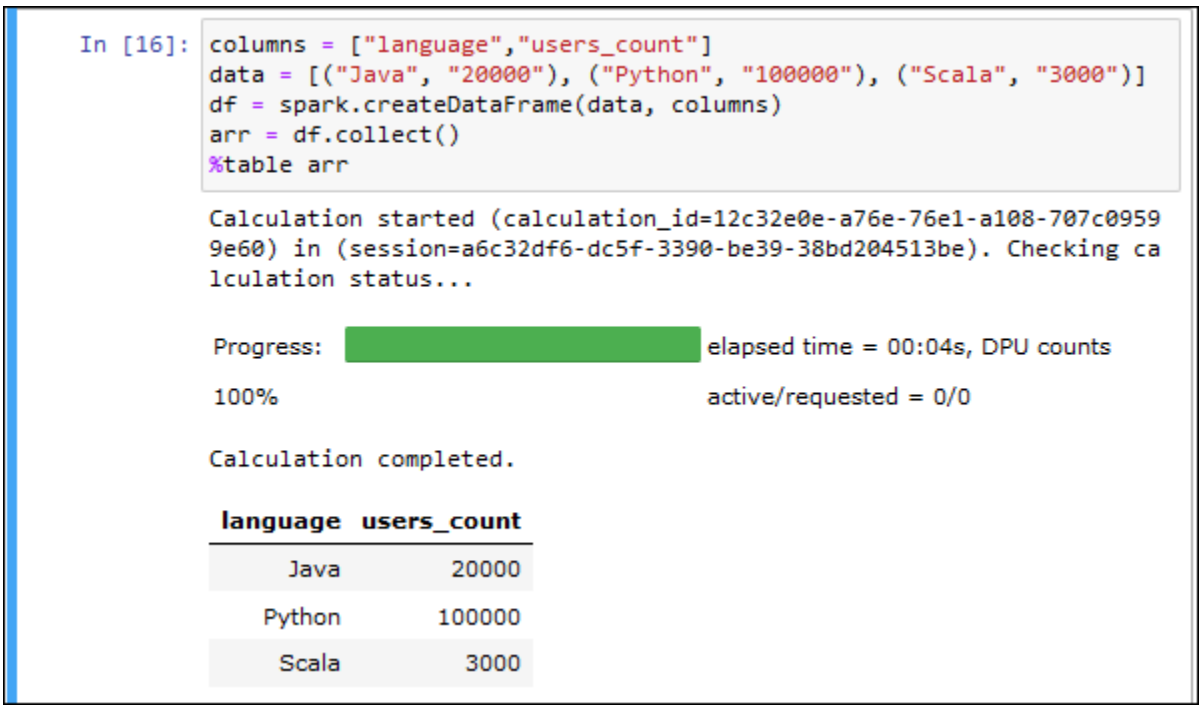
用于创建数据图表的魔术命令

本节中的行魔术专门针对特定类型的数据或与图形库结合使用渲染数据。

%table

可使用 `%table` 魔术命令以表格式显示数据框数据。

以下示例创建了一个包含两列和三行数据的数据框，然后以表格式显示数据。



%matplot

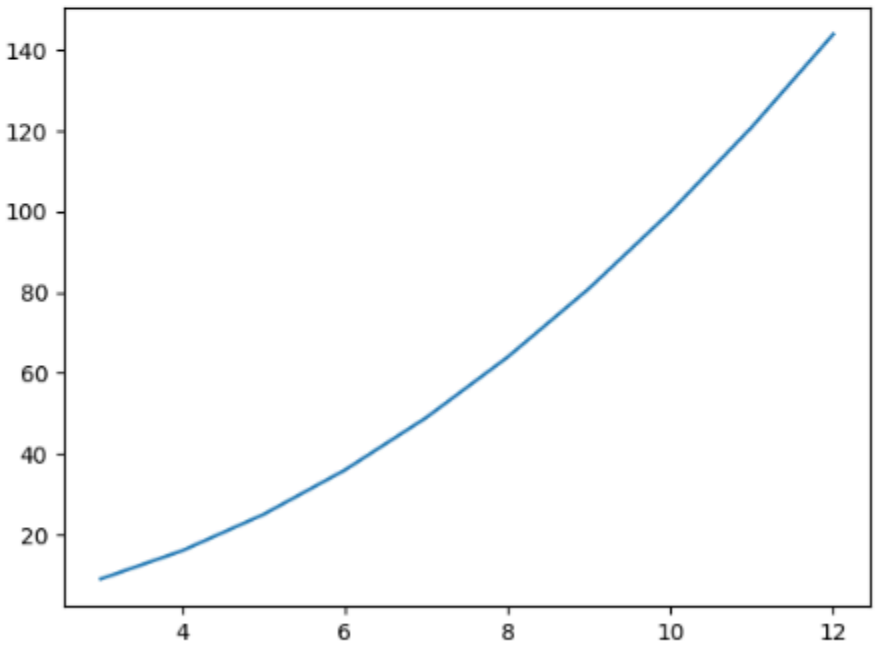
Matplotlib 是一个综合库，用于在 Python 中创建静态、动画演示和交互式可视化。在将 matplotlib 库导入笔记本单元格后，可使用 `%matplot` 魔术命令创建图表。

以下示例导入了 matplotlib 库，创建了一组 x 和 y 坐标，然后使用 `%matplot` 魔术命令创建点图。

```
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>]
```

同时使用 matplotlib 和 seaborn 库

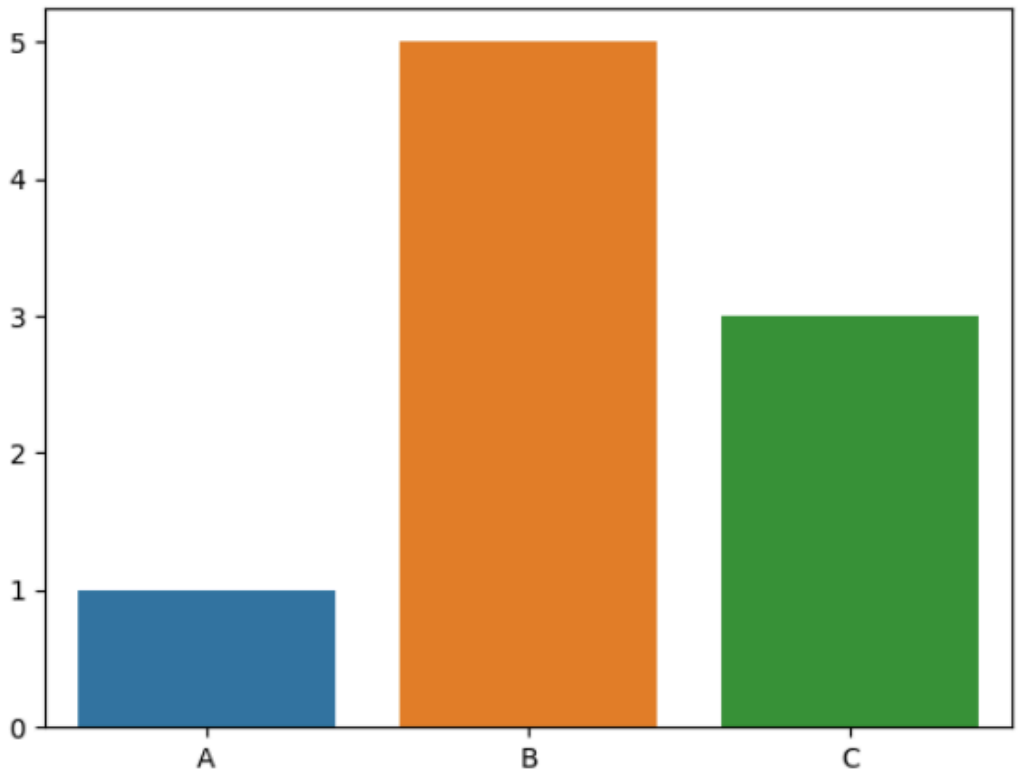
Seaborn 是一个用于在 Python 中创建统计图形的库。该库基于 matplotlib 构建，并与 pandas（Python 数据分析）数据结构紧密集成。也可以使用 `%matplot` 魔术命令渲染 seaborn 数据。

以下示例使用 matplotlib 和 seaborn 库创建简单的条形图。

```
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.
```

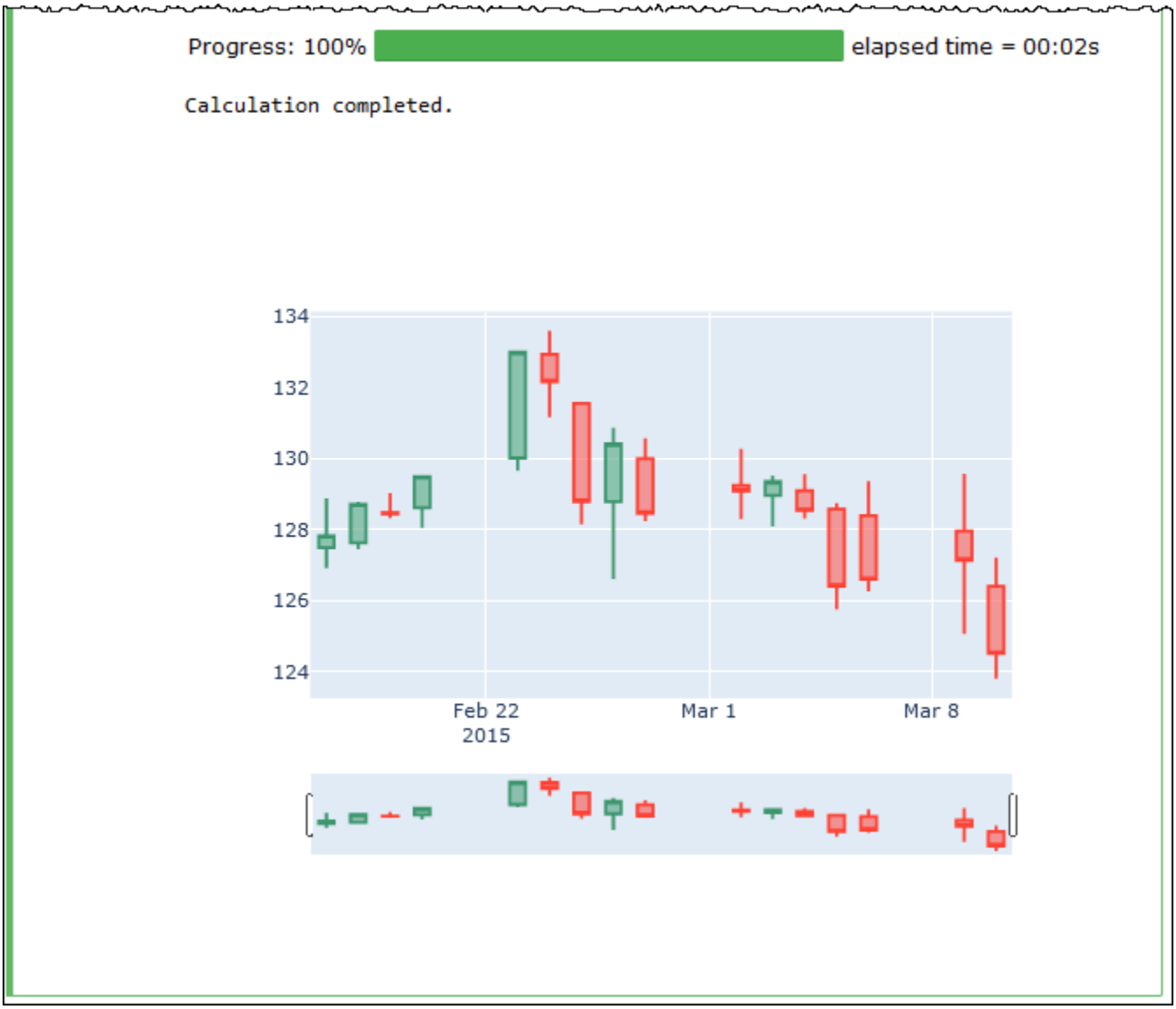


%plotly

Plotly 是一个适用于 Python 的开源图形库，可用于创建交互式图表。可使用 %ploty 魔术命令渲染 ploty 数据。

以下示例对股票价格数据使用 StringIO、plotly 和 pandas 库来创建 2015 年 2 月和 3 月的股票活动图表。

```
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
```



管理笔记本文件

除了使用笔记本资源管理器创建和打开笔记本外，您还可以使用它对笔记本进行重命名、删除、导出或导入，或查看笔记本的会话历史记录。

重命名笔记本

1. 终止要重命名的笔记本的所有活动会话。必须先终止笔记本的活动会话，然后才能重命名笔记本。
2. 打开 Notebook explorer（笔记本资源管理器）。
3. 在 Notebooks（笔记本）列表中，选择要重命名的笔记本的选项按钮。
4. 从 Actions（操作）菜单中，选择 Rename（重命名）。

5. 出现 Rename notebook（重命名笔记本）提示时，输入新名称，然后选择 Save（保存）。新的笔记本名称将在笔记本列表中显示。

删除笔记本

1. 终止要删除的笔记本的所有活动会话。必须先终止笔记本的活动会话，然后才能删除笔记本。
2. 打开 Notebook explorer（笔记本资源管理器）。
3. 在 Notebooks（笔记本）列表中，选择要删除的笔记本的选项按钮。
4. 从 Actions 菜单中选择 Delete 。
5. 出现 Delete notebook?（是否删除笔记本？）提示时，输入笔记本名称，然后选择 Delete（删除）以确认删除。笔记本名称已从笔记本列表中删除。

导出笔记本

1. 打开 Notebook explorer（笔记本资源管理器）。
2. 在 Notebooks（笔记本）列表中，选择要导出的笔记本的选项按钮。
3. 从 Actions（操作）菜单中，选择 Export file（导出文件）。

导入笔记本

1. 打开 Notebook explorer（笔记本资源管理器）。
2. 选择 Import File（导入文件）。
3. 浏览到要导入的文件在本地计算机上的位置，然后选择 Open（打开）。导入的笔记本将在笔记本列表中显示。

查看笔记本的会话历史记录

1. 打开 Notebook explorer（笔记本资源管理器）。
2. 在 Notebooks（笔记本）列表中，选择要查看其会话历史记录的笔记本的选项按钮。
3. 从 Actions（操作）菜单中选择 Session history（会话历史记录）。
4. 在 History（历史记录）选项卡上，选择 Session ID（会话 ID）以查看有关会话及其计算的信息。

在 Amazon Athena for Apache Spark 中使用非 Hive 表格式

在 Athena for Spark 中使用会话和笔记本时，除了 Apache Hive 表之外，还可以使用 Linux Foundation Delta Lake、Apache Hudi 和 Apache Iceberg 表。

注意事项和限制

在 Athena for Spark 中使用除 Apache Hive 以外的表格式时，请考虑以下几点：

- 除了 Apache Hive 之外，每个笔记本仅支持一种表格式。要在 Athena for Spark 中使用多种表格式，为每种表格式创建一个单独的笔记本。有关在 Athena for Spark 中创建笔记本的信息，请参阅创建您自己的笔记本。
- Delta Lake、Hudi 和 Iceberg 表格式已在 Athena for Spark 上使用 AWS Glue 作为元存储进行了测试。也可以使用其他元存储，但目前不支持这种用法。
- 要使用其他表格式，按照 Athena 控制台和本文档中的说明覆盖默认 `spark_catalog` 属性。这些非 Hive 目录除了可以读取其自己的表格式，还可以读取 Hive 表。

表格版本

下表显示了 Amazon Athena for Apache Spark 中支持的非 Hive 表版本。

表格式	支持的版本
Apache Iceberg	1.2.1
Apache Hudi	0.13
Linux Foundation Delta Lake	2.0.2

在 Athena for Spark 中，这些表格式 `.jar` 文件及其依赖项将加载到 Spark 驱动程序和执行程序的类路径中。

主题

- Apache Iceberg
- Apache Hudi
- Linux Foundation Delta Lake

Apache Iceberg

Apache Iceberg 是 Amazon Simple Storage Service（Amazon S3）中适用于大型数据集的开放表格式。它提供快速的大型表查询性能、原子提交、并发写入和 SQL 兼容表演进等功能。

要在 Athena for Spark 中使用 Apache Iceberg 表，配置以下 Spark 属性。当选择 Apache Iceberg 作为表格式时，这些属性是在 Athena for Spark 控制台中默认配置的属性。有关步骤，请参阅 编辑会话详细信息 和 创建您自己的笔记本。

```
"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"
```

以下过程演示了如何在 Athena for Spark 笔记本中使用 Apache Iceberg 表。在笔记本的新单元格中运行每个步骤。

在 Athena for Spark 中使用 Apache Iceberg 表

1. 定义要在笔记本中使用的常量。

   ```
   DB_NAME = "NEW_DB_NAME"
   TABLE_NAME = "NEW_TABLE_NAME"
   TABLE_S3_LOCATION = "s3://DOC-EXAMPLE-BUCKET"
   ```

2. 创建 Apache Spark DataFrame。

   ```
   columns = ["language","users_count"]
   data = [("Golang", 3000)]
   df = spark.createDataFrame(data, columns)
   ```

3. 创建数据库。

   ```
   spark.sql("CREATE DATABASE {} LOCATION '{}'".format(DB_NAME, TABLE_S3_LOCATION))
   ```

4. 创建空白 Apache Iceberg 表。

   ```
   spark.sql("""
   CREATE TABLE {}.{} (
   language string,
   ```

```
users_count int
) USING ICEBERG
""".format(DB_NAME, TABLE_NAME))
```

5. 在表中插入一行数据。

```
spark.sql("""INSERT INTO {}.{} VALUES ('Golang',
 3000)""".format(DB_NAME, TABLE_NAME))
```

6. 确认您可以查询新表。

```
spark.sql("SELECT * FROM {}.{}".format(DB_NAME, TABLE_NAME)).show()
```

有关使用 Spark DataFrames 和 Iceberg 表的更多信息和示例，请参阅 Apache Iceberg 文档中的 Spark 查询。

Apache Hudi

Apache Hudi是一个开源数据管理框架，可简化增量递增数据的处理。记录级别插入、更新、更新插入和删除操作的处理精度提高，从而减少开销。

要在 Athena for Spark 中使用 Apache Hudi 表，配置以下 Spark 属性。当选择 Apache Hudi 作为表格式时，这些属性是在 Athena for Spark 控制台中默认配置的属性。有关步骤，请参阅 编辑会话详细信息 和 创建您自己的笔记本。

```
"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"
```

以下过程向你展示了如何在 Athena for Spark 笔记本中使用 Apache Hudi 表。在笔记本的新单元格中运行每个步骤。

在 Athena for Spark 中使用 Apache Hudi 表

1. 定义要在笔记本中使用的常量。

```
DB_NAME = "NEW_DB_NAME"
TABLE_NAME = "NEW_TABLE_NAME"
TABLE_S3_LOCATION = "s3://DOC-EXAMPLE-BUCKET"
```

2. 创建 Apache Spark DataFrame。

```
columns = ["language","users_count"]
data = [("Golang", 3000)]
df = spark.createDataFrame(data, columns)
```

3. 创建数据库。

```
spark.sql("CREATE DATABASE {} LOCATION '{}'".format(DB_NAME, TABLE_S3_LOCATION))
```

4. 创建空白 Apache Hudi 表。

```
spark.sql("""
CREATE TABLE {}.{} (
language string,
users_count int
) USING HUDI
TBLPROPERTIES (
primaryKey = 'language',
type = 'mor'
);
""".format(DB_NAME, TABLE_NAME))
```

5. 在表中插入一行数据。

```
spark.sql("""INSERT INTO {}.{} VALUES ('Golang',
 3000)""".format(DB_NAME,TABLE_NAME))
```

6. 确认您可以查询新表。

```
spark.sql("SELECT * FROM {}.{}".format(DB_NAME, TABLE_NAME)).show()
```

Linux Foundation Delta Lake

Linux Foundation Delta Lake 是一种用于大数据分析的表格式。可使用 Athena for Spark 直接读取存储在 Amazon S3 中的 Delta Lake 表。

要在 Athena for Spark 中使用 Delta Lake 表，配置以下 Spark 属性。当选择 Delta Lake 作为表格式时，这些属性是在 Athena for Spark 控制台中默认配置的属性。有关步骤，请参阅 编辑会话详细信息 和 创建您自己的笔记本。

```
"spark.sql.catalog.spark_catalog" : "org.apache.spark.sql.delta.catalog.DeltaCatalog",
```

```
"spark.sql.extensions" : "io.delta.sql.DeltaSparkSessionExtension"
```

以下过程演示了如何在 Athena for Spark 笔记本中使用 Delta Lake 表。在笔记本的新单元格中运行每个步骤。

在 Athena for Spark 中使用 Delta Lake 表

1. 定义要在笔记本中使用的常量。

```
DB_NAME = "NEW_DB_NAME"
TABLE_NAME = "NEW_TABLE_NAME"
TABLE_S3_LOCATION = "s3://DOC-EXAMPLE-BUCKET"
```

2. 创建 Apache Spark DataFrame。

```
columns = ["language","users_count"]
data = [("Golang", 3000)]
df = spark.createDataFrame(data, columns)
```

3. 创建数据库。

```
spark.sql("CREATE DATABASE {} LOCATION '{}'".format(DB_NAME, TABLE_S3_LOCATION))
```

4. 创建空白 Delta Lake 表。

```
spark.sql("""
CREATE TABLE {}.{} (
  language string,
  users_count int
) USING DELTA
""".format(DB_NAME, TABLE_NAME))
```

5. 在表中插入一行数据。

```
spark.sql("""INSERT INTO {}.{} VALUES ('Golang',
 3000)""".format(DB_NAME, TABLE_NAME))
```

6. 确认您可以查询新表。

```
spark.sql("SELECT * FROM {}.{}".format(DB_NAME, TABLE_NAME)).show()
```

Amazon Athena for Apache Spark 中的 Python 库支持

本页面介绍 Amazon Athena for Apache Spark 中使用的运行时、库和程序包所使用的术语及所遵循的生命周期管理。

定义

- Amazon Athena for Apache Spark 是开源 Apache Spark 的自定义版本。要查看当前版本，在笔记本单元格中运行命令 `print(f'{spark.version}')`。
- Athena runtime（Athena 运行时）是代码运行的环境。该环境包括 Python 解释器和 PySpark 库。
- external library or package（外部库或程序包）是 Java、Scala JAR 或 Python 库，不属于 Athena 运行时的一部分，但可以包含在 Athena for Spark 任务中。外部程序包可以由 Amazon 或您构建。
- convenience package（便捷程序包）是 Athena 选择的外部包集合，您可以选择将其包含在 Spark 应用程序中。
- bundle（捆绑包）结合了 Athena 运行时和便捷程序包。
- user library（用户库）是显式添加到 Athena for Spark 任务中的外部库或程序包。
  - 用户库是外部程序包，不属于便捷程序包的一部分。用户库需要加载和安装，与写入某些 `.py` 文件、将其压缩，然后将 `.zip` 文件添加到应用程序一样。
- Athena for Spark application（Athena for Spark 应用程序）是您提交到 Athena for Spark 的任务或查询。

生命周期管理

运行时版本控制和弃用

Python 解释器是 Athena 运行时的主要组件。由于 Python 是一种不断发展的语言，因此会定期发布新版本，并取消对旧版本的支持。Athena 不建议您使用已弃用的 Python 解释器版本来运行程序，并强烈建议您尽可能使用最新的 Athena 运行时。

Athena 运行时弃用计划如下所示：

1. Athena 提供新的运行时后，将继续支持之前的运行时 6 个月。在此期间，Athena 将对之前的运行时应用安全补丁和更新。
2. 6 个月后，Athena 将终止对之前运行时的支持。Athena 将不再对之前的运行时应用安全补丁和其他更新。使用之前运行时的 Spark 应用程序将不再有资格获得技术支持。

3. 12 个月后，您将无法再在使用之前运行时的工作组中更新或编辑 Spark 应用程序。建议您在这段时间结束之前更新 Spark 应用程序。这段时间结束后，您仍然可以运行现有笔记本，但任何仍在使用之前运行时的笔记本都会记录相应的警告。
4. 18 个月后，您将无法再使用之前的运行时在工作组中运行任务。

便捷程序包版本控制和弃用

便捷程序包的内容会随时间而变化。Athena 偶尔会添加、删除或升级这些便捷程序包。

Athena 使用以下便捷程序包准则：

- 便捷程序包具有简单的版本控制方案，例如 1、2、3。
- 每个便捷程序包版本均包含外部程序包的特定版本。Athena 创建便捷程序包后，便捷程序包的外部程序包集合及其相应版本不会更改。
- 每当 Athena 包含新的外部程序包、删除外部程序包或升级一个或多个外部程序包的版本时，Athena 都会创建新的便捷程序包版本。

如果 Athena 弃用程序包所使用的 Athena 运行时，同时也会弃用该便捷程序包。Athena 可以更快地弃用程序包，以限制其支持的捆绑包数量。

便捷程序包弃用计划遵循 Athena 运行时弃用计划。

预安装的 Python 库列表

预安装的 Python 库包括以下内容。

```
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
```

注意事项

- 不支持 MLlib（Apache Spark 机器学习库）和 `pyspark.ml` 包。
- 目前，Athena for Spark 会话不支持 `pip install`。

有关将 Python 库导入 Amazon Athena for Apache Spark 的信息，请参阅 将文件和 Python 库导入 Amazon Athena for Apache Spark。

将文件和 Python 库导入 Amazon Athena for Apache Spark

本文档提供有关如何将文件和 Python 库导入 Amazon Athena for Apache Spark 的示例。

注意事项和限制

- Python 版本 - 目前，Athena for Spark 使用 Python 版本 3.9.16。请注意，Python 包对次要 Python 版本敏感。

- Athena for Spark 架构 - Athena for Spark 在 ARM64 架构上使用 Amazon Linux 2。请注意，某些 Python 库不会为此架构分发二进制文件。
- 二进制共享对象（SO）- 由于 SparkContext addPyFile 方法检测不到二进制共享对象，因此无法在 Athena for Spark 中用于添加取决于共享对象的 Python 包。
- 弹性分布式数据集（RDD）- 不支持 RDD。
- Dataframe.foreach - 不支持 PySpark DataFrame.foreach 方法。

示例

这些示例使用以下约定。

- 占位符 Amazon S3 位置 `s3://DOC-EXAMPLE-BUCKET`。将该项替换为您自己的 S3 存储桶位置。
- 从 Unix Shell 执行的所有代码块均显示为 *`directory_name`* `$`。例如，目录 `/tmp` 中的命令 `ls` 及其输出显示如下：

```
/tmp $ ls
```

输出

```
file1 file2
```

- 将文件写入本地临时目录后将其添加到笔记本中
- 从 Amazon S3 导入文件
- 添加 Python 文件并注册 UDF
- 导入 Python .zip 文件
- 将 Python 库的两个版本作为单独的模块导入
- 从 PyPI 导入 Python .zip 文件
- 从 PyPI 导入具有依赖项的 Python .zip 文件

导入文本文件以用于计算

本节中的示例显示如何导入文本文件以用于 Athena for Spark 笔记本中的计算。

将文件写入本地临时目录后将其添加到笔记本中

以下示例显示如何将文件写入本地临时目录、将其添加到笔记本并对其进行测试。

```
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()
```

输出

```
Calculation completed.
+---+---+-------+
| _1| _2|    col|
+---+---+-------+
|  1|  a|[aaaaa]|
|  2|  b|[bbbbb]|
+---+---+-------+
```

从 Amazon S3 导入文件

以下示例显示如何将文件从 Amazon S3 导入到笔记本中并对其进行测试。

将文件从 Amazon S3 导入到笔记本中

1. 创建一个名为 `test.txt` 的文件，其中有一行包含值 `5`。
2. 将文件添加到 Amazon S3 中的存储桶。此示例使用位置 `s3://DOC-EXAMPLE-BUCKET`。

3. 使用以下代码将文件导入到笔记本中并对其进行测试。

```
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()
```

输出

```
Calculation completed.
+---+---+-------+
| _1| _2|    col|
+---+---+-------+
|  1|  a|[aaaaa]|
|  2|  b|[bbbbb]|
+---+---+-------+
```

添加 Python 文件

本节中的示例显示如何将 Python 文件和库添加到 Athena 中的 Spark 笔记本。

添加 Python 文件并注册 UDF

以下示例显示如何将 Python 文件从 Amazon S3 添加到笔记本并注册 UDF。

将 Python 文件添加到笔记本并注册 UDF

1. 使用您自己的 Amazon S3 位置创建包含以下内容的 `s3://DOC-EXAMPLE-BUCKET/file1.py` 文件：

```
def xyz(input):
    return 'xyz  - udf ' + str(input);
```

2. 在同一 S3 位置创建包含以下内容的 s3://DOC-EXAMPLE-BUCKET/file2.py 文件：

```
from file1 import xyz
def uvw(input):
    return 'uvw -> ' + xyz(input);
```

3. 在 Athena for Spark 笔记本中，运行以下命令。

```
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)
```

输出

```
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... |
+---+---+--------------------+
```

导入 Python .zip 文件

您可以使用 Python `addPyFile` 和 `import` 方法，将 Python .zip 文件导入笔记本。

> **Note**
>
> 导入 Athena Spark 的 `.zip` 文件可能仅包含 Python 包。例如，不支持包含基于 C 的文件的包。

将 Python **`.zip`** 文件导入笔记本

1. 在本地计算机上的桌面目录（例如 `\tmp`）中，创建一个名为 `moduletest` 的目录。
2. 在 `moduletest` 目录中，创建一个名为 `hello.py` 的文件，该文件包含以下内容：

```
def hi(input):
    return 'hi ' + str(input);
```

3. 在同一目录中，添加一个名为 `__init__.py` 的空文件。

   如果列出目录内容，则它们应类似于以下内容。

```
/tmp $ ls moduletest
__init__.py       hello.py
```

4. 使用 `zip` 命令将两个模块文件放入名为 `moduletest.zip` 的文件中。

```
moduletest $ zip -r9 ../moduletest.zip *
```

5. 将 `.zip` 文件上传到 Amazon S3 中的存储桶。
6. 使用以下代码将 Python `.zip` 文件导入笔记本。

```
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()
```

输出

```
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|
+---+---+----+
```

将 Python 库的两个版本作为单独的模块导入

以下代码示例显示如何将两个不同版本的 Python 库作为两个单独的模块从 Amazon S3 中的某个位置添加和导入。该代码会从 S3 添加每个库文件，将其导入，然后打印库版本以验证导入。

```
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__)
```

输出

```
3.15.0
```

```
import simplejson_v3_17_6
print(simplejson_v3_17_6.__version__)
```

输出

```
3.17.6
```

从 PyPI 导入 Python .zip 文件

此示例使用 pip 命令从 Python 程序包索引（PyPI）下载 bpabel/piglatin 项目的 Python .zip 文件。

从 PyPI 导入 Python .zip 文件

1. 在本地桌面上，使用以下命令创建名为 testpiglatin 的目录并创建虚拟环境。

```
/tmp $ mkdir testpiglatin
/tmp $ cd testpiglatin
testpiglatin $ virtualenv .
```

输出

```
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. 创建名为 unpacked 的子目录以保存项目。

```
testpiglatin $ mkdir unpacked
```

3. 使用 pip 命令将该项目安装到 unpacked 目录。

```
testpiglatin $ bin/pip install -t $PWD/unpacked piglatin
```

输出

```
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. 检查目录的内容。

```
testpiglatin $ ls
```

输出

```
bin lib pyvenv.cfg unpacked
```

5. 更改为 unpacked 目录并显示内容。

```
testpiglatin $ cd unpacked
unpacked $ ls
```

输出

```
piglatin piglatin-1.0.6.dist-info
```

6. 使用 zip 命令将 piglatin 项目的内容放入名为 library.zip 的文件中。

```
unpacked $ zip -r9 ../library.zip *
```

输出

```
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. （可选）使用以下命令在本地测试导入。

   a. 将 Python 路径设置为 library.zip 文件位置然后启动 Python。

```
/home $ PYTHONPATH=/tmp/testpiglatin/library.zip
/home $ python3
```

输出

```
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. 导入库并运行测试命令。

```
>>> import piglatin
>>> piglatin.translate('hello')
```

输出

```
'ello-hay'
```

8. 使用以下命令从 Amazon S3 添加 `.zip` 文件，将其导入 Athena 中的笔记本，然后对其进行测试。

```
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()
```

输出

```
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|
```

```
+---+-----+--------+
```

从 PyPI 导入具有依赖项的 Python .zip 文件

此示例从 PyPI 导入 md2gemini 程序包，该程序包将 markdown 中的文本转换为 Gemini 文本格式。该程序包具有以下依赖项：

```
cjkwrap
mistune
wcwidth
```

导入具有依赖项的 Python .zip 文件

1. 在本地计算机上，使用以下命令创建名为 testmd2gemini 的目录并创建虚拟环境。

```
/tmp $ mkdir testmd2gemini
/tmp $ cd testmd2gemini
testmd2gemini$ virtualenv .
```

2. 创建名为 unpacked 的子目录以保存项目。

```
testmd2gemini $ mkdir unpacked
```

3. 使用 pip 命令将该项目安装到 unpacked 目录。

```
/testmd2gemini $ bin/pip install -t $PWD/unpacked md2gemini
```

输出

```
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. 更改为 unpacked 目录并检查内容。

```
testmd2gemini $ cd unpacked
unpacked $ ls -lah
```

输出

```
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. 使用 zip 命令将 md2gemini 项目的内容放入名为 md2gemini.zip 的文件中。

```
unpacked $ zip -r9 ../md2gemini *
```

输出

```
  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. （可选）使用以下命令测试库是否可以在您的本地计算机上运行。

   a. 将 Python 路径设置为 md2gemini.zip 文件位置然后启动 Python。

```
/home $ PYTHONPATH=/tmp/testmd2gemini/md2gemini.zip
/home python3
```

b. 导入库并运行测试。

```
>>> from md2gemini import md2gemini
>>> print(md2gemini('[abc](https://abc.def)'))
```

输出

```
https://abc.def abc
```

7. 使用以下命令从 Amazon S3 添加 `.zip` 文件，将其导入 Athena 中的笔记本，然后执行非 UDF 测试。

```
# (non udf test)
sc.addPyFile('s3://DOC-EXAMPLE-BUCKET/md2gemini.zip')
from md2gemini import md2gemini
print(md2gemini('[abc](https://abc.def)'))
```

输出

```
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. 使用以下命令执行 UDF 测试。

```
# (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()
```

输出

```
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...|
+---+--------------------+--------------------+
```

添加 JAR 文件和自定义 Spark 配置

在 Amazon Athena for Apache Spark 中创建或编辑会话时，可以使用 Spark 属性为会话指定 `.jar` 文件、包或其他自定义配置。要指定 Spark 属性，可使用 Athena 控制台、AWS CLI 或 Athena API。

使用 Athena 控制台指定 Spark 属性

在 Athena 控制台中，您可以在创建笔记本或编辑当前会话时指定 Spark 属性。

在创建笔记本或编辑会话详细信息对话框中添加属性

1. 展开 Spark 属性。
2. 要添加属性，使用在表中编辑或在 JSON 中编辑选项。
   - 对于在表中编辑选项，选择添加属性添加属性，或者选择移除移除属性。使用键和值框输入属性名称及其值。
     - 要添加自定义 `.jar` 文件，使用 `spark.jars` 属性。
     - 要指定包文件，使用 `spark.jars.packages` 属性。
   - 要直接输入和编辑配置，选择在 JSON 中编辑选项。在 JSON 文本编辑器中，您可以执行以下任务：
     - 要将 JSON 文本复制到剪贴板，选择复制。
     - 要从 JSON 编辑器中移除所有文本，选择清除。
     - 要配置换行或为 JSON 编辑器选择颜色主题，选择设置（齿轮）图标。

注意事项

- 可在 Athena for Spark 中设置属性，这与直接在 SparkConf 对象上设置 Spark 属性相同。
- 启动前缀为 `spark.` 的所有 Spark 属性。将忽略带有其他前缀的属性。
- 在 Athena 上，并非所有 Spark 属性都可用于自定义配置。如果您提交的 `StartSession` 请求配置受限，则会话将无法启动。
  - 您不能使用 `spark.athena.` 前缀，因为它是预留前缀。

使用 AWS CLI 或 Athena API 提供自定义配置

要使用 AWS CLI 或 Athena API 提供会话配置，使用 StartSession API 操作或 start-session CLI 命令。在 `StartSession` 请求中，使用 EngineConfiguration 对象的 `SparkProperties` 字段以 JSON 格式传递您的配置信息。这将启动具有您指定的配置的会话。有关请求语法，请参阅《Amazon Athena API 参考》中的 StartSession。

对会话启动错误进行故障排除

如果在会话启动期间出现自定义配置错误，Athena for Spark 控制台将显示一个错误消息横幅。要对会话启动错误进行故障排除，您可以检查会话状态更改或日志记录信息。

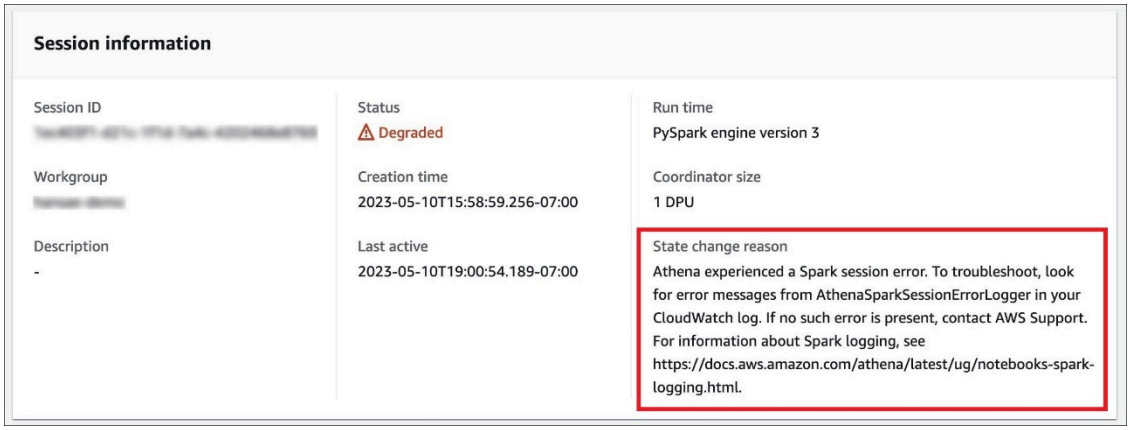
查看会话状态更改信息

您可以从 Athena 笔记本编辑器或 Athena API 中获取有关会话状态更改的详细信息。

在 Athena 控制台中查看会话状态信息

1. 在 Athena 笔记本编辑器中，从右上角的会话菜单中选择查看详细信息。
2. 查看当前会话选项卡。会话信息部分显示会话 ID、工作组、状态和状态更改原因等信息。

   以下屏幕截图示例显示了会话信息对话框的状态更改原因部分中 Athena 中 Spark 会话错误的信息。



使用 Athena API 查看会话状态信息

- 在 Athena API 中，可在 SessionStatus 对象的 `StateChangeReason` 字段中查找会话状态更改信息。

> **Note**
>
> 在您手动停止会话之后，或者如果会话在空闲超时（默认值为 20 分钟）后停止，StateChangeReason 值将更改为会话已按请求终止。

使用日志记录对会话启动错误进行故障排除

Amazon CloudWatch 会记录会话启动期间发生的自定义配置错误。在 CloudWatch Logs 中，搜索 `AthenaSparkSessionErrorLogger` 中的错误消息，以对会话启动失败进行故障排除。

有关 Spark 日志记录的更多信息，请参阅在 Athena 中记录 Spark 应用程序事件。

有关在 Athena for Spark 中对会话进行故障排除的更多信息，请参阅 对会话进行故障排除。

支持的数据和存储格式

下表显示了 Athena for Apache Spark 中原生支持的格式。

Data format（数据格式）	读取	写入	Write compression（写入压缩）
parquet	是	是	无、未压缩、snappy、gzip
orc	是	是	无、snappy、zlib、lzo
json	是	是	bzip2、gzip、deflate
csv	是	是	bzip2、gzip、deflate
文本	是	是	无、bzip2、gzip、deflate
二进制文件	是	不适用	不适用

使用 CloudWatch 指标监控 Apache Spark 计算

选择启用 Spark 的工作组的 Publish CloudWatch metrics 选项后，Athena 会向 Amazon CloudWatch 发布与计算相关的指标。在 CloudWatch 控制台中，您可以创建自定义控制面板，并为指标设置警报和触发器。

Athena 会将以下指标发布到 `AmazonAthenaForApacheSpark` 命名空间下的 CloudWatch 控制台：

- `DPUCount` – 会话期间为执行计算而使用的 DPU 数量。

该指标具有以下维度：

- `SessionId` – 提交计算的会话 ID。
- `WorkGroup` – 工作组名称。

在 Amazon CloudWatch 控制台中查看启用 Spark 的工作组的指标

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 控制台。
2. 在导航窗格中，依次选择 Metrics（指标）、All metrics（所有指标）。

3. 选择 AmazonAthenaForApacheSpark 命名空间。

借助 CLI 查看指标

- 请执行以下操作之一：
  - 要列出启用 Spark 的 Athena 工作组指标，请打开命令提示符，然后使用以下命令：

```
aws cloudwatch list-metrics --namespace "AmazonAthenaForApacheSpark"
```

  - 要列出所有可用的指标，请使用以下命令：

```
aws cloudwatch list-metrics
```

在 Athena 中进行 Apache Spark 计算的 CloudWatch 指标与维度列表

如果您在启用 Spark 的 Athena 工作组中启用了 CloudWatch 指标，Athena 会按工作组将以下指标发送到 CloudWatch。该指标使用 AmazonAthenaForApacheSpark 命名空间。

指标名称	描述
DPUCount	会话期间为执行计算而使用的 DPU（数据处理单元）数量。DPU 是对处理能力的相对度量，它由 4 个 vCPU 的计算容量和 16GB 内存组成。

该指标具有以下维度。

维度	描述
SessionId	提交计算的会话 ID。
工作组	工作组的名称。

在 Athena for Spark 中启用申请方付款 Amazon S3 存储桶

将 Amazon S3 存储桶配置为申请方付款时，将对运行查询的用户账户收取与查询有关的数据访问和数据传输费用。有关更多信息，请参阅《Amazon S3 用户指南》中的使用申请方付款存储桶进行存储传输和使用。

在 Athena for Spark 中，申请方付款存储桶按会话启用，而不是按工作组启用。简而言之，启用申请方付款存储桶包括以下步骤：

1. 在 Amazon S3 控制台中，启用存储桶属性中的申请方付款，然后添加存储桶策略以指定访问权限。
2. 在 IAM 控制台中，创建允许访问存储桶的 IAM policy，然后将该策略附加到将用于访问申请方付款存储桶的 IAM 角色。
3. 在 Athena for Spark 中，添加会话属性以启用申请方付款功能。

1. 在 Amazon S3 存储桶上启用申请方付款并添加存储桶策略

在 Amazon S3 存储桶上启用申请方付款

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶列表中，选择要启用申请方付款的存储桶的链接。
3. 在存储桶页面上，选择属性选项卡。
4. 向下滚动到申请方付款部分，然后选择编辑。
5. 在编辑申请方付款页面上，选择启用，然后选择保存更改。
6. 选择 Permissions（权限）选项卡。
7. 在存储桶策略部分中，选择编辑。
8. 在编辑存储桶策略页面中，将所需的存储桶策略应用于源存储桶。以下示例策略允许访问所有 AWS 主体（`"AWS": "*"`），但您的访问权限可以更加精细。例如，您可能只想在另一个账户中指定特定的 IAM 角色。

```
{
    "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. 创建 IAM policy 并将其附加到 IAM 角色

接下来，创建 IAM policy 以允许访问存储桶。然后，将该策略附加到将用于访问申请方付款存储桶的角色。

为申请方付款存储桶创建 IAM policy 并将该策略附加到角色

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在 IAM 控制台导航窗格中，选择策略。
3. 选择创建策略。
4. 选择 JSON。
5. 在策略编辑器中，添加如下所示的策略：

```
{
    "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. 选择下一步。
7. 在审核和创建页面上，输入策略名称和可选描述，然后选择创建策略。
8. 在导航窗格中，选择角色。
9. 在角色页面中，查找要使用的角色，然后选择角色名称链接。
10. 在权限策略部分中，选择添加权限、附加策略。
11. 在其他权限策略部分中，选中您创建的策略对应的复选框，然后选择添加权限。

3. 添加 Athena for Spark 会话属性

在为申请方付款配置 Amazon S3 存储桶和相关权限后，您可以在 Athena for Spark 会话中启用该功能。

在 Athena for Spark 会话中启用申请方付款存储桶

1. 在笔记本编辑器中，从右上角的 Session（会话）菜单中选择 Edit session（编辑会话）。
2. 展开 Spark 属性。
3. 选择在 JSON 中编辑。
4. 在 JSON 文本编辑器中，输入以下内容：

```
{
  "spark.hadoop.fs.s3.useRequesterPaysHeader":"true"
}
```

5. 选择保存。

启用 Apache Spark 加密

您可以在 Athena 上启用 Apache Spark 加密。这样做可以加密 Spark 节点之间的传输中数据，还可以加密 Spark 本地存储的静态数据。为了增强这些数据的安全性，Athena 使用以下加密配置：

```
spark.io.encryption.keySizeBits="256"
spark.io.encryption.keygen.algorithm="HmacSHA384"
```

要启用 Spark 加密，可使用 Athena 控制台、AWS CLI 或 Athena API。

使用 Athena 控制台启用 Spark 加密

新建启用了 Spark 加密的笔记本

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。
3. 请执行以下操作之一：
   - 在 Notebook explorer（笔记本资源管理器）中，选择 Create notebook（创建笔记本）。
   - 在 Notebook editor（笔记本编辑器）中，选择 Create notebook（创建笔记本），或选择加号图标（+）添加笔记本。
4. 在笔记本名称中，输入笔记本的名称。
5. 展开 Spark 属性选项。
6. 选择打开 Spark 加密。
7. 选择创建。

您创建的笔记本会话已加密。像往常一样使用新笔记本。当您稍后启动使用笔记本的新会话时，也会对新会话进行加密。

您还可以使用 Athena 控制台为现有笔记本启用 Spark 加密。

为现有笔记本启用加密

1. 为先前创建的笔记本打开一个新会话。
2. 在笔记本编辑器中，从右上角的 Session（会话）菜单中选择 Edit session（编辑会话）。
3. 在编辑会话详细信息对话框中，展开 Spark 属性。
4. 选择打开 Spark 加密。
5. 选择保存。

控制台将启动启用了加密的新会话。您稍后为此笔记本创建的会话也将启用加密。

使用 AWS CLI 启用 Spark 加密

可在启动会话时使用 AWS CLI 通过指定相应的 Spark 属性启用加密。

使用 AWS CLI 启用 Spark 加密

1. 使用如下所示的命令创建指定 Spark 加密属性的引擎配置 JSON 对象。

```
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. 在 AWS CLI 中，使用 `athena start-session` 命令并将您创建的 JSON 对象传递给 `--engine-configuration` 参数，如以下示例所示：

```
aws athena start-session \
   --region "region" \
   --work-group "your-work-group" \
   --engine-configuration "$ENGINE_CONFIGURATION_JSON"
```

使用 Athena API 启用 Spark 加密

要使用 Athena API 启用 Spark 加密，使用 StartSession 操作及其 EngineConfiguration `SparkProperties` 参数在 `StartSession` 请求中指定加密配置。

在 Athena for Spark 中配置跨账户 AWS Glue 访问

本主题介绍如何将消费者账户 *666666666666* 和所有者账户 *999999999999* 配置为跨账户 AWS Glue 访问。配置账户后，消费者账户可以从 Athena for Spark 对所有者的 AWS Glue 数据库和表运行查询。

1. 在 AWS Glue 中，提供对消费者角色的访问权限

在 AWS Glue 中，所有者创建了一个策略，为消费者的角色提供对所有者 AWS Glue 数据目录的访问权限。

添加允许消费者角色访问所有者数据目录的 AWS Glue 策略

1. 使用目录所有者的账户登录 AWS Management Console。
2. 通过 https://console.aws.amazon.com/glue/ 打开 AWS Glue 控制台。
3. 在导航窗格中，展开数据目录，然后选择目录设置。
4. 在数据目录设置页面的权限部分，添加如下所示的策略。本策略为使用者账户 *666666666666* 提供了访问拥有者账户 *999999999999* 中数据目录的角色。

```
{
  "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. 配置消费者账户的访问权限

在消费者账户中，创建允许访问所有者 AWS Glue Data Catalog、数据库和表的策略，并将该策略附加到角色。以下示例使用消费者账户 *666666666666*。

创建访问所有者 AWS Glue Data Catalog 的 AWS Glue 策略

1. 使用消费者账户登录 AWS Management Console。
2. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
3. 在导航窗格中，展开访问管理，然后选择策略。
4. 选择创建策略。
5. 在指定权限页面上，选择 JSON。
6. 在策略编辑器中，输入如下所示的 JSON 语句，该语句允许对所有者账户的数据目录进行 AWS Glue 操作。

```
{
    "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. 选择下一步。
8. 在审核并创建页面上，对于策略名称，输入策略的名称。
9. 选择创建策略。

接下来，您可以在消费者账户中使用 IAM 控制台将您刚刚创建的策略附加到消费者账户将用于访问所有者数据目录的 IAM 一个或多个角色。

将 AWS Glue 策略附加到消费者账户中的角色

1. 在消费者账户 IAM 控制台的导航窗格中，选择角色。
2. 在角色页面上，找到您要将策略附加到的角色。
3. 选择添加权限，然后选择附加策略。
4. 找到您刚刚创建的策略。

5. 选择策略对应的复选框，然后选择添加权限。
6. 重复上述步骤，将策略添加到要使用的其他角色。

3. 配置会话并创建查询

在 Athena Spark 中，在请求者账户中，使用指定的角色创建会话，通过创建笔记本或编辑当前会话来测试访问权限。配置会话属性时，请指定以下选项之一：

- Glue 目录分隔符 - 使用这种方法，您可以在查询中包含所有者账户 ID。如果您要使用会话查询来自不同所有者的数据目录，请使用此方法。
- Glue 目录 ID - 使用这种方法，您可以直接查询数据库。如果您要使用会话仅查询单个所有者的数据目录，则此方法会更方便。

使用 AWS Glue 目录分隔符方法

编辑会话属性时，添加以下内容：

```
{
    "spark.hadoop.aws.glue.catalog.separator": "/"
}
```

在单元格中运行查询时，请使用与以下示例类似的语法。请注意，在 FROM 子句中，数据库名称前必须有目录 ID 和分隔符。

```
df = spark.sql('SELECT requestip, uri, method, status FROM `999999999999/
mydatabase`.cloudfront_logs LIMIT 5')
df.show()
```

使用 AWS Glue 目录 ID 方法

编辑会话属性时，输入以下属性。将 *999999999999* 替换为所有者账户 ID。

```
{
    "spark.hadoop.hive.metastore.glue.catalogid": "999999999999"
}
```

在单元格中运行查询时，请使用与以下类似的语法。请注意，在 FROM 子句中，数据库名称前不能有目录 ID 和分隔符。

```
df = spark.sql('SELECT * FROM mydatabase.cloudfront_logs LIMIT 10')
df.show()
```

其他 资源

授予 AWS Glue 数据目录跨账户访问权限

《AWS Lake Formation 开发人员指南》中的同时使用 AWS Glue 和 Lake Formation 管理跨账户权限。

AWS 规范性指导模式中的使用 Amazon Athena 配置对共享 AWS Glue Data Catalog 的跨账户访问权限。

Amazon Athena for Apache Spark 服务限额

服务限额（也称为限制）指 AWS 账户 使用的服务资源或操作的最大数量。有关可在 Amazon Athena for Spark 中使用的其他 AWS 服务服务限额的更多信息，请参阅 Amazon Web Services 一般参考 中的 AWS 服务限额。

> **Note**
>
> 新 AWS 账户 的初始限额可能较低，但会随着时间的推移而增加。Amazon Athena for Apache Spark 会监控每个 AWS 区域 内的账户使用情况，然后根据您的使用情况自动增加限额。如果您的需求超过规定的限制，请联系客户支持。

下表列出了 Amazon Athena for Apache Spark 的服务限额。

名称	默认值	可调整	描述
Apache Spark DPU 并发	160	不支持	AWS 区域 内单一账户的 Apache Spark 计算可并发使用的最大数据处理单元（DPU）数。DPU 是对处理能力的相对度量，它由 4 个 vCPU 的计算容量和 16GB 内存组成。
Apache Spark 会话 DPU 并发	60	不支持	会话内 Apache Spark 计算可并发使用的最大 DPU 数。

Athena 笔记本 API

以下列表包含指向 Athena 笔记本 API 操作的参考链接。有关数据结构及其他 Athena API 操作，请参阅 Amazon Athena API Reference（Amazon Athena API 参考）。

- CreateNotebook
- CreatePresignedNotebookUrl
- DeleteNotebook
- ExportNotebook
- GetCalculationExecution
- GetCalculationExecutionCode
- GetCalculationExecutionStatus
- GetNotebookMetadata
- GetSession
- GetSessionStatus
- ImportNotebook
- ListApplicationDPUSizes
- ListCalculationExecutions
- ListExecutors
- ListNotebookMetadata
- ListNotebookSessions
- ListSessions
- StartCalculationExecution
- StartSession
- StopCalculationExecution
- TerminateSession
- UpdateNotebook
- UpdateNotebookMetadata

Athena for Spark 中的已知问题

本页面记录了 Athena for Apache Spark 中的一些已知问题。

创建表时出现非法参数异常

尽管 Spark 不允许创建位置属性为空的数据库，但如果数据库是在 Spark 外部创建的，AWS Glue 中数据库的 LOCATION 属性可能为空。

如果您创建表并指定 LOCATION 字段为空的 AWS Glue 数据库，可能会出现以下异常：IllegalArgumentException: Cannot create a path from an empty string.（IllegalArgumentException：无法从空字符串创建路径。）

例如，如果 AWS Glue 中的默认数据库 LOCATION 字段为空，以下命令会引发异常：

```
spark.sql("create table testTable (firstName STRING)")
```

建议的解决方法 A – 使用 AWS Glue 向正在使用的数据库添加位置。

向 AWS Glue 数据库添加位置

1. 登录 AWS Management Console，然后打开 AWS Glue 控制台，网址为：https://console.aws.amazon.com/glue/。
2. 在导航窗格中，选择 Databases（数据库）。
3. 在数据库列表中，选择要编辑的数据库。
4. 在数据库的详细信息页面上，选择 Edit（编辑）。
5. 在 Update a database（更新数据库）页面上，在 Location（位置）处输入 Amazon S3 位置。
6. 选择 Update Database（更新数据库）。

建议的解决方法 B – 使用其他 AWS Glue 数据库，该数据库在 Amazon S3 中具有现有有效位置。例如，如果您有一个名为 dbWithLocation 的数据库，请使用命令 spark.sql("use dbWithLocation") 切换到该数据库。

建议的解决方法 C – 使用 Spark SQL 创建表时，为 location 指定一个值，如下例所示。

```
spark.sql("create table testTable (firstName STRING)
       location 's3://DOC-EXAMPLE-BUCKET/'").
```

建议的解决方法 D – 如果您在创建表时指定了位置，但问题仍然出现，请确保您提供的 Amazon S3 路径包含尾部正斜杠。例如，以下命令会引发非法参数异常：

```
spark.sql("create table testTable (firstName STRING)
       location 's3://DOC-EXAMPLE-BUCKET'")
```

要更正此问题，请向该位置添加尾部斜杠（例如 's3:// DOC-EXAMPLE-BUCKET/'）。

在工作组位置创建的数据库

如果您使用 spark.sql('create database db') 之类的命令创建数据库，但不指定数据库的位置，Athena 会在您的工作组位置创建一个子目录，并将该位置用于新创建的数据库。

AWS Glue 默认数据库中 Hive 托管的表问题

如果 AWS Glue 中默认数据库的 Location 属性不为空并指定 Amazon S3 中的有效位置，并且您使用 Athena for Spark 在 AWS Glue 默认数据库中创建 Hive 托管表，则数据将写入在 Athena Spark 工作组中指定的 Amazon S3 位置，而不是 AWS Glue 数据库指定的位置。

是否会出现此问题取决于 Apache Hive 如何处理其默认数据库。Apache Hive 在 Hive 仓库根位置中创建表数据，该位置可能与实际的默认数据库位置不同。

当您使用 Athena for Spark 在 AWS Glue 中的默认数据库下创建 Hive 托管表时，AWS Glue 表元数据可以指向两个不同位置。当您尝试 INSERT 或 DROP TABLE 操作时，这可能会导致意外行为。

要重现该问题，步骤如下：

1. 在 Athena for Spark 中，使用以下方法之一创建或保存 Hive 托管表：
   - SQL 语句，例如 CREATE TABLE $tableName
   - PySpark 命令，例如未在 Dataframe API 中指定 path 选项的 df.write.mode("overwrite").saveAsTable($tableName)。

   此时，AWS Glue 控制台可能会在 Amazon S3 中显示错误的表位置。
2. 在 Athena for Spark 中，使用 DROP TABLE $table_name 语句删除您创建的表。
3. 在运行 DROP TABLE 语句后，您会注意到 Amazon S3 中的底层文件仍然存在。

要解决该问题，可以执行下列操作之一：

解决方案 A - 创建 Hive 托管表时使用其他 AWS Glue 数据库。

解决方案 B - 为 AWS Glue 中的默认数据库指定一个空位置。然后，在默认数据库中创建托管表。

Athena for Spark 和 Athena SQL 之间的 CSV 和 JSON 文件格式不兼容

由于开源 Spark 存在的已知问题，在 Athena for Spark 中根据 CSV 或 JSON 数据创建表时，可能无法从 Athena SQL 中读取该表，反之亦然。

例如，您可能使用以下方法之一在 Athena for Spark 中创建表：

- 使用下面的 `USING csv` 语法：

```
spark.sql('''CREATE EXTERNAL TABLE $tableName (
$colName1 $colType1,
$colName2 $colType2,
$colName3 $colType3)
USING csv
PARTITIONED BY ($colName1)
LOCATION $s3_location''')
```

- 使用以下 DataFrame API 语法：

```
df.write.format('csv').saveAsTable($table_name)
```

由于开源 Spark 存在的已知问题，从 Athena SQL 对生成的表进行查询可能失败。

建议的解决方案 - 尝试在 Athena for Spark 中使用 Apache Hive 语法创建表。有关更多信息，请参阅 Apache Spark 文档中的 CREATE HIVEFORMAT TABLE。

对 Athena for Spark 进行故障排除

使用以下信息来解决在 Athena 上使用笔记本和会话时可能遇到的问题。

主题

- 对启用 Spark 的工作组进行故障排除
- 使用 Spark EXPLAIN 语句对 Spark SQL 进行故障排除
- 记录 Athena 中的 Spark 应用程序事件
- 使用 CloudTrail 对 Athena 笔记本 API 调用进行故障排除
- 绕过 68k 代码块大小限制
- 对会话进行故障排除

- 对表进行故障排除
- 获取支持

对启用 Spark 的工作组进行故障排除

可使用以下信息在 Athena 中对启用 Spark 的工作组进行故障排除。

使用现有 IAM 角色时，会话会停止响应

如果您没有为启用 Spark 的工作组创建新的 AWSAthenaSparkExecutionRole，而是更新或选择了现有 IAM 角色，则会话可能会停止响应。在这种情况下，您可能需要向启用 Spark 的工作组执行角色添加以下信任和权限策略。

添加以下示例信任策略。该策略包含对执行角色进行混淆代理检查。将 *111122223333*、*aws-region* 和 *workgroup-name* 的值替换为您正在使用的 AWS 账户 ID、AWS 区域 和工作组。

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

为启用笔记本的工作组添加类似于以下默认策略的权限策略。修改占位符 Amazon S3 位置和 AWS 账户 ID，使其与您正在使用的位置和 ID 相对应。将 DOC-EXAMPLE-BUCKET、*aws-*

region、*111122223333* 和 *workgroup-name* 的值替换为您正在使用的 Amazon S3 存储桶、AWS 区域、AWS 账户 ID 和工作组。

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

使用 Spark EXPLAIN 语句对 Spark SQL 进行故障排除

您可以将 Spark EXPLAIN 语句与 Spark SQL 结合使用，以对 Spark 代码进行故障排除。以下代码和输出示例显示了这种用法。

Example – Spark SELECT 语句

```
spark.sql("select * from select_taxi_table").explain(True)
```

输出

```
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 – Spark 数据框

以下代码示例显示如何将 EXPLAIN 与 Spark 数据框结合使用。

```
taxi1_df=taxi_df.groupBy("VendorID", "passenger_count").count()
taxi1_df.explain("extended")
```

输出

```
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>
```

记录 Athena 中的 Spark 应用程序事件

Athena 笔记本编辑器允许使用标准 Jupyter、Spark 和 Python 日志记录。您可以使用 `df.show()` 来显示 PySpark DataFrame 内容，也可以使用 `print("Output")` 来显示单元格输出中的值。`stdout`、`stderr` 和 `results` 计算输出将写入 Amazon S3 中的查询结果存储桶位置。

将 Spark 应用程序事件记录到 Amazon CloudWatch

Athena 会话还可以将日志写入您正在使用的账户中的 Amazon CloudWatch。

了解日志流和日志组

CloudWatch 将日志活动组织到日志流和日志组中。

日志流 – CloudWatch 日志流是共享同一个源的一系列日志事件。CloudWatch Logs 中每个独立的日志源构成一个独立的日志流。

日志组 – 在 CloudWatch Logs 中，日志组是一组具有相同保留、监控和访问控制设置的日志流。

对可属于一个日志组的日志流数没有限制。

在 Athena 中，当您首次启动笔记本会话时，Athena 会在 CloudWatch 中创建日志组，该日志组会使用启用 Spark 的工作组名称，如下例所示。

```
/aws-athena/workgroup-name
```

该日志组将为会话中的每个执行程序接收一个日志流，该执行程序至少会产生一个日志事件。执行程序是笔记本会话可以向 Athena 请求的最小计算单位。在 CloudWatch 中，日志流名称以会话 ID 和执行程序 ID 开头。

有关 CloudWatch 日志组和日志流的更多信息，请参阅《Amazon CloudWatch Logs 用户指南》中的使用日志组和日志流。

在 Athena for Spark 中使用标准记录程序对象

在 Athena for Spark 会话中，您可以使用以下两个全局标准记录程序对象，将日志写入 Amazon CloudWatch：

- athena_user_logger – 仅向 CloudWatch 发送日志。如果您想要将 Spark 应用程序的信息直接记录到 CloudWatch，请使用该对象，如下例所示。

```
athena_user_logger.info("CloudWatch log line.")
```

该示例将日志事件写入 CloudWatch，如下所示：

```
AthenaForApacheSpark: 2022-01-01 12:00:00,000 INFO builtins: CloudWatch log line.
```

- athena_shared_logger – 出于支持目的，向 CloudWatch 和 AWS 发送相同的日志。您可以使用该对象与 AWS 服务团队共享日志以进行故障排除，如下例所示。

```
athena_shared_logger.info("Customer debug line.")
var = [...some variable holding customer data...]
athena_shared_logger.info(var)
```

该示例将 debug 行和 `var` 变量的值记录到 CloudWatch Logs，并将每行的副本发送到 AWS Support。

> **Note**
>
> 为保护您的隐私，不会与 AWS 共享计算代码和结果。确保您对 `athena_shared_logger` 的调用只会写入您希望对 AWS Support 可见的信息。

提供的记录程序将通过 Apache Log4j 写入事件，并继承此接口的日志记录级别。可能的日志级别值包括 `DEBUG`、`ERROR`、`FATAL`、`INFO` 以及 `WARN` 或 `WARNING`。您可以使用记录程序上相应的命名函数来生成这些值。

> **Note**
>
> 请勿重新绑定名称 `athena_user_logger` 或 `athena_shared_logger`。这样做会使日志记录对象无法在会话的剩余时间内写入 CloudWatch。

示例：将笔记本事件记录到 CloudWatch

以下过程显示如何将 Athena 笔记本事件记录到 Amazon CloudWatch Logs。

将 Athena 笔记本事件记录到 Amazon CloudWatch Logs

1. 按照 Amazon Athena 上的 Apache Spark 入门，在 Athena 中创建具有唯一名称的启用 Spark 的工作组。本教程使用工作组名称 `athena-spark-example`。
2. 按照 创建您自己的笔记本 中的步骤创建笔记本并启动新会话。
3. 在 Athena 笔记本编辑器的新笔记本单元格中，输入以下命令：

   ```
   athena_user_logger.info("Hello world.")
   ```

4. 运行单元格。
5. 执行以下操作之一，以检索当前会话 ID：

   - 查看单元格输出（例如 `... session=72c24e73-2c24-8b22-14bd-443bdcd72de4`）。
   - 在新单元格中，运行魔术命令 `%session_id`。

6. 保存会话 ID。
7. 使用用于运行笔记本会话的同一 AWS 账户，从 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 控制台。
8. 在 CloudWatch 控制台导航窗格中，选择 Log groups（日志组）。
9. 在日志组列表中，选择具有启用 Spark 的 Athena 工作组名称的日志组，如下例所示。

   ```
   /aws-athena/athena-spark-example
   ```

   Log streams（日志流）部分包含工作组的一个或多个日志流链接的列表。每个日志流名称均包含会话 ID、执行程序 ID 以及由正斜杠字符分隔的唯一 UUID。

   例如，如果会话 ID 为 `5ac22d11-9fd8-ded7-6542-0412133d3177`，执行程序 ID 为 `f8c22d11-9fd8-ab13-8aba-c4100bfba7e2`，则日志流的名称类似于以下示例。

   ```
   5ac22d11-9fd8-ded7-6542-0412133d3177/f8c22d11-9fd8-ab13-8aba-c4100bfba7e2/f012d7cb-cefd-40b1-90b9-67358f003d0b
   ```

10. 为会话选择日志流链接。
11. 在 Log events（日志事件）页面上，查看 Message（消息）列。

    您运行的单元格的日志事件类似于以下内容：

    ```
    AthenaForApacheSpark: 2022-01-01 12:00:00,000 INFO builtins: Hello world.
    ```

12. 返回到 Athena 笔记本编辑器。
13. 在新单元格中，输入以下代码。代码会将变量记录到 CloudWatch：

```
x = 6
athena_user_logger.warn(x)
```

14. 运行单元格。
15. 返回 CloudWatch 控制台的 Log events（日志事件）页面，可查看同一日志流。
16. 日志流现在包含具有如下消息的日志事件条目：

```
AthenaForApacheSpark: 2022-01-01 12:00:00,000 WARN builtins: 6
```

使用 CloudTrail 对 Athena 笔记本 API 调用进行故障排除

要对笔记本 API 调用进行故障排除，您可以检查 Athena CloudTrail 日志以调查异常情况或查找用户发起的操作。有关将 CloudTrail 与 Athena 结合使用的详细信息，请参阅 使用 AWS CloudTrail 记录 Amazon Athena API 调用。

以下示例显示了 Athena 笔记本 API 的 CloudTrail 日志条目：

- StartSession
- TerminateSession
- ImportNotebook
- UpdateNotebook
- StartCalculationExecution

StartSession

以下示例显示了笔记本 StartSession 事件的 CloudTrail 日志。

```
{
    "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

以下示例显示了笔记本 TerminateSession 事件的 CloudTrail 日志。

```
{
    "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

以下示例显示了笔记本 ImportNotebook 事件的 CloudTrail 日志。出于安全原因，将隐藏某些内容。

```
{
    "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

以下示例显示了笔记本 UpdateNotebook 事件的 CloudTrail 日志。出于安全原因，将隐藏某些内容。


```
{
    "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

以下示例显示了笔记本 StartCalculationExecution 事件的 CloudTrail 日志。出于安全原因，将隐藏某些内容。

```
{
    "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"
```

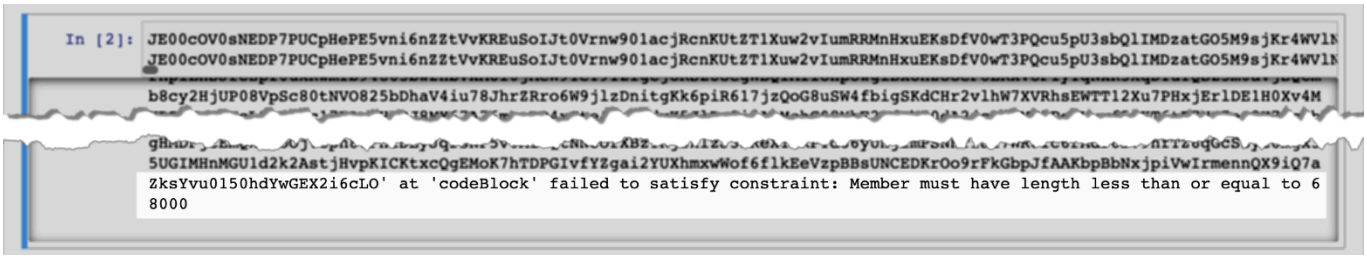
```
            }
        }
    },
    "eventTime": "2022-10-14T16:52:37Z",
    "eventSource": "athena.amazonaws.com",
    "eventName": "StartCalculationExecution",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.14",
    "userAgent": "Boto3/1.24.84 Python/3.8.14 Linux/4.14.225-175.364.amzn2.aarch64
 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"
    }
}
```


绕过 68k 代码块大小限制

Athena for Spark 的已知计算代码块大小限制为 68000 个字符。当您运行代码块超过此限制的计算时，可能会收到以下错误消息：

“codeBlock”中的“...”无法满足约束条件：成员长度必须小于或等于 68000

下图显示了 Athena 控制台笔记本编辑器中的此错误。



当您使用 AWS CLI 运行包含大型代码块的计算时，可能会出现同样的错误，如以下示例所示。

```
aws athena start-calculation-execution \
    --session-id "{SESSION_ID}" \
    --description "{SESSION_DESCRIPTION}" \
    --code-block "{LARGE_CODE_BLOCK}"
```

该命令会给出以下错误消息：

“codeBlock”中的 *{LARGE_CODE_BLOCK}* 无法满足约束条件：成员长度必须小于或等于 68000

解决办法

要解决此问题，将包含您的查询或计算代码的文件上传到 Amazon S3。然后，使用 boto3 读取文件并运行您的 SQL 或代码。

以下示例假设您已经将包含您的 SQL 查询或 Python 代码的文件上传到 Amazon S3。

SQL 示例

以下示例代码从 Amazon S3 存储桶读取 `large_sql_query.sql` 文件，然后运行该文件包含的大型查询。

```
s3 = boto3.resource('s3')
def read_s3_content(bucket_name, key):
    response = s3.Object(bucket_name, key).get()
    return response['Body'].read()

# SQL
sql = read_s3_content('bucket_name', 'large_sql_query.sql')
df = spark.sql(sql)
```

PySpark 示例

以下示例代码从 Amazon S3 读取 `large_py_spark.py` 文件，然后运行该文件包含的大型代码块。

```
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)
```

对会话进行故障排除

使用本主题中的信息对会话问题进行故障排除。

会话运行状况不佳

如果您收到错误消息 Session in unhealthy state. Please create a new session（会话运行状况不佳，请创建新会话），请终止现有会话并创建一个新会话。

无法与笔记本服务器建立连接

打开笔记本后，可能显示以下错误消息：

```
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.
```

原因

当 Athena 打开笔记本后，Athena 会创建会话并使用预签名的笔记本 URL 连接到笔记本。与笔记本的连接使用 WSS（WebSocket Secure）协议。

出现错误的可能原因如下：

- 本地防火墙（例如公司防火墙）正在阻止 WSS 流量。
- 本地计算机上的代理或防病毒软件正在阻止 WSS 连接。

解决方案

假设您在 `us-east-1` 区域中有 WSS 连接，如下所示：

```
wss://94c2bcdf-66f9-4d17-9da6-7e7338060183.analytics-gateway.us-east-1.amazonaws.com/
api/kernels/33c78c82-b8d2-4631-bd22-1565dc6ec152/channels?session_id=
7f96a3a048ab4917b6376895ea8d7535
```

要解决错误，请使用以下策略之一。

- 使用通配符模式语法允许列出各个 AWS 区域 和 AWS 账户 中端口 443 上的 WSS 流量。

  ```
  wss://*amazonaws.com
  ```

- 使用通配符模式语法允许列出某个 AWS 区域 中端口 443 上和您指定的 AWS 区域 各个 AWS 账户 中的 WSS 流量。下面的示例使用了 us-east-1。

  ```
  wss://*analytics-gateway.us-east-1.amazonaws.com
  ```

对表进行故障排除

创建表时无法创建路径错误

错误消息：IllegalArgumentException: Cannot create a path from an empty string.
（IllegalArgumentException：无法从空字符串创建路径。）

原因：当您在 Athena 中使用 Apache Spark 在 AWS Glue 数据库中创建表并且数据库的 LOCATION 属性为空时，可能会发生此错误。

建议的解决方法：有关更多信息和解决方案，请参阅 创建表时出现非法参数异常。

查询 AWS Glue 表时出现 AccessDeniedException

错误消息：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：无法验证默认数据库是否存在：com.amazonaws.services.glue.model.AccessDeniedException：用户：arn:aws:sts::aws-account-id:assumed-role/AWSAthenaSparkExecutionRole-unique-identifier/AthenaExecutor-unique-identifier

无权执行：资源上的 glue:GetDatabase：arn:aws:glue:aws-region:aws-account-id:catalog：因为没有基于身份的策略允许 glue:GetDatabase 操作（服务：AWSGlue；状态代码：400；错误代码：AccessDeniedException；请求 ID：request-id；代理：null）]

原因：启用 Spark 的工作组的执行角色缺少访问 AWS Glue 资源的权限。

建议的解决方法：要解决此问题，请向您的执行角色授予访问 AWS Glue 资源的权限，然后编辑 Amazon S3 存储桶策略以向执行角色授予访问权限。

以下过程将详细介绍这些步骤。

向 AWS Glue 资源授予您的执行角色权限

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 如果控制台导航窗格不可见，请选择左侧的扩展菜单。

3. 在 Athena 控制台导航窗格中，选择 Workgroups（工作组）。
4. 在 Workgroups（工作组）页面上，选择要查看的工作组的链接。
5. 在工作组的 Overview Details（概述详细信息）页面上，选择 Role ARN（角色 ARN）链接。该链接将在 IAM 控制台中打开 Spark 执行角色。
6. 在 Permissions policies（权限策略）部分中，选择链接的角色策略名称。
7. 选择 Edit policy（编辑策略），然后选择 JSON。
8. 将 AWS Glue 访问权限添加到角色。通常，您可以为 `glue:GetDatabase` 和 `glue:GetTable` 操作添加权限。有关配置 IAM 角色的更多信息，请参阅《IAM 用户指南》中的添加和删除 IAM 身份权限。

9. 选择 Review policy (查看策略)，然后选择 Save changes (保存更改)。
10. 编辑 Amazon S3 存储桶策略，以向执行角色授予访问权限。请注意，您必须授予角色访问存储桶和存储桶中对象的权限。有关步骤，请参阅《Amazon Simple Storage Service 用户指南》中的使用 Amazon S3 控制台添加存储桶策略。

获取支持

要获得 AWS 的帮助，请从 AWS Management Console 中依次选择 Support（支持）、Support Center（支持中心）。为便于您体验，请准备好以下信息：

- Athena 查询 ID
- 会话 ID
- 计算 ID

发布说明

按发布日期描述 Amazon Athena 功能、改进和错误修复。

主题

2024 年 Athena 发布说明

2024 年 4 月 26 日

发布时间：2024 年 4 月 26 日

Athena 发布了 JDBC 驱动程序版本 3.2.0。有关此驱动程序版本的更多信息，请参阅 Amazon Athena JDBC 3.x 发布说明。要下载 JDBC 3.x 驱动程序，请参阅 JDBC 3.x 驱动程序下载。

2024 年 4 月 24 日

发布时间：2024 年 4 月 24 日

Athena 宣布推出以下修复和改进。

- Parquet – Athena 现在在 Parquet 中支持对不包含在列表或地图组中的未注释重复基元字段进行向后兼容读取。此更改可防止返回无提示的错误结果，并改进架构不匹配的错误消息传递。

  有关更多信息，请参阅 GitHub.com 上的在 Parquet 中支持对未注释重复基元字段进行向后兼容读取。

- Iceberg OPTIMIZE – 解决了在 `WHERE` 子句中使用非分区键筛选器时导致数据丢失的 `OPTIMIZE` 查询问题。有关更多信息，请参阅 OPTIMIZE。

2024 年 4 月 16 日

发布时间：2024 年 4 月 16 日

使用新推出的 Amazon Athena 联合查询传递功能，可直接在底层数据来源上运行全部查询。联合传递查询有助于充分利用原始数据来源的独特函数、查询语言和性能。例如，可以使用 PartiQL 语言在 DynamoDB 上运行 Athena 查询。若要运行 `SELECT` 查询（这些查询可聚合、联接或调用 Athena 中不可用的数据来源函数），也可以使用联合传递查询。使用传递查询可以减少 Athena 处理的数据量，从而缩短查询时间。

有关更多信息，请参阅 运行联合传递查询。要将当前使用的连接器升级到最新版本，请参阅 更新数据来源连接器。

2024 年 4 月 10 日

发布时间：2024 年 4 月 10 日

Athena 发布了以下功能和改进功能。

ODBC 1.2.3.1000 驱动程序

发布了适用于 Athena 的 ODBC 1.2.3.1000 驱动程序。

已解决的问题：

- 代理服务器连接问题 – 在无根证书的情况下使用代理服务器时，连接器无法建立连接。

有关更多信息以及下载 ODBC 1.x 驱动程序、发布说明和文档，请参阅 Athena ODBC 1.x 驱动程序。

JDBC 2.1.5 驱动程序

发布了适用于 Athena 的 JBDC 2.1.5 驱动程序。

更新与增强功能：

- 已将 AWS Java SDK 更新到版本 1.12.687。

- 已将 Jackson 库更新到版本 2.16.0。
- 已将 Logback 库更新到版本 1.3.14。

有关更多信息以及下载 JDBC 2.x 驱动程序、发布说明和文档，请参阅 Athena JDBC 2.x 驱动程序。

2024 年 4 月 8 日

发布时间：2024 年 4 月 8 日

Athena 宣布推出 ODBC 驱动程序版本 2.0.3.0。有关更多信息，请参阅 2.0.3.0 版本注释。要下载新的 ODBC 版本 2 驱动程序，请参阅 ODBC 2.x 驱动程序下载。有关连接信息，请参阅Amazon Athena ODBC 2.x。

2024 年 3 月 15 日

发布时间：2024 年 3 月 18 日

Amazon Athena 宣布在加拿大西部（卡尔加里）区域推出 Athena SQL。

有关每个 AWS 区域 提供的 AWS 服务 的完整列表，请参阅按区域划分的 AWS 服务。

2024 年 2 月 15 日

发布时间：2024 年 2 月 15 日

Athena 发布了 JDBC 驱动程序版本 3.1.0。

Amazon Athena JDBC 驱动程序版本 3.1.0 添加了对 Microsoft Active Directory 联合身份验证服务（AD FS）Windows 集成式身份验证和基于表单的身份验证的支持。3.1.0 版本还包含其他细微改进以及错误修复。

要下载 JDBC 驱动程序版本 3，请参阅 JDBC 3.x 驱动程序下载。

2024 年 1 月 31 日

发布时间：2024 年 1 月 31 日

Athena 发布了以下功能和改进功能。

- Hudi 升级 – 现在，您可以使用 Athena SQL 查询 Hudi 0.14.0 表了。有关使用 Athena SQL 查询 Hudi 表的信息，请参阅 使用 Athena 查询 Apache Hudi 数据集。

2023 年的 Athena 发布说明

2023 年 12 月 14 日

发布时间：2023 年 12 月 14 日

Athena 宣布推出以下修复和改进。

Athena 发布了 JDBC 驱动程序版本 2.1.3。该驱动程序解决了以下问题：

- 日志记录经过改进，可以避免与 Spring Boot 和 Gradle 应用程序日志记录发生冲突。
- 在使用 `executeBatch()` JDBC 方法插入记录时，该驱动程序仅错误地插入了一条记录。由于 Athena 不支持批量执行查询，因此该驱动程序现在会在您使用 `executeBatch()` 时报告错误。要解决此限制，您可以循环提交单个查询。

要下载新的 JDBC 驱动程序、发布说明和文档，请参阅 Athena JDBC 2.x 驱动程序。

2023 年 12 月 9 日

发布时间：2023 年 12 月 9 日

为 Athena 发布了 ODBC 1.2.1.1000 驱动程序。

功能和增强功能：

- 更新了 RStudio 支持 – ODBC 驱动程序现在支持 macOS 上的 RStudio。
- 支持单个目录和架构 – 连接器现在可以返回单个目录和架构。有关更多信息，请参阅可下载的安装和配置指南。

已解决的问题：

- 预准备语句 – 当运行采用列架构且带参数数组的预准备语句时，连接器返回了错误的查询结果。
- 列大小 – 当选择 `$file_modified_time` 系统列时，连接器返回了错误的列大小。
- SQLPrepare – 在 `SELECT` 查询中绑定与 `SQLPrepare` 相关的参数时，连接器返回了错误。

有关更多信息以及下载新驱动程序、版本注释和文档，请参阅 Athena ODBC 1.x 驱动程序。

2023 年 12 月 7 日

发布时间：2023 年 12 月 7 日

Athena 宣布推出 ODBC 驱动程序版本 2.0.2.1。有关更多信息，请参阅 2.0.2.1 版本注释。要下载新的 ODBC 版本 2 驱动程序，请参阅 ODBC 2.x 驱动程序下载。有关连接信息，请参阅Amazon Athena ODBC 2.x。

2023 年 12 月 5 日

发布时间：2023 年 12 月 5 日

现在，您可以创建采用 AWS IAM Identity Center 身份验证模式的 Athena SQL 工作组。这些工作组支持 IAM Identity Center 的可信身份传播功能。可信身份传播允许跨 AWS 分析服务（例如 Amazon Athena 和 Amazon EMR Studio）使用身份。

有关更多信息，请参阅 使用已启用 IAM Identity Center 的 Athena 工作组。

2023 年 11 月 28 日

发布时间：2023 年 11 月 28 日

您现在可以查询 Amazon S3 Express One Zone 存储类中的数据，以快速获得查询结果。S3 Express One Zone 是一种高性能、单可用区存储类，旨在为最常访问的数据和延迟敏感型应用程序提供一致的个位数毫秒级数据访问。首先，请将数据移至 S3 Express One Zone 存储空间并使用 AWS Glue Data Catalog 对数据进行分类，以便在 Athena 中获得无缝查询体验。

有关更多信息，请参阅 查询 S3 Express One Zone 数据。

2023 年 11 月 27 日

发布时间：2023 年 11 月 27 日

Athena 发布了以下功能和改进功能。

- Glue Data Catalog – Glue Data Catalog 视图跨 AWS 服务（例如 Amazon Athena 和 Amazon Redshift）提供单一通用视图。在 Glue Data Catalog 视图中，访问权限由视图的创建用户（而不是

视图的查询用户）定义。这些视图提供更好的访问控制，有助于确保记录的完整性，增强安全性，并且可以防止对基础表的访问。

有关更多信息，请参阅 使用 AWS Glue Data Catalog 视图。

- CloudTrail Lake 支持 – 您现在可以使用 Amazon Athena 来分析 AWS CloudTrail Lake 中的数据。AWS CloudTrailLake 是 CloudTrail 的托管数据湖，用于聚合、永恒存储和分析活动日志，以进行审计、安全和运营调查。要从 Athena 查询 CloudTrail Lake 活动日志，您无需移动数据或构建单独的数据处理管道。无需 ETL 操作。

  要开始使用，请在 CloudTrail Lake 中启用数据联合身份验证。当您与 AWS Glue Data Catalog 分享 CloudTrail Lake 事件数据存储元数据时，CloudTrail 会创建必要的 AWS Glue Data Catalog 资源并使用 AWS Lake Formation 注册数据。在 Lake Formation 中，您可以指定可使用 Athena 查询事件数据存储的用户和角色。

  有关更多信息，请参阅《AWS CloudTrail 用户指南》中的 Enable Lake query federation。

2023 年 11 月 17 日

发布时间：2023 年 11 月 17 日

Athena 发布了以下功能和改进功能。

功能

- 成本型优化器 – Athena 宣布使用来自 AWS Glue 的统计数据全面进行基于成本的优化。要在 Athena SQL 中优化查询，您可以请求 Athena 为 AWS Glue 中的表收集表级或列级统计数据。如果查询中的所有表都有统计数据，Athena 会使用这些统计数据来检查备选执行计划，然后选择最有可能最快的执行计划。

  有关更多信息，请参阅 使用成本型优化器。

- Amazon EMR Studio 集成 – 您现在可以在 Amazon EMR Studio 中使用 Athena，而不必直接使用 Athena 控制台。利用 Amazon EMR 中的 Athena 集成，您可以执行以下任务：
  - 执行 Athena SQL 查询
  - 查看查询结果
  - 查看查询历史记录
  - 查看保存的查询
  - 执行参数化查询

- 查看数据目录的数据库、表和视图

有关更多信息，请参阅AWS 服务 与 Athena 的集成主题中的 Amazon EMR Studio。

- 嵌套访问控制 – Athena 宣布支持 Lake Formation 对嵌套数据的访问控制。在 Lake Formation 中，您可以对具有 `struct` 数据类型的嵌套列定义和应用数据筛选条件。您可以使用数据筛选功能来限制用户对嵌套列子结构的访问权限。有关如何为嵌套数据创建数据筛选条件的信息，请参阅《AWS Lake Formation Developer Guide》中的 Creating a data filter。
- 预置容量使用指标 – Athena 宣布推出新的 CloudWatch 容量预留指标。您可以使用新指标来跟踪已预置的 DPU 数量和查询所使用的 DPU 数量。查询完成后，您还可以查看查询使用的 DPU 数量。

  有关更多信息，请参阅 使用 CloudWatch 指标监控 Athena 查询。

改进

- 错误消息更改 – `Insufficient Lake Formation permissions` 错误消息现在显示为 `Table not found` 或 `Schema not found`。进行此更改是为了防止恶意行为者从错误消息中推断出表或数据库资源的存在。

2023 年 11 月 16 日

发布时间：2023 年 11 月 16 日

Athena 发布了一个新的 JDBC 驱动程序，该驱动程序改善了连接、查询和可视化来自兼容的 SQL 开发和商业智能应用程序的数据的体验。新的驱动程序易于升级。驱动程序可以直接从 Amazon S3 读取查询结果，让您可以更快地获得查询结果。

有关更多信息，请参阅 Athena JDBC 3.x 驱动程序。

2023 年 10 月 31 日

发布时间：2023 年 10 月 31 日

Amazon Athena 宣布预置容量有 1 小时预留。从今天开始，您可以在一小时后预留和释放预置容量。这一变化使得优化需求会随时间变化的工作负载的成本变得更加简单。

预置容量是 Athena 中提供的工作负载管理功能，可帮助您优化、控制和扩展最重要的交互式工作负载。您可以随时添加容量以增加可并发运行的查询数量，控制可以使用容量的工作负载，并在工作负载之间共享容量。

有关更多信息，请参阅 管理查询处理容量。有关定价信息，请访问 Amazon Athena 定价页面。

2023 年 10 月 25 日

发布时间：2023 年 10 月 26 日

Athena 宣布推出以下修复和改进。

jackson-core package – 数值大于 1000 个字符的 JSON 文本现在将失败。此修复解决了 sonatype-2022-6438 安全问题。

2023 年 10 月 17 日

发布时间：2023 年 10 月 17 日

Athena 宣布推出 ODBC 驱动程序版本 2.0.2.0。有关更多信息，请参阅 2.0.2.0 版本注释。要下载新的 ODBC 版本 2 驱动程序，请参阅 ODBC 2.x 驱动程序下载。有关连接信息，请参阅Amazon Athena ODBC 2.x。

2023 年 9 月 26 日

发布时间：2023 年 9 月 26 日

Athena 发布了以下功能和改进功能。

- 适用于 Delta Lake 表的 Lake Formation 读取支持。有关在 Athena 中使用 Delta Lake 表的更多信息，请参阅查询 Linux Foundation Delta Lake 表。

2023 年 8 月 23 日

发布时间：2023 年 8 月 23 日

Amazon Athena 宣布在以色列（特拉维夫）区域推出 Athena SQL。

有关每个 AWS 区域 提供的 AWS 服务 的完整列表，请参阅按区域划分的 AWS 服务。

2023 年 8 月 10 日

发布时间：2023 年 8 月 10 日

Athena 宣布推出以下修复和改进。

ODBC 驱动程序版本 2.0.1.1

Athena 宣布推出 ODBC 驱动程序版本 2.0.1.1。有关更多信息，请参阅 2.0.1.1 版本注释。要下载新的 ODBC 版本 2 驱动程序，请参阅 ODBC 2.x 驱动程序下载。有关连接信息，请参阅Amazon Athena ODBC 2.x。

JDBC 驱动程序版本 2.1.1

Athena 发布了 JDBC 驱动程序版本 2.1.1。该驱动程序解决了以下问题：

- 使用包含正则表达式的语句创建表时发生的错误。
- 导致 `ApplicationName` 连接参数应用不正确的问题。

要下载新的 JDBC 驱动程序、发布说明和文档，请参阅 通过 JDBC 连接到 Amazon Athena。

2023 年 7 月 31 日

发布时间：2023 年 7 月 31 日

Amazon Athena 宣布在其他 AWS 区域 推出 Athena SQL。

此版本扩展了 Athena SQL 的覆盖区域，将亚太地区（海得拉巴）、亚太地区（墨尔本）、欧洲（西班牙）和欧洲（苏黎世）包括在内。

有关每个 AWS 区域 提供的 AWS 服务 的完整列表，请参阅按区域划分的 AWS 服务。

2023 年 7 月 27 日

发布时间：2023 年 7 月 27 日

Athena 发布 Google BigQuery 连接器版本 2023.30.1。此版本的连接器缩短了查询执行时间，并增加了针对 BigQuery 私有端点进行查询的支持。

有关 Google BigQuery 连接器的信息，请参阅 Amazon Athena Google BigQuery 连接器。有关更新现有数据来源连接器的信息，请参阅 更新数据来源连接器。

2023 年 7 月 24 日

发布时间：2023 年 7 月 24 日

Athena 宣布推出以下修复和改进。

- 使用并集查询 - 使用并集提高了某些查询的性能。
- 使用类型比较进行联接 - 修复了包含两种不同类型比较的 JOIN 语句可能出现的查询失败问题。
- 嵌套列上的子查询 - 修复了在嵌套列上关联子查询时与查询失败有关的问题。
- Iceberg 视图 - 修复了 Apache Iceberg 视图中时间戳列精度的兼容性问题。现在，无论列是在 Athena 引擎版本 2 还是 Athena 引擎版本 3 上创建，都可读取包含时间戳列的 Iceberg 视图。

2023 年 7 月 20 日

发布时间：2023 年 7 月 20 日

Athena 发布 JDBC 驱动程序版本 2.1.0。该驱动程序包含新的增强功能并解决了一个问题。

增强功能

以下 Jackson JSON 解析器库已升级：

- jackson-annotations 2.15.2（以前为 2.14.0）
- jackson-core 2.15.2（以前为 2.14.0）
- jackson-databind 2.15.2（以前为 2.14.0）

已解决的问题

- 修复了使用 sql2o 库时传递数组参数的问题。

有关更多信息以及下载新驱动程序、版本注释和文档，请参阅 通过 JDBC 连接到 Amazon Athena。

2023 年 7 月 13 日

发布时间：2023 年 9 月 19 日

Athena 发布了以下功能和改进功能。

- EXPLAIN ANALYZE – 在 EXPLAIN ANALYZE 的输出中增加了对队列、分析、计划和执行时间的支持。
- EXPLAIN – 现在，当查询包含聚合时，EXPLAIN 输出会显示统计信息。
- Parquet Hive SerDe – 增加了允许在读取 Parquet 数据时忽略处理统计数据的 parquet.ignore.statistics 属性。有关信息，请参阅忽略 Parque 统计数据。

有关 EXPLAIN 和 EXPLAIN ANALYZE 的更多信息，请参阅 在 Athena 中使用 EXPLAIN 和 EXPLAIN ANALYZE。有关 Parquet Hive SerDe 的更多信息，请参阅 Parquet SerDe。

2023 年 7 月 3 日

发布时间：2023 年 7 月 25 日

从 2023 年 7 月 3 日起，Athena 开始编辑 CloudTrail 日志中的查询字符串。现在，查询字符串的值为 ***OMITTED***。进行此更改是为了防止意外泄露可能包含敏感信息的表名或筛选器值。如果您之前依赖 CloudTrail 日志来访问完整的查询字符串，我们建议您使用 Athena::GetQueryExecution API 并从 CloudTrail 日志传入 responseElements.queryExecutionId 的值。有关更多信息，请参阅 Amazon Athena API 参考中的 GetQueryExecution。

2023 年 6 月 30 日

发布时间：2023 年 6 月 30 日

Athena 查询编辑器现在支持提前输入代码建议，以实现更快的查询创作体验。现在您可以使用以下功能，以更高的准确性和更高的效率编写 SQL 查询：

- 在您输入时，将实时显示关键字、局部变量、片段和目录项的建议。
- 当您在数据库名称或表名称后面输入一个点时，该编辑器会显示一系列的表或列，方便您从中进行选择。
- 当您将光标悬停在片段建议上时，摘要会显示该片段的语法和用法的简要概述。
- 为了提高代码的可读性，还更新了关键字及其突出显示规则，使其与 Trino 和 Hive 的最新语法保持一致。

该功能已默认启用。您可以在代码编辑器首选项设置中启用或禁用该功能。

要在 Athena 查询编辑器中试用提前输入代码建议，请访问 Athena 控制台，网址为 https://console.aws.amazon.com/athena/。

2023 年 6 月 29 日

发布时间：2023 年 6 月 29 日

- Athena 宣布推出 ODBC 驱动程序版本 2.0.1.0。有关更多信息，请参阅 2.0.1.0 版本注释。要下载新的 ODBC 版本 2 驱动程序，请参阅 ODBC 2.x 驱动程序下载。有关连接信息，请参阅Amazon Athena ODBC 2.x。

- Athena 及其功能现已在中东（阿联酋）区域可用。有关每个 AWS 区域 提供的 AWS 服务 的完整列表，请参阅按区域划分的 AWS 服务。

2023 年 6 月 28 日

发布时间：2023 年 6 月 28 日

现在您可以使用 Amazon Athena 查询从 S3 Glacier Flexible Retrieval（以前称为 Glacier）和 S3 Glacier Deep Archive Amazon S3 存储类还原的对象。您可以针对每个表配置此功能。只有 Athena 引擎版本 3 上的 Apache Hive 表支持该功能。

有关更多信息，请参阅 查询还原的 Amazon S3 Glacier 对象。

2023 年 6 月 12 日

发布时间：2023 年 6 月 12 日

Athena 宣布推出以下修复和改进。

- Parquet Reader timestamps – 增加了对 Parquet Reader 的时间戳读取 `bigint`（毫秒）的支持。此更新与 Athena 引擎版本 2 中的支持相同。
- EXPLAIN ANALYZE – 在 `EXPLAIN ANALYZE` 的查询统计数据和输出中增加了物理输入读取时间。有关 `EXPLAIN ANALYZE` 的信息，请参阅在 Athena 中使用 EXPLAIN 和 EXPLAIN ANALYZE。
- INSERT – 改进了使用 `INSERT` 写入的表的查询性能。有关 `INSERT` 的信息，请参阅INSERT INTO。
- Delta Lake 表 - 更正了 Delta Lake 表上存在的 `DROP TABLE` 问题，该问题导致这些表在同时修改时无法完全删除。

2023 年 6 月 8 日

发布时间：2023 年 6 月 8 日

Amazon Athena for Apache Spark 宣布推出以下新功能。

- 支持自定义 Java 库和配置 - 现在，您可以在 Athena 中为 Apache Spark 会话使用自己的 Java 包和自定义配置。使用 Spark 属性通过 Athena 控制台、AWS CLI 或 Athena API 指定 `.jar` 文件、包或其他自定义配置。有关更多信息，请参阅 添加 JAR 文件和自定义 Spark 配置。

- 支持 Apache Hudi、Apache Iceberg 和 Delta Lake 表 - Athena for Spark 现在支持 Apache Iceberg、Apache Hudi 和 Linux Foundation Delta Lake 开源数据湖存储表格式。有关更多信息，请参阅 在 Amazon Athena for Apache Spark 中使用非 Hive 表格式 和针对在 Athena for Spark 中使用 Apache Iceberg、Apache Hudi 和 Linux Foundation Delta Lake 表的各个主题。
- 针对 Apache Spark 的加密支持 - 在 Athena for Spark 中，您现在可以对 Spark 节点之间的传输中数据以及 Spark 存储在磁盘上的本地静态数据启用加密。要启用 Spark 加密，可使用 Athena 控制台、AWS CLI 或 Athena API。有关更多信息，请参阅 启用 Apache Spark 加密。

有关 Amazon Athena for Apache Spark 的更多信息，请参阅 在 Amazon Athena 中使用 Apache Spark。

2023 年 6 月 2 日

发布时间：2023 年 6 月 2 日

现在，您可以在 Athena 中删除容量预留，并使用 AWS CloudFormation 模板指定 Athena 容量预留。

- 删除容量预留 - 您现在可以在 Athena 中删除已取消的容量预留。必须取消预留，然后才能将其删除。删除容量预留会立即从您的账户中移除该预留。无法再引用已删除的预留，包括通过其 ARN 进行引用。要删除预留，您可以使用 Athena 控制台或 Athena API。有关更多信息，请参阅《Amazon Athena 用户指南》中的 删除容量预留 和 Amazon Athena API 参考中的 DeleteCapacityReservation。
- 使用 AWS CloudFormation 模板进行容量预留 - 现在您可以使用 AWS CloudFormation 模板通过 `AWS::Athena::CapacityReservation` 资源指定 Athena 容量预留。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的 AWS::Athena::CapacityReservation。

有关在 Athena 中使用容量预留来配置容量的更多信息，请参阅 管理查询处理容量。

2023 年 5 月 25 日

发布时间：2023 年 5 月 25 日

Athena 已发布数据来源连接器更新，可提高联合查询性能。新的下推优化和动态筛选功能使更多操作可以在源数据库中执行，而不是在 Athena 中执行。这些优化功能减少了查询运行时间和扫描的数据量。这些改进功能需要采用 Athena 引擎版本 3。

以下连接器已更新：

- Azure Data Lake 存储

- Azure Synapse
- Cloudera Hive
- Cloudera Impala
- Db2
- DynamoDB
- Google BigQuery
- Hortonworks
- MySQL
- Oracle
- PostgreSQL
- Redshift
- SAP HANA
- Snowflake
- SQL Server
- Teradata

有关升级数据来源连接器的信息，请参阅 更新数据来源连接器。

2023 年 5 月 18 日

发布时间：2023 年 5 月 18 日

您现在可以使用 AWS PrivateLink 将 IPv6 入站连接到 Amazon Athena。

Amazon Athena 已将其对通过互联网协议版本 6（IPv6）端点的入站连接的支持范围扩大到包括 AWS PrivateLink。从今天开始，除了之前可用的公共 IPv6 端点外，您还可以通过 Amazon Virtual Private Cloud（Amazon VPC）使用 AWS PrivateLink 安全私密地连接到 Athena。

互联网的快速发展正在耗尽互联网协议版本 4（IPv4）地址的可用性。IPv6 可将可用地址的数量增加几倍，这样您就不必再管理您的 VPC 中的重叠地址空间了。在此版本中，您现在可以将 IPv6 寻址的优势与 AWS PrivateLink 的安全性和性能优势相结合。

要通过编程方式连接到 AWS 服务，您可以使用 AWS CLI 或 AWS SDK 来指定端点。有关服务端点和 Athena 服务端点的更多信息，请参阅 Amazon Web Services 一般参考 中的 AWS 服务端点和 Amazon Athena 端点和限额。

2023 年 5 月 15 日

发布时间：2023 年 5 月 15 日

Athena 宣布发布适用于 DynamoDB、CloudWatch Logs、CloudWatch Metrics 和 AWS CMDB 的 Apache Spark DataSourceV2（DSV2）连接器。使用新的 DSV2 连接器可通过 Spark 查询这些数据来源。DSV2 连接器使用的参数与其相应的 Athena 联合连接器相同。DSV2 连接器直接在 Spark 工作线程上运行，无需您部署 Lambda 函数即可使用它们。

有关更多信息，请参阅 适用于 Apache Spark 的 Athena 数据来源连接器。

2023 年 5 月 10 日

发布时间：2023 年 5 月 10 日

为 Athena 发布了 ODBC 1.1.20 驱动程序。

功能和增强功能：

- Lake Formation 端点覆盖支持。
- ADFS 身份验证插件有一个用于设置 Relying Party 值 (LoginToRP) 的新参数。
- AWS 库更新。

错误修复：

- 当 `SQLPrepare()` 方法提交失败时，准备好的语句取消分配失败。
- 将 C 类型转换为 SQL 类型时，绑定准备好的语句参数时出错。
- 当 `EXPLAIN` 和 `EXPLAIN ANALYZE` 查询使用 `SQLPrepare()` 和 `SQLExecute()` 时无法返回数据。

有关更多信息以及下载新驱动程序、版本注释和文档，请参阅 通过 ODBC 连接到 Amazon Athena。

2023 年 5 月 8 日

发布时间：2023 年 5 月 8 日

Athena 宣布推出以下修复和改进。

- 更新了 Hudi 集成 - Athena 更新了与 Apache Hudi 的集成。您现在可以使用 Athena 来查询 Hudi 0.12.2 表，现在还支持 Hudi 表的 Hudi 元数据列表。有关信息，请参阅使用 Athena 查询 Apache Hudi 数据集和Hudi 元数据列表。
- 时间戳转换修复 - 将时间戳转换的处理更正为精度较低的数据类型。以前，Athena 引擎版本 3 错误地将值四舍五入为目标类型，而不是在转换期间将其截断。

  以下示例说明了修复之前的错误处理。

  示例 1：从以微秒为单位的时间戳转换为毫秒

  示例数据

```
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
```

  以下查询尝试检索与特定值匹配的时间戳。

```
SELECT *
FROM table
WHERE timestamps.col = timestamp'2020-06-10 15:55:23.383'
```

  查询返回以下结果。

```
A, 2020-06-10 15:55:23.383
C, 2020-06-10 15:55:23.383
E, 2020-06-10 15:55:23.383
```

  在修复之前，Athena 没有包含值 `2020-06-10 15:55:23.383945` 或 `2020-06-10 15:55:23.383945278`，因为它们被四舍五入为 `2020-06-10 15:55:23.384`。

  示例 2：从时间戳转换为日期

  以下查询返回了错误的结果。

```
SELECT date(timestamp '2020-12-31 23:59:59.999')
```

结果

```
2021-01-01
```

在修复之前，Athena 将该值向上取整，从而向前推进一天。现在，这些值会被截断而不是向上取整。

2023 年 4 月 28 日

发布时间：2023 年 4 月 28 日

现在，您可以在 Amazon Athena 上使用容量预留对完全托管的计算容量运行 SQL 查询。

预置容量提供工作负载管理功能，帮助您优化、控制和扩展最重要的交互式工作负载。您可以随时添加容量以增加可并发运行的查询数量，控制可以使用容量的工作负载，并在工作负载之间共享容量。

有关更多信息，请参阅 管理查询处理容量。有关定价信息，请访问 Amazon Athena 定价页面。

2023 年 4 月 17 日

发布时间：2023 年 4 月 17 日

Athena 发布 JDBC 驱动程序版本 2.0.36。该驱动程序包含新的功能并解决了一个问题。

新功能

- 现在您可以在 AD FS 身份验证中使用可自定义的信赖方标识符。
- 现在您可以将使用连接器的应用程序的名称添加到用户代理字符串中。

已解决的问题

- 修复了使用 `getSchema()` 检索不存在的架构时发生的错误。

有关更多信息以及下载新驱动程序、版本注释和文档，请参阅 通过 JDBC 连接到 Amazon Athena。

2023 年 4 月 14 日

发布时间：2023 年 6 月 20 日

Athena 宣布推出以下修复和改进。

- 当您将字符串转换为时间戳时，需要在日期和时间或时区之间留出一个空格。有关更多信息，请参阅将字符串转换为时间戳时，日期和时间值之间需要空间。
- 删除了时间戳精度处理方式的重大变化。为了保持 Athena 引擎版本 2 和 Athena 引擎版本 3 之间的一致性，时间戳精度现在默认为毫秒而不是微秒。
- 现在，Athena 在运行查询时会始终强制其访问查询输出存储桶。请确保所有运行 StartQueryExecution 操作的 IAM 主体对查询输出存储桶都具有 S3:GetBucketLocation 权限。

2023 年 4 月 4 日

发布时间：2023 年 4 月 4 日

现在，您可以使用 Amazon Athena 在联合数据来源上创建和查询视图。使用单个联合视图查询多个外部表或数据子集。这简化了所需的 SQL，使您能够灵活地对必须使用 SQL 查询数据的最终用户的数据来源进行混淆处理。

有关更多信息，请参阅使用视图 和编写联合查询。

2023 年 3 月 30 日

发布时间：2023 年 3 月 30 日

Amazon Athena 宣布在其他 AWS 区域 推出 Amazon Athena for Apache Spark。

此版本将 Amazon Athena for Apache Spark 的可用性扩展到包括亚太地区（孟买）、亚太地区（新加坡）、亚太地区（悉尼）和欧洲地区（法兰克福）。

有关 Amazon Athena for Apache Spark 的更多信息，请参阅 在 Amazon Athena 中使用 Apache Spark。

2023 年 3 月 28 日

发布时间：2023 年 3 月 28 日

Athena 宣布推出以下修复和改进。

- 在对 `GetQueryExecution` 和 `BatchGetQueryExecution` Athena API 操作的响应中，新 `subStatementType` 字段显示运行的查询类型（例如：`SELECT`、`INSERT`、`UNLOAD`、`CREATE_TABLE` 或 `CREATE_TABLE_AS_SELECT`）。
- 修复了 Apache Hive 写入操作无法正确加密清单文件的问题。

- Athena 引擎版本 3 现在可以正确处理 approx_percentile 函数中的 NaN 值和 Infinity 值。approx_percentile 函数以定百分比返回数据集的近似百分位数。

  Athena 引擎版本 2 错误地将 NaN 视为大于 Infinity 的值。Athena 引擎版本 3 现在可以按照其他分析和统计函数中对这些值的处理方式处理 NaN 和 Infinity。以下几点将更详细地描述新行为。

  - 如果数据集中存在 NaN，则 Athena 将返回 NaN。
  - 如果 NaN 不存在，但 Infinity 存在，Athena 会将 Infinity 视为一个非常大的数字。
  - 如果存在多个 Infinity 值，Athena 会将它们视为同一个非常大的数字。如有必要，Athena 会输出 Infinity。
  - 如果单个数据集同时具有 Infinity 和 -Double.MAX_VALUE，且百分位数结果为 -Double.MAX_VALUE，则 Athena 返回 -Infinity。
  - 如果单个数据集同时具有 Infinity 和 Double.MAX_VALUE，且百分位数结果为 Double.MAX_VALUE，则 Athena 返回 Infinity。
  - 要从计算中排除 Infinity 和 NaN，请使用 is_finite() 函数，如下例所示。

    ```
    approx_percentile(x, 0.5) FILTER (WHERE is_finite(x))
    ```

2023 年 3 月 27 日

发布时间：2023 年 3 月 27 日

现在，您可以在 Amazon Athena 中指定 Athena SQL 工作组的最低加密级别。此功能可确保 Athena SQL 工作组中所有查询的结果都以您指定的加密级别或更高的加密级别进行加密。您可以从多个加密强度级别中进行选择，以保护您的数据。要配置所需的最低加密级别，您可以使用 Athena 控制台、AWS CLI、API 或 SDK。

最低加密功能不适用于启用 Apache Spark 的工作组。有关更多信息，请参阅 为工作组配置最低加密。

2023 年 3 月 17 日

发布时间：2023 年 3 月 17 日

Athena 宣布推出以下修复和改进。

- 修复了 Amazon Athena DynamoDB 连接器的一个问题，该问题导致查询失败，并显示错误消息 KeyConditionExpressions 每个键只能包含一个条件。

之所以出现此问题，是因为与 Athena 引擎版本 2 相比，Athena 引擎版本 3 识别了下推更多种类谓词的机会。在 Athena 引擎版本 3 中，像 `some_column LIKE 'someprefix%` 这样的子句作为筛选器谓词下推，这些谓词对给定列应用下限和上限。Athena 引擎版本 2 没有下推这些谓词。在 Athena 引擎版本 3 中，当 `some_column` 是排序键列时，引擎会将筛选器谓词下推到 DynamoDB 连接器。然后，筛选器谓词会被进一步下推到 DynamoDB 服务。由于 DynamoDB 在排序键上不支持多个筛选器，因此 DynamoDB 会返回错误。

要更正此问题，请将您的 Amazon Athena DynamoDB 连接器更新到版本 2023.11.1。有关更新连接器的说明，请参阅 更新数据来源连接器。

2023 年 3 月 8 日

发布时间：2023 年 3 月 8 日

Athena 宣布推出以下修复和改进。

- 修复了联合查询的一个问题，该问题导致时间戳谓词值以微秒而不是毫秒的形式发送。

2023 年 2 月 15 日

发布时间：2023 年 2 月 15 日

Athena 宣布推出以下修复和改进。

- 现在，您可以使用客户端加密对 Amazon S3 中的数据进行加密，以进行 Iceberg 写入操作。
- 修复了影响 Amazon S3 中针对 Iceberg 写入操作的服务器端加密的问题。

2023 年 1 月 31 日

发布时间：2023 年 1 月 31 日

您现在可以使用 Amazon Athena 来查询 Google Cloud Storage 中的数据。与 Amazon S3 一样，Google Cloud Storage 是一种在存储桶中存储数据的托管式服务。使用适用于 Google Cloud Storage 的 Athena 连接器对外部数据运行交互式联合身份查询。

有关更多信息，请参阅 Amazon Athena Google Cloud Storage 连接器。

2023 年 1 月 20 日

发布时间：2023 年 1 月 20 日

您现在可以查看有关 Athena 压缩支持的扩展文档。分别增加了有关 Hive 表压缩、Iceberg 表压缩 和 ZSTD 压缩级别 的单独主题。

有关更多信息，请参阅 Athena 压缩支持。

2023 年 1 月 3 日

发布时间：2023 年 1 月 3 日

Athena 宣布推出以下更新：

- Hive 元存储的其他命令 – 您可以使用 Athena 连接到作为元数据目录自行管理的 Apache Hive 元存储，并查询存储在 Amazon S3 中的数据。在此版本中，您可以使用 `CREATE TABLE AS`（CTAS）、`INSERT INTO` 和其他 12 个数据定义语言（DDL）命令与 Apache Hive 元存储进行交互。您可以使用这组扩展的 SQL 功能直接从 Athena 管理 Hive Metastore 架构。

  有关更多信息，请参阅 将 Athena 数据连接器用于外部 Hive 元数据仓。
- JDBC 驱动程序版本 2.0.35 – Athena 发布了 JDBC 驱动程序版本 2.0.35。JDBC 2.0.35 驱动程序包含以下更新：
  - 该驱动程序现在将以下库用于 Jackson JSON 解析器。
    - jackson-annotations 2.14.0（以前为 2.13.2）
    - jackson-core 2.14.0（以前为 2.13.2）
    - jackson-databind 2.14.0（以前为 2.13.2.2）
  - 已停止支持 JDBC 版本 4.1。

  有关更多信息以及下载新驱动程序、发布说明和文档，请参阅 通过 JDBC 连接到 Amazon Athena。

2022 年的 Athena 发布说明

2022 年 12 月 14 日

发布时间：2022 年 12 月 14 日

现在，您可以使用适用于 Kafka 的 Amazon Athena 连接器，对流数据运行 SQL 查询。例如，您可以在 Amazon Managed Streaming for Apache Kafka（Amazon MSK）中对实时流数据运行分析查询，并将其与 Amazon S3 数据湖中的历史数据结合。

适用于 Kafka 的 Amazon Athena 连接器支持在多个流引擎上进行查询。您可以使用 Athena，在 Amazon MSK 预置集群和无服务器集群、自行管理的 Kafka 部署以及 Confluent Cloud 中的流数据上运行 SQL 查询。

有关更多信息，请参阅 Amazon Athena MSK 连接器。

2022 年 12 月 2 日

发布时间：2022 年 12 月 2 日

Athena 发布了 JDBC 驱动程序版本 2.0.34。JDBC 2.0.34 驱动程序包括以下新功能和已解决的问题：

- 查询结果重用支持 – 现在，您可以在指定的时间限制内重用先前执行的查询的结果，而不必在每次运行查询时都让 Athena 重新计算结果。有关更多信息，请参阅 Installation and Configuration Guide（《安装和配置指南》）（可从 JDBC 下载页面获取）和 重用查询结果。
- Ec2InstanceMetadata 支持 – JDBC 驱动程序现在支持使用 IAM 实例配置文件的 Ec2InstanceMetadata 身份验证方法。
- 基于字符的异常修复 – 修复了包含某些语言字符的查询出现的异常。
- 漏洞修复 – 更正了与 AWS 依赖项（打包有连接器）相关的漏洞。

有关更多信息以及下载新驱动程序、版本注释和文档，请参阅 通过 JDBC 连接到 Amazon Athena。

2022 年 11 月 30 日

发布时间：2022 年 11 月 30 日

现在，您可以在 Athena 上以交互方式创建和运行 Apache Spark 应用程序以及与 Jupyter 兼容的笔记本。使用 Spark 在 Athena 上运行数据分析，无需规划、配置或管理资源。提交 Spark 代码进行处理，然后直接接收结果。使用 Amazon Athena 控制台中简化的笔记本体验，以通过 Python 或 Athena 笔记本 API 开发 Apache Spark 应用程序。

Amazon Athena 上的 Apache Spark 无服务器，可通过提供即时计算实现自动按需扩展，从而满足不断变化的数据卷和处理要求。

有关更多信息，请参阅 在 Amazon Athena 中使用 Apache Spark。

2022 年 11 月 18 日

发布时间：2022 年 11 月 18 日

现在，您可以使用适用于 IBM Db2 的 Amazon Athena 连接器，查询来自 Athena 的 Db2。例如，您可以通过 Db2 上的数据仓库和 Amazon S3 中的数据湖运行分析查询。

Amazon Athena Db2 连接器通过 Lambda 环境变量显示了多个配置选项。有关配置选项、参数、连接字符串、部署和限制的信息，请参阅 Amazon Athena IBM Db2 连接器。

2022 年 11 月 17 日

发布时间：2022 年 11 月 17 日

Athena 引擎版本 3 中的 Apache Iceberg 支持现在提供以下增强的 ACID 事务功能：

- ORC 和 Avro 支持 – 使用 Apache Avro 和 Apache ORC 基于行和列的文件格式创建 Iceberg 表。对这些格式的支持是对 Parquet 现有支持的补充。
- MERGE INTO – 使用 `MERGE INTO` 命令可以有效实现大规模数据合并。`MERGE INTO` 将 `INSERT`、`UPDATE` 和 `DELETE` 操作合并为一个事务。这不仅可以减少数据管道中的处理开销，还可以减少 SQL 编写开销。有关更多信息，请参阅更新 Iceberg 表数据 和MERGE INTO。
- CTAS 和 VIEW 支持 – 将 `CREATE TABLE AS SELECT`（CTAS）和 `CREATE VIEW` 语句与 Iceberg 表结合使用。有关更多信息，请参阅CREATE TABLE AS 和CREATE VIEW。
- VACUUM 支持 – 您可以使用 `VACUUM` 语句，通过删除不再需要的快照和数据对数据湖进行优化。您可以使用此功能来提高读取性能并满足 GDPR 等法规要求。有关更多信息，请参阅优化 Iceberg 表和VACUUM。

这些新功能需要 Athena 引擎版本 3，并且可在支持 Athena 的所有区域使用。您可以将其与 Athena 控制台、驱动程序或 API 结合使用。

有关使用 Athena 中 Iceberg 的信息，请参阅 使用 Apache Iceberg 表。

2022 年 11 月 14 日

发布时间：2022 年 11 月 14 日

Amazon Athena 现在支持 IPv6 端点进行入站连接，您可以使用这些端点通过 IPv6 调用 Athena 函数。您可以使用此功能来满足 IPv6 合规性要求，以及处理 IPv4 和 IPv6 之间的地址转换，而无需其他网络设备。

要使用此功能，请将应用程序配置为使用新 Athena 双堆栈端点（支持 IPv4 和 IPv6）。双堆栈端点使用格式 `athena.region.api.aws`。例如，美国东部（弗吉尼亚州北部）区域中的双堆栈端点为 `athena.us-east-1.api.aws`。

当您向双堆栈 Athena 端点发出请求时，端点会解析为 IPv6 或 IPv4 地址，具体取决于您的网络和客户端使用的协议。要通过编程方式连接到 AWS 服务，您可以使用 AWS CLI 或 AWS SDK 来指定端点。

有关服务端点的更多信息，请参阅 AWS 服务端点。要了解有关 Athena 服务端点的更多信息，请参阅 AWS 文档中的 Amazon Athena 端点和配额。

您可以使用新的 Athena 双堆栈端点进行入站连接，这不会产生额外的费用。双堆栈端点已在全部 AWS 区域 中正式发布。

2022 年 11 月 11 日

发布时间：2022 年 11 月 11 日

Athena 宣布推出以下修复和改进。

- 扩展的 Lake Formation 精细访问控制 – 现在，您可以在 Athena 查询中使用 AWS Lake Formation 精细访问控制策略，来查询以任何受支持的文件或表格格式存储的数据。您可以在 Lake Formation 中使用精细访问控制，通过数据筛选条件来限制对查询结果中的数据访问，从而实现列级、行级和单元级安全性。Athena 中支持的表格格式包括 Apache Iceberg、Apache Hudi 和 Apache Hive。扩展的精细访问控制可在 Athena 支持的所有区域中使用。扩展的表和文件格式支持需要 Athena 引擎版本 3，它可提供新功能和改进的查询性能，但不会改变在 Lake Formation 中设置精细访问控制策略的方式。

  在 Athena 中使用这种扩展的精细访问控制时，请注意以下事项：

  - EXPLAIN – 在 Lake Formation 中定义的行或单元格筛选信息以及查询统计信息未在 `EXPLAIN` 和 `EXPLAIN ANALYZE` 的输出中显示。有关 Athena 中 `EXPLAIN` 的信息，请参阅 在 Athena 中使用 EXPLAIN 和 EXPLAIN ANALYZE。
  - 外部 Hive 元存储 – Apache Hive 隐藏列不能用于精细访问控制筛选，精细访问控制不支持 Apache Hive 的隐藏系统表。有关更多信息，请参阅 注意事项和限制主题中的 将 Athena 数据连接器用于外部 Hive 元数据仓。
  - 查询统计信息 – 如果查询具有在 Lake Formation 中定义的行级筛选条件，Athena 查询统计信息中不会显示阶段级输入和输出行数以及数据大小信息。有关查看 Athena 查询统计信息的信息，请参阅 查看已完成查询的统计数据和执行详细信息 和 GetQueryRuntimeStatistics。

- 工作组 – 同一 Athena 工作组中的用户可以查看由 Lake Formation 精细访问控制配置为可供工作组访问的数据。有关使用 Athena 查询注册到 Lake Formation 的数据的信息，请参阅 使用 Athena 查询向 AWS Lake Formation 注册的数据。

  有关在 Lake Formation 中使用精细访问控制的信息，请参阅 AWS 大数据博客中的使用 AWS Lake Formation 管理精细访问控制。
- Athena 联合查询 – Athena 联合查询现在保留 `struct` 对象中字段名称的原始大小写。以前，`struct` 字段名称会自动变为小写。

2022 年 11 月 8 日

发布时间：2022 年 11 月 8 日

现在，您可以使用查询结果重用缓存功能来加速 Athena 中的重复查询。重复查询是一种 SQL 查询，与最近提交的查询相同，会产生相同的结果。当您需要运行相同的多个查询时，结果重用缓存可以减少生成结果所需的时间。结果重用缓存还可以减少扫描的字节数，从而降低成本。

有关更多信息，请参阅 重用查询结果。

2022 年 10 月 13 日

发布时间：2022 年 10 月 13 日

Athena 宣布推出 Athena 引擎版本 3。

Athena 升级了其 SQL 查询引擎，以包含 Trino 开源项目的最新功能。除支持 Athena 引擎版本 2 的所有功能外，Athena 引擎版本 3 还包括 50 多个新的 SQL 函数、30 个新功能和 90 多项查询性能改进。借助今天的发布，Athena 还引入了一种用于开源软件管理的持续集成方法，以提高与 Trino 和 Presto 项目同步的能力，从而让您能够更快地获得在 Athena 引擎中集成和优化的社区改进。

有关更多信息，请参阅 Athena 引擎版本 3。

2022 年 10 月 10 日

发布时间：2022 年 10 月 10 日

Athena 发布了 JDBC 驱动程序版本 2.0.33。JDBC 2.0.33 驱动程序包括以下更改：

- 在凭证提供程序类的用户代理字符串中添加了新的驱动程序版本、JDBC 版本和插件名称属性。
- 更正了错误消息并添加了必要的信息。

- 现在，如果连接关闭或 Athena 预编译语句执行失败，则取消预编译语句的分配。

有关更多信息以及下载新驱动程序、版本注释和文档，请参阅 通过 JDBC 连接到 Amazon Athena。

2022 年 9 月 23 日

发布时间：2022 年 9 月 26 日

Amazon Athena Neptune 连接器现在支持不区分大小写的列和表名称匹配。

- Neptune 数据来源连接器可以解析 Neptune 表上区分大小写的列名称，即使 AWS Glue 中表的列名称均为小写。要启用此行为，请在 Neptune 连接器 Lambda 函数上将 `enable_caseinsensitivematch` 环境变量设置为 `true`。
- 由于 AWS Glue 仅支持小写表名称，因此在为 Neptune 创建 AWS Glue 表时，请指定 AWS Glue 表参数 `"glabel" = table_name`。

有关 Neptune 连接器的更多信息，请参阅 Amazon Athena Neptune 连接器。

2022 年 9 月 13 日

发布时间：2022 年 9 月 13 日

Athena 宣布推出以下修复和改进。

- 外部 Hive 元存储 — 当 `WHERE` 子句包含外部 Hive 元存储 (EHMS) 中不存在的分区时，Athena 现在将返回 `NULL` 而不是引发异常。新行为与 AWS Glue Data Catalog 的行为匹配。
- 参数化查询 — 参数化查询中的值现在可以转换为 `DOUBLE` 数据类型。
- Apache Iceberg — 现在，当 Amazon S3 存储桶上启用对象锁定时，对 Iceberg 表的写入操作会成功。

2022 年 8 月 31 日

发布时间：2022 年 8 月 31 日

Amazon Athena 宣布 Athena 和其功能在亚太地区（雅加达）区域可用。

此版本扩展了 Athena 在亚太地区的可用性，包括亚太地区（香港）、亚太地区（雅加达）、亚太地区（孟买）、亚太地区（大阪）、亚太地区（首尔）、亚太地区（新加坡）、亚太地区（悉尼）和亚太

地区（东京）。有关在这些区域和其他区域提供的 AWS 服务 的完整列表，请参阅 AWS 区域 服务列表。

2022 年 8 月 23 日

发布时间：2022 年 8 月 23 日

Athena Query Federation SDK 发行版 v2022.32.1 包含以下更改：

- 在 Amazon Athena Oracle 数据来源连接器中增加了对基于 SSL 的 Amazon RDS 实例连接的支持。支持仅限于传输层安全性协议（TLS）以及客户端对服务器的身份验证。由于 Amazon RDS 不支持相互身份验证，因而此更新不包括对相互身份验证的支持。

  有关更多信息，请参阅 Amazon Athena Oracle 连接器。

2022 年 8 月 3 日

发布时间：2022 年 8 月 3 日

Athena 发布了 JDBC 驱动程序版本 2.0.32。JDBC 2.0.32 驱动程序包括以下更改：

- 发送到 Athena SDK 的 `User-Agent` 字符串已扩展为包含驱动程序版本、JDBC 规范版本和身份验证插件名称。
- 修复了没有为 `CheckNonProxyHost` 参数提供任何值时会出现的 `NullPointerException`。
- 修复了 BrowserSaml 身份验证插件中的 `login_url` 解析问题。
- 修复了将 `UseProxyforIdp` 参数设置为 `true` 时出现的代理主机问题。

有关更多信息以及下载新驱动程序、版本注释和文档，请参阅 通过 JDBC 连接到 Amazon Athena。

2022 年 8 月 1 日

发布时间：2022 年 8 月 1 日

Athena 宣布改进 Athena 查询联合身份验证 SDK 和 Athena 预构建的数据源连接器。改进包括以下方面：

- 结构解析 — 修复了 Athena 查询联合身份验证 SDK 中的 `GlueFieldLexer` 解析问题，该总是导致某些复杂结构无法显示其所有数据。此问题影响了基于 Athena 联合身份验证 SDK 构建的连接器。
- AWS Glue 表 — 增加了对 AWS Glue 表中 `set` 和 `decimal` 列类型的额外支持。

- DynamoDB 连接器 — 增加了忽略 DynamoDB 属性名称大小写的功能。有关更多信息，请参阅 Amazon Athena DynamoDB 连接器 页面 参数 一节中的 `disable_projection_and_casing`。

有关更多信息，请参阅 GitHub 上的 Athena 查询联合身份验证版本 v2022.30.2。

2022 年 7 月 21 日

发布时间：2022 年 7 月 21 日

现在，您可以在 Athena 控制台中使用性能指标和交互式、可视化查询分析工具来分析和调试查询。查询性能数据和执行详细信息可以帮助您识别查询中的瓶颈，查看查询的每个阶段的运算符和统计信息，跟踪各个阶段之间的数据流量，并验证查询谓词的影响。现在，您可以：

- 只需单击一下即可访问查询的分布式和逻辑执行计划。
- 在阶段开始运行之前探索每个阶段的操作。
- 通过排队、计划和执行阶段所用时间的指标，直观显示已完成查询的性能。
- 获取有关查询处理和输出的行数和源数据量的信息。
- 查看在上下文中呈现并格式化为交互式图表的查询的精细执行详细信息。
- 使用精确的阶段级执行详细信息来了解查询中的数据流。
- 使用同样在今天发布的新 API 以编程方式分析查询性能数据以获取查询运行时统计数据。

要了解如何在查询中使用这些功能，请观看 AWS YouTube 频道上的视频教程使用新的查询分析工具优化 Amazon Athena 查询。

有关文档，请参阅 查看 SQL 查询的执行计划 和 查看已完成查询的统计数据和执行详细信息。

2022 年 7 月 11 日

发布时间：2022 年 7 月 11 日

现在，您可以直接从 Athena 控制台或 API 运行参数化查询，而无需提前准备 SQL 语句。

现在，当您在 Athena 控制台中运行问号形式的参数的查询时，用户界面会提示您直接输入参数值。这样就无需在每次运行查询时都在查询编辑器中修改文字值。

如果您使用增强版查询执行 API，现在可以在单次调用中提供执行参数及其值。

有关更多信息，请参阅本用户指南中的 使用参数化查询 和 AWS 大数据博客文章使用 Amazon Athena 参数化查询以将数据作为服务提供。

2022 年 7 月 8 日

发布时间：2022 年 7 月 8 日

Athena 宣布推出以下修复和改进。

- 修复了导致查询失败的 SageMaker 终端节点（UDF）的 DATE 列转换处理问题。

2022 年 6 月 6 日

发布时间：2022 年 6 月 6 日

Athena 发布了 JDBC 驱动程序版本 2.0.31。JDBC 2.0.31 驱动程序包括以下更改：

- log4j 依赖关系问题 – 解决了因 log4j 依赖关系引起的 Cannot find driver class（找不到驱动程序类）错误消息。

有关更多信息以及下载新驱动程序、版本注释和文档，请参阅 通过 JDBC 连接到 Amazon Athena。

2022 年 5 月 25 日

发布时间：2022 年 5 月 25 日

Athena 宣布推出以下修复和改进。

- Iceberg 支持
  - 推出对跨区域查询的支持。您现在可以查询与您所用 AWS 区域不同的 AWS 区域 中的 Iceberg 表。中国区域不支持跨区域查询。
  - 推出对服务器端加密配置的支持。您现在可以使用 SSE-S3/SSE-KMS 来加密 Amazon S3 中的 Iceberg 写操作数据。

  有关在 Athena 中使用 Apache Iceberg 的更多信息，请参阅使用 Apache Iceberg 表。
- JDBC 2.0.30 驱动程序发行版

  适用于 Athena 的 JDBC 2.0.30 驱动程序包含以下改进：
  - 修复了影响参数化编写的语句的数据竞争问题。
  - 修复了 Gradle 构建环境中出现的应用程序启动问题。

  要下载 JDBC 2.0.30 驱动程序、发布说明和文档，请参阅通过 JDBC 连接到 Amazon Athena。

2022 年 5 月 6 日

发布时间：2022 年 5 月 6 日

发布了适用于 Athena 的 JDBC 2.0.29 和 ODBC 1.1.17 驱动程序。

这些驱动程序包括以下更改：

- 更新了 SAML 插件浏览器启动过程。

有关这些更改的更多信息以及下载新驱动程序、版本注释和文档，请参阅 通过 JDBC 连接到 Amazon Athena 和 通过 ODBC 连接到 Amazon Athena。

2022 年 4 月 22 日

发布时间：2022 年 4 月 22 日

Athena 宣布推出以下修复和改进。

- 修复了使用满足以下条件时分区缓存的分区索引和筛选功能问题：
  - `partition_filtering.enabled` 密钥已在表的 AWS Glue 表属性中设置为 `true`。
  - 同一个表被多次使用，但使用了不同的分区筛选值。

2022 年 4 月 21 日

发布时间：2022 年 4 月 21 日

您现在可以使用 Amazon Athena 对新数据源运行联合查询，这些数据源包括 Google BigQuery、Azure Synapse 和 Snowflake。新数据源连接器包括：

- 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

有关 Athena 支持的数据来源的完整列表，请参阅 可用数据来源连接器。

为了更轻松地浏览可用源并连接到数据，您现在可以从 Athena 控制台中已更新的 Data Sources（数据源）屏幕进行搜索、排序和筛选可用连接器。

要了解有关查询联合源的信息，请参阅 使用 Amazon Athena 联合查询 和 编写联合查询。

2022 年 4 月 13 日

发布时间：2022 年 4 月 13 日

Athena 发布了 JDBC 驱动程序版本 2.0.28。JDBC 2.0.28 驱动程序包括以下更改：

- JWT 支持 – 驱动程序现在支持使用 JSON Web 令牌（JWT）进行身份验证。有关将 JWT 与 JDBC 驱动程序一起使用的信息，请参阅安装和配置指南（可从 JDBC 驱动程序页面下载）。
- 更新了 Log4j 库 – JDBC 驱动程序现在使用以下 Log4j 库：
  - Log4j-api 2.17.1（以前为 2.17.0）
  - Log4j-core 2.17.1（以前为 2.17.0）
  - Log4j-jcl 2.17.2
- 其他改进 – 新驱动程序还包括以下改进和错误修复：
  - Athena 预编译语句功能现在可以通过 JDBC 获得。有关预编译语句的信息，请参阅 使用参数化查询。
  - Athena JDBC SAML 联合身份验证现已在中国区域正常运行。
  - 其他小改进。

有关更多信息以及下载新驱动程序、版本注释和文档，请参阅 通过 JDBC 连接到 Amazon Athena。

2022 年 3 月 30 日

发布时间：2022 年 3 月 30 日

Athena 宣布推出以下修复和改进。

- 跨区域查询 – 现在，您可以使用 Athena 跨 AWS 区域 查询位于 Amazon S3 存储桶中的数据，其中包括亚太地区（香港）、中东（巴林）、非洲（开普敦）和欧洲（米兰）。中国区域不支持跨区域查询。
  - 有关可以使用 Athena 的 AWS 区域 列表，请参阅 Amazon Athena 端点和配额。
  - 有关启用在默认情况下禁用的 AWS 区域 的信息，请参阅启用区域。
  - 有关跨区域查询的信息，请参阅 跨区域查询。

2022 年 3 月 18 日

发布时间：2022 年 3 月 18 日

Athena 宣布推出以下修复和改进。

- 动态筛选 - 动态筛选通过向相应表的每条记录高效应用筛选条件，实现了整数列的改进。
- Iceberg – 修复了在编写大于 2GB 的 Iceberg Parcia 文件时会导致故障的问题。
- 未压缩输出 - CREATE TABLE 语句现在支持写入未压缩的文件。要写入未压缩的文件，请使用以下语法：
  - CREATE TABLE（文本文件或 JSON）– 在 TBLPROPERTIES 中，请指定 `write.compression = NONE`。
  - CREATE TABLE (Parquet) – 在 TBLPROPERTIES 中，请指定 `parquet.compression = UNCOMPRESSED`。
  - CREATE TABLE (ORC) – 在 TBLPROPERTIES 中，请指定 `orc.compress = NONE`。
- 压缩 – 修复了插入文本文件表的问题，这些文件表在使用非默认压缩方法时以某种格式创建了压缩文件，但使用了另一种压缩格式文件扩展名。
- Avro – 修复了从 Avro 文件中读取固定类型的小数时出现的问题。

2022 年 3 月 2 日

发布时间：2022 年 3 月 2 日

Athena 发布了以下功能和改进功能。

- 您现在可以在查询结果存储桶启用了 ACL时向 Amazon S3 存储桶拥有者授予对查询结果的完全控制权。有关更多信息，请参阅 指定查询结果位置。

- 您现在可更新现有的命名查询。有关更多信息，请参阅 使用已保存的查询。

2022 年 2 月 23 日

发布时间：2022 年 2 月 23 日

Athena 宣布推出以下修复和性能改进。

- 改进了内存处理以提高性能并减少内存错误。
- Athena 现在可读取将时区信息存储在 Stripe 页脚中的 ORC 时间戳列，并且可写入页脚中包含时区（UTC）信息的 ORC 文件。如果要读取的 ORC 文件是在非 UTC 时区环境中创建的，则仅会影响 ORC 时间戳读取行为。
- 修复了会导致查询计划不理想的符号链接表大小估计不正确的问题。
- 现在可以通过 Athena 控制台从 Hive 元数据仓数据源查询横向爆炸视图。
- 改进了 Amazon S3 读取错误消息，以包含更详细的 Amazon S3 错误代码信息。
- 修复了会导致 ORC 格式的输出文件与 Apache Hive 3.1 不兼容的问题。
- 修复了会导致某些 DML 和 DDL 查询中带引号的表名失败的问题。

2022 年 2 月 15 日

发布时间：2022 年 2 月 15 日

Amazon Athena 增加了所有 AWS 区域的活动 DML 查询配额。活动查询包括正在运行的查询和已排队的查询。在此次更改后，您现在可以比以前拥有更多处于活动状态的 DML 查询。

有关 Athena 服务限额的信息，请参阅 服务限额。有关您使用 Athena 的区域的查询限额，请参阅 AWS 一般参考 中的 Amazon Athena 端点和限额。

要监控配额使用情况，您可以使用 CloudWatch 使用情况指标。Athena 会在 `AWS/Usage` 命名空间中发布 `ActiveQueryCount` 指标。有关更多信息，请参阅 监控 Athena 用量指标。

检查使用情况后，您可以通过 Service Quotas 控制台申请提高配额。如果您之前已经申请了增加账户的配额，只要申请的配额高于活动 DML 查询的新原定设置配额，则申请的配额继续有效。否则，所有账户都使用新的原定设置。

2022 年 2 月 14 日

发布时间：2022 年 2 月 14 日

此发行版在 Athena GetQueryExecution API 操作的 AthenaError 响应对象中增加了 ErrorType 子字段。

现有的 ErrorCategory 字段可指示失败查询的一般来源（系统、用户或其他），而新的 ErrorType 字段可提供有关所出现错误的更详细信息。结合这两个字段中的信息，可以深入洞察查询失败的原因。

有关更多信息，请参阅 Athena 错误目录。

2022 年 2 月 9 日

发布时间：2022 年 2 月 9 日

旧版 Athena 控制台不再可用。全新的 Athena 控制台支持早期版本控制台的所有功能，但具有更易于使用的现代化界面，并包括新功能，提升了制定查询、分析数据和管理使用情况的体验。要使用新的 Athena 控制台，请访问 https://console.aws.amazon.com/athena/。

2022 年 2 月 8 日

发布时间：2022 年 2 月 8 日

预期存储桶拥有者 – 作为额外的安全措施，您现在可以选择指定您希望其成为 Athena 中查询结果输出位置存储桶拥有者的 AWS 账户 ID。如果查询结果存储桶拥有者的账户 ID 与您指定的 ID 不匹配，则输出到存储桶的尝试将会失败，并出现 Amazon S3 权限错误。您可以在客户端或工作组级别进行此设置。

有关更多信息，请参阅 指定查询结果位置。

2022 年 1 月 28 日

发布时间：2022 年 01 月 28 日

Athena 发布以下引擎功能增强。

- Apache Hudi – 读取时合并 (MoR) 表上的快照查询现在可以读取具有 INT64 数据类型的时间戳列。
- UNION 查询 – 针对多次扫描同一个表的某些 UNION 查询提高性能并减少数据扫描。
- 分离查询 – 针对筛选条件中每个分区列仅具有分离值的查询提高性能。
- 分区投影增强功能

- 现在允许在 `injected` 类型的列筛选条件中使用多个分离值。有关更多信息，请参阅 注入的类型。
- 针对基于字符串类型且在筛选条件中仅具有分离值的列提高性能，例如 CHAR 或 VARCHAR。

2022 年 1 月 13 日

发布时间：2022 年 1 月 13 日

发布了适用于 Athena 的 JDBC 2.0.27 和 ODBC 1.1.15 驱动程序。

JDBC 2.0.27 驱动程序包括以下更改：

- 已更新驱动程序以检索外部目录。
- 扩展驱动程序版本号现在作为 Athena API 调用的一部分包含在 `user-agent` 字符串中。

ODBC 1.1.15 驱动程序包括以下更改：

- 更正第二次调用 `SQLParamData()` 时出现的问题。

有关这些更改的更多信息以及下载新驱动程序、版本注释和文档，请参阅 通过 JDBC 连接到 Amazon Athena 和 通过 ODBC 连接到 Amazon Athena。

2021 年的 Athena 发布说明

2021 年 11 月 26 日

发布时间：2021 年 11 月 26 日

Athena 发布了 Athena ACID 事务的公开预览版，该版本在 Athena 的 SQL 数据操作语言 (DML) 中增加了写入、删除、更新和时间旅行操作。Athena ACID 事务使多个并发用户能够对 Amazon S3 数据进行可靠的行级修改。Athena ACID 事务基于 Apache Iceberg 表格式而构建，与也支持 Iceberg 表格式的其他服务和引擎兼容，例如 Amazon EMR 和 Apache Spark。

Athena ACID 事务和熟悉的 SQL 语法简化了对业务和监管数据的更新。例如，要响应数据擦除请求，您可以执行 SQL `DELETE` 操作。要手动更正记录，您可以使用单个 `UPDATE` 语句。要恢复最近删除的数据，您可以使用 `SELECT` 语句发出时间旅行查询。可以通过 Athena 控制台、API 操作以及 ODBC 和 JDBC 驱动程序访问 Athena 事务。

有关更多信息，请参阅 使用 Athena ACID 事务。

2021 年 11 月 24 日

发布时间：2021 年 11 月 24 日

Athena 宣布支持读取和写入采用 ZStandard 压缩的 ORC、Parquet 和 textfile 数据。在写入采用 ZStandard 压缩的数据时，Athena 使用 ZStandard 压缩级别 3。

有关 Athena 中数据压缩的信息，请参阅 Athena 压缩支持。

2021 年 11 月 22 日

发布时间：2021 年 11 月 22 日

您现在可以从 Amazon Athena 控制台管理 AWS Step Functions 工作流，从而更轻松地构建可扩展的数据处理管道，基于自定义业务逻辑执行查询，自动执行管理和提示任务等。

Step Functions 现已与 Athena 的升级版控制台集成，您可以使用该控制台查看调用 Athena 的状态机的交互式工作流图。要开始使用，请从左侧导航面板中选择 Workflows（工作流）。如果您有带有 Athena 查询的现有状态机，请选择一个状态机以查看交互式工作流图。如果您是 Step Functions 新手，可以通过从 Athena 控制台启动示例项目并根据您的使用案例对其进行自定义入手。

有关更多信息，请参阅使用 Amazon Athena 和 AWS Step Functions 构建和编排 ETL 管道，或者查阅 Step Functions 文档。

2021 年 11 月 18 日

发布时间：2021 年 11 月 18 日

Athena 发布新功能和改进功能。

- 对于包含 DISTINCT、ORDER BY 或两者均含的聚合查询，支持溢出到磁盘，如以下示例所示：

```
SELECT array_agg(orderstatus ORDER BY orderstatus)
FROM orders
GROUP BY orderpriority, custkey
```

- 解决了使用 DISTINCT 的查询的内存处理问题。为了避免使用 DISTINCT 查询时显示类似以下内容的错误消息：Query exhausted resources at this scale factor（查询耗尽此缩放系数的资源），请选择 DISTINCT 基数较低的列，或者减小查询的数据大小。

- 在不会指定特定列的 SELECT COUNT(*) 查询中，现在仅保留计数而不进行行缓冲，从而提高了性能和内存使用率。
- 推出了以下字符串函数。
  - translate(source, from, to) – 返回 source 字符串，并将在 from 字符串中找到的字符替换为 to 字符串中的相应字符。如果 from 字符串包含重复项，则只使用第一项。如果 source 字符不存在于 from 字符串中，则复制 source 字符时不进行转换。如果 from 字符串中匹配字符的索引大于 to 字符串的长度，则生成的字符串中将省略该字符。
  - concat_ws(string0, array(varchar)) – 使用 string0 作为分隔符，返回数组中的一串元素。如果 string0 为 null，则返回值为 null。将跳过数组中的任何 null 值。
- 修复了在尝试访问 struct 中缺少的子字段时查询失败的错误。现在，查询针对缺少的子字段返回 null。
- 修复了十进制数据类型哈希不一致的问题。
- 修复了分区中有太多列时会导致资源耗尽的问题。

2021 年 11 月 17 日

发布时间：2021 年 11 月 17 日

Amazon Athena 现在支持分区索引，以加快对 AWS Glue Data Catalog 中分区表的查询。

在查询分区表时，Athena 会检索可用的表分区并筛选出与查询相关的子集。随着新数据和分区的添加，处理分区需要更长时间，查询运行时间可能会增加。为了优化分区处理并提升高度分区表的查询性能，Athena 现在支持 AWS Glue 分区索引。

有关更多信息，请参阅 AWS Glue 分区索引和筛选。

2021 年 11 月 16 日

发布时间：2021 年 11 月 16 日

全新的改进版 Amazon Athena 控制台现已在可用 Athena 的 AWS 商业区域和 GovCloud 区域正式发布。全新的 Athena 控制台支持早期版本控制台的所有功能，但具有更易于使用的现代化界面，并包括新功能，提升了制定查询、分析数据和管理使用情况的体验。现在，您可以：

- 从经过重新设计的查询选项卡栏重新排列、导航到或关闭多个查询选项卡。
- 改进了 SQL 和文本格式，可更轻松地读取和编辑查询。

- 除了下载完整的结果集之外，还可以将查询结果复制到剪贴板。
- 对查询历史记录、保存的查询和工作组进行排序，以及选择要显示或隐藏的列。
- 使用简化的界面配置数据源和工作组，只需单击几次即可完成。
- 设置显示查询结果、查询历史记录、换行等的首选项。
- 通过全新和改进的键盘快捷键和嵌入式产品文档提高工作效率。

在今日发布后，经过重新设计的控制台现为默认控制台。若要告诉我们您的体验，请选择控制台左下角的 Feedback（反馈）。

如果需要，您可以使用早期版本的控制台：登录到您的 AWS 账户，选择 Amazon Athena，然后从左侧导航面板中取消选择 New Athena experience（新 Athena 体验）。

2021 年 11 月 12 日

发布时间：2021 年 11 月 12 日

您现在可以使用 Amazon Athena 对位于您自己账户以外的 AWS 账户中的数据源运行联合查询。今天之前，查询这些数据需要数据源及其连接器才能使用与查询这些数据的用户相同的 AWS 账户。

作为数据管理员，您可以通过与数据分析师的账户共享数据连接器来启用跨账户联合查询。作为数据分析师，您可以将数据管理员与您共享的数据连接器添加到您的账户。对原始账户中连接器的配置更改将自动应用于共享连接器。

有关启用跨账户联合查询的信息，请参阅 启用跨账户联合查询。要了解有关查询联合源的信息，请参阅 使用 Amazon Athena 联合查询 和 编写联合查询。

2021 年 11 月 2 日

发布时间：2021 年 11 月 2 日

现在，您可以使用 Athena 中的 `EXPLAIN ANALYZE` 语句查看 SQL 查询的分布式执行计划以及每项操作的成本。

有关更多信息，请参阅 在 Athena 中使用 EXPLAIN 和 EXPLAIN ANALYZE。

2021 年 10 月 29 日

发布时间：2021 年 10 月 29 日

Athena 发布了 JDBC 2.0.25 和 ODBC 1.1.13 驱动程序，并推出了新功能和改进功能。

JDBC 和 ODBC 驱动程序

发布了适用于 Athena 的 JDBC 2.0.25 和 ODBC 1.1.13 驱动程序。这两个驱动程序都支持浏览器 SAML 多重验证，且可以配置为与任何 SAML 2.0 提供商配合使用。

JDBC 2.0.25 驱动程序包括以下更改：

- 支持浏览器 SAML 身份验证。该驱动程序包括浏览器 SAML 插件，可以配置为与任何 SAML 2.0 提供商配合使用。
- 支持 AWS Glue API 调用。您可以使用 `GlueEndpointOverride` 参数覆盖 AWS Glue 端点。
- 已将 `com.simba.athena.amazonaws` 类路径更改为 `com.amazonaws`。

ODBC 1.1.13 驱动程序包括以下更改：

- 支持浏览器 SAML 身份验证。该驱动程序包括浏览器 SAML 插件，可以配置为与任何 SAML 2.0 提供商配合使用。有关如何将浏览器 SAML 插件与 ODBC 驱动程序结合使用的示例，请参阅 使用 ODBC、SAML 2.0 和 Okta 身份提供商配置单点登录。
- 现在，当您使用 ADFS、Azure AD 或浏览器 Azure AD 进行身份验证时，可以配置角色会话持续时间。

有关这些更改和其他更改的更多信息以及下载新驱动程序、发布说明和文档，请参阅 通过 JDBC 连接到 Amazon Athena 和 通过 ODBC 连接到 Amazon Athena。

功能和改进

Athena 发布了以下功能和改进功能。

- 推出了新的优化规则，以避免在某些情况下重复表扫描。

2021 年 10 月 4 日

发布时间：2021 年 10 月 4 日

Athena 发布了以下功能和改进功能。

- SQL OFFSET – `SELECT` 语句现在支持 SQL `OFFSET` 子句。有关更多信息，请参阅 SELECT。

- CloudWatch 使用情况指标 – 现在，Athena 在 `AWS/Usage` 命名空间中发布了 `ActiveQueryCount` 指标。有关更多信息，请参阅 监控 Athena 用量指标。
- 查询计划 – 修复了在极少数情况下可能导致查询计划超时的错误。

2021 年 9 月 16 日

发布时间：2021 年 9 月 16 日

Athena 发布了以下新功能和改进功能。

功能

- 添加了对使用 `write_compression` 表属性在 CTAS 中指定文本文件和 JSON 压缩的支持。您也可以在 CTAS 中为 Parquet 和 ORC 格式指定 `write_compression` 属性。有关更多信息，请参阅 CTAS 表属性。
- 现在支持将 BZIP2 压缩格式用于编写文本文件和 JSON 文件。有关 Athena 中压缩格式的更多信息，请参阅 Athena 压缩支持。

改进

- 修复了无法将身份信息发送到 UDF Lambda 函数的错误。
- 修复了分离筛选条件的谓词下推问题。
- 修复了十进制类型的哈希问题。
- 修复了不必要的统计数据收集问题。
- 删除了不一致的错误消息。
- 通过在 Worker 节点中应用动态分区修剪，提高了广播联接性能。
- 对于联合查询：
  - 更改了配置，减少了联合查询中的 `CONSTRAINT_VIOLATION` 错误。

2021 年 9 月 15 日

发布时间：2021 年 9 月 15 日

您现在可以使用经过重新设计的 Amazon Athena 控制台（预览版）。发布了新的 Athena JDBC 驱动程序。

Athena 控制台预览版

现在，您可以从已推出 Athena 的任何 AWS 区域 区域使用经过重新设计的 Amazon Athena 控制台（预览版）。此全新控制台支持现有控制台的所有功能，但界面更易于使用、更现代化。

要切换到新控制台，请登录您的 AWS 账户，然后选择 Amazon Athena。从 AWS 控制台导航栏选择 Switch to the new console（切换到新控制台）。要返回到默认控制台，请从左侧导航面板中取消选择 New Athena experience（新 Athena 体验）。

立即开始使用新控制台。请选择左下角的 Feedback（反馈），告诉我们您的体验。

Athena JDBC 驱动程序 2.0.24

Athena 发布适用于 Athena 的 JDBC 驱动程序版本 2.0.24。此版本更新了对所有凭证提供商的代理支持。该驱动程序现在支持对 `NonProxyHosts` 连接属性不支持的所有主机进行代理身份验证。

为方便起见，此版本包括带有和不带有 AWS SDK 的 JDBC 驱动程序下载版本。此 JDBC 驱动程序版本支持同时将 AWS SDK 和 Athena JDBC 驱动程序嵌入在项目中。

有关更多信息以及要下载新驱动程序、发布说明和文档，请参阅 通过 JDBC 连接到 Amazon Athena。

2021 年 8 月 31 日

发布时间：2021 年 8 月 31 日

Athena 发布了以下功能增强和错误修复。

- Athena 联合增强 – Athena 添加了对映射类型的支持，并更好地支持作为Athena Query Federation 软件开发工具包一部分的复杂类型。此版本还包括一些内存增强功能和性能优化。
- 新错误类别 – 在错误消息中引入了 `USER` 和 `SYSTEM` 错误类别。这些类别可帮助您区分可以自行修复的错误 (`USER`) 和可能需要 Athena 支持帮助处理的错误 (`SYSTEM`)。
- 联合查询错误消息收发 – 已更新联合查询相关错误的 `USER_ERROR` 分类。
- JOIN – 修复了与溢出到磁盘相关的错误和内存问题，以提高性能并减少 `JOIN`操作中的内存错误。

2021 年 8 月 12 日

发布时间：2021 年 08 月 12 日

为 Athena 发布了 ODBC 1.1.12 驱动程序。此版本纠正了与 `SQLPrepare()`、`SQLGetInfo()` 和 `EndpointOverride` 相关的问题。

要下载新驱动程序、发布说明和文档，请参阅 通过 ODBC 连接到 Amazon Athena。

2021 年 8 月 6 日

发布时间：2021 年 8 月 6 日

Amazon Athena 宣布 Athena 和其功能在亚太地区（大阪）区域可用。

此版本扩展了 Athena 在亚太地区的可用性，包括亚太地区（香港）、亚太地区（孟买）、亚太地区（大阪）、亚太地区（首尔）、亚太地区（新加坡）、亚太地区（悉尼）和亚太地区（东京）。有关在这些区域和其他区域提供的 AWS 服务 的完整列表，请参阅 AWS 区域 服务列表。

2021 年 8 月 5 日

发布时间：2021 年 8 月 5 日

您可以使用 UNLOAD 语句将 SELECT 查询的输出写为 PARQUET、ORC、AVRO 和 JSON 格式。

有关更多信息，请参阅 UNLOAD。

2021 年 7 月 30 日

发布时间：2021 年 7 月 30 日

Athena 发布了以下功能增强和错误修复。

- 动态筛选和分区修剪 – 改进提高了性能，减少了在某些查询中扫描的数据量，如以下示例所示。

  此示例假定 Table_B 是一个未分区的表，其文件大小总量小于 20 MB。对于这样的查询，从 Table_A 读取的数据量较少，查询完成的速度也会更快。

  ```
  SELECT *
  FROM Table_A
  JOIN Table_B ON Table_A.date = Table_B.date
  WHERE Table_B.column_A = "value"
  ```

- ORDER BY with LIMIT，DISTINCT with LIMIT – 改进了使用 ORDER BY 或 DISTINCT 且后面跟一个 LIMIT 子句的查询的性能。
- S3 Glacier Deep Archive 文件 – 当 Athena 查询同时包含 S3 Glacier Deep Archive 文件和非 S3 Glacier 文件的表时，Athena 现在将为您跳过 S3 Glacier Deep Archive 文件。以前，您需要手动将

这些文件从查询位置移走，否则查询将失败。如果要使用 Athena 查询 S3 Glacier Deep Archive 存储中的对象，则必须还原这些文件。有关更多信息，请参阅《Amazon S3 用户指南》中的恢复已归档的对象。

- 修复了 CTAS `bucketed_by` 表属性创建的空文件未正确加密的错误。

2021 年 7 月 21 日

发布时间：2021 年 7 月 21 日

借助 2021 年 7 月发布的 Microsoft Power BI Desktop，您可以使用 Amazon Athena 的本机数据源连接器创建报表和控制面板了。Amazon Athena 的连接器可作为 Power BI 中的标准连接器使用，支持 DirectQuery，并支持通过 Power BI 网关对大型数据集进行分析和内容刷新。

由于连接器使用现有的 ODBC 数据源名称 (DSN) 连接到 Athena 并在其上运行查询，因此它需要 Athena ODBC 驱动程序。要下载最新的 ODBC 驱动程序，请参阅 通过 ODBC 连接到 Amazon Athena。

有关更多信息，请参阅 使用 Amazon Athena Power BI 连接器。

2021 年 7 月 16 日

发布时间：2021 年 7 月 16 日

Amazon Athena 已经更新了与 Apache Hudi 的集成。Hudi 是一种开源数据管理框架，用于简化 Amazon S3 数据湖中的增量数据处理。更新的集成使您能够使用 Athena 查询通过 Amazon EMR、Apache Spark、Apache Hive 或其他兼容服务管理的 Hudi 0.8.0 表。此外，Athena 现在还支持两个附加功能：对读取时合并 (MOR) 表进行快照查询，以及对引导启动表的读取支持。

Apache Hudi 提供记录级别的数据处理，可帮助您简化变更数据捕获 (CDC) 管道的开发，遵守以欧盟《一般数据保护条例（GDPR）》为宗旨的更新和删除操作，并更好地管理来自需要数据插入和事件更新的传感器或设备的流数据。0.8.0 版本可以更轻松地将大型 Parquet 表迁移到 Hudi，而无需复制数据，因此您可以通过 Athena 对其进行查询和分析。您可以使用 Athena 对快照查询的新支持来获得串流表更新的近实时视图。

要了解有关将 Hudi 与 Athena 结合使用的更多信息，请参阅 使用 Athena 查询 Apache Hudi 数据集。

2021 年 7 月 8 日

发布时间：2021 年 7 月 8 日

为 Athena 发布了 ODBC 1.1.11 驱动程序。ODBC 驱动程序现在可以使用 JSON Web Token (JWT) 对连接进行身份验证了。在 Linux 上，“Workgroup”属性的默认值已设置为“Primary”。

有关更多信息以及要下载新驱动程序、发布说明和文档，请参阅 通过 ODBC 连接到 Amazon Athena。

2021 年 7 月 1 日

发布时间：2021 年 7 月 1 日

2021 年 7 月 1 日，预览工作组的特殊处理结束。尽管 AmazonAthenaPreviewFunctionality 工作组仍保留了其名称，但其不再具有特殊地位。您可以继续使用 AmazonAthenaPreviewFunctionality 工作组来查看、修改、组织和运行查询。但是，使用以前在预览版本中功能的查询现在受标准 Athena 计费条款和条件的约束。有关定价的信息，请参阅 Amazon Athena 定价。

2021 年 6 月 23 日

发布时间：2021 年 6 月 23 日

为 Athena 发布了 JDBC 2.0.23 和 ODBC 1.1.10 驱动程序。这两个驱动程序都提供了更好的读取性能，并支持 EXPLAIN 语句和参数化查询。

EXPLAIN 语句显示 SQL 查询的逻辑或分布式执行计划。参数化查询使同一查询能够多次使用，并且在运行时提供不同的值。

JDBC 版本还添加了对 Active Directory Federation Services 2019 的支持，以及适用于 AWS STS 的自定义端点覆盖选项。ODBC 版本修复了 IAM 配置文件凭证的问题。

有关详细信息以及下载新驱动程序、发布说明和文档，请参阅 通过 JDBC 连接到 Amazon Athena 和 通过 ODBC 连接到 Amazon Athena。

2021 年 5 月 12 日

发布时间：2021 年 5 月 12 日

您现在可以使用 Amazon Athena 从除您自己以外的账户中注册一个 AWS Glue 目录了。在您为 AWS Glue 配置所需的 IAM 权限之后，就可以使用 Athena 运行跨账户查询。

有关更多信息，请参阅 从另一个账户注册 AWS Glue Data Catalog 和 授予 AWS Glue 数据目录跨账户访问权限。

2021 年 5 月 10 日

发布时间：2021 年 05 月 10 日

为 Athena 发布了 ODBC 驱动程序版本 1.1.9.1001。此版本修复了使用 Azure Active Directory (AD) 时的 `BrowserAzureAD` 身份验证类型问题。

要下载新驱动程序、发布说明和文档，请参阅 通过 ODBC 连接到 Amazon Athena。

2021 年 5 月 5 日

发布时间：2021 年 5 月 5 日

现在，您可以在联合查询中使用 Amazon Athena Vertica 连接器来查询来自 Athena 的 Vertica 数据源了。例如，您可以通过 Vertica 上的数据仓库和 Amazon S3 中的数据湖运行分析查询。

要部署 Athena Vertica 连接器，请访问 AWS Serverless Application Repository 中的 AthenaVerticaConnector 页面。

Amazon Athena Vertica 连接器通过 Lambda 环境变量显示了多个配置选项。有关配置选项、参数、连接字符串、部署和限制的信息，请参阅 Amazon Athena Vertica 连接器。

有关使用 Vertica 连接器的深入信息，请参阅 AWS 大数据博客中的使用 Athena 联合查询软件开发工具包 在 Amazon Athena 中查询 Vertica 数据源。

2021 年 4 月 30 日

发布时间：2021 年 4 月 30 日

为 Athena 发布了驱动程序 JDBC 2.0.21 和 ODBC 1.1.9。这两个版本都支持使用 Azure Active Directory (AD) 进行 SAML 身份验证，以及使用 PingFederate SAML 身份验证。JDBC 版本还支持参数化查询。有关 Athena 中的参数化查询的信息，请参阅 使用参数化查询。

要下载新驱动程序、发布说明和文档，请参阅 通过 JDBC 连接到 Amazon Athena 和 通过 ODBC 连接到 Amazon Athena。

2021 年 4 月 29 日

发布时间：2021 年 4 月 29 日

Amazon Athena 宣布在中国（北京）和中国（宁夏）区域推出 Athena 引擎版本 2。

有关 Athena 引擎版本 2 的更多信息，请参阅 Athena 引擎版本 2。

2021 年 4 月 26 日

发布时间：2021 年 4 月 26 日

Athena 引擎版本 2 中的窗口值函数现在支持 `IGNORE NULLS` 和 `RESPECT NULLS`。

有关更多信息，请参阅 Presto 文档中的值函数。

2021 年 4 月 21 日

发布时间：2021 年 4 月 21 日

Amazon Athena 宣布在欧洲（米兰）和非洲（开普敦）区域推出 Athena 引擎版本 2。

有关 Athena 引擎版本 2 的更多信息，请参阅 Athena 引擎版本 2。

2021 年 4 月 5 日

发布时间：2021 年 4 月 5 日

EXPLAIN 语句

现在，您可以使用 Athena 中的 `EXPLAIN` 语句查看 SQL 查询的执行计划。

有关更多信息，请参阅 在 Athena 中使用 EXPLAIN 和 EXPLAIN ANALYZE 和 了解 Athena EXPLAIN 语句结果。

在 SQL 查询中的 SageMaker 机器学习模型

使用 Amazon SageMaker 的机器学习模型推理现已在 Amazon Athena 中公开提供。在 SQL 查询中使用机器学习模型可通过在 SQL 查询中调用函数让复杂的任务（例如异常检测、客户群分析和时间序列预测）变得简单。

有关更多信息，请参阅 在 Amazon Athena 中使用机器学习（ML）。

用户定义的函数 (UDF)

用户定义的函数 (UDF) 现已在 Athena 中公开提供。使用 UDF 可以利用自定义函数来处理单个 SQL 查询中的记录或记录组。

有关更多信息，请参阅 使用用户定义函数进行查询。

2021 年 3 月 30 日

发布时间：2021 年 3 月 30 日

Amazon Athena 宣布在亚太地区（香港）和中东（巴林）区域推出 Athena 引擎版本 2。

有关 Athena 引擎版本 2 的更多信息，请参阅 Athena 引擎版本 2。

2021 年 3 月 25 日

发布时间：2021 年 3 月 25 日

Amazon Athena 宣布在欧洲（斯德哥尔摩）区域推出 Athena 引擎版本 2。

有关 Athena 引擎版本 2 的更多信息，请参阅 Athena 引擎版本 2。

2021 年 3 月 5 日

发布时间：2021 年 3 月 5 日

Amazon Athena 宣布在加拿大（中部）、欧洲（法兰克福）和南美洲（圣保罗）区域推出 Athena 引擎版本 2。

有关 Athena 引擎版本 2 的更多信息，请参阅 Athena 引擎版本 2。

2021 年 2 月 25 日

发布时间：2021 年 2 月 25 日

Amazon Athena 宣布在亚太地区（首尔）、亚太地区（新加坡）、亚太地区（悉尼）、欧洲（伦敦）和欧洲（巴黎）区域中推出 Athena 引擎版本 2。

有关 Athena 引擎版本 2 的更多信息，请参阅 Athena 引擎版本 2。

2020 年的 Athena 发布说明

2020 年 12 月 16 日

发布时间：2020 年 12 月 16 日

Amazon Athena 宣布在其他区域推出 Athena 引擎版本 2、Athena 联合查询和 AWS PrivateLink。

Athena 引擎版本 2 和 Athena 联合查询

Amazon Athena 宣布在亚太地区（孟买）、亚太地区（东京）、欧洲（爱尔兰）和美国西部（加利福尼亚北部）区域中推出 Athena 引擎版本 2 和 Athena 联合查询。Athena 引擎版本 2 和联合查询已在美国东部（弗吉尼亚北部）、美国东部（俄亥俄）、美国西部（俄勒冈）区域中提供。

有关更多信息，请参阅 Athena 引擎版本 2 和 使用 Amazon Athena 联合查询。

AWS PrivateLink

适用于 Athena 的 AWS PrivateLink 现已在欧洲（斯德哥尔摩）区域受支持。有关适用于 Athena 的 AWS PrivateLink 的信息，请参阅 使用接口 VPC 终端节点连接到 Amazon Athena。

2020 年 11 月 24 日

发布时间：2020 年 11 月 24 日

为 Athena 发布了驱动程序 JDBC 2.0.16 和 ODBC 1.1.6。这些版本在账户级别支持 Okta 验证多重身份验证 (MFA)。您还可以使用 Okta MFA 将 SMS 身份验证和 Google Authenticator 身份验证配置为验证因素。

要下载新驱动程序、发布说明和文档，请参阅 通过 JDBC 连接到 Amazon Athena 和 通过 ODBC 连接到 Amazon Athena。

2020 年 11 月 11 日

发布时间：2020 年 11 月 11 日

Amazon Athena 宣布在美国东部（弗吉尼亚北部）、美国东部（俄亥俄）和美国西部（俄勒冈）区域中推出 Athena 引擎版本 2 和联合查询。

Athena 引擎版本 2

Amazon Athena 宣布在美国东部（弗吉尼亚北部）、美国东部（俄亥俄）和美国西部（俄勒冈）区域中推出新查询引擎版本：Athena 引擎版本 2。

Athena 引擎版本 2 包括性能增强和新功能，例如对 Parquet 格式数据的架构演变支持、附加的地理空间函数、支持读取嵌套架构以降低成本以及 JOIN 和 AGGREGATE 操作中的性能增强。

- 有关改进、突破性更改和错误修复的信息，请参阅 Athena 引擎版本 2、
- 有关如何升级的信息，请参阅 更改 Athena 引擎版本。
- 有关测试查询的信息，请参阅 在引擎版本升级之前测试查询。

联合 SQL 查询

现在，您可以在美国东部（弗吉尼亚北部）、美国东部（俄亥俄）和美国西部（俄勒冈）区域中使用 Athena 的联合查询，而无需使用 `AmazonAthenaPreviewFunctionality` 工作组。

使用联合 SQL 查询跨关系数据来源、非关系数据来源、对象数据来源和自定义数据来源运行 SQL 查询。通过联合查询，您可以通过提交一个 SQL 查询，扫描来自本地运行或托管在云中的多个来源的数据。

由于如下原因，对分布在应用程序之间的数据进行分析可能很复杂且耗时：

- 分析所需的数据通常分布在关系、键值、文档、内存、搜索、图形、对象、时间序列和分类账数据存储中。
- 为了分析跨这些来源的数据，分析师需要构建复杂的管道，来将这些数据提取、转换和加载到数据仓库中，以便可以进行查询。
- 访问各种来源中的数据需要学习新的编程语言和数据访问构造。

Athena 中的联合 SQL 查询允许用户在不移动数据的情况下进行查询，消除了这种复杂性。分析师可以使用熟悉的 SQL 构造跨多个数据源 `JOIN` 数据以进行快速分析，并将结果存储在 Amazon S3 中以备随后使用。

数据来源连接器

要处理联合查询，Athena 会使用在 AWS Lambda 上运行的Athena 数据源连接器。以下开源、预构建的连接器由 Athena 编写和测试。使用它们在 Athena 中对其对应的数据源运行 SQL 查询。

- CloudWatch
- CloudWatch 指标
- DocumentDB
- DynamoDB
- OpenSearch
- HBase
- Neptune
- Redis
- Timestream
- TPC 基准 DS（TPC-DS）

自定义数据来源连接器

使用 Athena Query Federation 软件开发工具包，开发人员可以建立与任何数据源的连接器，以使 Athena 可以针对该数据源运行 SQL 查询。Athena Query Federation 连接器使得 AWS 提供的连接器之外的连接器也能够享受联合查询的优势。由于连接器在 AWS Lambda 上运行，因此您不必管理基础设施或计划扩展以应对尖峰需求。

后续步骤

- 要了解有关联合查询功能的详细信息，请参阅 使用 Amazon Athena 联合查询。
- 要开始使用现有连接器，请参阅部署连接器并连接到数据来源。
- 要了解如何使用 Athena Query Federation 软件开发工具包构建自己的数据源连接器，请参阅 GitHub 上的示例 Athena 连接器。

2020 年 10 月 22 日

发布时间：2020 年 10 月 22 日

您现在可以使用 AWS Step Functions 调用 Athena 了。AWS Step Functions 可以使用 Amazon States Language 直接控制特定的 AWS 服务。您可以将 Step 函数与 Athena 结合使用，以启动和停止查询执行、获取查询结果、运行临时或计划数据查询，以及从 Amazon S3 中的数据湖检索结果。

有关更多信息，请参阅《AWS Step Functions 开发人员指南》中的使用 Step Functions 调用 Athena。

2020 年 7 月 29 日

发布时间：2020 年 7 月 29 日

发布了 JDBC 驱动程序版本 2.0.13。此版本支持使用多个在 Athena 中注册的数据目录、用于身份验证的 Okta 服务以及与 VPC 终端节点的连接。

要下载和使用驱动程序的新版本，请参阅 通过 JDBC 连接到 Amazon Athena。

2020 年 7 月 9 日

发布时间：2020 年 7 月 9 日

Amazon Athena 添加了对查询压缩的 Hudi 数据集的支持，并添加了 AWS CloudFormation `AWS::Athena::DataCatalog` 资源，用于创建、更新或删除您在 Athena 中注册的数据目录。

查询 Apache Hudi 数据集

Apache Hudi是一个开源数据管理框架，可简化增量递增数据的处理。Amazon Athena 现在支持查询基于 Amazon S3 的数据湖中 Apache Hudi 数据集的读取优化视图。

有关更多信息，请参阅 使用 Athena 查询 Apache Hudi 数据集。

AWS CloudFormation Data Catalog 资源

要使用 Amazon Athena 的联合查询功能查询任何数据源，则必须首先在 Athena 中注册您的数据目录。现在，您可以使用 AWS CloudFormation `AWS::Athena::DataCatalog` 资源以创建、更新或删除您在 Athena 中注册的数据目录。

有关更多信息，请参阅《AWS CloudFormation 用户指南》中的 AWS::Athena::DataCatalog。

2020 年 6 月 1 日

发布时间：2020 年 6 月 1 日

通过 Amazon Athena 将 Apache Hive 元数据仓用作元目录

现在，您除了适用于 Athena 的 AWS Glue Data Catalog 之外，还可以将 Athena 连接到一个或多个 Apache Hive 元数据仓。

要连接到自托管的 Hive 元数据仓，您需要一个 Athena Hive 元数据仓连接器。Athena 提供了您可以使用的参考实施连接器。该连接器在您的账户中作为 AWS Lambda 函数运行。

有关更多信息，请参阅 将 Athena 数据连接器用于外部 Hive 元数据仓。

2020 年 5 月 21 日

发布时间：2020 年 5 月 21 日

Amazon Athena 增加了对分区投影的支持。使用分区投影可加快对高度分区表的查询处理，并自动执行分区管理。有关更多信息，请参阅 使用 Amazon Athena 分区投影。

2020 年 4 月 1 日

发布时间：2020 年 4 月 1 日

除了美国东部（弗吉尼亚北部）区域之外，Amazon Athena 联合查询、用户定义函数（UDF）、机器学习推理和外部 Hive 元数据仓功能现已在亚太地区（孟买）、欧洲（爱尔兰）和美国西部（俄勒冈）区域开放预览。

2020 年 3 月 11 日

发布时间：2020 年 3 月 11 日

Amazon Athena 现在为查询状态转换发布 Amazon EventBridge 事件。在多个状态间进行查询转换时（例如从“正在运行”状态转换至“成功”或“已取消”等最终状态），Athena 将向 EventBridge 发布一个查询状态变更事件。该事件包含有关查询状态转换的信息。有关更多信息，请参阅 使用 Amazon EventBridge 事件监控 Athena 查询。

2020 年 3 月 6 日

发布时间：2020 年 3 月 6 日

现在，您可以使用 AWS CloudFormation `AWS::Athena::WorkGroup` 资源来创建和更新 Amazon Athena 工作组。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的 AWS::Athena::WorkGroup。

2019 年的 Athena 发布说明

2019 年 11 月 26 日

发布时间：2019 年 12 月 17 日

Amazon Athena 添加了如下支持：跨关系数据源、非关系数据源、对象数据源和自定义数据源运行 SQL 查询；在 SQL 查询、用户定义函数 (UDF) 中调用机器学习模型（预览版）；通过 Amazon Athena 使用 Apache Hive 元数据仓作为元数据目录（预览版），以及四个附加查询相关指标。

联合 SQL 查询

使用联合 SQL 查询跨关系数据来源、非关系数据来源、对象数据来源和自定义数据来源运行 SQL 查询。

现在，您可以使用 Athena 的联合查询来扫描存储在关系数据源、非关系数据源、对象数据源和自定义数据源中的数据。通过联合查询，您可以通过提交一个 SQL 查询，扫描来自本地运行或托管在云中的多个来源的数据。

由于如下原因，对分布在应用程序之间的数据进行分析可能很复杂且耗时：

- 分析所需的数据通常分布在关系、键值、文档、内存、搜索、图形、对象、时间序列和分类账数据存储中。

- 为了分析跨这些来源的数据，分析师需要构建复杂的管道，来将这些数据提取、转换和加载到数据仓库中，以便可以进行查询。
- 访问各种来源中的数据需要学习新的编程语言和数据访问构造。

Athena 中的联合 SQL 查询允许用户在不移动数据的情况下进行查询，消除了这种复杂性。分析师可以使用熟悉的 SQL 构造跨多个数据源 `JOIN` 数据以进行快速分析，并将结果存储在 Amazon S3 中以备随后使用。

数据来源连接器

Athena 会使用在 AWS Lambda 上运行的Athena 数据源处理联合查询。使用这些开源数据源连接器在 Athena 中跨 Amazon DynamoDB、Apache HBase、Amazon Document DB、Amazon CloudWatch、Amazon CloudWatch Metrics 和符合 JDBC 规范的关系数据库（如 MySQL 和 Apache 2.0 许可证下的 PostgreSQL）运行联合 SQL 查询。

自定义数据来源连接器

使用 Athena Query Federation 软件开发工具包，开发人员可以建立与任何数据源的连接器，以使 Athena 可以针对该数据源运行 SQL 查询。Athena Query Federation 连接器使得 AWS 提供的连接器之外的连接器也能够享受联合查询的优势。由于连接器在 AWS Lambda 上运行，因此您不必管理基础设施或计划扩展以应对尖峰需求。

预览可用性

Athena 联合查询在美国东部（弗吉尼亚北部）区域中推出了预览版。

后续步骤

- 要开始预览，请按照 Athena 预览功能常见问题中的说明进行操作。
- 要了解有关联合查询功能的详细信息，请参阅使用 Amazon Athena 联合查询（预览版）。
- 要开始使用现有连接器，请参阅部署连接器并连接到数据来源。
- 要了解如何使用 Athena Query Federation 软件开发工具包构建自己的数据源连接器，请参阅 GitHub 上的示例 Athena 连接器。

在 SQL 查询中调用机器学习模型

您现在可以调用机器学习模型，以直接从您的 Athena 查询中获得推理。在 SQL 查询中使用机器学习模型可让复杂的任务（例如异常检测、客户群分析和销售预测）变得像在 SQL 查询中调用函数一样简单。

ML 模型

您可以使用 Amazon SageMaker 提供的十几种内置机器学习算法，训练自己的模型或从 AWS Marketplace 查找和订阅模型包并部署在 Amazon SageMaker Hosting Services 上。不需要其他设置。您可以通过 Athena 控制台、Athena API 和 Athena 的预览 JDBC 驱动程序在 SQL 查询中调用这些机器学习模型。

预览可用性

Athena 的机器学习功能现已在美国东部（弗吉尼亚北部）区域中提供预览版。

后续步骤

- 要开始预览，请按照 Athena 预览功能常见问题中的说明进行操作。
- 要了解有关机器学习功能的详细信息，请参阅将机器学习 (ML) 与 Amazon Athena（预览版）结合使用。

用户定义函数 (UDF)（预览版）

您可以编写自定义标量函数，并在您的 Athena 查询中调用它们。您可以使用 Athena Query Federation 软件开发工具包 在 Java 中编写 UDF。在提交到 Athena 的 SQL 查询中使用 UDF 时，将会在 AWS Lambda 上调用和执行该 UDF。UDF 可以同时在 SQL 查询的 `SELECT` 和 `FILTER` 子句中使用。您可以在同一查询中调用多个 UDF。

预览可用性

Athena UDF 功能在美国东部（弗吉尼亚北部）区域中提供预览版模式。

后续步骤

- 要开始预览，请按照 Athena 预览功能常见问题中的说明进行操作。
- 要了解更多信息，请参阅使用用户定义的函数进行查询（预览）。
- 有关 UDF 实现的示例，请参阅 GitHub 上的 Amazon Athena UDF Connector。
- 要了解如何使用 Athena Query Federation 软件开发工具包编写自己的函数，请参阅使用 Lambda 创建和部署 UDF。

通过 Amazon Athena（预览版）将 Apache Hive 元数据仓用作元目录

现在，您除了适用于 Athena 的 AWS Glue Data Catalog 之外，还可以将 Athena 连接到一个或多个 Apache Hive 元数据仓。

元数据仓连接器

要连接到自托管的 Hive 元数据仓，您需要一个 Athena Hive 元数据仓连接器。Athena 提供了您可以使用的参考实施连接器。该连接器在您的账户中作为 AWS Lambda 函数运行。有关更多信息，请参阅为外部 Hive 元数据仓使用 Athena 数据连接器（预览版）。

预览可用性

Hive 元数据仓功能在美国东部（弗吉尼亚北部）区域中在预览模式中可用。

后续步骤

- 要开始预览，请按照 Athena 预览功能常见问题中的说明进行操作。
- 要了解有关此功能的更多信息，请访问我们的将 Athena 数据连接器用于外部 Hive 元数据仓（预览版）。

与查询相关的新指标

Athena 现在发布其他查询指标，以帮助您了解 Amazon Athena 性能。Athena 将查询相关指标发布到了 Amazon CloudWatch。在此版本中，Athena 发布以下其他查询指标：

- Query Planning Time（查询计划时间）– 计划查询所花费的时间。这包括从数据源检索表分区所花费的时间。
- Query Queuing Time（查询队列时间）– 查询在队列中等待资源的时间。
- Service Processing Time（服务处理时间）– 查询引擎完成执行后写入结果所需的时间。
- Total Execution Time（总执行时间）– Athena 运行查询所花费的时间。

要使用这些新的查询指标，您可以创建自定义控制面板，在 CloudWatch 中设置指标的警报和触发器，或者直接从 Athena 控制台使用预填充的控制面板。

后续步骤

有关更多信息，请参阅使用 CloudWatch 指标监控 Athena 查询。

2019 年 11 月 12 日

发布时间：2019 年 12 月 17 日

Amazon Athena 现已在中东（巴林）区域提供。

2019 年 11 月 8 日

发布时间：2019 年 12 月 17 日

Amazon Athena 现已在美国西部（加利福尼亚北部）和欧洲（巴黎）区域推出。

2019 年 10 月 8 日

发布时间：2019 年 12 月 17 日

Amazon Athena 现在允许您通过 Virtual Private Cloud (VPC) 中的一个接口 VPC 终端节点直接连接到 Athena。利用此功能，您可以将您的查询安全提交到 Athena，而无需 VPC 中的互联网网关。

要创建接口 VPC 终端节点以连接到 Athena，您可以使用 AWS Management Console 或 AWS Command Line Interface (AWS CLI)。有关创建接口终端节点的信息，请参阅创建接口终端节点。

在使用接口 VPC 终端节点时，您的 VPC 与 Athena API 之间的通信是安全的，并且一直处于 AWS 网络中。使用此功能不会产生额外的 Athena 费用。适用接口 VPC 终端节点费用。

要了解有关此功能的更多信息，请参阅使用接口 VPC 终端节点连接到 Amazon Athena。

2019 年 9 月 19 日

发布时间：2019 年 12 月 17 日

Amazon Athena 添加了对使用 `INSERT INTO` 语句向现有表插入新数据的支持。您可以基于在源表上运行的 `SELECT` 查询语句，或基于作为查询语句的一部分提供的一组值，将新行插入到目标表中。支持的数据格式包括 Avro、JSON、ORC、Parquet 和文本文件。

`INSERT INTO` 语句还可以帮助您简化 ETL 流程。例如，您可以在单一查询中使用 `INSERT INTO` 从 JSON 格式的源表中选择数据，并以 Parquet 格式写入目标表。

`INSERT INTO` 语句将根据在 `SELECT` 阶段中扫描的字节数收费，类似于 Athena 对 `SELECT` 查询进行收费的方式。有关更多信息，请参阅 Amazon Athena 定价。

有关使用 `INSERT INTO` 的更多信息，包括支持的格式、SerDes 和示例，请参阅《Athena 用户指南》中的 INSERT INTO。

2019 年 9 月 12 日

发布时间：2019 年 12 月 17 日

Amazon Athena 现已在亚太地区（香港）区域中提供。

2019 年 8 月 16 日

发布时间：2019 年 12 月 17 日

Amazon Athena添加了对在 Amazon S3 申请方付款存储桶中查询数据的支持。

将 Amazon S3 存储桶配置为申请方付款时，申请方而非存储桶拥有者将支付 Amazon S3 请求和数据传输费用。在 Athena 中，工作组管理员现在可以配置工作组设置，以允许工作组成员查询 S3 申请方付款存储桶。

有关如何为您的工作组配置申请方付款设置的信息，请参阅《Amazon Athena 用户指南》中的创建工作组。有关申请方付款存储桶的更多信息，请参阅《Amazon Simple Storage Service 开发人员指南》中的申请方付款存储桶。

2019 年 8 月 9 日

发布时间：2019 年 12 月 17 日

Amazon Athena 现在支持强制执行 AWS Lake Formation 策略，用于对新数据库或现有数据库、表和列进行精细访问控制，这些数据库、表和列均是为存储在 Amazon S3 中存储的数据在 AWS Glue Data Catalog 中定义的。

您可以在以下 AWS 区域 使用此功能：美国东部（俄亥俄）、美国东部（弗吉尼亚北部）、美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、亚太地区（东京）和欧洲（爱尔兰）。使用此功能不会产生额外的费用。

有关使用此功能的更多信息，请参阅使用 Athena 查询向 AWS Lake Formation 注册的数据。有关 AWS Lake Formation 的更多信息，请参阅 AWS Lake Formation。

2019 年 6 月 26 日

Amazon Athena 现已在欧洲（斯德哥尔摩）区域推出。有关受支持的区域列表，请参阅 AWS 区域 和端点。

2019 年 5 月 24 日

发布时间：2019 年 5 月 24 日

Amazon Athena 现已在 AWS GovCloud（美国东部）和 AWS GovCloud（美国西部）区域提供。有关受支持的区域列表，请参阅 AWS 区域 和端点。

2019 年 3 月 5 日

发布时间：2019 年 3 月 5 日

Amazon Athena 现已在加拿大（中部）区域中提供。有关受支持的区域列表，请参阅 AWS 区域 和端点。发布新版本的 ODBC 驱动程序，支持 Athena 工作组。有关更多信息，请参阅 ODBC 驱动程序发行说明。

要下载 ODBC 驱动程序版本 1.0.5 及其文档，请参阅通过 ODBC 连接到 Amazon Athena。有关此版本的更多信息，请参阅 ODBC 驱动程序发行说明。

要将工作组与 ODBC 驱动程序结合使用，请在连接字符串中设置新连接属性 `Workgroup`，如以下示例所示：

```
Driver=Simba Athena ODBC
 Driver;AwsRegion=[Region];S3OutputLocation=[S3Path];AuthenticationType=IAM
 Credentials;UID=[YourAccessKey];PWD=[YourSecretKey];Workgroup=[WorkgroupName]
```

有关更多信息，请在《ODBC 驱动程序安装和配置指南版本 1.0.5》中搜索“工作组”。在工作组上使用标签时，对 ODBC 驱动程序连接字符串没有更改。要使用标签，请将 ODBC 驱动程序升级到最新版本，即此当前版本。

此驱动程序版本可让您使用 Athena API 工作组操作来创建和管理工作组，使用 Athena API 标签操作来添加、列出或删除工作组上的标签。在您开始之前，请确保您在 IAM 中拥有资源级别的权限以对工作组和标签执行操作。

有关更多信息，请参阅：

- 使用工作组运行查询 和 工作组策略示例
- 为 Athena 资源添加标签 和 基于标签的 IAM 访问控制策略

如果您使用 JDBC 驱动程序或 AWS 软件开发工具包，请升级到驱动程序和开发工具包的最新版本，这二者均已包含对 Athena 中工作组和标签的支持。有关更多信息，请参阅 通过 JDBC 连接到 Amazon Athena。

2019 年 2 月 22 日

发布时间：2019 年 2 月 22 日

Amazon Athena 中添加了工作组的标签支持。标签包含您定义的一个键和一个值。您在标记工作组时，将为其分配自定义元数据。您可以根据 AWS 标记最佳实践，将标签添加到工作组以帮助进行分类。您可以使用标签限制对工作组的访问，以及用于跟踪成本。例如，为每个成本中心创建一个工作组。然后，通过将标签添加到这些组，您可以跟踪每个成本中心的 Athena 支出。有关更多信息，请参阅《AWS Billing and Cost Management 用户指南》中的使用账单标签。

您可以通过 Athena 控制台或 API 操作来使用标签。有关更多信息，请参阅 为 Athena 资源添加标签。

在 Athena 控制台中，您可以将一个或多个标签添加到每个工作组并按标签进行搜索。工作组是 Athena 中 IAM 控制的资源。在 IAM 中，您可以限制哪些人可以在您创建的工作组上添加、删除或列出标签。您还可以使用具有可选标签参数的 `CreateWorkGroup` API 操作，向工作组添加一个或多个标签。要添加、删除或列出标签，请使用 `TagResource`、`UntagResource` 和 `ListTagsForResource`。有关更多信息，请参阅 使用标签操作。

要允许用户在创建工作组时添加标签，请确保您向每个用户授予 `TagResource` 和 `CreateWorkGroup` API 操作的 IAM 权限。有关更多信息以及示例，请参阅 基于标签的 IAM 访问控制策略。

在工作组上使用标签时，对 JDBC 驱动程序没有更改。如果您创建新的工作组，并使用 JDBC 驱动程序或 AWS 软件开发工具包，则需要升级到最新版本的驱动程序和开发工具包。有关信息，请参阅通过 JDBC 连接到 Amazon Athena。

2019 年 2 月 18 日

发布时间：2019 年 2 月 18 日

添加了通过在工作组中运行查询来控制成本的功能。有关信息，请参阅使用工作组控制查询访问和成本。改进了在 Athena 中使用的 JSON OpenX SerDe，修复了 Athena 不忽略转换为 `GLACIER` 存储类别的对象的问题，并添加了用于查询 Network Load Balancer 日志的示例。

进行了以下更改：

- 添加对工作组的支持。可使用工作组分隔用户、团队、应用程序或工作负载，并对每个查询或整个工作组可处理的数据量设置限制。由于工作组用作 IAM 资源，所以您可以使用资源级别的权限来控制对特定工作组的访问。您还可以在 Amazon CloudWatch 中查看与查询相关的指标、通过配置扫描的数据量限制来控制查询成本、创建阈值以及在突破这些阈值时触发操作，例如 Amazon SNS 告警。有关更多信息，请参阅 使用工作组运行查询 和 使用 CloudWatch 指标和事件控制成本和监控查询。

工作组是 IAM 资源。有关 IAM 中与工作组相关的操作、资源和条件的完整列表，请参阅服务授权参考中的 Amazon Athena 的操作、资源和条件键。创建新的工作组之前，请确保您使用的是工作组 IAM policy 和 AWS 托管策略：AmazonAthenaFullAccess。

您可以在控制台中、通过工作组 API 操作或通过 JDBC 驱动程序开始使用工作组。有关简要过程，请参阅设置工作组。要下载具有工作组支持的 JDBC 驱动程序，请参阅通过 JDBC 连接到 Amazon Athena。

如果您将工作组与 JDBC 驱动程序结合使用，则必须使用 `Workgroup` 配置参数在连接字符串中设置工作组名称，如下面的示例所示：

```
jdbc:awsathena://AwsRegion=<AWSREGION>;UID=<ACCESSKEY>;
PWD=<SECRETKEY>;S3OutputLocation=s3://DOC-EXAMPLE-BUCKET/<athena-
output>-<AWSREGION>/;
Workgroup=<WORKGROUPNAME>;
```

您运行 SQL 语句或者对驱动程序进行 JDBC API 调用的方式没有更改。驱动程序将工作组名称传递到 Athena。

要了解工作组引入的区别，请参阅 Athena 工作组 API和工作组故障排除。

- 改进了 Athena 中使用的 JSON OpenX SerDe。这些改进包括但不限于以下内容：
  - 支持 `ConvertDotsInJsonKeysToUnderscores` 属性。设置为 `TRUE` 时，它允许 SerDe 使用下划线替换键名称中的点。例如，如果 JSON 数据集包含名为 `"a.b"` 的键，您可以在 Athena 中使用此属性来定义列名 `"a_b"`。默认为 `FALSE`。预设情况下，Athena 不允许在列名中使用点。
  - 支持 `case.insensitive` 属性。预设情况下，Athena 要求 JSON 数据集中的所有键使用小写。使用 `WITH SERDE PROPERTIES ("case.insensitive"= FALSE;)` 允许您在数据中使用区分大小写的键名。默认为 `TRUE`。设置为 `TRUE` 时，SerDe 将所有大写列转换为小写。

  有关更多信息，请参阅 OpenX JSON SerDe。
- 修复了 Athena 在处理由 Amazon S3 生命周期策略存档到 Glacier 中的 Amazon S3 对象时，返回 `"access denied"` 错误消息的问题。作为修复此问题的结果，Athena 将忽略转换为 `GLACIER` 存储类别的对象。Athena 不支持查询 `GLACIER` 存储类中的数据。

  有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 the section called “对 Athena 中的数据和 Simple Storage Service (Amazon S3) 中的表的要求” 和转换为 GLACIER 存储类（对象归档）。

- 添加了查询 Network Load Balancer 访问日志的示例，该日志接收有关传输层安全性 (TLS) 请求的信息。有关更多信息，请参阅 the section called “网络负载均衡器”。

2018 年的 Athena 发布说明

2018 年 11 月 20 日

发布时间：2018 年 11 月 20 日

发布了 JDBC 和 ODBC 驱动程序的新版本，支持通过 AD FS 和 SAML 2.0（安全断言标记语言 2.0）对 Athena API 的联合访问。有关详细信息，请参阅 JDBC 驱动程序发行说明和 ODBC 驱动程序发行说明。

在此版本中，Active Directory 联合身份验证服务 (AD FS 3.0) 支持对 Athena 的联合访问。访问通过支持 SAML 2.0 的 JDBC 或 ODBC 驱动程序版本建立。有关配置对 Athena API 联合访问的信息，请参阅 the section called “启用对 Athena API 的联合身份访问”。

要下载 JDBC 驱动程序版本 2.0.6 及其文档，请参阅通过 JDBC 连接到 Amazon Athena。有关此版本的更多信息，请参阅 JDBC 驱动程序发布说明。

要下载 ODBC 驱动程序版本 1.0.4 及其文档，请参阅通过 ODBC 连接到 Amazon Athena。有关此版本的更多信息，请参阅 ODBC 驱动程序发布说明。

有关 AWS 中 SAML 2.0 支持更多信息，请参阅《IAM 用户指南》中的关于 SAML 2.0 联合验证。

2018 年 10 月 15 日

发布时间：2018 年 10 月 15 日

如果已升级到 AWS Glue Data Catalog，则有对以下各项提供支持的两个新功能：

- Data Catalog 元数据的加密。如果选择加密 Data Catalog 中的元数据，则必须将特定策略添加到 Athena。有关更多信息，请参阅访问 AWS Glue Data Catalog 中的加密元数据。
- 对 AWS Glue Data Catalog 中的资源的精细访问权限。现在，您可以定义基于身份的 (IAM) 策略，这类策略限制或允许访问 Athena 中使用的 Data Catalog 中的特定数据库和表。有关更多信息，请参阅 针对 AWS Glue Data Catalog 中数据库和表的精细访问权限。

> **Note**
>
> 数据位于 Amazon S3 存储桶中，并且数据访问权限由 对 Amazon S3 的访问权限 控制。要访问数据库和表中的数据，请继续将访问控制策略用于存储该数据的 Amazon S3 存储桶。

2018 年 10 月 10 日

发布时间：2018 年 10 月 10 日

Athena 支持 `CREATE TABLE AS SELECT`，这会从 `SELECT` 查询语句的结果创建表。有关详细信息，请参阅从查询结果创建表 (CTAS)。

创建 CTAS 查询之前，请务必在 Athena 文档中了解其行为。它包含下列相关信息：在 Amazon S3 中保存查询结果的位置，存储 CTAS 查询结果支持的格式列表，可以创建的分区数，以及支持的压缩格式。有关更多信息，请参阅 CTAS 查询的注意事项和限制。

使用 CTAS 查询可以：

- 在一个步骤中从查询结果创建表。
- 在 Athena 控制台中创建 CTAS 查询，使用示例。有关语法的信息，请参阅 CREATE TABLE AS。
- 将查询结果转换其他存储格式，例如 Parquet、ORC、AVRO、JSON 和 TEXTFILE。有关更多信息，请参阅CTAS 查询的注意事项和限制 和列式存储格式。

2018 年 9 月 6 日

发布时间：2018 年 9 月 06 日

发布了 ODBC 驱动程序新版本 (版本 1.0.3)。预设情况下，新版本的 ODBC 驱动程序会流式处理结果，而不是通过结果处理分页，从而允许商业智能工具更快地检索大型数据集。此版本还包括改进、错误修复以及对“将 SSL 与代理服务器结合使用”的文档更新。有关详细信息，请参阅该驱动程序的发行说明。

有关下载 ODBC 驱动程序版本 1.0.3 及其文档的信息，请参阅通过 ODBC 连接到 Amazon Athena。

流式处理结果功能适用于此新版本的 ODBC 驱动程序。它不适用于 JDBC 驱动程序。有关流式处理结果的信息，请参阅《ODBC 驱动程序安装和配置指南》，并搜索 UseResultsetStreaming。

ODBC 驱动程序版本 1.0.3 是该驱动程序的前一个版本的简易替代。建议迁移到最新驱动程序。

> **Important**
>
> 要使用 ODBC 驱动程序版本 1.0.3，请遵循以下要求：
>
> - 保持端口 444 对出站流量开放。
> - 将 `athena:GetQueryResultsStream` 策略操作添加到 Athena 的策略列表中。此策略操作不会直接通过 API 公开，仅作为流式处理结果支持的一部分与 ODBC 以及 JDBC 驱动程序一起使用。有关策略示例，请参阅AWS 托管策略：AWSQuicksightAthenaAccess。

2018 年 8 月 23 日

发布时间：2018 年 8 月 23 日

添加了对这些 DDL 相关的功能的支持并且修复了几个错误，如下所示：

- 添加了对 Parquet 格式的数据的 `BINARY` 和 `DATE` 数据类型以及 Avro 格式的数据的 `DATE` 和 `TIMESTAMP` 数据类型的支持。
- 添加了对 DDL 查询中的 `INT` 和 `DOUBLE` 的支持。`INTEGER` 是 `INT` 的别名，`DOUBLE PRECISION` 是 `DOUBLE` 的别名。
- 改进了 `DROP TABLE` 和 `DROP DATABASE` 查询的性能。
- 删除了在数据存储桶为空是在 Amazon S3 中创建 `_$folder$` 对象的操作。
- 修复了在未提供任何分区值时 `ALTER TABLE ADD PARTITION` 引发了错误的问题。
- 修复了在语句中已指定限定名称后检查分区时 `DROP TABLE` 忽略了数据库名称的问题。

有关 Athena 中支持的数据类型的更多信息，请参阅Amazon Athena 中的数据类型。

有关 Athena、JDBC 驱动程序中的类型和 Java 数据类型之间受支持的数据类型映射的信息，请参阅《JDBC 驱动程序安装和配置指南》中的数据类型一节。

2018 年 8 月 16 日

发布时间：2018 年 8 月 16 日

发布了 JDBC 驱动程序版本 2.0.5。预设情况下，新版本的 JDBC 驱动程序会流式处理结果，而不是通过结果处理分页，从而允许商业智能工具更快地检索大型数据集。与 JDBC 驱动程序的前一个版本相比，当前版本具有以下性能改进：

- 在提取小于 10K 的行时提高了约 2 倍的性能。
- 在提取大于 10K 的行时提高了约 5-6 倍的性能。

流式处理结果功能仅适用于 JDBC 驱动程序。它不适用于 ODBC 驱动程序。您无法将其与 Athena API 结合使用。有关流式处理结果的信息，请参阅《JDBC 驱动程序安装和配置指南》，并搜索 UseResultsetStreaming。

有关下载 JDBC 驱动程序版本 2.0.5 及其文档的信息，请参阅通过 JDBC 连接到 Amazon Athena。

JDBC 驱动程序版本 2.0.5 是该驱动程序的前一个版本 (2.0.2) 的简易替代。要确保可以使用 JDBC 驱动程序版本 2.0.5，请将 `athena:GetQueryResultsStream` 策略操作添加到 Athena 的策略列表。Q 此策略操作不会直接通过 API 公开，仅作为流式处理结果支持的一部分与 JDBC 驱动程序一起使用。有关策略示例，请参阅AWS 托管策略：AWSQuicksightAthenaAccess。有关从该驱动程序的版本 2.0.2 迁移到版本 2.0.5 的更多信息，请参阅《JDBC 驱动程序迁移指南》。

如果要从 1.x 驱动程序迁移到 2.x 驱动程序，需要将现有配置迁移到新配置。我们强烈建议您迁移到该驱动程序的最新版本。有关更多信息，请参阅 JDBC 驱动程序迁移指南。

2018 年 8 月 7 日

发布时间：2018 年 8 月 7 日

现在，您可以采用 GZIP 格式将 Amazon Virtual Private Cloud 流日志直接存储在 Amazon S3 中，其中，您可以在 Athena 中查询它们。有关更多信息，请参阅查询 Amazon VPC 流日志和 Amazon VPC 流日志现在可以传送至 S3。

2018 年 6 月 5 日

发布时间：2018 年 6 月 5 日

主题

- 对于视图的支持
- 错误消息改进和更新
- 错误修复

对于视图的支持

添加了对视图的支持。现在，您可以在 Athena 中使用 CREATE VIEW、DESCRIBE VIEW、DROP VIEW、SHOW CREATE VIEW 和 SHOW VIEWS。每次，当您在一个查询中引用一个视图时，定义该视图的查询都会运行。有关更多信息，请参阅 使用视图。

错误消息改进和更新

- 将 GSON 2.8.0 库包含到 CloudTrail SerDe 中，以解决 CloudTrail SerDe 的问题并启用 JSON 字符串的解析。
- 在 Athena 中，针对 Parquet（有时也针对 ORC），通过允许对列重新排序，增强了对分区的架构验证。这使 Athena 能够更好地处理随着时间推移架构演变的变化，以及通过 AWS Glue 爬网程序添加的表。有关更多信息，请参阅 处理架构更新。
- 添加了对于 `SHOW VIEWS` 的分析支持。
- 对最常见的错误消息进行了以下改进：
  - 当 SerDe 无法分析 Athena 查询中的列时，用描述性错误消息替换了内部错误消息。以往，当发生解析错误时，Athena 会发出一个内部错误。新的错误消息的内容为："HIVE_BAD_DATA: Error parsing field value for field 0: java.lang.String cannot be cast to org.openx.data.jsonserde.json.JSONObject" (HIVE_BAD_DATA：分析字段 0 的字段值时出错：无法将 java.lang.String 强制转换为 org.openx.data.jsonserde.json.JSONObject)。
  - 通过添加更多详细信息，改进了有关权限不足的错误消息。

错误修复

修复了以下错误：

- 修复了一个问题，允许 `REAL` 内部转换为 `FLOAT` 数据类型。这改进了与返回 AWS Glue 数据类型的 `FLOAT` 爬网程序的集成。
- 修复了 Athena 不能将 AVRO `DECIMAL`（一种逻辑类型）转换为 `DECIMAL` 类型的问题。
- 通过用 `WHERE` 子句来引用 `TIMESTAMP` 数据类型的值，修复了 Athena 不返回 Parquet 数据查询结果的问题。

2018 年 5 月 17 日

发布时间：2018 年 5 月 17 日

在 Athena 中，查询并发配额从 5 个提高到 20 个。这意味着，您可以同时提交并运行最多 20 个 DDL 查询和 20 个 SELECT 查询。注意，DDL 和 SELECT 查询的并发配额是分开的。

在 Athena 中，并发配额定义为可并发提交到服务的查询数。您可以同时提交最多 20 个同类型查询（DDL 或 SELECT）。如果您提交的查询超出并发查询配额，Athena API 将显示一条错误消息。

在您将查询提交给 Athena 后，它会根据总体服务负载和传入请求的数量，通过分配资源来处理查询。我们会持续监控和调整服务，以便尽快处理您的查询。

有关信息，请参阅服务限额。这是一个可调节的限额。您可以使用 Service Quotas 控制台为并发查询请求增加配额。

2018 年 4 月 19 日

发布时间：2018 年 4 月 19 日

JDBC 驱动程序的新版本 (版本 2.0.2) 已发布，该版本支持以 Array 数据类型的形式返回 ResultSet 数据，并实现了多项改进，修复了错误。有关详细信息，请参阅该驱动程序的发行说明。

有关下载 JDBC 驱动程序新版本 2.0.2 及其文档的信息，请参阅通过 JDBC 连接到 Amazon Athena。

JDBC 驱动程序的最新版本为 2.0.2。如果要从 1.x 驱动程序迁移到 2.x 驱动程序，需要将现有配置迁移到新配置。强烈建议迁移到最新驱动程序。

有关该驱动程序新版本中引入的更改的信息，请参阅《JDBC 驱动程序迁移指南》。

2018 年 4 月 6 日

发布时间：2018 年 4 月 6 日

使用自动完成功能在 Athena 控制台中键入查询。

2018 年 3 月 15 日

发布时间：2018 年 3 月 15 日

增加直接通过 CloudTrail 控制台为 CloudTrail 日志文件自动创建 Athena 表的功能。有关信息，请参阅使用 CloudTrail 控制台为 CloudTrail 日志创建 Athena 表 。

2018 年 2 月 2 日

发布时间：2018 年 2 月 12 日

添加了一种为使用 GROUP BY 子句的内存密集型查询将中间数据安全地卸载到磁盘的功能。这提高了此类查询的可靠性，可防止“Query resource exhausted”错误。

2018 年 1 月 19 日

发布时间：2018 年 1 月 19 日

Athena 使用 Presto（一种开源分布式查询引擎）运行查询。

对于 Athena，没有要管理的版本。我们已经以透明方式将 Athena 中的基础引擎升级到一个基于 Presto 0.172 版的版本。在您的末端不需要执行任何操作。

通过升级，您现在可以使用 Presto 0.172 函数和运算符，包括 Athena 中的 Presto 0.172 Lambda Expressions。

此版本的主要更新，包括社区提供的修补程序，包括：

- 支持忽略标题。您可以在定义表时使用 skip.header.line.count 属性，以允许 Athena 忽略标题。目前，对于使用 LazySimpleSerDe 和 OpenCSV SerDe 的查询，支持此功能，但对于 Grok 或 Regex SerDe 则不支持。
- 在 STRING 函数中支持 CHAR(n) 数据类型。CHAR(n) 的范围是 [1.255]，而 VARCHAR(n) 的范围是 [1,65535]。
- 支持关联子查询。
- 支持 Presto Lambda 表达式和函数。
- 改进了 DECIMAL 类型和运算符的性能。
- 支持筛选聚合，例如 SELECT sum(col_name) FILTER，其中 id > 0。
- DECIMAL、TINYINT、SMALLINT 和 REAL 数据类型的下推谓词。
- 支持定量比较谓词：ALL、ANY 和 SOME。
- 增加了函数：arrays_overlap()、array_except()、levenshtein_distance()、codepoint()、skewne 和 typeof()。
- 增加了接受时区参数的 from_unixtime() 函数的变体。
- 增加了 bitwise_and_agg() 和 bitwise_or_agg() 聚合函数。
- 增加了 xxhash64() 和 to_big_endian_64() 函数。
- 向 json_extract() 和 json_extract_scalar() 函数中添加了对转义双引号或反斜杠的支持 (将反斜杠与 JSON 路径下标一起使用)。这会更改任何使用反斜杠的调用的语义，因为反斜杠以前被视为普通字符。

有关函数和运算符的更多信息，请参阅本指南中的 DML 查询、函数和运算符，以及 Presto 文档中的函数和运算符。

Athena 并非支持所有 Presto 功能。有关更多信息，请参阅限制。

2017 年的 Athena 发布说明

2017 年 11 月 13 日

发布时间：2017 年 11 月 13 日

增加了对将 Athena 连接到 ODBC 驱动程序的支持。有关信息，请参阅通过 ODBC 连接到 Amazon Athena。

2017 年 11 月 1 日

发布时间：2017 年 11 月 1 日

增加了对查询地理空间数据以及对亚太地区 (首尔)、亚太地区 (孟买) 和欧洲 (伦敦) 区域的支持。有关信息，请参阅 查询地理空间数据 及 AWS 区域 和端点。

2017 年 10 月 19 日

发布时间：2017 年 10 月 19 日

增加了对欧洲 (法兰克福) 的支持。有关受支持的区域列表，请参阅 AWS 区域 和端点。

2017 年 10 月 3 日

发布时间：2017 年 10 月 3 日

使用 AWS CloudFormation 创建命名 Athena 查询。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的 AWS::Athena::NamedQuery。

2017 年 9 月 25 日

发布时间：2017 年 9 月 25 日

添加了对亚太地区（悉尼）的支持。有关受支持的区域列表，请参阅 AWS 区域 和端点。

2017 年 8 月 14 日

发布时间：2017 年 8 月 14 日

增加与 AWS Glue Data Catalog 的集成以及用于从 Athena 托管数据目录更新到 AWS Glue Data Catalog 的迁移向导。有关更多信息，请参阅 与 AWS Glue 集成。

2017 年 8 月 4 日

发布时间：2017 年 8 月 4 日

增加了对 Grok SerDe 的支持，它为非结构化文本文件 (例如日志) 中的记录添加了更简便的模式匹配。有关更多信息，请参阅 Grok SerDe。增加了键盘快捷键，用以通过控制台滚动查看查询历史记录 (使用 Windows 时为 CTRL + ⇧/⇩，使用 Mac 时为 CMD + ⇧/⇩)。

2017 年 6 月 22 日

发布时间：2017 年 6 月 22 日

增加了对亚太地区（东京）和亚太地区（新加坡）区域的支持。有关受支持的区域列表，请参阅 AWS 区域 和端点。

2017 年 6 月 8 日

发布时间：2017 年 6 月 8 日

增加了对欧洲（爱尔兰）的支持。有关更多信息，请参阅AWS 区域 和端点。

2017 年 5 月 19 日

发布时间：2017 年 5 月 19 日

增加了 Amazon Athena API 和针对 Athena 的 AWS CLI 支持；将 JDBC 驱动程序更新到了 1.1.0 版；解决了各种问题。

- Amazon Athena 允许对 Athena 进行应用程序编程。有关更多信息，请参阅 Amazon Athena API 参考。最新 AWS 软件开发工具包包括对 Athena API 的支持。对于指向文档和下载的链接，请参阅适用于 Amazon Web Services 的工具中的 SDK 部分。
- AWS CLI 包括用于 Athena 的新命令。有关更多信息，请参阅 Amazon Athena API 参考。
- 推出了新的 JDBC 驱动程序 1.1.0，它支持新的 Athena API 以及最新功能和错误修复程序。请在 https://downloads.athena.us-east-1.amazonaws.com/drivers/AthenaJDBC41-1.1.0.jar 上下载该驱动程序。我们建议您升级到最新 Athena JDBC 驱动程序；但您可能会继续使用早期驱动程序版本。早期驱动程序版本不支持 Athena API。有关更多信息，请参阅 通过 JDBC 连接到 Amazon Athena。

- 早期版本的 Athena 中特定于策略声明的操作已被弃用。如果您升级到 JDBC 驱动程序 1.1.0 版，并且让客户管理型或内联 IAM policy 附加到 JDBC 用户，则必须更新 IAM policy。相比之下，早期版本的 JDBC 驱动程序不支持 Athena API，因此您只能在附加到早期版本 JDBC 用户的策略中指定已弃用的操作。出于此原因，您应该不需要更新客户管理型或内联 IAM policy。
- 在 Athena API 发布之前，这些特定于策略的操作就已经被用在 Athena 中。策略中的这些已废弃操作只适用于 1.1.0 版本之前的 JDBC 驱动程序。如果要升级 JDBC 驱动程序，请用列出的适当 API 操作替换允许或拒绝已废弃操作的策略语句，否则会发生错误：

已弃用的特定于策略的操作	对应的 Athena API 操作
`athena:RunQuery`	`athena:StartQueryExecution`
`athena:CancelQueryExecution`	`athena:StopQueryExecution`
`athena:GetQueryExecutions`	`athena:ListQueryExecutions`

改进

- 将查询字符串长度限制提高到 256 KB。

错误修复

- 修复了一个在控制台中滚动结果时导致查询结果显示不正确的问题。
- 修正了一个由 Amazon S3 数据文件中 `\u0000` 的字符串导致错误的问题。
- 修复了一个导致取消通过 JDBC 驱动程序进行查询的请求失败的问题。
- 修复了一个导致 AWS CloudTrail SerDe 无法使用美国东部（俄亥俄）中的 Amazon S3 数据的问题。
- 修复了一个导致 `DROP TABLE` 在分区表上失败的问题。

2017 年 4 月 4 日

发布时间：2017 年 4 月 4 日

增加了对 Amazon S3 数据加密的支持，并发布了具有加密支持、改进和错误修复的 JDBC 驱动程序更新 (1.0.1 版)。

功能

- 增加了以下加密功能：
  - 支持查询 Amazon S3 中的加密数据。
  - 支持加密 Athena 查询结果。
- 新版本的驱动程序支持新的加密功能，添加了改进，并修复了问题。
- 增加了使用 `ALTER TABLE` 添加、替换和更改列的功能。有关详细信息，请参阅 Hive 文档中的修改列。
- 增加了对查询 LZO 压缩数据的支持。

有关更多信息，请参阅 静态加密。

改进

- 通过页面大小改进提高了 JDBC 查询性能，返回 1000 行，而不是 100 行。
- 增加了使用 JDBC 驱动程序接口取消查询的功能。
- 增加了在 JDBC 连接 URL 中指定 JDBC 选项的功能。请参阅 通过 JDBC 连接到 Amazon Athena 获取最新的 JDBC 驱动程序。
- 增加了驱动程序中的 PROXY 设置，现在可以使用适用于 Java 的 AWS 软件开发工具包中的 ClientConfiguration 来设置它。

错误修复

修复了以下错误：

- 当使用 JDBC 驱动程序接口发出多个查询时，会发生限制错误。
- 当投影十进制数据类型时，JDBC 驱动程序将会停止。
- JDBC 驱动程序将以字符串的形式返回每个数据类型，无论数据类型在表中是如何定义的都是如此。例如，使用 `resultSet.GetObject()` 选择一个定义为 `INT` 数据类型的列将会返回 `STRING` 数据类型，而不是 `INT`。
- JDBC 驱动程序将会在建立连接时验证凭据，而不是在运行查询时进行验证。

- 在与 URL 一起指定架构时，通过 JDBC 驱动程序进行的查询将会失败。

2017 年 3 月 24 日

发布时间：2017 年 3 月 24 日

增加了 AWS CloudTrail SerDe，提高了性能、解决了分区问题。

功能

- 添加了 AWS CloudTrail SerDe，但已被 Hive JSON SerDe 所取代以进行 CloudTrail 日志读取。有关查询 CloudTrail 日志的信息，请参阅 查询 AWS CloudTrail 日志。

改进

- 提高了扫描大量分区时的性能。
- 提高了 `MSCK Repair Table` 操作的性能。
- 增加了查询在主要区域之外的区域存储的 Amazon S3 数据的功能。除了标准 Athena 费用外，Amazon S3 的标准区域间数据传输费率也适用。

错误修复

- 修复了一个在未加载任何分区时可能发生“table not found error”的错误。
- 修复了一个避免引发 `ALTER TABLE ADD PARTITION IF NOT EXISTS` 查询异常的错误。
- 修复了 `DROP PARTITIONS` 中的一个错误。

2017 年 2 月 20 日

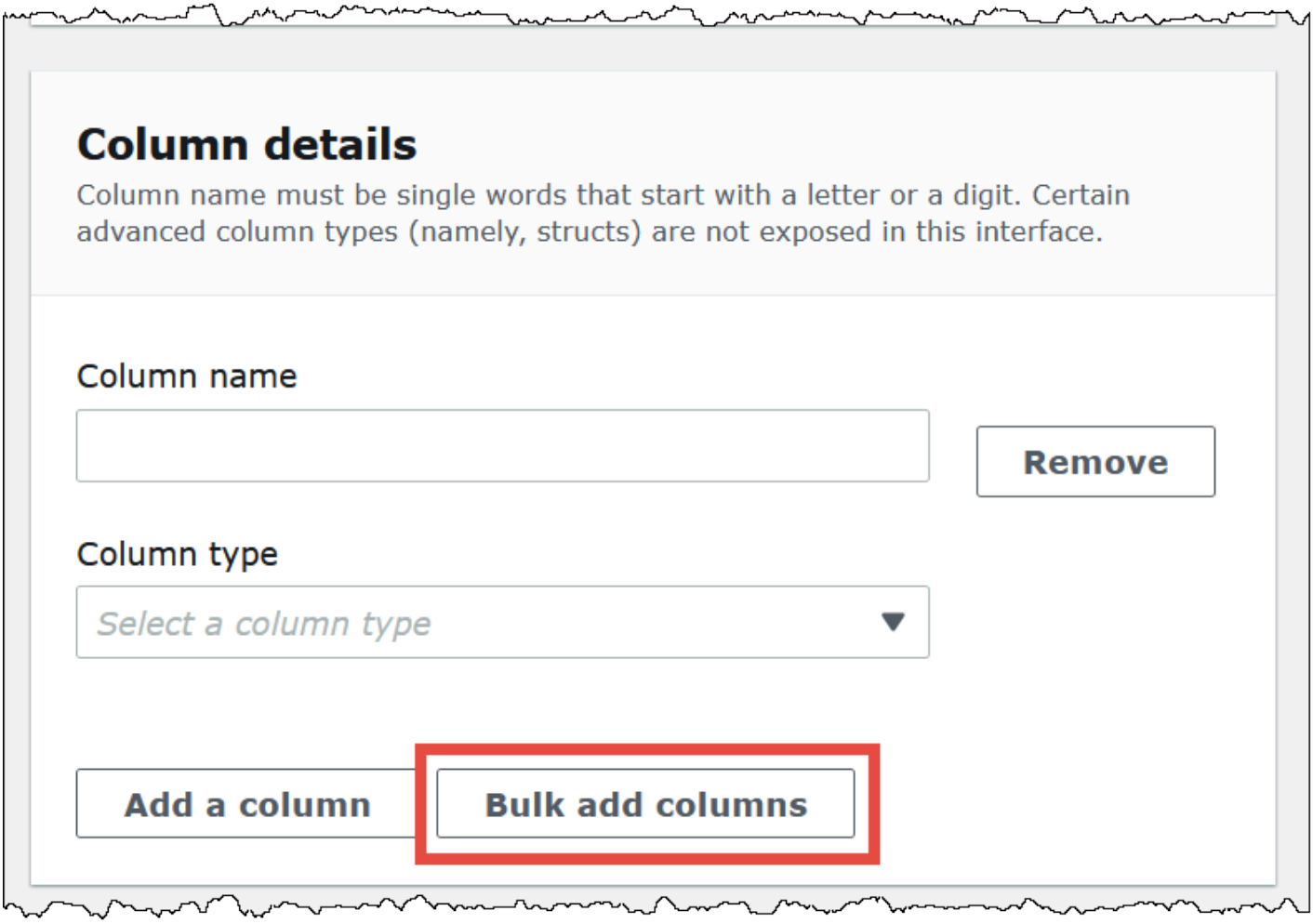
发布时间：2017 年 2 月 20 日

增加了对 AvroSerDe 和 OpenCSVSerDe、美国东部（俄亥俄）区域以及在控制台向导中批量编辑列的支持。改进了大型 Parquet 表的性能。

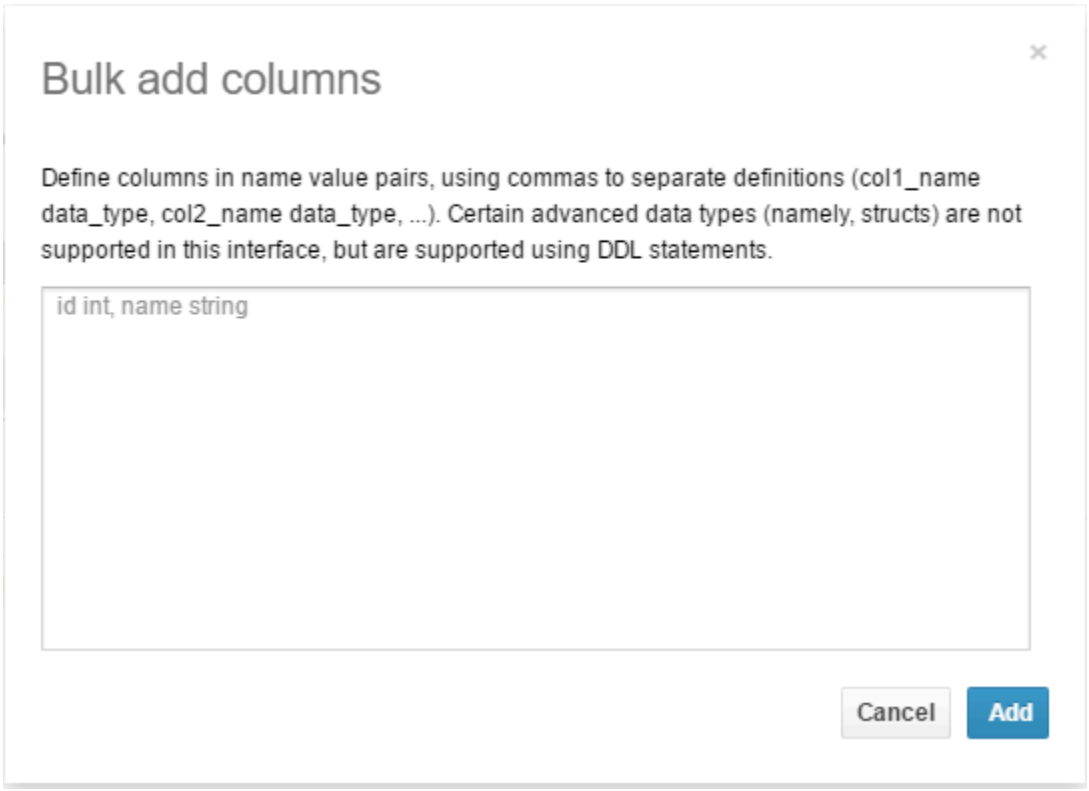
功能

- 引入了对新 SerDes 的支持：

  - Avro SerDe
  - 用于处理 CSV 的 OpenCSVSerDe
- 美国东部（俄亥俄）区域 (us-east-2) 启动。现在，您可以在此区域中运行查询。
- 现在，您可以使用 Create Table From S3 bucket data（根据 S3 存储桶数据创建表）表单来批量定义表架构。在查询编辑器中，选择 Create（创建）、S3 bucket data（S3 存储桶数据），然后选择 Column details（列详细信息）部分中的 Bulk add columns（批量添加列）。



在文本框中键入名称值对，然后选择 Add。



改进

- 改进了大型 Parquet 表的性能。

文档历史记录

文档最近更新时间：2024 年 5 月 28 日。

我们经常更新文档来处理您的反馈意见。下表介绍 Amazon Athena 文档的重要补充部分。并非所有更新都呈现出来。

更改	描述	发行日期
更新了 `AmazonAthenaFullAccess` 托管式策略。	在 AmazonAthenaFullAccess 托管式策略中增加了 `datazone:ListDomains` 、`datazone:ListProjects` 和 `datazone:ListAccountEnvironments` 权限。借助这些新增的操作，Athena 用户可以使用 Amazon DataZone 域、项目和环境。有关更多信息，请参阅 在 Athena 中使用 Amazon DataZone。	2024 年 1 月 3 日
更新了 `AmazonAthenaFullAccess` 托管式策略。	在 AmazonAthenaFullAccess 托管式策略中增加了 `glue:StartColumnStatisticsTaskRun` 、`glue:GetColumnStatisticsTaskRun` 和 `glue:GetColumnStatisticsTaskRuns` 权限。借助这些新增的操作，Athena 可以调用 AWS Glue，以检索基于成本的优化器功能的统计数据。有关更多信息，请参阅 使用成本型优化器。	2024 年 1 月 3 日
添加了关于已启用 IAM Identity Center 的 Athena 工作组的文档。	您可以创建采用 IAM Identity Center 身份验证模式的 Athena SQL 工作组。这些工作组支持跨 AWS 服务（例如 Amazon Athena 和 Amazon EMR Studio）使用相同的身份。有关更多信息，请参阅 使用已启用 IAM Identity Center 的 Athena 工作组。	2023 年 12 月 5 日
添加了有关查询 S3 Express One Zone 数据的文档	您可以使用 Athena 查询 Amazon S3 Express One Zone 存储类中的数据。有关更多信息，请参阅 查询 S3 Express One Zone 数据。	2023 年 11 月 28 日
添加了关于 Glue Data Catalog 视图的文档。	您可以使用 Glue Data Catalog 视图跨 AWS 服务（例如 Amazon Athena 和 Amazon Redshift）提供单一通用视	2023 年 11 月 27 日

更改	描述	发行日期
	图。有关更多信息，请参阅 使用 AWS Glue Data Catalog 视图。	
添加了关于成本型优化器功能的文档。	您可以使用 AWS Glue 中的统计信息来优化 Athena SQL 中的查询。有关更多信息，请参阅 使用成本型优化器。	2023 年 11 月 17 日
添加了有关 Athena JDBC 3.x 驱动程序的文档	您可以使用 Athena JDBC 3.x 驱动程序直接从 Amazon S3 读取查询结果。JDBC 3.x 驱动程序支持 JDBC 2.x 驱动程序支持的几乎所有身份验证方法。有关更多信息，请参阅 Athena JDBC 3.x 驱动程序。	2023 年 11 月 16 日
添加了有关在 Athena 中使用 DataZone 的文档。	您可以使用 DataZone 来简化在 Athena、AWS 和 Lake Formation 等 AWS Glue 分析服务中的体验。有关更多信息，请参阅 在 Athena 中使用 Amazon DataZone。	2023 年 10 月 4 日
添加了有关容量预留的文档。	现在，您可以在 Amazon Athena 上使用容量预留对完全托管的计算容量运行 SQL 查询。有关更多信息，请参阅 管理查询处理容量。	2023 年 4 月 28 日
添加了有关查询联合视图的文档。	现在，可以在 Athena 中的联合数据来源上创建和查询视图。有关更多信息，请参阅 查询联合视图。	2023 年 4 月 4 日
添加了有关在 Amazon S3 中防止节流的文档。	有关更多信息，请参阅 防止 Amazon S3 节流。	2023 年 3 月 24 日
更新了 `AmazonAthenaFullAccess` 托管式策略。	在 AmazonAthenaFullAccess 托管式策略中增加了 `pricing:GetProducts` 增加的操作提供了对 AWS Billing and Cost Management 的访问权限。有关更多信息，请参阅《AWS Billing and Cost Management API 参考》中的 GetProducts。	2023 年 1 月 25 日
扩展了有关 Athena 压缩支持的文档。	增加了 Hive 表压缩、Iceberg 表压缩 和 ZSTD 压缩级别 的单独主题。有关更多信息，请参阅 Athena 压缩支持。	2023 年 1 月 20 日

更改	描述	发行日期
添加了 Amazon Athena for Apache Spark 的文档。	现在，您可以在 Amazon Athena 上以交互方式创建和运行 Apache Spark 应用程序和与 Jupyter 兼容的笔记本。有关更多信息，请参阅 在 Amazon Athena 中使用 Apache Spark。	2022 年 11 月 30 日
添加了 Athena IBM Db2 连接器的文档。	现在，您可以使用适用于 IBM Db2 的 Amazon Athena 连接器从 Athena 查询 Db2。有关更多信息，请参阅 Amazon Athena IBM Db2 连接器	2022 年 11 月 18 日
添加了用于查询结果重用的文档。	在 Athena 中重新运行查询时，现在可以选择重用上次存储的查询结果。此可以提高性能并降低扫描字节数方面的成本。有关更多信息，请参阅 重用查询结果。	2022 年 11 月 8 日
更新了 CloudTrail 日志的文档。	用于查询 CloudTrail 日志的 `CREATE TABLE` DDL 已更新，以使用 JSON SerDe，而不是 CloudTrail SerDe。有关更多信息，请参阅 查询 AWS CloudTrail 日志。	2022 年 11 月 3 日
增加了 Athena 引擎版本 3 的文档。	有关 Athena 引擎版本 3 的更多信息，请参阅 Athena 引擎版本 3。	2022 年 10 月 13 日
增加了有关使用 Okta 插件为 ODBC 配置 SSO 的教程。	配置 Amazon Athena ODBC 驱动程序和 Okta 插件，以使用 Okta 身份提供者添加单点登录（SSO）功能。有关更多信息，请参阅 使用 Okta 插件和 Okta 身份提供者为 ODBC 配置 SSO。	2022 年 8 月 23 日
增加文档，以便在 Athena 控制台中查看查询计划和统计数据。	您可以使用 Athena 查询编辑器查看有关查询如何运行的图形展示，以及有关完整查询如何运行的图表、详细信息和统计信息。有关更多信息，请参阅 查看 SQL 查询的执行计划 和 查看已完成查询的统计数据和执行详细信息。	2022 年 7 月 21 日
添加了文档，用于查询外部 Hive 元存储中的 Apache Hive 视图。	您可以使用 Athena 查询在外部 Hive 元存储中创建的 Apache 视图。某些 Hive 函数不受支持或需要特殊处理。有关更多信息，请参阅 使用 Hive 视图。	2022 年 4 月 22 日

更改	描述	发行日期
增加了有关已保存的查询的文档。	您可以使用 Athena 中保存的查询功能来保存、回调、编辑和重命名查询。有关更多信息，请参阅本指南中的使用已保存的查询或《Amazon Athena API 参考》中 UpdateNamedQuery。	2022 年 2 月 28 日
添加了 Apache Iceberg 支持的预览文档。	Athena 支持对 Apache Iceberg 表进行读取、历史状态和写入查询，这些表对数据和数据的元数据仓分别使用 Apache Parquet 格式和 AWS Glue 目录。有关更多信息，请参阅 使用 Apache Iceberg 表。	2021 年 11 月 26 日
添加了跨账户联合查询的文档。	您可以使用跨账户联合查询功能，查询其他账户中的数据来源。有关设置权限以启用此功能的信息，请参阅 启用跨账户联合查询。	2021 年 11 月 12 日
增加了有关 Athena UNLOAD 语句的文档。	使用 UNLOAD 语句将来自 SELECT 语句的查询结果写入 Apache Parquet、ORC、Apache Avro 和 JSON 格式。有关更多信息，请参阅 UNLOAD。	2021 年 8 月 5 日
增加了有关 Athena EXPLAIN 语句功能的文档。	有关更多信息，请参阅 在 Athena 中使用 EXPLAIN 和 EXPLAIN ANALYZE 和 了解 Athena EXPLAIN 语句结果。	2021 年 4 月 5 日
添加了有关 Athena 故障排除和性能优化的页面。	有关更多信息，请参阅 在 Athena 中进行故障排除 和 Athena 中的性能优化。	2020 年 12 月 30 日
添加了 Athena 引擎版本控制和 Athena 引擎版本 2 的文档。	有关更多信息，请参阅 Athena 引擎版本控制。	2020 年 11 月 11 日
更新了一般可用版本的联合查询文档。	有关更多信息，请参阅 使用 Amazon Athena 联合查询 和 将 Athena 与 CalledVia 上下文键结合使用。	2020 年 11 月 11 日

更改	描述	发行日期
添加了有关将 JDBC 驱动程序与 Lake Formation 搭配使用以联合访问 Athena 的文档。	有关更多信息，请参阅 将 Lake Formation 和 Athena JDBC 和 ODBC 驱动程序用于对 Athena 进行联合访问 和 教程：使用 Lake Formation 和 JDBC 配置 Okta 用户对 Athena 的联合访问。	2020 年 9 月 25 日
添加了 Amazon Athena OpenSearch 数据连接器的文档。	有关更多信息，请参阅 Amazon Athena OpenSearch 连接器。	2020 年 7 月 21 日
添加了查询胡迪数据集的文档。	有关更多信息，请参阅 使用 Athena 查询 Apache Hudi 数据集。	2020 年 7 月 9 日
添加了有关查询存储在 Amazon S3 中的 Apache Web 服务器日志和 IIS Web 服务器日志的文档。	有关更多信息，请参阅 查询存储在 Amazon S3 中的 Apache 日志 和 查询存储在 Amazon S3 中的互联网信息服务器（IIS）日志。	2020 年 7 月 8 日
增加了有关用于外部 Hive 元数据仓的 Athena 数据连接器的一般发布文档。	有关更多信息，请参阅 将 Athena 数据连接器用于外部 Hive 元数据仓。	2020 年 6 月 1 日
添加了有关标记数据目录资源的文档。	有关更多信息，请参阅 为 Athena 资源添加标签。	2020 年 6 月 1 日
添加了有关分区投影的文档。	有关更多信息，请参阅 使用 Amazon Athena 分区投影。	2020 年 5 月 21 日
更新了 Athena 的 Java 代码示例。	有关更多信息，请参阅 代码示例。	2020 年 5 月 11 日

更改	描述	发行日期
添加了关于查询 Amazon GuardDuty 调查结果的主题。	有关更多信息，请参阅 查询 Amazon GuardDuty 调查结果。	2020 年 3 月 19 日
添加了关于使用 CloudWatch Events 监控 Athena 查询状态转换的主题。	有关更多信息，请参阅 使用 Amazon EventBridge 事件监控 Athena 查询。	2020 年 3 月 11 日
添加了有关使用 Athena 查询 AWS Global Accelerator 流日志的主题。	有关更多信息，请参阅 查询 AWS Global Accelerator 流日志。	2020 年 2 月 6 日

更改	描述	发行日期
• 添加了有关将 CTAS 与 INSERT INTO 结合使用以将未分区源中的数据添加到已分区的目标的文档。 • 添加了适用于 Athena 的 1.1.0 预览版 ODBC 驱动程序的下载链接。 • 更正了 `SHOW DATABASES LIKE` 正则表达式的说明。 • 更正了 CTA 主题中的 `partitioned_by` 语法。 • 其他较小的修复。	文档更新包括但不限于以下主题： • 将 CTAS 和 INSERT INTO 用于 ETL 和数据分析 • 通过 ODBC 连接到 Amazon Athena（1.1.0 预览功能现在包含在 1.1.2 ODBC 驱动程序中。） • SHOW DATABASES • CREATE TABLE AS	2020 年 2 月 4 日
添加了有关将 CTAS 与 INSERT INTO 结合使用以将已分区源中的数据添加到已分区的目标的文档。	有关更多信息，请参阅 使用 CTAS 和 INSERT INTO 绕过 100 分区限制。	2020 年 1 月 22 日
更新了查询结果位置信息。	Athena 不再创建“默认”查询结果位置。有关更多信息，请参阅 指定查询结果位置。	2020 年 1 月 20 日

更改	描述	发行日期
添加了有关查询 AWS Glue Data Catalog 的主题。更新了有关 Athena 中的服务限额（以前称作“服务限制”）的信息。	有关更多信息，请参阅以下主题： • 查询 AWS Glue Data Catalog • 服务限额	2020 年 1 月 17 日
更正了有关 OpenCSVSerDe 的主题，以说明应以 UNIX 数字格式指定 TIMESTAMP 类型。	有关更多信息，请参阅 用于处理 CSV 的 OpenCSVSerDe。	2020 年 1 月 15 日
更新了有关加密的安全主题，以说明 Athena 不支持非对称密钥。	Athena 仅支持读取和写入数据的对称密钥。有关更多信息，请参阅 支持的 Amazon S3 加密选项。	2020 年 1 月 8 日
增加了有关对使用自定义 AWS KMS 密钥加密的 Amazon S3 存储桶进行跨账户存取的信息。	有关更多信息，请参阅 跨账户访问使用自定义 AWS KMS 密钥加密的存储桶。	2019 年 12 月 13 日
添加了有关联合查询、外部 Hive 元存储、机器学习和用户定义的函数的文档。添加了新的 CloudWatch 指标。	有关更多信息，请参阅以下主题： • 使用 Amazon Athena 联合查询 • 可用数据来源连接器 • 将 Athena 数据连接器用于外部 Hive 元数据仓 • 在 Amazon Athena 中使用机器学习（ML） • 使用用户定义函数进行查询 • Athena 的 CloudWatch 指标与维度列表	2019 年 11 月 26 日

更改	描述	发行日期
增加了有关新 INSERT INTO 命令的部分，并更新了查询结果位置信息以支持数据清单文件。	有关更多信息，请参阅 INSERT INTO 和 使用查询结果、最近查询和输出文件。	2019 年 9 月 18 日
增加了有关接口 VPC 终端节点 (PrivateLink) 支持的部分。更新了 JDBC 驱动程序。更新了有关丰富的 VPC 流日志的信息。	有关更多信息，请参阅 使用接口 VPC 终端节点连接到 Amazon Athena、查询 Amazon VPC 流日志 和 通过 JDBC 连接到 Amazon Athena。	2019 年 9 月 11 日
增加了与 AWS Lake Formation 集成的部分。	有关更多信息，请参阅 使用 Athena 查询向 AWS Lake Formation 注册的数据。	2019 年 6 月 26 日
更新了与其他 AWS 服务保持一致性的“安全性”部分。	有关更多信息，请参阅 Amazon Athena 安全性。	2019 年 6 月 26 日
增加了有关查询 AWS WAF 日志的部分。	有关更多信息，请参阅 查询 AWS WAF 日志。	2019 年 5 月 31 日

更改	描述	发行日期
发布新版本的 ODBC 驱动程序，支持 Athena 工作组。	要下载 ODBC 驱动程序版本 1.0.5 及其文档，请参阅通过 ODBC 连接到 Amazon Athena。在工作组上使用标签时，对 ODBC 驱动程序连接字符串没有更改。要使用标签，请将 ODBC 驱动程序升级到最新版本，即此当前版本。 此驱动程序版本可让您使用 Athena API 工作组操作来创建和管理工作组，使用 Athena API 标签操作来添加、列出或删除工作组上的标签。在您开始之前，请确保您在 IAM 中拥有资源级别的权限以对工作组和标签执行操作。	2019 年 3 月 5 日
Amazon Athena 中添加了工作组的标签支持。	标签包含您定义的一个键和一个值。您在标记工作组时，将为其分配自定义元数据。例如，为每个成本中心创建一个工作组。然后，通过将标签添加到这些组，您可以跟踪每个成本中心的 Athena 支出。有关更多信息，请参阅《AWS Billing and Cost Management 用户指南》中的使用账单标签。	2019 年 2 月 22 日
改进了 Athena 中使用的 JSON OpenX SerDe。	这些改进包括但不限于以下内容： • 支持 `ConvertDotsInJsonKeysToUnderscores` 属性。设置为 `TRUE` 时，它允许 SerDe 使用下划线替换键名称中的点。例如，如果 JSON 数据集包含名为 `"a.b"` 的键，您可以在 Athena 中使用此属性来定义列名 `"a_b"`。默认为 `FALSE`。预设情况下，Athena 不允许在列名中使用点。 • 支持 `case.insensitive` 属性。预设情况下，Athena 要求 JSON 数据集中的所有键使用小写。使用 `WITH SERDE PROPERTIES ("case.insensitive"= FALSE;)` 允许您在数据中使用区分大小写的键名。默认为 `TRUE`。设置为 `TRUE` 时，SerDe 将所有大写列转换为小写。 有关更多信息，请参阅 OpenX JSON SerDe。	2019 年 2 月 18 日

更改	描述	发行日期
添加对工作组的支持。	可使用工作组分隔用户、团队、应用程序或工作负载，并对每个查询或整个工作组可处理的数据量设置限制。由于工作组用作 IAM 资源，所以您可以使用资源级别的权限来控制对特定工作组的访问。您还可以在 Amazon CloudWatch 中查看与查询相关的指标、通过配置扫描的数据量限制来控制查询成本、创建阈值以及在突破这些阈值时触发操作，例如 Amazon SNS 告警。有关更多信息，请参阅 使用工作组运行查询 和 使用 CloudWatch 指标和事件控制成本和监控查询。	2019 年 2 月 18 日
增加了对分析来自 Network Load Balancer 的日志的支持。	增加了用于分析来自 Network Load Balancer 的日志的示例 Athena 查询。这些日志将接收有关发送到 Network Load Balancer 的传输层安全性 (TLS) 请求的详细信息。您可以使用这些访问日志分析流量模式并解决问题。有关信息，请参阅the section called “网络负载均衡器”。	2019 年 1 月 24 日
发布了 JDBC 和 ODBC 驱动程序的新版本，支持通过 AD FS 和 SAML 2.0（安全断言标记语言 2.0）对 Athena API 的联合访问。	在此驱动程序版本中，Active Directory 联合身份验证服务 (AD FS 3.0) 支持对 Athena 的联合访问。访问通过支持 SAML 2.0 的 JDBC 或 ODBC 驱动程序版本建立。有关配置对 Athena API 联合访问的信息，请参阅 the section called “启用对 Athena API 的联合身份访问”。	2018 年 11 月 10 日
添加了对 Athena 中数据库和表的精细访问控制支持。此外，在 Athena 中添加了策略，允许您在 Data Catalog 中加密数据库和表元数据。	添加了对创建对 AWS Glue Data Catalog 中的资源（如 Athena 中使用的数据库和表）提供精细访问控制的基于身份的 (IAM) 策略的支持。 此外，您可以通过将特定策略添加到 Athena，在 Data Catalog 中加密数据库和表元数据。 有关详细信息，请参阅针对 AWS Glue Data Catalog 中数据库和表的精细访问权限。	2018 年 10 月 15 日

更改	描述	发行日期
增加了对 CREATE TABLE AS SELECT 语句的支持。 文档中有了其他改进。	增加了对 CREATE TABLE AS SELECT 语句的支持。请参阅从查询结果创建表（CTAS）、CTAS 查询的注意事项和限制和 CTAS 查询的示例。	2018 年 10 月 10 日
发行 ODBC 驱动程序版本 1.0.3，此版本具有对流式处理结果的支持而不是在页面中提取这些结果。 文档中有了其他改进。	ODBC 驱动程序版本 1.0.3 支持流式处理结果，并包括了改进、错误修复以及对“将 SSL 与代理服务器结合使用”的文档更新。 有关下载 ODBC 驱动程序版本 1.0.3 及其文档的信息，请参阅通过 ODBC 连接到 Amazon Athena。	2018 年 9 月 6 日
发行 JDBC 驱动程序版本 2.0.5，此版本具有对流式处理结果的默认支持而不是在页面中提取这些结果。 文档中有了其他改进。	发行 JDBC 驱动程序 2.0.5，此版本具有对流式处理结果的默认支持而不是在页面中提取这些结果。有关信息，请参阅通过 JDBC 连接到 Amazon Athena。	2018 年 8 月 16 日

更改	描述	发行日期
更新了查询 Amazon Virtual Private Cloud 流日志的文档，流日志可直接以 GZIP 格式存储在 Amazon S3 中。 更新了查询 ALB 日志的示例。	更新了查询 Amazon Virtual Private Cloud 流日志的文档，流日志可直接以 GZIP 格式存储在 Amazon S3 中。有关信息，请参阅查询 Amazon VPC 流日志。 更新了查询 ALB 日志的示例。有关信息，请参阅查询 Application Load Balancer 日志。	2018 年 8 月 7 日
添加了对视图的支持。添加了各种数据存储格式的架构操作指南。	添加了对视图的支持。有关信息，请参阅使用视图。 更新了此指南中有关如何处理各种数据存储格式的架构更新的指导。有关信息，请参阅处理架构更新。	2018 年 6 月 5 日
默认查询并发限制从 5 个提高到 20 个。	您可以同时提交并运行最多 20 个 DDL 查询和 20 个 SELECT 查询。有关信息，请参阅服务限额。	2018 年 5 月 17 日
增加了查询选项卡，并支持用户在查询编辑器中配置自动完成功能。	增加了查询选项卡，并支持用户在查询编辑器中配置自动完成功能。有关信息，请参阅开始使用。	2018 年 5 月 8 日
发布了 JDBC 驱动程序版本 2.0.2。	发布了 JDBC 驱动程序新版本 (版本 2.0.2)。有关信息，请参阅通过 JDBC 连接到 Amazon Athena。	2018 年 4 月 19 日
增加自动填写功能，方便用户在 Athena 控制台中键入查询。	增加自动填写功能，方便用户在 Athena 控制台中键入查询。	2018 年 4 月 6 日

更改	描述	发行日期
增加直接通过 CloudTrail 控制台为 CloudTrail 日志文件创建 Athena 表的功能。	增加直接通过 CloudTrail 控制台为 CloudTrail 日志文件自动创建 Athena 表的功能。有关信息，请参阅使用 CloudTrail 控制台为 CloudTrail 日志创建 Athena 表 。	2018 年 3 月 15 日
增加了对使用 GROUP BY 安全地将中间数据卸载到磁盘进行查询的支持。	添加了一种为使用 GROUP BY 子句的内存密集型查询将中间数据安全地卸载到磁盘的功能。这提高了此类查询的可靠性，可防止“Query resource exhausted”错误。有关更多信息，请参阅2018 年 2 月 2 日的发布说明。	2018 年 2 月 2 日
增加了对 Presto 版本 0.172 的支持。	已将 Amazon Athena 中的底层引擎升级到基于 Presto 0.172 版的版本。有关更多信息，请参阅2018 年 1 月 19 日的发布说明。	2018 年 1 月 19 日
增加了对 ODBC 驱动程序的支持。	增加了对将 Athena 连接到 ODBC 驱动程序的支持。有关信息，请参阅使用 ODBC 连接到 Amazon Athena。	2017 年 11 月 13 日
增加了对亚太地区（首尔）、亚太地区（孟买）和欧洲（伦敦）区域的支持。增加了对查询地理空间数据的支持。	增加了对查询地理空间数据以及对亚太地区（首尔）、亚太地区（孟买）和欧洲（伦敦）区域的支持。有关信息，请参阅查询地理空间数据及 AWS 区域 和端点。	2017 年 11 月 1 日
增加了对欧洲（法兰克福）的支持。	增加了对欧洲（法兰克福）的支持。有关受支持的区域列表，请参阅 AWS 区域 和端点。	2017 年 10 月 19 日
增加对使用 AWS CloudFormation 进行命名 Athena 查询的支持。	增加对使用 AWS CloudFormation 创建命名 Athena 查询的支持。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的 AWS::Athena::NamedQuery。	2017 年 10 月 3 日

更改	描述	发行日期
添加了对亚太地区（悉尼）的支持。	添加了对亚太地区（悉尼）的支持。有关受支持的区域列表，请参阅 AWS 区域 和端点。	2017 年 9 月 25 日
在本指南中添加了一节，用于查询 AWS 服务 日志和不同类型的数据，包括映射、数组、嵌套数据和包含 JSON 的数据。	增加 查询 AWS 服务 日志 的示例以及在 Athena 中查询不同类型数据的示例。有关信息，请参阅使用 Amazon Athena 运行 SQL 查询。	2017 年 9 月 5 日
增加了对 AWS Glue Data Catalog 的支持。	增加与 AWS Glue Data Catalog 的集成以及用于从 Athena 托管数据目录更新到 AWS Glue Data Catalog 的迁移向导。有关更多信息，请参阅与 AWS Glue 集成以及 AWS Glue。	2017 年 8 月 14 日
增加了对 Grok SerDe 的支持。	增加了对 Grok SerDe 的支持，它为非结构化文本文件 (例如日志) 中的记录添加了更简便的模式匹配。有关更多信息，请参阅 Grok SerDe。添加了键盘快捷键以使用控制台滚动查询历史记录。	2017 年 8 月 4 日
添加了对亚太地区（东京）的支持	增加了对亚太地区（东京）和亚太地区（新加坡）区域的支持。有关受支持的区域列表，请参阅 AWS 区域 和端点。	2017 年 6 月 22 日
增加了对欧洲（爱尔兰）的支持。	增加了对欧洲（爱尔兰）的支持。有关更多信息，请参阅 AWS 区域 和端点。	2017 年 6 月 8 日
添加了 Amazon Athena API 和 AWS CLI 支持。	为 Athena 添加了 Amazon Athena API 和 AWS CLI 支持。将 JDBC 驱动程序更新到了版本 1.1.0。	2017 年 5 月 19 日
增加了对 Amazon S3 数据加密的支持。	增加了对 Amazon S3 数据加密的支持，并发布了具有加密支持、改进和错误修复的 JDBC 驱动程序更新 (1.0.1 版)。有关更多信息，请参阅 静态加密。	2017 年 4 月 4 日

更改	描述	发行日期
增加 AWS CloudTrail SerDe。	增加 AWS CloudTrail SerDe、改进性能和解决分区问题。 • AWS CloudTrail SerDe 已由 Hive JSON SerDe 取代，以读取 CloudTrail 日志。有关查询 CloudTrail 日志的信息，请参阅 查询 AWS CloudTrail 日志。 • 提高了扫描大量分区时的性能。 • 提高了 `MSCK Repair Table` 操作的性能。 • 增加了查询在主要区域之外的区域存储的 Amazon S3 数据的功能。除了标准 Athena 费用外，Amazon S3 的标准区域间数据传输费率也适用。	2017 年 3 月 24 日
增加了对美国东部（俄亥俄）的支持。	增加了对 Avro SerDe 和 用于处理 CSV 的 OpenCSVSerDe、美国东部（俄亥俄）以及在控制台向导中批量编辑列的支持。改进了大型 Parquet 表的性能。	2017 年 2 月 20 日
	《Amazon Athena 用户指南》的初始版本。	2016 年 11 月

AWS 术语表

有关最新的 AWS 术语，请参阅 AWS 词汇表参考 中的 AWS 词汇表。